

MOBILE APPLICATION DEVELOPMENT (MAD) – LAB 6

Objectives:

- To understand the file handling
- Understanding the usage of Adapters
- Practice Activities

OBJECTIVE 1: Understanding File Storage on Android

- Android uses a file system that's similar to disk-based file systems on other platforms.
- A **File** object works well for reading or writing large amounts of data in start-to-finish order without skipping around.

➤ Choose Where to store data

Internal storage:

- It's always available.
- Files saved here are accessible by only your app.
- When the user uninstalls your app, the system removes all your app's files from internal storage.

Internal storage is best when you want to be sure that neither the user nor other apps can access your files.

External storage:

- It's not always available, because the user can mount the external storage as USB storage and in some cases remove it from the device.
- It's world-readable, so files saved here may be read outside of your control.
- When the user uninstalls your app, the system removes your app's files from here only if you save them in the directory from `getExternalFilesDir()`.

➤ Saving data on Internal Storage

You can acquire the appropriate directory as a File by calling one of two methods:

- **getFilesDir()**
Returns a File representing an internal directory for your app.
- **getCacheDir()**
Returns a File representing an internal directory for your app's temporary cache files. Be sure to delete each file once it is no longer needed and implement a reasonable size limit for the amount of memory you use at any given time, such as 1MB.

```
String filename = "myfile";
String fileContents = "Hello world!";
FileOutputStream outputStream;

try {
    outputStream = openFileOutput(filename, Context.MODE_PRIVATE);
    outputStream.write(fileContents.getBytes());
    outputStream.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

➤ Opening a file from Internal Storage

