

Android practicum part 4

# Introduction



Welcome to the big apple.

The marketing department calls:

Yes I see. Another spinnoff. Where? Ah New York?

Which colors? Uhm. Blue. Or grey. Or both…

This week you will further refine the CSI app.

A list with crimes should be shown for every criminal.

Don’t forget to adjust the style. Blue please.

Good luck.

# Another Copy

Just like last week, start by making a copy of the project. Give it a proper name.

# Badguys: Pick any: cheat, criminal, knave, pilferer purloiner, racketeer, robber, rogue, shark, shyster, swindler, villan.

In Hollywood, criminals are always fictional characters. This part is about creating these criminals.

The first activity of the app lets you choose a criminal, and sends the chosen crook to next activity. Usually an online database is used to look up what crimes the criminal has perpetrated, but that’s out of scope for this week.

In the example directory of smte4 you can find the ‘CSI\_week4’ project. You can use the files in this example for this practicum.

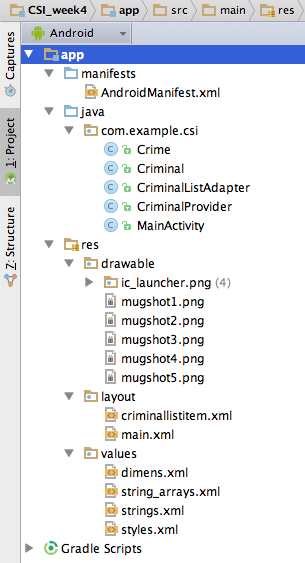


Figure 1: Picture of example workspace with ‘**CSI\_week4**’ project.

In the example project you will find the artifacts as described in the table below.

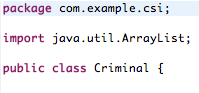
|  |  |
| --- | --- |
| MainActivity.java | Some example code on how to use the CriminalProvider to access the criminals and crimes. |
| Criminal.java | Class for storing details of the criminal. An object of a Criminal refers to a list of crimes. |
| Crime.java | Class for storing details about the crime. (Don’t do the crime if you can’t do the time…) |
| CriminalProvider.java | A helper class you can use to get access to a list of smooth criminals. |
| CriminalListAdapter.java | Class for translating a Criminal to a ListView item in the ListView for criminals. |
| res/layout/criminallistitem.xml | Layout file for a single criminal item in the ListView. |
| res/drawable/mugshot[1..5].png | Images for the criminals. Feel free to replace with criminals too your liking. |
| res/values/string\_arrays.xml | In this xml file we placed string-arrays the properties of the criminal and crimes. Feel free to adjust with more inspirational texts…! |

*Table 1: Overview of important java and xml files in the example project.*

Copy the blue marked files in table 1 from the example into your own project (you can copy the files by using your favorite file-explorer).

Some tips:

* Change the name of the package in the top of each java file if needed:



* Use MainActivity.java as an inspiration for the remainder of this practicum.

# Additional reference material

If you want to read how some parts of the provided code work, such as arrays and random numbers, take a look at the following links:

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>

<http://developer.android.com/reference/java/util/Random.html>

If you like to look at an example of an application with a list view please take a look at the ‘**ListView\_PerfumeExample**’ project within the smte4 example code.

## Show the villains face!

The first activity should show a list with criminals.

Because a lot of info can be shown about criminals, the simple ListView from last week will not suffice, and you will have to make your own list item view.

Add the following elements to the linearlayout of criminallistitem.xml.

* An **ImageView** to show the mugshot of the criminal.
* A **TextView** to show the name
* A **TextView** to show the total bounty

Be aware that these list items are shown in a vertical row, so keep the alignment horizontal.

Part 1: Tie the criminals to the list

To show the custom list item, you have to use an adapter class that will return the correct views to the ListView. Use the given CriminalListAdapter.java and update the overriden GetView method to return the correct view.

**

This method is called by the ListView for every row in the list. The position parameter indicates which row. Use the requested criminal from the list of criminals (attribute of the CriminalListAdapter class).

