



ANDROID FUNDAMENTALS

Seven Advanced Academy



Screen Navigation



Lesson 12

Contents

- Back navigation
- Hierarchical navigation
 - Up navigation
 - Descendant navigation
 - Navigation drawer for descendant navigation
 - Lists and carousels for descendant navigation
 - Ancestral navigation
 - Lateral navigation



Tasks

- Continue adding features to the Droid Cafe project from the previous practical.
- Provide the Up button in the app bar to navigate to the previous screen within an activity.
- Create a new app with tabs for navigating activity screens that can also be swiped.



Two forms of navigation

Temporal or back navigation

- provided by the device's back button
- controlled by the Android system's back stack

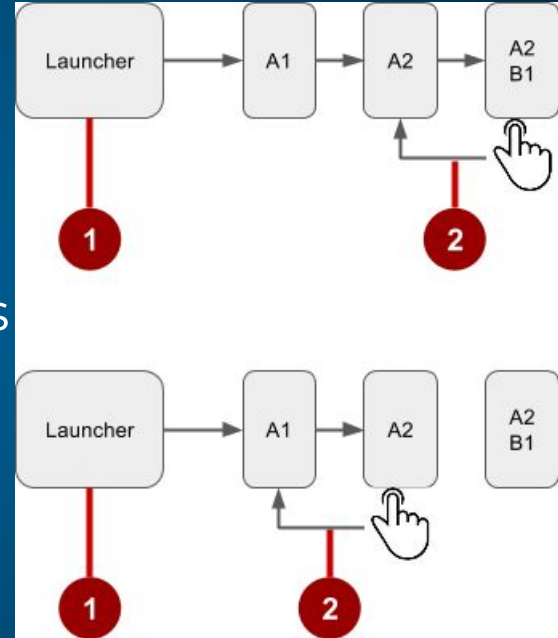
Ancestral or up navigation

- provided by the app's action bar
- controlled by defining parent-child relationships between activities in the Android manifest

Back Navigation

Navigation through history of screens

1. History starts from Launcher
2. User clicks the Back  button to navigate to the previous screens in reverse order



Changing Back button behavior

- Android system manages the back stack and Back button
- If in doubt, don't change
- Only override, if necessary to satisfy user expectation

For example: In an embedded browser, trigger browser's default back behavior when user presses device Back button



Overriding onBackPressed()

```
@Override  
public void onBackPressed() {  
    // Add the Back key handler here.  
    return;  
}
```



Hierarchical Navigation

Hierarchical navigation patterns

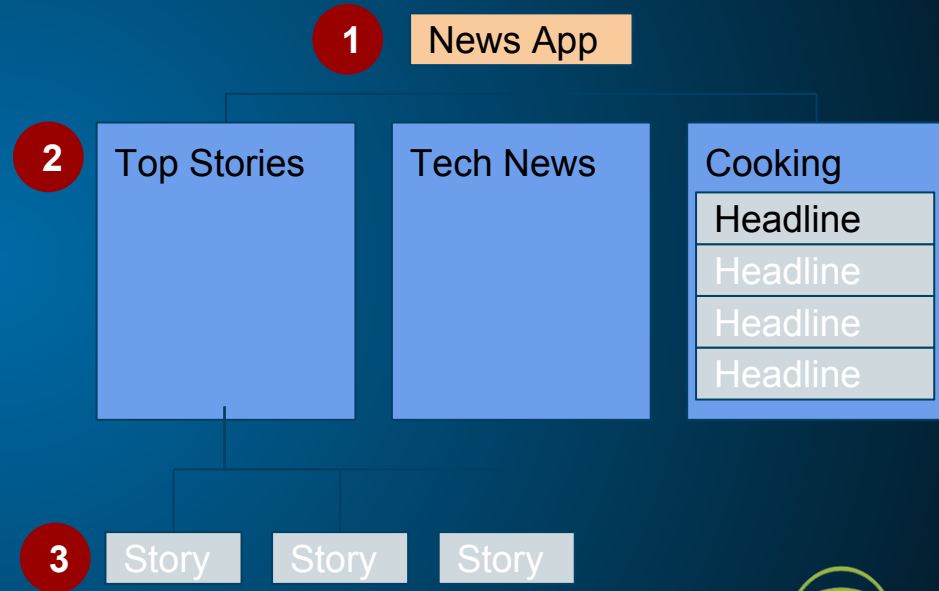
- Parent screen—Screen that enables navigation down to child screens, such as home screen and main activity
- Collection sibling—Screen enabling navigation to a collection of child screens, such as a list of headlines
- Section sibling—Screen with content, such as a story



