# 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

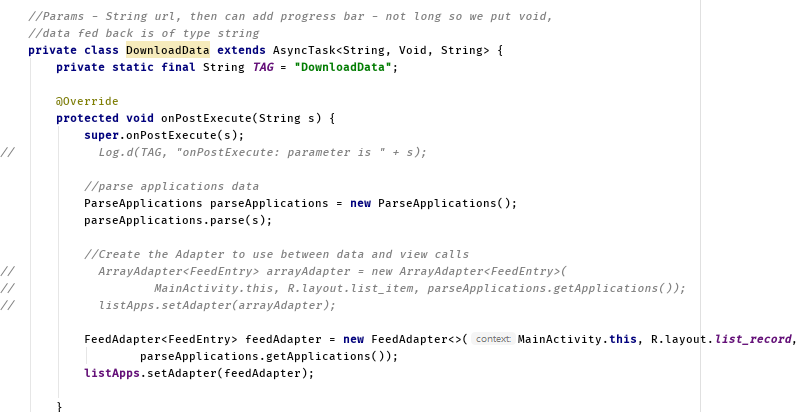
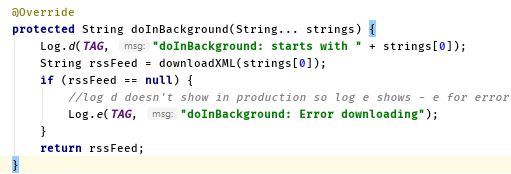


Figure - DownloadData AsyncTask

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)



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 - doInBackground inside AsyncTask

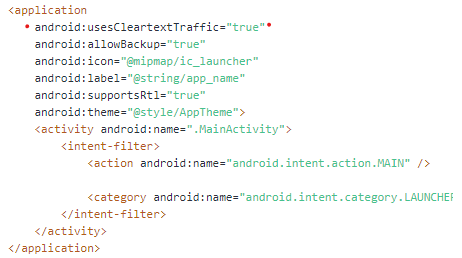


Figure - Add Clear Text Traffic

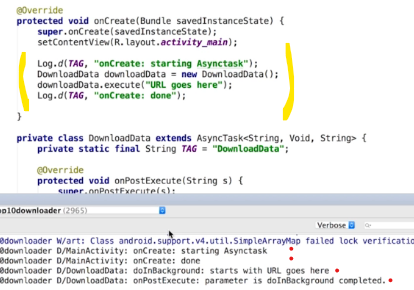


Figure - onCreate test

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

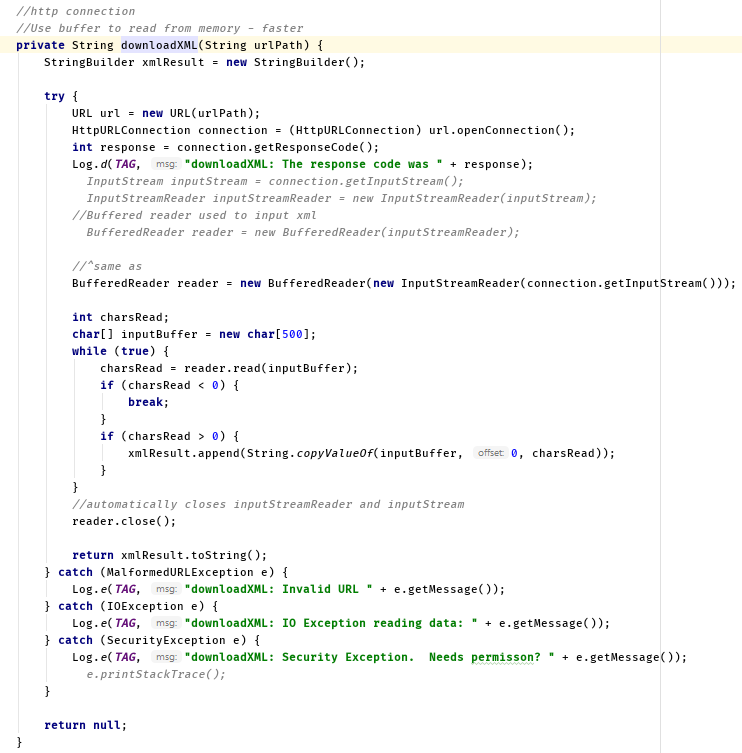


Figure - Download XML

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.