Because creating new rows is relatively costly in terms of memory and time, the Android OS tries to reuse existing rows, where only the data has to be adjusted.

If this is the case, then the convertView parameter contains such a reusable view.

So do the following:

* Check if convertView is not null.
* If it is null, create a new view. You can use an inflater to load the view from the layout:

LayoutInflater inflater = (LayoutInflater)context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

View criminalView = inflater.inflate(R.layout.criminallistitem, null);

When you have a valid criminalView object, you can use its findViewById to find the individual UI components and set the correct value (Mugshot, Name and total bounty).

The total bounty has to be calculated by adding all bounties from all crimes.

# Come together: connect the ListView to CriminalProvider and CriminalListAdapter

After all the hard work, it’s time to really use the list.

Open CriminalsListActivity make adjustments to the OnCreate method.

In the method body you have to create a CriminalProvider object:

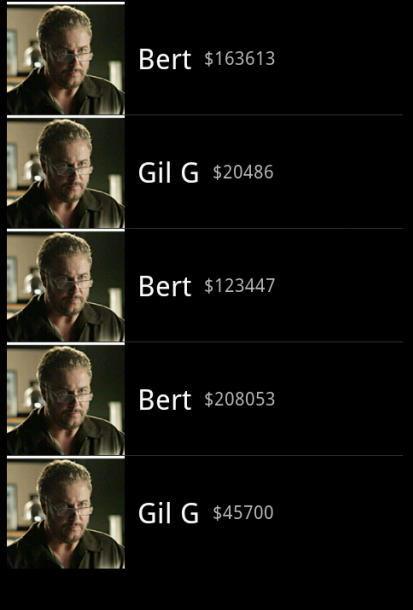
CriminalProvider criminalProvider = new CriminalProvider(…);

Pass the activity itself as a constructor parameter: use ‘this’ or ‘getApplicationContext()’ to do so.

After that you have to create a CriminalListAdapter object. Also pass the activity (this) as a parameter as well as a list with criminals, that you can get from the CriminalProvider by calling the getCriminals() method.

Finally attach the listadapter to the ListView by using the method setAdapter of the ListView object.

Result::



## Pass the bandit

By using the CriminalProvider you can now show more details from every criminal in the first activity.

However we also want to show details from the badguy in the next Activity.

The picture below will show how the position number is transferred from the criminal list activity to the criminal detail activity. The detail activity will reconstruct the criminal data by using the CriminalProvider and selecting the selected Criminal at the received position from the list.



Last week you passed the name (String) of the villain, by using an Intent. With the CriminalProvider class it’s easier to just sent the position (Integer) of the selected criminal to the next activity.

You can use Intent.putExtra to do so. Give it the name “chosenCriminalPosition” and use the selected item position in the list as the value.

Part 2:Receive and show the bandit

Pas in CSI\_weekXActivity.java de OnCreate methode aan.

Make adjustments to the OnCreate method in CSI\_weekXActivity.java .

Start by creating a CriminalProvider object.

After that retrieve the selected item from the previous activity by calling intent.getIntExtra()and store it in a variable called chosenCriminalPosition .

Use that position the find the Criminal object. Use the GetCriminal(chosenCriminalPosition) method on the CriminalProvider object.

Now adjust the code to show all the criminal data in the UI. (Mugshot, Name, Gender, Dersciption, Bounty etc).

Part 3:Show crimes list

In the current criminal details activity, the mugshot takes up a big part of the screen, so there is no room to show the individual crimes. Adjust the UI so the mugshot and the table are shown side to side (horizontal).

Place a ListView below the criminal picture that contains all the crimes that the criminal has committed.

Since we added a ListView for the crimes, you should roughly perform the same steps as you did with the criminals list.

* Create a new layout where you will show the crime details (name, bounty, description) on each row. Call it crimelistitem.xml.
* Create a new adapter class. Call it CrimeListAdapter.java and adjust the constructor so a List<Crime> can be passed as a parameter.
* Adjust the GetView method of the adapter. Let it use the layout in crimelistitem.xml and set the correct vallues for each crime.
* Connect the listadapter to the ListView