## Current State of the Code

If you skipped the previous nodes and are re-joining us, welcome back! At this point, your app should play the corresponding audio file when you click on each word. Congratulations on getting the feature working! You can compare your code with the latest version of our [**code**](https://github.com/udacity/ud839_Miwok/tree/aa0a0d83fc4a036df53e297fae12b96fcde61a8a) on GitHub.

## Debugging Tip

Before we continue with the rest of the app, I would like to provide a quick debugging tip, if you didn’t already know it.

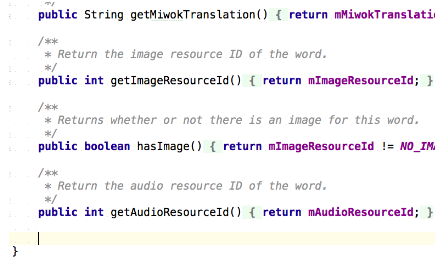
You can use log statements to figure out the state of any Java object. The easiest way to print out the contents of a Java object is to provide an implementation of toString() method. The purpose of this method is to represent the whole object as a string, usually for debugging purposes.

## Word Class Example

Let me show you an example with the Word object. Within the Word class, override the toString method.

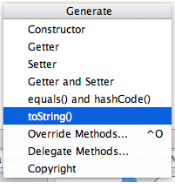
**1)** Put your cursor in an empty space in the file (that is outside of the existing methods, but still within the class).

See example:

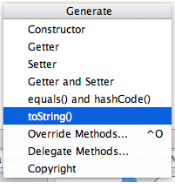


**2)** Then use a keyboard shortcut to automatically “Generate a method (Getters, Setters, Constructors, toString, etc..)”. On Windows, the keyboard shortcut is ALT + Insert. On Mac, the keyboard shortcut is CMD + N.

**3)** You should see this dialog pop-up. Select toString().



**4)** Accept the default choices by hitting OK.



**5)** This toString() method will be automatically generated by Android Studio and added to your Word class. The method concatenates a bunch of variables and text, in order to print out all the variables in your Word class as a string. As you can see, the return value of the method is a single String.

*/\*\**

*\* Returns the string representation of the {@link Word} object.*

*\*/*

@Override

**public** String **toString**() {

**return** "Word{" +

"mDefaultTranslation='" + mDefaultTranslation + '\'' +

", mMiwokTranslation='" + mMiwokTranslation + '\'' +

", mAudioResourceId=" + mAudioResourceId +

", mImageResourceId=" + mImageResourceId +

'}';

}

**6)** Then whenever you have a Word object, such as within the onItemClick() method of the OnItemClickListener, you can print out the Word object to the logs.

// Set a click listener to play the audio when the list item is clicked on listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {

// Get the {@link Word} object at the given position the user clicked on

Word word = words.get(position);

Log.v("NumbersActivity", "Current word: " + word);

// Create and setup the {@link MediaPlayer} for the audio resource associated with the current word

mMediaPlayer = MediaPlayer.create(PhrasesActivity.this, word.getAudioResourceId());

// Start the audio file

mMediaPlayer.start();

}

});

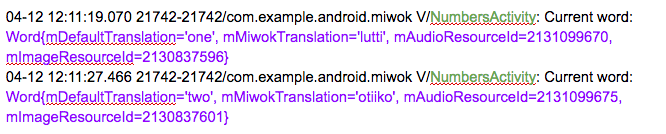
**Note:** If you concatenate (with the “+” operator) a string with a Word object, then Java will implicitly call the toString() method on the object. That means, these two statements are equivalent:

Log.v("NumbersActivity", "Current word: " + word);

OR

Log.v("NumbersActivity", "Current word: " + word.toString());

**7)** Then in the Android logcat, you will something like this output. I highlighted the log tag in green font color (NumbersActivity), and I highlighted the return value of the Word toString() in purple font color.



To summarize, we’ve just covered how you can override the toString() method of any class, which is optional. However, it’s a helpful practice because in the future, you may want to print the current state of an object to the logs (instead of using the step-by-step debugger in Android Studio).

16\_**Quiz: Async Callbacks In Android**

***Hint:***Setup the callback ONLY after you have called the mediaPlayer.start() method, otherwise the callback won’t get triggered.

