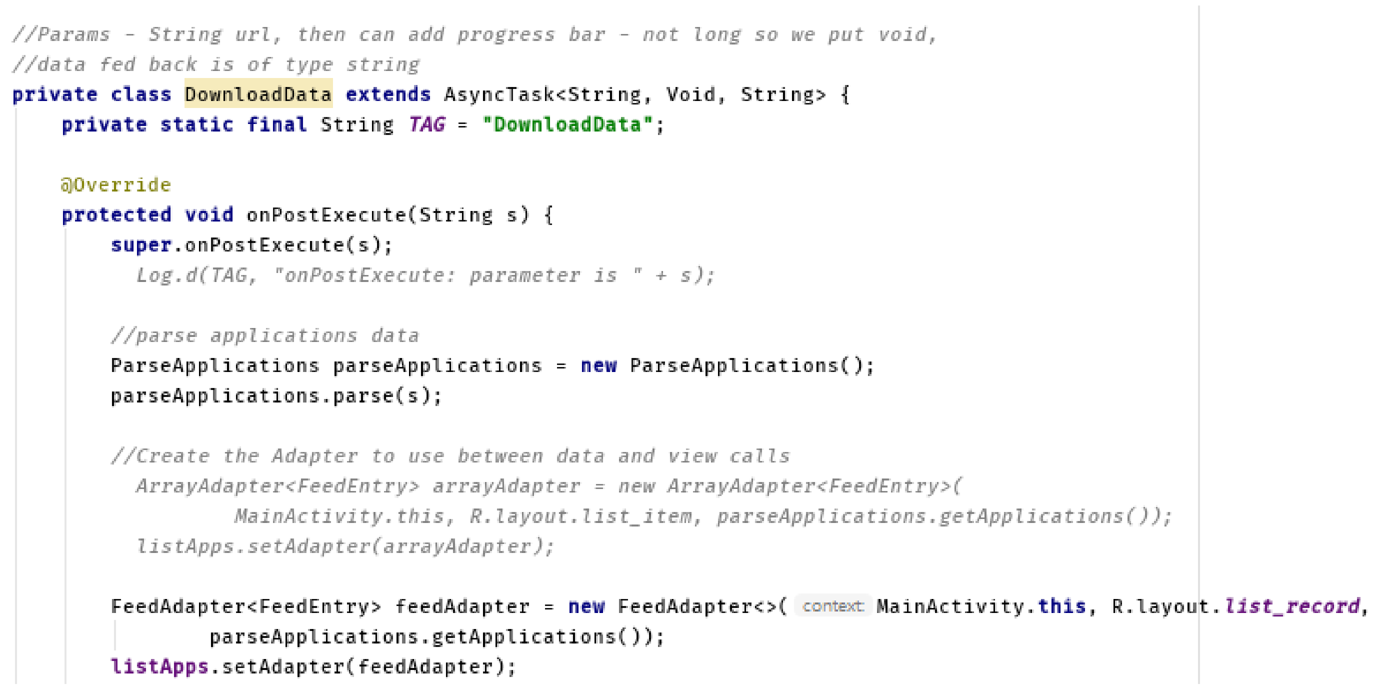
Top 10 Apple App Downloads

Covered:

- Downloading data over the internet
- Running tasks in the background using an AsyncTask
- Why avoid AsyncTask in Kotlin
- Apple RSS Feed – parsing XML to extract data from a downloaded feed
- The ListVIew widget for displaying a scrollable list of items
- Adapters for providing data to ListViews and other objects. And creation of own Custom Adapter
- Context: what and why we need them
- Creating a menu for app

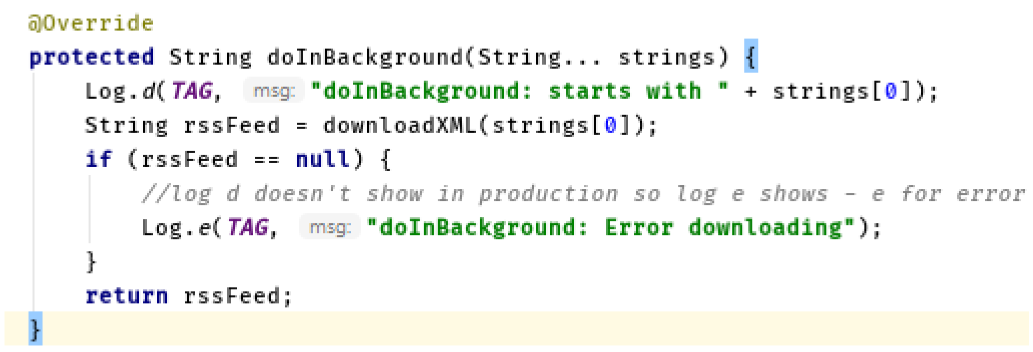
AsyncTask



```
//Params - String url, then can add progress bar - not long so we put void,
//data fed back is of type string
private class DownloadData extends AsyncTask<String, Void, String> {
    private static final String TAG = "DownloadData";

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
          Log.d(TAG, "onPostExecute: parameter is " + s);

        //parse applications data
        ParseApplications parseApplications = new ParseApplications();
        parseApplications.parse(s);

        //Create the Adapter to use between data and view calls
          ArrayAdapter<FeedEntry> arrayAdapter = new ArrayAdapter<FeedEntry>(
                  MainActivity.this, R.layout.list_item, parseApplications.getApplications());
          listApps.setAdapter(arrayAdapter);

        FeedAdapter<FeedEntry> feedAdapter = new FeedAdapter<>( context: MainActivity.this, R.layout.list_record,
                parseApplications.getApplications());
        listApps.setAdapter(feedAdapter);
```

Figure 1 - DownloadData AsyncTask

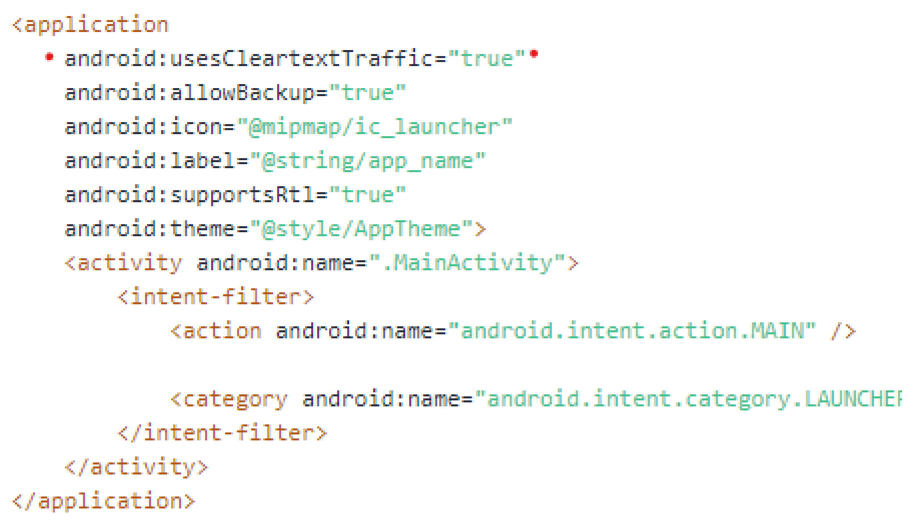
Ctrl-o to get overrides, onPostExecute runs on the main UI thread once background processes are completed. doInBackground is the main method which does the processing on the other thread (no on the UI thread)



```java
@Override
protected String doInBackground(String... strings) {
    Log.d(TAG, msg: "doInBackground: starts with " + strings[0]);
    String rssFeed = downloadXML(strings[0]);
    if (rssFeed == null) {
        //log d doesn't show in production so log e shows - e for error
        Log.e(TAG, msg: "doInBackground: Error downloading");
    }
    return rssFeed;
}
```

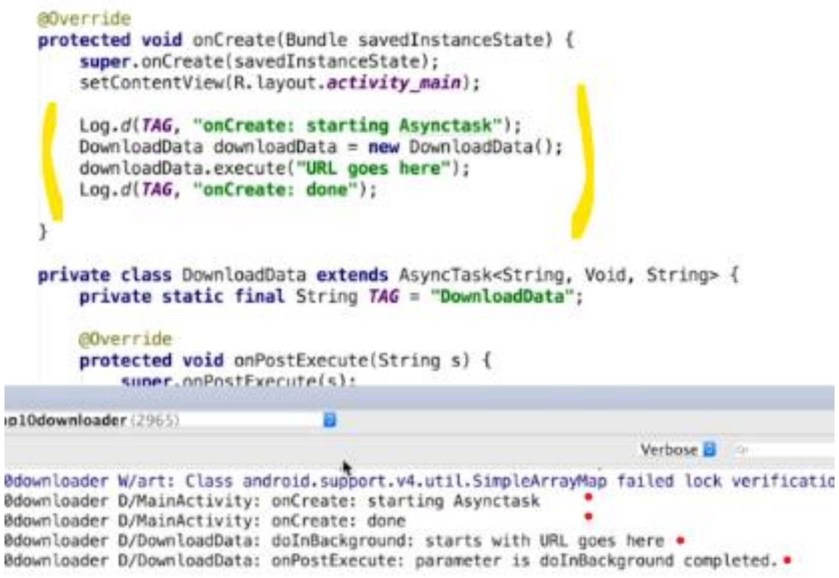
The ellipses: Variable length argument lists were introduced in Java 5 and allow you to provide several values of the same type. When use ellipses as parameter the values get passed into the method as an array – array of strings in this case. Only call method with single parameter i.e. first in array [0]. Log.e for if null – error downloading. Pulls method downloadXML – shown later.

Figure 2 - doInBackground inside AsyncTask



```xml
<application
    android:usesCleartextTraffic="true"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHEI
        </intent-filter>
    </activity>
</application>
```

Figure 3 - Add Clear Text Traffic



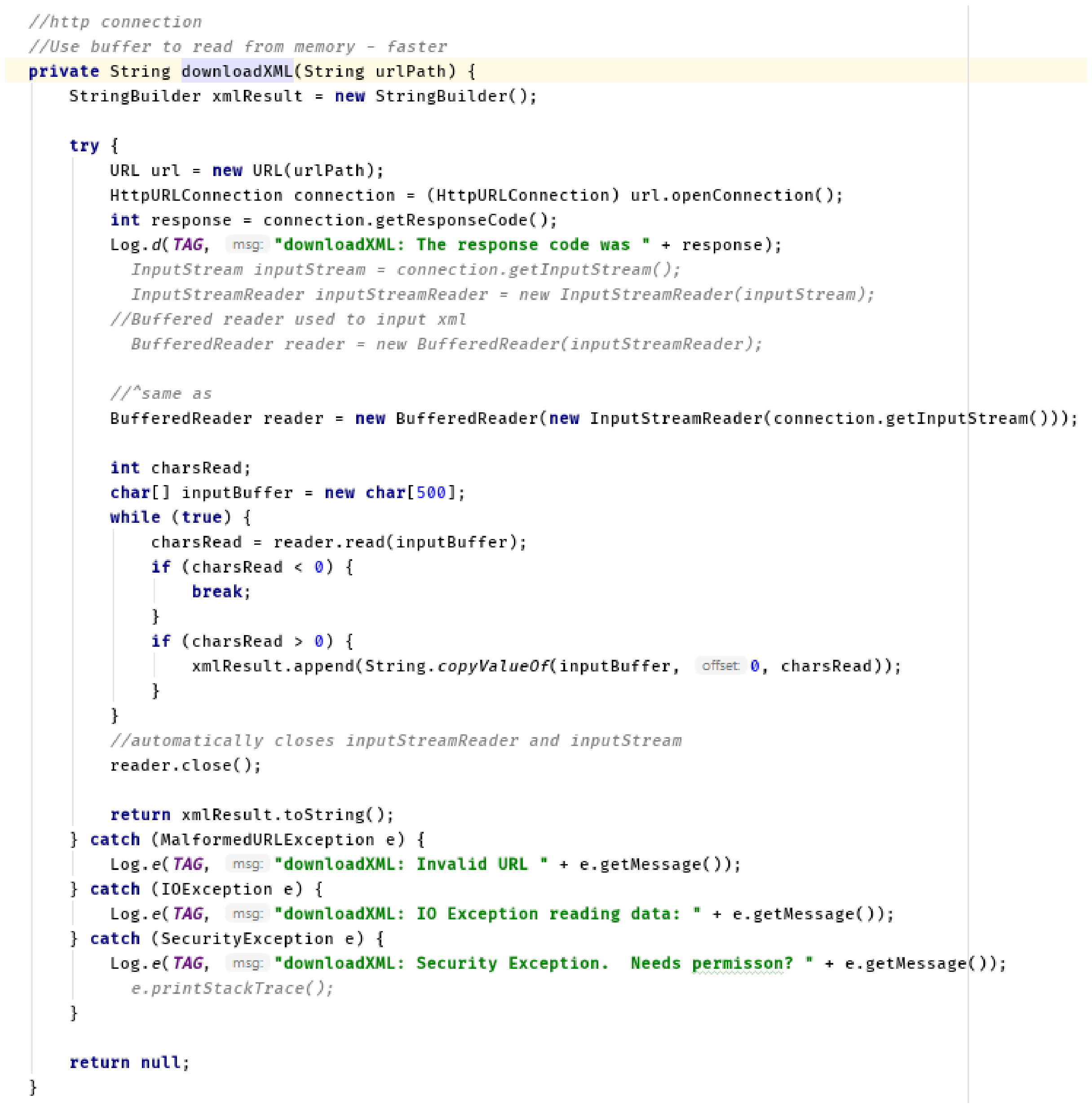
```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.d(TAG, "onCreate: starting Asynctask");
    DownloadData downloadData = new DownloadData();
    downloadData.execute("URL goes here");
    Log.d(TAG, "onCreate: done");
}

private class DownloadData extends AsyncTask<String, Void, String> {
    private static final String TAG = "DownloadData";

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
```

```
p10downloader (2965)
Verbose
@downloader W/art: Class android.support.v4.util.SimpleArrayMap failed lock verificatio
@downloader D/MainActivity: onCreate: starting Asynctask
@downloader D/MainActivity: onCreate: done
@downloader D/DownloadData: doInBackground: starts with URL goes here
@downloader D/DownloadData: onPostExecute: parameter is doInBackground completed.
```

Figure 4 - onCreate test

OnCreate does not wait for doInBackground, it completes in this case before background processes starts. Asynchronous – The doInBackground performs task on separate thread. When finishes returns a string due to specified as the third parameter in DownloadData. Then automatically calls the onPostExecute method, with parameter string – the return value from doInBackground.

