Android Developer Fundamentals V2

Preferences and settings

Lesson 9

9.2 App settings

Contents

- What are settings?
- Setting screens
- Implement settings
- Default settings
- Save and retrieve settings
- Respond to changes in settings
- Summaries for settings
- Settings Activity template

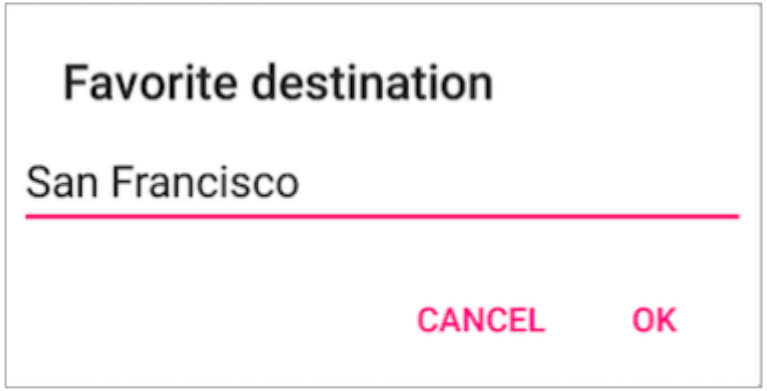
Settings

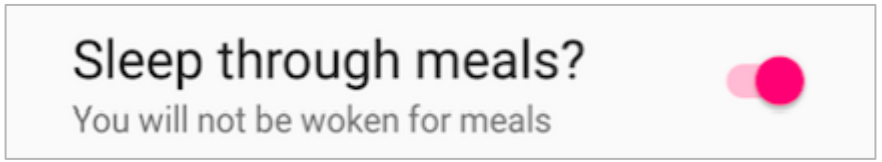
What are app settings?

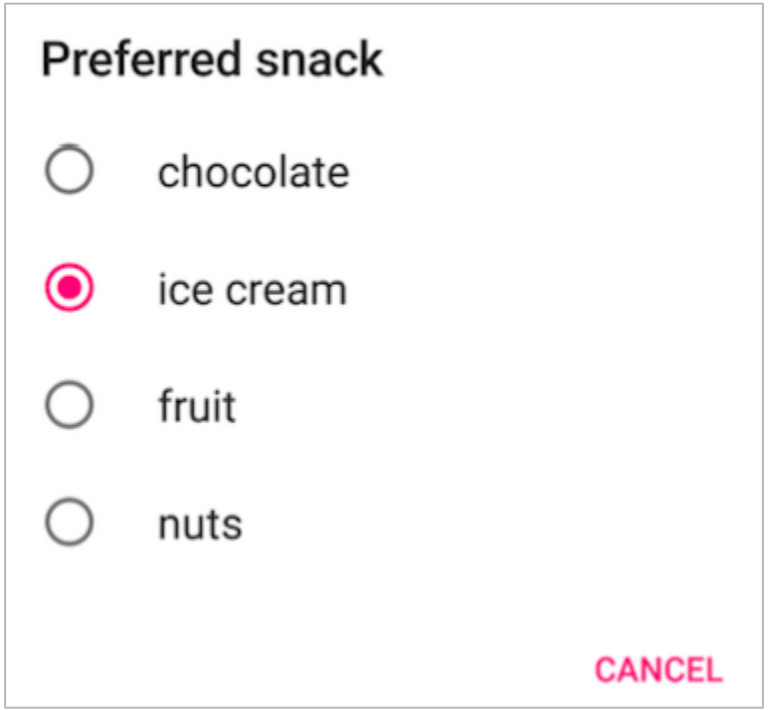
- Users can set features and behaviors of app

  Examples:
  - Home location, defaults units of measurement
  - Notification behavior for specific app
- For values that change infrequently and are relevant to most users
- If values change often, use options menu or nav drawer

