Charles Sturt University

Topic 6: Fragments and SQLite

Robin GAO

School of Computing and Mathematics

1. Proposal Peer Review
   - Value: 20%
   - Due Date: 25-Aug-2019
   - Return Date: 13-Sep-2019
   - Submission method options: Turnitin (online)

2. *You will be given a project proposal*
   - Random selection
   - ***DO NOT post Anonymously***
   - Leave constructive feedback
     - Clarity of the proposal
     - Technical soundness
     - Difficulty level of the project
     - Overall suggestions
   - Review the assigned proposals and provide a report based on your review

Charles Sturt University

Task

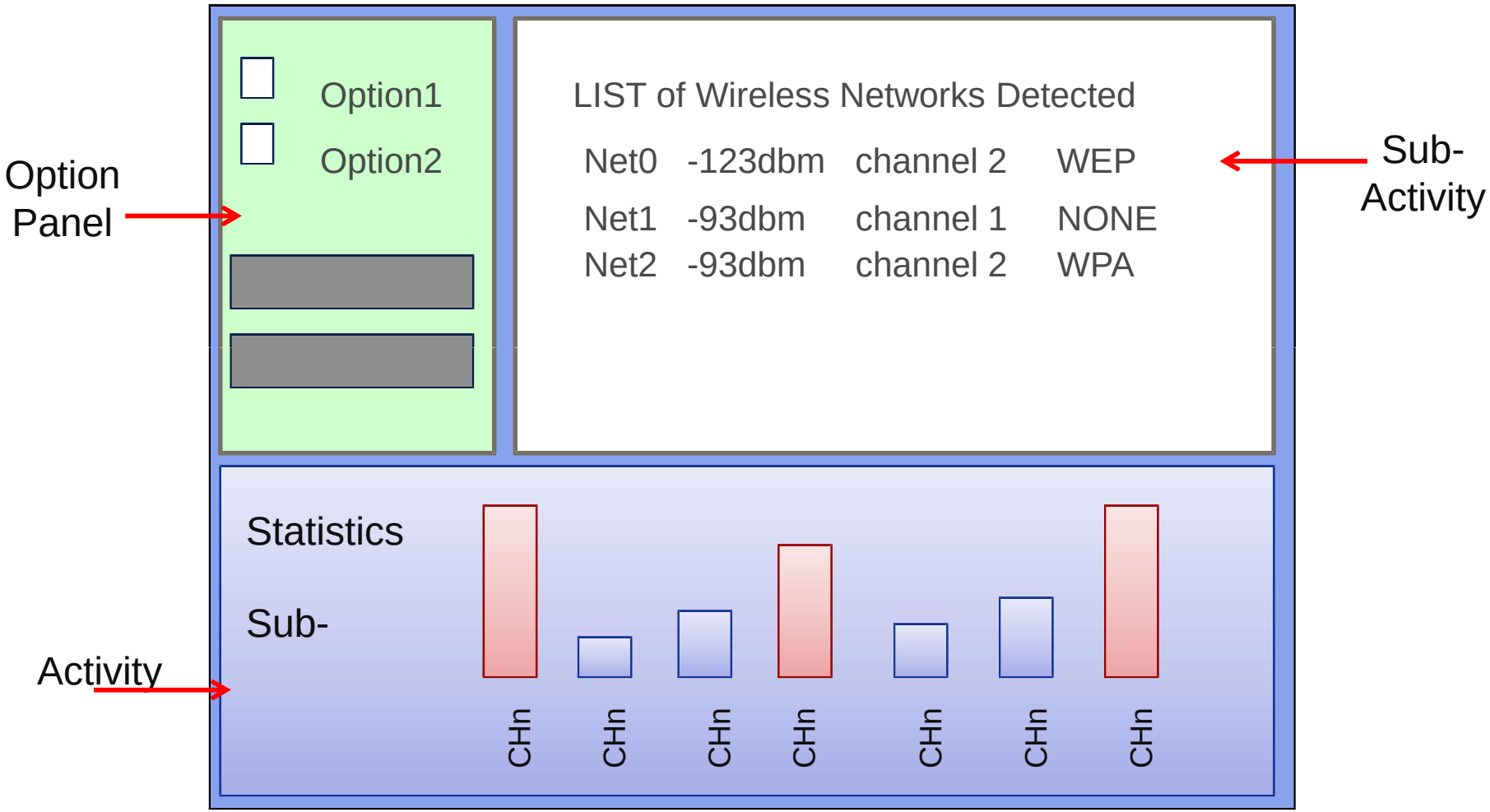
- This assessment is designed to peer review the project proposals submitted in Assessment Item 1.
- This will be an anonymous peer review.
- The subject coordinator will assign specific project proposals to each student for a peer review.
- You are required to review the assigned proposals and provide a report based on your review.
- This report will be graded and will count towards your final grade.