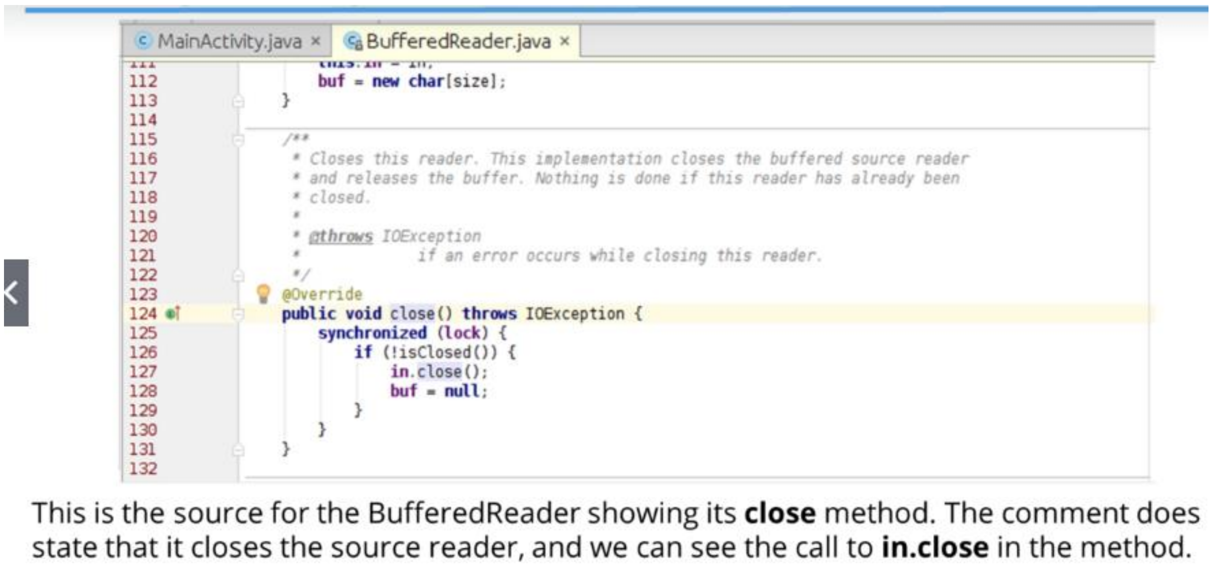
Download Data



```java
//http connection
//Use buffer to read from memory - faster
private String downloadXML(String urlPath) {
    StringBuilder xmlResult = new StringBuilder();

    try {
        URL url = new URL(urlPath);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        int response = connection.getResponseCode();
        Log.d(TAG,  msg: "downloadXML: The response code was " + response);
          InputStream inputStream = connection.getInputStream();
          InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
        //Buffered reader used to input xml
          BufferedReader reader = new BufferedReader(inputStreamReader);

        //^same as
        BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));

        int charsRead;
        char[] inputBuffer = new char[500];
        while (true) {
            charsRead = reader.read(inputBuffer);
            if (charsRead < 0) {
                break;
            }
            if (charsRead > 0) {
                xmlResult.append(String.copyValueOf(inputBuffer,  offset: 0, charsRead));
            }
        }
        //automatically closes inputStreamReader and inputStream
        reader.close();

        return xmlResult.toString();
    } catch (MalformedURLException e) {
        Log.e(TAG,  msg: "downloadXML: Invalid URL " + e.getMessage());
    } catch (IOException e) {
        Log.e(TAG,  msg: "downloadXML: IO Exception reading data: " + e.getMessage());
    } catch (SecurityException e) {
        Log.e(TAG,  msg: "downloadXML: Security Exception.  Needs permisson? " + e.getMessage());
          e.printStackTrace();
    }

    return null;
}
```

Figure 5 - Download XML

Uses a Buffered Reader - Buffers data coming in from stream. So, instead of repeatedly accessing the hard drive or network, a block of data is read into buffer in memory, so the program can read from the buffer. Closing the buffered reader will also close the inputStreamReader which will close the inputStream automatically. Uses String builder as more efficient that concatenating strings. Uses a try block to allow for catch of errors. If expected larger download can increase the in charsRead to a greater number. While loops keeps going round until the end of inputaStream reached, if charsRead < 0 then reached end of data and can break, else charsRead variable will hold count of read data.


MainActivity.java × BufferedReader.java ×

```java
        this.in = in;
        buf = new char[size];
    }

    /**
     * Closes this reader. This implementation closes the buffered source reader
     * and releases the buffer. Nothing is done if this reader has already been
     * closed.
     *
     * @throws IOException
     *             if an error occurs while closing this reader.
     */
    @Override
    public void close() throws IOException {
        synchronized (lock) {
            if (!isClosed()) {
                in.close();
                buf = null;
            }
        }
    }

```

This is the source for the BufferedReader showing its **close** method. The comment does state that it closes the source reader, and we can see the call to **in.close** in the method.

Figure 6 - Buffered Reader

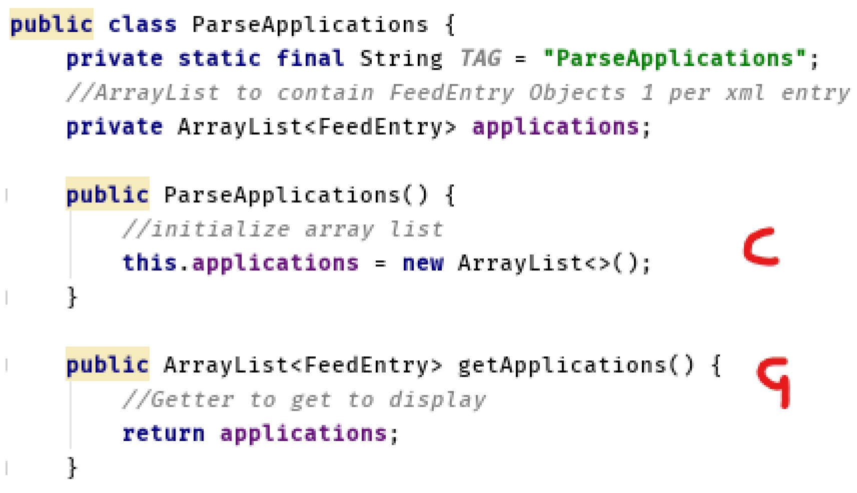
```xml
<uses-permission android:name="android.permission.INTERNET"/>
```

Figure 7 - Add Internet Permission in Manifest

```java
public class FeedEntry {

    private String name;
    private String artist;
    private String releaseDate;
    private String summary;
    private String imageURL;

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getArtist() { return artist; }

    public void setArtist(String artist) { this.artist = artist; }

    public String getReleaseDate() { return releaseDate; }

    public void setReleaseDate(String releaseDate) { this.releaseDate = releaseDate; }

    public String getSummary() { return summary; }

    public void setSummary(String summary) { this.summary = summary; }

    public String getImageURL() { return imageURL; }

    public void setImageURL(String imageURL) { this.imageURL = imageURL; }
```

Figure 8 - Separate Class, Feed Entry

Getter and Setters for the entries to be fed in.



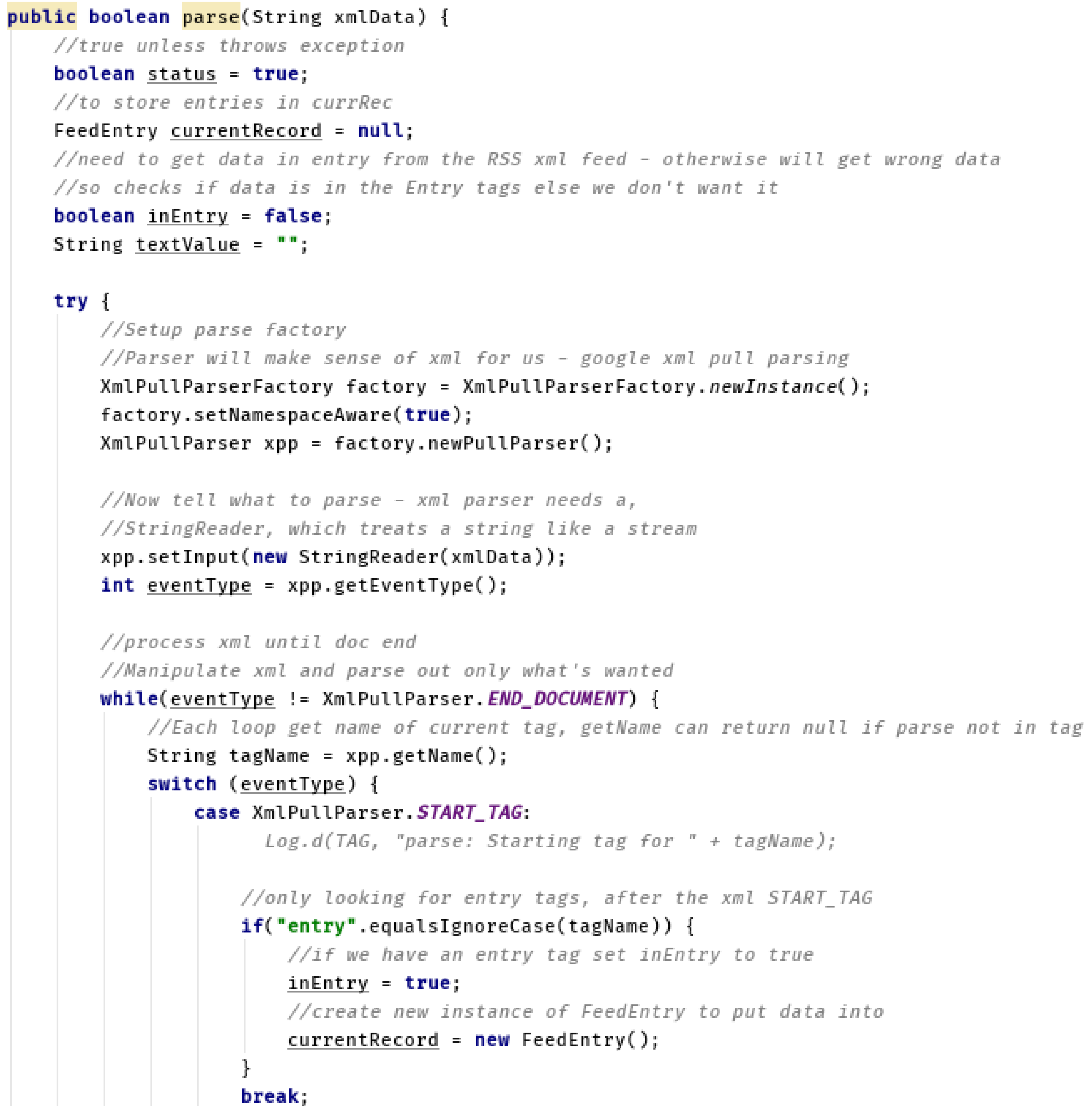
```java
public class ParseApplications {
    private static final String TAG = "ParseApplications";
    //ArrayList to contain FeedEntry Objects 1 per xml entry
    private ArrayList<FeedEntry> applications;

    public ParseApplications() {
        //initialize array list
        this.applications = new ArrayList<>();
    }

    public ArrayList<FeedEntry> getApplications() {
        //Getter to get to display
        return applications;
    }
```