Example settings



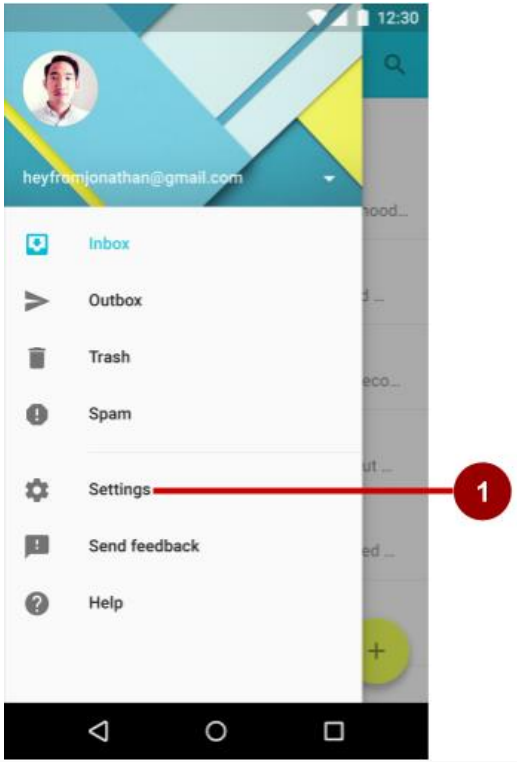


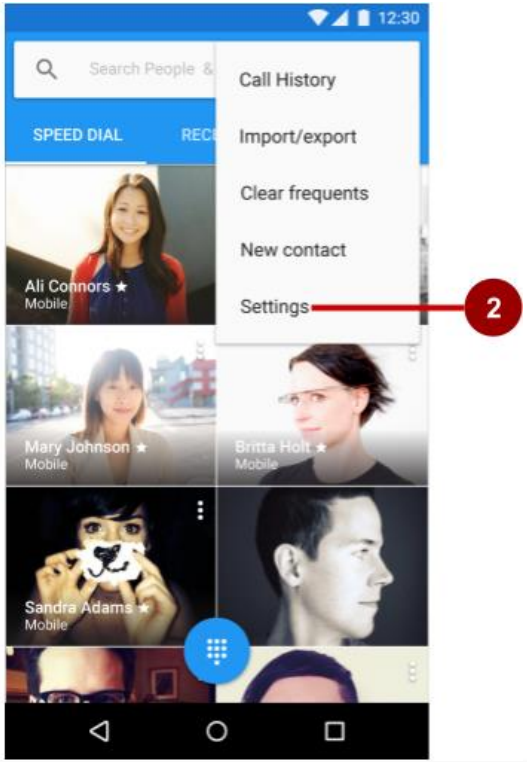


Accessing settings

Users access settings through:

1. Navigation drawer
2. Options menu

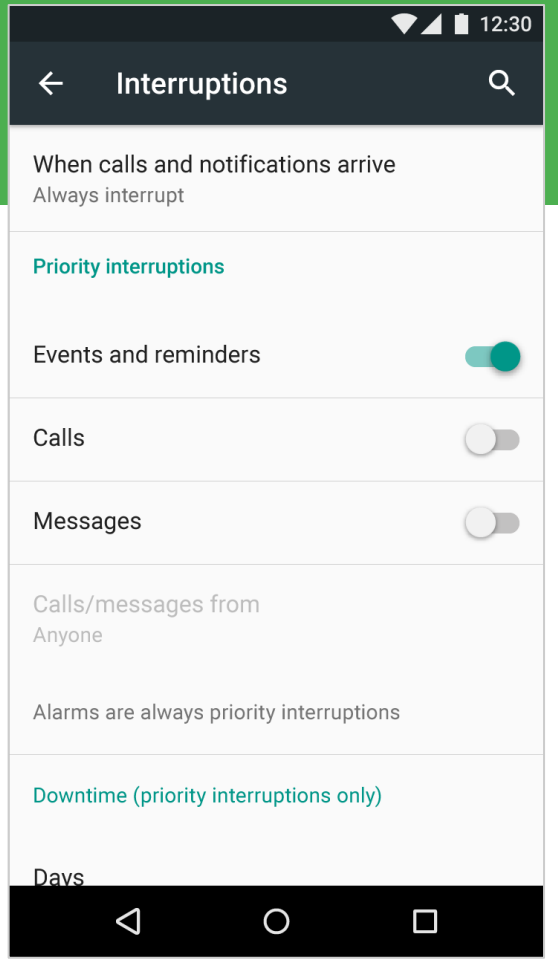




Setting screens

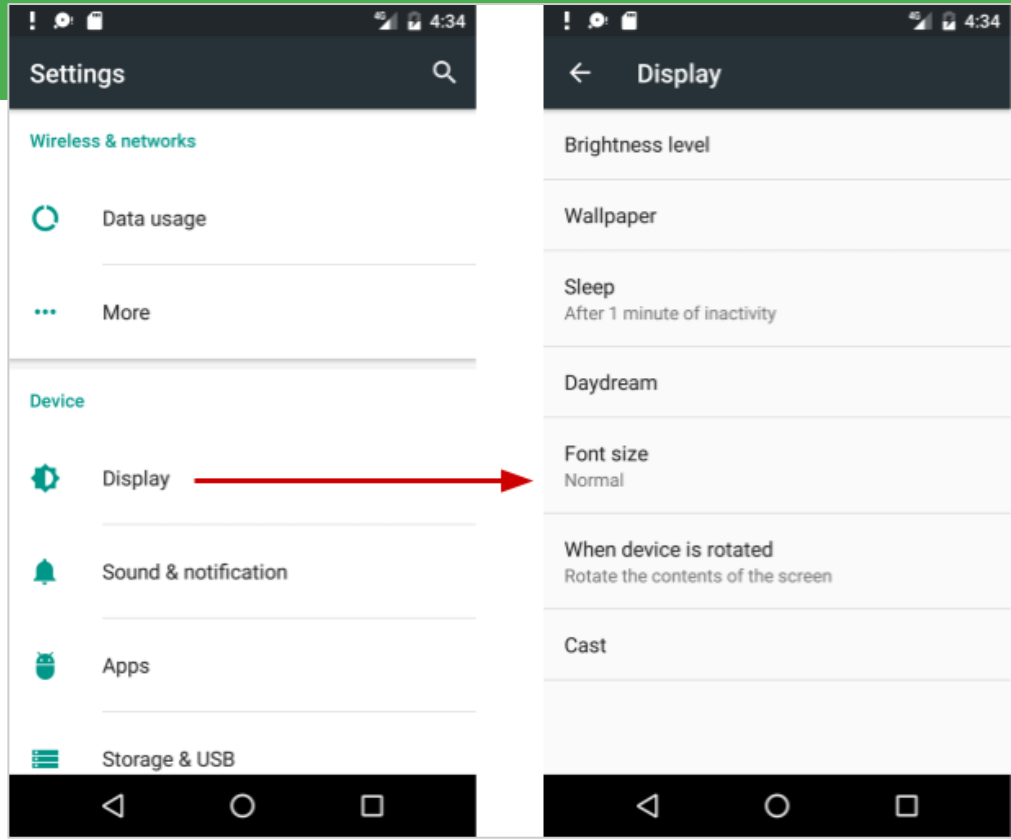
Organize your settings

- Predictable, manageable number of options
- 7 or less: arrange according to priority with most important at top
- 7-15 settings: group related settings under section dividers



16+ Settings

- Group into screens opened from main Settings screen



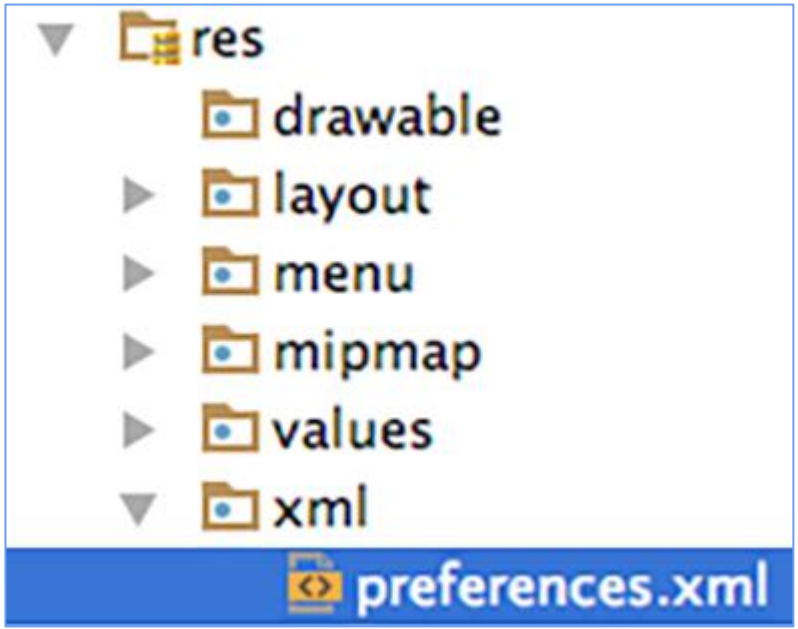
View versus Preference

- Use Preference objects instead of View objects in your Settings screens
- Design and edit Preference objects in the layout editor just like you do for View objects