Charles Sturt University

Task

- Your review of a project proposal should take the following points into consideration:
  - Clarity of the proposal presentation (well structured, easy to read, logical statements, proper motivation statement)
  - Technical soundness of the proposed project (clarity of the technical terms, technical soundness of the proposal, appropriate selected references/material, creativity and originality of the proposed project)
  - Difficulty level of the proposed project (project requirements such as user interface & interaction, internet connection, database, visualization, use of camera, location information, etc. Constructive suggestions for modifications if required)
  - Overall suggestions to the project proposal (make constructive and positive suggestions)
- Your review, comments and statements should be specific. Avoid generic statements, justify your suggestions /opinions.
- Submit the review report via EASTS by 25 August.

Android: Application Case Study



Charles Sturt University

Android: Fragments

Fragment A portion of the user interface in an Activity.

Introduced from **Android 3.0** (API Level 11)

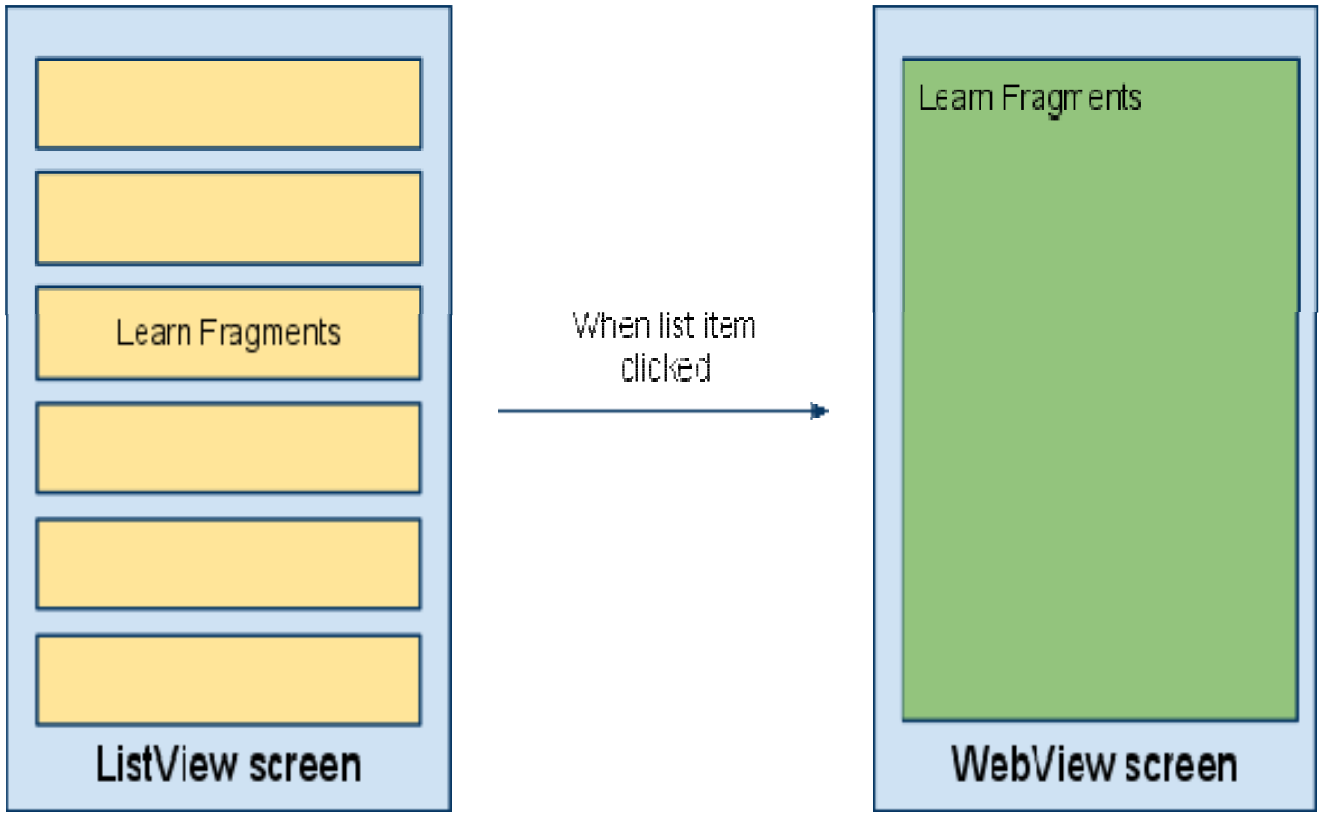
Practically, a Fragment is a modular section of an Activity.