```
File directory = context.getFilesDir();
File file = new File(directory, filename);
```

Fragment with View Pager (WhatsApp like fragments):

https://github.com/codepath/android_guides/wiki/ViewPager-with-FragmentPagerAdapter

MOBILE APPLICATION DEVELOPMENT (MAD) – LAB 6

➤ Saving data on External Storage

After you request storage permissions and verify that storage is available, you can save two different types of files:

- **Public files:** Files that should be freely available to other apps and to the user. When the user uninstalls your app, these files should remain available to the user. For example, photos captured by your app or other downloaded files should be saved as public files.
- **Private files:** Files that rightfully belong to your app and will be deleted when the user uninstalls your app. Although these files are technically accessible by the user and other apps because they are on the external storage, they don't provide value to the user outside of your app

➤ Request permission before accessing external storage

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>

<manifest ...>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

➤ Methods to get access to the storage

- **getExternalFilesDir()**
getExternalFilesDir() creates a directory that is deleted when the user uninstalls your app.
- **getExternalStoragePublicDirectory()**
You use **getExternalStoragePublicDirectory()** for files that are shared and will not be deleted when app is uninstalled.

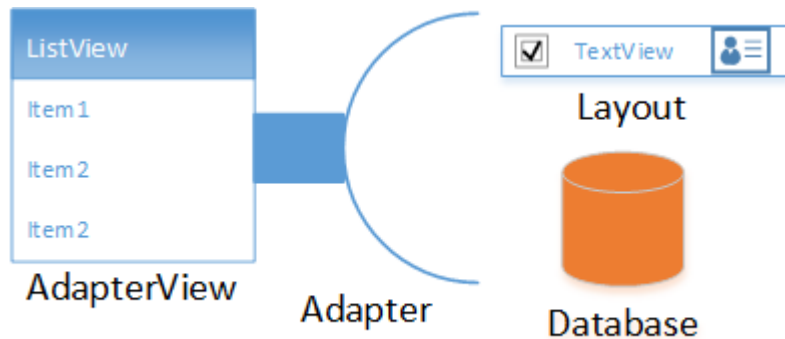
Fragment with View Pager (WhatsApp like fragments):

https://github.com/codepath/android_guides/wiki/ViewPager-with-FragmentManager

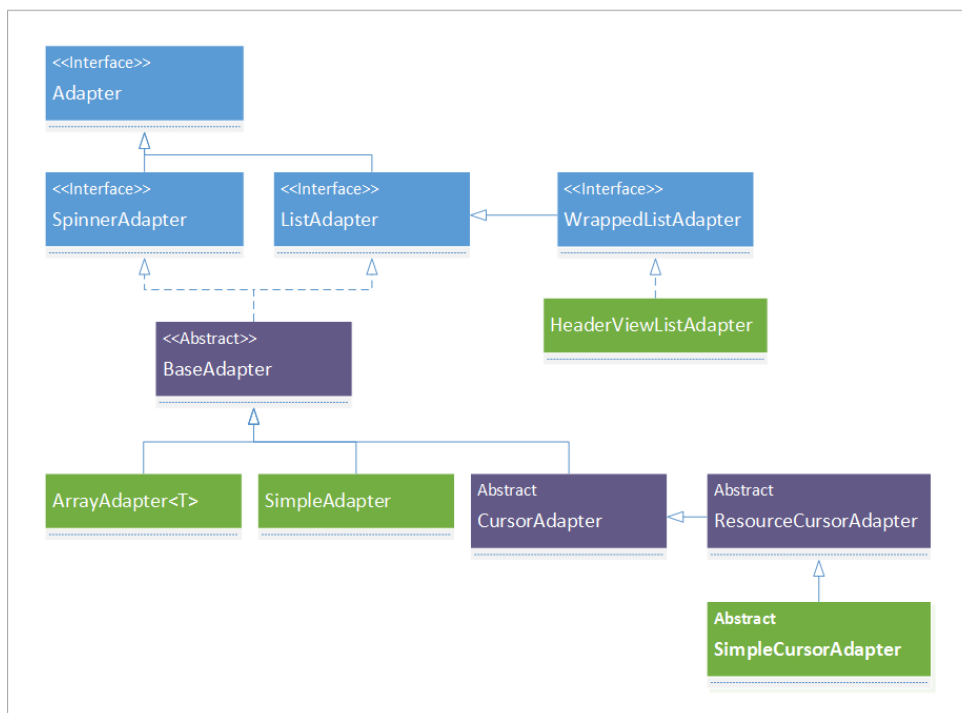
MOBILE APPLICATION DEVELOPMENT (MAD) – LAB 6

Objective 2: Introduction to Adapters.

- A bridge between an AdapterView and the underlying data for that view.
- An AdapterView is a group of widgets (aka view) components in Android that include the ListView, Spinner, and GridView.
- AdapterView also provides the layout of the underlying data for the view



Types of Adapters



Adapter View Methods

- **getCount()**: indicates to Android how many items (or rows) are in the data set that will be presented in the AdapterView.
- **getItem(int pos)**: get the data item associated with the item (or row) from the AdapterView passed as a parameter to the method. This method will be used by Android to fetch the appropriate data to build the item/row in the AdapterView.

Fragment with View Pager (WhatsApp like fragments):

https://github.com/codepath/android_guides/wiki/ViewPager-with-FragmentPagerAdapter

MOBILE APPLICATION DEVELOPMENT (MAD) – LAB 6

- **getItemId(int pos):** This method returns the data set's id for a item/row position of the AdapterView. Typically, the data set id matches the AdapterView rows so this method just returns the same value.
- **getView(int position, View convertView, ViewGroup parent):** This method creates the View (which may be a single View component like a TextView or a complex set of widgets in a layout) that displays the data for the specified (by position) item/row in the AdapterView