17\_**Quiz: Cleaning Up Media Resources**

**Code snippet:** [**Helper method**](https://gist.github.com/udacityandroid/f4ec40027031ba7de9352465f143c816) to release media player resources. Make sure that you have a global MediaPlayer variable called mMediaPlayer in your category activity files, in order for the code snippet to work. **Additional resources:**

* Documentation for the MediaPlayer [**release()**](http://developer.android.com/reference/android/media/MediaPlayer.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics#release()) method
* [**MediaPlayer state diagram**](http://developer.android.com/reference/android/media/MediaPlayer.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics#StateDiagram)

[**These are the changes needed to complete the quiz.**](https://github.com/udacity/ud839_Miwok/commit/c2a89bc69a3af9e8aa6d37f1fdaa16e3f6dc4514) [**This is the state of the code after completing the quiz.**](https://github.com/udacity/ud839_Miwok/tree/c2a89bc69a3af9e8aa6d37f1fdaa16e3f6dc4514)

Emoji art supplied by [**http://emojione.com**](http://emojione.com/)

21\_ **Quiz: Mediaplayer And Activity Lifecycle**

**Hint:** **[MediaPlayer release() method](http://developer.android.com/reference/android/media/MediaPlayer.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics" \l "release()" \t "_blank)**

[**These are the changes needed to complete the quiz.**](https://github.com/udacity/ud839_Miwok/commit/315130f547fe9b853d59fb01fd77426a7b19ba8c) [**This is the state of the code after completing the quiz.**](https://github.com/udacity/ud839_Miwok/tree/315130f547fe9b853d59fb01fd77426a7b19ba8c)

**22\_Quiz: Learn About Audio Focus**

Skim the following articles to answer the quiz questions.

* [**How to properly handle audio interruptions**](https://medium.com/google-developers/how-to-properly-handle-audio-interruptions-3a13540d18fa#.jkibca8ml) by Developer Advocate Joanna Smith
* [**Managing Audio Focus**](http://developer.android.com/training/managing-audio/audio-focus.html)

23\_ **Video: AudioManager Service**

In this course, we won’t be creating our own service, but you can learn more about services in Android [**here**](http://developer.android.com/guide/components/services.html).

For now, just know that AudioManager is a system service. You can interact with it like any other Java class.

24\_ **Quiz: Request Audio Focus**

For a list of possible streamType values, see the constants that start with the name “STREAM\_” in the **[AudioManager](http://developer.android.com/reference/android/media/AudioManager.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics" \t "_blank)**class.

For a list of possible durationHint values, see the constants in the parameter description of the **[requestAudioFocus](http://developer.android.com/reference/android/media/AudioManager.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics" \l "requestAudioFocus(android.media.AudioManager.OnAudioFocusChangeListener,%20int,%20int)" \t "_blank)**method.

**Hint:** They should both be integer constants from the AudioManager class (spelled in capitalized letters)

25\_ **Quiz: Audio Focus States**

See possible audio focus states in **[OnAudioFocusStateChangeListener](http://developer.android.com/reference/android/media/AudioManager.OnAudioFocusChangeListener.html" \t "_blank)**

[**How to properly handle audio interruptions**](https://medium.com/google-developers/how-to-properly-handle-audio-interruptions-3a13540d18fa#.jkibca8ml)

26\_ **Quiz: Manage Audio Focus In The App**

**Previous Quiz Solutions:**

* [**Solution from Learn About Audio Focus quiz**](https://docs.google.com/document/d/11LgNsw9YXuBJubnQaRCIH-WZfiGbWMiaaueuKdwb5fE/pub)
* [**Solution from Request Audio Focus quiz**](https://docs.google.com/document/d/1uCVf6phRJYTzYycZ2zyreFXqvs6sJG-kUN_JjQCqVUo/pub)
* [**Solution from Audio Focus States quiz**](https://docs.google.com/document/d/1JHliOqs-YRKzxGMinUSNjVgg1hgvE3r2uVcedD3JUV8/pub)

**Articles:**

* [**How to properly handle audio interruptions**](https://medium.com/google-developers/how-to-properly-handle-audio-interruptions-3a13540d18fa#.jkibca8ml)
* [**Managing Audio Focus**](http://developer.android.com/training/managing-audio/audio-focus.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics)

**Reference documentation:**

* [**AudioManager**](http://developer.android.com/reference/android/media/AudioManager.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics)
* [**OnAudioFocusStateChangeListener**](http://developer.android.com/reference/android/media/AudioManager.OnAudioFocusChangeListener.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics)

**(Optional) Mid-way checkpoint:**After you’ve attempted task #2, if you want to check your work against our implementation, here’s the [**code**](https://gist.github.com/anonymous/6d00bb2e8524fb78bb50f118a6180b8a) for the OnAudioFocusChangeListener.