DESIGN PHILOSOPHY

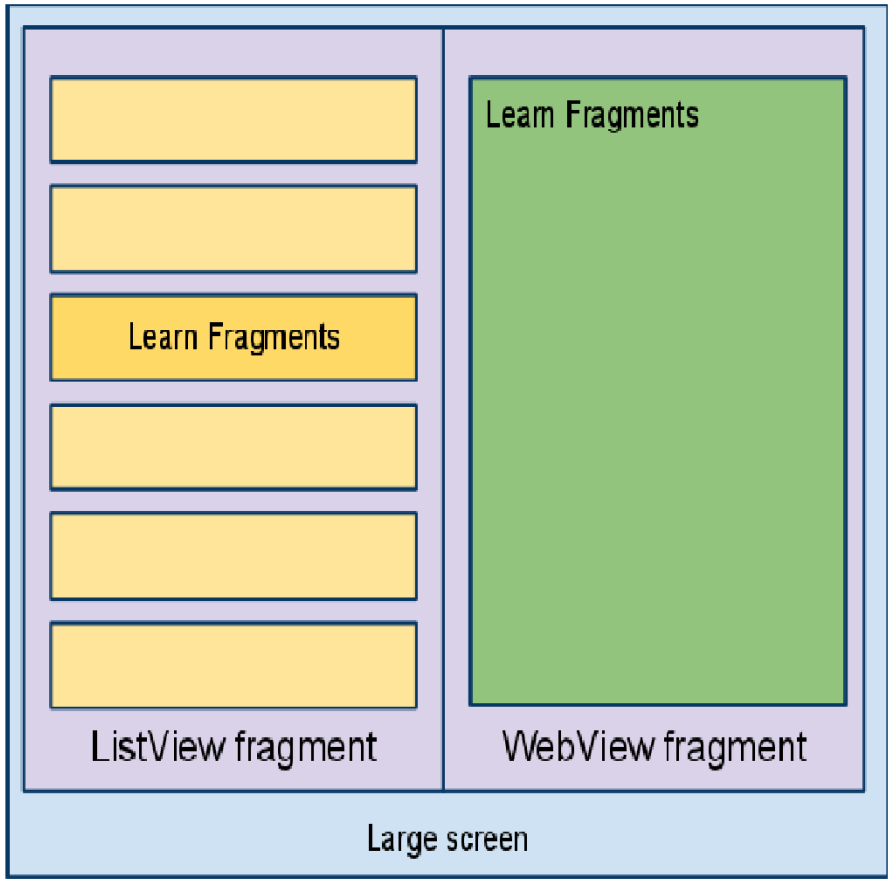
- **Structure** an Activity as a collection of Fragments.

- **Reuse** a Fragment on different Activities …

EXAMPLE: Structuring an Application using multiple Activities.

