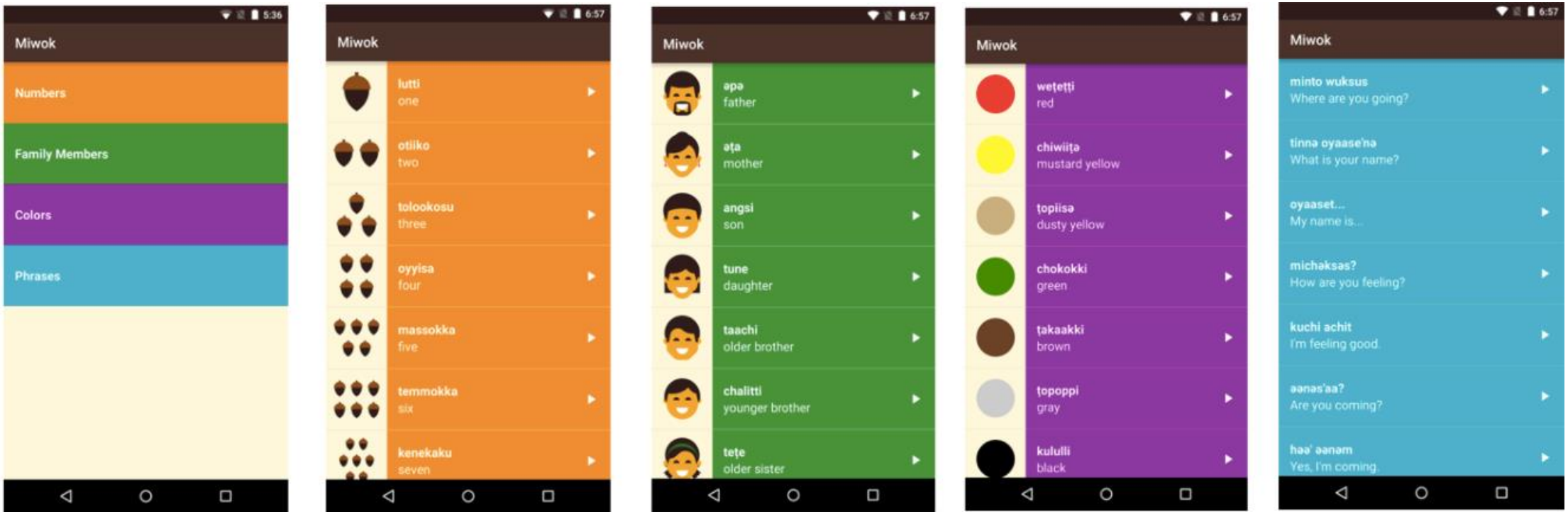
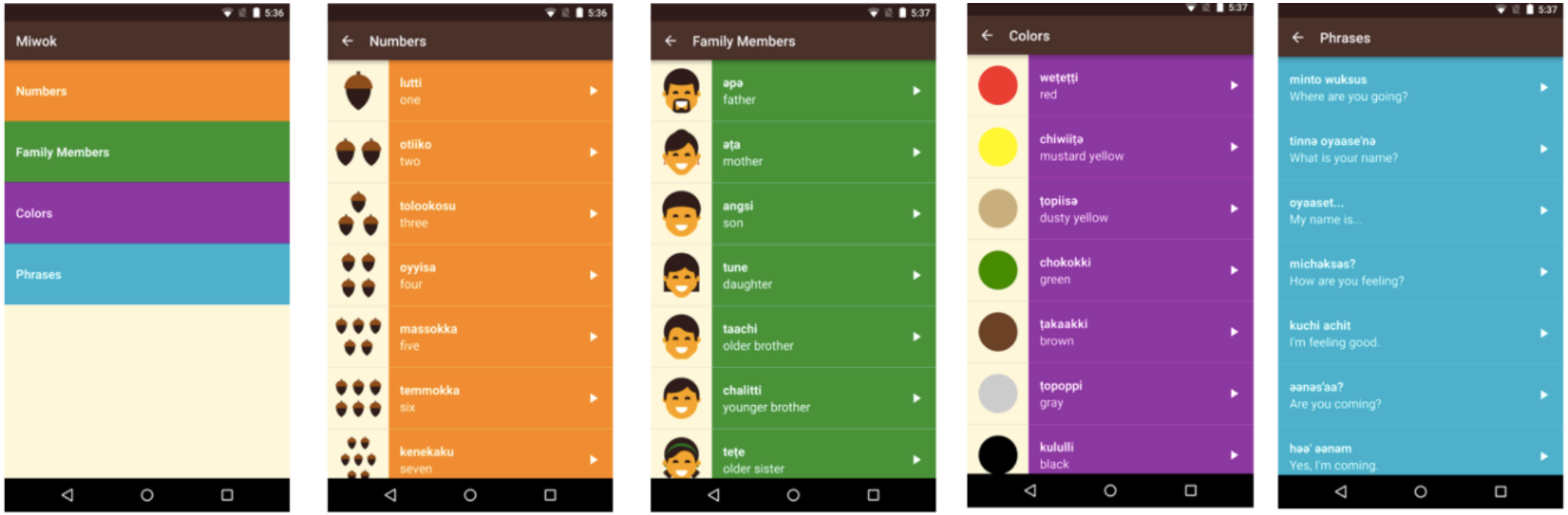
Miwok Part 5