Example of a screen hierarchy

1. Parent screen
2. Children: collection siblings
3. Children: section siblings

Use activities or fragments to implement a hierarchy



Types of hierarchical navigation

- Descendant navigation
 - Down from a parent screen to one of its children
 - From a list of headlines to a story summary to a story
- Ancestral navigation
 - Up from a child or sibling screen to its parent
 - From a story summary back to the headlines
- Lateral navigation
 - From one sibling to another sibling
 - Swiping between tabbed views



Descendant Navigation

Descendant navigation

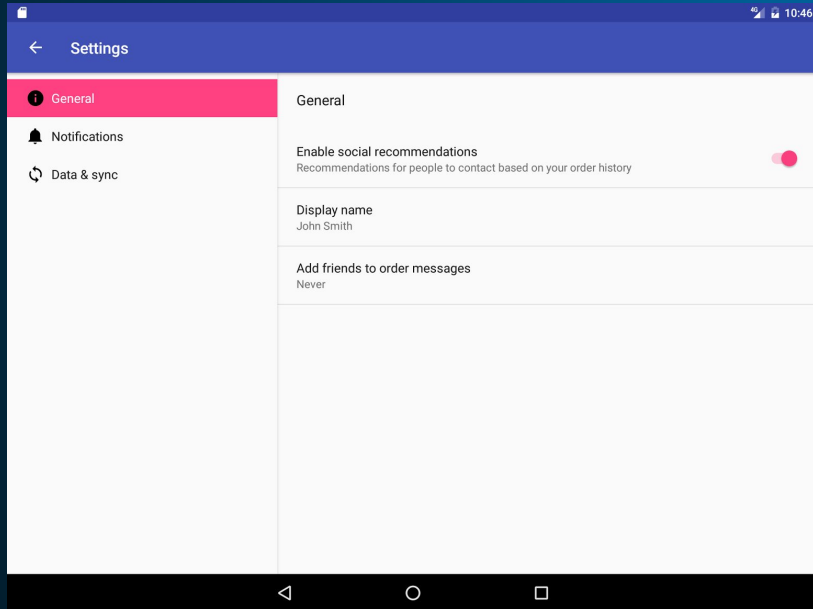
Descendant navigation

- Down from a parent screen to one of its children
- From the main screen to a list of headlines to a story

