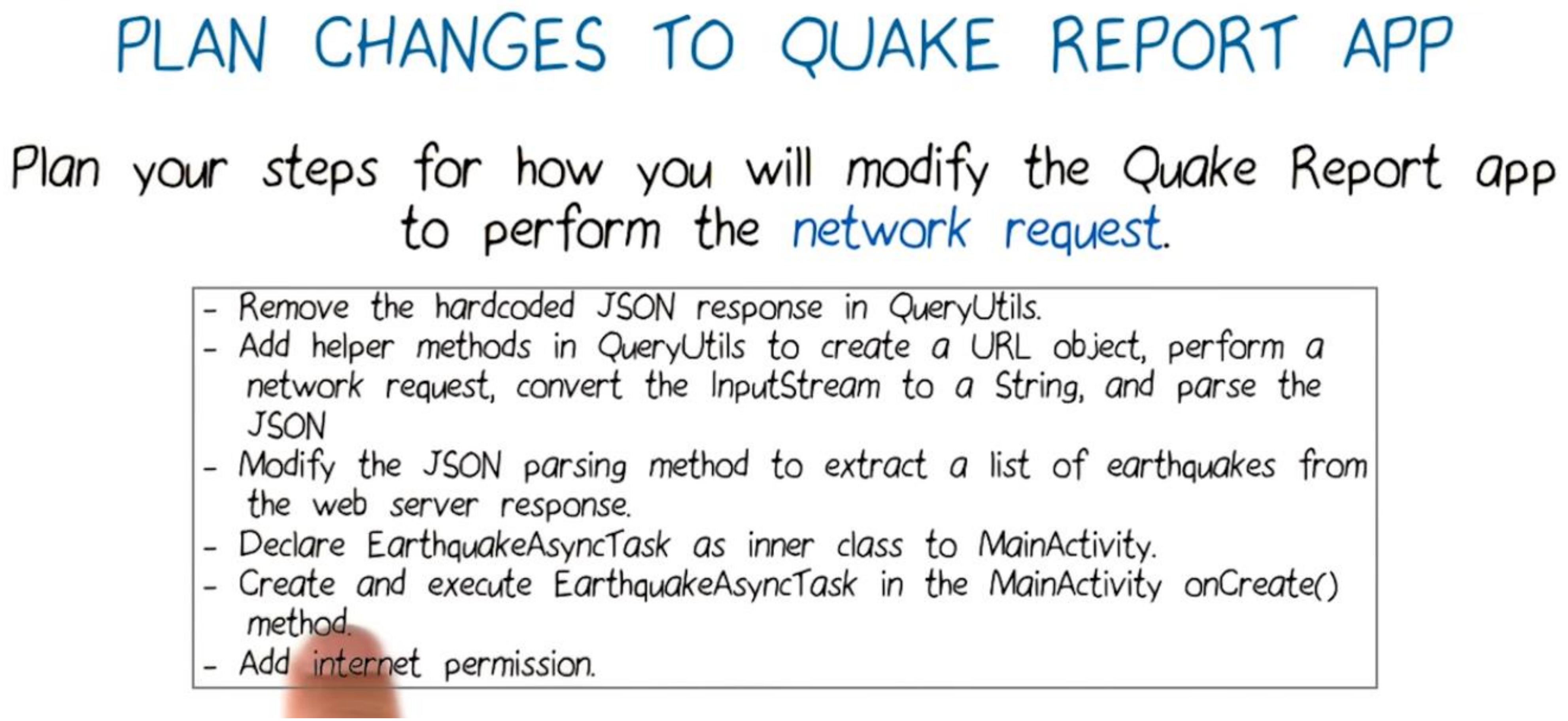
QuakeReport part 2

PLAN



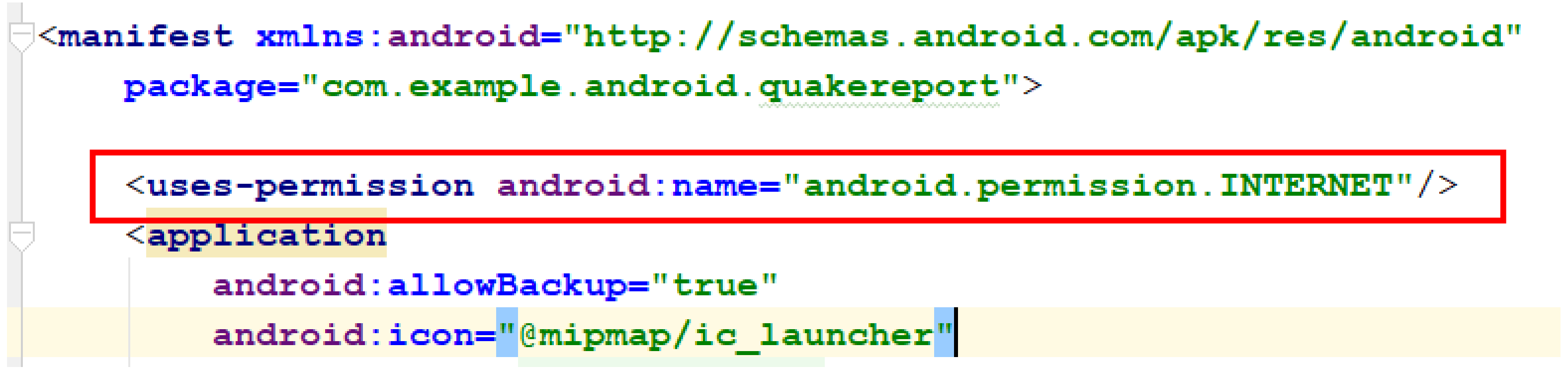
PLAN CHANGES TO QUAKE REPORT APP

Plan your steps for how you will modify the Quake Report app to perform the network request.

- Remove the hardcoded JSON response in QueryUtils.
- Add helper methods in QueryUtils to create a URL object, perform a network request, convert the InputStream to a String, and parse the JSON
- Modify the JSON parsing method to extract a list of earthquakes from the web server response.
- Declare EarthquakeAsyncTask as inner class to MainActivity.
- Create and execute EarthquakeAsyncTask in the MainActivity onCreate() method.
- Add internet permission.

09 - Fetch earthquake data via network request

- First declare the internet permission in the AndroidManifest.xml file so that the app can access the network.
- In AndroidManifest.xml

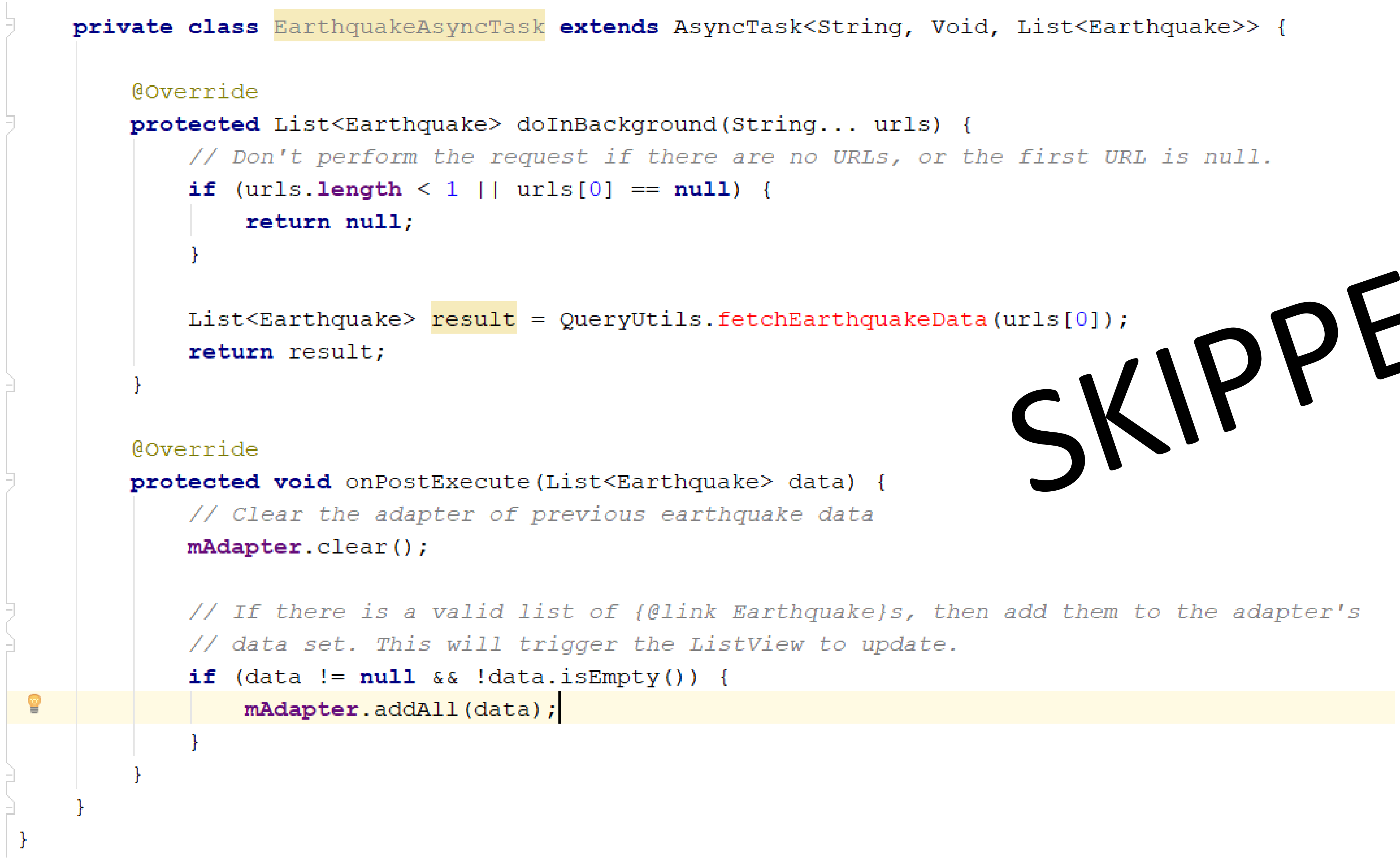


```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.quakereport">

    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
```

EarthquakeActivity.java

```
public class EarthquakeActivity extends AppCompatActivity
                                                     {

    private static final String LOG_TAG = EarthquakeActivity.class.getName();

    /** URL for earthquake data from the USGS dataset */
    private static final String USGS_REQUEST_URL =
            "https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&orderby=time&minmag=6&limit=10";

    private EarthquakeAdapter mAdapter;
```

EarthquakActivity.java – Create AsyncTask

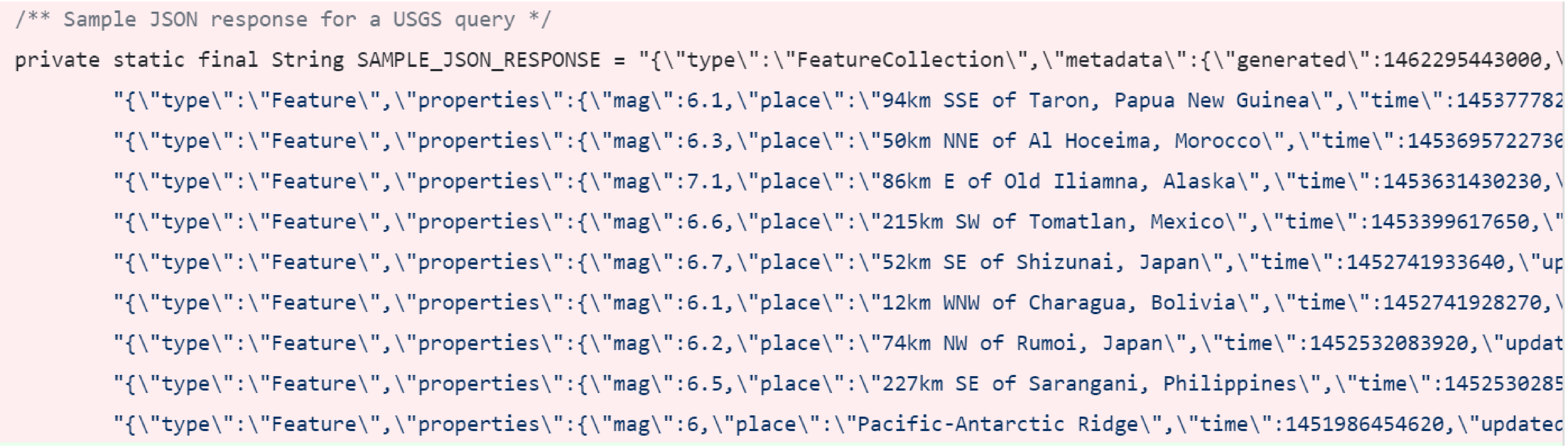


```
    private class EarthquakeAsyncTask extends AsyncTask<String, Void, List<Earthquake>> {

        @Override
        protected List<Earthquake> doInBackground(String... urls) {
            // Don't perform the request if there are no URLs, or the first URL is null.
            if (urls.length < 1 || urls[0] == null) {
                return null;
            }

            List<Earthquake> result = QueryUtils.fetchEarthquakeData(urls[0]);
            return result;
        }

        @Override
        protected void onPostExecute(List<Earthquake> data) {
            // Clear the adapter of previous earthquake data
            mAdapter.clear();

            // If there is a valid list of {@link Earthquake}s, then add them to the adapter's
            // data set. This will trigger the ListView to update.
            if (data != null && !data.isEmpty()) {
                mAdapter.addAll(data);
            }
        }
    }
}
```

