MOBILE APPLICATION DEVELOPMENT (MAD) : LAB 4

OBJECTIVES

- Understanding Android Intent Filter
- Understanding ListView
- Practice Activities

OBJECTIVE 1: Understanding Intent-Filter

- Intent Filter is way to sort out the intents, called using Implicit Intent.
- Structured description of Intent values to be matched.
- An Intent Filter can match against actions, categories, and data (either via its type, scheme, and/or path) in an Intent

Filter Rules

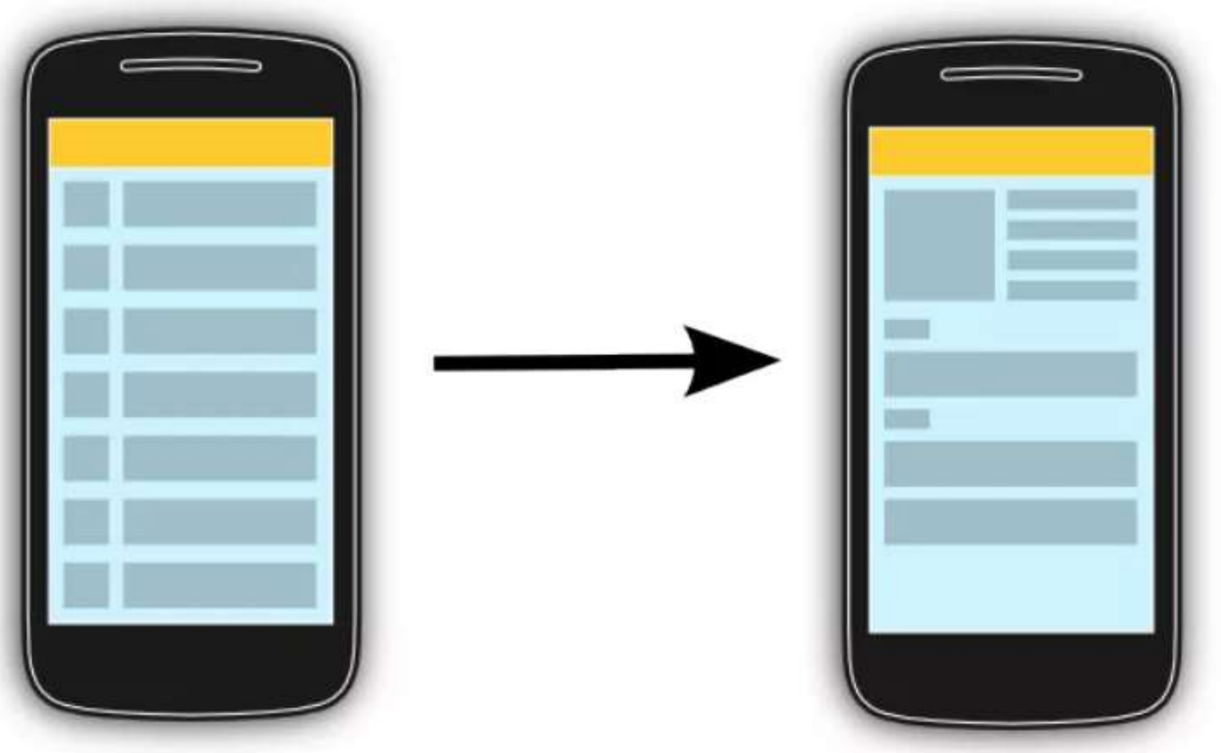
- An **Intent Filter** to match an Intent, three conditions must hold: the **action** and **category** must match, and the **data** (both the **data type** and **data scheme+authority+path** if specified) must match.
- **Action** matches if any of the given values match the Intent action; if the filter specifies no actions, then it will only match Intents that do not contain an action.
- **Categories** match if all of the categories in the Intent match categories given in the filter.