Current Miwok App (without UP button)



Whit UP Button



UP Button – Modify android_manifest.xml

This is what a single category activity in the AndroidManifest.xml should look like when you're done:

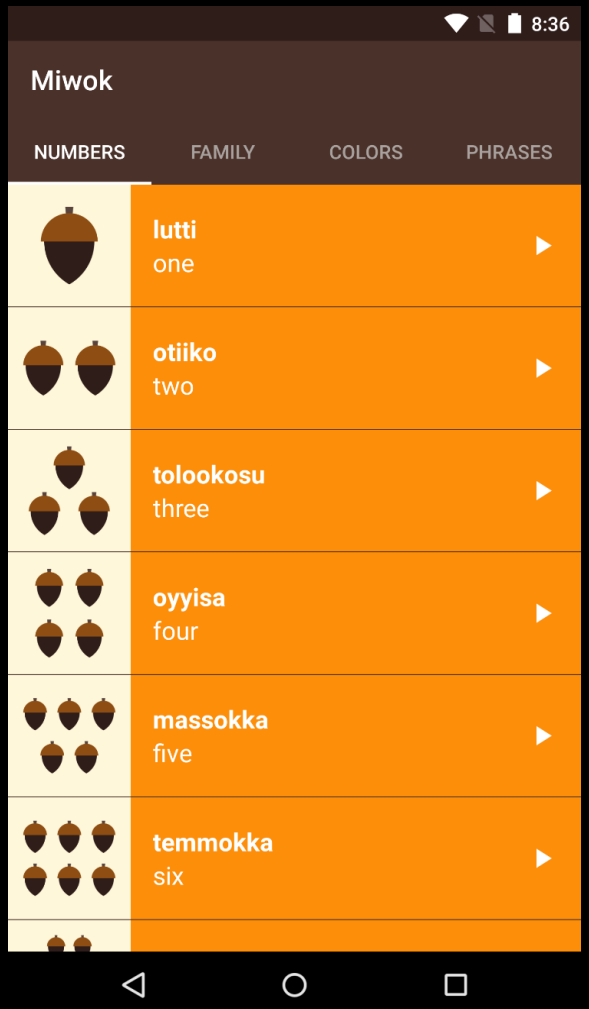
```
<activity
        android:name=".NumbersActivity"
        android:label="@string/category_numbers"
        android:parentActivityName=".MainActivity">

        <!-- Parent activity meta-data to support 4.0 and lower -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value=".MainActivity"/>
    </activity>
```