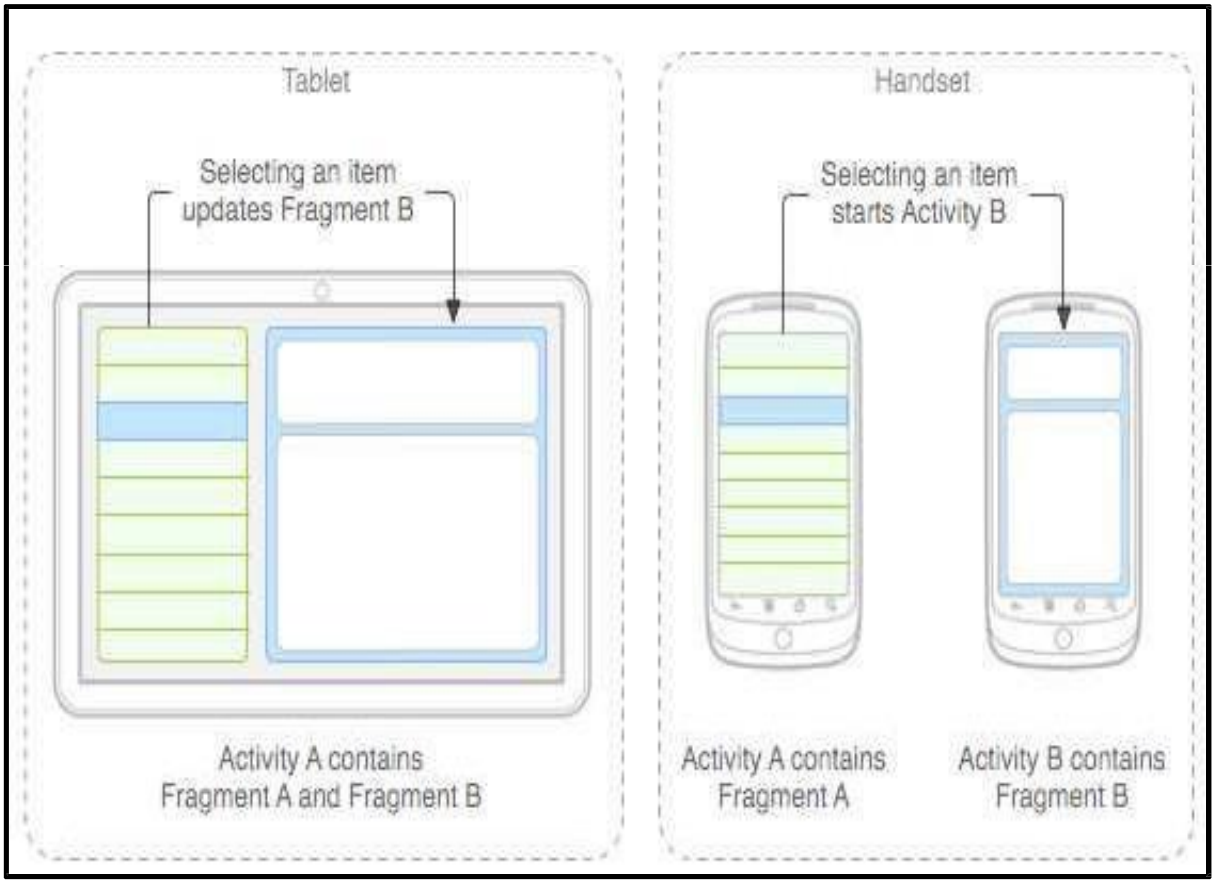


EXAMPLE: Structuring an Application using 1 Activity and 2 Fragments.



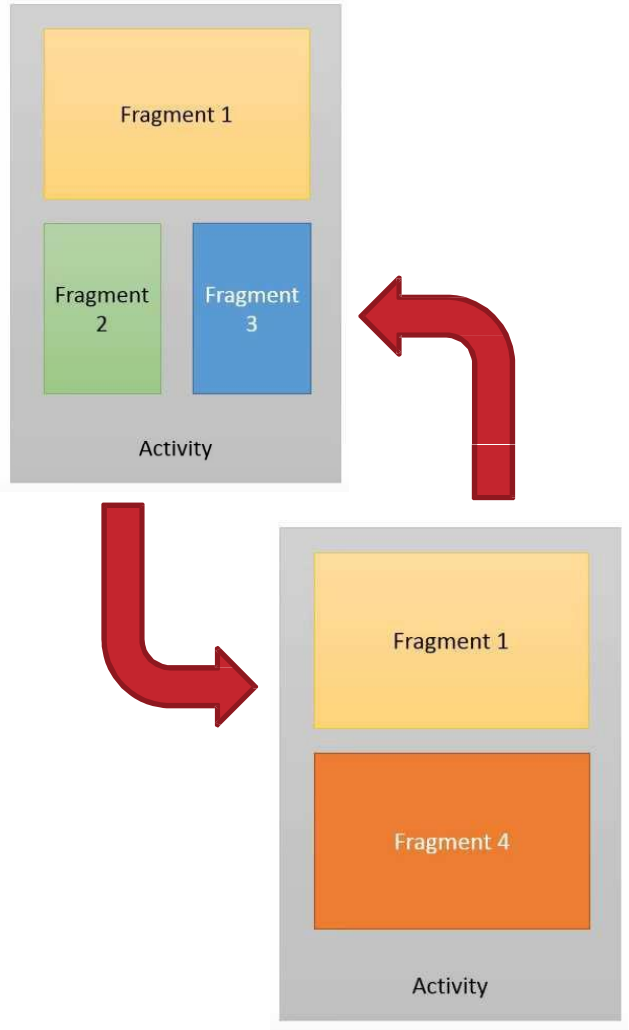
Android: Fragment Transactions

EXAMPLE: Using Fragments on Different Devices (Smartphone/Tab)



Fragments

- A Fragment is simply a self-contained user interface or behaviour
  - It has its own lifecycle
  - Hosted within Activity Classes
- Allow for increased UI flexibility
  - Allows the user to break up the screen workflow into separate components
  - Can be mix and matched in different ways
  - Dependant on screen space available
- Can reuse the same fragment across several activities
- Basically think of them as mini Activities
  - Have their own class and layouts.



