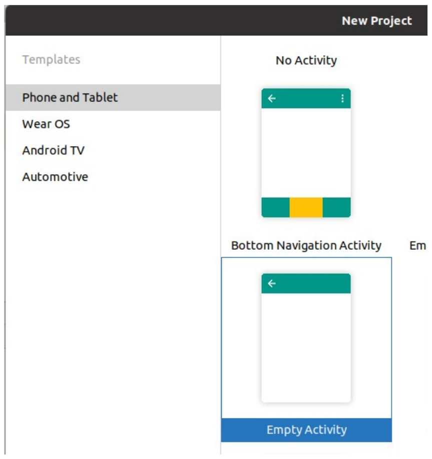
This tutorial guides you through creating a simple preference activity. And setting a preference listener

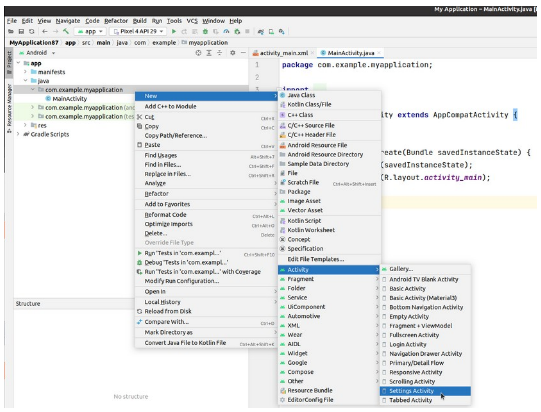
Remember the following shortcut keys to help speed things up

- ctrl-j live templates
- Alt-6 restore LogCat
- ctrl shift backspace- goto place last edited
- Alt+Enter quick fix/Extract string
- alt+Insert constructors, getter, setter
- ctrl-b - go to definition
- ctrl+Space code completion
- ctrl-o override members
- ctrl-H inheritance type heiarchy
- ctrl-alt-L Reformat code

1. Create an ‘Empty Activity’ project.



2. Add a settings Activity to this project.



```
package com.example.myapplication;

import
ity extends AppCompatActivity {
reate(Bundle savedInstanceState) {
(savedInstanceState);
(R.layout.activity_main);
```

3. Modify activity_main.xml to have a switch and a FAB (floating action button)
By the end of this tutorial, when you set the switch, your app will start listening for changes in the settings activity and will react accordingly. The FAB will invoke a Settings Activity that you create.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/cl"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="@color/white"
    android:padding="10dp">

    <androidx.appcompat.widget.SwitchCompat
        android:id="@+id/switch1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="toggle_preference_change_listener"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintBottom_toBottomOf="parent" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/floatingActionButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:clickable="true"
        android:focusable="true"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:srcCompat="@android:drawable/ic_dialog_info"
        android:contentDescription="Show Settings Activity" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

4. Modify root_preferences.xml (in res/xml). Add the following to the bottom (**before the </PreferenceScreen> closing tag!**) to include a seekbar (Notice that seekbar is not one of the Palette choices). Make sure it renders correctly.

```
    <PreferenceCategory app:title="Seekbar demonstration">
        <SeekBarPreference xmlns:android="http://schemas.android.com/apk/res/android"
            android:max="50"
            app:defaultValue="50"
            app:enabled="true"
            app:key="seek_bar_key"
            app:min="0"
            app:showSeekBarValue="true"
            app:title="Main Activity background shade"
            app:summary="lower is lighter, higher is darker" />
    </PreferenceCategory>
```

5.Hook up the FAB. It will invoke the settings activity (use alt-enter to import any needed packages). In MainActivity.java onCreate(…)

```java
//manage the FAB
FloatingActionButton fab = findViewById(R.id.floatingActionButton);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent myIntent = new Intent(MainActivity.this,SettingsActivity.class);
        startActivity(myIntent);
    }
});
```

6.Hook up the switch. It will turn on preferencechange listening. So your MainActivity will be notified whenever something changes in the SettingsActivity. In MainActivity.java onCreate(…)

```java
//get a reference to the switch
SwitchCompat s=findViewById(R.id.switch1);

//you want to know if it's state has changed, it derives from CompoundButton
//use it's OnCheckedChangeListener to get notified when it changes and its state
s.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
        togglePreferenceChangeListener(b);
    }
});
```

7. Get a reference to the defaultSharedPreferences.

Add the following as a class variable.

```java
//need these to track changes
private SharedPreferences myPreference;
```

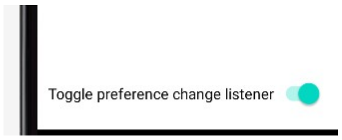
And get a reference to it in onCreate(..)

```java
// lets get a handle to default shared prefs
myPreference = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
```

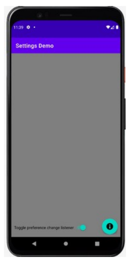
8. Add the togglePreferenceChangeListener function.

