ANDROID

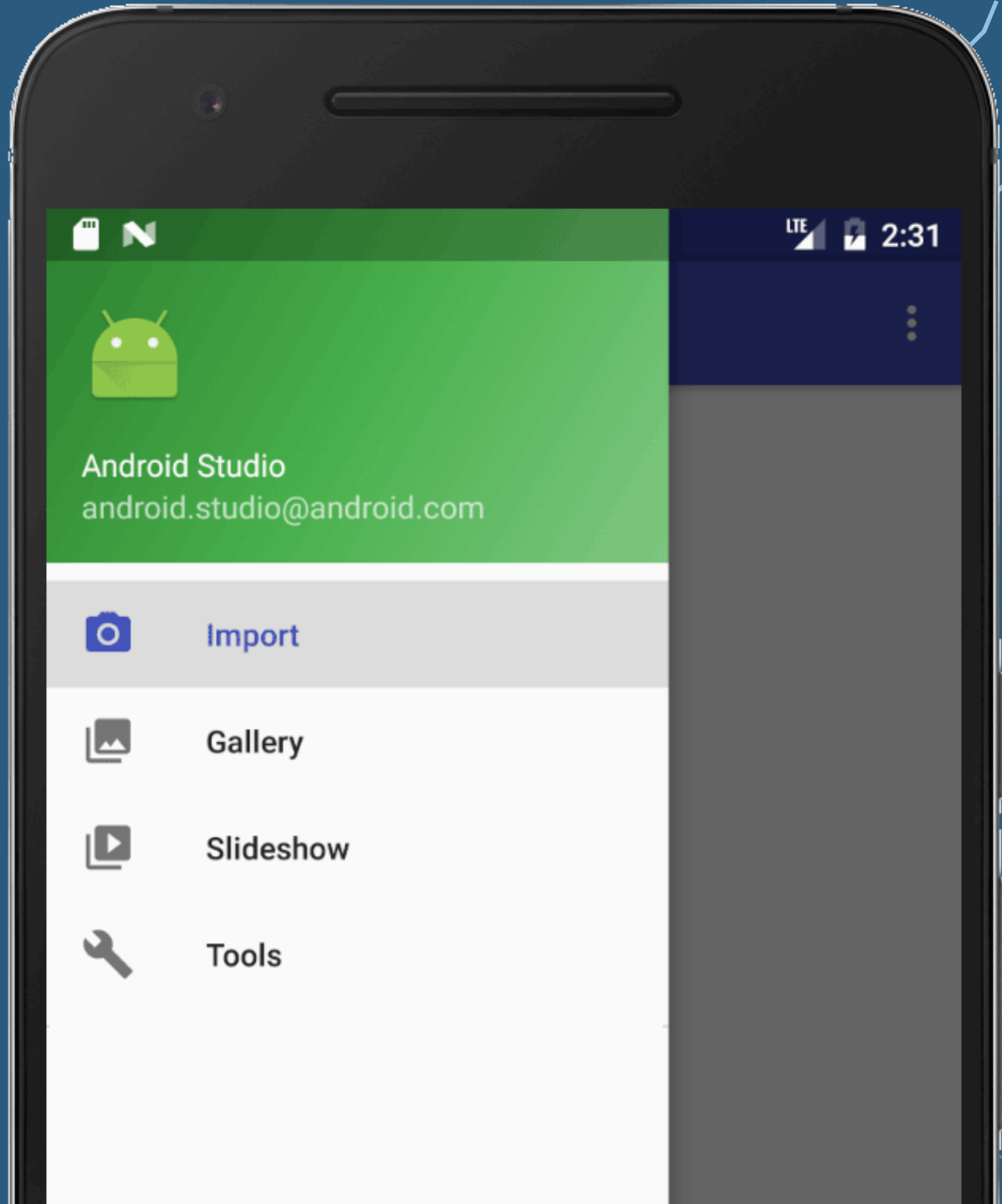
Navigation, Fragments, Notifications, Toasts, and searches

