Buse Çarık 10/08/18

INTERRA R&D Material Design

Material Design

Snackbars

https://github.com/InterraMaterialDesign/Snackbars

**INDEX**

1. **INTRODUCTION…………………………………………………….…………………3**
   1. **Purpose of this document………………………………………………………..….3**
   2. **Overview……………………………………………………………………………..3**
2. **WHAT IS SNACKBARS……….……………………………………………………….4**
   1. **Material Design Concept……………………………………………………………6**
   2. **Customizing a Snackbar……………………………………………….…….……10**
   3. **Positioning…………………………………………….……………………………14**
   4. **Other Inherited Methods……………………….…………………………………15**
3. **ADVANTAGES AND DISADVANTAGES………………..…………………………15**
4. **REQUIREMENTS FOR IMPLEMENTATION…………….….……………………16**
5. **AREAS OF USAGE…………………………………………………………………….16**
6. **REFERENCES…………………………………………….……………………………17**

**INTRODUCTION**

**PURPOSE OF THIS DOCUMENT**

The aim of this project is analyzing and implementing Snackbars.

**OVERVIEW**

The project that is described in this document, examine the Snackbars’ description, where and when it is used and how it should be designed according to Material Design.

**Snackbars**

Snackbars inform the user about an operation that has occurred or will occur. They are introduced with Material Design as an alternative to pop-up messages and they are still under development. They are extended from android.support.design.widget.BaseTransientBottomBar. They appear at the bottom of the screen on mobiles and bottom left on large devices. They usually dismiss without requirement of an action.

“Snackbar will only work in Activities that are ***AppCompatActivity***. If your activity extends a standard Activity or ActionBarActivity then you will need to either change your base class or use a third-party implementation of a Snackbar.”

To create a snackbar ***make***(View view, CharSequence text, int duration) method is called. It takes three parameters. First it needs a view to attach own view. Then, the short message that is reached to the user is entered with a CharSequence or with the resource id of the string resource. This text can be updated later with ***setText***(CharSequence message) method. The last parameter sets the display duration of the snackbar. To make visible the snackbar on the screen, you should use ***show***() method.

View containerView = findViewById(R.id.coordinator);

Snackbar.make(containerView, "Normal snackbar", Snackbar.LENGTH\_LONG)

.show();

Snackbar snackbar = Snackbar.make(containerView, R.string.custom\_snackbar, Snackbar.LENGTH\_LONG);

snackbar.show();

They have constants for duration which are ***LENGTH\_SHORT***, ***LENGTH\_LONG*** and ***LENGTH\_INDEFINITE***. In the short length option, the snackbars are displayed for short period of time like 4 seconds; in the long length option, they are shown for long period of time which is 10 seconds. In indefinite option, the snackbars are stayed for an indefinite time, they are disappeared with swipe-off action. The swipe-off action is only available in direction right. One of the inherited methods is ***setDuration***(int) and with that method these constants can be customized. Or without these constants and method, you can set the duration with numbers.

The most efficient way to use the snackbars is to work with the ***Coordinator Layout***. Because, snackbars are attached to a view and if they are attached to the coordinator layout, the snackbars gain extra features. First, it ensures the best placement for snackbars with other bottom UI components like FAB and navigation with providing interaction between child views. When a snackbar is appeared on the screen, the Coordinator Layout shifts up automatically the UI components like FAB to prevent the overlap. Also, it provides the swipe-to-dismiss behavior. Therefore, when a snackbar is created with ***make***() method, passing coordinator layout as a view parameter is the most efficient way.

One of the important features of the snackbar is that you can add action to the snackbars with ***setAction***(CharSequence text, View.OnClickListener listener) method. Actions can be for undo or retry operations. The text also can be resource id of string resource, and it provides text for the text button. View.OnClickListener performs click on the action. When the text button is clicked, it calls the listener’s ***onClick***() method then the snackbar will disappeared.

snackbar.setAction("RETRY", new View.OnClickListener() {

@Override

public void onClick(View view) {

//code when the user is clicked

}

});

To be notified when a snackbar is displayed or dismissed; Snackbar.Callback class can be provided. To add this class, ***addCallback***(BaseCallback<B> callback) method can be used. This class has two methods which are ***onDismissed***(Snackbar transientBottomBar, int event) and ***onShown***(Snackbar sb). onDismissed method is called when a snackbar is gone from the screen. onShown method is called when a snackbar becomes visible on the screen.

\*\*\*Multiple snackbars cannot be shown on a screen at the same time, they should be appeared in order. To show another snackbar, the first one’s duration must be finished or is dismissed.