SKIPPED

Modify QueryUtils.java

delete the SAMPLE_JSON_RESPONSE constant in the QueryUtils class

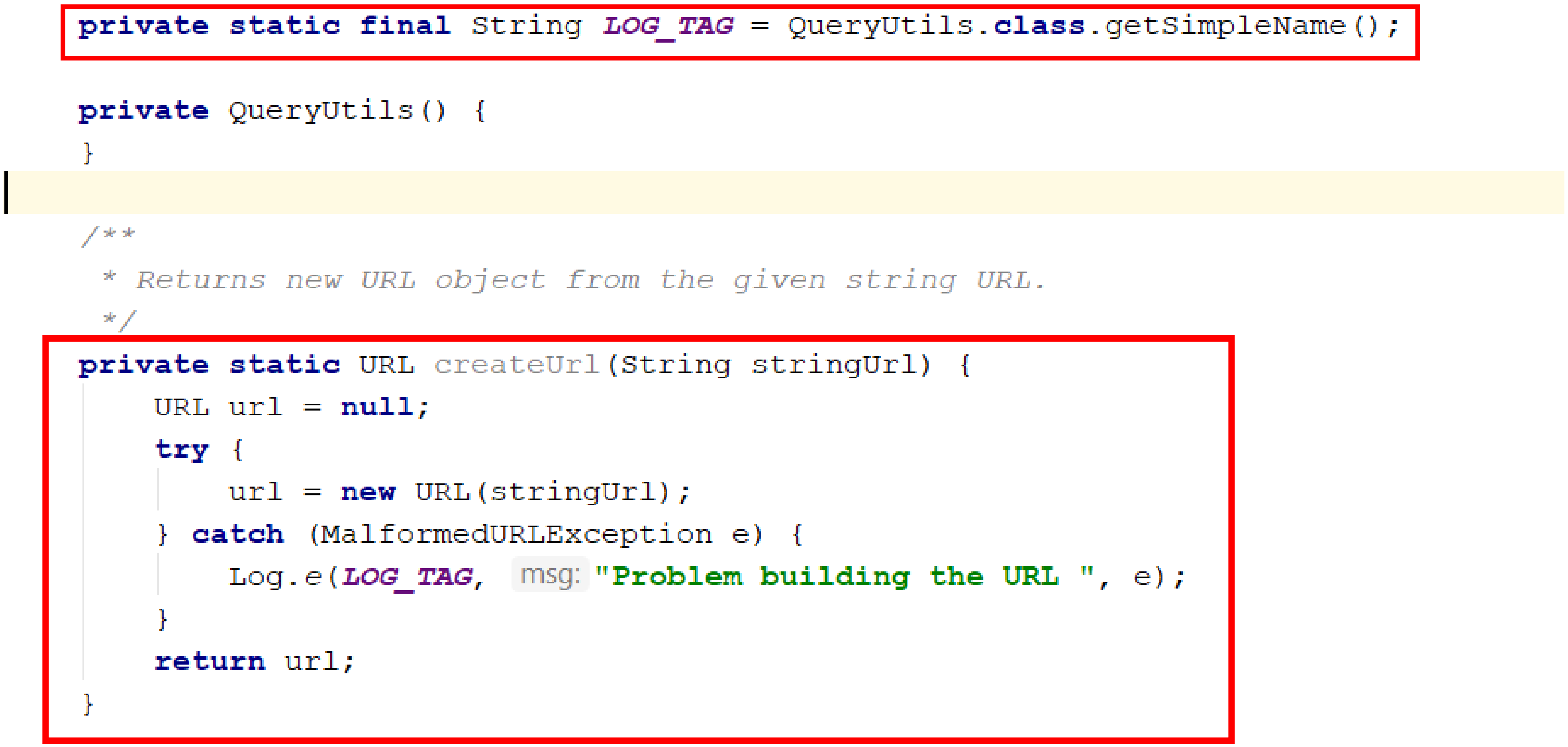


```
/** Sample JSON response for a USGS query */
private static final String SAMPLE_JSON_RESPONSE = "{\"type\":\"FeatureCollection\",\"metadata\":{\"generated\":1462295443000,\
        "{\"type\":\"Feature\",\"properties\":{\"mag\":6.1,\"place\":\"94km SSE of Taron, Papua New Guinea\",\"time\":145377782
        "{\"type\":\"Feature\",\"properties\":{\"mag\":6.3,\"place\":\"50km NNE of Al Hoceima, Morocco\",\"time\":145369572273
        "{\"type\":\"Feature\",\"properties\":{\"mag\":7.1,\"place\":\"86km E of Old Iliamna, Alaska\",\"time\":1453631430230,\
        "{\"type\":\"Feature\",\"properties\":{\"mag\":6.6,\"place\":\"215km SW of Tomatlan, Mexico\",\"time\":1453399617650,\"
        "{\"type\":\"Feature\",\"properties\":{\"mag\":6.7,\"place\":\"52km SE of Shizunai, Japan\",\"time\":1452741933640,\"up
        "{\"type\":\"Feature\",\"properties\":{\"mag\":6.1,\"place\":\"12km WNW of Charagua, Bolivia\",\"time\":1452741928270,\
        "{\"type\":\"Feature\",\"properties\":{\"mag\":6.2,\"place\":\"74km NW of Rumoi, Japan\",\"time\":1452532083920,\"updat
        "{\"type\":\"Feature\",\"properties\":{\"mag\":6.5,\"place\":\"227km SE of Sarangani, Philippines\",\"time\":145253028
        "{\"type\":\"Feature\",\"properties\":{\"mag\":6,\"place\":\"Pacific-Antarctic Ridge\",\"time\":1451986454620,\"update
```

In QueryUtils.java – Create these helper methods

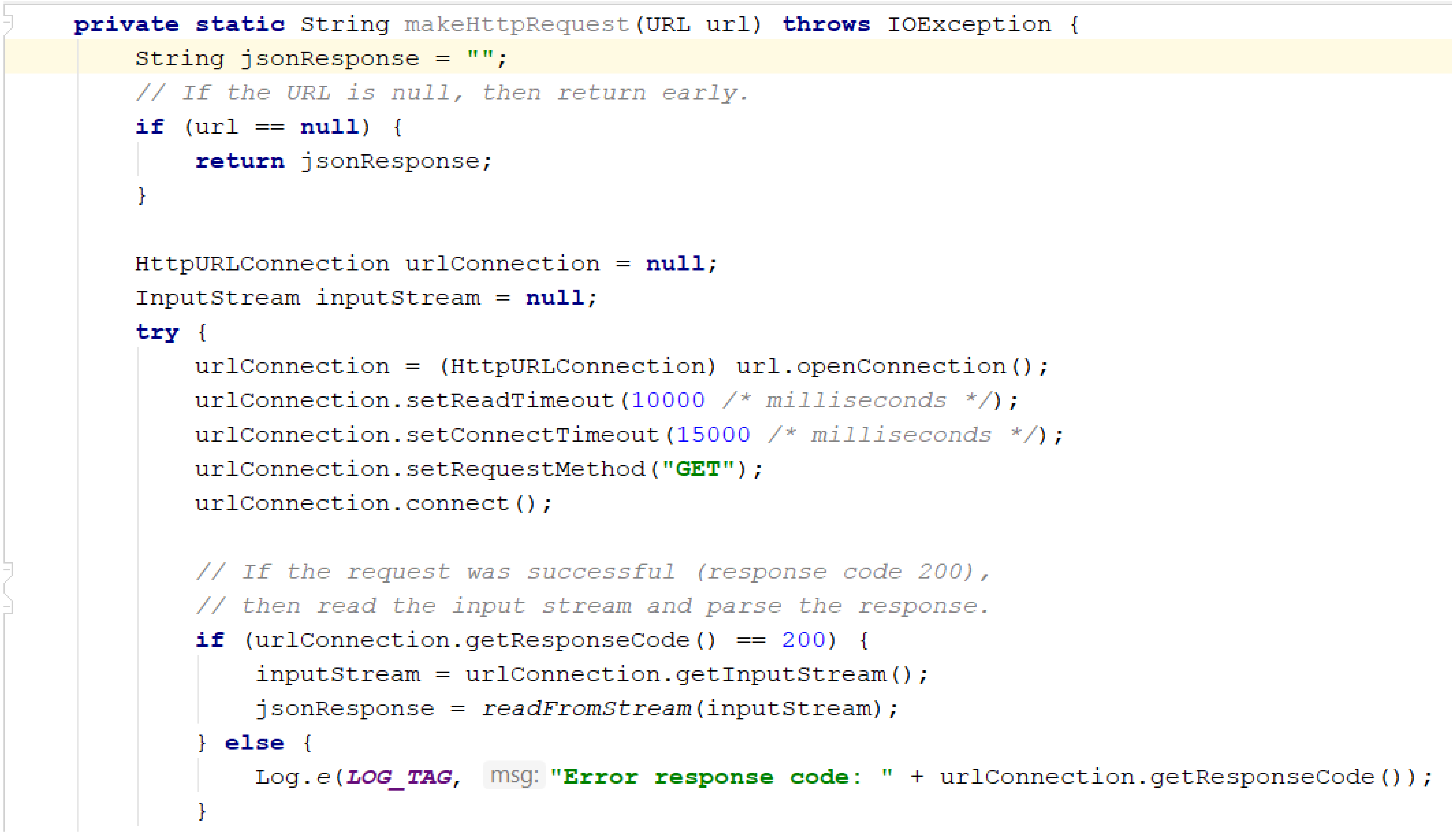
- createUrl()
- makeHttpRequest()
- readFromStream
- fetchEartquakeData()

QueryUtils.java -- createUrl()

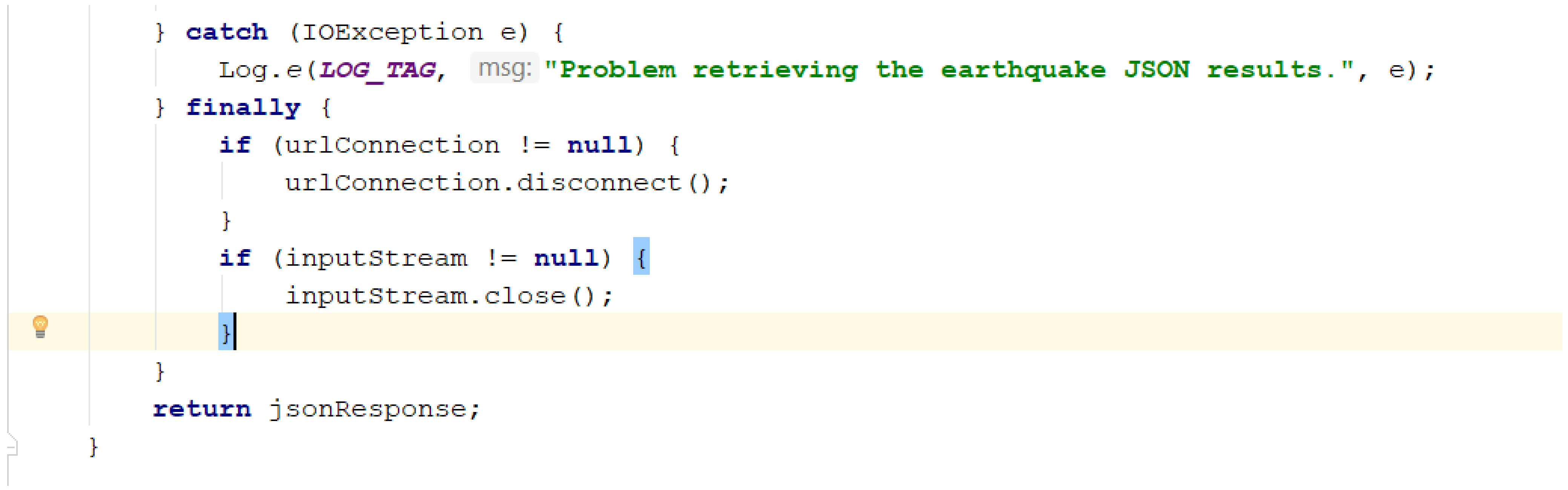


```java
private static final String LOG_TAG = QueryUtils.class.getSimpleName();

private QueryUtils() {
}

/**
 * Returns new URL object from the given string URL.
 */
private static URL createUrl(String stringUrl) {
    URL url = null;
    try {
        url = new URL(stringUrl);
    } catch (MalformedURLException e) {
        Log.e(LOG_TAG, msg: "Problem building the URL ", e);
    }
    return url;
}
```

QueryUtils.java – makeHttpRequest()

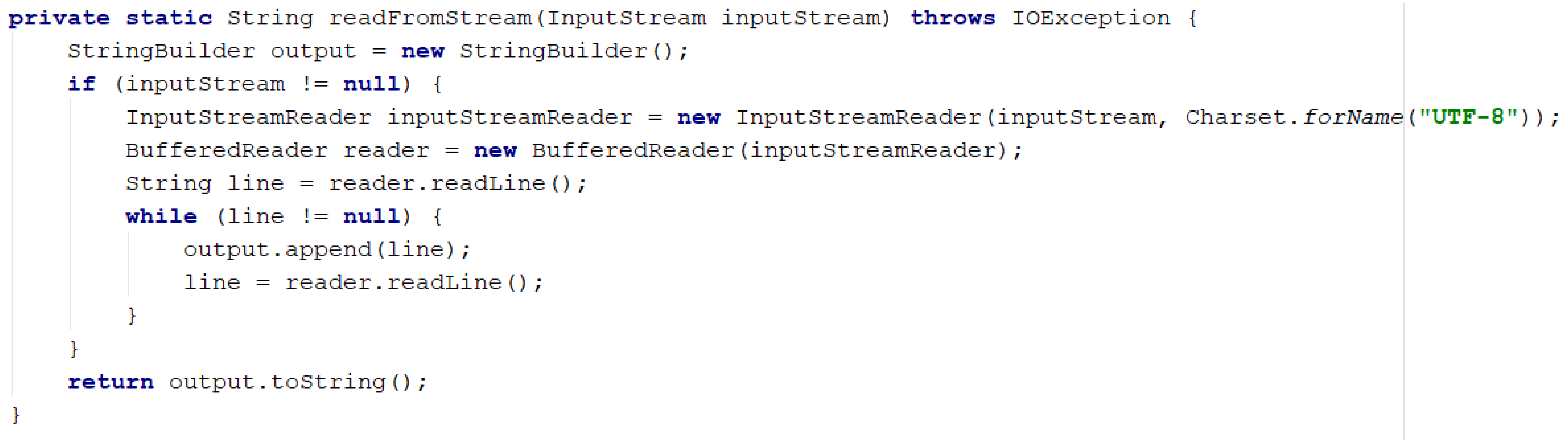


```java
    private static String makeHttpRequest(URL url) throws IOException {
        String jsonResponse = "";
        // If the URL is null, then return early.
        if (url == null) {
            return jsonResponse;
        }

        HttpURLConnection urlConnection = null;
        InputStream inputStream = null;
        try {
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setReadTimeout(10000 /* milliseconds */);
            urlConnection.setConnectTimeout(15000 /* milliseconds */);
            urlConnection.setRequestMethod("GET");
            urlConnection.connect();

            // If the request was successful (response code 200),
            // then read the input stream and parse the response.
            if (urlConnection.getResponseCode() == 200) {
                inputStream = urlConnection.getInputStream();
                jsonResponse = readFromStream(inputStream);
            } else {
                Log.e(LOG_TAG,  msg: "Error response code: " + urlConnection.getResponseCode());
            }
```



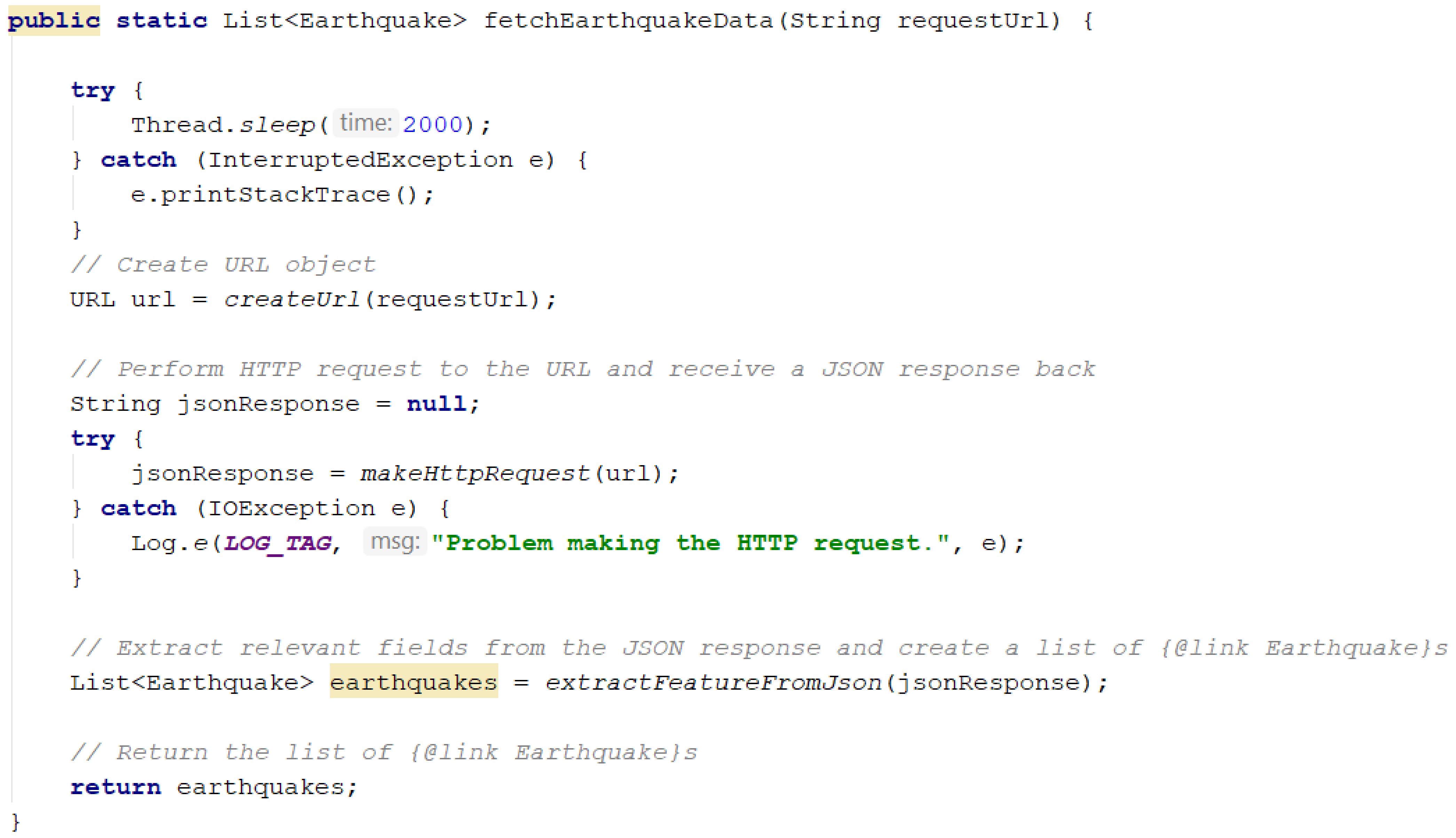
```
        } catch (IOException e) {
            Log.e(LOG_TAG, msg: "Problem retrieving the earthquake JSON results.", e);
        } finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
            if (inputStream != null) {
                inputStream.close();
            }
        }
        return jsonResponse;
    }
```

