Mathew McGerty-S1036834 Mobile Programming Document

For the Mobile Programming Coursework I was required to produce an android application using java and eclipse to design and produce information about car park spaces that are available in Glasgow. I was required to parse an xml stream and extract appropriate pieces of information for my design of my application to be produced. For my coursework I mainly focused on getting the design of the application displaying the main information that has been parsed and handling the portrait and landscape layouts of the app.

Design - Portrait

For the design for the application I have created two xml files for my portrait layout. The first one being my main.xml and in this xml was going to be designed using a relative layout and was going to display my list view. The purpose of using a list view being that it would display the carparks in a list that will be scrollable so you that a lot of information can be displayed.

In the second xml file I created was going to be called cp\_layout.xml and this was going to have all of my text views in order to display my information that is being parsed In the format that I feel would best be displayed.

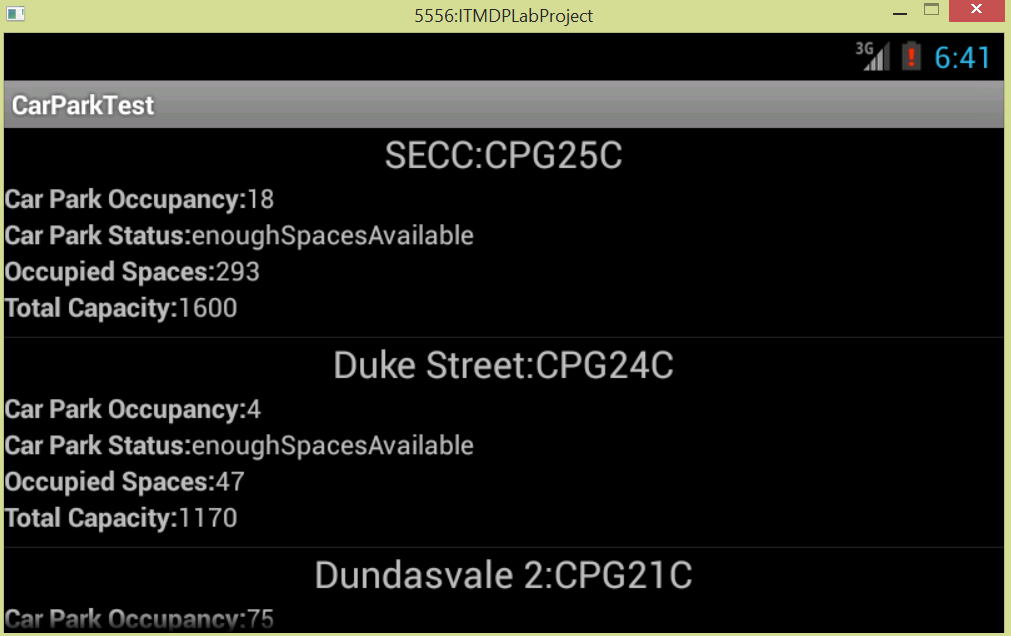
For the first text view was going to have the name of the carparks that I was going to have displayed would be horizontally centred so that the name of the car park would be displayed I the middle of the screen. Also the text size would be set to 20sp so that the size of the text would be larger so the name of the carpark will stand out more. For the next two text views will have the car park occupancy and the first text view will have information that will display the text showing the person using the app what it is exactly what it is that is being displayed. This will be aligned to the left to show that it comes before the integer and it is set to be bold so that it will show more emphasis out to the viewer. The next text view will then show the data that is being parsed from the text view. It will be displayed to the right of the text view that is tagged as occupancy so that it will be displayed right after the name of whatever is parsed is shown. These text views will also be displayed below the first text view that was displaying the car parks The rest of the text views all follow the same pattern for the rest of the data that is going to be parsed Inside of the list view all being displayed below on another and for each pieces of the information parsed will have two separate text views the first one having the text that displays about the information being parsed and the second one displays the actual parsing of the data.

Once all of the xml data has the required design then I had to create three classes. The main one that was going to display all of the data being parsed in text views inside of a list view was going to be my car park adaptor class that I created. This class was going to display all of my information in rows for each of the car parks that are being parsed to make it look like a list inside of a list view. A lot of the class involves setting text that has been retrieved from the separate parts of the xml into text views that I have set up. Then I would have to call upon this array class inside the on create method in order for my app to display the data the way that I intended it to.