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

ANDROID NAVIGATION DRAWER

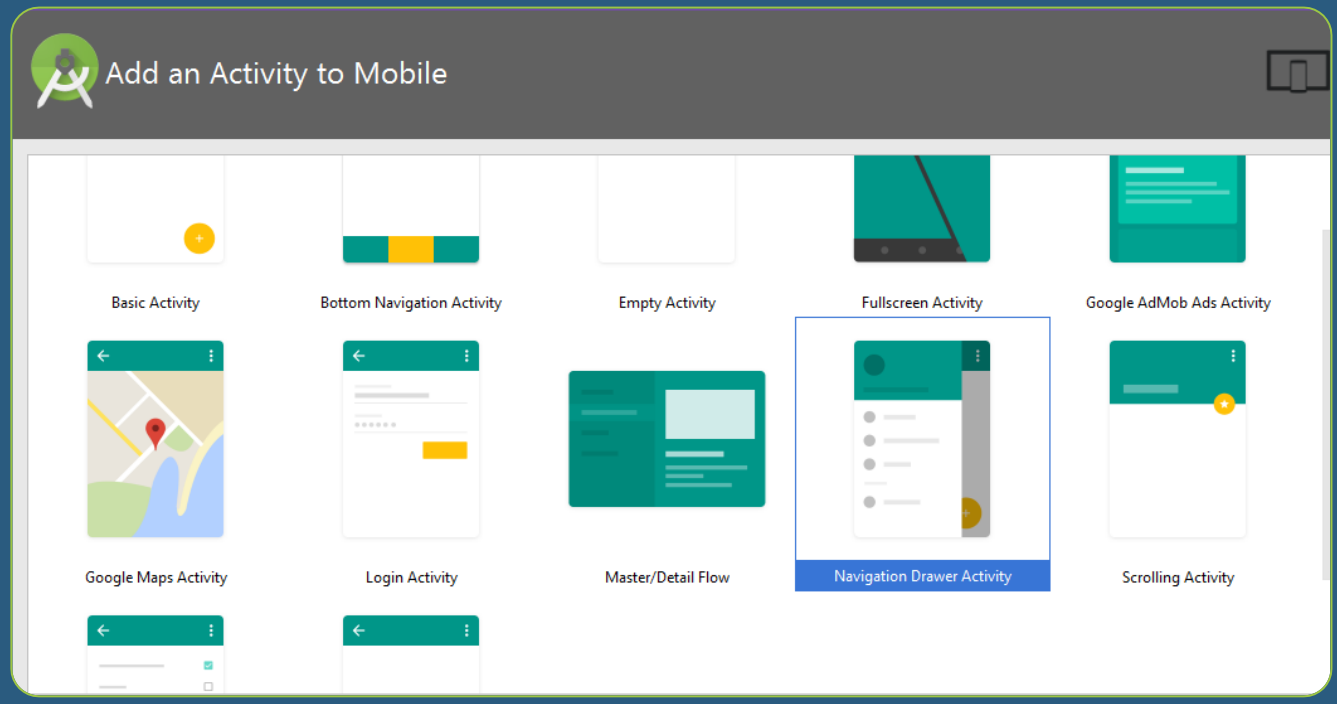
- The navigation drawer is a menu on the left-hand side of the screen
- Used to navigate between different activities in your app
  - Example: Home, Search, Account, Settings, Logout, etc
- To show the drawer, swipe from the left to the right
- You can design your own drawer, or just use Android's built-in template

ANDROID NAVIGATION DRAWER



ANDROID NAVIGATION DRAWER

- To use the navigation drawer, do one of the following
  - Create a new project with a Navigation Drawer Activity
  - Use and existing project and add a Navigation Drawer Activity
- To change the items that will display on your Navigation Drawer, edit
  - activity_main_drawer.xml