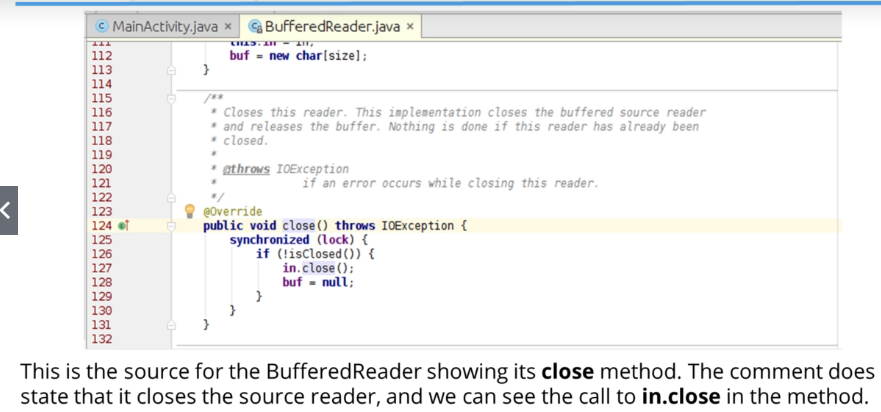


Figure - Buffered Reader



Figure - Add Internet Permission in Manifest

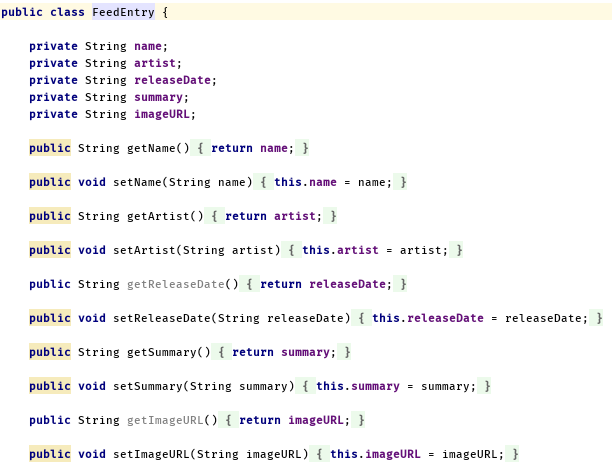


Figure - Separate Class, Feed Entry

Getter and Setters for the entries to be fed in.

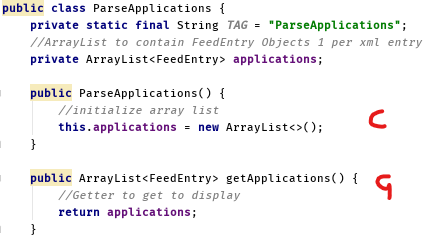


Figure - Separate Class, Parse Applications

Constructor and Getter.

