Android Developer Fundamentals V2

User Interaction

Lesson 4

4.4 User navigation

Contents

Two forms of navigation

◁ Back (temporal) navigation

- Provided by the device's Back button
- Controlled by the Android system back stack

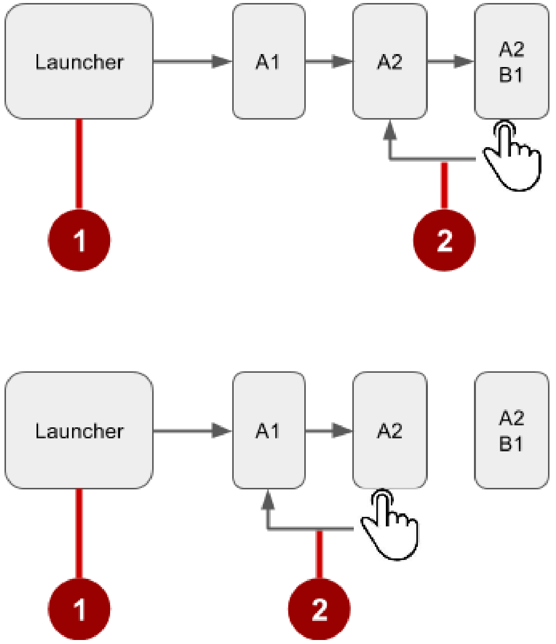
← Ancestral (Up) navigation

- Up button provided in app bar
- Controlled by defining parent `Activity` for child `Activity` in the AndroidManifest.xml

Back Navigation

Navigation through history of screens

1. Historys starts from Launcher
2. User clicks the Back ◁ button to navigate to 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

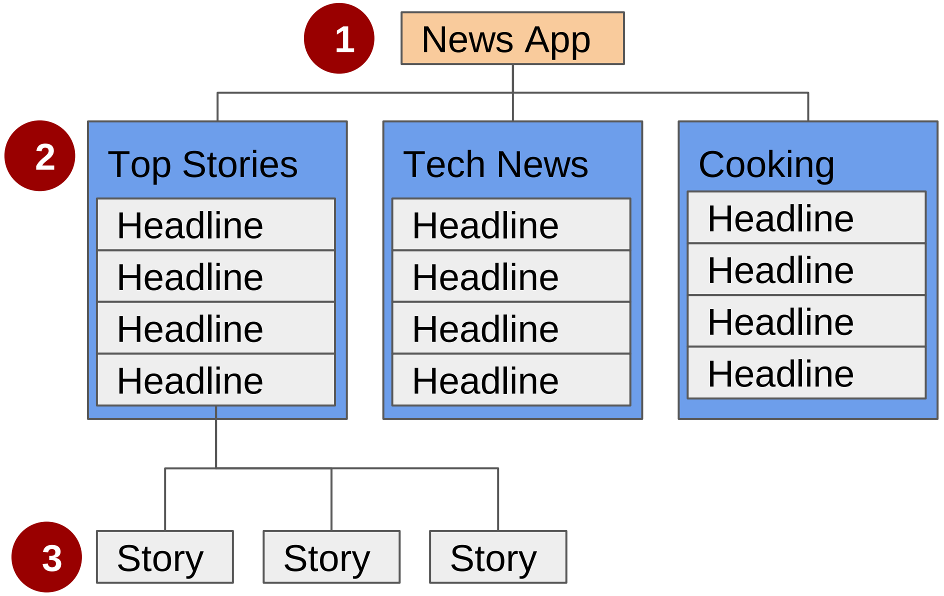
Hierarchical navigation patterns

- **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
2. Children: collection siblings
3. Children: section siblings

- Use `Activity` for parent screen
- Use `Activity` or `Fragment` for children screens