c

9

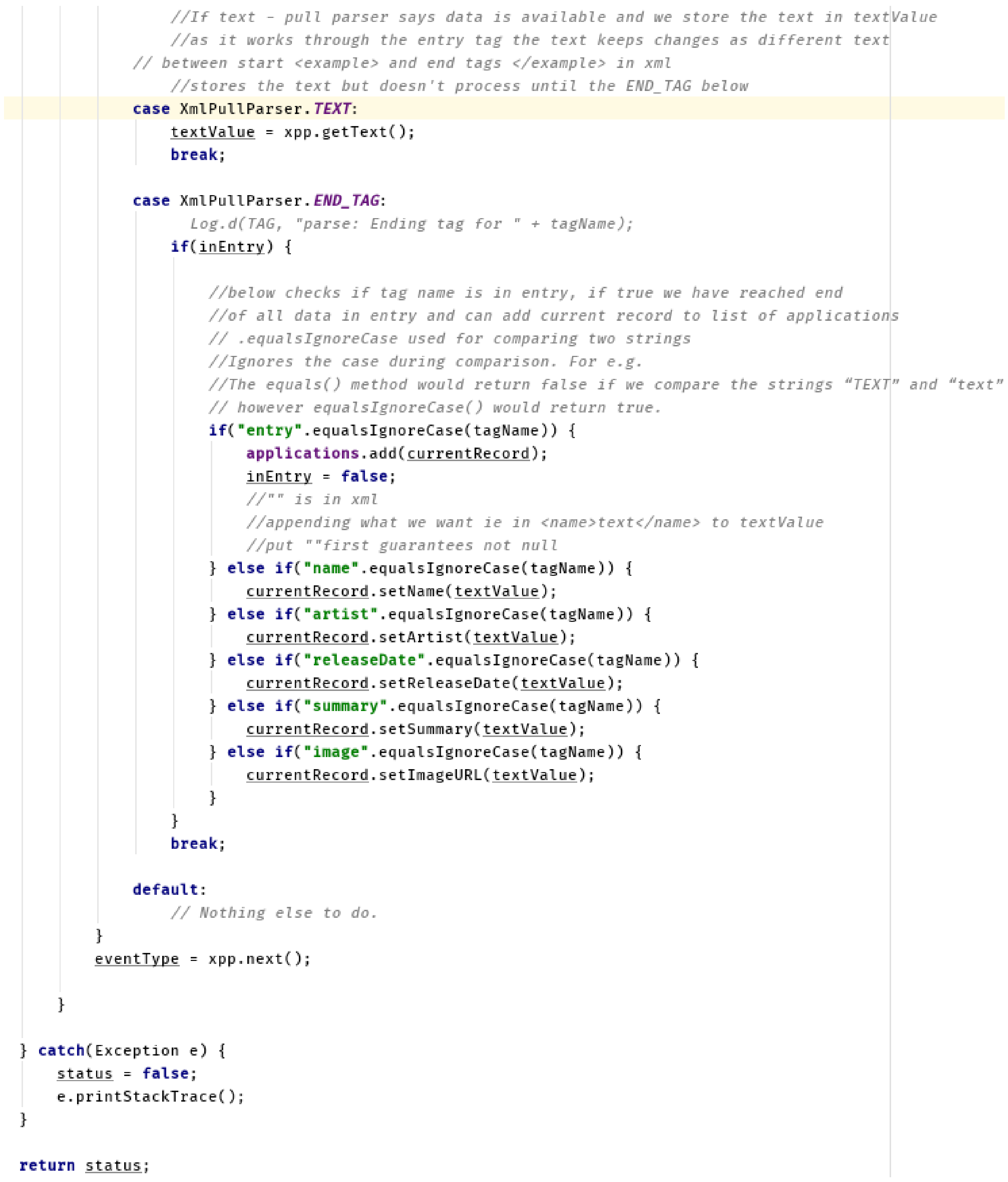
Figure 9 - Separate Class, Parse Applications

Constructor and Getter.



```java
public boolean parse(String xmlData) {
    //true unless throws exception
    boolean status = true;
    //to store entries in currRec
    FeedEntry currentRecord = null;
    //need to get data in entry from the RSS xml feed - otherwise will get wrong data
    //so checks if data is in the Entry tags else we don't want it
    boolean inEntry = false;
    String textValue = "";

    try {
        //Setup parse factory
        //Parser will make sense of xml for us - google xml pull parsing
        XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
        factory.setNamespaceAware(true);
        XmlPullParser xpp = factory.newPullParser();

        //Now tell what to parse - xml parser needs a,
        //StringReader, which treats a string like a stream
        xpp.setInput(new StringReader(xmlData));
        int eventType = xpp.getEventType();

        //process xml until doc end
        //Manipulate xml and parse out only what's wanted
        while(eventType != XmlPullParser.END_DOCUMENT) {
            //Each loop get name of current tag, getName can return null if parse not in tag
            String tagName = xpp.getName();
            switch (eventType) {
                case XmlPullParser.START_TAG:
                      Log.d(TAG, "parse: Starting tag for " + tagName);

                    //only looking for entry tags, after the xml START_TAG
                    if("entry".equalsIgnoreCase(tagName)) {
                        //if we have an entry tag set inEntry to true
                        inEntry = true;
                        //create new instance of FeedEntry to put data into
                        currentRecord = new FeedEntry();
                    }
                    break;
```

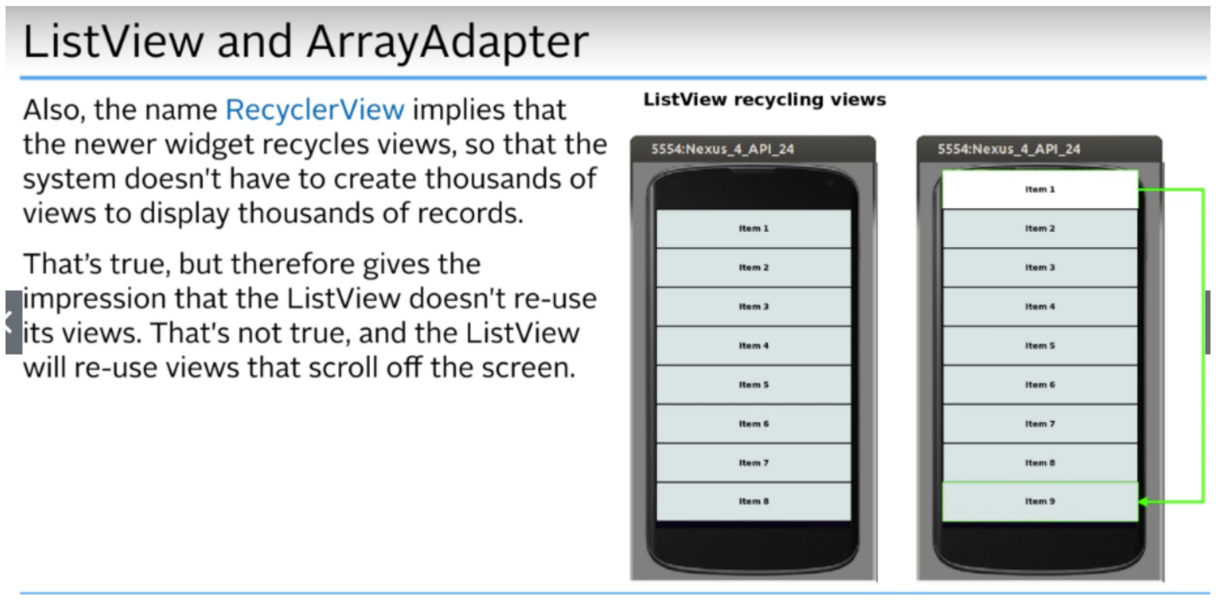
Figure 10 - Boolean Parse XML Data 1



```java
                //If text - pull parser says data is available and we store the text in textValue
                //as it works through the entry tag the text keeps changes as different text
            // between start <example> and end tags </example> in xml
                //stores the text but doesn't process until the END_TAG below
            case XmlPullParser.TEXT:
                textValue = xpp.getText();
                break;

            case XmlPullParser.END_TAG:
                  Log.d(TAG, "parse: Ending tag for " + tagName);
                if(inEntry) {

                    //below checks if tag name is in entry, if true we have reached end
                    //of all data in entry and can add current record to list of applications
                    // .equalsIgnoreCase used for comparing two strings
                    //Ignores the case during comparison. For e.g.
                    //The equals() method would return false if we compare the strings "TEXT" and "text"
                    // however equalsIgnoreCase() would return true.
                    if("entry".equalsIgnoreCase(tagName)) {
                        applications.add(currentRecord);
                        inEntry = false;
                        //"" is in xml
                        //appending what we want ie in <name>text</name> to textValue
                        //put ""first guarantees not null
                    } else if("name".equalsIgnoreCase(tagName)) {
                        currentRecord.setName(textValue);
                    } else if("artist".equalsIgnoreCase(tagName)) {
                        currentRecord.setArtist(textValue);
                    } else if("releaseDate".equalsIgnoreCase(tagName)) {
                        currentRecord.setReleaseDate(textValue);
                    } else if("summary".equalsIgnoreCase(tagName)) {
                        currentRecord.setSummary(textValue);
                    } else if("image".equalsIgnoreCase(tagName)) {
                        currentRecord.setImageURL(textValue);
                    }
                }
                break;

            default:
                // Nothing else to do.
        }
        eventType = xpp.next();

    }

} catch(Exception e) {
    status = false;
    e.printStackTrace();
}

return status;
```

Figure 11 - Boolean Parse XML Data 2

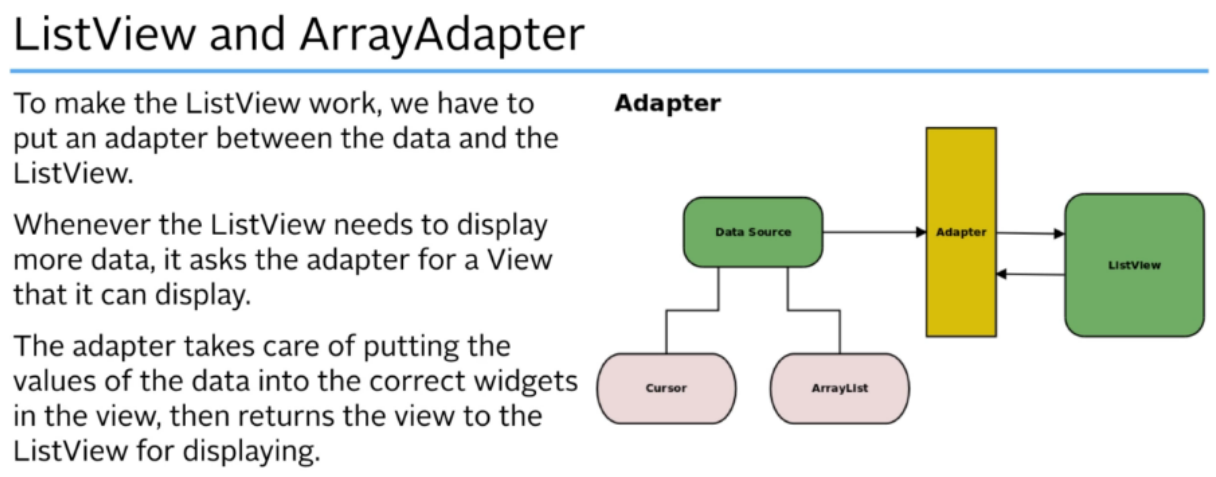
List View and Array Adapter


ListView and ArrayAdapter

Also, the name RecyclerView implies that the newer widget recycles views, so that the system doesn't have to create thousands of views to display thousands of records.

That's true, but therefore gives the impression that the ListView doesn't re-use its views. That's not true, and the ListView will re-use views that scroll off the screen.

Figure 12 - List View


ListView and ArrayAdapter

To make the ListView work, we have to put an adapter between the data and the ListView.

Whenever the ListView needs to display more data, it asks the adapter for a View that it can display.

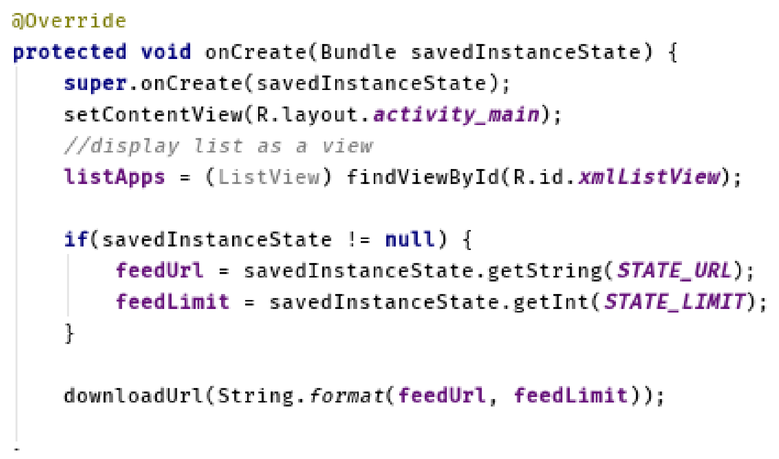
The adapter takes care of putting the values of the data into the correct widgets in the view, then returns the view to the ListView for displaying.