QueryUtils.java –readFromStream()



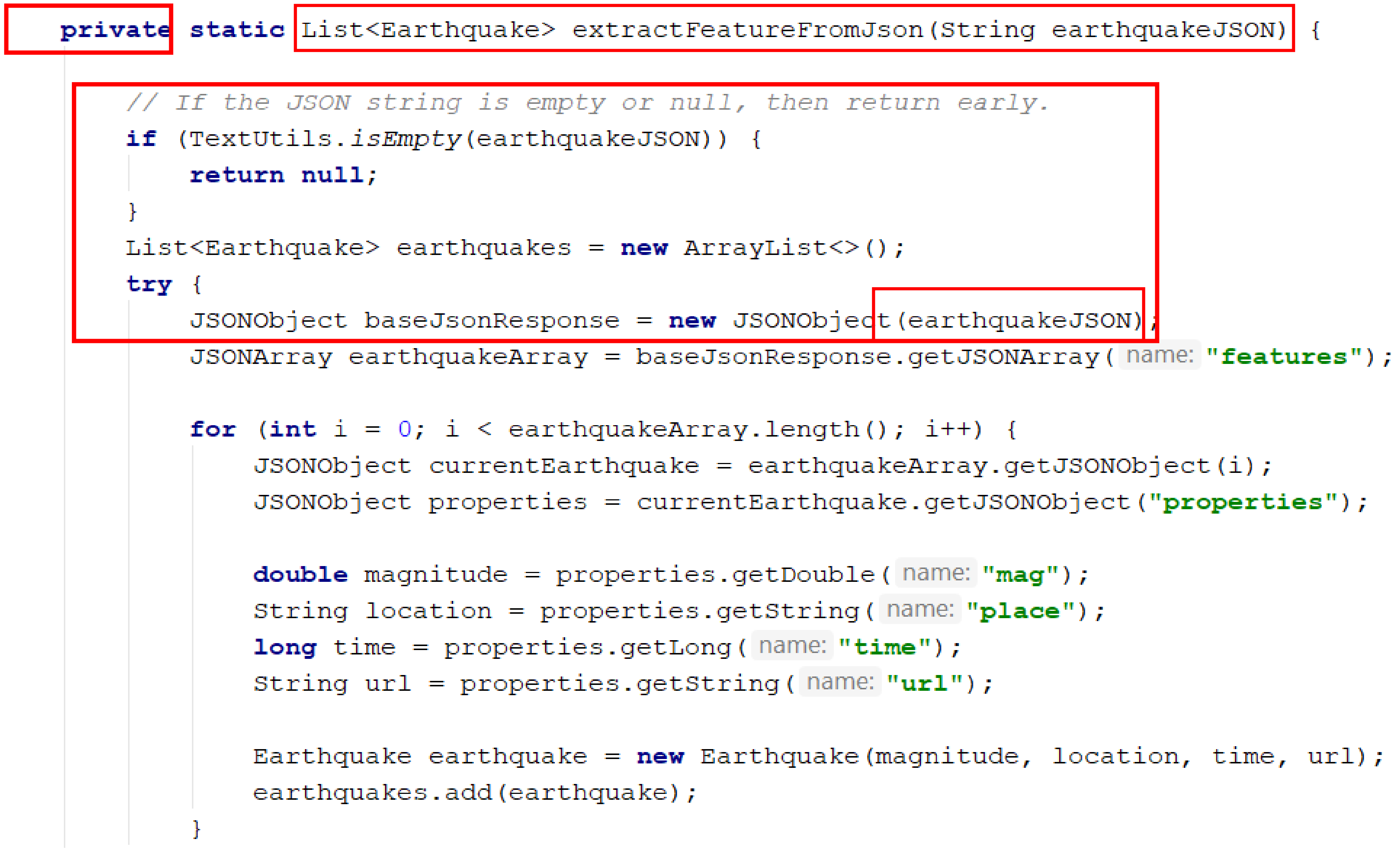
```java
private static String readFromStream(InputStream inputStream) throws IOException {
    StringBuilder output = new StringBuilder();
    if (inputStream != null) {
        InputStreamReader inputStreamReader = new InputStreamReader(inputStream, Charset.forName("UTF-8"));
        BufferedReader reader = new BufferedReader(inputStreamReader);
        String line = reader.readLine();
        while (line != null) {
            output.append(line);
            line = reader.readLine();
        }
    }
    return output.toString();
}
```

QueryUtils.java –fetchEarthquakeData()



```java
public static List<Earthquake> fetchEarthquakeData(String requestUrl) {

    try {
        Thread.sleep( time: 2000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    // Create URL object
    URL url = createUrl(requestUrl);

    // Perform HTTP request to the URL and receive a JSON response back
    String jsonResponse = null;
    try {
        jsonResponse = makeHttpRequest(url);
    } catch (IOException e) {
        Log.e(LOG_TAG,  msg: "Problem making the HTTP request.", e);
    }

    // Extract relevant fields from the JSON response and create a list of {@link Earthquake}s
    List<Earthquake> earthquakes = extractFeatureFromJson(jsonResponse);

    // Return the list of {@link Earthquake}s
    return earthquakes;
}
```

Modify extractEarthquakes() and **rename** it to extractFeatureFromJson()



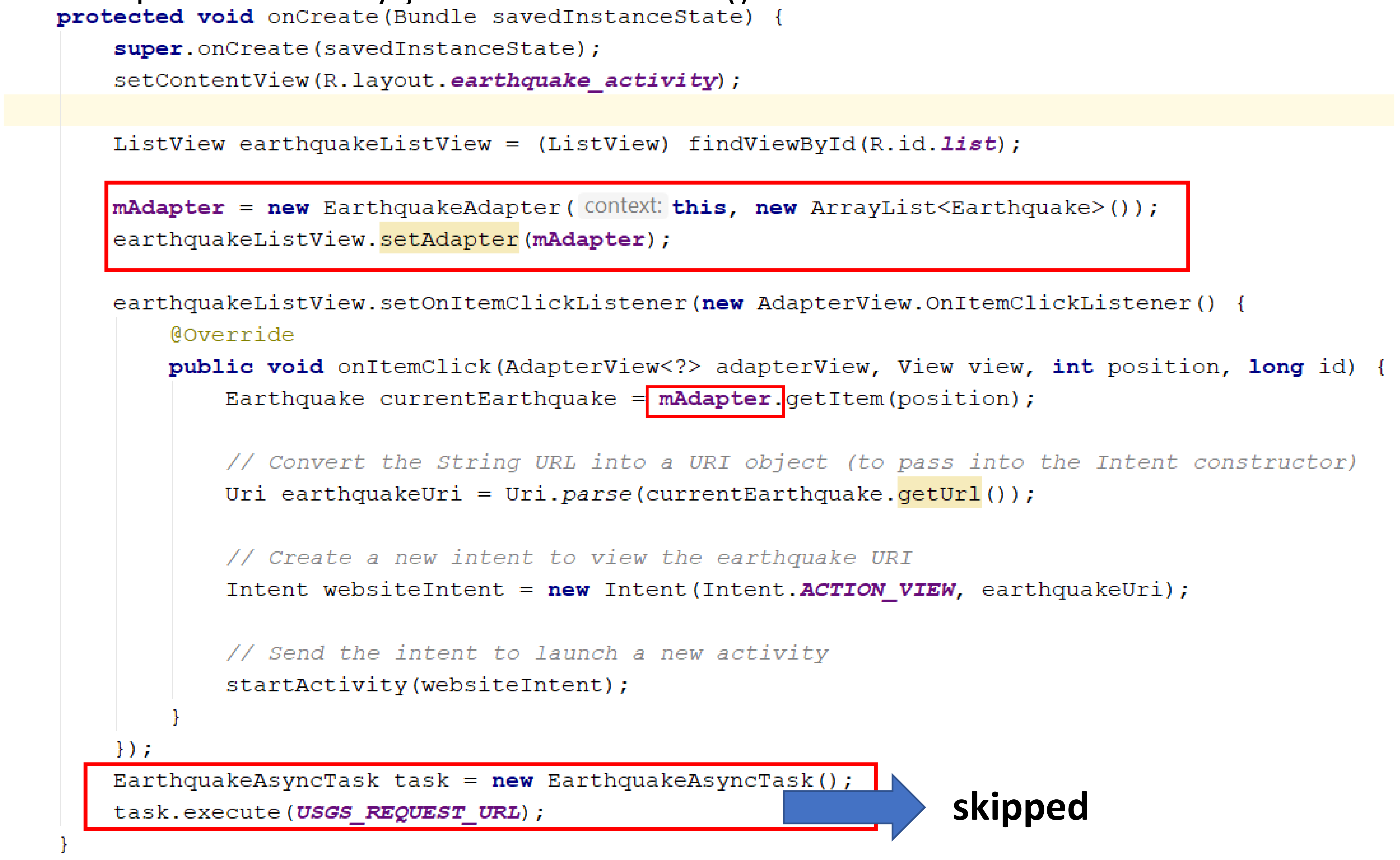
```
    private static List<Earthquake> extractFeatureFromJson(String earthquakeJSON) {

        // If the JSON string is empty or null, then return early.
        if (TextUtils.isEmpty(earthquakeJSON)) {
            return null;
        }
        List<Earthquake> earthquakes = new ArrayList<>();
        try {
            JSONObject baseJsonResponse = new JSONObject(earthquakeJSON);
            JSONArray earthquakeArray = baseJsonResponse.getJSONArray( name: "features");

            for (int i = 0; i < earthquakeArray.length(); i++) {
                JSONObject currentEarthquake = earthquakeArray.getJSONObject(i);
                JSONObject properties = currentEarthquake.getJSONObject("properties");

                double magnitude = properties.getDouble( name: "mag");
                String location = properties.getString( name: "place");
                long time = properties.getLong( name: "time");
                String url = properties.getString( name: "url");

                Earthquake earthquake = new Earthquake(magnitude, location, time, url);
                earthquakes.add(earthquake);
            }
```

EarthquakeActivity.java – onCreate()

- Remove this code

```
// Create a fake list of earthquake locations.
ArrayList<Earthquake> earthquakes = QueryUtils.extractEarthquakes();
```

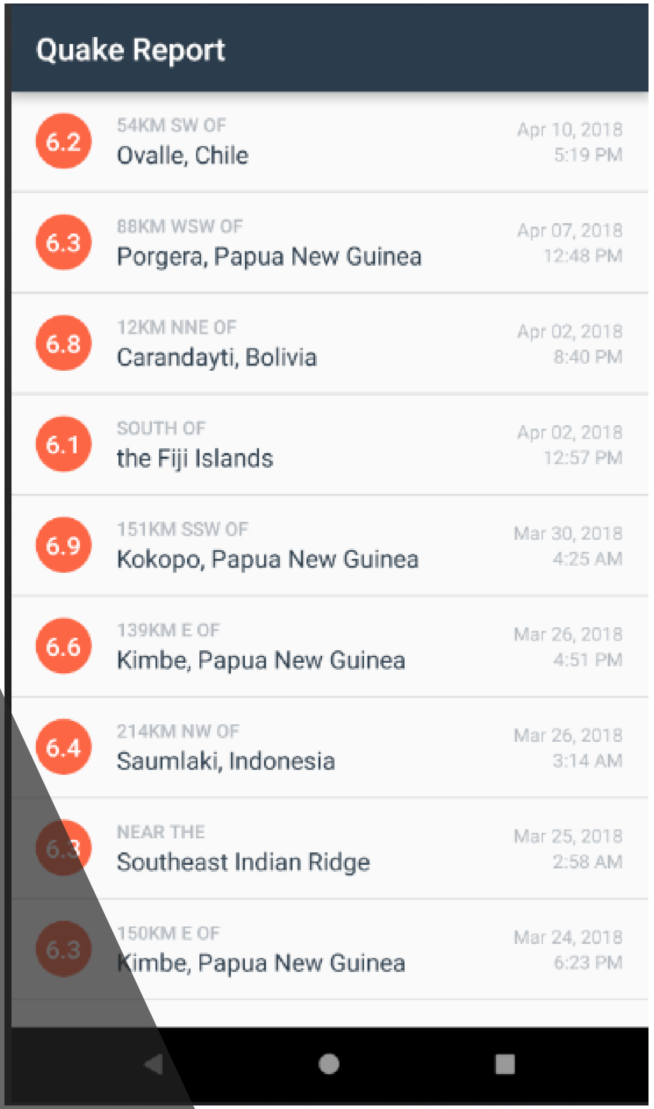
EarthquakeActivity.java – onCreate()



```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.earthquake_activity);

    ListView earthquakeListView = (ListView) findViewById(R.id.list);

    mAdapter = new EarthquakeAdapter( context: this, new ArrayList<Earthquake>());
    earthquakeListView.setAdapter(mAdapter);

    earthquakeListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {
            Earthquake currentEarthquake = mAdapter.getItem(position);

            // Convert the String URL into a URI object (to pass into the Intent constructor)
            Uri earthquakeUri = Uri.parse(currentEarthquake.getUrl());

            // Create a new intent to view the earthquake URI
            Intent websiteIntent = new Intent(Intent.ACTION_VIEW, earthquakeUri);

            // Send the intent to launch a new activity
            startActivity(websiteIntent);
        }
    });
    EarthquakeAsyncTask task = new EarthquakeAsyncTask();
    task.execute(USGS_REQUEST_URL);
}
```