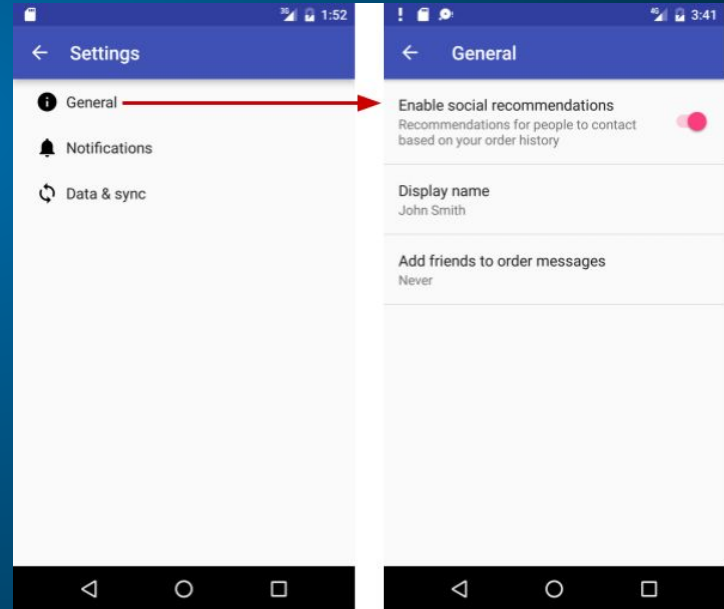


Master/detail flow

Side-by side on tablets



Multiple screens on phone



Controls for descendant navigation

- Buttons, image buttons on main screen
- Other clickable views with text and icons
- Arranged in horizontal or vertical rows, or as a grid
- List items on collection screens



Navigation Drawer for Descendant Navigation

Create menu resource

Create XML menu resource (menu_context.xml)

```
<item
    android:id="@+id/context_edit"
    android:title="@string/edit"
    android:orderInCategory="10"/>

<item
    android:id="@+id/context_share"
    android:title="@string/share"
    android:orderInCategory="20"/>
```



Register a view to a context menu

- in onCreate() of the activity
- registers [View.OnCreateContextMenuListener](#)
- Does not specify which context menu!

```
TextView article_text = (TextView)  
findViewById(R.id.article);  
registerForContextMenu(article_text);
```



Implement onCreateContextMenu()

Specifies which context menu

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenu.ContextMenuInfo menuInfo)
{
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_context, menu);
}
```

Implement onOptionsItemSelected()

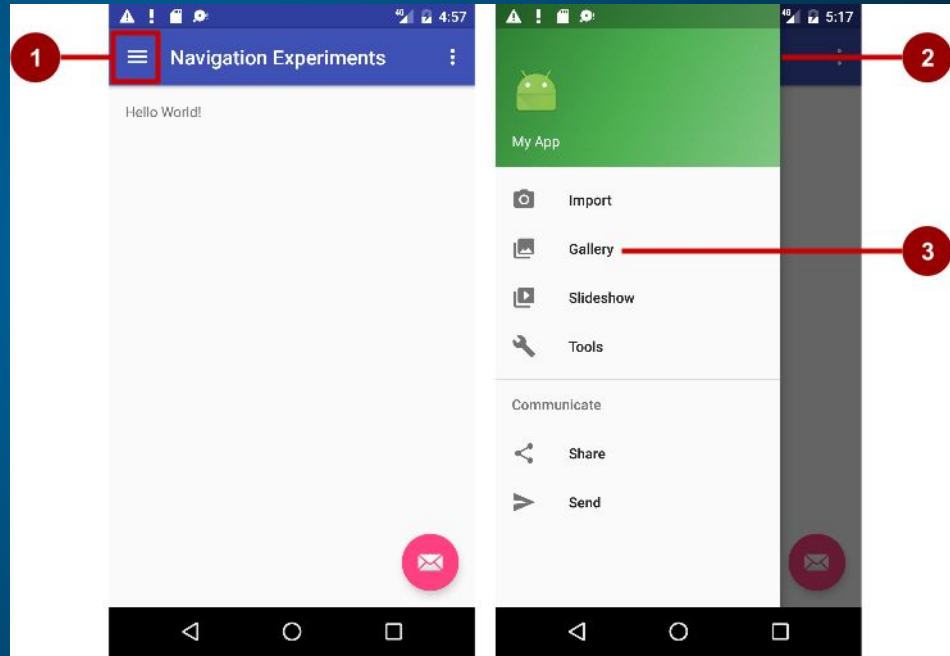
```
@Override
public boolean onOptionsItemSelected(MenuItem
item) {
    switch (item.getItemId()) {
        case R.id.context_edit:
            editNote();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```