**Material Design Concept**

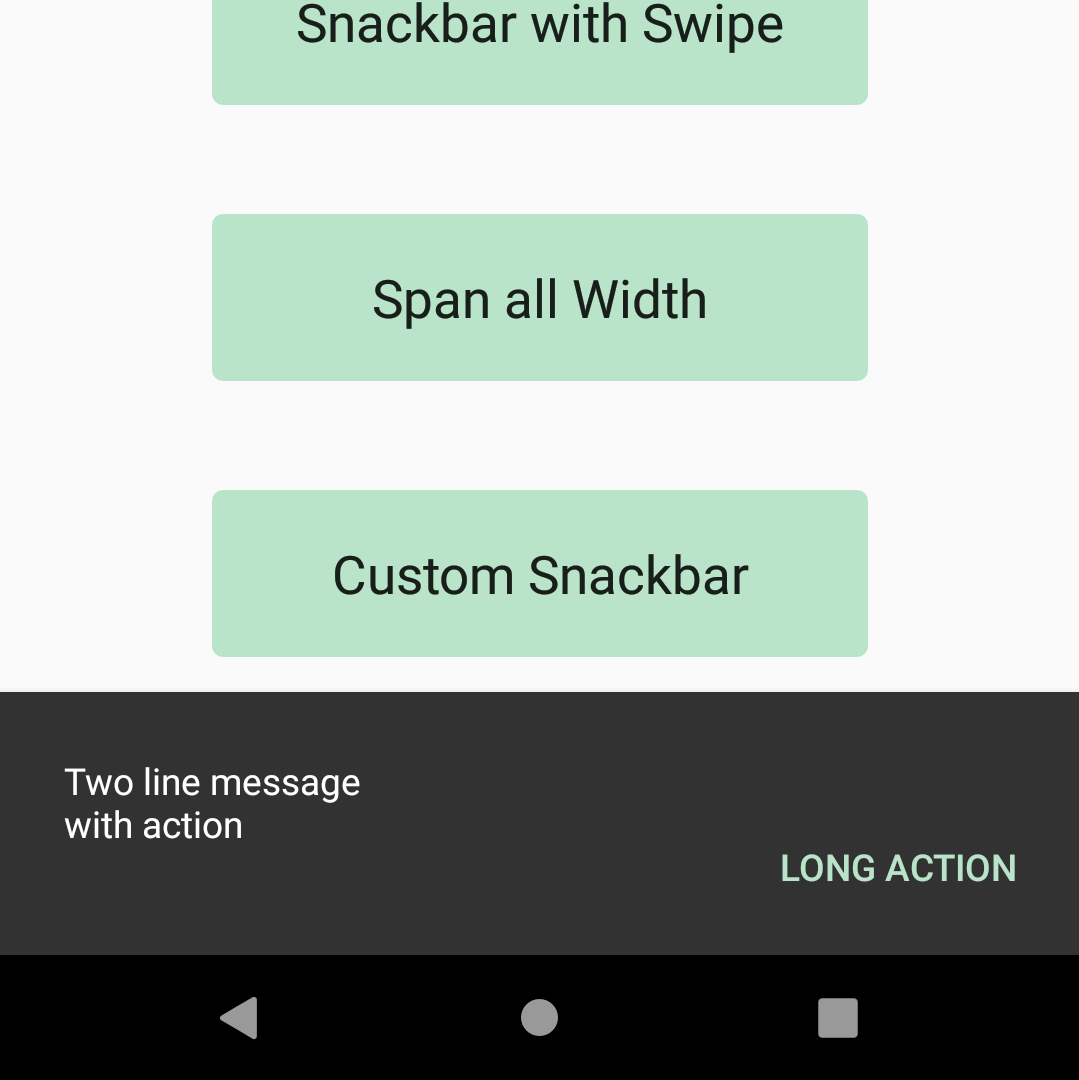
According to Material Design, a snackbar consists of a text label and an optional action within a container. A snackbar cannot have multiple actions. The action cannot be “cancel” or “dismiss” because most of the time, they are disappeared automatically. The snackbars perform actions with text buttons. Text button’s color should be distinguishable from the text label. By default, text button’s color is colorAccent. You should not use filled or elevated button instead of text button because it draws a lot attention.