Figure 13 - Array Adapter

If the adapter receives a View from the ListView, it'll replace any data in it with the data for the current record, the send it back to the ListView so it can be displayed. If the Adapter receives a View from ListVoew, replaces data in it with the data for current record, sends it back to ListView to be displayed. Data soruce could be a Cursor from a database, or an ArrayList – which is what our data has been parsed into.

* In computer science, a **database cursor** is a control structure that enables traversal over the records in a **database**. **Cursors** facilitate subsequent processing in conjunction with the traversal, such as retrieval, addition and removal of **database** records.

ArrayAdapter is very basic and can put data into a single TextView widget. You also have little control over data presentation, the array adapter just uses the object's toString method and puts returned string into a TextView. So can override and create your own toString method in the Application class to handle this.

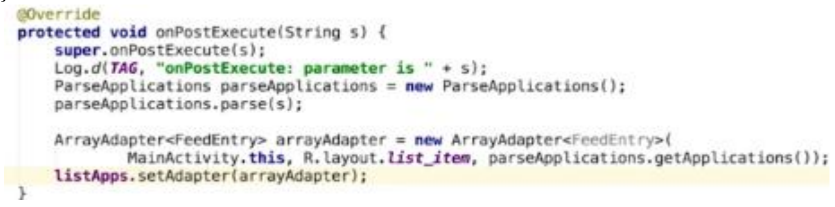


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //display list as a view
    listApps = (ListView) findViewById(R.id.xmlListView);

    if(savedInstanceState != null) {
        feedUrl = savedInstanceState.getString(STATE_URL);
        feedLimit = savedInstanceState.getInt(STATE_LIMIT);
    }

    downloadUrl(String.format(feedUrl, feedLimit));
```

Figure 14 - onCreate ListView, State & URL

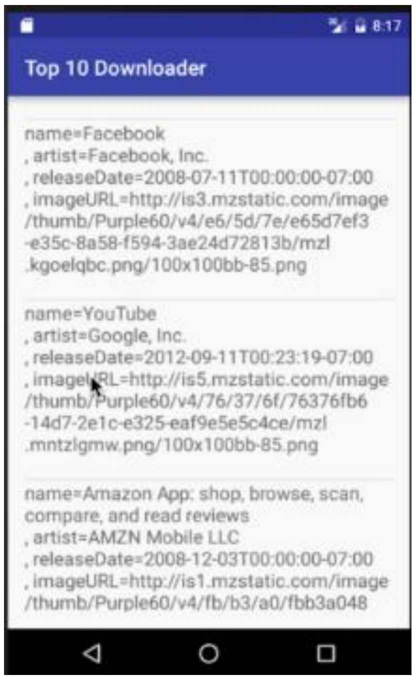
Call ListView in onCreate, using findViewById. Now need to connect data to ListView using an adapter – Data is stored in an array list in the parseApplications class, get by calling getApplications method.

```
public ArrayList<FeedEntry> getApplications() {
    //Getter to get to display
    return applications;
}
```



```
@Override
protected void onPostExecute(String s) {
    super.onPostExecute(s);
    Log.d(TAG, "onPostExecute: parameter is " + s);
    ParseApplications parseApplications = new ParseApplications();
    parseApplications.parse(s);

    ArrayAdapter<FeedEntry> arrayAdapter = new ArrayAdapter<FeedEntry>(
            MainActivity.this, R.layout.list_item, parseApplications.getApplications());
    listApps.setAdapter(arrayAdapter);
}
```

Figure 15 - Default Adapter Setup



8:17

Top 10 Downloader

name=Facebook
, artist=Facebook, Inc.
, releaseDate=2008-07-11T00:00:00-07:00
, imageURL=http://is3.mzstatic.com/image
/thumb/Purple60/v4/e6/5d/7e/e65d7ef3
-e35c-8a58-f594-3ae24d72813b/mzl
.kgoelqbc.png/100x100bb-85.png

name=YouTube
, artist=Google, Inc.
, releaseDate=2012-09-11T00:23:19-07:00
, imageURL=http://is5.mzstatic.com/image
/thumb/Purple60/v4/76/37/6f/76376fb6
-14d7-2e1c-e325-eaf9e5e5c4ce/mzl
.mntzlgmw.png/100x100bb-85.png

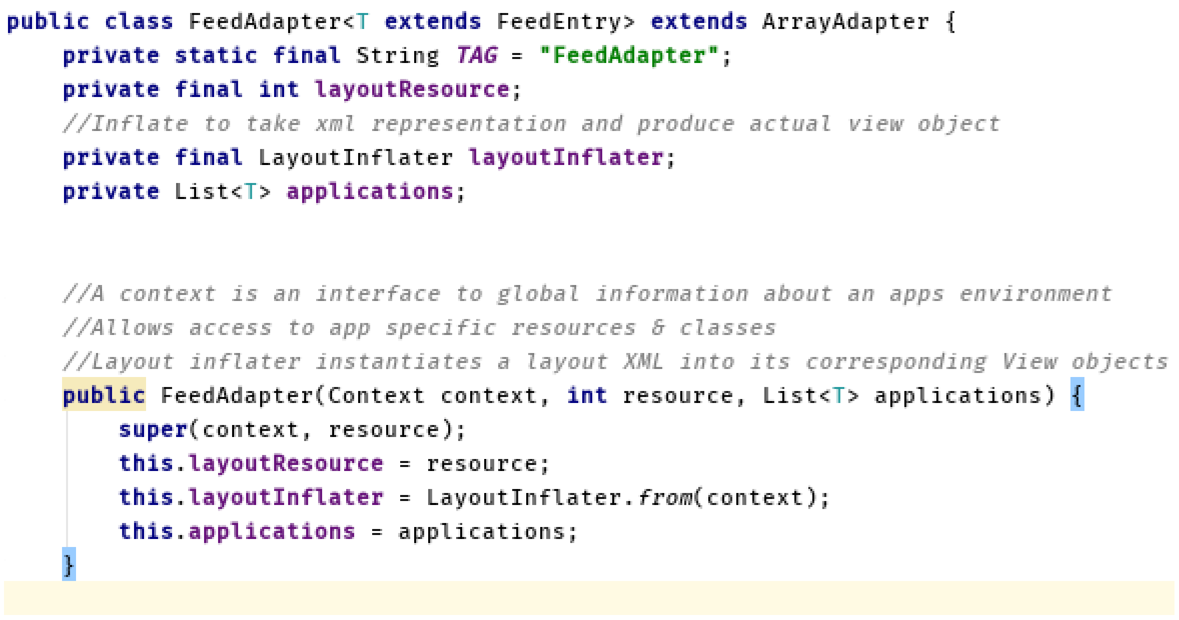
name=Amazon App: shop, browse, scan,
compare, and read reviews
, artist=AMZN Mobile LLC
, releaseDate=2008-12-03T00:00:00-07:00
, imageURL=http://is1.mzstatic.com/image
/thumb/Purple60/v4/fb/b3/a0/fbb3a048

Figure 16 - Display with default ArrayAdapter<>

You must override the toString method, or will return the package name of your project.

Custom Adapter Creation

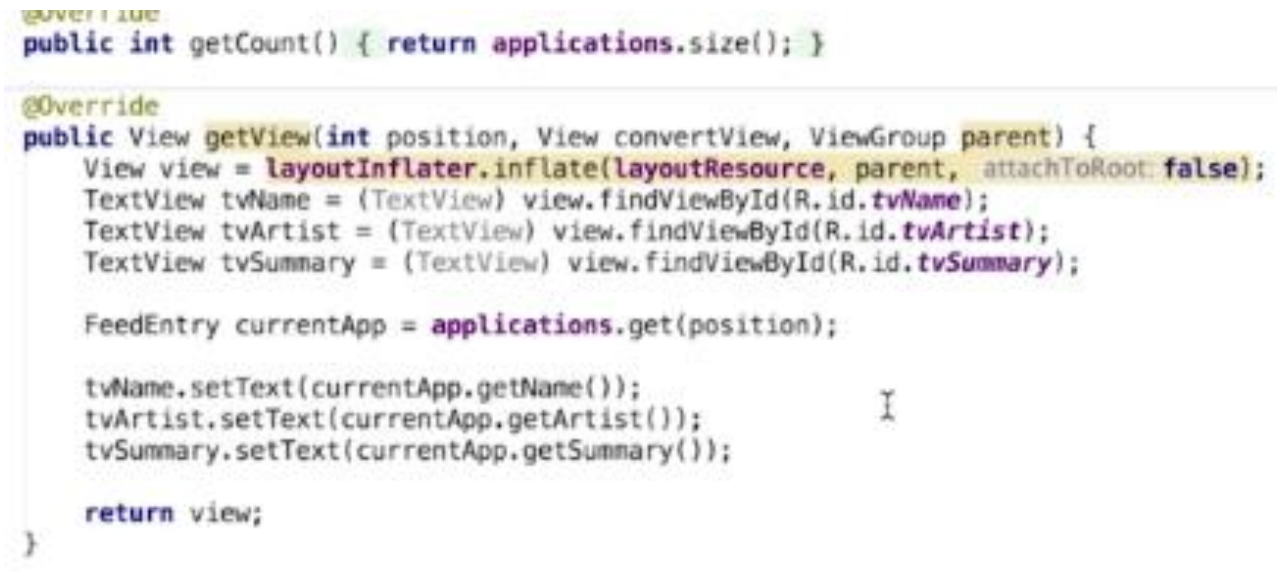
New Class FeedAdapter extends ArrayAdapter



```java
public class FeedAdapter<T extends FeedEntry> extends ArrayAdapter {
    private static final String TAG = "FeedAdapter";
    private final int layoutResource;
    //Inflate to take xml representation and produce actual view object
    private final LayoutInflater layoutInflater;
    private List<T> applications;


    //A context is an interface to global information about an apps environment
    //Allows access to app specific resources & classes
    //Layout inflater instantiates a layout XML into its corresponding View objects
    public FeedAdapter(Context context, int resource, List<T> applications) {
        super(context, resource);
        this.layoutResource = resource;
        this.layoutInflater = LayoutInflater.from(context);
        this.applications = applications;
    }
```

Figure 17 - Define & Construct FeedAdapter

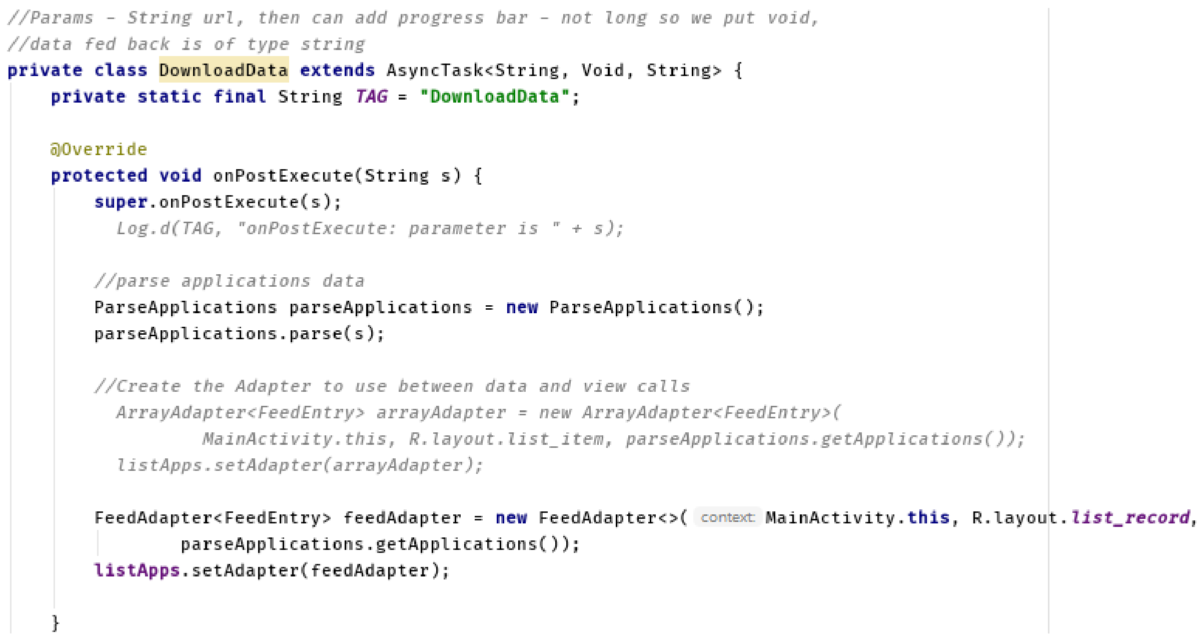
Get a Layout Inflator from Context and get Context from Constructor. Context is an interface to global information about an application environment. Allows access to specific resources and classes, as well as up-calls for application-level operations such as learning activities, broadcasting and receiving intents, etc. this. Refers to the current instance of the class –referring to itself.