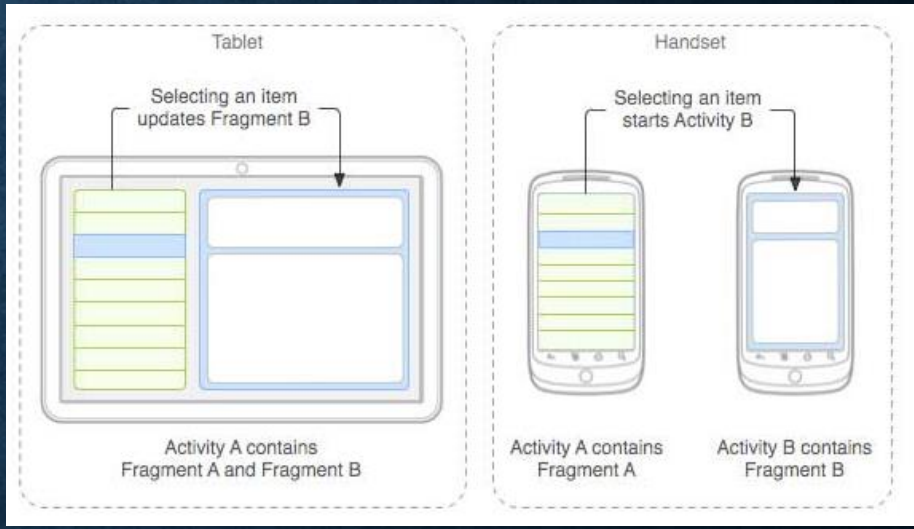


ANDROID FRAGMENTS

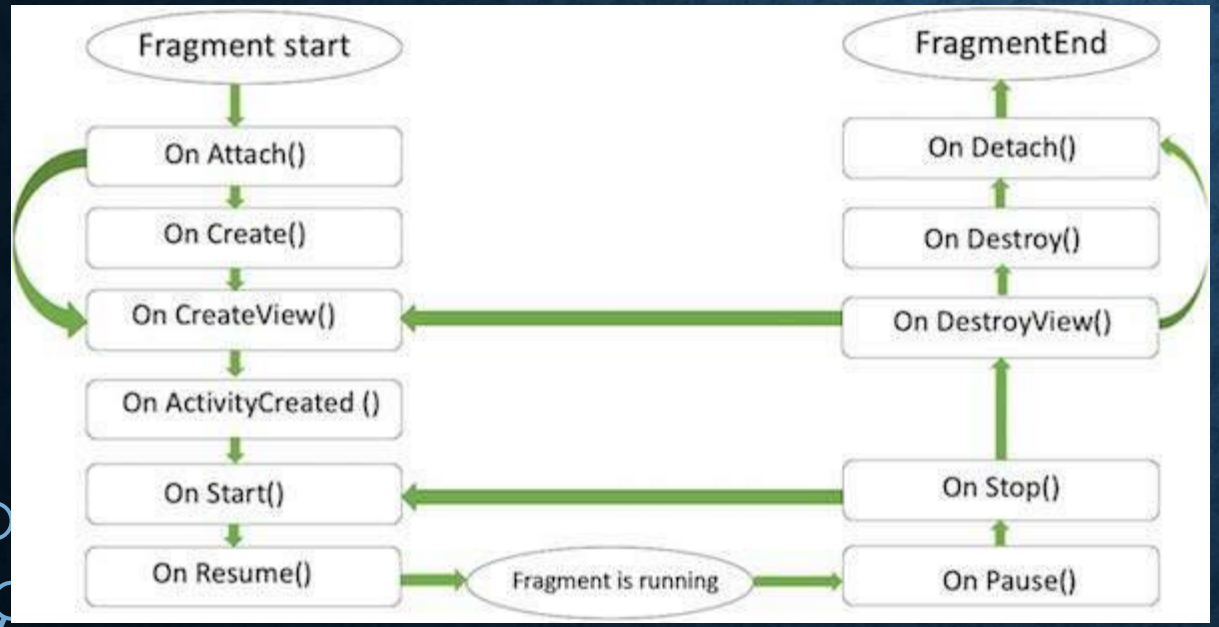
- For each of the options in your Navigation Drawer, you can use
  - A separate activity
  - Fragments
- Fragments can be seen as sub-activities
- Each activity can contain zero or more fragments