Define Settings in a Preference Screen

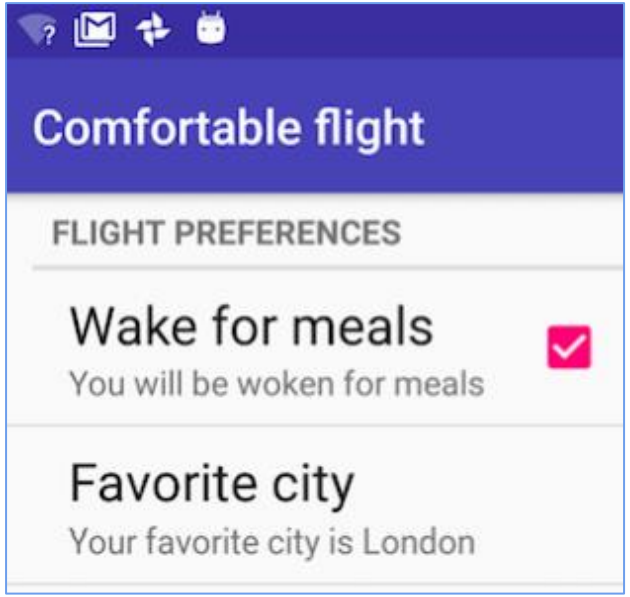
- Define settings in a preferences screen
- It is like a layout
- define in:

  res > xml > preferences.xml



Preference Screen example

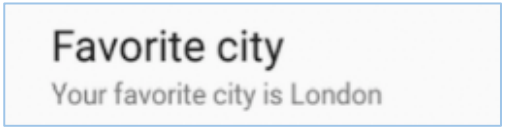
```
<PreferenceScreen>
  <PreferenceCategory
    android:title="Flight Preferences">
    <CheckBoxPreference
       android:title="Wake for meals"
       ... />
    <EditTextPreference
       android:title="Favorite city"
       .../>
  </PreferenceCategory>
</PreferenceScreen>
```



Every Preference must have a key

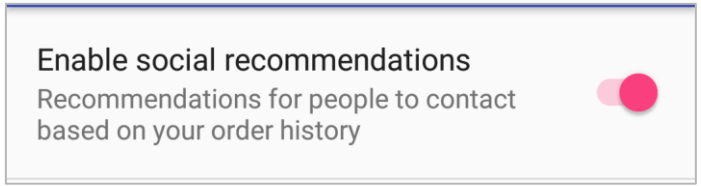
- Every preference must have a key
- Android uses the key to save the setting value

```
<EditTextPreference
    android:title="Favorite city"
    android:key="fav_city"
    … />
```



SwitchPreference

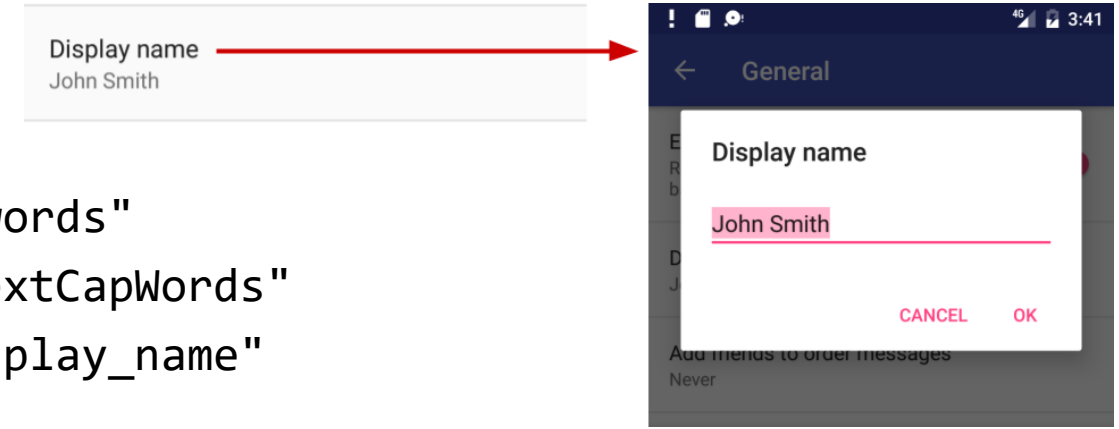
```
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">

    <SwitchPreference
        android:defaultValue="true"
        android:title="@string/pref_title_social"
        android:key="switch"
        android:summary="@string/pref_sum_social" />
</PreferenceScreen>
```