https://developer.android.com/guide/components/fragments.html

Fragment Usage and Implementing

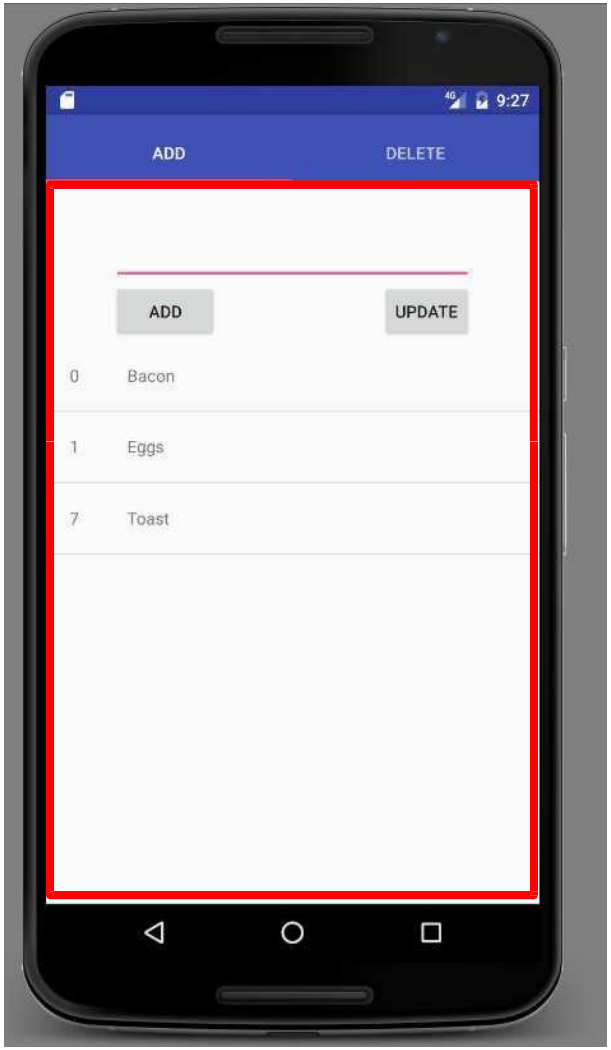
Fragments are mainly used for:

- Tabs
- List Views
- Dialog Boxes
- Tablet layouts

Created slightly differently

- New Class that extends Fragment
  - Has a corresponding View just like an activity
- Activity needs a ViewPager to manage Fragments
- Fragments need to be ‘inflated’ to be active
- Just like FABs we need to set listeners for interactions
  - Especially for elements that aren’t buttons like list items

- A tabbed App has the tabs in the activity
  - Each tab is a fragment
  - The tabs themselves need listeners



Android: Fragment Creation

To define a new Fragment create a subclass of Fragment.

```
public class MyFragment extends Fragment { …}
```

PROPERTY of a Fragment:

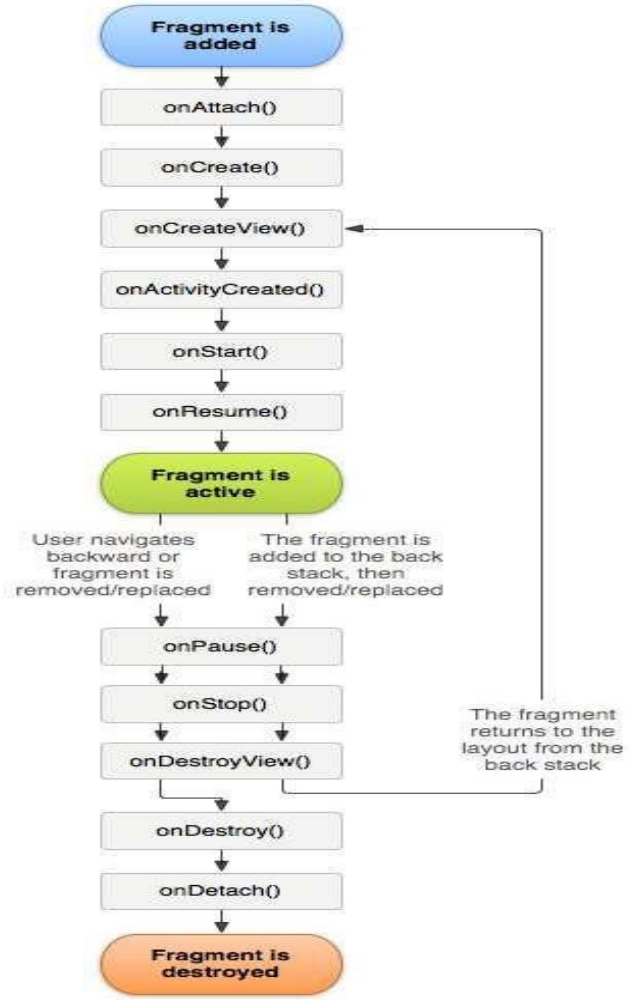
- Has its own **lifecycle** (partially connected with the Activity lifecyle)
- Has its own **layout** (or may have)
- Can receive its own **input events**
- Can be added or removed while the Activity is running.