[**These are the changes needed to complete the quiz.**](https://github.com/udacity/ud839_Miwok/commit/2b548eae0e57d7ad0cbb9c77656fd24215ec6cb4) [**This is the state of the code after completing the quiz.**](https://github.com/udacity/ud839_Miwok/tree/2b548eae0e57d7ad0cbb9c77656fd24215ec6cb4)

Emoji art supplied by [**http://emojione.com**](http://emojione.com/)

27\_ **Quiz: Add Audio Icon To List Item Layout**

Download the [**play arrow icon**](https://design.google.com/icons/#ic_play_arrow) from the Material Design icons site (use the white 36dp version). Remember to include versions of the icon for all densities into your app (from mdpi → xxxhdpi).

[**These are the changes needed to complete the quiz.**](https://github.com/udacity/ud839_Miwok/commit/387971be04f57f04e175ebaaa836f2947232bff2) [**This is the state of the code after completing the quiz.**](https://github.com/udacity/ud839_Miwok/tree/387971be04f57f04e175ebaaa836f2947232bff2)

28\_

**Quiz: Touch Feedback For Clickable Views**

Emoji art supplied by [**http://emojione.com**](http://emojione.com/)

# 29\_29\_ Pressed States on Category Views

# 29\_ Pressed States on Category Views

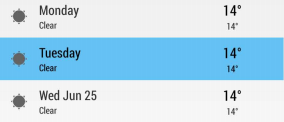
Tour Guide App

Touch feedback provides the user with instantaneous visual confirmation at the point of contact where they interacted with the UI elements on the screen. It is vital that the apps you develop on Android have touch feedback. They can make your app appear fast and responsive, even if nothing else happens yet.

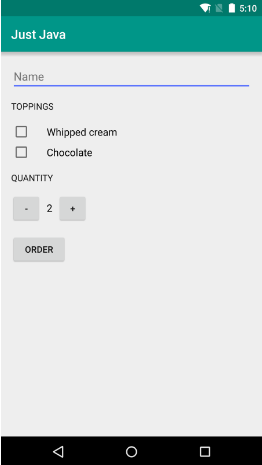
Ever since Material Design was introduced in Lollipop, there is a circular ripple animation that happens when you interact with a UI elements.



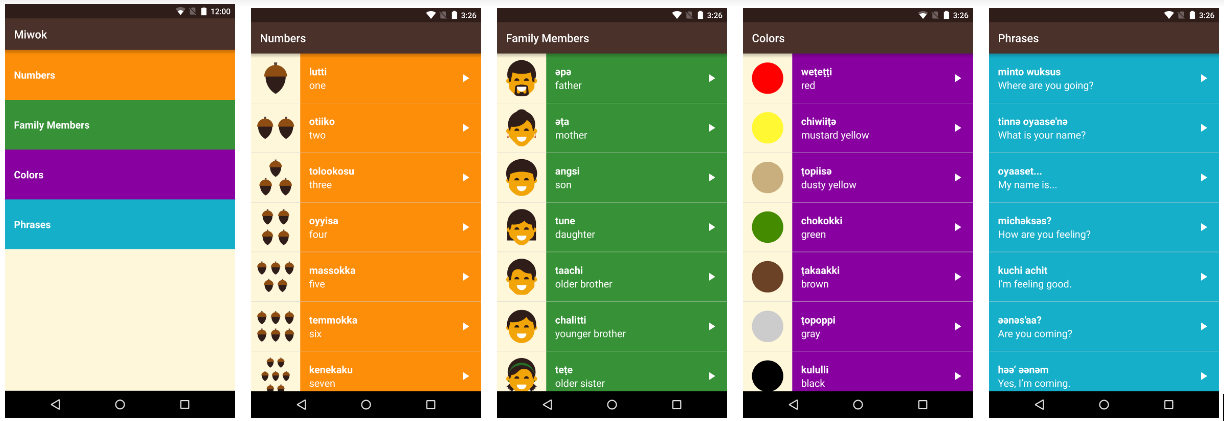
On older versions of Android, there is a static colored pressed state.