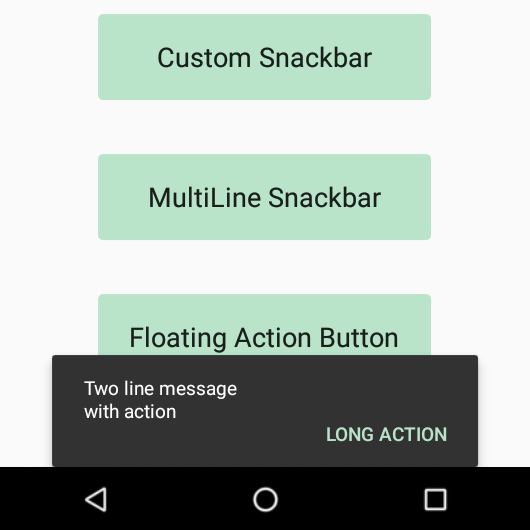
The text labels should consist of at most two lines on mobile; in desktops and tablets should be a single line. In long actions, the action can be in third line. In mobile, to use the multiline text, you can set the maximum number of lines of the text view in the snackbar.

View snackbarView = snackbar.getView();

TextView textView = snackbarView.findViewById(android.support.design.R.id.snackbar\_text);

textView.setMaxLines(3);

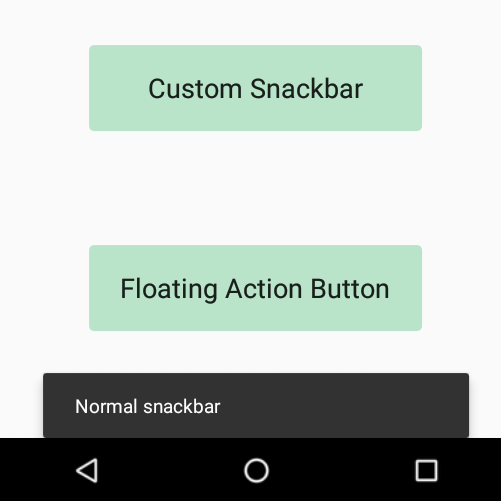


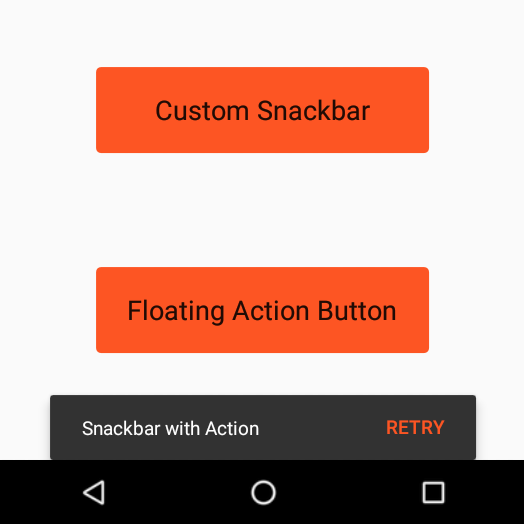


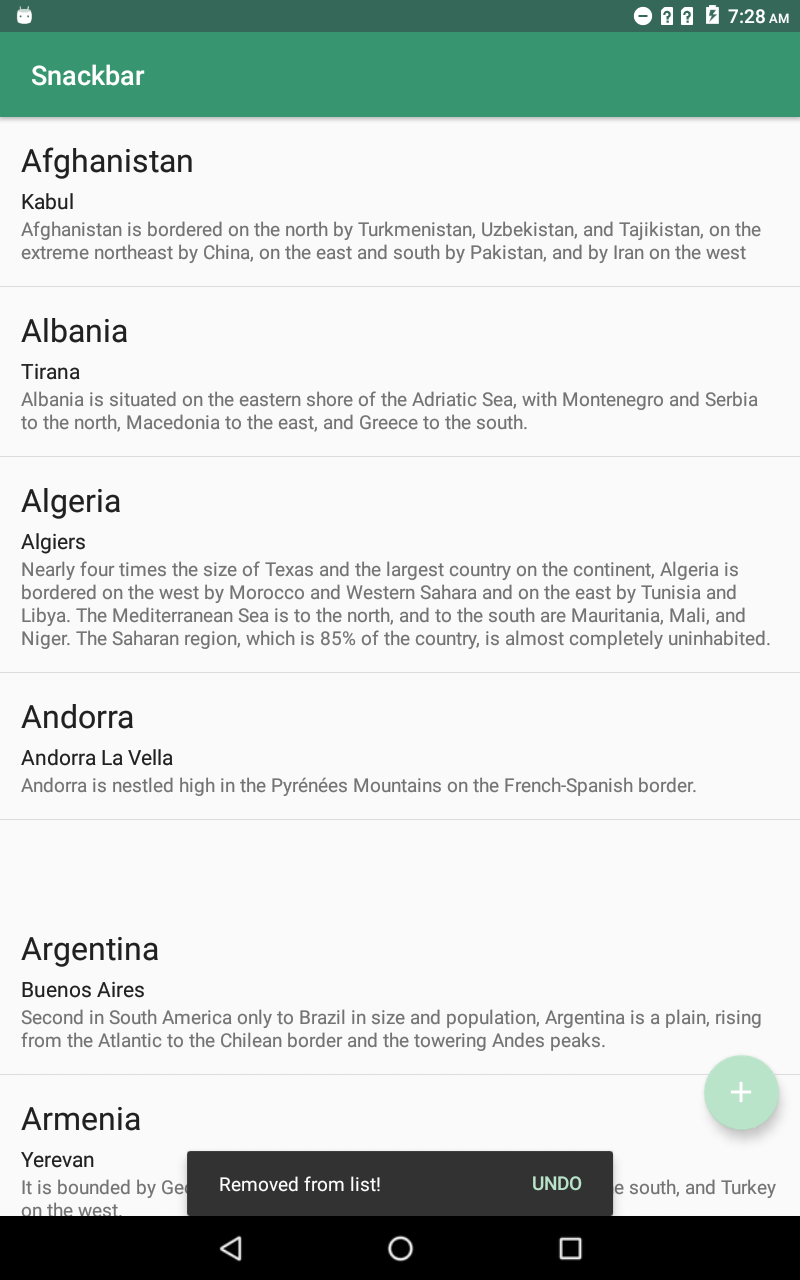
The containers have grey(#323232) background. For a readable text, the container’s background should be opaque. And, it should have elevation.