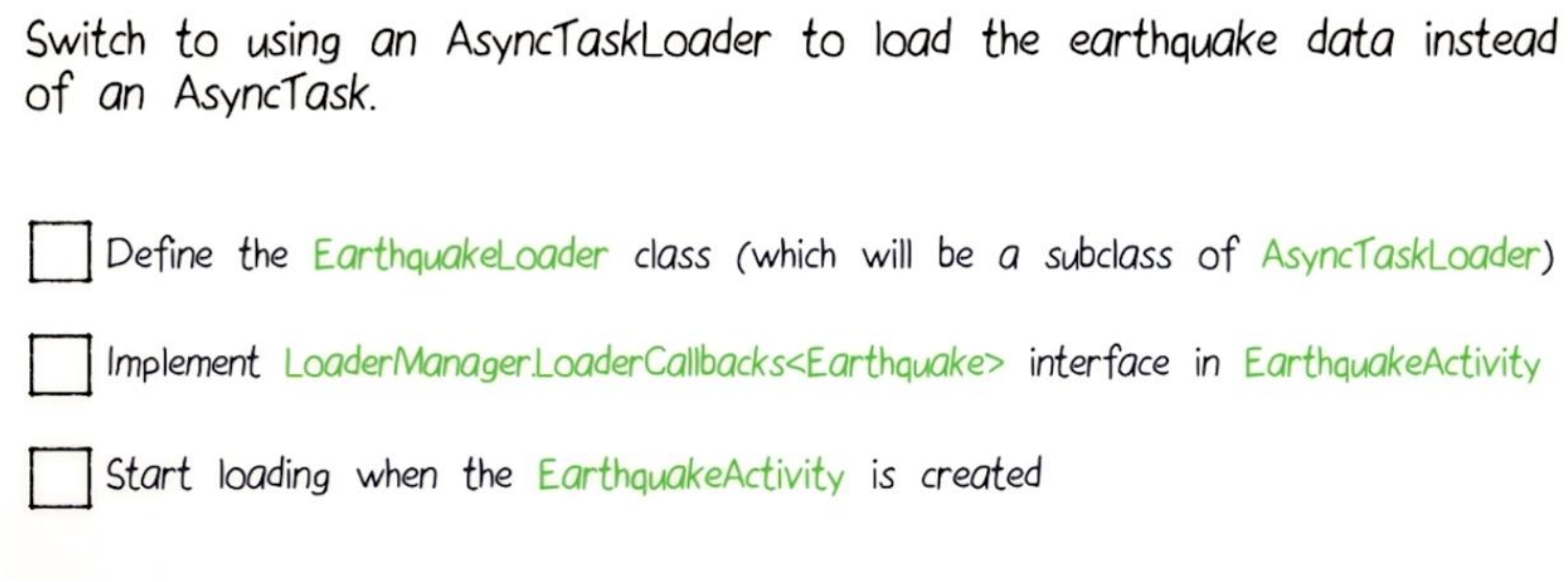
skipped

Run APP


Quake Report

6.2	54KM SW OF	Ovalle, Chile	Apr 10, 2018 5:19 PM
6.3	88KM WSW OF	Porgera, Papua New Guinea	Apr 07, 2018 12:48 PM
6.8	12KM NNE OF	Carandayti, Bolivia	Apr 02, 2018 8:40 PM
6.1	SOUTH OF	the Fiji Islands	Apr 02, 2018 12:57 PM
6.9	151KM SSW OF	Kokopo, Papua New Guinea	Mar 30, 2018 4:25 AM
6.6	139KM E OF	Kimbe, Papua New Guinea	Mar 26, 2018 4:51 PM
6.4	214KM NW OF	Saumlaki, Indonesia	Mar 26, 2018 3:14 AM
6.3	NEAR THE	Southeast Indian Ridge	Mar 25, 2018 2:58 AM
6.3	150KM E OF	Kimbe, Papua New Guinea	Mar 24, 2018 6:23 PM

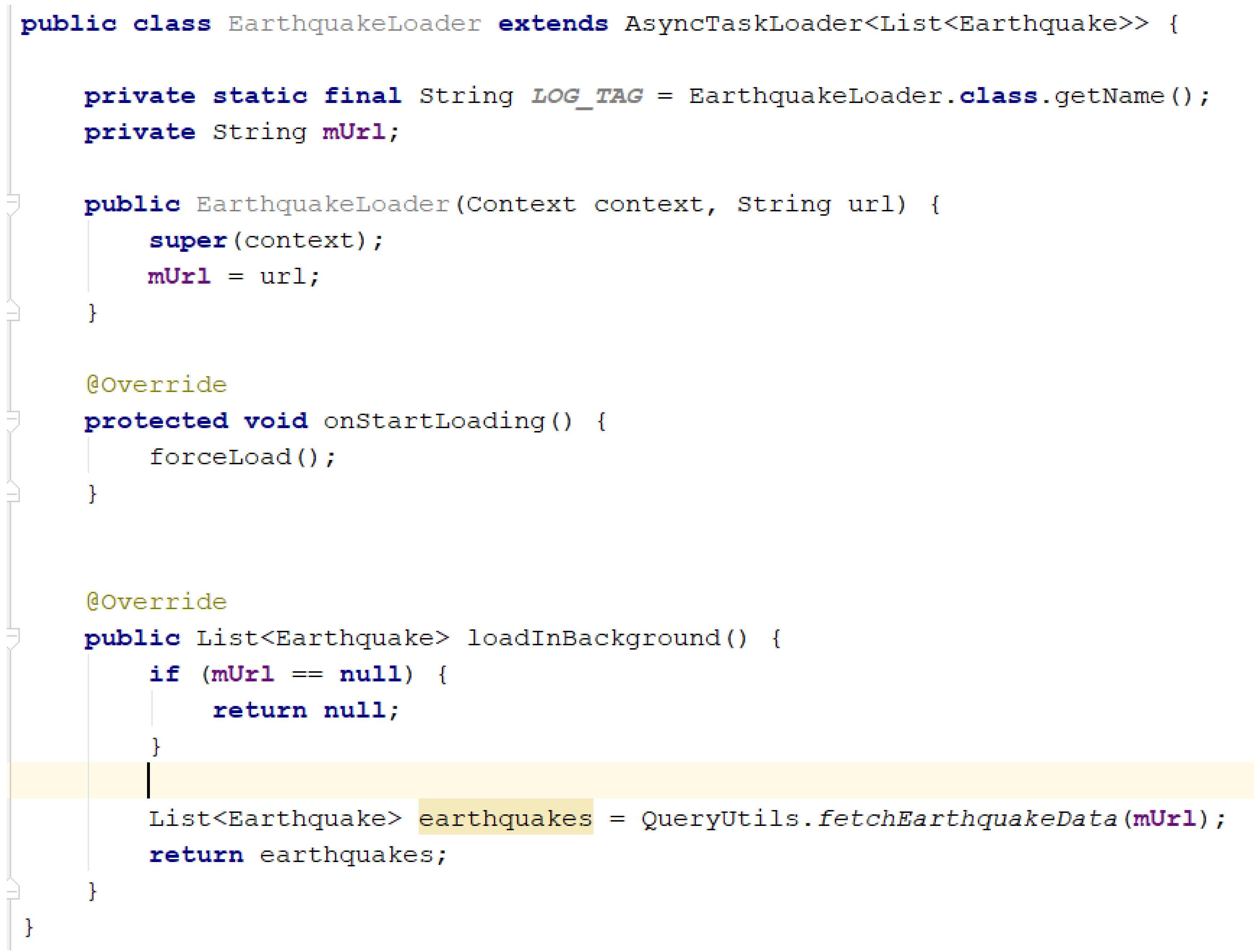
10 - Switch from AsyncTask to loader



Switch to using an AsyncTaskLoader to load the earthquake data instead of an AsyncTask.

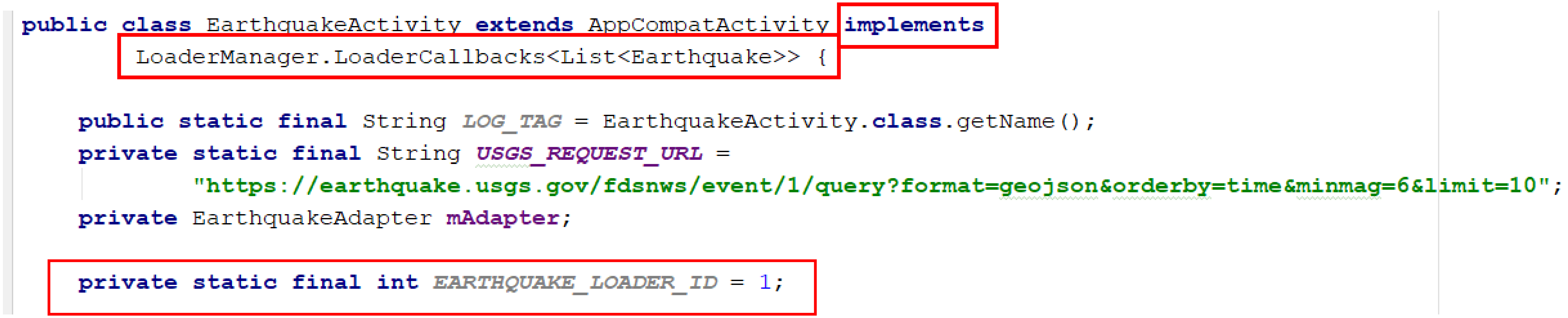
- ☐ Define the EarthquakeLoader class (which will be a subclass of AsyncTaskLoader)
- ☐ Implement LoaderManager.LoaderCallbacks<Earthquake> interface in EarthquakeActivity
- ☐ Start loading when the EarthquakeActivity is created

Create EarthquakeLoader.java



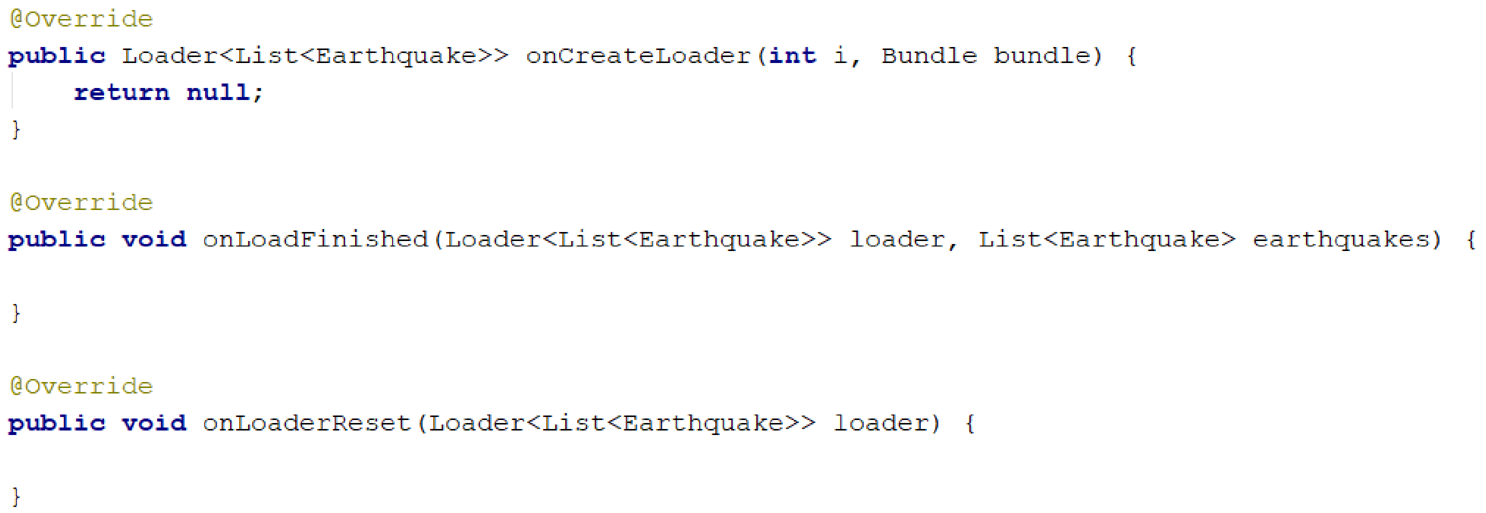
```
public class EarthquakeLoader extends AsyncTaskLoader<List<Earthquake>> {

    private static final String LOG_TAG = EarthquakeLoader.class.getName();
    private String mUrl;

    public EarthquakeLoader(Context context, String url) {
        super(context);
        mUrl = url;
    }

    @Override
    protected void onStartLoading() {
        forceLoad();
    }


    @Override
    public List<Earthquake> loadInBackground() {
        if (mUrl == null) {
            return null;
        }

        List<Earthquake> earthquakes = QueryUtils.fetchEarthquakeData(mUrl);
        return earthquakes;
    }
}
```

EarthquakeActivity.java



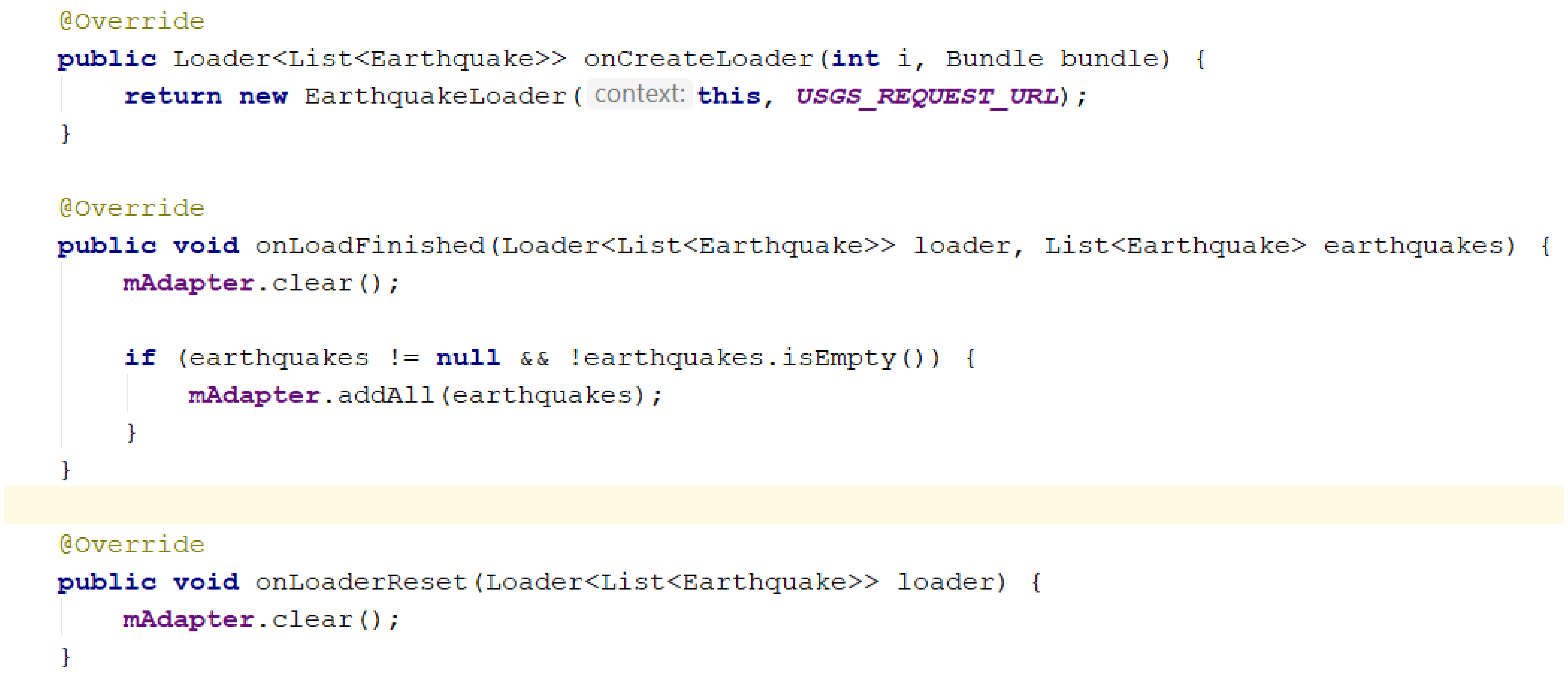
```java
public class EarthquakeActivity extends AppCompatActivity implements
        LoaderManager.LoaderCallbacks<List<Earthquake>> {

    public static final String LOG_TAG = EarthquakeActivity.class.getName();
    private static final String USGS_REQUEST_URL =
            "https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&orderby=time&minmag=6&limit=10";
    private EarthquakeAdapter mAdapter;

    private static final int EARTHQUAKE_LOADER_ID = 1;
```

- Implement method:
  - onCreateLoader()
  - onLoadFinished()
  - onLoaderReset()