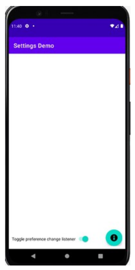
```
private SharedPreferences.OnSharedPreferenceChangeListener listener = null;
private void togglePreferenceChangeListener(boolean enablePreferenceListener) {
        // this is the bit that listens for any preference changes to defaultsharedpreferences
    // (the prefs that the pref activity accesses)
    //you can also implements OnSharedPreferenceChangeListener for the mainactivity and then
    //register to have have the mainactivity listen for changes like this
    //myPreference.registerOnSharedPreferenceChangeListener(this);
    //and forgo whats below
    if (listener == null) {
        listener = new SharedPreferences.OnSharedPreferenceChangeListener() {
            @Override
            public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
                Toast.makeText(MainActivity.this, "Handle change of Key=" + key, Toast.LENGTH_SHORT).show();
            }
        };
    }

    if(enablePreferenceListener)
        // register the listener (turn it on)
        myPreference.registerOnSharedPreferenceChangeListener(listener);
    else
         //or unregister it (turn it off)
        myPreference.unregisterOnSharedPreferenceChangeListener(listener);
}
```

9. The app now responds to settings activity changes iff switch1 in the main activity is enabled

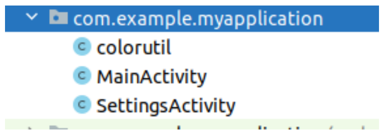


10. The seekbar's purpose is to fade the background of the mainactivity from white to gray. For instance the left image below is when seekbar is set to 0, the right when its at 50.





11. The background color for the constraintlayout is white (#FFFFFFFF) as defined in the res/values/colors file. We want to vary that from #66666666 to #FFFFFFFF using the seekbar. This color is in ARGB format, with the A being the Alpha (transparancy) channel.
We can multiply each of the color channels (R G and B) by some fraction between 0.5 and 1 to achieve our goal. To that end add the java class 'colorutil' to the project by clicking on the java package name and selecting 'new', then 'Java class'. It should look like this when you are done;



Fill this class with this code;

```
import android.graphics.Color;
/**
 * All this does is change the luminosity of the color gray
 * from black to white and all shades in between
 * Ripped off directly from
 * https://gist.github.com/martintreurnicht/f6bbb20a43211bc2060e
 */
public class colorutil {
     public static int mult(int color, double fraction) {
        int red = Color.red(color);
        int green = Color.green(color);
        int blue = Color.blue(color);
        red = multColor(red, fraction);
        green = multColor(green, fraction);
        blue = multColor(blue, fraction);
        int alpha = Color.alpha(color);
        return Color.argb(alpha, red, green, blue);
    }
    private static int multColor(int color, double fraction) {
        return (int)Math.max((color * fraction), 0);
    }
}
```

12. Add a method to MainActivity.java to set the background color of it's constrainlayout

```
private void setBackgroundColor(String key) {
    //get the new value of the slider and convert it to a fraction between .5 and 1.0
    double fract= (MainActivity.this.myPreference.getInt(key,50)/100.0) + .5;

    //get the original white
    int color = ContextCompat.getColor(MainActivity.this, R.color.white);

    //scale the white
    color=colorutil.mult(color,fract);

    //set the background color for this viewgroup
    ConstraintLayout cl=findViewById(R.id.cl);
    cl.setBackgroundColor(color);
}
```

13. Call the above method in togglePreferenceChangeListener. Add the following lines after the Toast.

```
if(key.equals(SEEK_BAR_KEY))
    setBackgroundColor(key);
```

14. Uh Oh, what is ***SEEK_BAR_KEY?*** Hover over this red value, hit alt-enter, and add a string constant to the class. Should look like this in the end.

```
public class MainActivity extends AppCompatActivity{
            //key to the seekbar preference in the settings activity
            //settingsactivity will store seekbar value to disk using this key
            private static final String SEEK_BAR_KEY = "seek_bar_key";
```

15. And to make sure we set the background to the proper color whenever we restart the app, add the following call to the end of onCreate().

```
//set the background color according to defaultsharedprefs saved value
setBackgroundColor(SEEK_BAR_KEY);
```