\*\*\*Snackbars do not contain icons. If you need an icon, you should prefer alert.

Snackbars should not block the Floating Action Button. They are not placed front or behind of the FAB.







**Customizing A Snackbar**

You can customize the appearance of the snackbar. Firstly, you can change the text color of the action with ***setActionTextColor***(ColorStateList colors) method (or setActionTextColor(int color)).

Snackbar.setActionTextColor(getResources().getColor(R.color.black));

By default, the text label’s color is white. To customize this color:

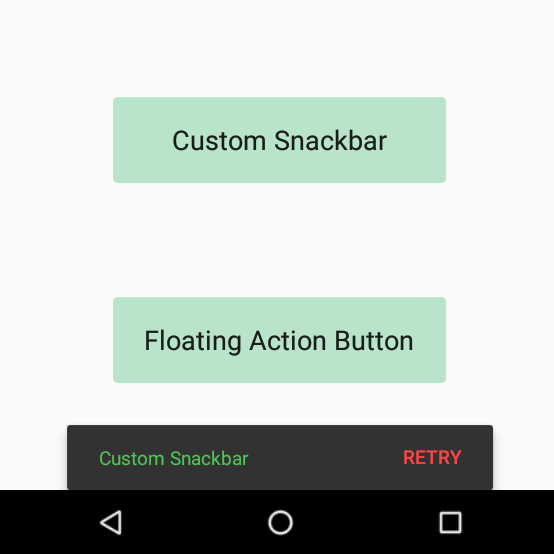
int snackbarTextId = android.support.design.R.id.snackbar\_text;

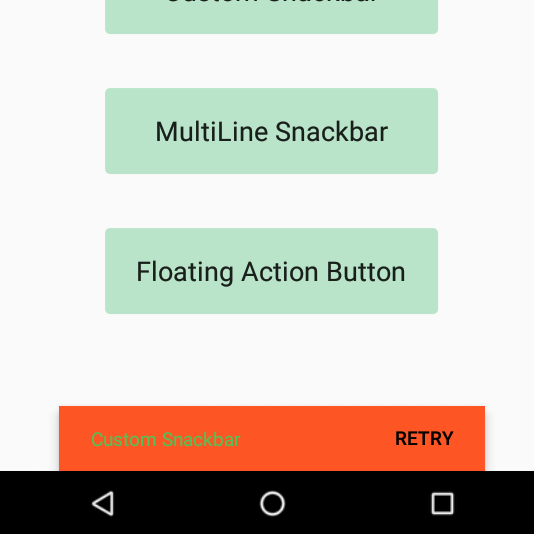
TextView textView = (TextView) snackbar.getView().findViewById(snackbarTextId);

textView.setTextColor(getResources().getColor(R.color.green));

The background color also can change with ***setBackgroundColor***(int color) method.