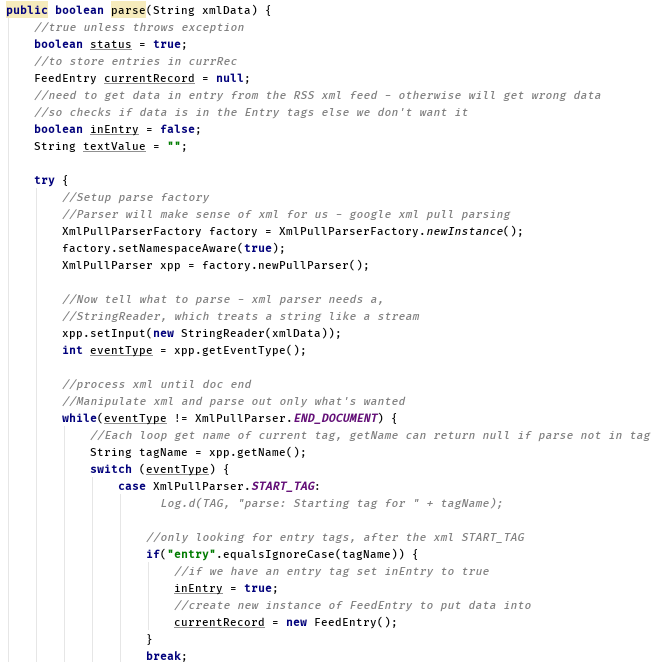


Figure - Boolean Parse XML Data 1



Figure - Boolean Parse XML Data 2

## List View and Array Adapter

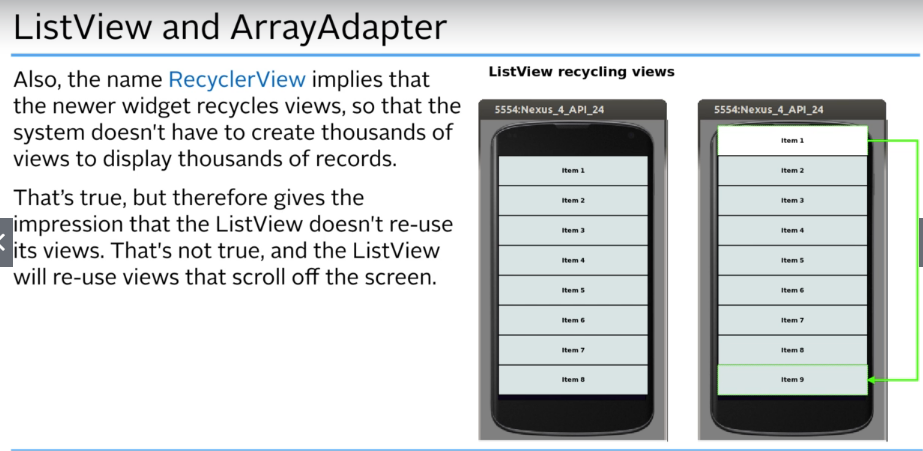


Figure - List View

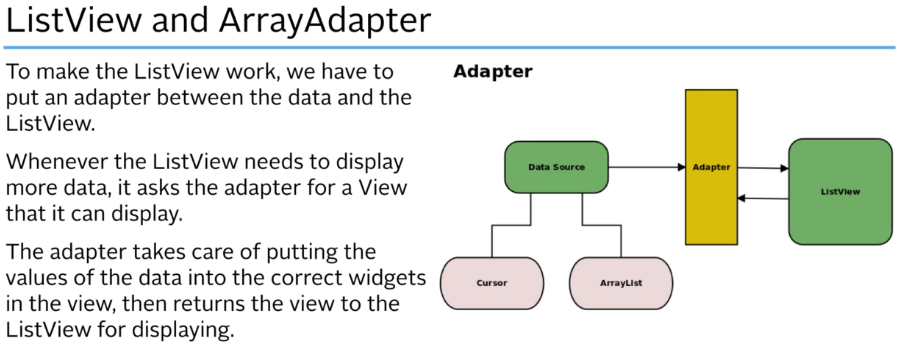


Figure - Array Adapter

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.