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />

    <category android:name="android.intent.category.DEFAULT" />

    <data android:scheme="http" />
</intent-filter>
```

OBJECTIVE 1: Understanding Intent-Filter

Displays a vertically-scrollable collection of views, where each view is positioned immediatelybelow the previous view in the list. For a more modern, flexible, and performant approach to displaying lists, use RecyclerView.

Typically the user interacts with the list via the toolbar, for example, via a button which refreshes the list. Individual list items can be selected. This selection can update the toolbar or can trigger a detailed screen for the selection. The following graphic sketches that. On the selection of a list item another activity is started.



➔ Possible input types for lists

The input of a list (items in the list) can be arbitrary Java objects. The adapter extracts the correct data from the data object and assigns this data to the views in the row of the ListView.

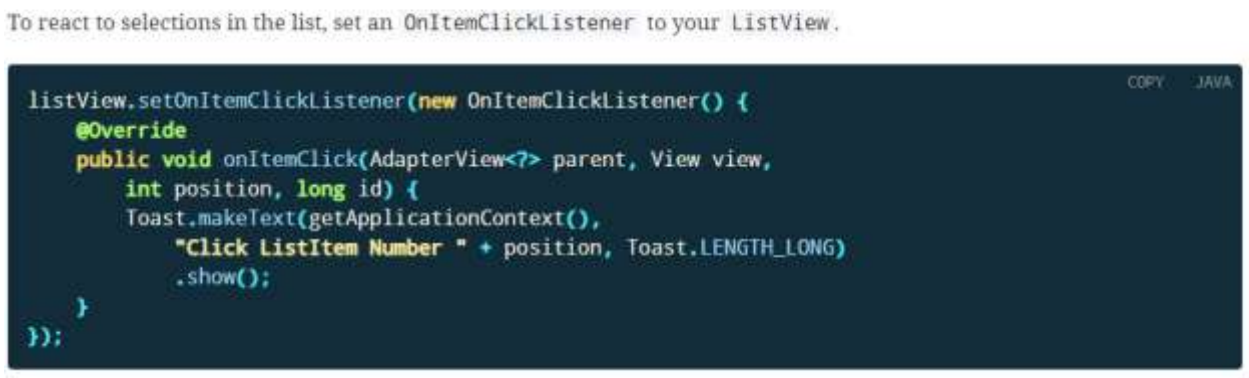
These items are typically called the data model of the list. An adapter can receive data as input

➔ Adapters

An adapter manages the data model and adapts it to the individual entries in the widget. An adapter extends the BaseAdapter class.

Every line in the widget displaying the data consists of a layout which can be as complex as you want. A typical line in a list has an image on the left side and two text lines in the middle as depicted in the following graphic.

➔ Listener



To react to selections in the list, set an `OnItemClickListener` to your `ListView`.