Snackbar.getView().setBackgroundColor(Color.colorSecondary);



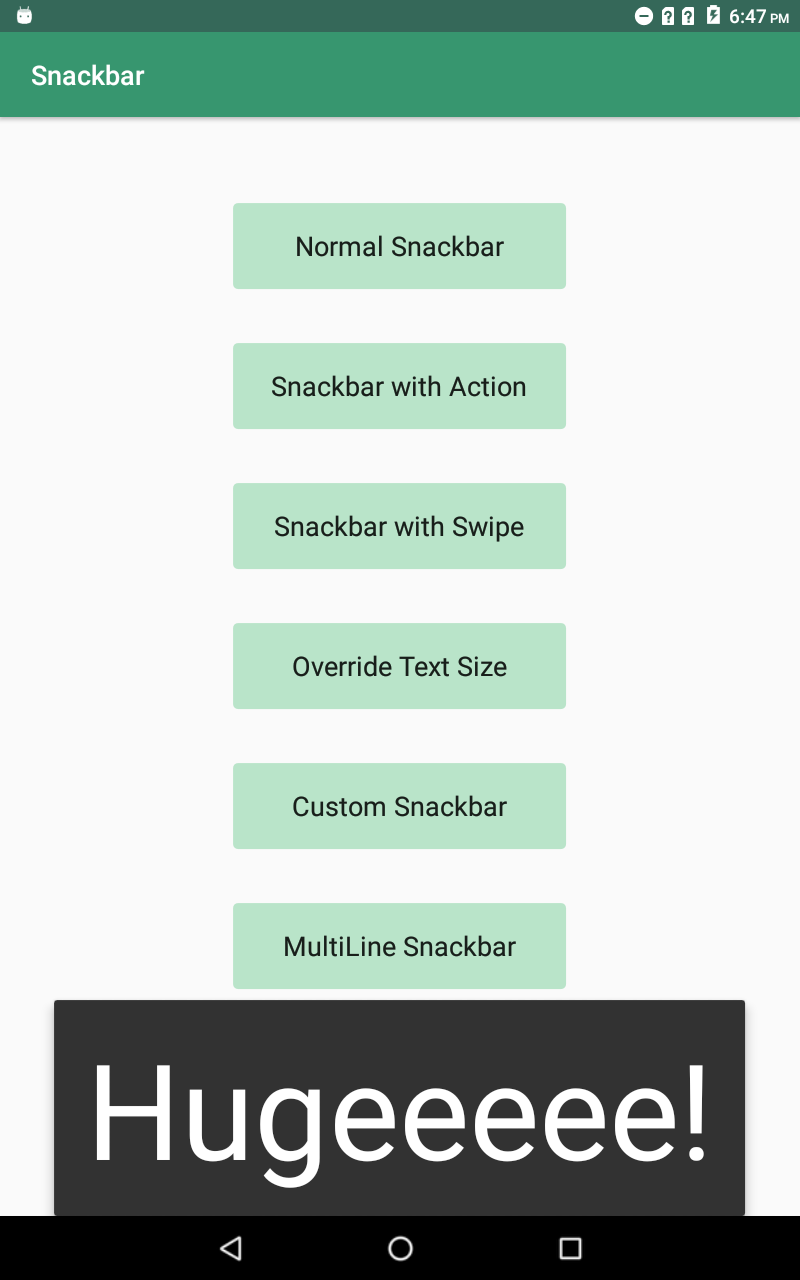


Snackbars’ size of text view can also be customized. You can change the size of the text. Same as the changing the color of the text. First, you should get the snackbar’s text view.

int snackbarTextId = android.support.design.R.id.snackbar\_text;

TextView textView = snackbar.getView().findViewById(snackbarTextId);

textView.setTextSize(100);



Size of the snackbar cannot be directly customized but you can reach the snackbar’s layout. Then you can change the width and height of the snackbar, also you can further customize your snackbar with features from the layout.

Snackbar.SnackbarLayout layout = (Snackbar.SnackbarLayout) snackbar.getView();

int width = containerView.getWidth();

layout.setMinimumWidth(width); layout.setBackgroundResource(R.drawable.corner\_radius);