Contextual Action Bar

Navigation drawer

1. Icon in app bar
2. Header
3. Menu item



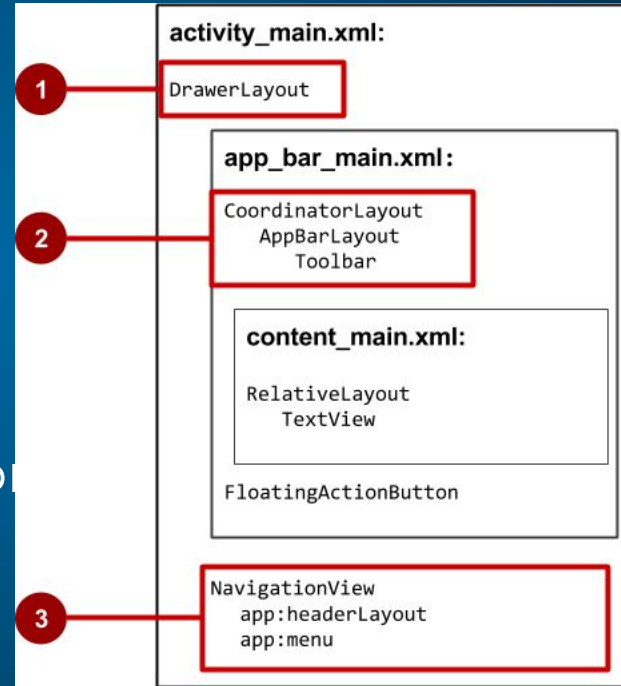
Steps for navigation drawer

1. Create layouts for drawer, drawer header, drawer menu items, app bar, activity screen contents
2. Add navigation drawer and item listeners to activity code
3. Handle the navigation drawer menu item selections



Navigation drawer activity layout

1. DrawerLayout is root view
2. CoordinatorLayout contains app bar layout with a Toolbar
3. App content screen layout
4. NavigationView with layouts for header and selectable items



Other descendant navigation patterns

- Vertical list, such as [RecyclerView](#)
- Vertical grid, such as [GridView](#)
- Lateral navigation with a Carousel
- Multi-level menus, such as the Options menu
- Master/detail navigation flow

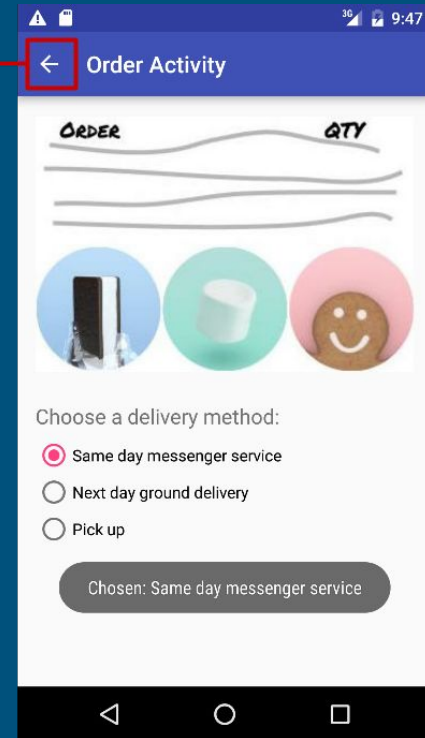


Ancestral Navigation

Ancestral navigation (Up button)



Enable user to go up from a section or child screen to the parent



Declare activity's parent in Android manifest

```
<activity android:name=".OrderActivity"
    android:label="@string/title_activity_order"
    android:parentActivityName="com.example.android.
        optionsmenuorderactivity.MainActivity">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity"/>
</activity>
```