```java
listView.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
        Toast.makeText(getApplicationContext(),
            "Click ListItem Number " + position, Toast.LENGTH_LONG)
            .show();
    }
});
```

➔ Using ArrayAdapter

The ArrayAdapter class can handle a list or arrays of Java objects as input. Every Java object is mapped to one row. By default, it maps the toString() method of the object to a view in the row layout.

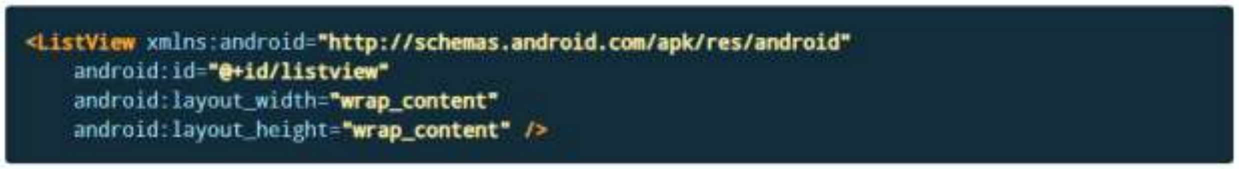
You can define the ID of the view in the constructor of the ArrayAdapter otherwise the android.R.id.text1 ID is used as default.

The ArrayAdapter class allows to remove all elements in its underlying data structure with the clear() method call. You can then add new elements via the add() method or a Collection via the addAll() method.

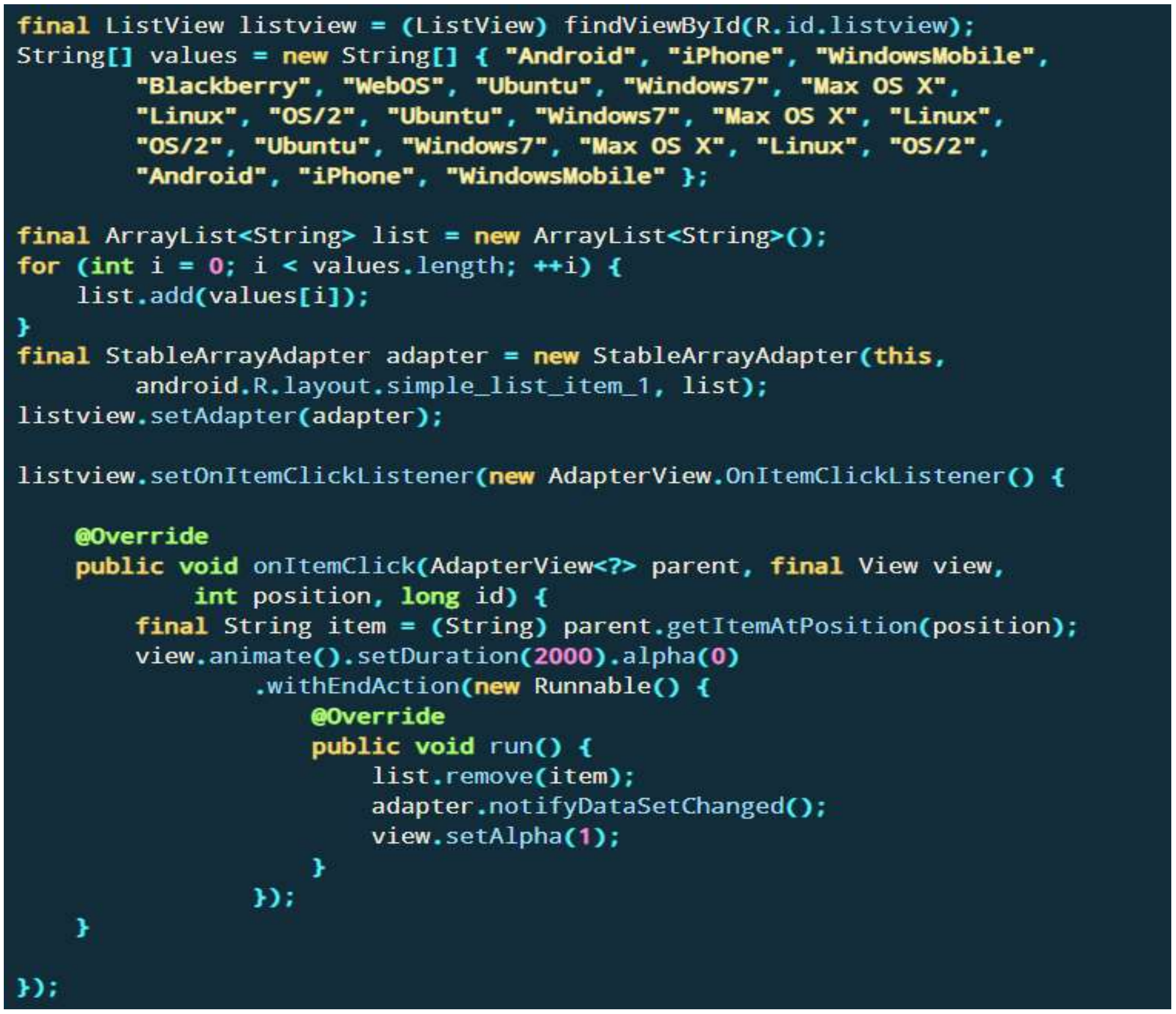
You can also directly modify the underlying data structure and call the notifyDataSetChanged() method on the adapter to notify it about the changes in data.

ListView example with ArrayAdapter

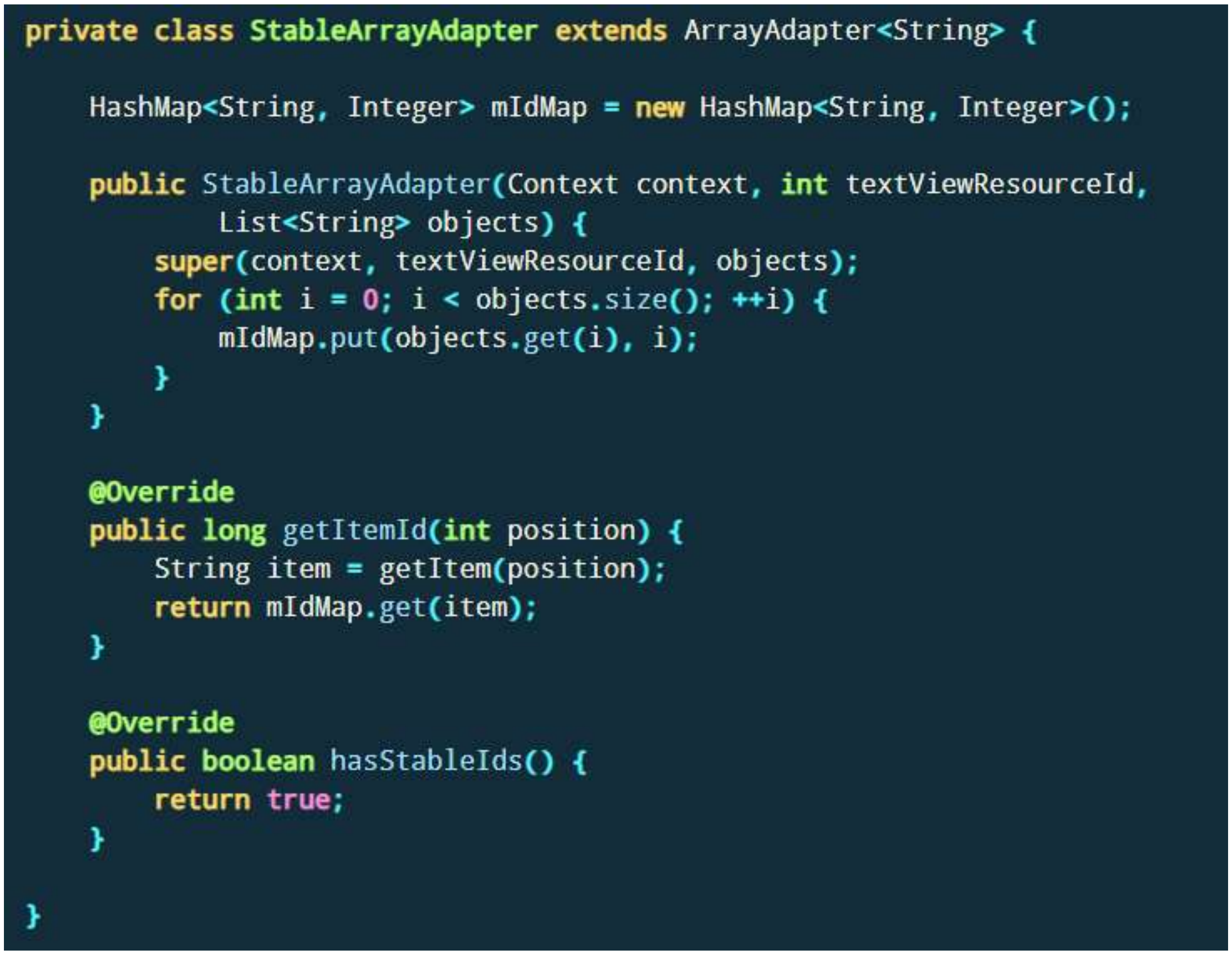
The following listing shows a layout file called activity_listviewexampleactivity.xml which includes a ListView