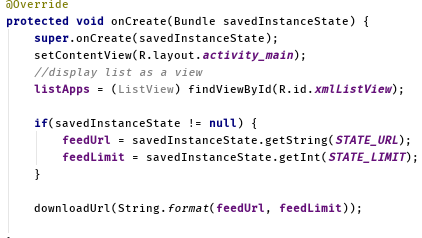


Figure - onCreate ListView, State & URL

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**;  
}

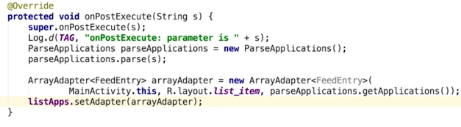


Figure - Default Adapter Setup

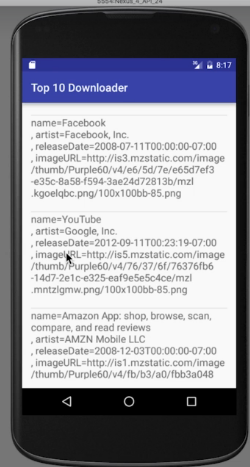


Figure - Display with default ArrayAdapter<>

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

## Custom Adapter Creation

New Class FeedAdapter extends ArrayAdapter

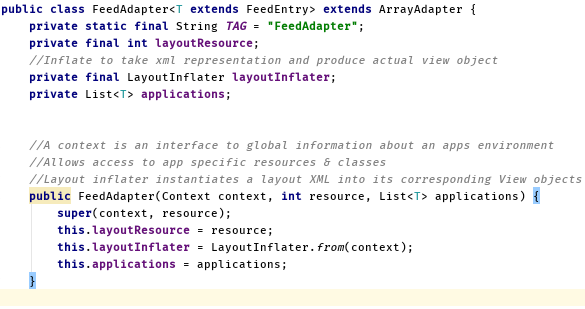


Figure - Define & Construct FeedAdapter

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.

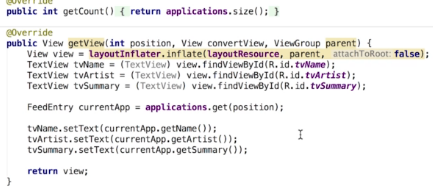


Figure – Get Count, View & Holder

Override the two methods.