Modify other categories

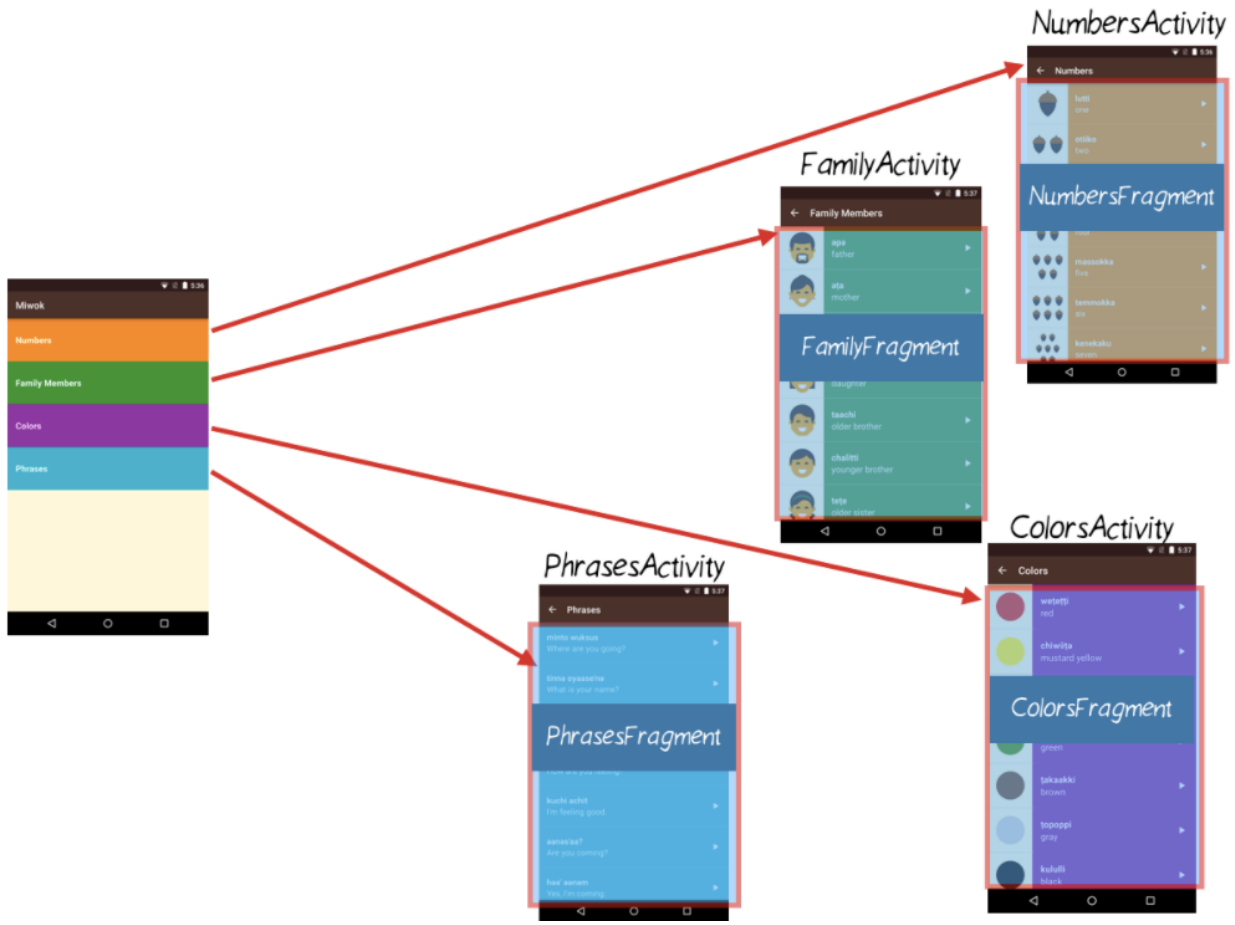
MIWOK APP 2.0

What to do?

- Create a new project: Tabbed Activity.
- Refactor the 4 activities into the fragments.
- Modify the MainActivity so it contains 4 pages, where each page is a Fragment.



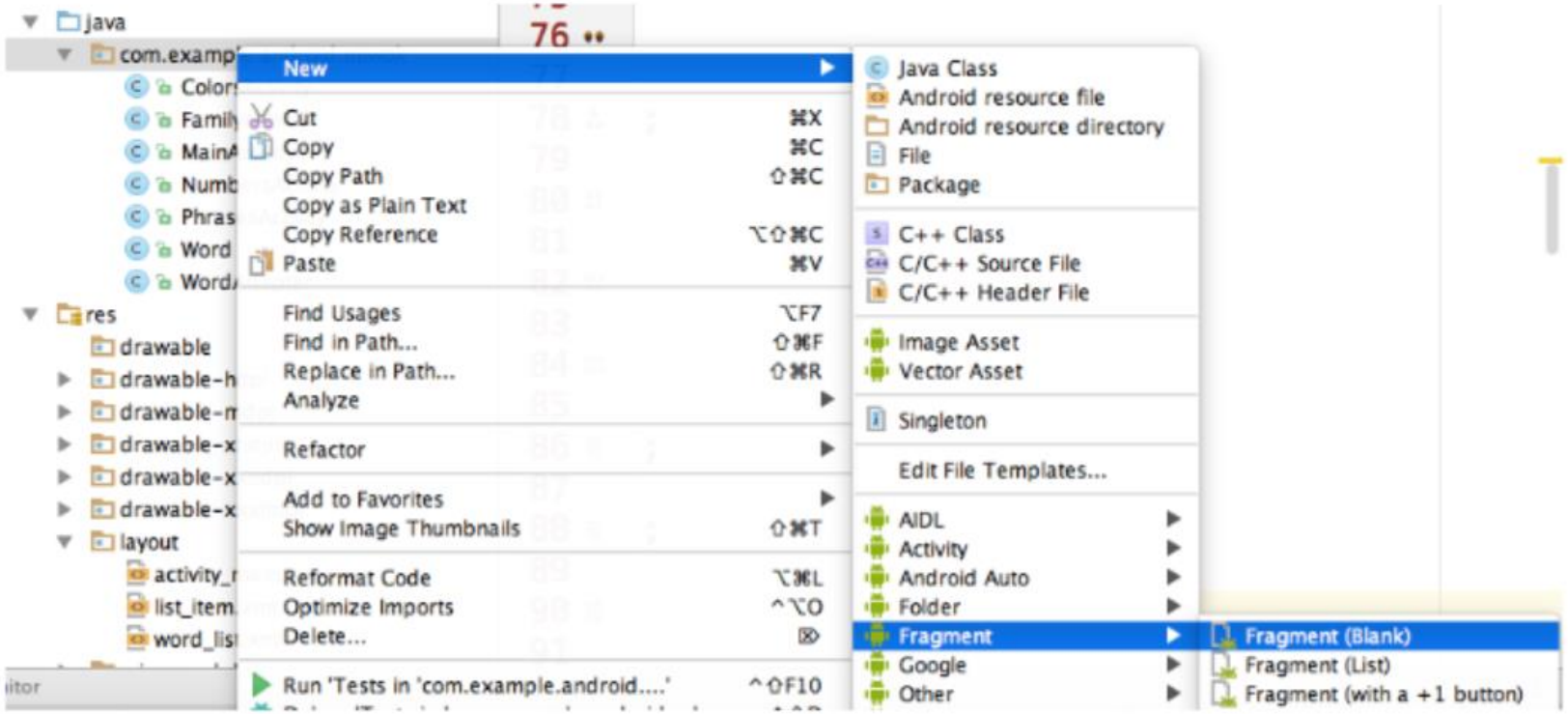
34 - Refactor 4 activities into 4 fragments



	NumbersActivity.java	NumbersFragment.java
Global variables: ***MediaPlayer, AudioManager, OnCompletionListener, OnAudioFocusChangeListener***	Same	Same
Creating the view	In onCreate() activity lifecycle callback: *Set the content view, setup the UI*	In onCreateView() fragment lifecycle callback: *Inflate the XML layout, setup the UI, return the root view*
Leaving the screen	In onStop() activity lifecycle callback: *Release the media player*	In onStop() fragment lifecycle callback: *Release the media player*
Helper method: ***releaseMediaPlayer()***	Same	Same