Types of hierarchical navigation

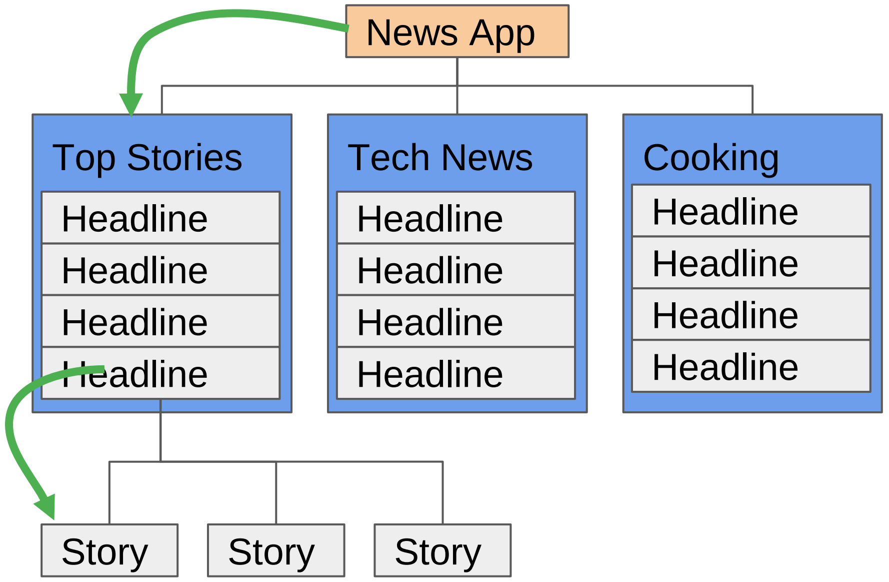
- Descendant navigation
  - Down from a parent screen to one of its children
  - From a list of headlines—to a story summary—to a story
- Ancestral navigation
  - Up from a child or sibling screen to its parent
  - From a story summary back to the headlines
- Lateral navigation
  - From one sibling to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

Descendant Navigation

Descendant navigation

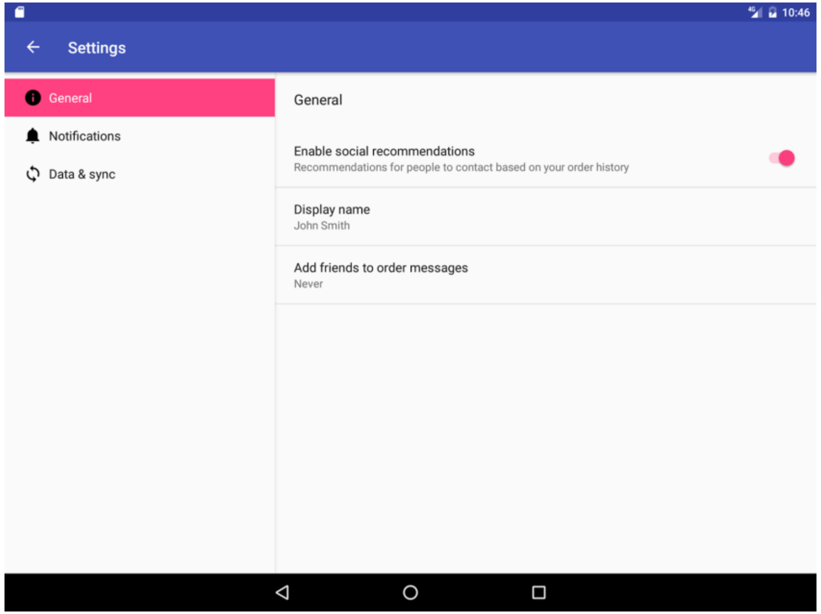
Descendant navigation

- 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


10:46

Settings

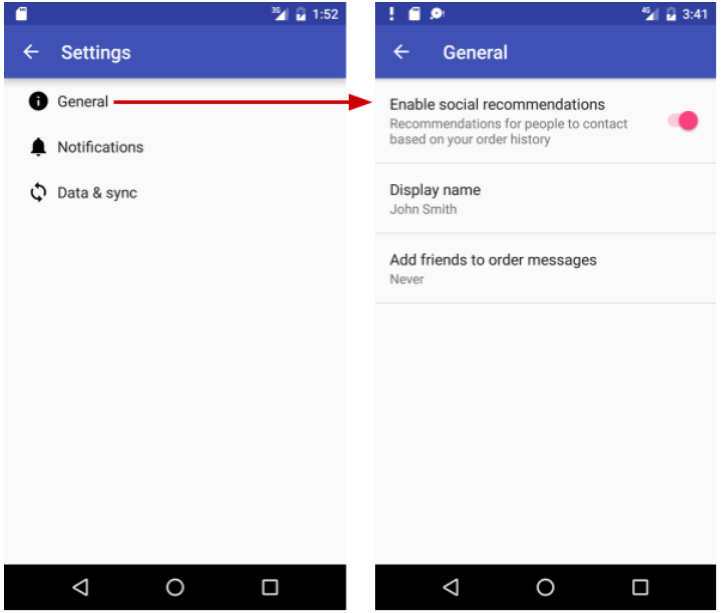
General
Notifications
Data & sync

General

Enable social recommendations
Recommendations for people to contact based on your order history

Display name
John Smith

Add friends to order messages
Never


1:52

Settings

General
Notifications
Data & sync

3:41

General

Enable social recommendations
Recommendations for people to contact based on your order history