Android: Adding a Fragment to the UI

Specify layout properties for the Fragment as it was a View.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="horizontal" >
<fragment android:name="it.cs.android30.FragmentOne"
android:id="@+id/f1" android:layout_width="wrap_content"
android:layout_height="fill_parent"
/>
<fragment android:name="it.cs.android30.FragmentTwo"
android:id="@+id/f2" android:layout_width="wrap_content"
android:layout_height="fill_parent"
/>
</LinearLayout>
```

Android: Fragment Lifecycle



Several **callback methods** to handle various stages of a Fragment lifecycle:

onCreate() called when creating the Fragment.

onCreateView() called when it is time for the Fragment to draw the user interface the first time.

onPause() called when the user is leaving the Fragment.

Android: Fragment Creation

onCreateView() must return the **View** associated to the UI of the Fragment (if any) …

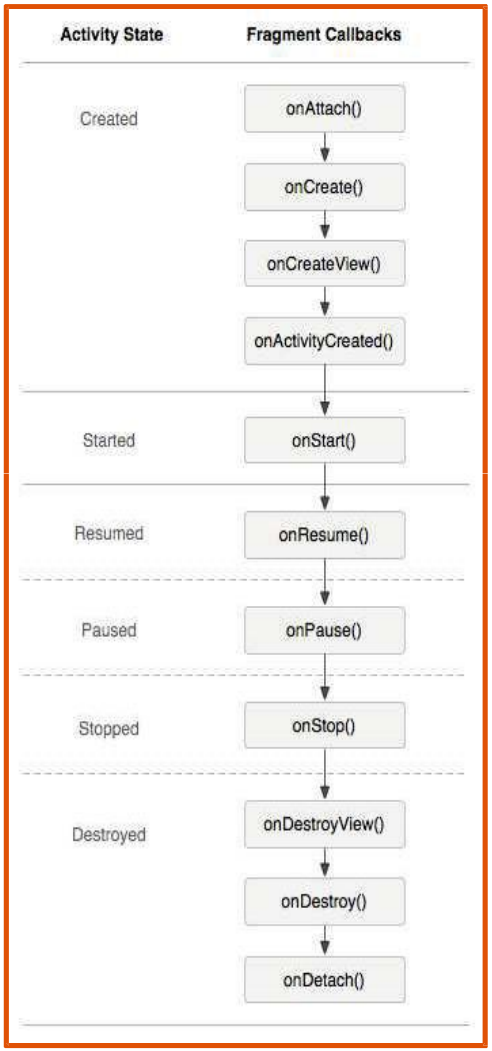
```
public class ExampleFragment extends Fragment {

@Override
public View onCreateView(LayoutInflater inflater,
ViewGroup container, Bundle saved) {

return inflater.inflate(R.layout.example_fragment,
container, false);

}
```

Android: Fragment Lifecycle



The lifecycle of the Activity in which the Fragment lives directly affects the lifecycle of the Fragment.

onPause (Activity) **onPause** (Fragment)

onStart (Activity) **onStart** (Fragment)

onDestroy (Activity) **onDestroy** (Fragment)

Fragments have also extra lifecycle callbacks to enable runtime creation/destroy.

Android: Managing Fragments

A **Fragment** can get a reference to the Activity

```
Activity getActivity()
```

An **Activity** can get a reference to the Fragment

```
...
ExampleFragment fragment=(ExampleFragment)
getFragmentManager().findFragmentById(R.id.ex
ample_fragment)
```

The **FragmentManager** manages the Fragment associated to the current Activity.

Android: Managing Fragments

In some cases, a Fragment must share an event with the Activity … how to do it?

1. Define a **callback** interface inside the Fragment

```
public interface OnArticleSelectedListener {
    public void onArticleSelected(Uri uri);
    ...
}
```

1. Require that the host Activity implements it

Charles Sturt University

Android: Fragment Transactions

- Fragments can be added/removed/replaced while the Activity is running …

- Each set of changes to the Activity is called a **Transaction**.

- **Transaction** can be saved in order to allow a user to navigate backward among Fragments when he clicks on the “Back” button.

Android: Fragment Transactions

1. **ACQUIRE** an instance of the FRAGMENT MANAGER

```
FragmentManager     man=getFragmentManager();
FragmentTransaction transaction=man.beginTransaction();
```

2. **CREATE** new Fragment and Transaction

```
FragmentExample     newFragment=new
```

3. **SAVE** to backStack and **COMMIT**