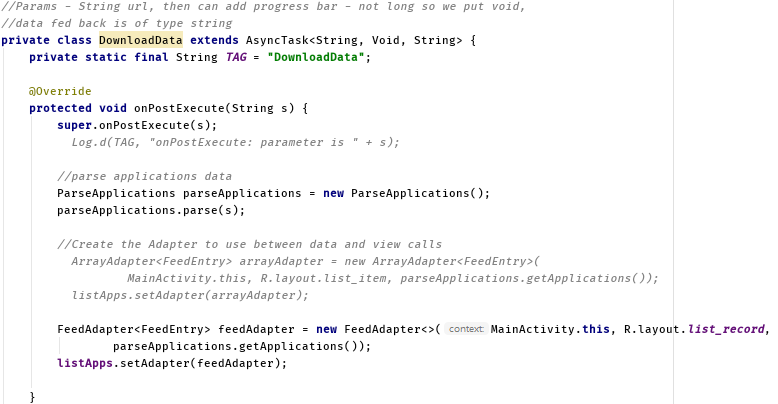


Figure - To use new Feed Adapter

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

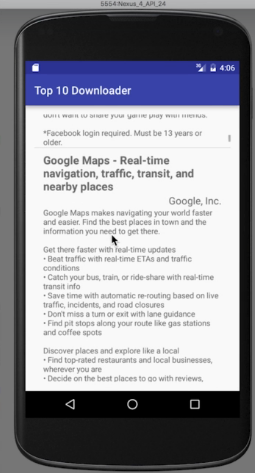


Figure – Works

## 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.

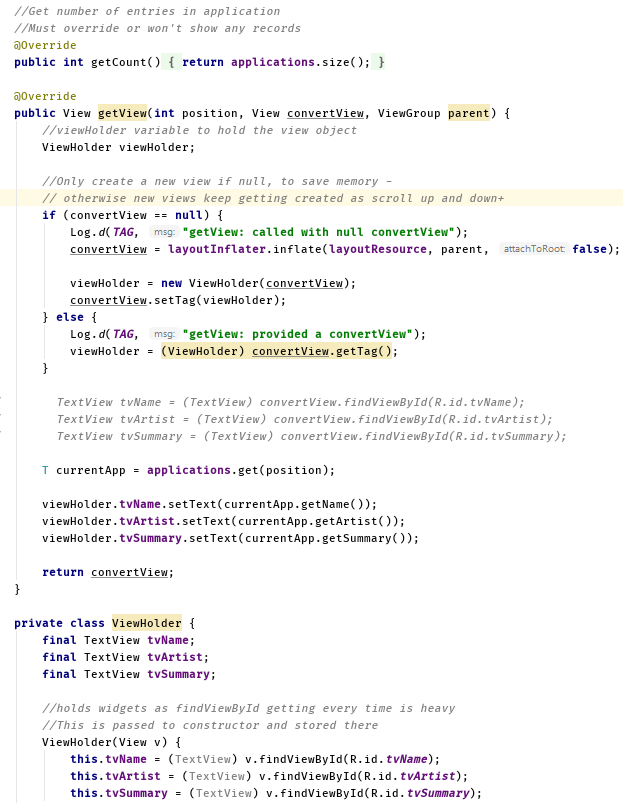


Figure - Change Feed Adapter Count & View

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:

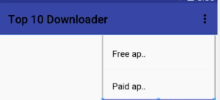


Figure - Example Menu Addition

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).

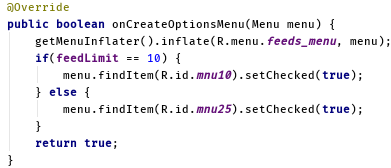


Figure - Options On Create Menu

Displays menu – No current functionality.



Figure - 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.

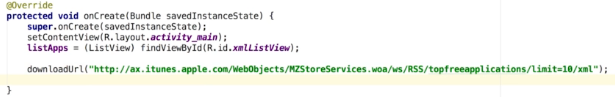


Figure - Change On Create

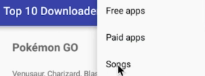


Figure – Functionality should work

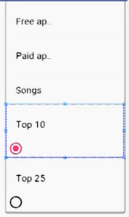


Figure - Add further menu items

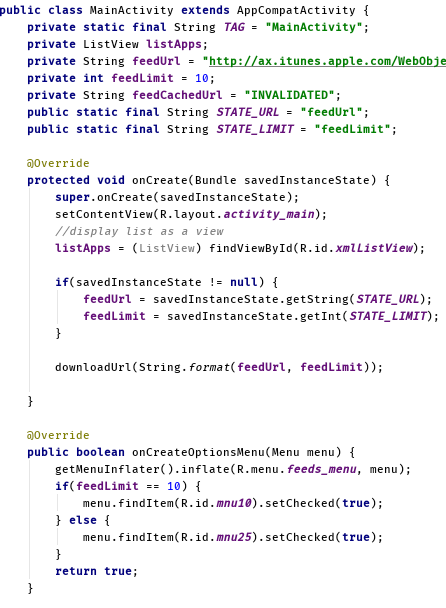


Figure - Functionality of Toggle Top 10/25



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

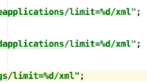


Figure - Change feedURL limit to %d

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.

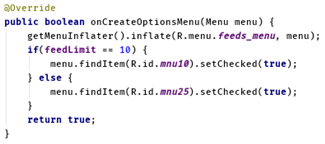


Figure - Restore Feed Limit

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.

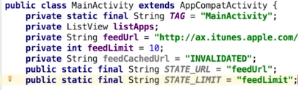


Figure - Set URL and Limit

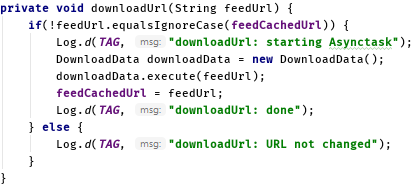


Figure - Download URL

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



Figure - 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.