Create NumbersFragment class

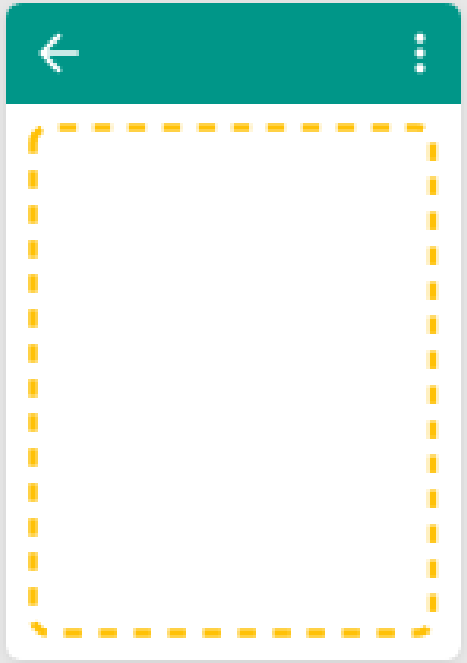
1) To start, create a new Java file for the NumbersFragment. Right click on the "com.example.android.miwok" folder. Go to New > Fragment > Fragment (Blank).



Configure Component

Android Studio

Creates a blank fragment that is compatible back to API level 4.



Fragment Name

NumbersFragment

- [ ] Create layout XML?
- [ ] Include fragment factory methods?
- [ ] Include interface callbacks?

Source Language

```
 * A simple {@link Fragment} subclass.
 */
public class NumbersFragment extends Fragment {


    public NumbersFragment() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        TextView textView = new TextView(getActivity());
        textView.setText("Hello blank fragment");
        return textView;
    }

}
```

Copy Global Variables from NumbersActivity to NumbersFragment

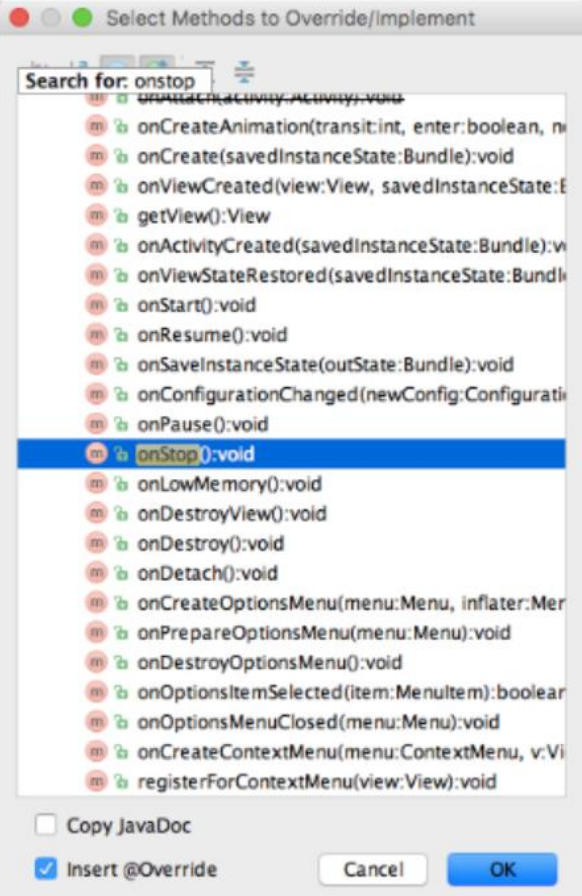
```java
public class NumbersFragment extends Fragment {
    private MediaPlayer mMediaPlayer;
    private AudioManager mAudioManager;

    private AudioManager.OnAudioFocusChangeListener mOnAudioFocusChangeListener = onAudioFocusChange(focusChange) → {
            if (focusChange == AudioManager.AUDIOFOCUS_LOSS_TRANSIENT ||
                    focusChange==AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK){
                mMediaPlayer.pause();
                mMediaPlayer.seekTo( i: 0);
            }else if (focusChange == AudioManager.AUDIOFOCUS_GAIN){
                mMediaPlayer.start();
            } else if (focusChange == AudioManager.AUDIOFOCUS_LOSS){
                releaseMediaPlayer();
            }
    };
    private MediaPlayer.OnCompletionListener mCompletionListener = new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mediaPlayer) {
            releaseMediaPlayer();
        }
    };
```

Copy releaseMediaPlayer() Put on the bottom before las }

```
    private void releaseMediaPlayer(){
        if (mMediaPlayer != null){
            mMediaPlayer.release();
            mMediaPlayer = null;
            mAudioManager.abandonAudioFocus(mOnAudioFocusChangeListener);
        }
    }
}
```

Override the Fragment’s onStop() method.

Move your cursor to an empty space in the class, where you can add a new method. Use the keyboard shortcut Ctl + O to pop up a dialog and select a method to override. Type in “onStop” and when you find that result, hit OK.