```
transaction.addToBackStack(null);
transaction.commit();
```

Charles Sturt University

- A Transaction is not performed till the **commit** …

- If **addToBackStack()** is not invoked the Fragment is destroyed and it is not possible to navigate back.

- If **addToBackStack()** is invoked the Fragment is stopped and it is possible to resume it when the user navigates back.

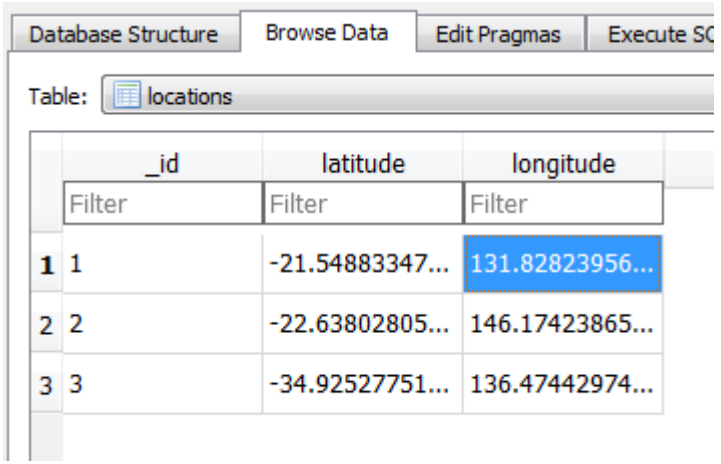
- **popBackStack**() simulate a Back from the user.

Charles Sturt University

SQLite

SQLite Introduction

- Applications need to display information or save information
  - Easiest way – Databases
- Android uses SQLite
  - Lightweight version of MySQL for local devices – don't need a server client
- Databases made of 'Tables'
  - Columns are category information
  - Rows are an entry
  - Primary keys are required
- Database Manager: http://sqlitebrowser.org/



SQLite Usage

- Can either create database from scratch in the app

OR

- Implement one into your app through a class

- Either way – you need to create a separate Database Manager Class
  - New class that extends SQLiteOpenHelper

- Can create a table(s) from scratch
  - Or read an existing database into the application

- DBManager also contains methods for accessing/modifying data
  - CRUD – Create Read Update Delete

- Activities/Fragments will use the DBManager

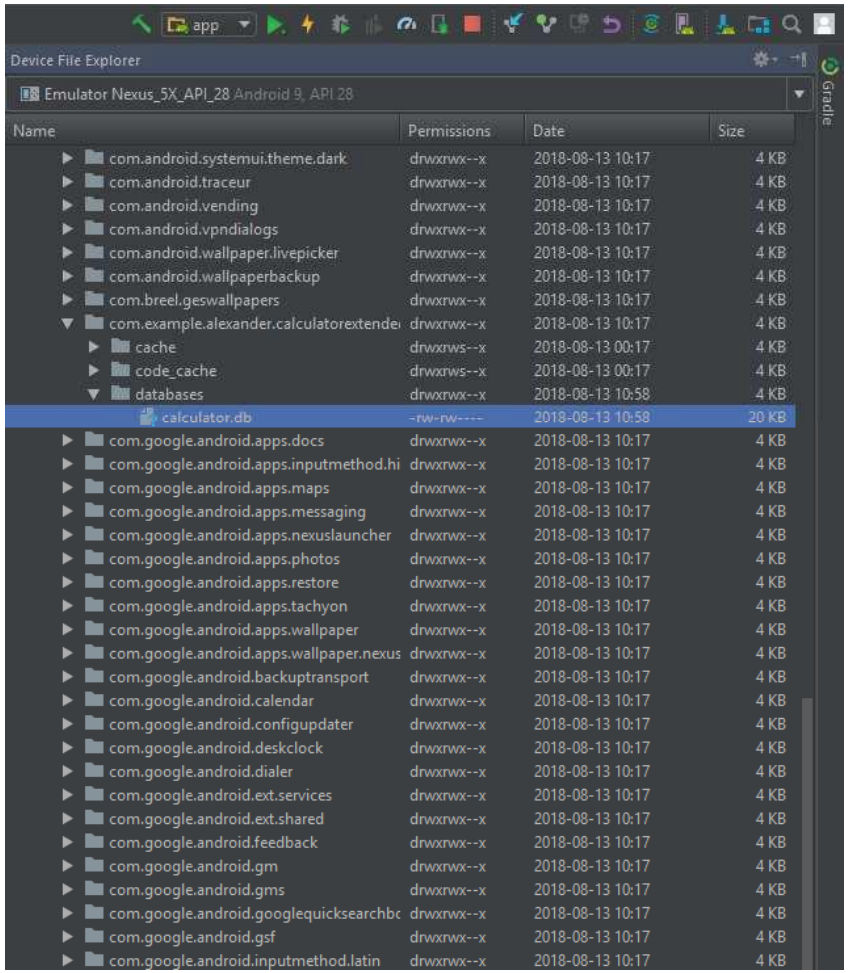
- Use Cursors to point to data entries

SQLite Database Access

- Accessing Databases is a bit of a pain
  - Won't be able to on your phone unless it is rooted

1. Run your application on the Emulator
2. In Android Studio: View > Tool Windows > Device File Explorer
3. Navigate to data > data > com.[your app name] >databases
4. Select your database.db and right click and Save As

- Issue with API 24, everything else is fine.
  - *Have your Emulator target API 23 if 24*



Charles Sturt University

Datastore – single, cross platform file (like an MS Access DB)

- Definitions
- Tables
- Indicies
- Data

Charles Sturt University

SQLite Data Types

This is quite different than the normal SQL data types so please read:

http://www.sqlite.org/datatype3.html

Charles Sturt University

NULL – null value

INTEGER - signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value

REAL - a floating point value, 8-byte IEEE floating point number.

TEXT - text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).

BLOB. The value is a blob of data, stored exactly as it was input.

Charles Sturt University

Charles Sturt University

SQLiteCloseable - An object created from a SQLiteDatabase that can be closed.

SQLiteCursor - A Cursor implementation that exposes results from a query on a SQLiteDatabase.

SQLiteDatabase - Exposes methods to manage a SQLite database.

SQLiteOpenHelper - A helper class to manage database creation and version management.

SQLiteProgram - A base class for compiled SQLite programs.

SQLiteQuery - A SQLite program that represents a query that reads the resulting rows into a CursorWindow.

SQLiteQueryBuilder - a convenience class that helps build SQL queries to be sent to SQLiteDatabase objects.

SQLiteStatement - A pre-compiled statement against a SQLiteDatabase that can be reused.

Charles Sturt University

- Contains the methods for: creating, opening, closing, inserting, updating, deleting and quering an SQLite database
- These methods are similar to JDBC but more method oriented than what we see with JDBC (remember there is not a RDBMS server running)

- This method will open an existing database or create one in the application data area