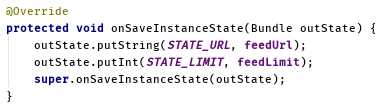


Figure - On Saved Instance State

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

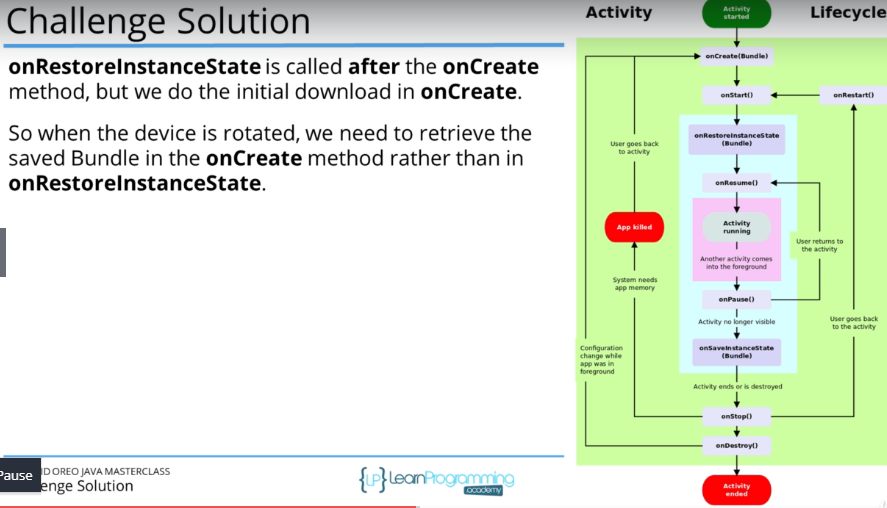


Figure - Activity Lifecycle Note

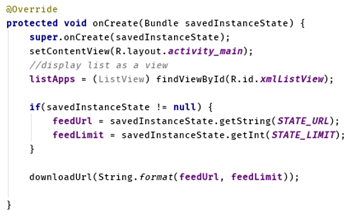


Figure - Recall Figure 28

## Generics - Custom Feed Adapter

Change class signature and change constructor to be generic

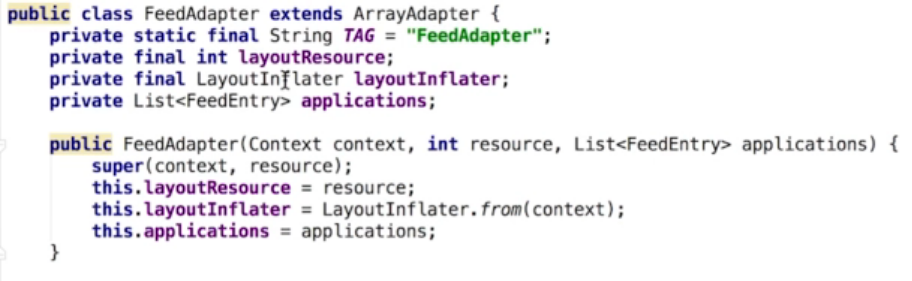


Figure – Before

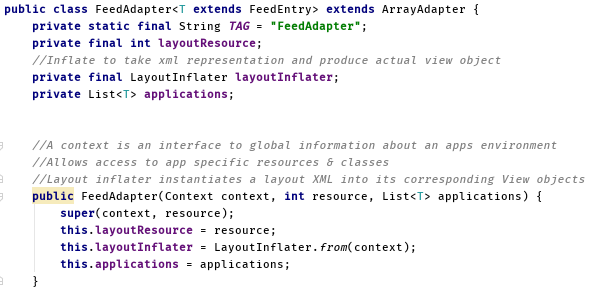


Figure – After

Change get View method



Figure – Before

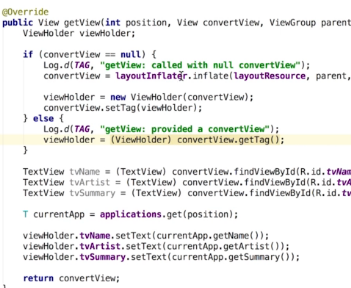


Figure – After

Last change is from On Post Execute in Main Activity