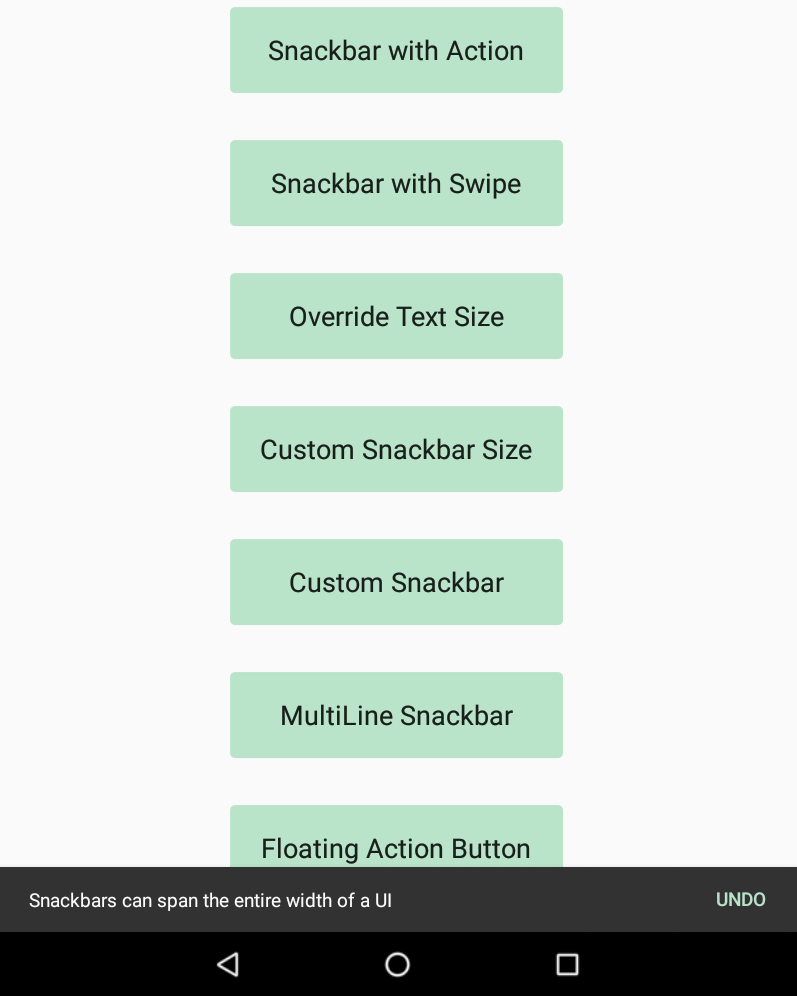
In that example, we tried to span the entire width of the UI with snackbar. MATCH\_PARENT constant cannot work in ***setMinimumWidth***() method hence, I use the coordinator layout’s width to reach the width of the screen. But the issue is that the snackbar’s corners have radius. To remove the roundness of the corners, a drawable resource file is used.