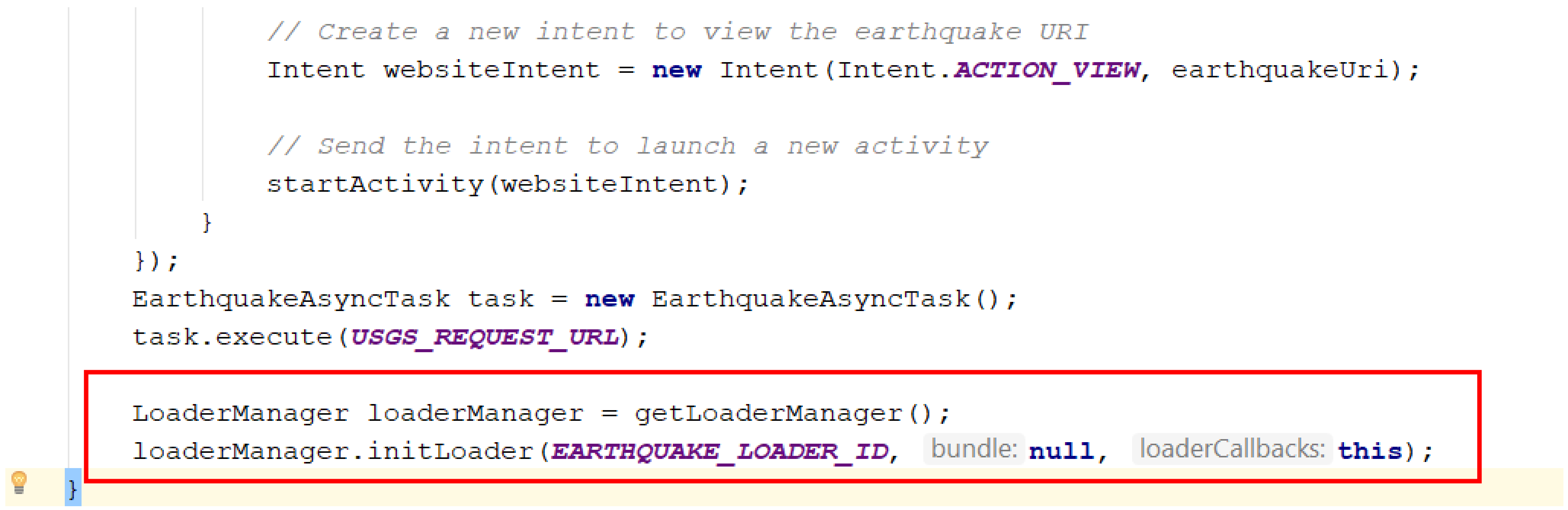
```java
@Override
public Loader<List<Earthquake>> onCreateLoader(int i, Bundle bundle) {
    return null;
}

@Override
public void onLoadFinished(Loader<List<Earthquake>> loader, List<Earthquake> earthquakes) {

}

@Override
public void onLoaderReset(Loader<List<Earthquake>> loader) {

}
```

EarthquakeActivity.java – onCreateLoader(), onLoadFinished(), and onLoaderReset()



```
@Override
public Loader<List<Earthquake>> onCreateLoader(int i, Bundle bundle) {
    return new EarthquakeLoader( context: this, USGS_REQUEST_URL);
}

@Override
public void onLoadFinished(Loader<List<Earthquake>> loader, List<Earthquake> earthquakes) {
    mAdapter.clear();

    if (earthquakes != null && !earthquakes.isEmpty()) {
        mAdapter.addAll(earthquakes);
    }
}

@Override
public void onLoaderReset(Loader<List<Earthquake>> loader) {
    mAdapter.clear();
}
```

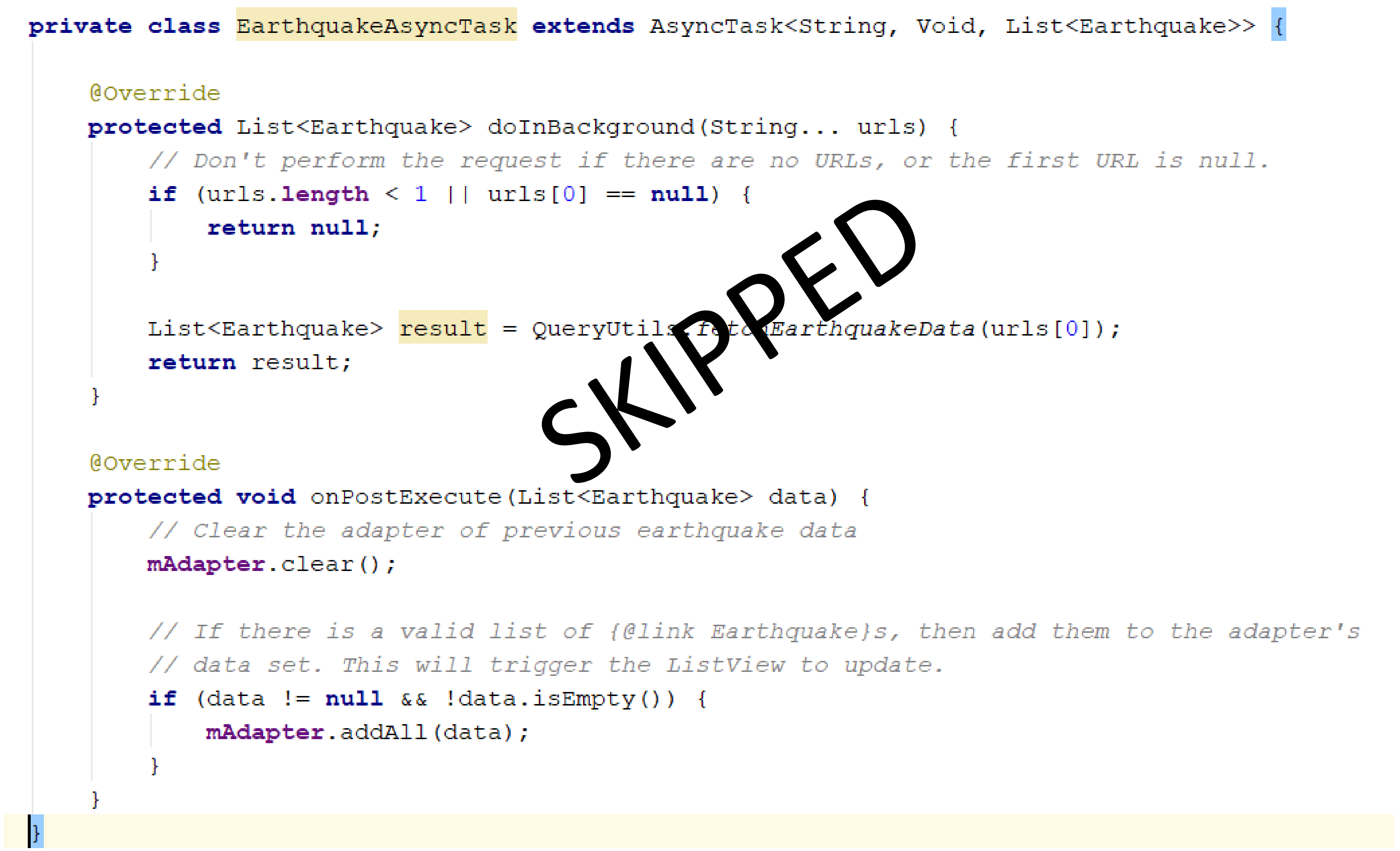
EarthquakeActivity – onCreate()



```
                // Create a new intent to view the earthquake URI
                Intent websiteIntent = new Intent(Intent.ACTION_VIEW, earthquakeUri);

                // Send the intent to launch a new activity
                startActivity(websiteIntent);
            }
        });
        EarthquakeAsyncTask task = new EarthquakeAsyncTask();
        task.execute(USGS_REQUEST_URL);

        LoaderManager loaderManager = getLoaderManager();
        loaderManager.initLoader(EARTHQUAKE_LOADER_ID,  bundle: null,  loaderCallbacks: this);
    }
```

EarthquakeActivity

- Delete EarthquakeAsyncTask

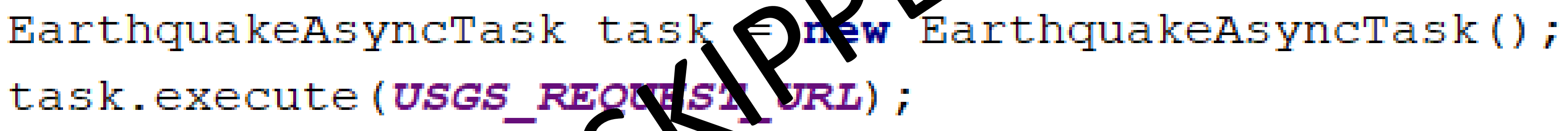


```java
private class EarthquakeAsyncTask extends AsyncTask<String, Void, List<Earthquake>> {

    @Override
    protected List<Earthquake> doInBackground(String... urls) {
        // Don't perform the request if there are no URLs, or the first URL is null.
        if (urls.length < 1 || urls[0] == null) {
            return null;
        }

        List<Earthquake> result = QueryUtils.fetchEarthquakeData(urls[0]);
        return result;
    }

    @Override
    protected void onPostExecute(List<Earthquake> data) {
        // Clear the adapter of previous earthquake data
        mAdapter.clear();

        // If there is a valid list of {@link Earthquake}s, then add them to the adapter's
        // data set. This will trigger the ListView to update.
        if (data != null && !data.isEmpty()) {
            mAdapter.addAll(data);
        }
    }
}
```

SKIPPED

EartquakeActivity.java -- onCreate

- Also delete this



```
EarthquakeAsyncTask task = new EarthquakeAsyncTask();
task.execute(USGS_REQUEST_URL);
```

SKIPPED

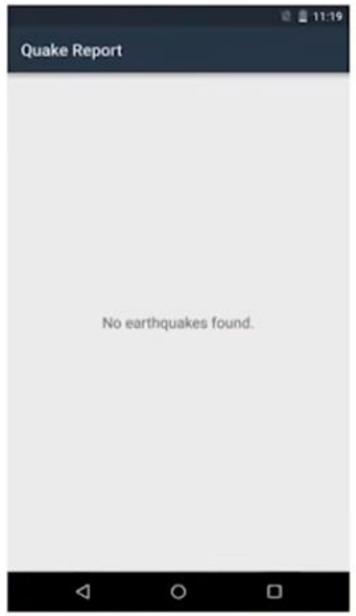
Run APP

- When you run the app on your device, there should be no errors, and you should see the same list of earthquakes as before - except your code underneath is a lot more robust!

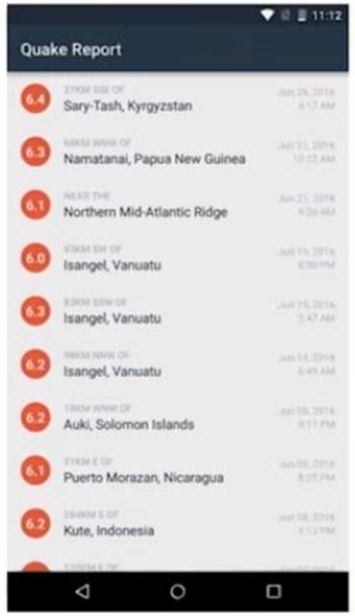
11 - Add empty state to ListView

EMPTY STATE FOR QUAKE REPORT APP

- ☐ Modify the earthquake_activity.xml layout to add the empty state TextView (as a sibling to the ListView).
- ☐ In the EarthquakeActivity.java file, use the ListView setEmptyView() method to set the empty state TextView onto the ListView.
- ☐ Set the text in the empty state TextView after the first load (in onLoadFinished() method)



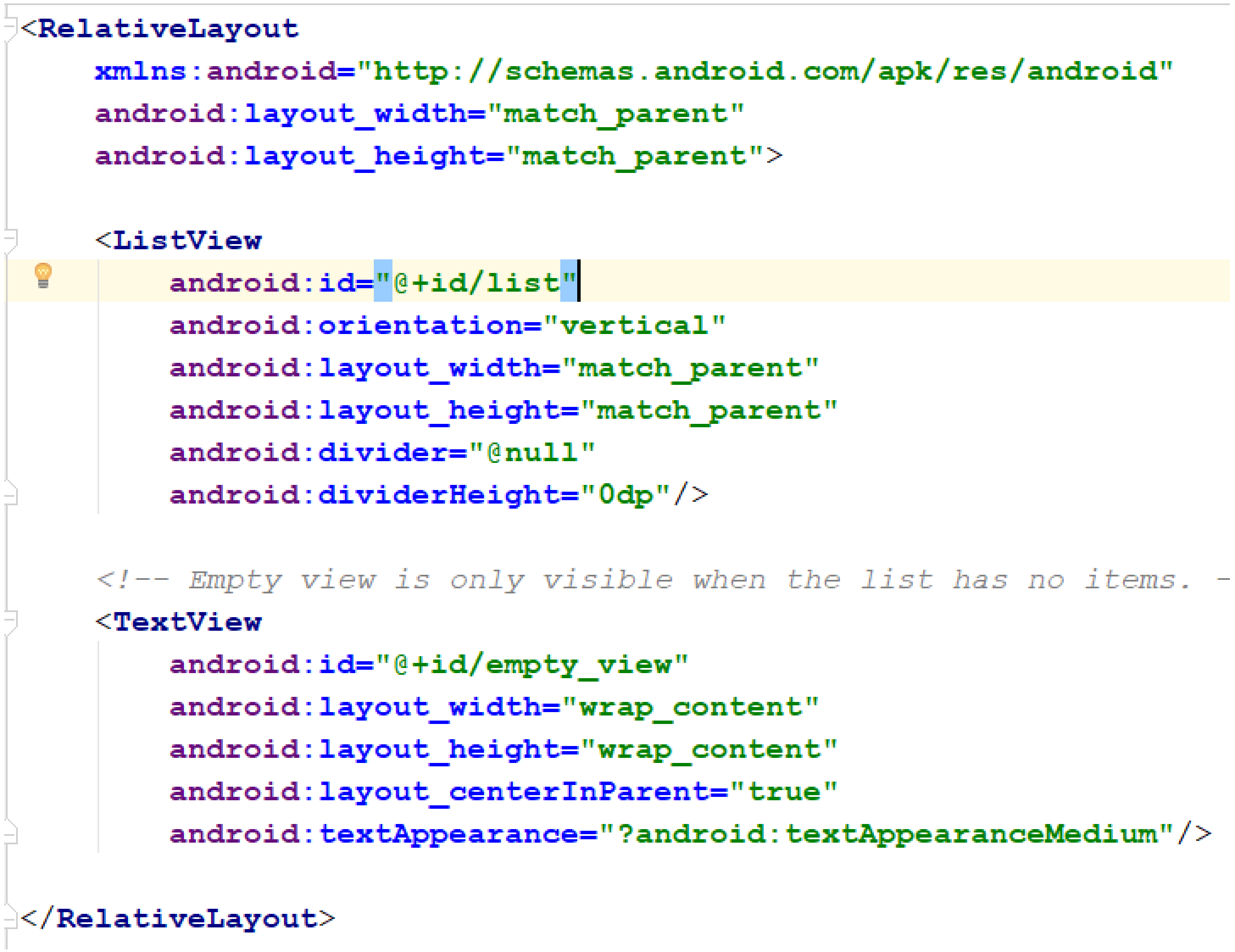
EMPTY VIEW


11:12
Quake Report
6.4 Sary-Tash, Kyrgyzstan
6.3 Namatanai, Papua New Guinea
6.1 Northern Mid-Atlantic Ridge
6.0 Isangel, Vanuatu
6.3 Isangel, Vanuatu
6.2 Isangel, Vanuatu
6.2 Auki, Solomon Islands
6.1 Puerto Morazan, Nicaragua
6.2 Kute, Indonesia