Android Studio will automatically add this method to your NumbersFragment class for you:

```
@Override
public void onStop() {
    super.onStop();
}
```

Modify the onStop() method so that it calls the releaseMediaPlayer method:

```
@Override
public void onStop() {
   super.onStop();

   // When the activity is stopped, release the media player resources because we won't
   // be playing any more sounds.
   releaseMediaPlayer();
}
```

Override the Fragment’s onCreateView() method.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
   View rootView = inflater.inflate(R.layout.word_list, container, false);

   /** TODO: Insert all the code from the NumberActivity’s onCreate() method after the setContentView meth
od call */

   return rootView;
}
```

Error #1: You will get get an error saying cannot resolve method “findViewById(int)” because the Fragment does not have a findViewById method, whereas the Activity did have that method (see link). `ListView listView = (ListView) findViewById(R.id.list);`

Fix the error by calling findViewById(int) on the rootView object, which should contain children views such as the ListView. `ListView listView = (ListView) rootView.findViewById(R.id.list);`

Error #2: You will get an error saying cannot resolve method “getSystemService(String)” because the Fragment does not have access to system services, whereas the Activity does (see link). `mAudioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);`

Fix the error by getting the Activity object instance first. This is the Activity that encloses the current Fragment, which will be the NumbersActivity for the NumbersFragment. Then call getSystemService(String) on that Activity object.

```
mAudioManager = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
```

Error #3: There's a problem with the arguments passed into the WordAdapter constructor because the first parameter "this" refers to this class (which is the NumbersFragment), and a Fragment is not a valid Context. However, the code used to work when "this" referred the NumbersActivity because an Activity is a valid Context. `WordAdapter adapter = new WordAdapter(this, words, R.color.category_numbers);`

Fix the error by passing in a reference to the Activity that encloses this Fragment as the context. `WordAdapter adapter = new WordAdapter(getActivity(), words, R.color.category_numbers);`

Error #4: When creating a MediaPlayer object, we need to pass in a context. Again, "this" refers to the NumbersFragment (and not the NumbersActivity), and the Fragment is not a valid Context. `mMediaPlayer = MediaPlayer.create(NumbersActivity.this, word.getAudioResourceId());`

Fix the error by passing in the activity for the first input parameter. `mMediaPlayer = MediaPlayer.create(getActivity(), word.getAudioResourceId());`

After fixing these 4 cases, there should be no more errors in this file! The NumbersFragment onCreateView() method should look like this.

Update Numbers Activity

- Under the res/layout directory, create a new layout file called activity_category.xml. The important part is that the view has an ID. We chose to give the view an ID called “container”.

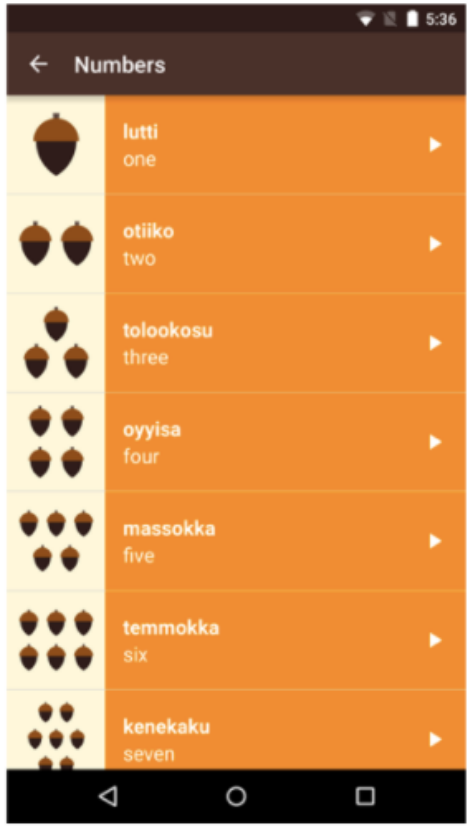
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/container"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"/>
```

- Now we need to update the NumbersActivity to use the NumbersFragment, otherwise there will be duplicate code that does the same thing in both classes.
- Replace the NumbersActivity code with this entire code snippet. We're going to use this simplified activity that sets the activity_category XML layout resource as the content view. Then a new NumbersFragment is created and inserted it into the container view, using a **FragmentTransaction** (no need to understand the details of this now). Since the container has "match_parent" for width and height, the NumbersFragment will take up the whole width and height of the screen.

```java
package com.example.android.miwok;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class NumbersActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_category);
    getSupportFragmentManager().beginTransaction()
            .replace(R.id.container, new NumbersFragment())
            .commit();
    }
}
```