```java
@Override
public int getCount() { return applications.size(); }

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View view = layoutInflater.inflate(layoutResource, parent, attachToRoot: false);
    TextView tvName = (TextView) view.findViewById(R.id.tvName);
    TextView tvArtist = (TextView) view.findViewById(R.id.tvArtist);
    TextView tvSummary = (TextView) view.findViewById(R.id.tvSummary);

    FeedEntry currentApp = applications.get(position);

    tvName.setText(currentApp.getName());
    tvArtist.setText(currentApp.getArtist());
    tvSummary.setText(currentApp.getSummary());

    return view;
}
```

Figure 18 – Get Count, View & Holder

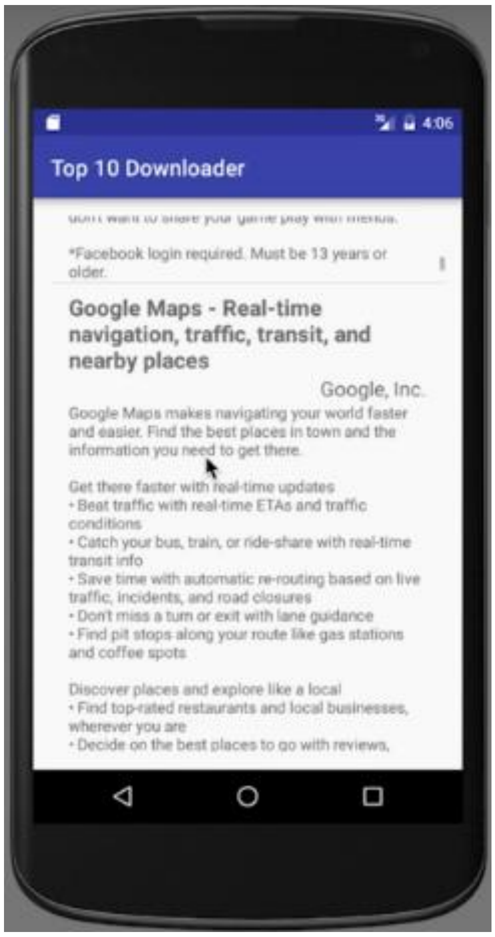
Override the two methods.



```
//Params - String url, then can add progress bar - not long so we put void,
//data fed back is of type string
private class DownloadData extends AsyncTask<String, Void, String> {
    private static final String TAG = "DownloadData";

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
          Log.d(TAG, "onPostExecute: parameter is " + s);

        //parse applications data
        ParseApplications parseApplications = new ParseApplications();
        parseApplications.parse(s);

        //Create the Adapter to use between data and view calls
          ArrayAdapter<FeedEntry> arrayAdapter = new ArrayAdapter<FeedEntry>(
                  MainActivity.this, R.layout.list_item, parseApplications.getApplications());
          listApps.setAdapter(arrayAdapter);

        FeedAdapter<FeedEntry> feedAdapter = new FeedAdapter<>( context: MainActivity.this, R.layout.list_record,
                parseApplications.getApplications());
        listApps.setAdapter(feedAdapter);

    }
```

Figure 19 - To use new Feed Adapter

Download Data in the Main Activity – Change onPostExecute to use the Feed Adapter.



4:06

Top 10 Downloader

*Facebook login required. Must be 13 years or older.

Google Maps - Real-time navigation, traffic, transit, and nearby places

Google, Inc.

Google Maps makes navigating your world faster and easier. Find the best places in town and the information you need to get there.

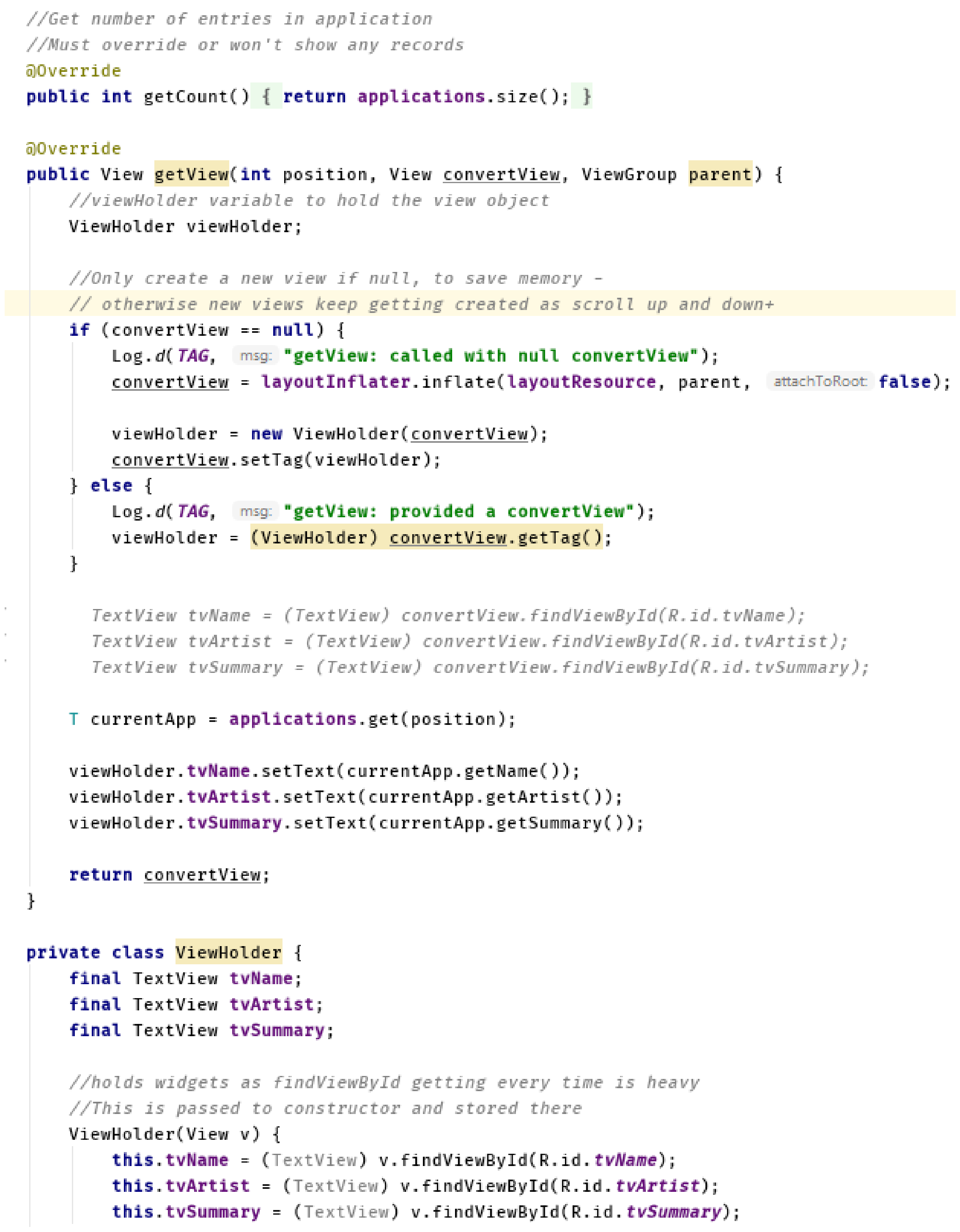
Get there faster with real-time updates
• Beat traffic with real-time ETAs and traffic conditions
• Catch your bus, train, or ride-share with real-time transit info
• Save time with automatic re-routing based on live traffic, incidents, and road closures
• Don't miss a turn or exit with lane guidance
• Find pit stops along your route like gas stations and coffee spots

Discover places and explore like a local
• Find top-rated restaurants and local businesses, wherever you are
• Decide on the best places to go with reviews,

Figure 20 – Works

Improvements

If you wanted to display a thousand items, it will create a thousand views, being time and memory expensive. FindViewById is slow as it has to scan layout from start each time it's called, checking to see which widget is the wanted one. Furthermore, if you were to scroll up and down the items, it would create new views. Can use Android Profiler to check memory usage.



```
//Get number of entries in application
//Must override or won't show any records
@Override
public int getCount() { return applications.size(); }

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    //viewHolder variable to hold the view object
    ViewHolder viewHolder;

    //Only create a new view if null, to save memory -
    // otherwise new views keep getting created as scroll up and down+
    if (convertView == null) {
        Log.d(TAG,  msg: "getView: called with null convertView");
        convertView = layoutInflater.inflate(layoutResource, parent,  attachToRoot: false);

        viewHolder = new ViewHolder(convertView);
        convertView.setTag(viewHolder);
    } else {
        Log.d(TAG,  msg: "getView: provided a convertView");
        viewHolder = (ViewHolder) convertView.getTag();
    }

      TextView tvName = (TextView) convertView.findViewById(R.id.tvName);
      TextView tvArtist = (TextView) convertView.findViewById(R.id.tvArtist);
      TextView tvSummary = (TextView) convertView.findViewById(R.id.tvSummary);

    T currentApp = applications.get(position);

    viewHolder.tvName.setText(currentApp.getName());
    viewHolder.tvArtist.setText(currentApp.getArtist());
    viewHolder.tvSummary.setText(currentApp.getSummary());

    return convertView;
}

private class ViewHolder {
    final TextView tvName;
    final TextView tvArtist;
    final TextView tvSummary;

    //holds widgets as findViewById getting every time is heavy
    //This is passed to constructor and stored there
    ViewHolder(View v) {
        this.tvName = (TextView) v.findViewById(R.id.tvName);
        this.tvArtist = (TextView) v.findViewById(R.id.tvArtist);
        this.tvSummary = (TextView) v.findViewById(R.id.tvSummary);
```

Figure 21 - Change Feed Adapter Count & View

No need to inflate view each time, as the ListView provides a view when it can. That's why we use the convertView parameter – if the ListView has a view it can use it passes a view to it in the convert view.

Check if null. If not null will get old view to use, by using the get tag method. Tag is an object so we're casting it as a view holder. We know it'll be a view, as it was put there using the setTag. This limits view creation and recycle views

Makes sense to store the widgets as they are not being changed only the view is changed. Can use the View Holder pattern to store, which uses a small class to hold the views. This limits calling to find view by id, which is slow and inefficient. Allows for smoother scrolling.

Adding a Menu

Create a new folder in res, of type menu, which is to be placed as a directory – menu. Now, create a new menu resource file- “feeds_menu”.

Create:

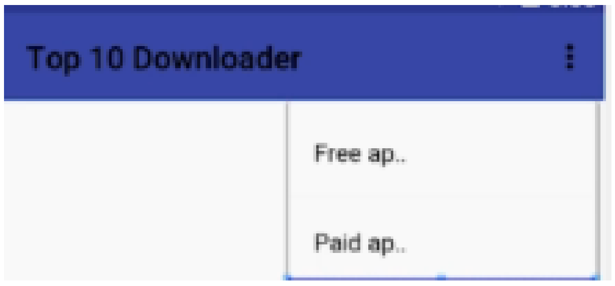


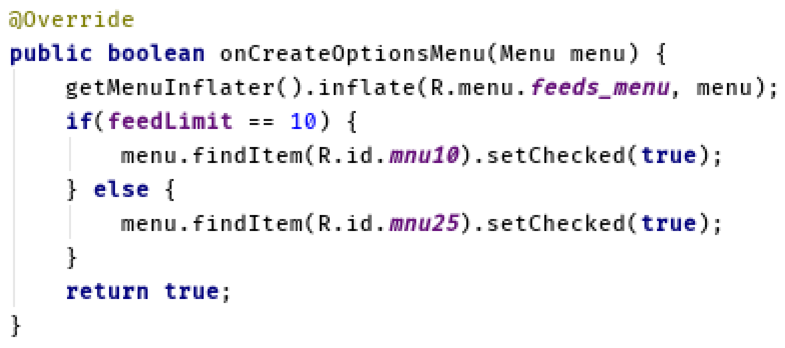
Figure 22 - Example Menu Addition

Now menu has been added – Need to add code in main activity to use what has been created.

Override:

```
@Override
public boolean onCreateOptionsMenu(Menu menu){

}

@Override
public boolean onOptionsItemSelected(MenuItem item){

}
```

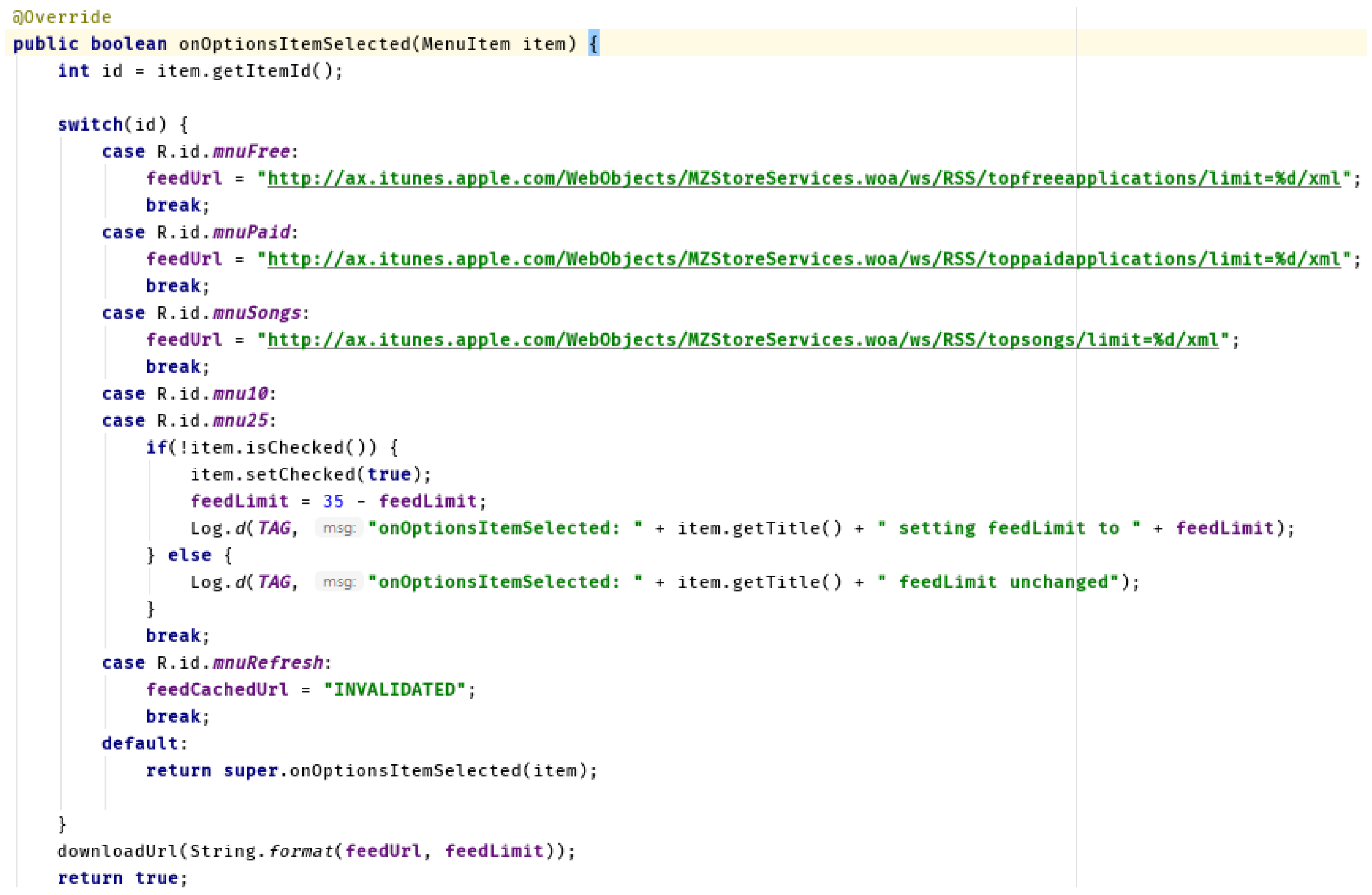
OnCreate Options is called when it’s time to inflate the activities menu; Meaning create the menu objects from the XML file so we’ve got a basic stub (stand in code).



```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.feeds_menu, menu);
    if(feedLimit == 10) {
        menu.findItem(R.id.mnu10).setChecked(true);
    } else {
        menu.findItem(R.id.mnu25).setChecked(true);
    }
    return true;
}
```

Figure 23 - Options On Create Menu

Displays menu – No current functionality.

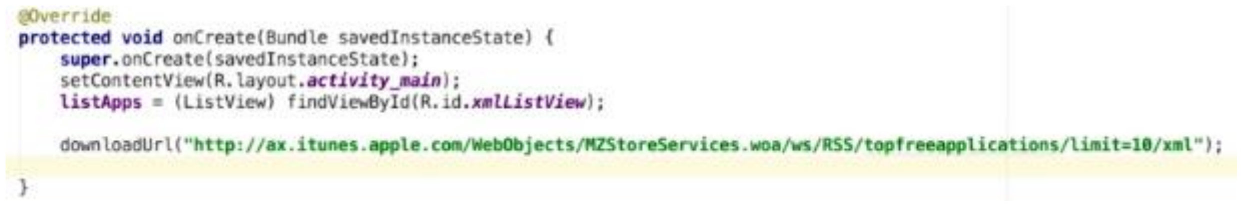


```java
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    switch(id) {
        case R.id.mnuFree:
            feedUrl = "http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/topfreeapplications/limit=%d/xml";
            break;
        case R.id.mnuPaid:
            feedUrl = "http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/toppaidapplications/limit=%d/xml";
            break;
        case R.id.mnuSongs:
            feedUrl = "http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/topsongs/limit=%d/xml";
            break;
        case R.id.mnu10:
        case R.id.mnu25:
            if(!item.isChecked()) {
                item.setChecked(true);
                feedLimit = 35 - feedLimit;
                Log.d(TAG,  msg: "onOptionsItemSelected: " + item.getTitle() + " setting feedLimit to " + feedLimit);
            } else {
                Log.d(TAG,  msg: "onOptionsItemSelected: " + item.getTitle() + " feedLimit unchanged");
            }
            break;
        case R.id.mnuRefresh:
            feedCachedUrl = "INVALIDATED";
            break;
        default:
            return super.onOptionsItemSelected(item);

    }
    downloadUrl(String.format(feedUrl, feedLimit));
    return true;
```

Figure 24 - Options Item Selected

Called whenever an item is selected from that options menu, so when this method is called android passes in the menu item that was selected from the menu. Get ID – to tell which item selected.

Can use a switch statement to set the URL of the feed, Add string to store it.



```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    listApps = (ListView) findViewById(R.id.xmlListView);

    downloadUrl("http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/topfreeapplications/limit=10/xml");

}
```

Figure 25 - Change On Create

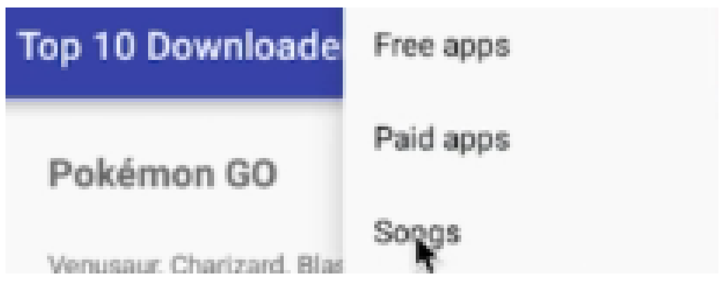


Figure 26 – Functionality should work

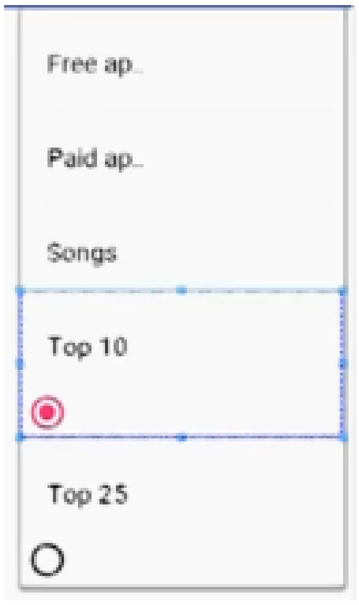
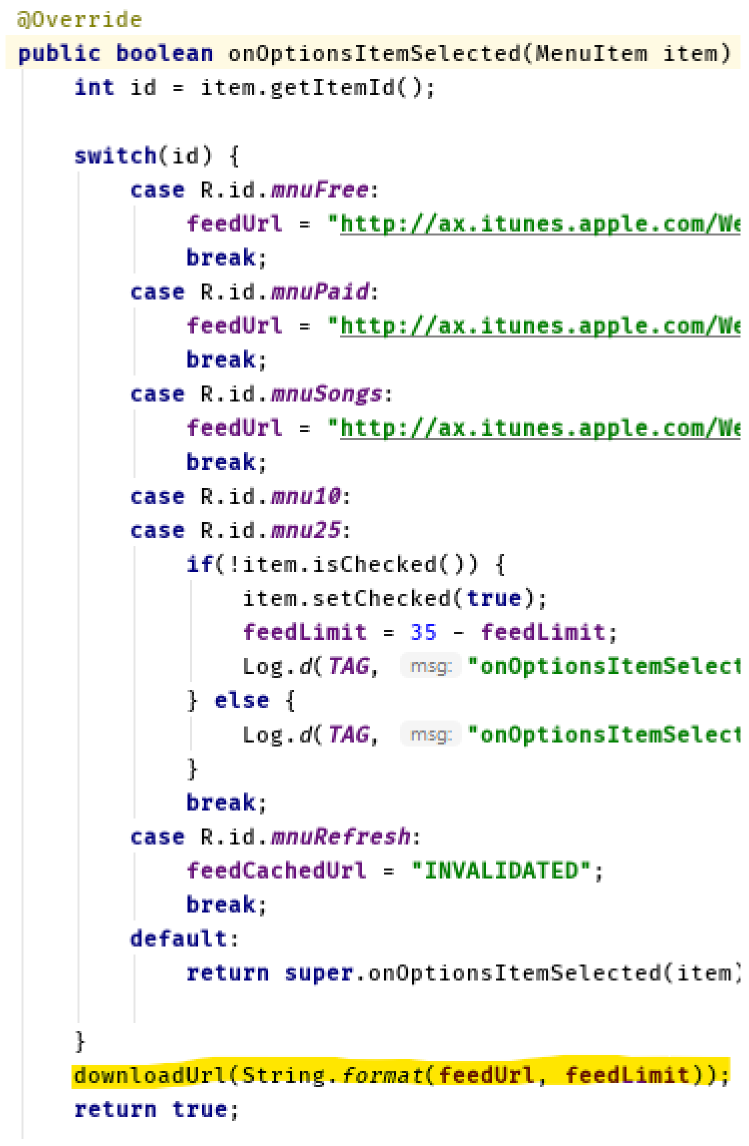


Figure 27 - Add further menu items

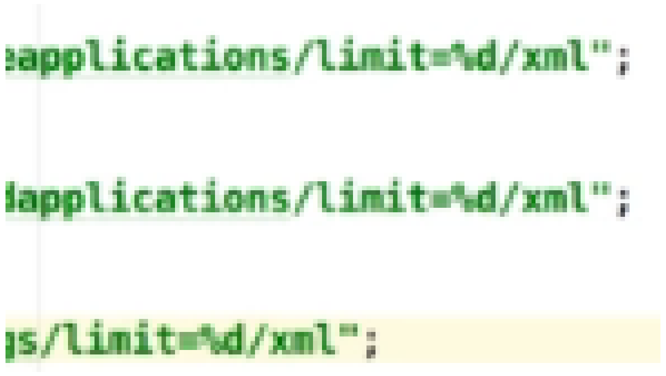
```java
public class MainActivity extends AppCompatActivity {
    private static final String TAG = "MainActivity";
    private ListView listApps;
    private String feedUrl = "http://ax.itunes.apple.com/WebObje
    private int feedLimit = 10;
    private String feedCachedUrl = "INVALIDATED";
    public static final String STATE_URL = "feedUrl";
    public static final String STATE_LIMIT = "feedLimit";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //display list as a view
        listApps = (ListView) findViewById(R.id.xmlListView);

        if(savedInstanceState != null) {
            feedUrl = savedInstanceState.getString(STATE_URL);
            feedLimit = savedInstanceState.getInt(STATE_LIMIT);
        }

        downloadUrl(String.format(feedUrl, feedLimit));

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.feeds_menu, menu);
        if(feedLimit == 10) {
            menu.findItem(R.id.mnu10).setChecked(true);
        } else {
            menu.findItem(R.id.mnu25).setChecked(true);
        }
        return true;
    }
```

Figure 28 - Functionality of Toggle Top 10/25



```
@Override
public boolean onOptionsItemSelected(MenuItem item)
    int id = item.getItemId();

    switch(id) {
        case R.id.mnuFree:
            feedUrl = "http://ax.itunes.apple.com/We
            break;
        case R.id.mnuPaid:
            feedUrl = "http://ax.itunes.apple.com/We
            break;
        case R.id.mnuSongs:
            feedUrl = "http://ax.itunes.apple.com/We
            break;
        case R.id.mnu10:
        case R.id.mnu25:
            if(!item.isChecked()) {
                item.setChecked(true);
                feedLimit = 35 - feedLimit;
                Log.d( TAG,  msg: "onOptionsItemSelect
            } else {
                Log.d( TAG,  msg: "onOptionsItemSelect
            }
            break;
        case R.id.mnuRefresh:
            feedCachedUrl = "INVALIDATED";
            break;
        default:
            return super.onOptionsItemSelected(item)

    }
    downloadUrl(String.format(feedUrl, feedLimit));
    return true;
```

Figure 29 - Change Options Item Selected to download URL, with specified limit. Delete local string variable “feedURL”.



