ANDROID GUIDE

Contents

[How to add Up Button in your app: 3](#_Toc523171994)

[**Intro:** 3](#_Toc523171995)

[**Working:** 3](#_Toc523171996)

[**Example:** 3](#_Toc523171997)

[**Reference:** 4](#_Toc523171998)

# How to add Up Button in your app:

**Intro:**

This consists of parent/child relationship. There is a difference between up button and back button. The “Back” button is part of the system navigation bar on Android (leftmost triangle icon). No matter which app you’re in, when you tap the “Back” button, you’ll go back to where you previously came from.



However, there are certain cases where “Back” and “Up” result in different behavior. The “Up” button ALWAYS leads you to the parent activity. The “Back” button can lead you to the parent activity, or the home screen, or to another app, depending on how you arrived at the current screen.

**Working:**

First we need to create two activities for example one is parent and second is child for creating the parent/child relationship. Then add getSupportActionBar().setDisplayHomeAsUpEnabled(**true**);

In your child activity on oncreate also do not forget to add in manifest of parent activity name in your child activity.

**Example:**

Parent Activity.xml:

<**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Another Activity"  
 android:onClick="anotherActivity"  
 android:layout\_gravity="center"**/>

Parent Activity.class:

**public void** anotherActivity(View view){  
 *//Toast.makeText(this, "works", Toast.LENGTH\_SHORT).show();* Intent i = **new** Intent(**this**,ChildActivity.**class**);  
 startActivity(i);  
}

Child Activity.class:

@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_child***);  
 getSupportActionBar().setDisplayHomeAsUpEnabled(**true**);  
}

Manifest:

<**activity android:name=".ChildActivity"  
 android:parentActivityName=".MainActivity"**>  
</**activity**>

**Reference:**

<https://developer.android.com/training/implementing-navigation/ancestral#java>

<https://github.com/yasir-raza/Android-Guide/tree/master/UpButton>

# Fragment View Pager:

**Intro:**

A [Fragment](https://developer.android.com/reference/android/support/v4/app/Fragment.html) represents a behavior or a portion of user interface in a [Fragment Activity](https://developer.android.com/reference/android/support/v4/app/FragmentActivity.html). You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities. You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).

When you add a fragment as a part of your activity layout, it lives in a [View Group](https://developer.android.com/reference/android/view/ViewGroup.html) inside the activity's view hierarchy and the fragment defines its own view layout. You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element, or from your application code by adding it to an existing [View Group](https://developer.android.com/reference/android/view/ViewGroup).

The View Pager works by getting its data from an adapter - called a Fragment Pager Adapter.  We want to customize the adapter to display our own fragments, so we have to use inheritance to subclass the Fragment Pager Adapter. By inheriting, we get all the functionality from the Fragment Pager Adapter for free, and we can add our own customization on top of it.

**Working:**

First we need to create a new project then add view pager in Main Activity xml then create a new class for example let’s name it as Simple Fragment Pager Adapter and extends it to Fragment Pager Adapter then implement override methods and constructor of it then create two new blank fragments lets name it as Monday Fragment & Tuesday Fragment. Now let’s create a view pager adapter object and simple fragment pager adapter object in Main Activity in on create function and set the simple fragment pager adapter into view pager. In Simple Fragment Pager Adapter Class update an override function of get item and get count. In get item function you can use it switch cases or if/else condition and return your fragment respectively and update get count function return depending upon the number of fragments you have.

We create the Simple Fragment Pager Adapter class and extend from the Fragment Pager Adapter class. When you launch the app on your device, first the View Pager asks the adapter how many pages there will be. In order for the View Pager to display page 0, the View Pager asks the adapter for the 0th fragment. See the Simple Fragment Pager Adapter get Item (int position) method. When the user swipes leftward, we move onto page 1, which means the View Pager asks the adapter for the fragment at position 1. When we get to page 2, the View Pager asks the adapter for the fragment at position 2. Thus, depending on which page (also known as position), the user has swiped to, the corresponding fragment gets shown.

**Example:**

MainActivity.xml

<**android.support.v4.view.ViewPager  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/viewPager"**>  
</**android.support.v4.view.ViewPager**>

MainActivity.class (on create function)

ViewPager viewPager = findViewById(R.id.***viewPager***);  
SimpleFragmentPagerAdapter adapter = **new** SimpleFragmentPagerAdapter(getSupportFragmentManager());  
viewPager.setAdapter(adapter);

SimpleFragmentPagerAdapter.class

@Override  
**public** Fragment getItem(**int** position) {  
 **if**(position == 0)  
 {  
 **return new** MondayFragment();  
 }  
 **else** {  
 **return new** TuesdayFragment();  
 }  
}  
@Override  
**public int** getCount() {  
 **return** 2;  
}

**Note: There is nothing inside multiple fragments except whatever you want in that activity.**

**Reference:**

[**https://developer.android.com/guide/components/fragments**](https://developer.android.com/guide/components/fragments)