SwitchPreference attributes

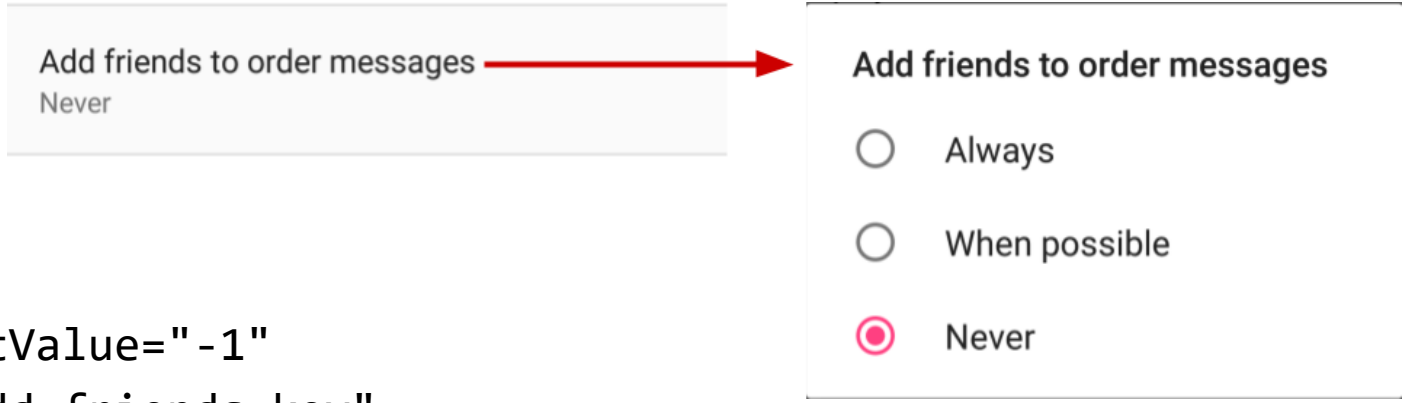
- `android:defaultValue`—true by default
- `android:summary`—text underneath setting, for some settings, should change to reflect value
- `android:title`—title/name
- `android:key`—key for storing value in SharedPreferences

EditTextPreference




```
<EditTextPreference
      android:capitalize="words"
      android:inputType="textCapWords"
      android:key="user_display_name"
      android:maxLines="1"
      android:defaultValue="@string/pref_default_display_name"
      android:title="@string/pref_title_display_name" />
```

ListPreference




```
<ListPreference
  android:defaultValue="-1"
  android:key="add_friends_key"
  android:entries="@array/pref_example_list_titles"
  android:entryValues="@array/pref_example_list_values"
  android:title="@string/pref_title_add_friends_to_messages" />
```

ListPreference

- Default value of -1 for no choice
- `android:entries`—Array of labels for radio buttons
- android:entryValues —Array of values radio button

Preference class

- Preference class provides View for each kind of setting
- associates View with SharedPreferences interface to store/retrieve the preference data
- Uses key in the Preference to store the setting value

Preference subclasses

- CheckBoxPreference—list item that shows a checkbox
- ListPreference—opens a dialog with a list of radio buttons
- SwitchPreference—two-state toggleable option
- EditTextPreference—that opens a dialog with an EditText
- RingtonePreference—lets user to choose a ringtone

Classes for grouping

- PreferenceScreen
  - root of a Preference layout hierarchy
  - at the top of each screen of settings
- PreferenceGroup
  - for a group of settings (Preference objects).
- PreferenceCategory
  - title above a group as a section divider

Implement settings

Settings UI uses fragments

- Use an Activity with a Fragment to display the Settings screen
- Use specialized Activity and Fragment subclasses that handle the work of saving settings

Activities and fragments for settings

- Android 3.0 and newer:
  - AppCompatActivity with PreferenceFragmentCompat
  - OR use Activity with PreferenceFragment
- Android older than 3.0 (API level 10 and lower):
  - build a special settings activity as an extension of the PreferenceActivity class (use the template!)

Lesson focusses on this!