Design-Landscape

For having all of my information displayed in landscape I would have to create a new folder called layout-land under the resources folder and then have to copy both the xml files into the new folder. I then had to add android configure changes to orientation and android screen orientation to sensor in order for the data to be loaded when I changed the screen from portrait to landscape when the application was running. This allowed the format of the application to remain the way I liked it when changing portrait to landscape.



Testing

During the testing phase of the application I had to employ different test methods in order to confirm that the code was doing what was required and when the application would suddenly have to stop working due a logic error inside of the code. This problem would usually have to take a while and a number of testing strategies would have to be fixed in order to employ it.

The first instance that I had to set up a test method was when I was monitoring the debug log through the use of log statements. This would involve using Log.e statements to confirm that when the application was running then during the if statements if the information was being parsed then a message in the log cat would displayed the message that the data was being parsed. During the parsing phase it wouldn’t be syntax errors I was getting but logic errors when the data was not being parsed because there would be a message that I created in Log.e saying that there is an error in the parsing of the data. A lot of the problems mainly had to do with the fact that I had entered the name of the tags incorrectly which when I had actually tested again It would show the information being parsed in the log cat.

The second instance that I had to set up a method that I would have to use would be using breakpoints. I would have to use breakpoints when the black screen would appear and the application would stop working. I would have to put a lot of breakpoints in the code and have to step over the code to confirm that each part was working correctly. When I did this I would discover that the debugger would crash when I tried to step over my array adaptor so I realized that something was wrong with the adaptor. It wasn’t until a while that I discovered that It wasn’t the array adaptor that was broken it was the face that the order in which my code was set up was incorrect in the case that I was setting up the array adaptor before I was parsing the code so no information was being put in the empty array adaptor.

The third instance would have to deal with the problem that it would only display the last car park in the list View after I eventually got the array adaptor working. Through the use of having breakpoints and using log statements I discovered that there was a problem with the parsing method itself. I had to change the parsing method so that it would display all of the carparks in the list view instead of just the last one. This would be due to using breakpoints and stepping over parts of the parsing method that I would be able to resolve this issue.

Xml

AndroidManifest.Xml

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

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"org.me.myandroidstuff"*

android:versionCode=*"1"*

android:versionName=*"1.0"*>

<uses-sdk android:minSdkVersion=*"7"* />

<application android:icon=*"@drawable/icon"* android:label=*"@string/app\_name"*>

<activity android:name=*"org.me.myandroidstuff.CarParkListingTestActivity"*

android:label=*"@string/app\_name"*

android:configChanges=*"orientation"*

android:screenOrientation=*"sensor"*>

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

</application>

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

</manifest>

Portrait

Main.Xml

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

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*>

<ListView

android:id=*"@+id/carParks"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* >

</ListView>

</RelativeLayout>

cp\_layout.Xml

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

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"* >

<TextView

android:id=*"@+id/car\_Park\_Identity"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerHorizontal=*"true"*

android:textSize=*"20sp"*

/>

<TextView

android:id=*"@+id/Occupancy"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"26dip"*

android:layout\_alignParentLeft=*"true"*

android:textStyle=*"bold"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

android:layout\_below=*"@id/car\_Park\_Identity"*

/>

<TextView

android:id=*"@+id/car\_Park\_Occupancy"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@id/car\_Park\_Identity"*

android:layout\_toRightOf=*"@id/Occupancy"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

/>

<TextView

android:id=*"@+id/CarParkStatus"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"26dip"*

android:layout\_alignParentLeft=*"true"*

android:textStyle=*"bold"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

android:layout\_below=*"@id/car\_Park\_Occupancy"*

/>

<TextView

android:id=*"@+id/car\_Park\_Status"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@id/car\_Park\_Occupancy"*

android:layout\_toRightOf=*"@id/CarParkStatus"*

/>

<TextView

android:id=*"@+id/OccupiedSpaces"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"26dip"*