```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/listview"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```



```
final ListView listview = (ListView) findViewById(R.id.listview);
String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
        "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
        "Linux", "OS/2", "Ubuntu", "Windows7", "Max OS X", "Linux",
        "OS/2", "Ubuntu", "Windows7", "Max OS X", "Linux", "OS/2",
        "Android", "iPhone", "WindowsMobile" };

final ArrayList<String> list = new ArrayList<String>();
for (int i = 0; i < values.length; ++i) {
    list.add(values[i]);
}
final StableArrayAdapter adapter = new StableArrayAdapter(this,
        android.R.layout.simple_list_item_1, list);
listview.setAdapter(adapter);

listview.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> parent, final View view,
            int position, long id) {
        final String item = (String) parent.getItemAtPosition(position);
        view.animate().setDuration(2000).alpha(0)
                .withEndAction(new Runnable() {
                    @Override
                    public void run() {
                        list.remove(item);
                        adapter.notifyDataSetChanged();
                        view.setAlpha(1);
                    }
                });
    }

});
```



```java
private class StableArrayAdapter extends ArrayAdapter<String> {

    HashMap<String, Integer> mIdMap = new HashMap<String, Integer>();

    public StableArrayAdapter(Context context, int textViewResourceId,
            List<String> objects) {
        super(context, textViewResourceId, objects);
        for (int i = 0; i < objects.size(); ++i) {
            mIdMap.put(objects.get(i), i);
        }
    }

    @Override
    public long getItemId(int position) {
        String item = getItem(position);
        return mIdMap.get(item);
    }

    @Override
    public boolean hasStableIds() {
        return true;
    }

}
```