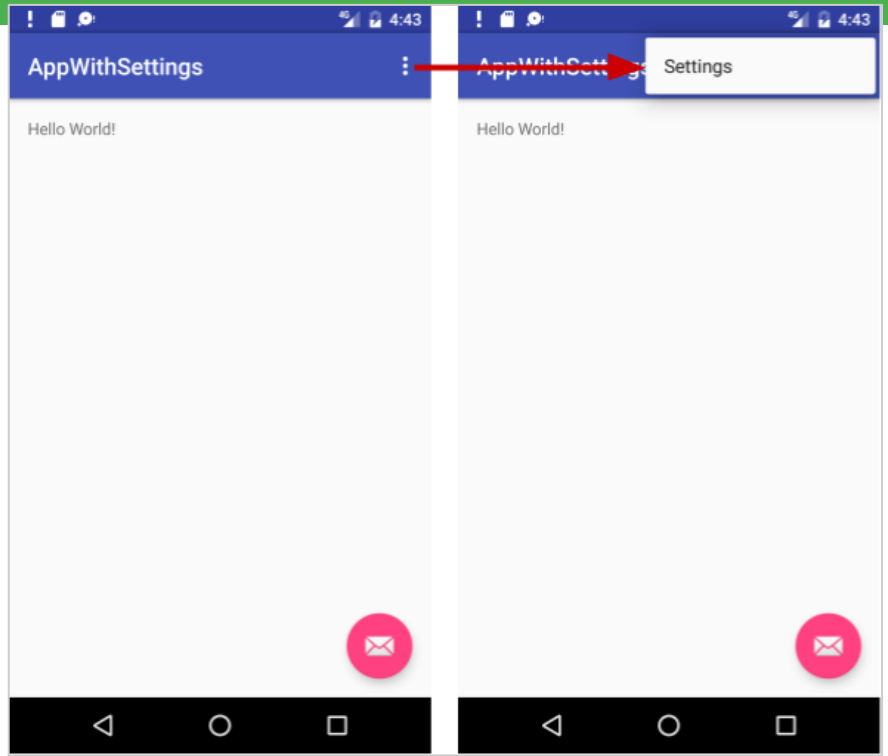
Steps to implement Settings

For AppCompatActivity with PreferenceFragmentCompat:

- Create the preferences screen
- Create an Activity for the settings
- Create a Fragment for the settings
- Add the preferenceTheme to the AppTheme
- Add code to invoke Settings UI

Basic Activity template

- Basic Activity template
  Includes options menu
- **Settings** menu item provided for options menu



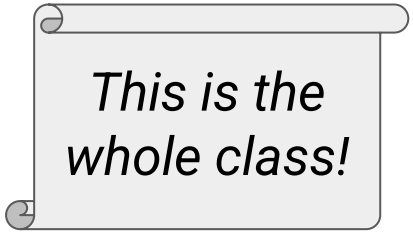
Create a Settings Activity subclass

- Extends AppCompatActivity
- in onCreate() display the settings Fragment:

```
getSupportFragmentManager()
    .beginTransaction()
    .replace(android.R.id.content,
             new MySettingsFragment())
    .commit();
```

Settings Activity example

```
public class MySettingsActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportFragmentManager().beginTransaction()
            .replace(android.R.id.content, new MySettingsFragment())
            .commit();
    }
}
```



Create a Settings Fragment subclass

- Extends PreferenceFragmentCompat
- Implement methods:
  - onCreatePreferences() displays the settings
  - setOnPreferenceChangeListener() handles any changes that need to happen when the user changes a preference (optional)

PreferenceFragment

```
public class MySettingsFragment
    extends PreferenceFragmentCompat { …}
```

- Blank fragments include onCreateView() by default
- Replace onCreateView() with onCreatePreferences() because this fragment displays a preferences screen

Settings Fragment example

```
public class MySettingsFragment extends PreferenceFragmentCompat {

    @Override
    public void onCreatePreferences(Bundle savedInstanceState,
                                    String rootKey) {
        setPreferencesFromResource(R.xml.preferences, rootKey);
    }
}
```

Add PreferenceTheme to app's theme

If using PreferenceFragmentCompat, set preferenceTheme in styles.xml:

```
<style name="AppTheme" parent="...">
    ...
    <item name="preferenceTheme">
        @style/PreferenceThemeOverlay
    </item>
  …
</style>
```

Invoke Settings UI

Send the Intent to start the Settings Activity:

- From Options menu, update onOptionItemsSelected()
- From Navigation drawer, update onItemClick() on the OnItemClickListener given to setOnItemClickListener

Default Settings

Default settings

- Set default to value most users would choose
  - All contacts
- Use less battery power
  - Bluetooth is off until the user turns it on
- Least risk to security and data loss
  - Archive rather than delete messages
- Interrupt only when important
  - When calls and notifications arrive

Set default values

- Use android:defaultValue in Preference view in xml:

```
<EditTextPreference
    android:defaultValue="London"
  … />
```

- In onCreate() of MainActivity, save default values.

Save default values in shared preferences

In onCreate() of MainActivity

```
PreferenceManager.setDefaultValues(
        this, R.xml.preferences, false);
```

- App context, such as `this`
- Resource ID of XML resource file with settings
- `false` only calls method the first time the app starts

Save and retrieve settings

Saving setting values

- No need to write code to save settings!
- If you use specialized Preference Activity and Fragment, Android automatically saves setting values in shared preferences

Get settings from shared preferences

- In your code, get settings from default shared preferences
- Use key as specified in preference view in xml

```
SharedPreferences sharedPref =
    PreferenceManager.getDefaultSharedPreferences(this);

String destinationPref =
     sharedPref.getString("fav_city", "Jamaica");
```

Get settings values from shared preferences

- In preference definition in xml:

```
<EditTextPreference
    android:defaultValue="London"
    android:key="fav_city" />
```

- In code, get fav_city setting:

```
String destinationPref =
    sharedPref.getString("fav_city", "Jamaica");
```

default setting value

is different than

default value returned by pref.getString() if key is not found in shared prefs

Respond to changes in settings

Listening to changes

- Display related follow-up settings
- Disable or enable related settings
- Change the summary to reflect current choice
- Act on the setting