BEFORE

NumbersActivity



Use layout
words_list.xml

AFTER

NumbersActivity
using
NumbersFragment

Use layout
activity_category.xml

Use layout
words_list.xml

Repeat REFACTOR for all activities and RUN

35 - Add ViewPager in MainActivity to swipe between 4 lists of words

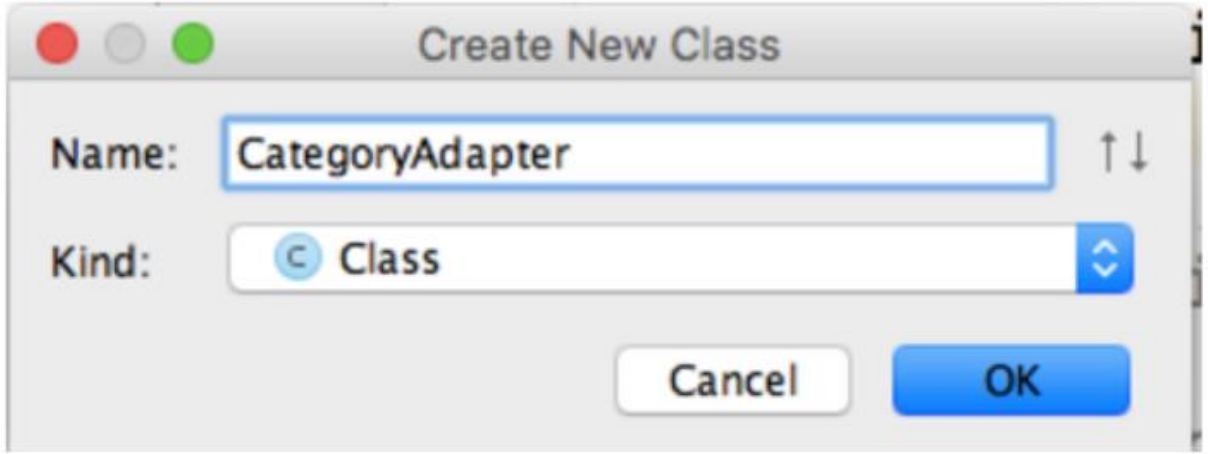
- First modify activity_main.xml layout to contain a ViewPager. You can delete the 4 category TextViews that used to be in this layout file.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/tan_background"
   android:orientation="vertical"
   tools:context="com.example.android.miwok.MainActivity">

   <android.support.v4.view.ViewPager
       android:id="@+id/viewpager"
       android:layout_width="match_parent"
       android:layout_height="match_parent"/>

</LinearLayout>
```

Create CategoryAdapter.java

2) In order to populate pages in the ViewPager, we need an adapter.

Create a new file for the adapter by right-clicking on the com.example.android.miwok folder in the Project directory pane. Then go to New > Java class. Create a new class called CategoryAdapter.



Modify CategoryAdapter

```java
package com.example.android.miwok;

import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;

/**
  * Created by katherinekuan on 4/14/16.
*/
public class CategoryAdapter extends FragmentPagerAdapter {

   public CategoryAdapter(FragmentManager fm) {
       super(fm);
   }

   @Override
   public Fragment getItem(int position) {
   return null;
   }

   @Override
   public int getCount() {
       return 0;
   }
}
```

5) Override the methods with the logic we want in our Miwok app. We need to think about:

Question: How many pages do we need in the ViewPager? *Answer:* 4 pages, so we should return 4 in the CategoryAdapter getCount() method.

Question: Which Fragment should we display if the position is 0? Or 1 or 2? *Answer:* Within the CategoryAdapter getItem(int position) method, we create a conditional if/else statement to return the appropriate category fragment for the given position.

```java
@Override
public Fragment getItem(int position) {
    if (position == 0) {
        return new NumbersFragment();
    } else if (position == 1) {
        return new FamilyFragment();
    } else if (position == 2) {
        return new ColorsFragment();
    } else {
        return new PhrasesFragment();
    }
}

/**
 * Return the total number of pages.
 */
@Override
public int getCount() {
    return 4;
}
```

6) Then in the MainActivity, we can hook up the CategoryAdapter to power the ViewPager. Delete all the old code related to the 4 category TextViews.

All we need to do is find the ViewPager that was declared in the XML layout. Then create a new CategoryAdapter, and set the adapter onto the ViewPager (using the setAdapter method).

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Set the content of the activity to use the activity_main.xml layout file
        setContentView(R.layout.activity_main);

        ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);

        CategoryAdapter adapter = new CategoryAdapter(getSupportFragmentManager());

        viewPager.setAdapter(adapter);

    }

}
```

Run the APP

- Main menu langsung ke Numbers
- Slide untuk berpindah dari Numbers ke family dan seterusnya

Jika waktu di RUN tidak ada masalah

- Delete:
  - NumbersActivity.java
  - FamilyActivity.java
  - ColorsActivity.java
  - PhrasesActivity.Java
  - activity_category.xml
- Also delete the activity declarations from the AndroidManifest.xml file.
- Run AGAIN

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.miwok">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Miwok"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

36 - Add tabs to ViewPager

Now let's add tabs so the user can tell that there are more pages to swipe to.

First you'll need to use the Android Design Support Library. This helps you create a Material Design app that runs even on older versions of Android. Learn more about the support library in this blogpost. There are periodic updates to the support library, so you can check this site for the latest updates or subscribe to the Android Developers blog.

Add Android Design Support Library to Your Project

1) In Android Studio, within the Project Directory pane, navigate to Miwok > app > build.gradle and open the build.gradle file.

Gradle is the tool that Android Studio uses to generate the apk (app file) that goes onto the device. For more info on configuring the build.gradle file, see this article. There's also an advanced Udacity course on this topic, so don't worry if you don't understand it now.

Project | Packages

- Miwok (~/Documents/Udacity/AndroidforBeginners2/P
  - .gradle
  - .idea
  - app
    - build
    - libs
    - src
    - .gitignore
    - app.iml
    - build.gradle
    - proguard-rules.pro
  - build
  - gradle
  - .gitignore
  - build.gradle
  - gradle.properties
  - gradlew
  - gradlew.bat
  - local.properties
  - Miwok.iml
  - settings.gradle
- External Libraries

app ×

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.1"

    defaultConfig {
        applicationId "com.example.android.miw
        minSdkVersion 15
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFi
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:
    compile 'com.android.support:support-v4:23
}
```

Modify dependencies and SYNC

```
compile 'com.android.support:design:23.3.0'
```

Afterwards, it should look similar to this:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.2.1'
    compile 'com.android.support:support-v4:23.2.1'
    compile 'com.android.support:design:23.3.0'
}
```