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

<corners android:radius="0dp"/>

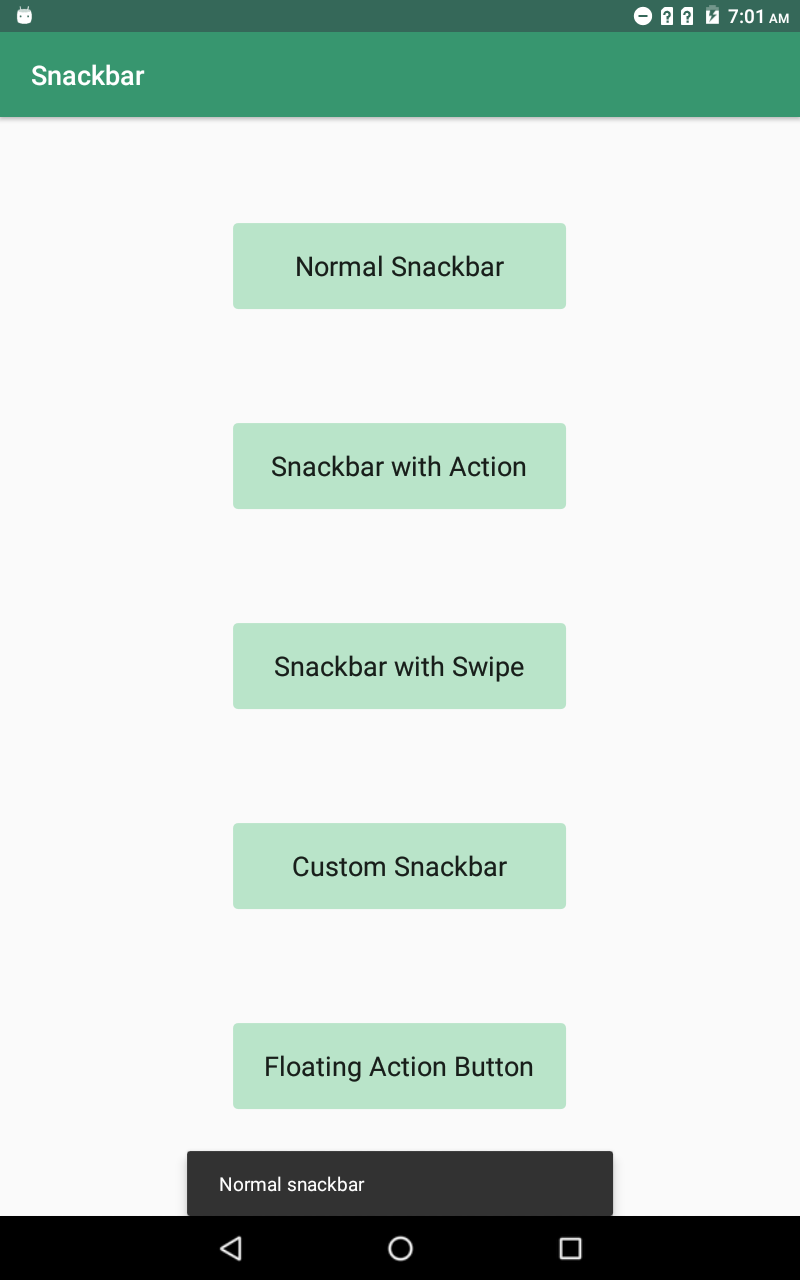
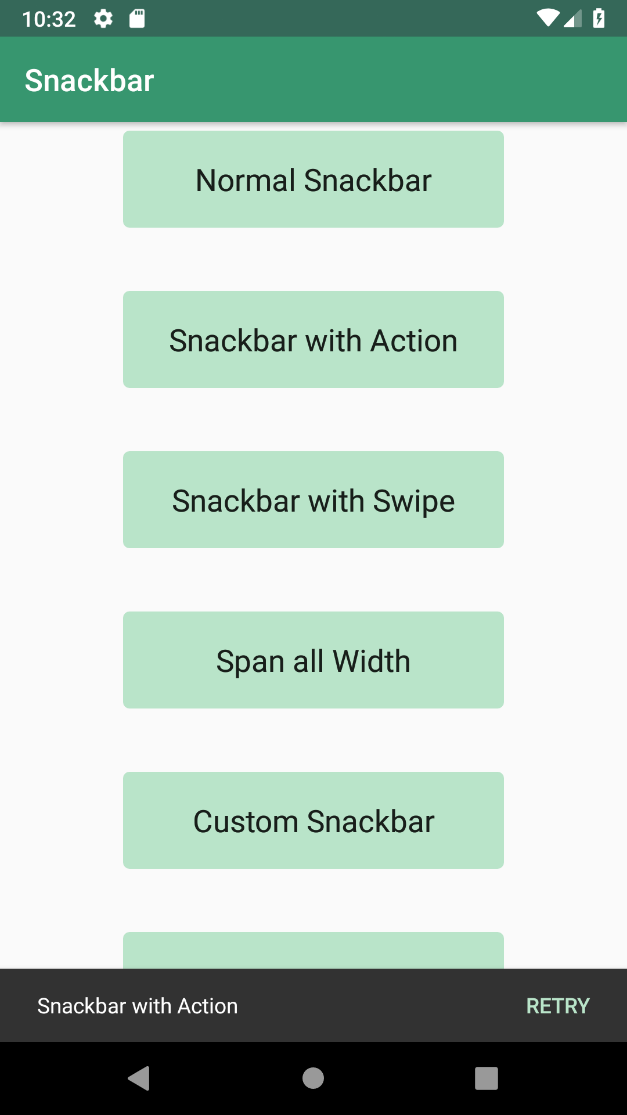
<solid android:color="#323232"/>

</shape>



**Positioning**

It is only possible to control the position of the Snackbar if you are using a CoordinatorLayout. By default, on mobile devices the Snackbar takes up the entire width of the screen, while on tablets and desktops the Snackbar does not take up the entire width of the device. The snackbars should appear in the bottom of the screen and in front of the app content. But they should not be placed in front of the navigation components and FAB. According to the Material Design, the snackbars should be place above the bottom navigation bar. The snackbars also should be appear above the footer elements. And in sheets, they are placed in front of them.