Display name
John Smith

Add friends to order messages
Never

Controls for descendant navigation

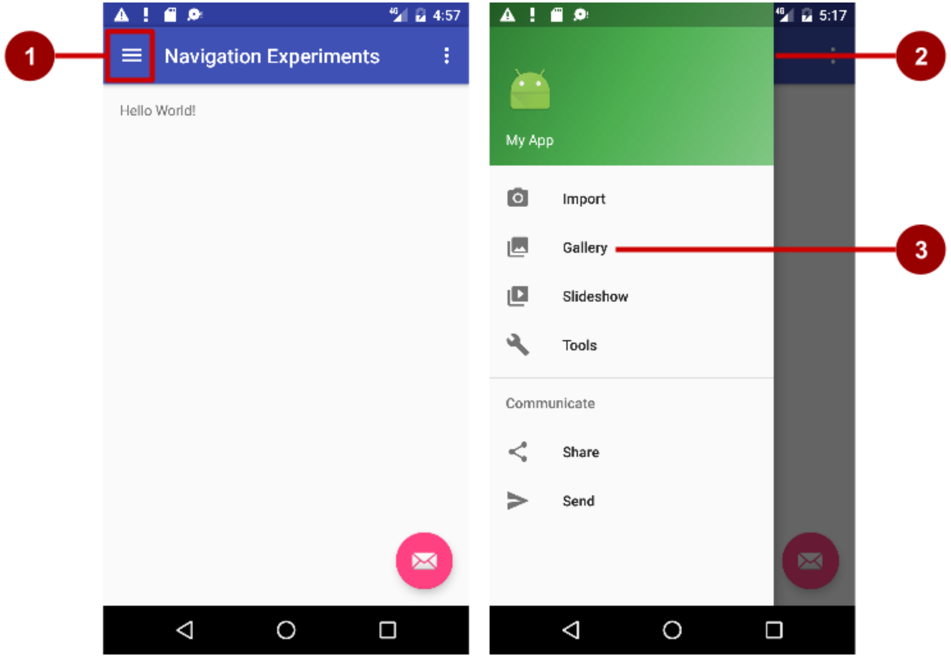
- Navigation drawer
- Buttons, image buttons on main screen
- Other clickable views with text and icons arranged in horizontal or vertical rows, or as a grid
- List items on collection screens

Navigation Drawer

Navigation drawer

Descendant navigation

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



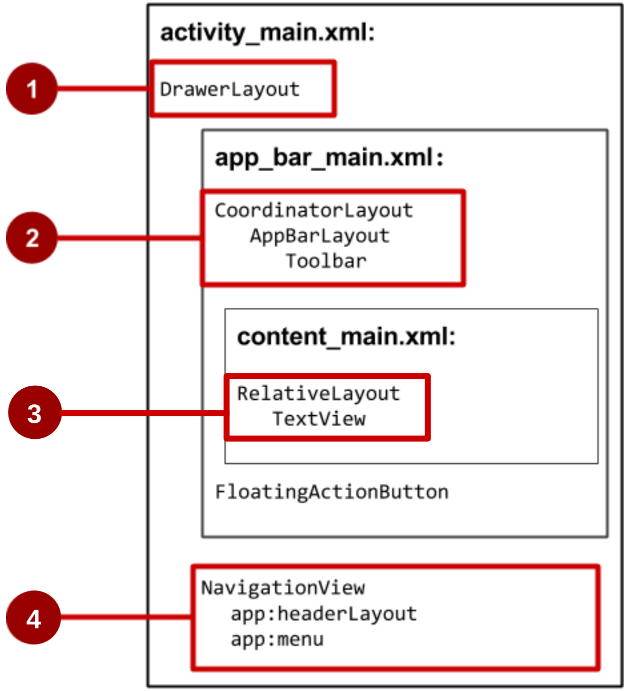
Layouts for for navigation drawer

Create layouts:

- A navigation drawer as the `Activity` layout root `ViewGroup`
- A navigation View for the drawer itself
- An app bar layout that includes room for a navigation icon button
- A content layout for the `Activity` that displays the navigation drawer
- A layout for the navigation drawer header

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



Steps to implement navigation drawer

1. Populate navigation drawer menu with item titles and icons
2. Set up navigation drawer and item listeners in the `Activity` code
3. Handle the navigation menu item selections