Modify activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:tools="http://schemas.android.com/tools"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:background="@color/tan_background"
     android:orientation="vertical"
     tools:context="com.example.android.miwok.MainActivity">

     <android.support.design.widget.TabLayout
         android:id="@+id/tabs"
         android:layout_width="match_parent"
         android:layout_height="wrap_content" />

     <android.support.v4.view.ViewPager
         android:id="@+id/viewpager"
         android:layout_width="match_parent"
         android:layout_height="match_parent"/>

  </LinearLayout>
```

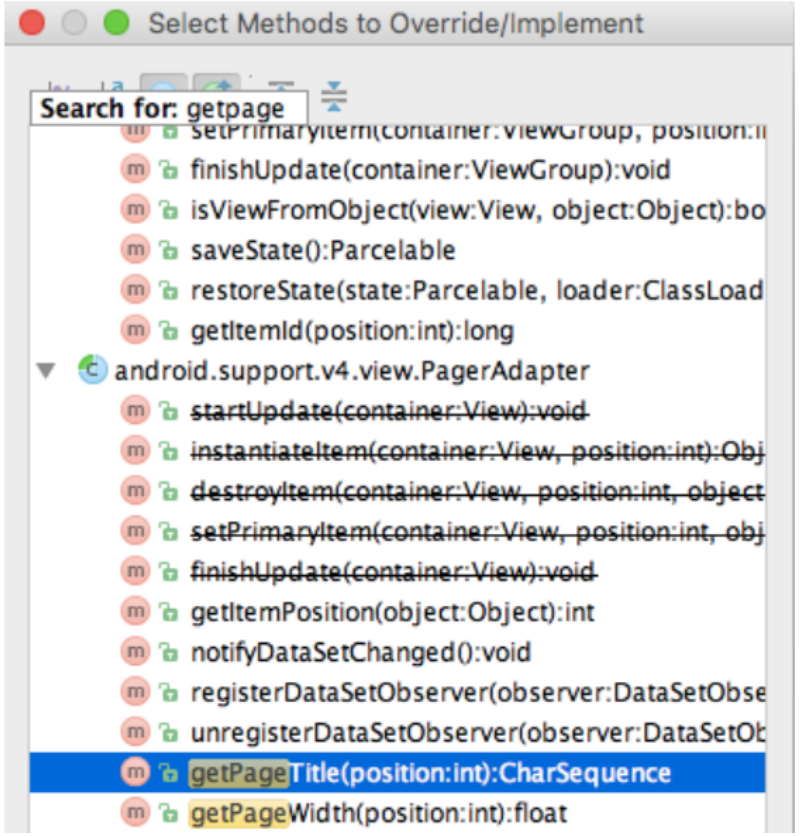
Modify ManiActivity.java

```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Set the content of the activity to use the activity_main.xml layout file
        setContentView(R.layout.activity_main);

        ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);

        CategoryAdapter adapter = new CategoryAdapter(getSupportFragmentManager());

        viewPager.setAdapter(adapter);

        //Tab Layout
        TabLayout tabLayout = (TabLayout) findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(viewPager);

    }
}
```

Modify CategoryAdapter.java

```
private Context mContext;

public CategoryAdapter (Context context, FragmentManager fm){
    super(fm);
    mContext = context;
}
```

To override the method, use the keyboard shortcut Ctl + O. Type in “getPageTitle” and hit OK.



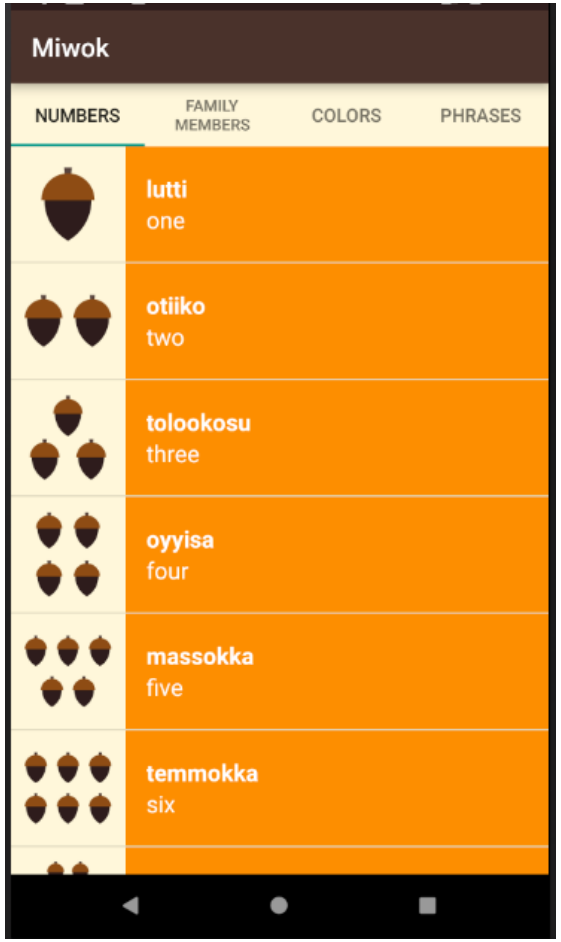