android:layout\_alignParentLeft=*"true"*

android:textStyle=*"bold"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

android:layout\_below=*"@id/car\_Park\_Status"*

/>

<TextView

android:id=*"@+id/occupied\_Spaces"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@id/car\_Park\_Status"*

android:layout\_toRightOf=*"@id/OccupiedSpaces"*

/>

<TextView

android:id=*"@+id/TotalCapacity"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"26dip"*

android:layout\_alignParentLeft=*"true"*

android:textStyle=*"bold"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

android:layout\_below=*"@id/occupied\_Spaces"*

/>

<TextView

android:id=*"@+id/total\_Capacity"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@id/occupied\_Spaces"*

android:layout\_toRightOf=*"@id/TotalCapacity"*

/>

</RelativeLayout>

LandScape

Main.xml

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

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*>

<ListView

android:id=*"@+id/carParks"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* >

</ListView>

</RelativeLayout>

cp\_layout.xml

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

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"horizontal"*>

<TextView

android:id=*"@+id/car\_Park\_Identity"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerHorizontal=*"true"*

android:textSize=*"20sp"*

/>

<TextView

android:id=*"@+id/Occupancy"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"26dip"*

android:layout\_alignParentLeft=*"true"*

android:textStyle=*"bold"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

android:layout\_below=*"@id/car\_Park\_Identity"*

/>

<TextView

android:id=*"@+id/car\_Park\_Occupancy"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@id/car\_Park\_Identity"*

android:layout\_toRightOf=*"@id/Occupancy"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

/>

<TextView

android:id=*"@+id/CarParkStatus"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"26dip"*

android:layout\_alignParentLeft=*"true"*

android:textStyle=*"bold"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

android:layout\_below=*"@id/car\_Park\_Occupancy"*

/>

<TextView

android:id=*"@+id/car\_Park\_Status"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@id/car\_Park\_Occupancy"*

android:layout\_toRightOf=*"@id/CarParkStatus"*

/>

<TextView

android:id=*"@+id/OccupiedSpaces"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"26dip"*

android:layout\_alignParentLeft=*"true"*

android:textStyle=*"bold"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

android:layout\_below=*"@id/car\_Park\_Status"*

/>

<TextView

android:id=*"@+id/occupied\_Spaces"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@id/car\_Park\_Status"*

android:layout\_toRightOf=*"@id/OccupiedSpaces"*

/>

<TextView

android:id=*"@+id/TotalCapacity"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"26dip"*

android:layout\_alignParentLeft=*"true"*

android:textStyle=*"bold"*

android:singleLine=*"true"*

android:ellipsize=*"marquee"*

android:layout\_below=*"@id/occupied\_Spaces"*

/>

<TextView

android:id=*"@+id/total\_Capacity"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@id/occupied\_Spaces"*

android:layout\_toRightOf=*"@id/TotalCapacity"*

/>

</RelativeLayout>

Class Code

carParkData.java

**package** org.me.myandroidstuff;

**public** **class** carParkData {

**public** String carParkIdentity;

**public** String carParkOccupancy;

**public** String carParkStatus;

**public** String occupiedSpaces;

**public** String totalCapacity;

**public** carParkData()

{

carParkIdentity = "";

carParkOccupancy = "";

carParkStatus = "";

occupiedSpaces = "";

totalCapacity = "";

}

**public** carParkData(String aCarParkIdentity, String aCarParkOccupancy, String aCarParkStatus,

String aOccupiedSpaces, String aTotalCapacity)

{

carParkIdentity = aCarParkIdentity;

carParkOccupancy = aCarParkOccupancy;

carParkStatus = aCarParkStatus;

occupiedSpaces = aOccupiedSpaces;

totalCapacity = aTotalCapacity;

}

**public** String getcarParkIdentity()

{

**return** carParkIdentity;

}

**public** **void** setcarParkIdentity(String aCarParkIdentity)

{

carParkIdentity = aCarParkIdentity;

}

**public** String getcarParkOccupancy()

{

**return** carParkOccupancy;

}

**public** **void** setcarParkOccupancy(String aCarParkOccupancy)

{

carParkOccupancy = aCarParkOccupancy;

}

**public** String getcarParkStatus()

{

**return** carParkStatus;

}

**public** **void** setcarParkStatus(String aCarParkStatus)

{

carParkStatus = aCarParkStatus;