Fragment with View Pager (WhatsApp like fragments):

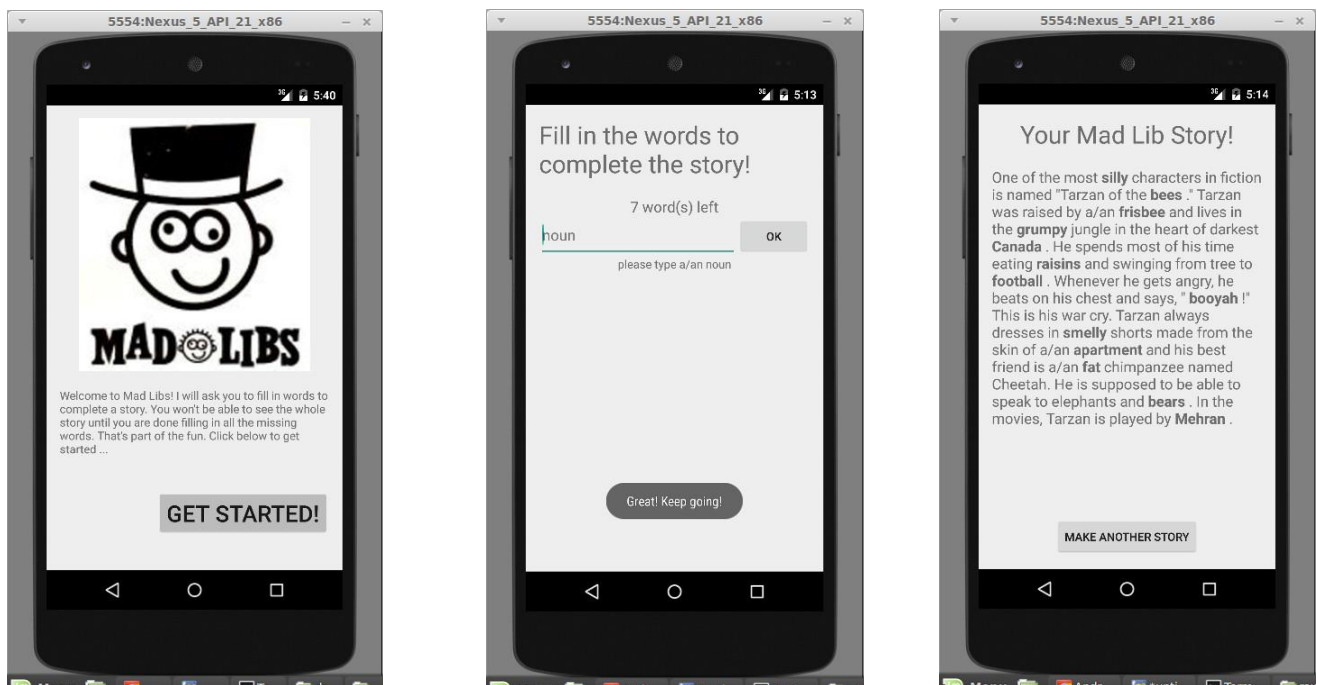
https://github.com/codepath/android_guides/wiki/ViewPager-with-FragmentPagerAdapter

OBJECTIVE 3: ACTIVITIES.

Activity 1: Complete the Task from Lab 4 – Activity 3 : MadLibs.

"Mad Libs" are short stories that have blanks called placeholders to be filled in. In the non-computerized version of this game, one person asks a second person to fill in each of the placeholders without the second person knowing the overall story. Once all placeholders are filled in, the second person is shown the resulting silly story.

Write an Android app that reads in a Mad Lib from a text file in a specific format. The text file represents placeholders as tokens that start and end with < > brackets, like <adjective> or <proper-noun>. Your app reads the file, looks for any such placeholders, and prompts the user to replace them with specific words. Once the user has typed in replacements for all placeholders, the completed story is shown on the screen. The screenshots below indicate a possible flow of the UI for such an app. Our flow has three activities: An initial "welcome" screen explaining the app, then a screen that repeatedly prompts the user to fill in placeholders, then a third activity to display the completed story. Of course you don't need to exactly match our sample's UI, but it may give you ideas.



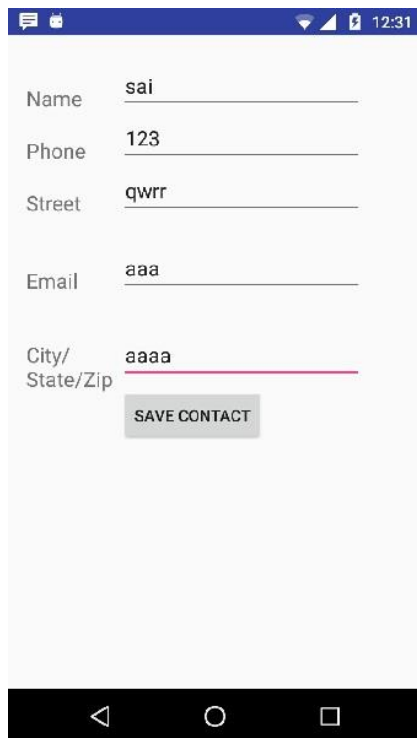
Fragment with View Pager (WhatsApp like fragments):

https://github.com/codepath/android_guides/wiki/ViewPager-with-FragmentManager

MOBILE APPLICATION DEVELOPMENT (MAD) – LAB 6

Activity 2: Add contacts File : Steps to follow

Steps	Description
1	You will use Android studio to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to get references of all the XML components and populate the contacts on listView.
3	Create new src/DBHelper.java that will manage the database work
4	Create a new Activity as DisplayContact.java that will display the contact on the screen
5	Modify the res/layout/activity_main to add respective XML components
6	Modify the res/layout/activity_display_contact.xml to add respective XML components
7	Modify the res/values/string.xml to add necessary string components
8	Modify the res/menu/display_contact.xml to add necessary menu components
9	Create a new menu as res/menu/mainmenu.xml to add the insert contact option
10	Run the application and choose a running android device and install the application on it and verify the results.



A screenshot of an Android application's contact form. The form has a light gray background and is titled 'Name' at the top. It contains several input fields: 'Name' with the value 'sai', 'Phone' with '123', 'Street' with 'qwrr', 'Email' with 'aaa', and 'City/State/Zip' with 'aaaa'. Below the 'City/State/Zip' field is a gray button labeled 'SAVE CONTACT'. The status bar at the top shows the time as 12:31 and various icons. The bottom navigation bar shows the back, home, and recent apps buttons.



Fragment with View Pager (WhatsApp like fragments):

https://github.com/codepath/android_guides/wiki/ViewPager-with-FragmentManager