In the last course, when we made the Just Java app, we used the standard Buttons from the Android framework. These buttons come up with standard feedback when you touch them.



In the Miwok app, we are creating our own clickable views - the category views in the MainActivity, and the list items in the category activities. We have to handle pressed states on our own.



By setting a view to have a background of ?android:attr/selectableItemBackground, the View will have a transparent background by default. When you touch or press the view, it will show the pressed state (which is the circular ripple animation). This behavior is the default touch feedback behavior defined in the Android framework.

## Modify the Miwok app

In the last exercise, you modified the numbers category TextView to have the selectable item background. However, we lost the colored background.

**1)** In this coding task, replace the contents of the activity\_main.xml with the below XML. Code is also provided in this [gist](https://gist.github.com/udacityandroid/e11b81e3bdcff1c073c6e24a4fdb3365). Each TextView is wrapped in a FrameLayout with a themed background, and then set the background of the TextView to be a ?android:attr/selectableItemBackground.

This way, we can keep the colored background per category view, as well as have touch feedback. And by the way, a [FrameLayout](http://developer.android.com/reference/android/widget/FrameLayout.html" \t "_blank) is simply a ViewGroup that typically contains 1 child view.

**In activity\_numbers.xml file**

**<?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">

*<!-- Numbers category -->*

<FrameLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="@color/category\_numbers">

<TextView

android:id="@+id/numbers"

style="@style/CategoryStyle"

android:background="?android:attr/selectableItemBackground"

android:text="@string/category\_numbers" />

</FrameLayout>

*<!-- Family category -->*

<FrameLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="@color/category\_family">

<TextView

android:id="@+id/family"

style="@style/CategoryStyle"

android:background="?android:attr/selectableItemBackground"

android:text="@string/category\_family" />

</FrameLayout>

*<!-- Colors category -->*

<FrameLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="@color/category\_colors">

<TextView

android:id="@+id/colors"

style="@style/CategoryStyle"

android:background="?android:attr/selectableItemBackground"

android:text="@string/category\_colors" />

</FrameLayout>

*<!-- Phrases category -->*

<FrameLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="@color/category\_phrases">

<TextView

android:id="@+id/phrases"

style="@style/CategoryStyle"

android:background="?android:attr/selectableItemBackground"

android:text="@string/category\_phrases" />

</FrameLayout>

</LinearLayout>

**2)** Then run the app on your device and test that each button in the MainActivity has touch feedback.

**Note:** You could move the background attribute to the CategoryStyle definition in styles.xml, but I just declared it inline here, so it’d be easier for you to add into your app.

### More Resources

We showed you a simplified way of adding touch feedback in your MainActivity in order to avoid going on too far of a tangent with new topics (such as state list drawables and managing resources for backward compatibility on older devices). However, our current approach does introduce more views into the view hierarchy, so it is less efficient.

If you want to learn the proper and more efficient way to setup touch feedback with a custom drawable, check out this video from the intermediate [Udacity Android Fundamentals course](https://classroom.udacity.com/courses/ud853/lessons/1623168625/concepts/16358494290923" \t "_blank). Or you can check out this [Codepath tutorial](http://guides.codepath.com/android/ripple-animation" \t "_blank) on ripple animations.

30\_Now, you need to make one last change so that in all the category activities (NumbersActivity, FamilyActivity, etc…), list items show a pressed state when clicked on.

## OPTION #1

In the list item layout, you can add a new View that overlaps all other views in the RelativeLayout and matches the RelativeLayout’s width and height. The View has a background of "?android:attr/selectableItemBackground”, so the View will be transparent by default, so you can see the contents of the list item. Then it will show a pressed state (gray ripple animation on Lollipop devices and above) when the list item is clicked on.

[Gist](https://gist.github.com/udacityandroid/e9399aa9b3ffb7690d15b4b7fbe148db) of modified list\_item.xml

## OPTION #2

Instead of adding a new view to the layout, in the word list layout, you can add the attributeandroid:drawSelectorOnTop="true" on the ListView XML element. With this one line change, the pressed state will be shown on each list item.

In word\_list.xml:

**<?xml version="1.0" encoding="utf-8"?>**

<ListView xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/list"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:drawSelectorOnTop="true"/>

Either option would work. After you make this change, test your app. Then you’re done with Lesson 4!