```
import android.database.sqlite.SQLiteDatabase;

   SQLiteDatabase    myDatabase;

myDatabase =  openOrCreateDatabase
   ("my_sqlite_database.db" ,

SQLiteDatabase.CREATE_IF_NECESSARY       , null);
```

Charles Sturt University

Important database configuration options include: version, locale, and thread-safe locking.

```
import java.util.Locale;

myDatabase.setVersion(1);
  myDatabase.setLockingEnabled(true);
  myDatabase.SetLocale(Locale.getDefault()
  );
```

```
String  createAuthor = "CREAT TABLE    authors (
                    id  INTEGER PRIMARY KEY AUTOINCREMENT,
                    fname  TEXT,
                    lname  TEXT);

myDatabase.execSQL(createAuthor);
```

```
import  android.content.ContentValues;

ContentValues  values = new  ContentValues( );
values.put("firstname" , "J.K."); values.put("lastname"
,               "Rowling");
long newAuthorID =      myDatabase.insert("tbl_authors"
, ""
,               values);
```

```
values, String whereClause, String[ ]
whereArgs)

public                                  void
updateBookTitle(Integer bookId, String
                                        newTitle)
{
ContentValues                           values = new
ContentValues(); values.put("title" , newTitle);
myDatabase.update("tbl_books" ,         values ,
"id=?"                                  , new String[ ]
                                        {bookId.toString()
} );
}
```

```
public void deleteBook(Integer  bookId)  {
     myDatabase.delete("tbl_books" , "id=?" ,
          new String[ ]   { bookId.toString( )  } )
;
}
```

Charles Sturt University

android.database

http://developer.android.com/reference/android/database/package-summary.html

Contains classes and interfaces to explore data returned through a content provider.

The main thing you are going to use here is the Cursor interface to get the data from the resultset that is returned by a query

http://developer.android.com/reference/android/database/Cursor.html

Charles Sturt University

Method of SQLiteDatabase class and performs queries on the DB and returns the results in a Cursor object

```
Cursor c =
      mdb.query(p1,p2,p3,p4,p5,p6,p7)
```

- p1 ; Table name (String)
- p2 ; Columns to return (String array)
- p3 ; WHERE clause (use null for all, ?s for selection args)
- p4 ; selection arg values for ?s of WHERE clause
- p5 ; GROUP BY (null for none) (String)
- p6 ; HAVING (null unless GROUP BY requires one) (String)
- p7 ; ORDER BY (null for default ordering)(String)
- p8 ; LIMIT (null for no limit) (String)

```
  mdb.query(abc,null,null,null,null,null,null);
```

SQL - "SELECT * FROM ABC WHERE C1=5"

SQLite - Cursor c = mdb.query(
abc,null,"c1=?" , new String[ ]
{"5"},null,null,null);

```
SQL - "SELECT title,id FROM BOOKS ORDER BY title ASC"
SQLite                  - String    {"title","id"};
colsToReturn            [ ]         = "title ASC";
String sortOrder Cursor c =
mdb.query("books",colsToReturn,
                  null,null,null,null,sortOrder);
```

Charles Sturt University