}

**public** String getoccupiedSpaces()

{

**return** occupiedSpaces;

}

**public** **void** setoccupiedSpaces(String aOccupiedSpaces)

{

occupiedSpaces = aOccupiedSpaces;

}

**public** String gettotalCapacity()

{

**return** totalCapacity;

}

**public** **void** settotalCapacity(String aTotalCapacity)

{

totalCapacity = aTotalCapacity;

}

**public** String toString()

{

String temp;

temp = carParkIdentity + " " + carParkOccupancy + " " + carParkStatus + " " +

occupiedSpaces + " " + totalCapacity;

**return** temp;

}

}

CarParkAdaptor.java

**package** org.me.myandroidstuff;

**import** java.util.ArrayList;

**import** java.util.LinkedList;

**import** android.content.Context;

**import** android.view.LayoutInflater;

**import** android.view.View;

**import** android.view.ViewGroup;

**import** android.widget.ArrayAdapter;

**import** android.widget.RelativeLayout;

**import** android.widget.TextView;

**public** **class** CarParkAdaptor **extends** ArrayAdapter<carParkData> {

**private** LinkedList<carParkData> alist;

**public** CarParkAdaptor(Context context, **int** textViewResourceId, LinkedList<carParkData> alist) {

**super**(context, textViewResourceId, alist);

// **TODO** Auto-generated constructor stub

**this**.alist = alist;

}

**public** View getView(**int** pos, View convertView, ViewGroup parent )

{

RelativeLayout row = (RelativeLayout)convertView;

**if**(**null** == row){

//No recycled View, we have to inflate one.

LayoutInflater inflater = (LayoutInflater)parent.getContext().getSystemService(Context.***LAYOUT\_INFLATER\_SERVICE***);

row = (RelativeLayout)inflater.inflate(R.layout.***cp\_layout***, parent, **false**);

}

//carParkData aCarParkIdentity = alist.get(pos);

TextView carParkIdentity = (TextView)row.findViewById(R.id.***car\_Park\_Identity***);

carParkIdentity.setText(getItem(pos).getcarParkIdentity());

TextView textCarParkOccupancy =(TextView)row.findViewById(R.id.***Occupancy***);

TextView carParkOccupancy = (TextView)row.findViewById(R.id.***car\_Park\_Occupancy***);

textCarParkOccupancy.setText("Car Park Occupancy:");

carParkOccupancy.setText(getItem(pos).getcarParkOccupancy());

TextView textcarParkStatus = (TextView)row.findViewById(R.id.***CarParkStatus***);

TextView carParkStatus = (TextView)row.findViewById(R.id.***car\_Park\_Status***);

textcarParkStatus.setText("Car Park Status:");

carParkStatus.setText(getItem(pos).getcarParkStatus());

TextView textOccupiedSpaces = (TextView)row.findViewById(R.id.***OccupiedSpaces***);

TextView occupiedSpaces = (TextView)row.findViewById(R.id.***occupied\_Spaces***);

textOccupiedSpaces.setText("Occupied Spaces:");

occupiedSpaces.setText(getItem(pos).getoccupiedSpaces());

TextView texttotalCapacity = (TextView)row.findViewById(R.id.***TotalCapacity***);

TextView totalCapacity = (TextView)row.findViewById(R.id.***total\_Capacity***);

texttotalCapacity.setText("Total Capacity:");

totalCapacity.setText(getItem(pos).gettotalCapacity());

**return** row;

}

}

CarParkListingTestActivity.java

**package** org.me.myandroidstuff;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.InputStreamReader;

**import** java.io.StringReader;

**import** java.net.HttpURLConnection;

**import** java.net.URL;

**import** java.net.URLConnection;

**import** java.util.ArrayList;

**import** java.util.LinkedList;

**import** org.xmlpull.v1.XmlPullParser;

**import** org.xmlpull.v1.XmlPullParserException;

**import** org.xmlpull.v1.XmlPullParserFactory;

//import com.makemyandroidapp.example.stacksites.R;