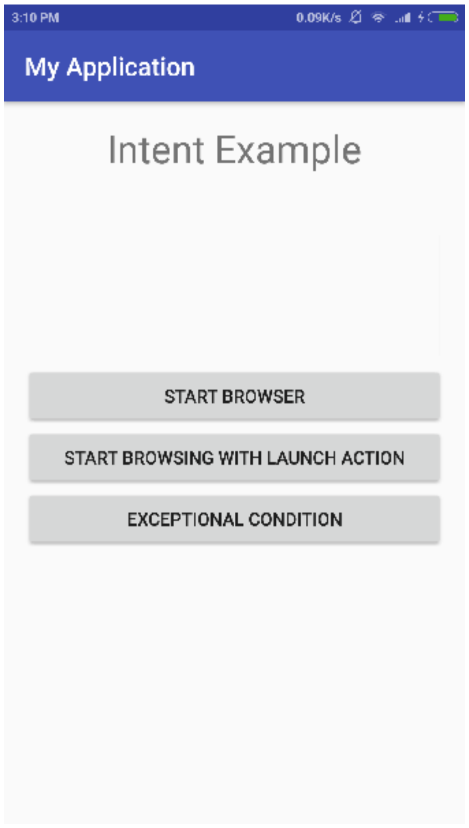
PRACTICE ACTIVITIES:

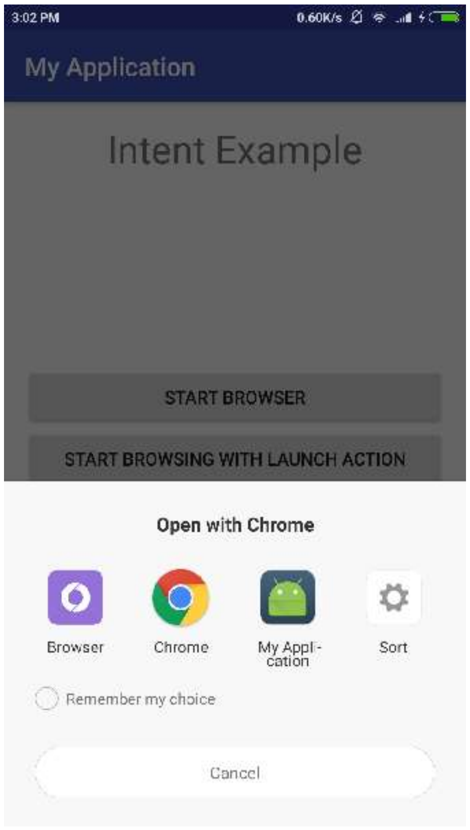
Activity 1: Android Internet Browser Steps:

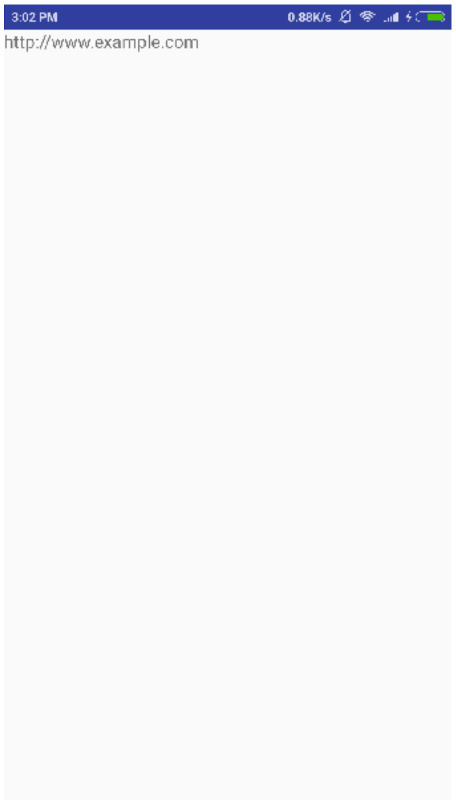
- Make Two Applications
- Add the following code in one app
- And call the implicit Intent from the other app using the below given intent action

```
<activity android:name="com.example.nisar.myapplication.CustomActivity">

         <intent-filter>

            <action android:name = "android.intent.action.VIEW" />

            <action android:name = "com.example.tutorialspoint7.myapplication.LAUNCH"
/>

            <category android:name = "android.intent.category.DEFAULT" />

            <data android:scheme = "http" />

         </intent-filter>

      </activity>
```

```
Intent i = new Intent("com.example.nisar.myapplication.
LAUNCH",Uri.parse("http://www.example.com"));

            startActivity(i);
```


3:10 PM
My Application
Intent Example
START BROWSER
START BROWSING WITH LAUNCH ACTION
EXCEPTIONAL CONDITION


3:02 PM
My Application
Intent Example
START BROWSER
START BROWSING WITH LAUNCH ACTION
Open with Chrome
Browser
Chrome
My Appli-cation
Sort
Remember my choice
Cancel


3:02 PM
http://www.example.com

Activity 2: ToDo List

Write a simple to-do list app that has a `ListView` of tasks that the user needs to complete. Initially the app is empty and has nothing in the to-do list. But if the user types text into a bottom `EditText` and clicks an Add button, the new item will be added to the top or bottom of the list.

It's also good to have a way to **remove items** from the list. You could achieve this by attaching a listener to the list that removes an item when that item is clicked on by the user. Or if you want to try something slightly different, try making it remove an item when the user performs a "long click" (pressing and holding the mouse on an item). You can do this by calling the `setOnItemLongClickListener` method of your list and passing an anonymous `AdapterView.OnItemLongClickListener` class. Android Studio can help you auto-generate the skeletons of these anonymous listener classes if you press Ctrl-Space in the editor at the right place in the code.