```
    @Override
    public int getCount() {
        return 4;
    }

    @Override
    public CharSequence getPageTitle(int position) {
        if (position == 0) {
            return mContext.getString(R.string.category_numbers);
        } else if (position == 1) {
            return mContext.getString(R.string.category_family);
        } else if (position == 2) {
            return mContext.getString(R.string.category_colors);
        } else {
            return mContext.getString(R.string.category_phrases);
        }
    }
}
```

5) Since we modified the CategoryAdapter constructor, we also need to update the MainActivity (which uses that constructor). When we create a CategoryAdapter, we pass in a Context (which is “this” or the activity) and the FragmentManager.

That declaration line might look like this:

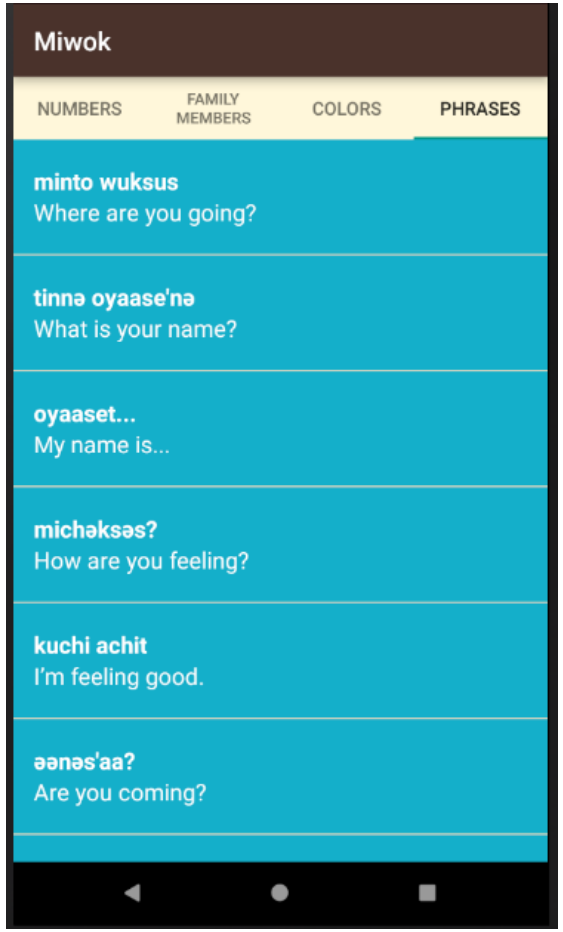
```
CategoryAdapter adapter = new CategoryAdapter(this, getSupportFragmentManager());
```

RUN APP


Miwok
NUMBERS
FAMILY MEMBERS
COLORS
PHRASES
lutti
one
otiiko
two
tolookosu
three
oyyisa
four
massokka
five
temmokka
six


Miwok
NUMBERS
FAMILY MEMBERS
COLORS
PHRASES
əpə
father
əṭa
mother
angsi
son
tune
daughter
taachi
older brother
chalitti
younger brother


Miwok
NUMBERS
FAMILY MEMBERS
COLORS
PHRASES
weṭeṭṭi
red
chiwiiṭə
mustard yellow
ṭopiisə
dusty yellow
chokokki
green
ṭakaakki
brown
ṭopoppi
gray


Miwok
NUMBERS
FAMILY MEMBERS
COLORS
PHRASES
minto wuksus
Where are you going?
tinnə oyaase'nə
What is your name?
oyaaset...
My name is...
michəksəs?
How are you feeling?
kuchi achit
I'm feeling good.
əənəs'aa?
Are you coming?

Modify string.xml

7) The first thing we notice is that the “Family Members” text is too long, compared to the other tab labels. In the strings.xml file, we can change the string to Family instead of Family Members.

When you modify strings, be careful of all the places that use that string. In this case, we only use the string in one place, so it’s safe to change.

```
<!-- Category name for the vocabulary words for family members [CHAR LIMIT=20] -->
<string name="category_family">Family</string>
```

Modify activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/primary_color"
   android:orientation="vertical"
   tools:context="com.example.android.miwok.MainActivity">

   <android.support.design.widget.TabLayout
       android:id="@+id/tabs"
       style="@style/CategoryTab"
       android:layout_width="match_parent"
       android:layout_height="wrap_content" />

   <android.support.v4.view.ViewPager
       android:id="@+id/viewpager"
       android:layout_width="match_parent"
       android:layout_height="match_parent"/>

</LinearLayout>
```

Modify style.xml

```xml
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/primary_color</item>
        <item name="colorPrimaryDark">@color/primary_dark_color</item>
        <item name="actionBarStyle">@style/MiwokAppBarStyle</item>
        <item name="android:windowContentOverlay">@null</item>
    </style>

    <!-- App bar style -->
    <style name="MiwokAppBarStyle" parent="style/Widget.AppCompat.Light.ActionBar.Solid.Inverse">
        <!-- Remove the shadow below the app bar -->
        <item name="elevation">0dp</item>
    </style>

    <!-- Style for a tab that displays a category name -->
    <style name="CategoryTab" parent="Widget.Design.TabLayout">
        <item name="tabIndicatorColor">@android:color/white</item>
        <item name="tabSelectedTextColor">@android:color/white</item>
        <item name="tabTextAppearance">@style/CategoryTabTextAppearance</item>
    </style>

    <!-- Text appearance style for a category tab -->
    <style name="CategoryTabTextAppearance" parent="TextAppearance.Design.Tab">
        <item name="android:textColor">#A8A19E</item>
    </style>
</resources>
```