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>

# View Pager:

**Intro:**

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

We create the SimpleFragmentPagerAdapter class and extend from the FragmentPagerAdapter class. When you launch the app on your device, first the ViewPager asks the adapter how many pages there will be. In order for the ViewPager to display page 0, the ViewPager asks the adapter for the 0th fragment. See the SimpleFragmentPagerAdapter getItem(int position) method. When the user swipes leftward, we move onto page 1, which means the ViewPager asks the adapter for the fragment at position 1. When we get to page 2, the ViewPager 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.