```
eapplications/limit=%d/xml";


dapplications/limit=%d/xml";


gs/limit=%d/xml";
```

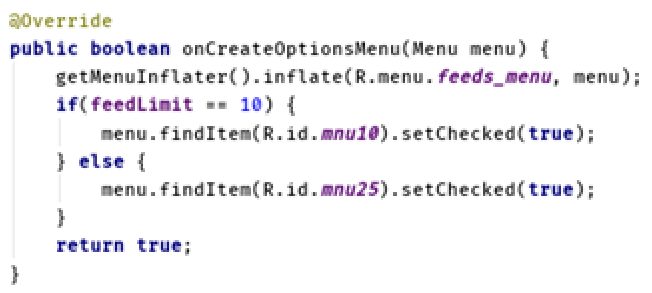
Figure 30 - Change feedURL limit to %d

An issue when running the app, is that it downloads the data again if the same item is selected. This can drain power and memory. Also can incur Roaming Data Charges. Screen rotation will also reset and put user on default Top 10, and get this data once more.

Fix – Storing the Value in the instant state bundle using the on saved instance state method

The code should store the last URL and only download the data again of the URL is changed. Need a way for the user to refresh data manually – Delegate to user. To save performing a download when the device orientation changes, better to cache the downloaded data in the same way that web browsers. That way can allow activity to be destroyed and recreate it when device is rotated. The re-downloading would read the data from the local cache instead of over the internet. However, for this app we will download data again to reduce complexity.

Need to set the correct menu limit once restored feed limit value has occurred. To deal with this, best place is in the onCreate options menu method – similar to working with widgets in layout, but working with items in a menu.

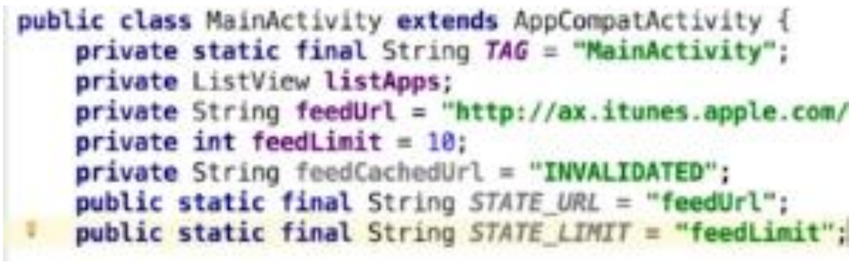


```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.feeds_menu, menu);
    if(feedLimit == 10) {
        menu.findItem(R.id.mnu10).setChecked(true);
    } else {
        menu.findItem(R.id.mnu25).setChecked(true);
    }
    return true;
}
```

Figure 31 - Restore Feed Limit

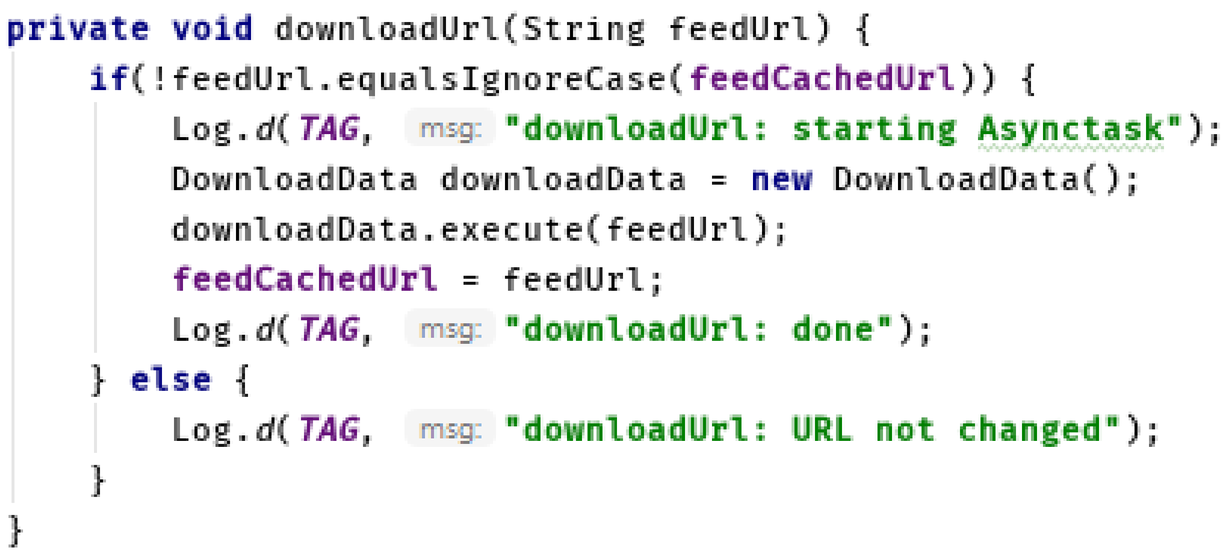
Can make sure item that's checked in the menu does actually reflect the value we're holding in the code

We want a download to occur when the refresh button is chosen from the menu. Need to compare cached URL with the one about to be downloaded. The downloadURL method is called whenever a URL is going to be downloaded; good place to perform check.



```
public class MainActivity extends AppCompatActivity {
    private static final String TAG = "MainActivity";
    private ListView listApps;
    private String feedUrl = "http://ax.itunes.apple.com/
    private int feedLimit = 10;
    private String feedCachedUrl = "INVALIDATED";
    public static final String STATE_URL = "feedUrl";
    public static final String STATE_LIMIT = "feedLimit";
```

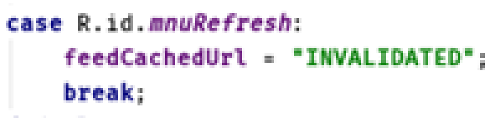
Figure 32 - Set URL and Limit



```
private void downloadUrl(String feedUrl) {
    if(!feedUrl.equalsIgnoreCase(feedCachedUrl)) {
        Log.d(TAG, msg: "downloadUrl: starting Asynctask");
        DownloadData downloadData = new DownloadData();
        downloadData.execute(feedUrl);
        feedCachedUrl = feedUrl;
        Log.d(TAG, msg: "downloadUrl: done");
    } else {
        Log.d(TAG, msg: "downloadUrl: URL not changed");
    }
}
```

Figure 33 - Download URL

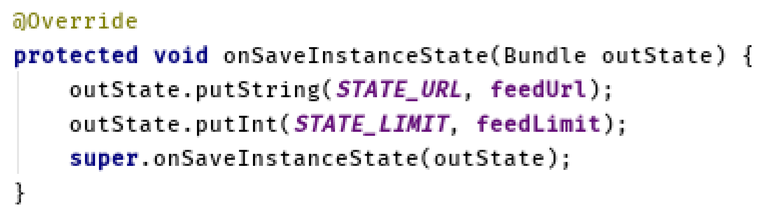
If not equal to the current cached URL then downloadData called and fed. Else, log for testing.



```
case R.id.mnuRefresh:
    feedCachedUrl = "INVALIDATED";
    break;
```

Figure 34 - Add this to onOptionsItemSelected

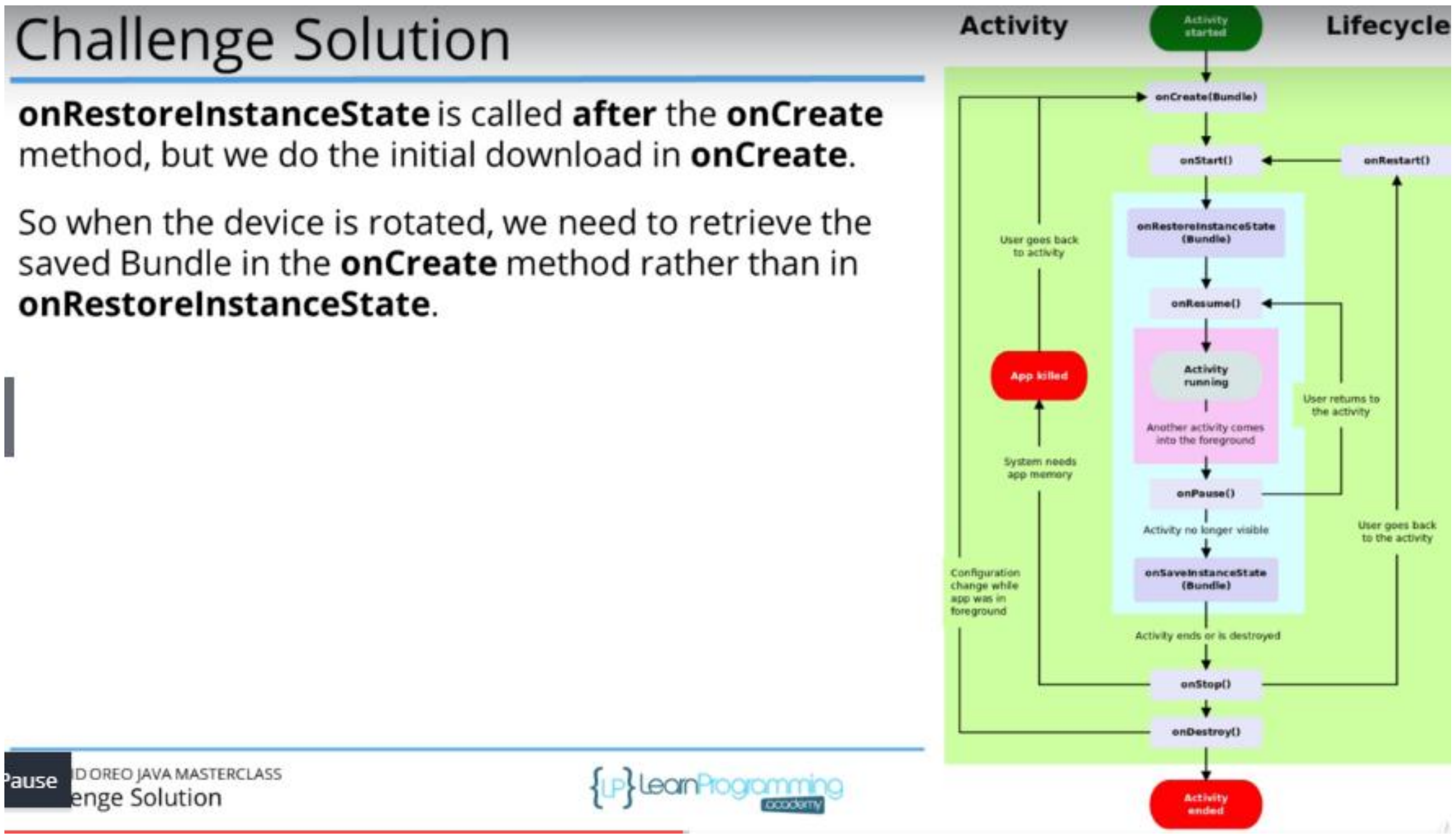
See Figure 29. Adds test here, assign feed cached URL to anything – Invalid. Now, the next time we then download after doing that, we're overriding it and are able to download the same URL, as if != to current URL – download.



```
@Override
protected void onSaveInstanceState(Bundle outState) {
    outState.putString(STATE_URL, feedUrl);
    outState.putInt(STATE_LIMIT, feedLimit);
    super.onSaveInstanceState(outState);
}
```

Figure 35 - On Saved Instance State

Put before super – this line creates and saves the bundle. This saves State of the URL, the feedURL. And the State limit, feedLimit.



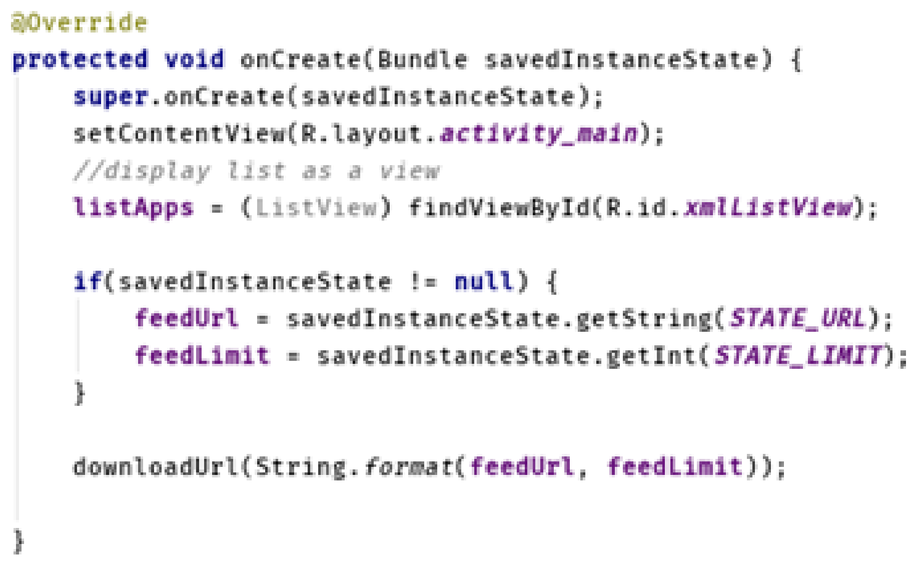
Challenge Solution

onRestoreInstanceState is called **after** the **onCreate** method, but we do the initial download in **onCreate**.

So when the device is rotated, we need to retrieve the saved Bundle in the **onCreate** method rather than in **onRestoreInstanceState**.

Activity
Lifecycle
Activity started
onCreate(Bundle)
onStart()
onRestart()
onRestoreInstanceState (Bundle)
User goes back to activity
onResume()
App killed
Activity running
User returns to the activity
Another activity comes into the foreground
System needs app memory
onPause()
Activity no longer visible
User goes back to the activity
onSaveInstanceState (Bundle)
Configuration change while app was in foreground
Activity ends or is destroyed
onStop()
onDestroy()
Activity ended
Pause
D OREO JAVA MASTERCLASS
enge Solution
{LP} LearnProgramming academy

Figure 36 - Activity Lifecycle Note

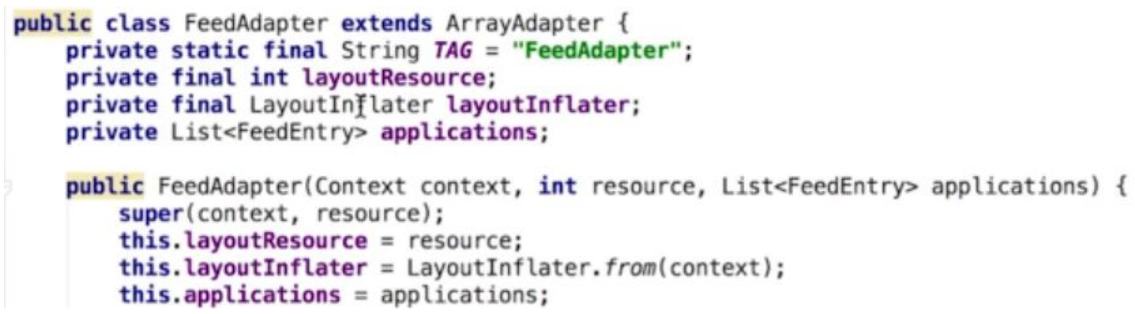


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //display list as a view
    listApps = (ListView) findViewById(R.id.xmlListView);

    if(savedInstanceState != null) {
        feedUrl = savedInstanceState.getString(STATE_URL);
        feedLimit = savedInstanceState.getInt(STATE_LIMIT);
    }

    downloadUrl(String.format(feedUrl, feedLimit));

}
```