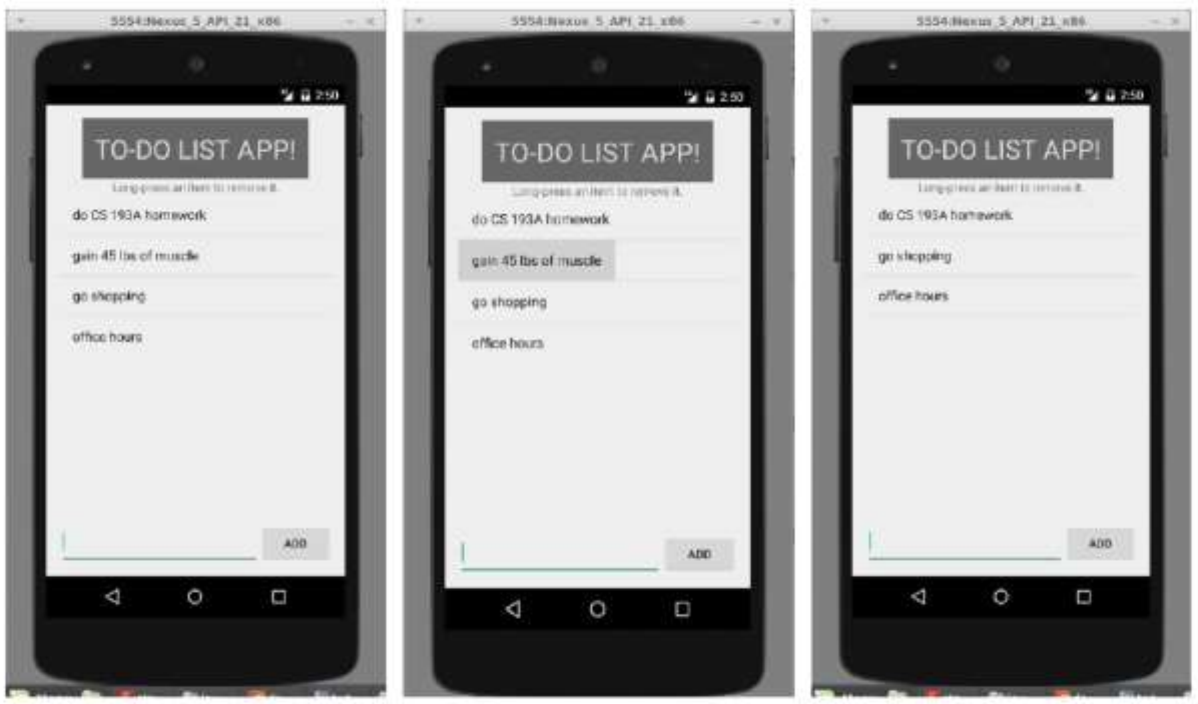
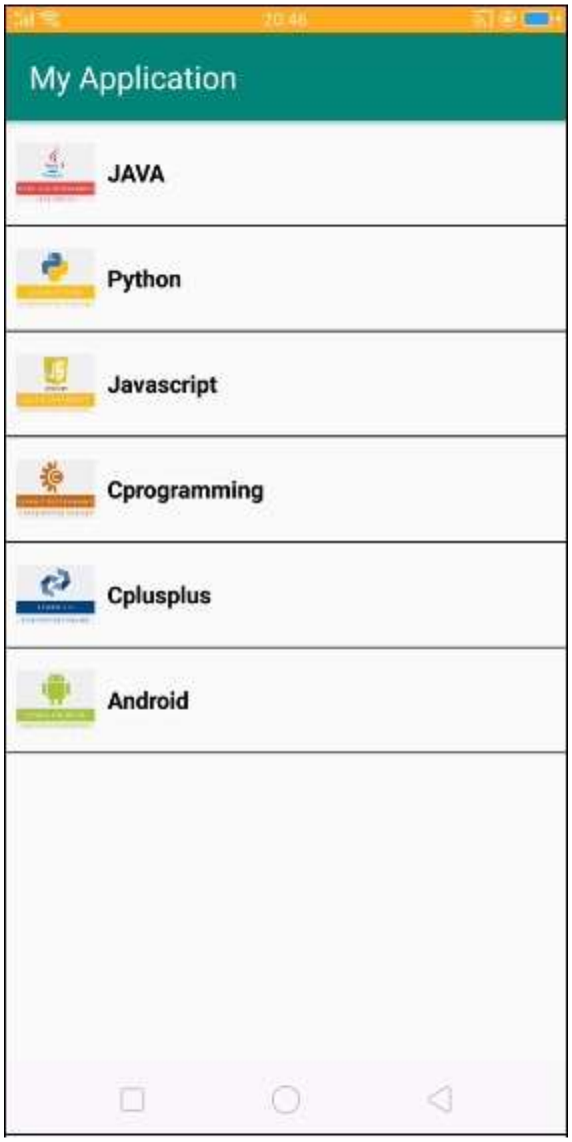


Figure: User long-clicking on second list item to delete it

If the items in your to-do list are stored into an `ArrayList`, by default the app's GUI won't notice when you add or remove an item from the list. That is, you'll modify the ArrayList state but the graphical list on the screen won't update to match. To fix this, you have to call the method `notifyDataSetChanged()` on your `ArrayAdapter` to tell it that the underlying array list has changed. To be able to do this, of course, you'll have to save your `ArrayList` and your `ArrayAdapter` as private fields inside your activity.

Activity 3: Create an application which shows the image and the text on the same line as "PhoneBook" Application.


My Application
JAVA
Python
Javascript
Cprogramming
Cplusplus
Android