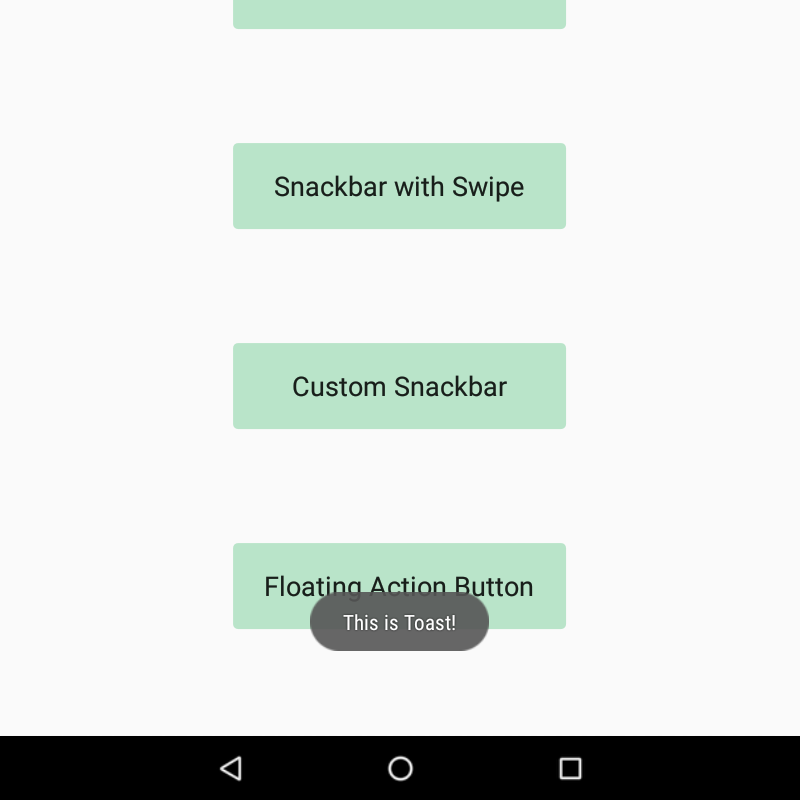
**Other Inherited Methods**

* dismiss(); dismisses the snackbar.
* getContext(); returns the snackbar 's context.
* getDuration(); returns the duration.
* getView(); returns the snackbar's view.
* isShown(); returns whether this snackbar is currently being shown.
* isShownorQueued(); returns whether this snackbar is currently being shown or is queued to be shown next.
* removeCallback(); removes the specified callback from the list of callbacks.

**Advantages and Disadvantages**

The snackbar is introduced with Material Design for an alternative to ***Toast***. Toasts are used for system-level notifications, but if you want to display a feedback message, you should prefer the snackbars.

There are two main differences between the snackbars and the toasts. One of them is that snackbars may contain single action but toasts cannot perform an action. Other one is that toast cannot stay on the screen permanently and it does not have swipe-to-dismiss behavior.



**Requirements for Implementation**

Snackbars are part of android design library. They are added with revision 22.2.0. To use snackbars, add the following requirements to the build.gradle file in the project:

dependencies {  
    implementation ' com.android.support:appcompat-v7:28.0.0 -alpha1'  
 implementation 'com.android.support: design:28.0.0 -alpha1’ }

Also, the Coordinator Layout is adding to the XML file as the following:

<android.support.design.widget.CoordinatorLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

</android.support.design.widget.CoordinatorLayout>

**Areas of Usage**

The snackbars are used for informing the users about operations. For example, its usage areas can be to give a validation error, confirming to user that an action was taken or notifying them to an update. Also, snackbars itself can perform an action like “UNDO”, “RETRY” or “OVERRIDE”. Undo is used to undoing actions that was just taken or Retry is for retyring an action that failed. These are often observed in list views such as email lists.

**References**

Android Developers. (n.d.). Retrieved June 27, 2018, from https://developer.android.com/

Android Material Design Snackbar Example. (2017, July 12). Retrieved August 10, 2018, from https://www.androidhive.info/2015/09/android-material-design-snackbar-example/

Android Material Design Snackbar Tutorial. (2017, May 01). Retrieved August 10, 2018, from http://www.androidtutorialshub.com/android-material-design-snackbar-tutorial/

Android Snackbar Example. (2015, June 10). Retrieved August 10, 2018, from https://www.truiton.com/2015/06/android-snackbar-example/

Homepage. (n.d.). Retrieved June 27, 2018, from <https://material.io/>

Lightfoot, J. (2017, July 10). Android Snackbar Tutorial: Setup, Action Handling, and UI Customization. Retrieved August 10, 2018, from https://spin.atomicobject.com/2017/07/10/android-snackbar-tutorial/

Material Design Snackbar in Android - How and When to Use. (2017, April 03). Retrieved August 10, 2018, from https://blog.iamsuleiman.com/material-design-snackbar/