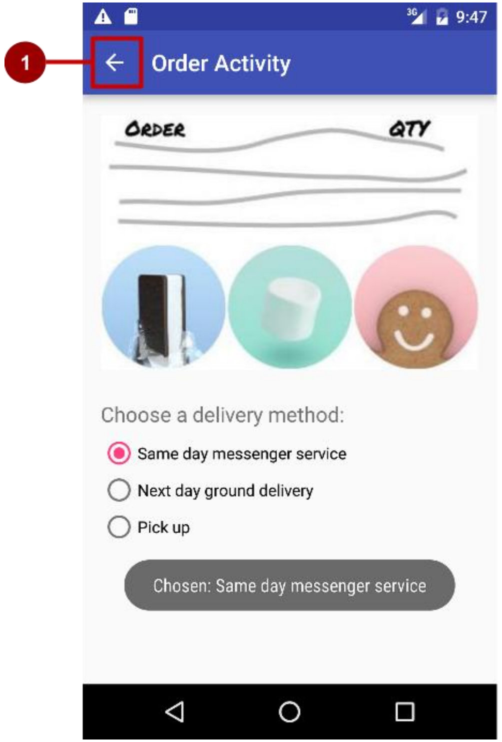
Other descendant navigation patterns

- 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



Choose a delivery method:

- [✓ Same day messenger service]
- Next day ground delivery
- Pick up

Chosen: Same day messenger service

Declare parent of child Activity—AndroidManifest

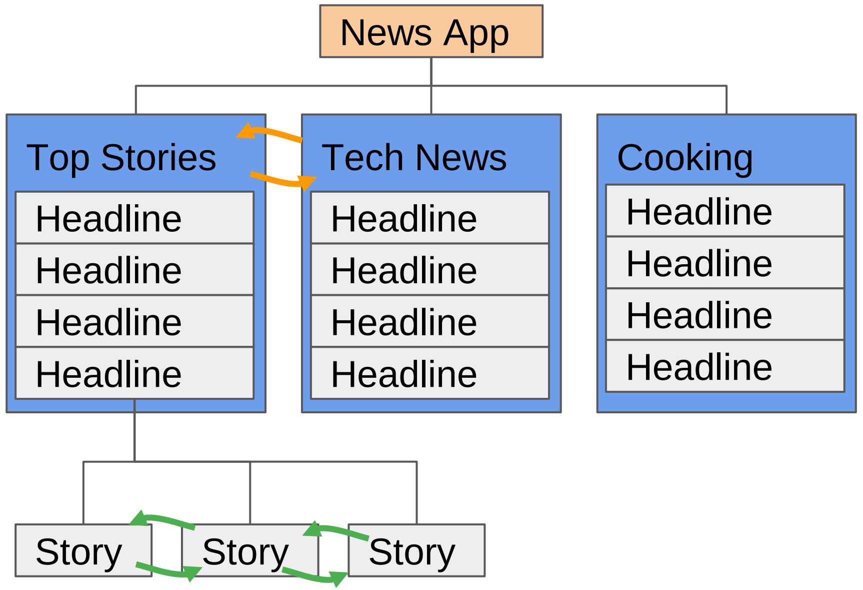
```
<activity android:name=".OrderActivity"
  android:label="@string/title_activity_order"
  android:parentActivityName="com.example.android.
                      optionsmenuorderactivity.MainActivity">
  <meta-data
     android:name="android.support.PARENT_ACTIVITY"
     android:value=".MainActivity"/>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

Lateral Navigation

Tabs and swipes

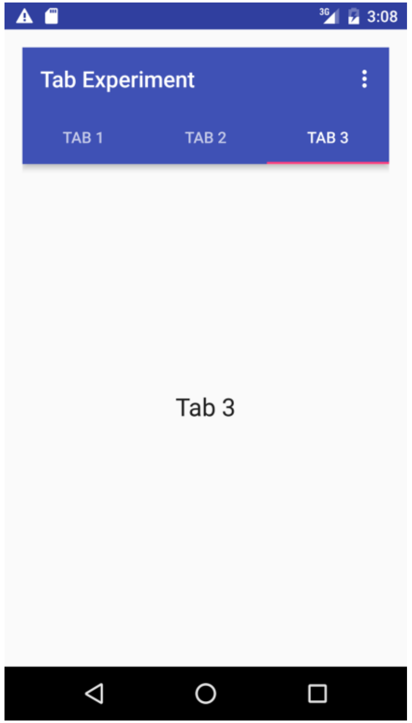
Lateral navigation

- 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. Use `PagerAdapter` to manage screens (each screen is a `Fragment`)
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 = 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 = 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.addOnTabSelectedListener(
            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) {} });
```

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

What's Next?

- Concept Chapter: 4.4 User navigation
- Practical: 4.4 User navigation

END