Lateral Navigation

Tabs and swipes

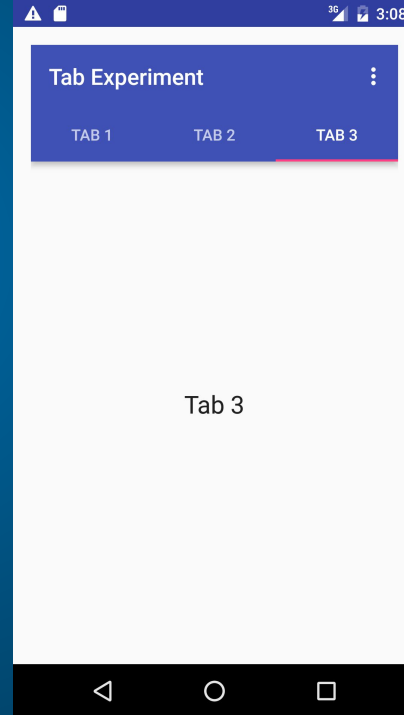
Lateral navigation

- Between siblings
- From a list of stories to a list in a different tab
- From story to story under the same tab



Benefits of using tabs and swipes

- A single, initially-selected tab—users have access to content without further navigation
- Navigate between related screens without visiting parent



Best practices with tabs

- Lay out horizontally
- Run along top of screen
- Persistent across related screens
- Switching should not be treated as history



Steps for implementing tabs

1. Define the tab layout using `TabLayout`
2. Implement a fragment and its layout for each tab
3. Implement a `PagerAdapter` from `FragmentPagerAdapter` or `FragmentStatePagerAdapter`
4. Create an instance of the tab layout
5. Manage screen views in fragments
6. Set a listener to determine which tab is tapped



Add tab layout below Toolbar

```
<android.support.design.widget.TabLayout  
    android:id="@+id/tab_layout"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_below="@id/toolbar"  
    android:background="?attr/colorPrimary"  
    android:minHeight="?attr/actionBarSize"  
  
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/  
>
```



Add view pager below TabLayout

```
<android.support.v4.view.ViewPager  
    android:id="@+id/pager"  
    android:layout_width="match_parent"  
    android:layout_height="fill_parent"  
    android:layout_below="@id/tab_layout" />
```



Create a tab layout in onCreate()

```
TabLayout tabLayout = (TabLayout)
findViewById(R.id.tab_layout);
tabLayout.addTab(tabLayout.newTab().setText("Tab 1"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 2"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 3"));
tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);
```



Add the view pager in onCreate()

```
final ViewPager viewPager = (ViewPager)
findViewById(R.id.pager);
final PagerAdapter adapter = new PagerAdapter (
    getSupportFragmentManager(), tabLayout.getTabCount());
viewPager.setAdapter(adapter);
```



Add the listener in onCreate()

```
viewPager.addOnPageChangeListener(  
    new  
    TabLayout.TabLayoutOnPageChangeListener(tabLayout));  
tabLayout.addTabSelectedListener(  
    new TabLayout.OnTabSelectedListener() {  
        @Override  
        public void onTabSelected(TabLayout.Tab tab) {  
            viewPager.setCurrentItem(tab.getPosition());}  
        @Override  
        public void onTabUnselected(TabLayout.Tab tab) {}  
        @Override  
        public void onTabReselected(TabLayout.Tab tab) {}  
    });
```



Let's code

App overview

In the previous practical you created an app called Droid Cafe in three parts, using the Basic Activity template. This template also provides an app bar at the top of the screen. You will learn how to add an Up button (a left-facing arrow) to the app bar for up navigation from the second activity (`OrderActivity`) to the main activity (`MainActivity`). This will complete the Droid Cafe app.

Demo : Using the App Bar and Tabs for Navigation



Source code

- [DroidCafePart3](#)
- [DroidCafe](#)
- [TabExperiment](#)
- [NavDrawerExperiment](#)



Learn More

- Navigation Design guide
d.android.com/design/patterns/navigation.html
- Designing effective navigation
d.android.com/training/design-navigation/index.html
- Creating a Navigation Drawer
d.android.com/training/implementing-navigation/nav-drawer.html
- Creating swipe views with tabs
d.android.com/training/implementing-navigation/lateral.html