Figure 37 - Recall Figure 28

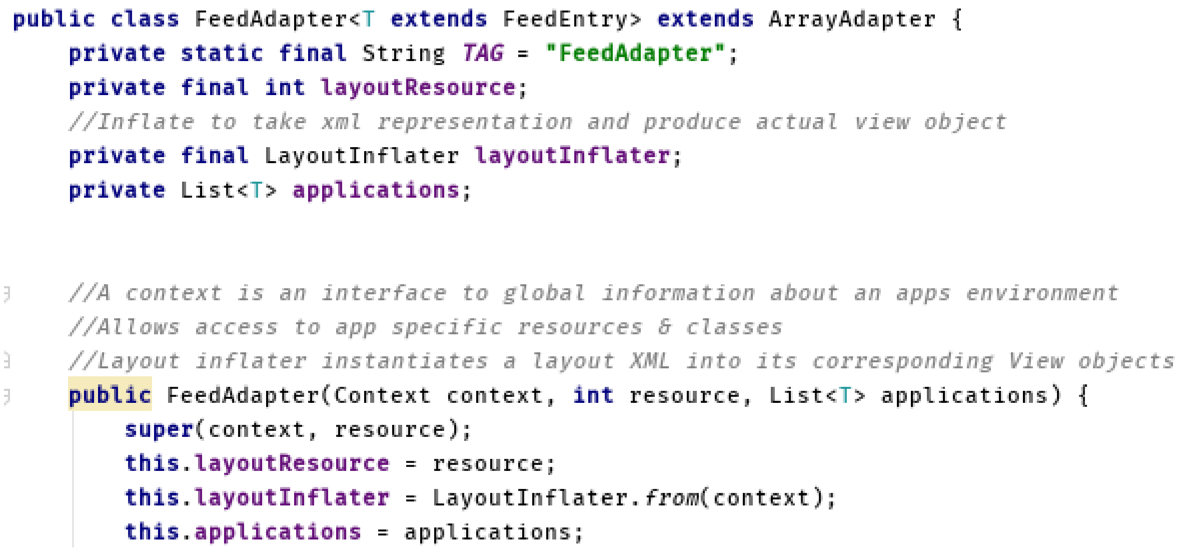
Generics - Custom Feed Adapter

Change class signature and change constructor to be generic



```java
public class FeedAdapter extends ArrayAdapter {
    private static final String TAG = "FeedAdapter";
    private final int layoutResource;
    private final LayoutInflater layoutInflater;
    private List<FeedEntry> applications;

    public FeedAdapter(Context context, int resource, List<FeedEntry> applications) {
        super(context, resource);
        this.layoutResource = resource;
        this.layoutInflater = LayoutInflater.from(context);
        this.applications = applications;
```

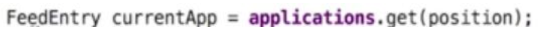
Figure 38 – Before



```java
public class FeedAdapter<T extends FeedEntry> extends ArrayAdapter {
    private static final String TAG = "FeedAdapter";
    private final int layoutResource;
    //Inflate to take xml representation and produce actual view object
    private final LayoutInflater layoutInflater;
    private List<T> applications;


    //A context is an interface to global information about an apps environment
    //Allows access to app specific resources & classes
    //Layout inflater instantiates a layout XML into its corresponding View objects
    public FeedAdapter(Context context, int resource, List<T> applications) {
        super(context, resource);
        this.layoutResource = resource;
        this.layoutInflater = LayoutInflater.from(context);
        this.applications = applications;
```

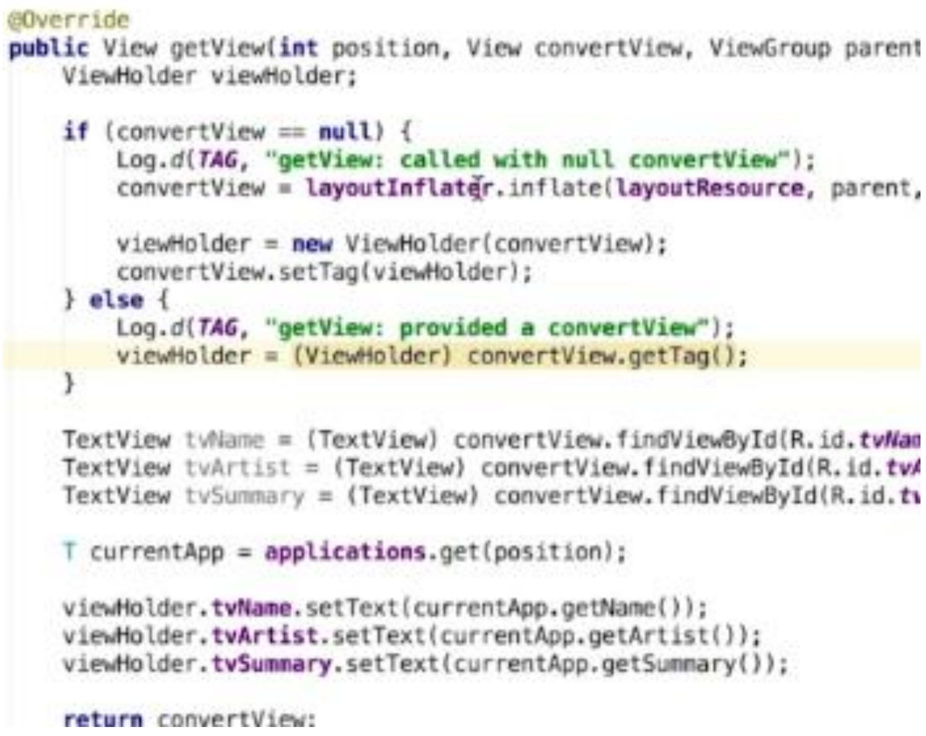
Figure 39 – After

Change get View method



```java
FeedEntry currentApp = applications.get(position);
```

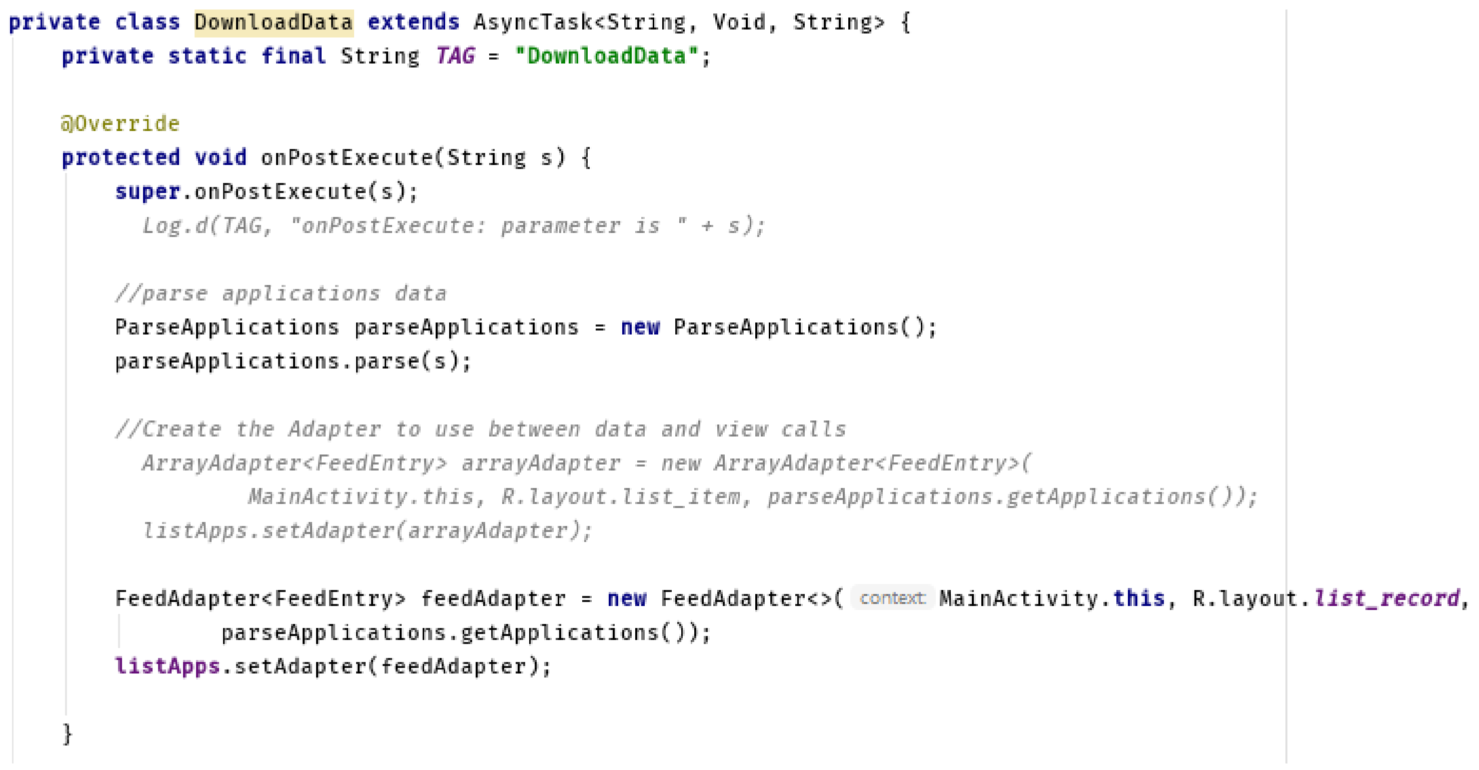
Figure 40 – Before



```java
@Override
public View getView(int position, View convertView, ViewGroup parent
    ViewHolder viewHolder;

    if (convertView == null) {
        Log.d(TAG, "getView: called with null convertView");
        convertView = layoutInflater.inflate(layoutResource, parent,

        viewHolder = new ViewHolder(convertView);
        convertView.setTag(viewHolder);
    } else {
        Log.d(TAG, "getView: provided a convertView");
        viewHolder = (ViewHolder) convertView.getTag();
    }

    TextView tvName = (TextView) convertView.findViewById(R.id.tvNam
    TextView tvArtist = (TextView) convertView.findViewById(R.id.tvA
    TextView tvSummary = (TextView) convertView.findViewById(R.id.tv

    T currentApp = applications.get(position);

    viewHolder.tvName.setText(currentApp.getName());
    viewHolder.tvArtist.setText(currentApp.getArtist());
    viewHolder.tvSummary.setText(currentApp.getSummary());

    return convertView;
```

Figure 41 – After

Last change is from On Post Execute in Main Activity



```
private class DownloadData extends AsyncTask<String, Void, String> {
    private static final String TAG = "DownloadData";

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
          Log.d(TAG, "onPostExecute: parameter is " + s);

        //parse applications data
        ParseApplications parseApplications = new ParseApplications();
        parseApplications.parse(s);

        //Create the Adapter to use between data and view calls
          ArrayAdapter<FeedEntry> arrayAdapter = new ArrayAdapter<FeedEntry>(
                  MainActivity.this, R.layout.list_item, parseApplications.getApplications());
          listApps.setAdapter(arrayAdapter);

        FeedAdapter<FeedEntry> feedAdapter = new FeedAdapter<>( context: MainActivity.this, R.layout.list_record,
                parseApplications.getApplications());
        listApps.setAdapter(feedAdapter);

    }
```