ANDROID FRAGMENTS

- Fragments can be reused across multiple activities
  - Can be dynamically added and removed from activities
- Fragments have their own lifecycle, separate from activities
  - However, still subject to the lifecycle of the parent activity
  - Example: If the activity is paused, all its child fragments are also paused
- The advantages of fragments is reuse and scaling/responsiveness for devices
  - Example: show 2 vertically stacked fragments on a phone, and show them horizontally (next to each other) on a tablet



ANDROID FRAGMENTS



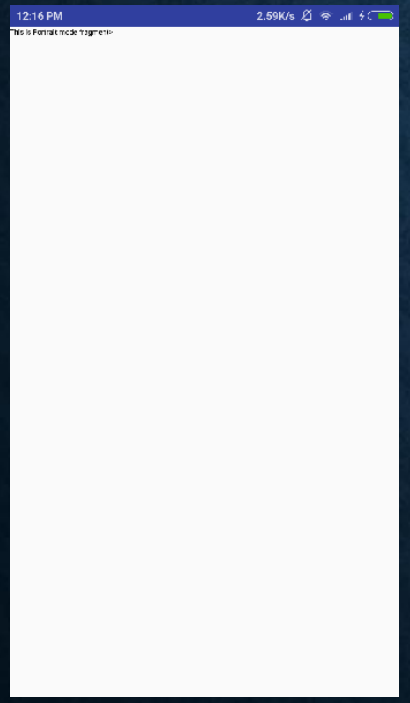
How to use a Fragment?

- How many fragments do you need?
- Next, create a class that will extend the Fragment Class. Here you need to override functions
- For each fragment you need an xml for the layout

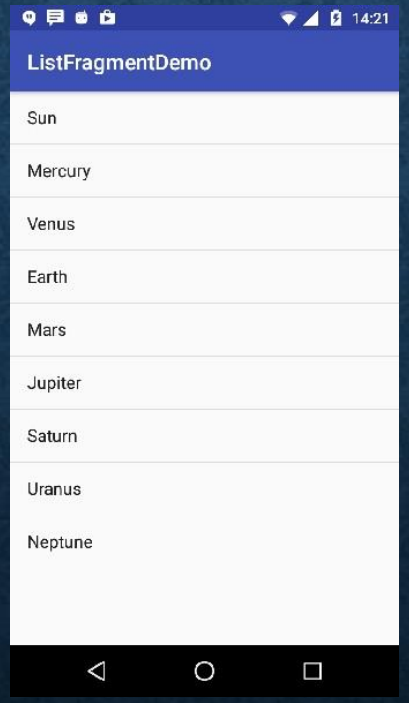
ANDROID FRAGMENTS

Types of Fragments

Frame Fragments



List Fragment



Fragment transitions

ANDROID FRAGMENTS

- Create a new FrameLayout
- For each option in the Navigation Drawer, if selected, load a different fragment into the layout
- Add code to content_main.xml

```
<FrameLayout
        android:id="@+id/main_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
/>
```

ANDROID FRAGMENTS

- Next, create a fragment for each of the screens you want to display
- In Android Studio
  - New > Fragment > Fragment (Blank)
- After creating a new fragment, delete all the automatically generate Java functions, except onCreateView()

ANDROID FRAGMENTS

- In MainActivity.java there is a onNavigationItemSelected function
  - Each time you select a navigation item, this function is fired
- Load a new fragment when the navigation is selected
  - MainFragment is your own customized fragment

```
public boolean onNavigationItemSelected(MenuItem item) {
        int id = item.getItemId();
        FragmentManager manager = getSupportFragmentManager();
        if(id == R.id.main_menu) {
                manager.beginTransaction()
                        .replace(R.id.main_container, new MainFragment())
                        .commit();
        }
}
```