LISTVIEW

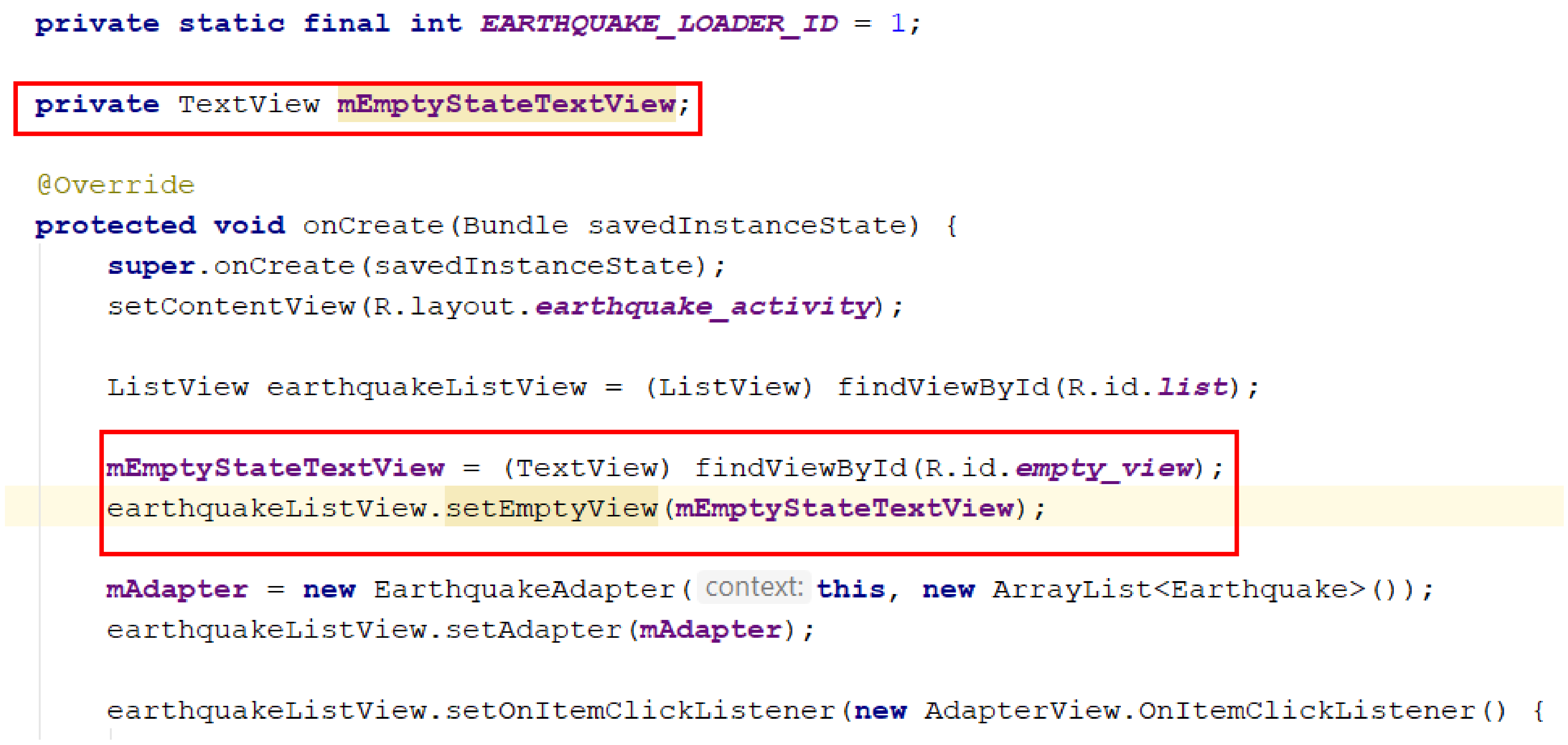
Can you think of a way to verify that the empty view works?
Hint: Temporarily comment out some code

Modify earthquake_activity.xml



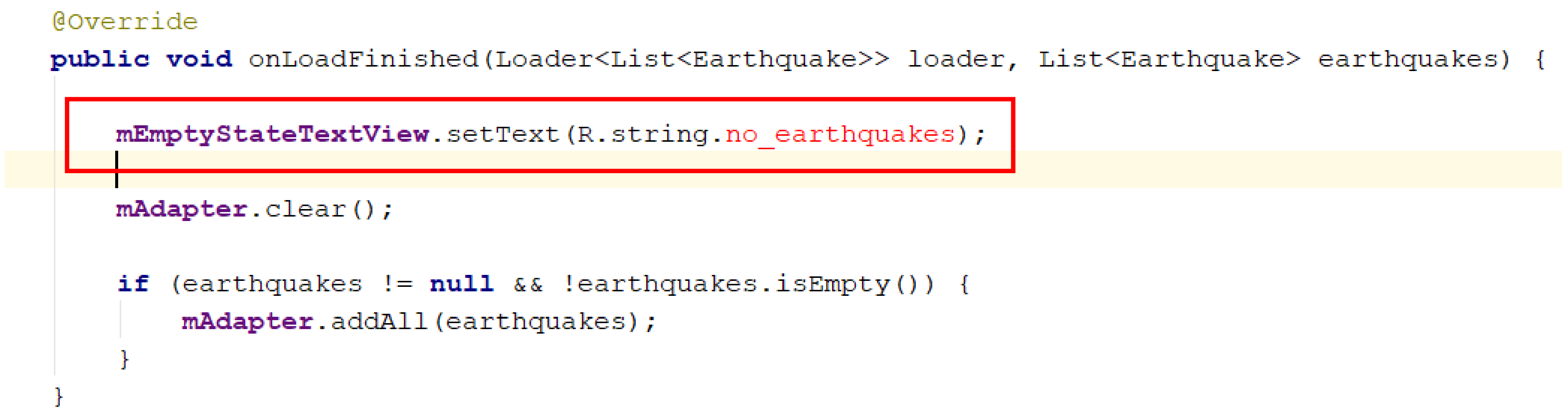
```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/list"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:divider="@null"
        android:dividerHeight="0dp"/>

    <!-- Empty view is only visible when the list has no items. -
    <TextView
        android:id="@+id/empty_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textAppearance="?android:textAppearanceMedium"/>

</RelativeLayout>
```

EarthquakeActivity.java – onCreate()



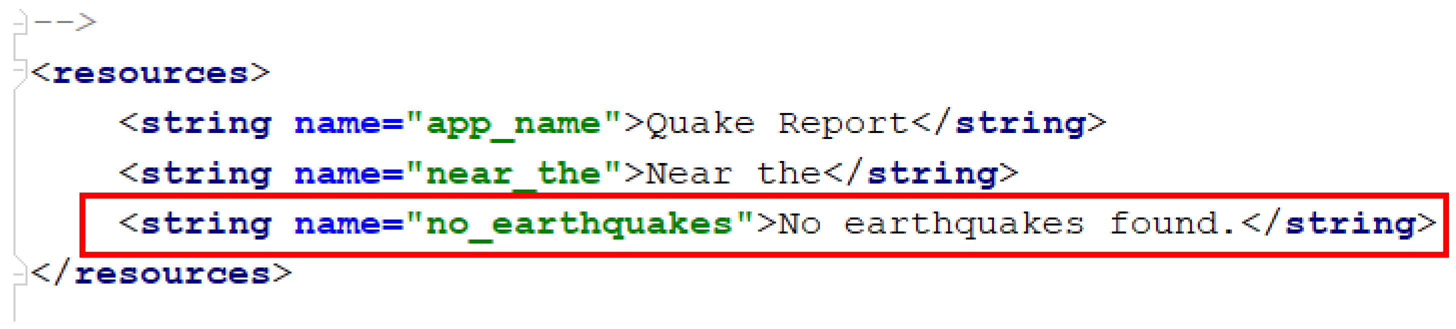
```
private static final int EARTHQUAKE_LOADER_ID = 1;

private TextView mEmptyStateTextView;

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

    ListView earthquakeListView = (ListView) findViewById(R.id.list);

    mEmptyStateTextView = (TextView) findViewById(R.id.empty_view);
    earthquakeListView.setEmptyView(mEmptyStateTextView);

    mAdapter = new EarthquakeAdapter( context: this, new ArrayList<Earthquake>());
    earthquakeListView.setAdapter(mAdapter);

    earthquakeListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
```

EarthquakeActivity.java – onLoadFinished()



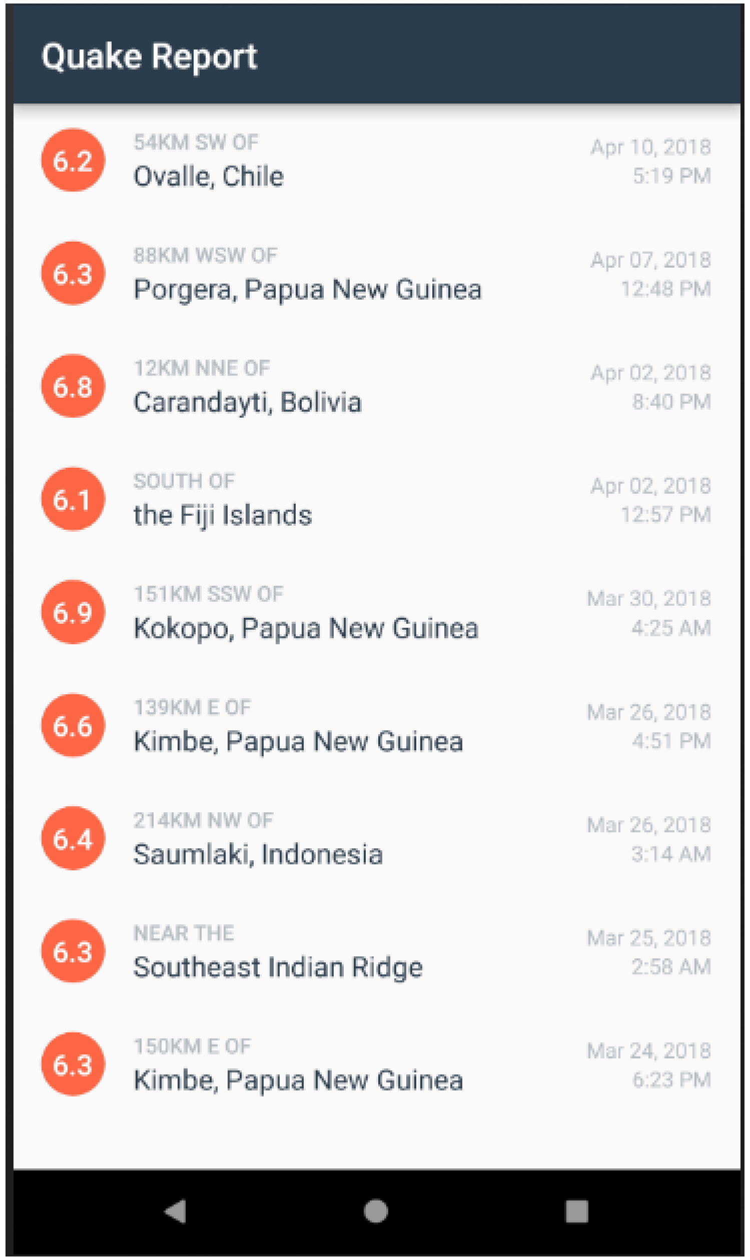
```java
@Override
public void onLoadFinished(Loader<List<Earthquake>> loader, List<Earthquake> earthquakes) {

    mEmptyStateTextView.setText(R.string.no_earthquakes);

    mAdapter.clear();

    if (earthquakes != null && !earthquakes.isEmpty()) {
        mAdapter.addAll(earthquakes);
    }
}
```

Modify strings.xml



```xml
-->
<resources>
    <string name="app_name">Quake Report</string>
    <string name="near_the">Near the</string>
    <string name="no_earthquakes">No earthquakes found.</string>
</resources>
```

Run the app on your device to check that it still works.


Quake Report

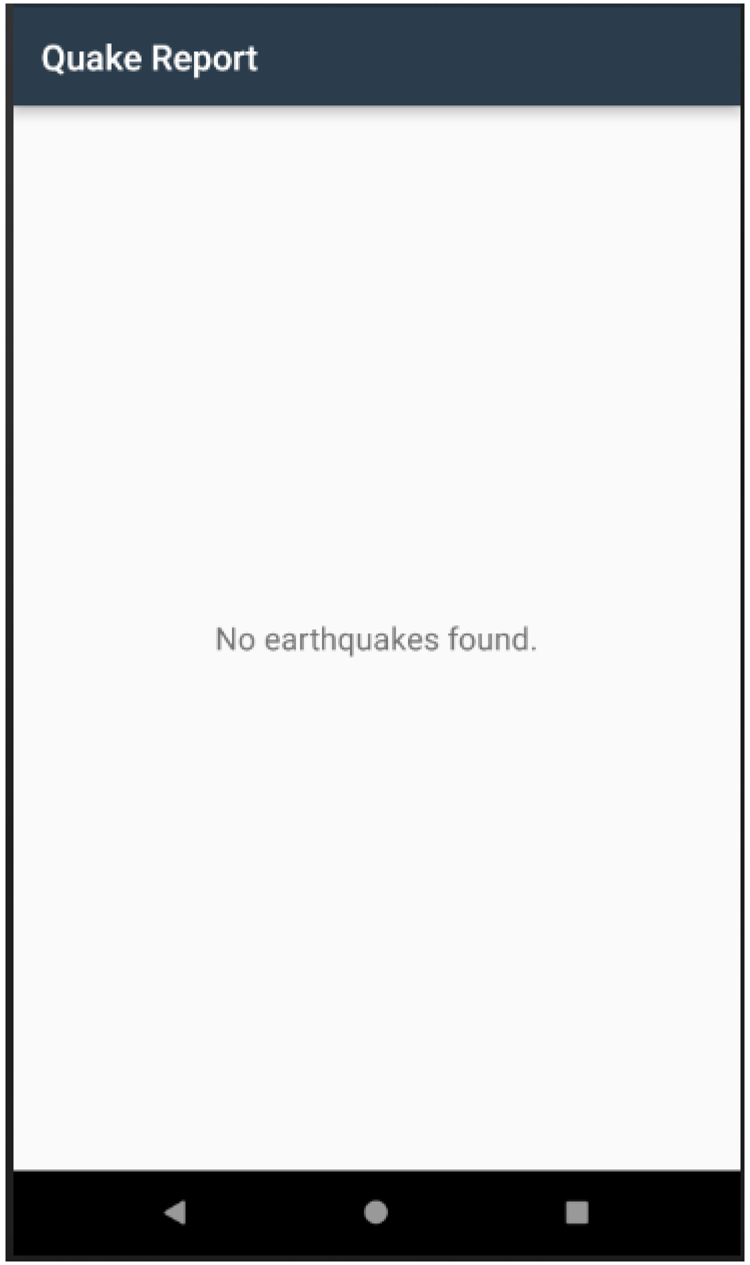
6.2	54KM SW OF Ovalle, Chile	Apr 10, 2018 5:19 PM
6.3	88KM WSW OF Porgera, Papua New Guinea	Apr 07, 2018 12:48 PM
6.8	12KM NNE OF Carandayti, Bolivia	Apr 02, 2018 8:40 PM
6.1	SOUTH OF the Fiji Islands	Apr 02, 2018 12:57 PM
6.9	151KM SSW OF Kokopo, Papua New Guinea	Mar 30, 2018 4:25 AM
6.6	139KM E OF Kimbe, Papua New Guinea	Mar 26, 2018 4:51 PM
6.4	214KM NW OF Saumlaki, Indonesia	Mar 26, 2018 3:14 AM
6.3	NEAR THE Southeast Indian Ridge	Mar 25, 2018 2:58 AM
6.3	150KM E OF Kimbe, Papua New Guinea	Mar 24, 2018 6:23 PM

To test the empty state, you can temporarily comment out the line of code that adds earthquake data to the adapter, which is the mAdapter.addAll(earthquakes) method call. This will pretend like 0 results came back from the web server, and you should see the empty state in the app.