  For example, if the setting changes the screen background, then change the background

Listen for changes to settings

- Define setOnPreferenceChangeListener()
- in onCreatePreferences() in the Settings Fragment

onCreatePreferences() example

```
@Override
public void onCreatePreferences(Bundle savedInstanceState,
                                String rootKey) {
   setPreferencesFromResource(R.xml.preferences, rootKey);
   ListPreference colorPref =
                (ListPreference) findPreference("color_pref");
   colorPref.setOnPreferenceChangeListener(
     // see next slide
     // ...);
}
```

onPreferenceChangeListener() example

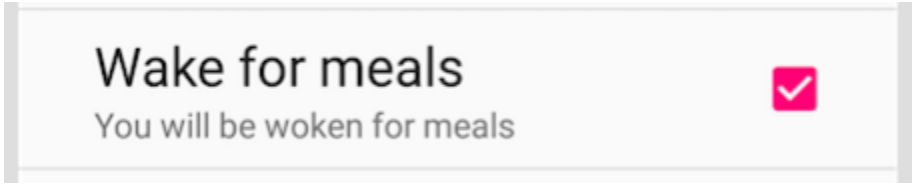
Example: change background color when setting changes

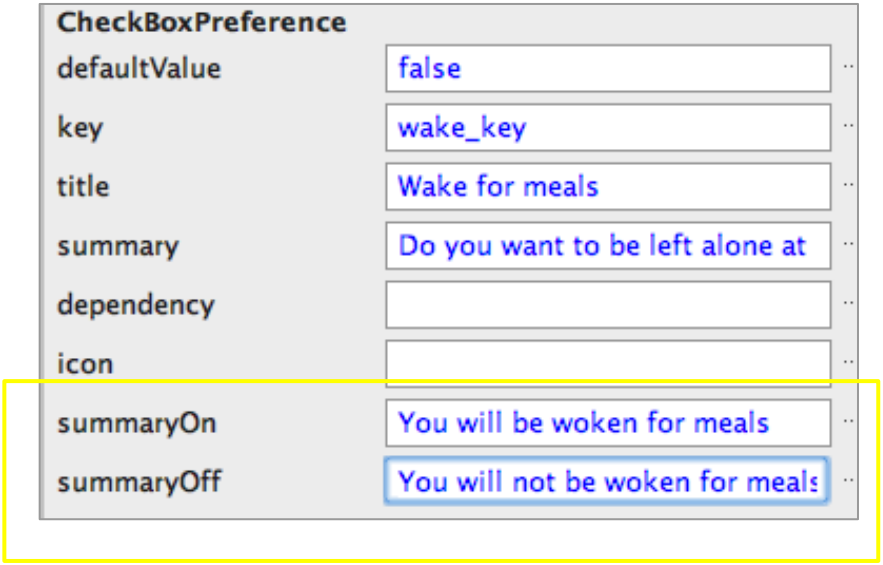
```
colorPref.setOnPreferenceChangeListener(
    new Preference.OnPreferenceChangeListener(){
        @Override
        public boolean onPreferenceChange(
           Preference preference, Object newValue){
               setMyBackgroundColor(newValue);
               return true;
        }
});
```

Summaries for settings

Summaries for true/false values

Set attributes to define conditional summaries for preferences that have true/false values

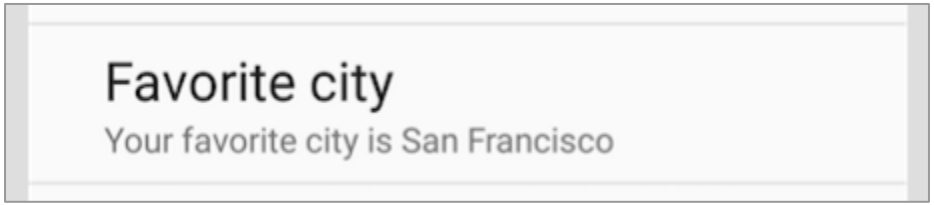


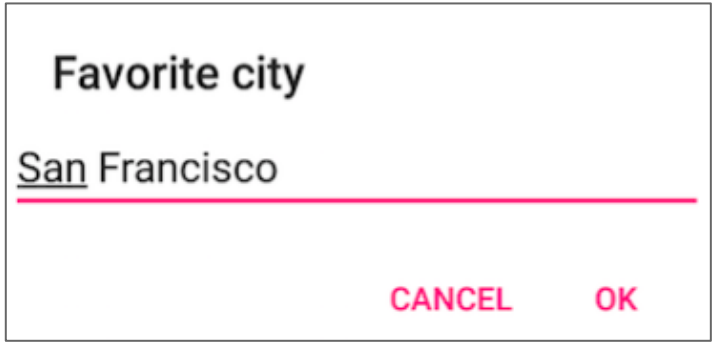


Summaries for other settings

For settings that have values other than true/false, update the summary when the setting value changes