ANDROID ACTIVITIES

- Instead of using Fragments for each screen, you can use activities instead
- Therefore, instead of using the FragmentManager, use Intents to open a new activity
- Activities are the main screen containing fragments, views and the Navigation Drawer
  - Hence, when loading a new activity, the old activity is unload
  - That old activity contains the Navigation Drawer, hence, the Navigation Drawer will also disappear
  - You will therefore have to add a Navigation Drawer to each individual activity
  - It is therefore easier and requires less code if you use Fragments

ANDROID BACK STACK

- All Android devices come with a back button
  - Hardware button on older phones/tablets
  - Software button on newer phones/tablets
  - Back button on remote controls for Android TV
- You should therefore not add your own back button to an app
- The back button can be used to go back to previous screens or exit the app if you are on the first screen
- Holding the back button for a few seconds kills the current app
  - Requires an option to be set in the Android settings

ANDROID BACK STACK

- Android provides default back navigation for many parts of the app
  - Specifically going back to previous activities
- Certain back operations or not handled by default and requires manual code to get it working
  - When the user enters a deep-level activity directly from a notification, an app widget, or the navigation drawer
  - Certain cases in which the user navigates between fragments
  - When the user navigates web pages in a WebView
  - Other exceptions, such as having a popup and first wanting to close the popup with the back button before tapping the back button again to go to the previous screen

ANDROID BACK STACK

- If you are using Fragments, you can use the FragmentManager's back stack

```
if(id == R.id.settings_fragment)
{
        manager.beginTransaction()
                .replace(R.id.main_container, new SettingsFragment())
                .addToBackStack("SettingsFragment")
                .commit();
}
```

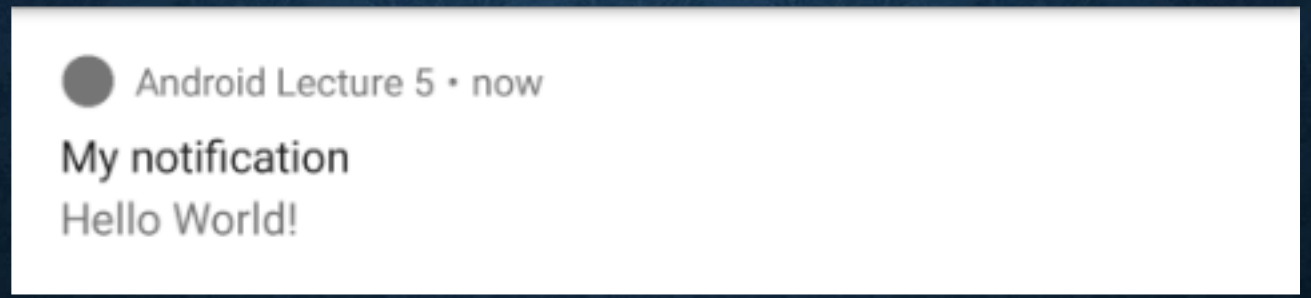
ANDROID NOTIFICATIONS

- Notifications are user interface elements that are displayed outside the app's normal UI
- Users can choose to view the notification or respond to them when it is convenient
- When the notification is clicked, typically the app is launched to a specific activity or fragment
- For example, and email client might show a notification if a new email arrives
  - Requires a background service or schedule job to check for new emails

ANDROID NOTIFICATIONS

- Notifications have an icon, title, and message

```
NotificationCompat.Builder builder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hellow World!");
```



ANDROID NOTIFICATIONS

- The notification allows the user to go directly to a specific activity in your app
- This action is defined as a PendingIntent containing an Intent that starts an activity in your app
- A PendingIntent object performs an action on your app's behalf
  - Often the action is executed at a later stage
  - Irrespective of whether or not your app is currently running

ANDROID NOTIFICATIONS

```
Intent result = new Intent(this, MainActivity.class);
TaskStackBuilder stack = TaskStackBuilder.create(this);
stack.addParentStack(MainActivity.class);
stack.addNextIntent(result);

PendingIntent pendingResult =
        stack.getPendingIntent(0, PendingIntent.FLAG_UPDATE_CURRENT);
builder.setContentIntent(pendingResult);
```