In EarthquakeActivity.java:

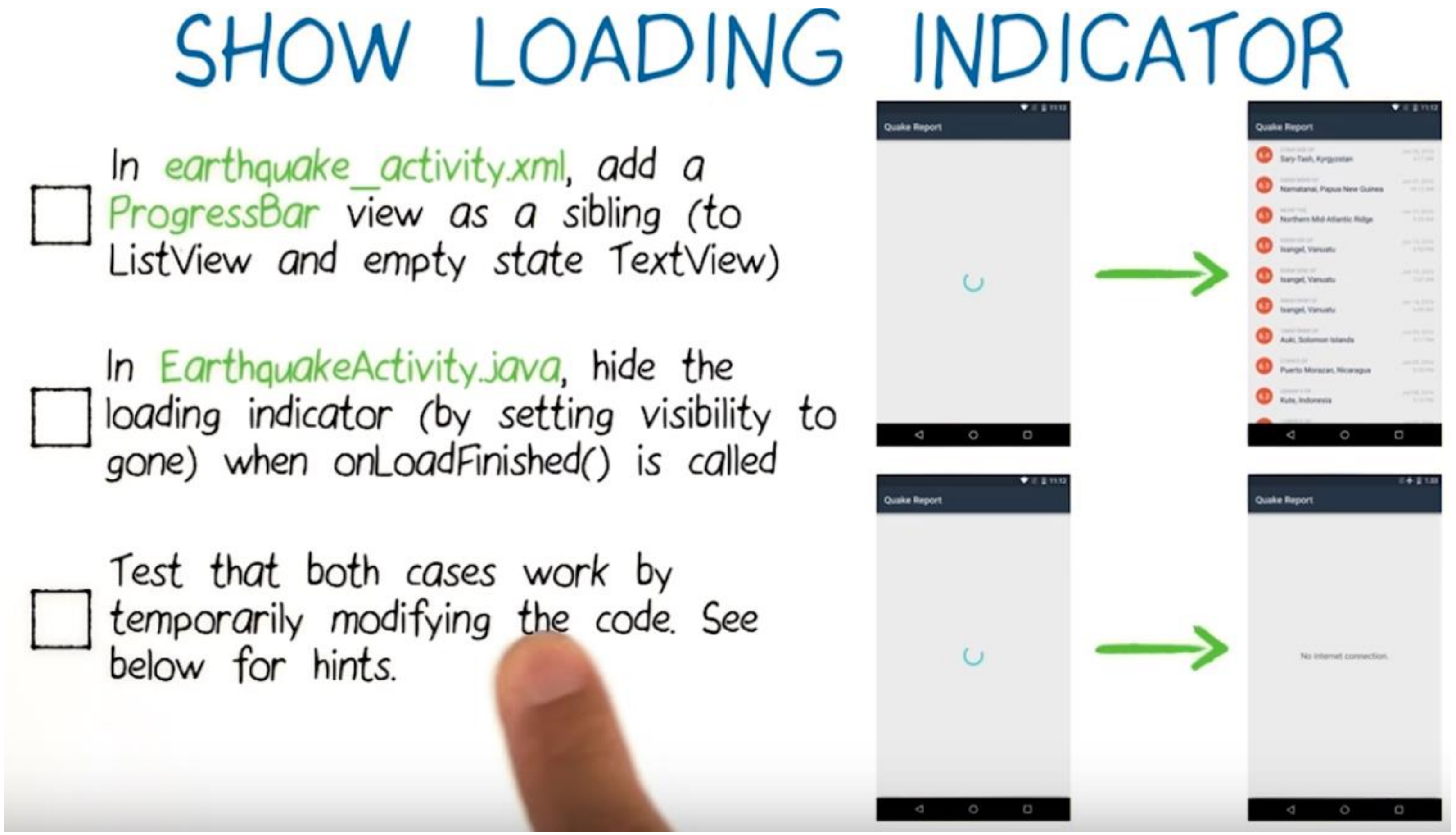
```
@Override
public void onLoadFinished(Loader<List<Earthquake>> loader, List<Earthquake> earthquakes) {
    // Set empty state text to display "No earthquakes found."
    mEmptyStateTextView.setText(R.string.no_earthquakes);

    // Clear the adapter of previous earthquake data
    mAdapter.clear();

    // If there is a valid list of {@link Earthquake}s, then add them to the adapter's
    // data set. This will trigger the ListView to update.
    if (earthquakes != null && !earthquakes.isEmpty()) {
        // mAdapter.addAll(earthquakes);
    }
}
```

Run APP


Quake Report

No earthquakes found.

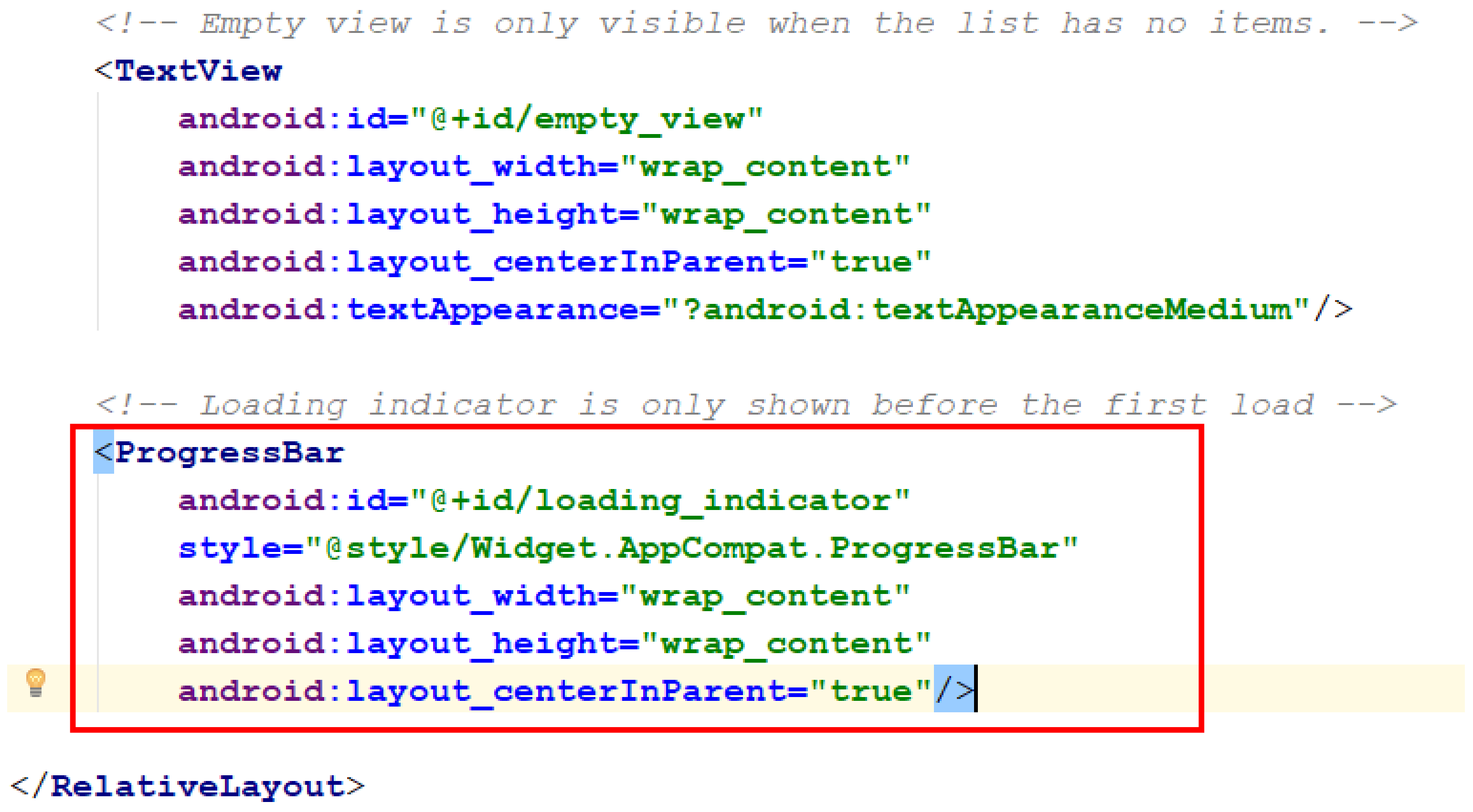
12 - Add loading indicator



SHOW LOADING INDICATOR

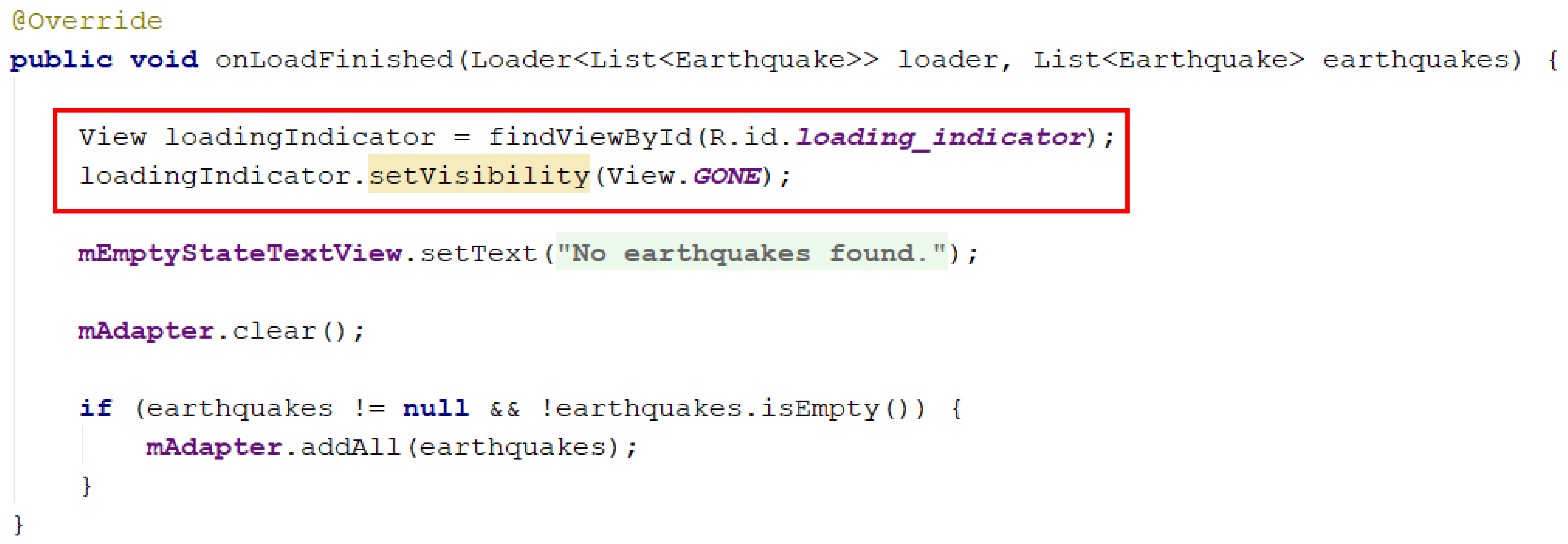
- [ ] In earthquake_activity.xml, add a ProgressBar view as a sibling (to ListView and empty state TextView)
- [ ] In EarthquakeActivity.java, hide the loading indicator (by setting visibility to gone) when onLoadFinished() is called
- [ ] Test that both cases work by temporarily modifying the code. See below for hints.

Modify earthquake_activity.xml



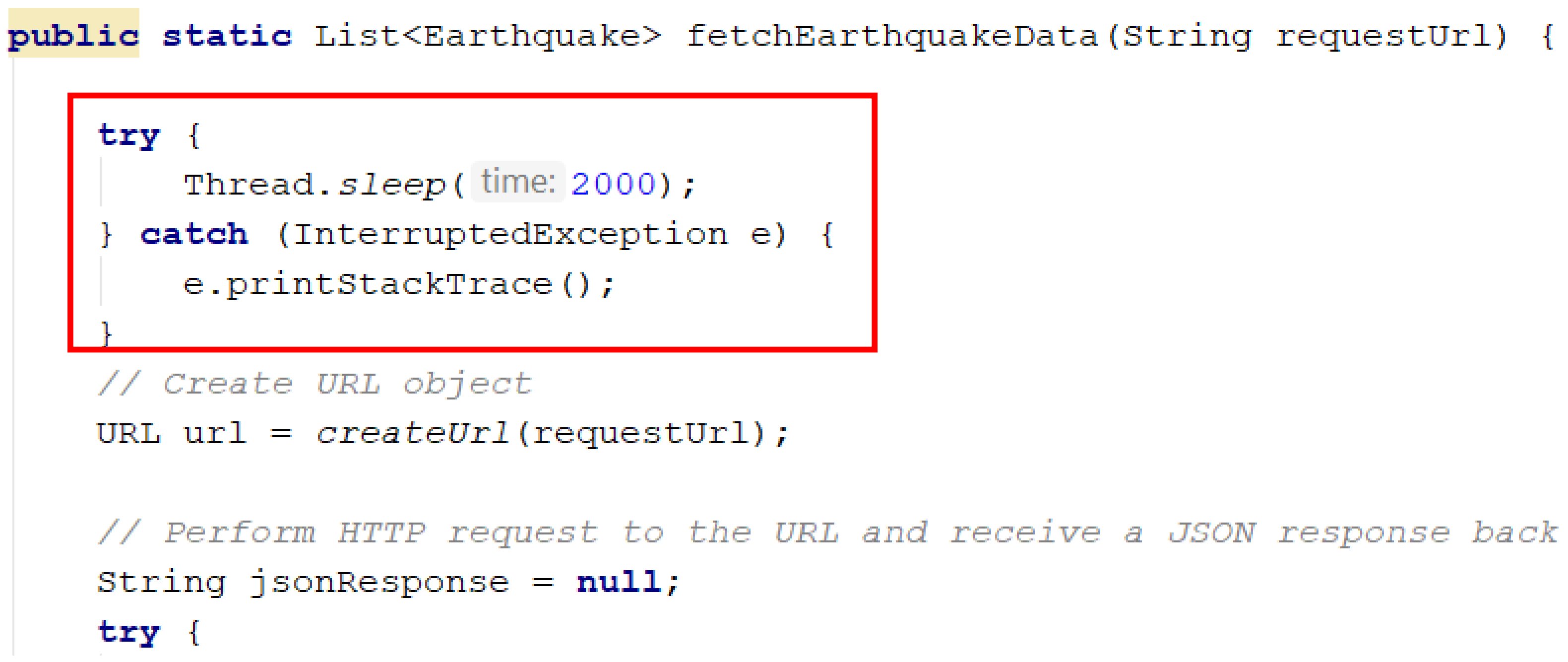
```
    <!-- Empty view is only visible when the list has no items. -->
    <TextView
        android:id="@+id/empty_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textAppearance="?android:textAppearanceMedium"/>

    <!-- Loading indicator is only shown before the first load -->
    <ProgressBar
        android:id="@+id/loading_indicator"
        style="@style/Widget.AppCompat.ProgressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"/>

</RelativeLayout>
```

EarthquakeActivity.java – onLoadFinished()

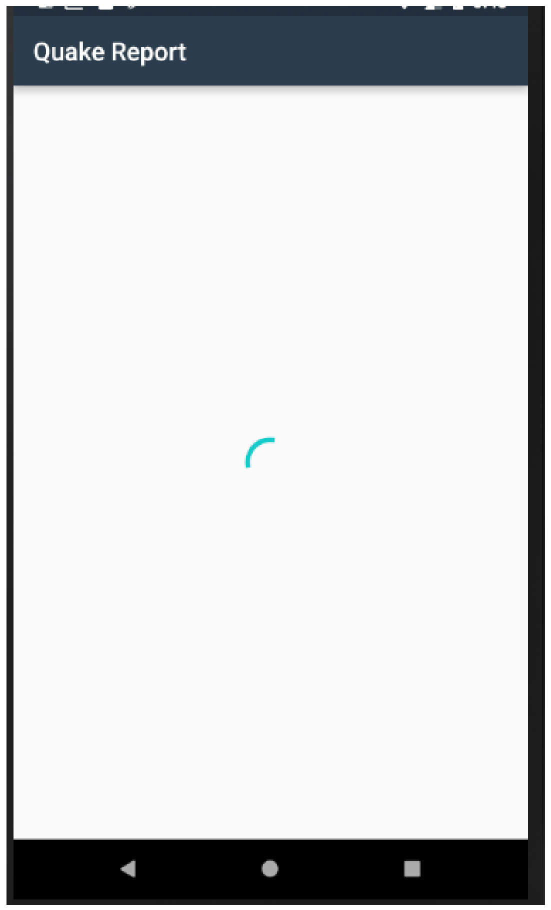


```java
@Override
public void onLoadFinished(Loader<List<Earthquake>> loader, List<Earthquake> earthquakes) {

    View loadingIndicator = findViewById(R.id.loading_indicator);
    loadingIndicator.setVisibility(View.GONE);

    mEmptyStateTextView.setText("No earthquakes found.");

    mAdapter.clear();

    if (earthquakes != null && !earthquakes.isEmpty()) {
        mAdapter.addAll(earthquakes);
    }
}
```