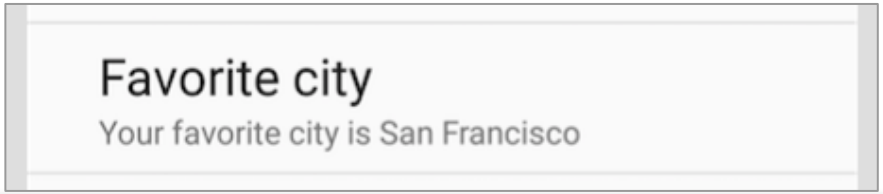
- Set the summary in onPreferenceChangeListener()





Set summary example

```
EditTextPreference cityPref = (EditTextPreference)
                                findPreference("fav_city");
cityPref.setOnPreferenceChangeListener(
  new Preference.OnPreferenceChangeListener(){
    @Override
    public boolean onPreferenceChange(Preference pref, Object value){
      String city = value.toString();
      pref.setSummary("Your favorite city is " + city);
      return true;
    }
  });
```



Activity Template

More complex?

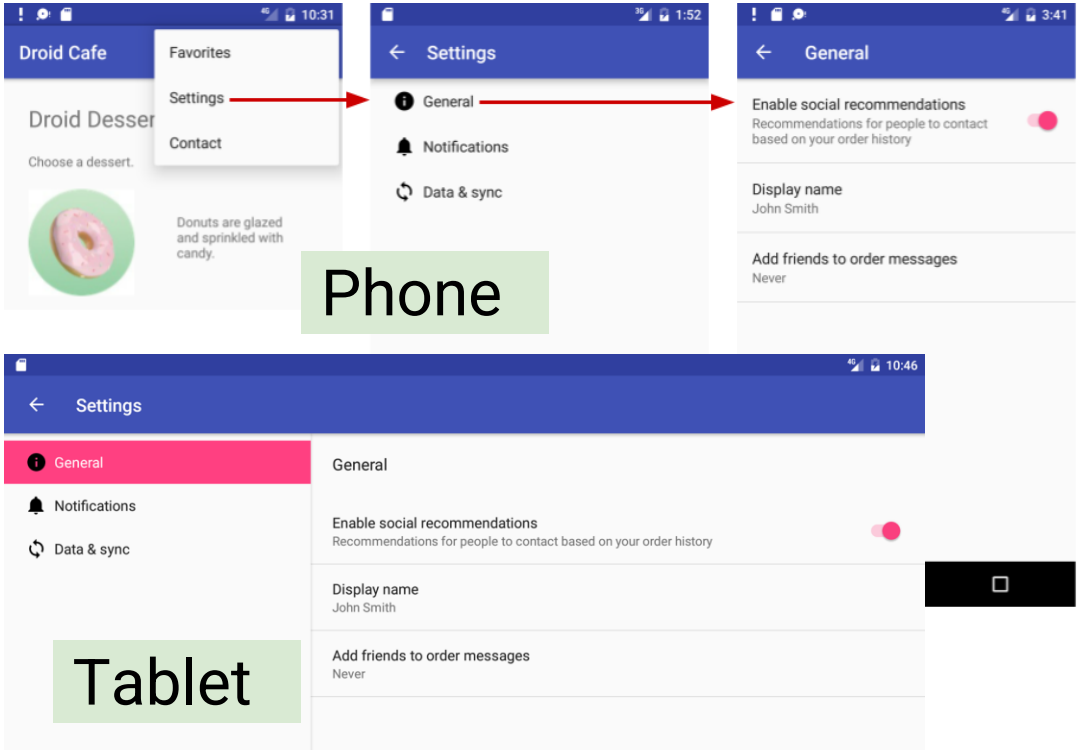
For anything more complex

?

use the Settings Activity template!

Settings Activity template

- Complex Settings
- Backwards compatibility
- Customize pre-populated settings
- Adaptive layout for phones and tablets



Learn more

Learn more

- Android Studio User Guide
- Settings (coding)
- Preference class
- PreferenceFragment
- Fragment
- SharedPreferences
- Saving Key-Value Sets
- Settings (design)

What's Next?

- Concept Chapter: 9.2 App settings
- Practical: 9.2 App settings

END