ANDROID NOTIFICATIONS

- After creating the notification and adding the optional action, you need to
  - Build the notification
  - Notify the NotificationManager of the new notification

```
NotificationManager notifier =
        (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
notifier.notify(notificationID, builder.build());
```

ANDROID NOTIFICATIONS

- You can load a specific fragment with a notification
- Update your Intent

```
Intent result = new Intent(this, MainActivity.class);
result.putExtra("menuFragment", "settingsPage");
```

- Then determine which of the fragments should be loaded if the activity, inside the activitie's onCreate() function launches

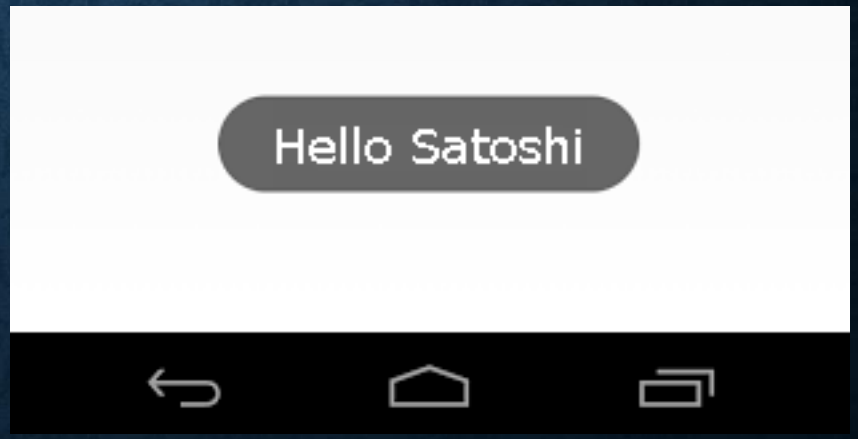
ANDROID NOTIFICATIONS

```
String fragment = getIntent().getString("menuFragment");
FragmentManager manager = getSupportFragmentManager();
if(fragment != null)
{
        if(fragment.equals("settingsPage"))
        {
                manager.beginTransaction()
                .replace(R.id.main_container, new SettingsFragment())
                .commit();
        }
}
```

ANDROID TOASTS

- Toasts are small popups with notification messages
- Toasts UI size is just large enough to contain the message
- The underlying activity remains active with the toasts shows

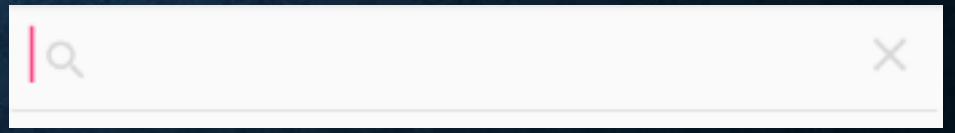
```
Context context = getApplicationContext();
CharSequence text = "Hello Statoshi";
int duration = Toast.LENGTH_SHORT;
Toast.makeText(context, text, duration).show();
```



ANDROID SEARCHES

- A SearchView provides a search functionality popup/toolbar
- Can be used to input some text and search that text
  - Example: Send a search query to your MySQL database

```
<android.support.v7.widget.SearchView
        android:id="@+id/searcher"
        android:layout_width="match_parent“
        android:layout_height="match_parent"
        android:clickable="true"
/>
```



ANDROID SEARCHES

- Then add some Java code that is executed on a search

```
view = (SearchView) view.findViewById(R.id.searcher);
view.setOnQueryTextListener(new SearchView.OnQueryTextListener() {

        @Override public boolean onQueryTextChange(String text) {
                // Search while typing, eg predictive text
                return false;
        }
        @Override public boolean onQueryTextSubmit(String text) {
                if(text.length > 0) {
                        // Do some search
                        return true;
                }
                return false;
        }
});
```