**import** android.app.Activity;

**import** android.content.Context;

**import** android.os.Bundle;

**import** android.util.Log;

**import** android.view.LayoutInflater;

**import** android.view.View;

**import** android.view.ViewGroup;

**import** android.widget.AdapterView;

**import** android.widget.ArrayAdapter;

**import** android.widget.ListView;

**import** android.widget.RelativeLayout;

**import** android.widget.TextView;

**import** android.widget.Toast;

**import** android.widget.AdapterView.OnItemClickListener;

**public** **class** CarParkListingTestActivity **extends** Activity

{

**private** TextView response;

**private** TextView errorText;

**private** String result;

**private** String sourceListingURL = "http://open.glasgow.gov.uk/api/live/parking.php?type=xml";

**private** ListView carParks;

**private** String text;

LinkedList <carParkData> alist;

// private CarParkAdaptor adapter;

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***main***);

LinkedList <carParkData> alist = **null**;

carParks = (ListView) findViewById(R.id.***carParks***);

alist = parseData(sourceListingURL);

// ArrayAdapter<carParkData> adapter = new ArrayAdapter<carParkData>(this, android.R.layout.simple\_list\_item\_1, alist);

// ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple\_list\_item\_1, items);

ArrayAdapter<carParkData> adapter = **new** CarParkAdaptor(getApplicationContext(), android.R.layout.***simple\_list\_item\_1***, alist);

carParks.setAdapter(adapter);

//Make call to parsing code

//Note this is not the best location

// carParks.setOnItemClickListener(new OnItemClickListener() {

// public void onItemClick(AdapterView<?> parent, View view, int position,

// long aCarParkCapacity) {

// String item = ((TextView)view).getText().toString();

// Toast.makeText(getBaseContext(), item, Toast.LENGTH\_LONG).show();

// }

//}

//);

// Write list to Log for testing

**if** (alist != **null**)

{

Log.*e*("MyTag","List not null");

**for** (Object o : alist)

{

Log.*e*("MyTag",o.toString());

}

}

**else**

{

Log.*e*("MyTag","List is null");

}

}

**public** LinkedList<carParkData> getCarParks()

{

**return** alist;

}

**private** LinkedList<carParkData> parseData(String dataToParse)

{

carParkData car = **null**;

//LinkedList <carParkData> alist = null;

alist = **new** LinkedList<carParkData>();

// Get the TextView object on which to display the results

//response = (TextView)findViewById(R.id.urlResponse);

**try**

{

dataToParse = *sourceListingString*(sourceListingURL);

XmlPullParserFactory factory = XmlPullParserFactory.*newInstance*();

factory.setNamespaceAware(**true**);

XmlPullParser xpp = factory.newPullParser();

xpp.setInput( **new** StringReader ( dataToParse ) );

**int** eventType = xpp.getEventType();

**while** (eventType != XmlPullParser.***END\_DOCUMENT***){

String tagname = xpp.getName();

**switch** (eventType) {

**case** XmlPullParser.***START\_TAG***:

**if**(tagname.equalsIgnoreCase("situationRecord")) {

car = **new** carParkData();

}

**break**;

**case** XmlPullParser.***TEXT***:

text = xpp.getText();

**break**;

**case** XmlPullParser.***END\_TAG***:

**if**(tagname.equalsIgnoreCase("situationRecord")) {

alist.add(car);

}**else** **if**(tagname.equalsIgnoreCase("carParkIdentity")) {

car.setcarParkIdentity(text);

}**else** **if**(tagname.equalsIgnoreCase("carParkOccupancy")) {

car.setcarParkOccupancy(text);

}**else** **if**(tagname.equalsIgnoreCase("carParkStatus")) {

car.setcarParkStatus(text);

}**else** **if**(tagname.equalsIgnoreCase("occupiedSpaces")) {

car.setoccupiedSpaces(text);

}**else** **if**(tagname.equalsIgnoreCase("totalCapacity")) {

car.settotalCapacity(text);

}

**break**;

**default**:

**break**;

}

// Found a start tag

//if(eventType == XmlPullParser.START\_TAG)

//{

// Check which Tag we have

// if (xpp.getName().equalsIgnoreCase("situation"))

// {

// alist = new LinkedList<carParkData>();

// }

// else

// if (xpp.getName().equalsIgnoreCase("situationRecord"))