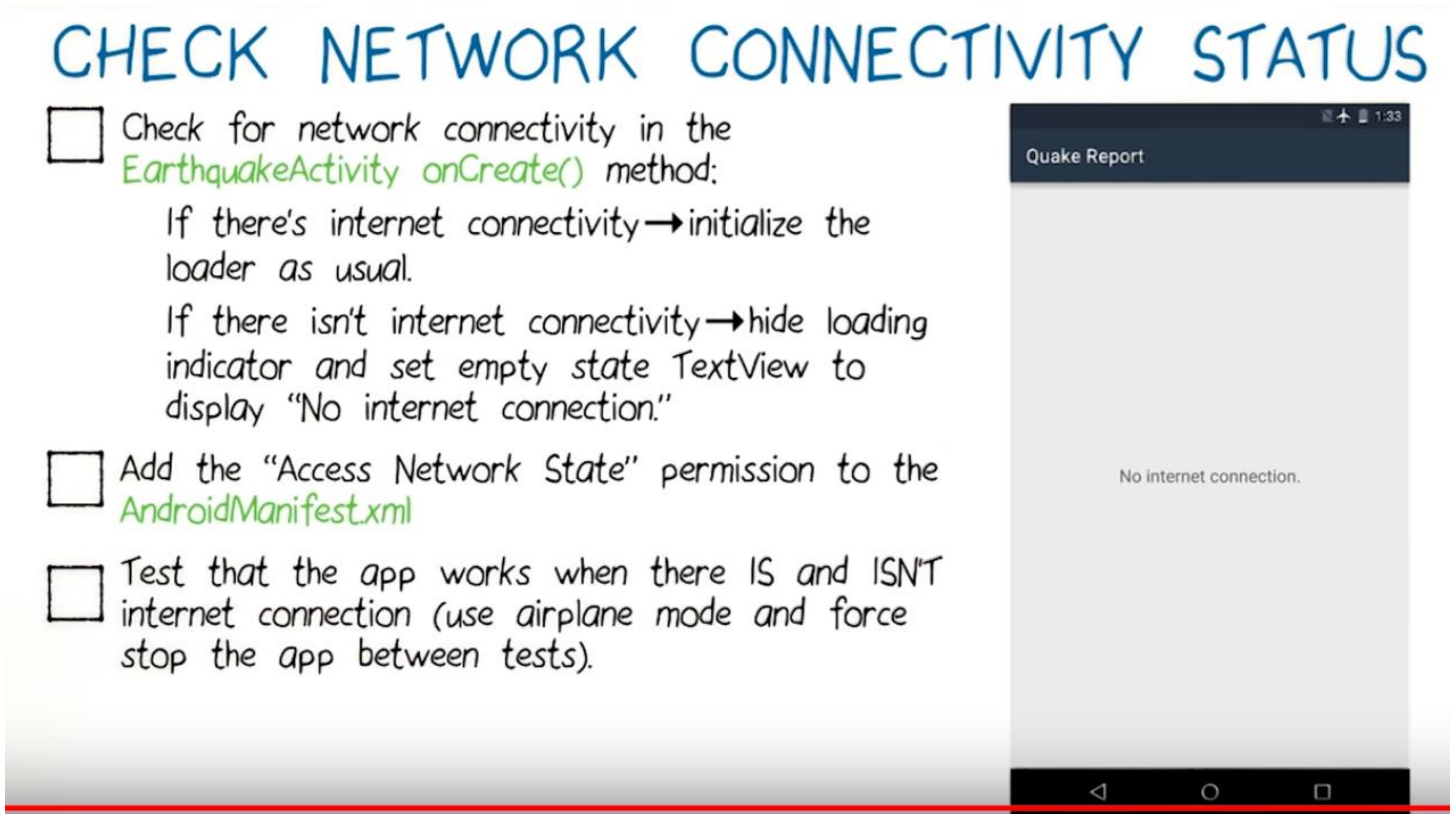
- Run the app on a device and the loading indicator should appear if the network call takes a long time. Sometimes, the internet connection may be so fast that the loading indicator is not visible on screen for enough time for the human eye to catch it.
- **Test #1: Force the background thread to sleep for 2 seconds (QueryUtils.java)**



```
public static List<Earthquake> fetchEarthquakeData(String requestUrl) {

    try {
        Thread.sleep( time: 2000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    // Create URL object
    URL url = createUrl(requestUrl);

    // Perform HTTP request to the URL and receive a JSON response back
    String jsonResponse = null;
    try {
```



13 - Check network connectivity status



CHECK NETWORK CONNECTIVITY STATUS

- [ ] Check for network connectivity in the EarthquakeActivity onCreate() method:

  If there's internet connectivity → initialize the loader as usual.

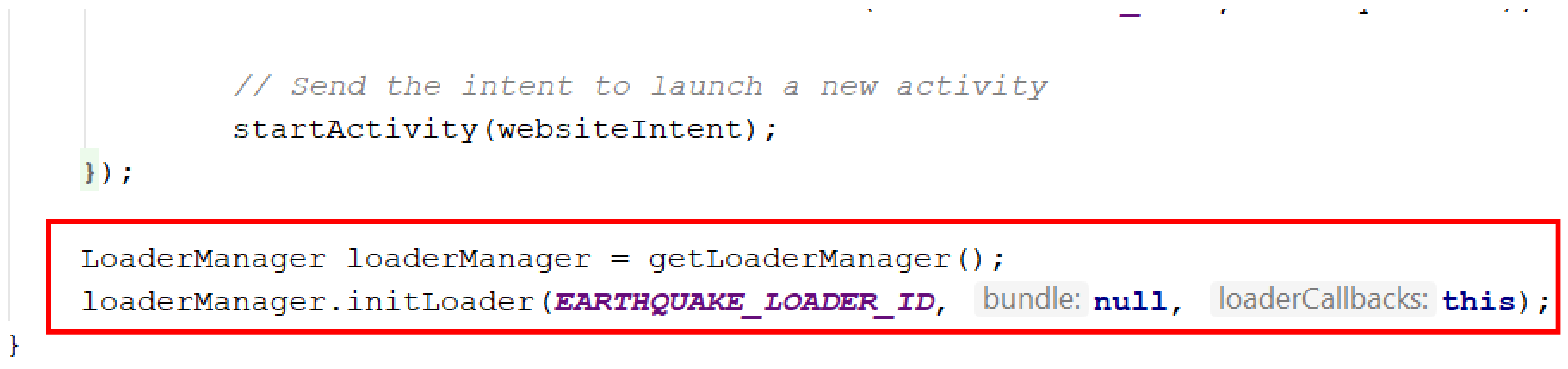
  If there isn't internet connectivity → hide loading indicator and set empty state TextView to display "No internet connection."

- [ ] Add the "Access Network State" permission to the AndroidManifest.xml

- [ ] Test that the app works when there IS and ISN'T internet connection (use airplane mode and force stop the app between tests).

EarthquakeActivity.java – onCreate()

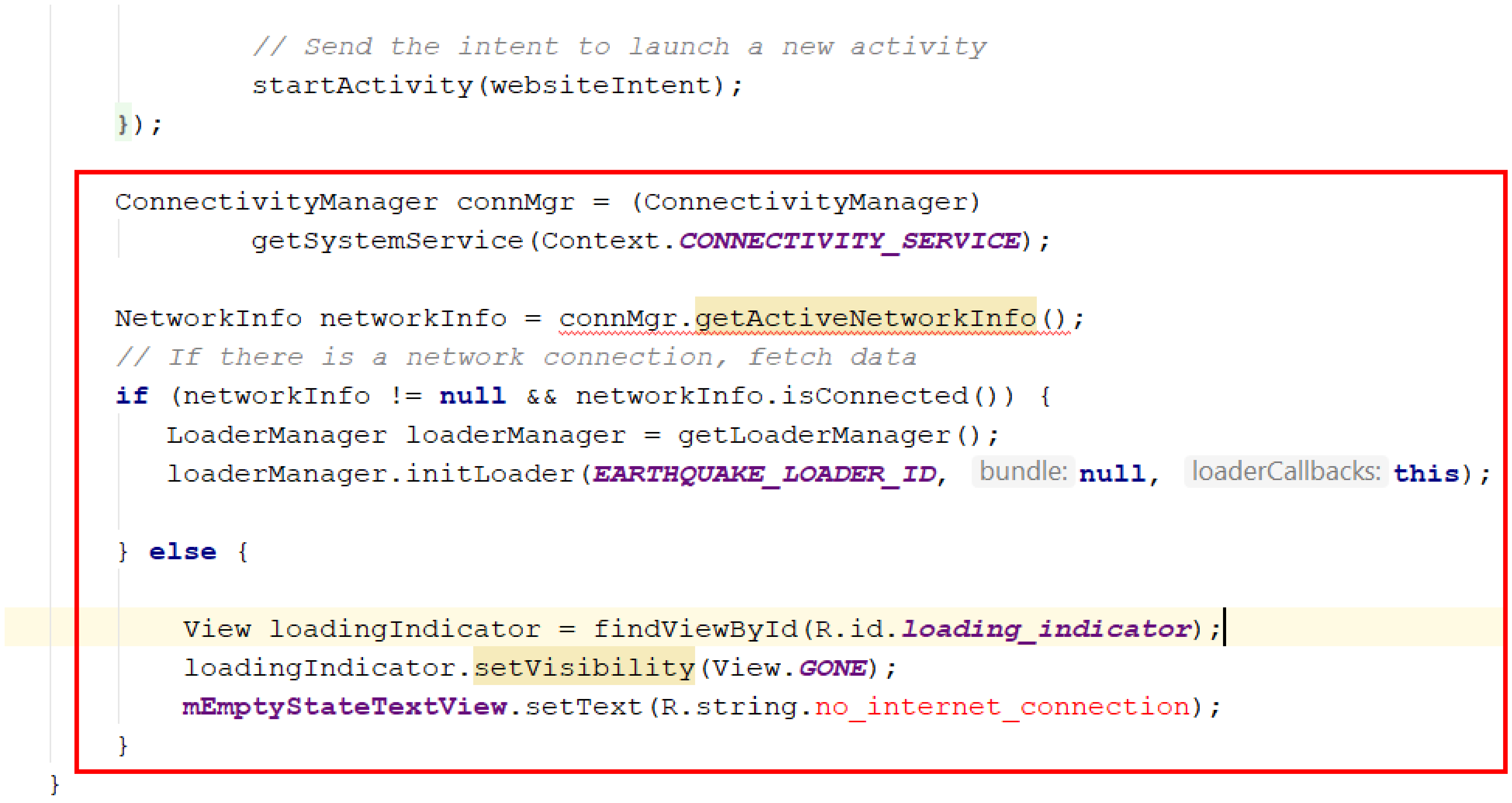
- Move this code



```
                // Send the intent to launch a new activity
                startActivity(websiteIntent);
        });

        LoaderManager loaderManager = getLoaderManager();
        loaderManager.initLoader(EARTHQUAKE_LOADER_ID,  bundle: null,  loaderCallbacks: this);
}
```

- into ConnectivityManager Code (as shown in the next slide)

EarthquakeActivity.java – onCreate()



```
            // Send the intent to launch a new activity
            startActivity(websiteIntent);
        });

        ConnectivityManager connMgr = (ConnectivityManager)
                getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        // If there is a network connection, fetch data
        if (networkInfo != null && networkInfo.isConnected()) {
            LoaderManager loaderManager = getLoaderManager();
            loaderManager.initLoader(EARTHQUAKE_LOADER_ID, bundle: null, loaderCallbacks: this);

        } else {

             View loadingIndicator = findViewById(R.id.loading_indicator);
             loadingIndicator.setVisibility(View.GONE);
             mEmptyStateTextView.setText(R.string.no_internet_connection);
        }
    }
```

Modify :

- Add permission to AndroidManifest.xml

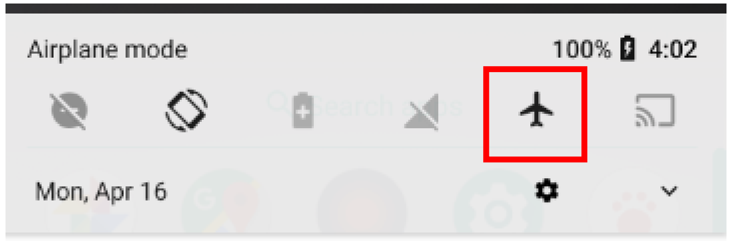
```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

- Add string to strings.xml

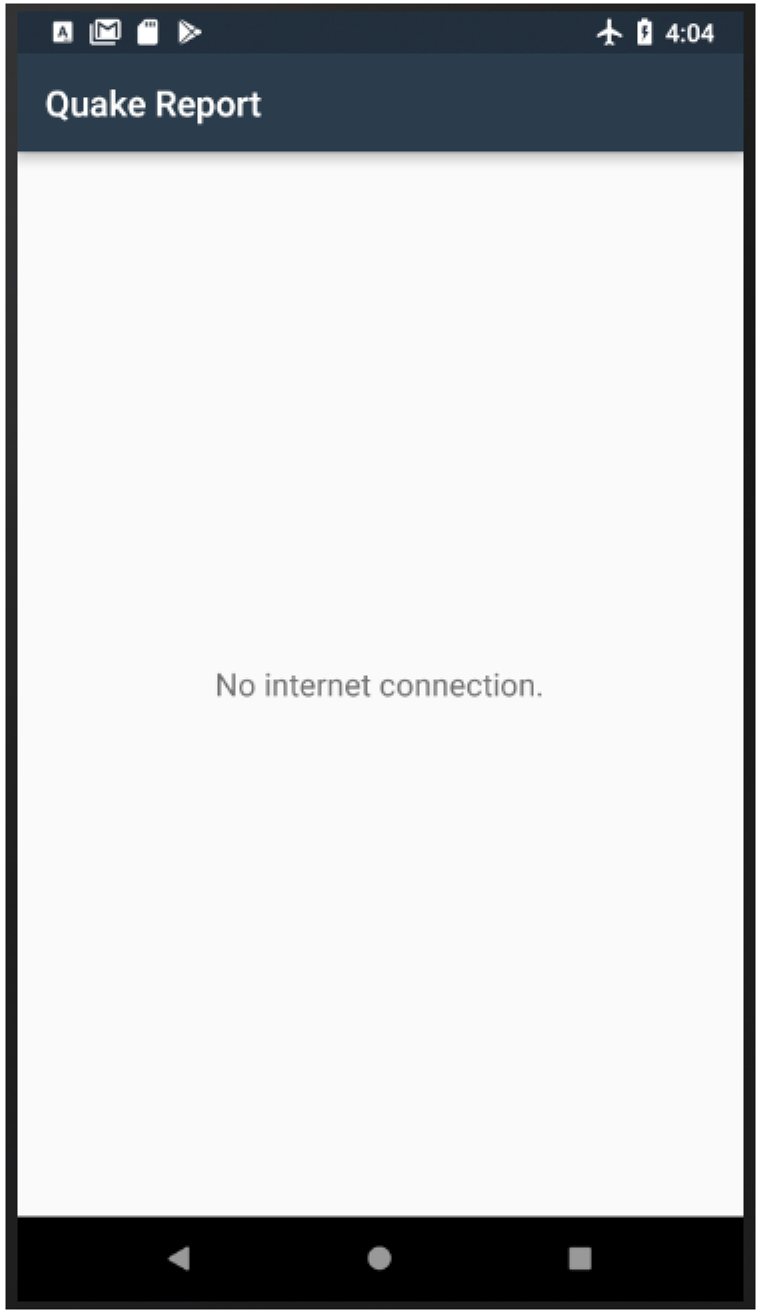
```
<!-- Error message when there is no internet connectivity [CHAR LIMIT=NONE] -->
<string name="no_internet_connection">No internet connection.</string>
```

Run APP

- Set “Air Plane Mode” to Active



- Clear/Kill App
- Run App


4:04
Quake Report
No internet connection.