// {

// Log.e("MyTag","Item Start Tag found");

// car = new carParkData();

//

// }

// else

// if (xpp.getName().equalsIgnoreCase("carParkIdentity"))

// {

// Now just get the associated text

// String temp = xpp.nextText();

// Do something with text

// Log.e("MyTag","carParkIdentity is " + temp);

// car.setcarParkIdentity(temp);

// }

// else

// Check which Tag we have

// if (xpp.getName().equalsIgnoreCase("carParkOccupancy"))

// {

// Now just get the associated text

// String temp = xpp.nextText();

// Do something with text

// Log.e("MyTag","carParkOccupancy is " + temp);

// car.setcarParkOccupancy(temp);

// }

// else

// Check which Tag we have

// if (xpp.getName().equalsIgnoreCase("carParkStatus"))

// {

// Now just get the associated text

// String temp = xpp.nextText();

// Do something with text

// Log.e("MyTag","carParkStatus is " + temp);

/// car.setcarParkStatus(temp);

// }

// else

// Check which Tag we have

// if (xpp.getName().equalsIgnoreCase("occupiedSpaces"))

// {

// Now just get the associated text

// String temp = xpp.nextText();

// Do something with text

// Log.e("MyTag","OccupiedSpaces is " + temp);

// car.setoccupiedSpaces(temp);

// }

// else

// Check which Tag we have

// if (xpp.getName().equalsIgnoreCase("totalCapacity"))

// {

// Now just get the associated text

// String temp = xpp.nextText();

// Do something with text

// Log.e("MyTag","TotalCapacity is " + temp);

// car.settotalCapacity(temp);

// }

//}

// if(eventType == XmlPullParser.END\_TAG)

// {

// if (xpp.getName().equalsIgnoreCase("situation"))

// {

// Log.e("MyTag","situation is " + car.toString());

/// alist.add(car);

// }

// else

// if (xpp.getName().equalsIgnoreCase("situationRecord"))

// {

// int size;

/// size = alist.size();

// Log.e("MyTag","situationRecord size is " + size);

// }

// }

// Get the next event

eventType = xpp.next();

}

}

// Get the data from the RSS stream as a string

// Do some processing of the data to get the individual parts of the XML stream

// At some point put this processing into a separate thread of execution

// Display the string in the TextView object just to demonstrate this capability

// This will need to be removed at some point

//response.setText(result);

//}

**catch**(XmlPullParserException ae1)

{

Log.*e*("MyTag","Parsing error" + ae1.toString());

}

**catch**(IOException ae)

{

// Handle error

response.setText("Error");

// Add error info to log for diagnostics

errorText.setText(ae.toString());

Log.*e*("MyTag","IO error during parsing");

}

Log.*e*("MyTag","End document");

**return** alist;

}

// Method to handle the reading of the data from the XML stream

**private** **static** String sourceListingString(String urlString)**throws** IOException

{

String result = "";

InputStream anInStream = **null**;

**int** response = -1;

URL url = **new** URL(urlString);

URLConnection conn = url.openConnection();

// Check that the connection can be opened

**if** (!(conn **instanceof** HttpURLConnection))

**throw** **new** IOException("Not an HTTP connection");

**try**

{

// Open connection

HttpURLConnection httpConn = (HttpURLConnection) conn;

httpConn.setAllowUserInteraction(**false**);

httpConn.setInstanceFollowRedirects(**true**);

httpConn.setRequestMethod("GET");

httpConn.connect();

response = httpConn.getResponseCode();

// Check that connection is Ok

**if** (response == HttpURLConnection.***HTTP\_OK***)

{

// Connection is Ok so open a reader

anInStream = httpConn.getInputStream();

InputStreamReader in= **new** InputStreamReader(anInStream);

BufferedReader bin= **new** BufferedReader(in);

// Read in the data from the XML stream

bin.readLine(); // Throw away the header

String line = **new** String();

**while** (( (line = bin.readLine())) != **null**)

{

result = result + "\n" + line;

}

}

}

**catch** (Exception ex)

{

**throw** **new** IOException("Error connecting");

}

// Return result as a string for further processing

**return** result;

} // End of sourceListingString

} // End of Activity class