Create new xml file in vaues for sdk more than 21 version and set directory name values-v21

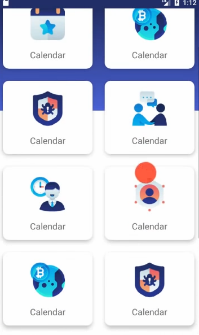
Then copy styles from common style file and before each name in items add android:? It is needed only for android upper than 21

Then extend main activity from Activity to avoid errors, but in versions lover than 21 you won’t have action bar

Then add to dependencies:

implementation 'com.android.support:appcompat-v7:28.0.0'

then delete common action bar and set toolbar



create like in photo

gridLayout (columnCount, rowCount -> CardView (layout\_columnWeight=”1”, layout\_rowWeight=”1” -> LinearLayout (orientation=”vertical”, gravity=”center”) -> ImageView , TextView…

imageView

scaleType=”centerCrop or fitXY” – crop image

ImageButton

Make src icon bigger

android:scaleType="fitCenter"  
android:scaleX="1.2"  
android:scaleY="1.2"

set image from server

import implementation 'com.squareup.picasso:picasso:2.5.2'

then:

Picasso.with(context).load(url).into(imageView);

To Picasso can also add:

.memoryPolicy(MemoryPolicy.NO\_CACHE) - is used to prevent loading the image from the stored cache.

.memoryPolicy(MemoryPolicy.NO\_STORE) - is used to not store the image in the cache at all

Picasso.with(context).load(image\_url).memoryPolicy(MemoryPolicy.NO\_CACHE).into(imageView); - example

.placeholder(R.drawable…) - A placeholder is commonly used to display a drawable image while the main image gets loaded into the imageview. This is essential in cases when the image takes time to load from the web.

.error() - An error drawable is generally used in cases where there’s a failure in loading the image. In this case the interim placeholder image gets replaced by the error drawable that’s placed inside .error() method

.priority(HIGH/NORMAL/LOW)

gradient

create in draw able xml file with shape root and:

<gradient

android:startColor=”…”

android:endColor=”…”

android:angle=”…”/>

then set background somewhere as this file

LinearLayout

android:layout\_weight="1" – this element will fill all free space

RelativeLayout

Layout\_cenerInParent=”true” – set view at the center of its parent

app:layout\_constrainedWidth="true" – default size will be wrap content, but if it became smaller than its constrains it will became smaller to

app:layout\_constraintWidth\_percent="0.5" – set size as …% from its parent

app:layout\_constraintDimensionRatio="10:5" – set size as relation one to another and set fill parent

XML commands

Elevation – shadow

Layout\_alignParentRight = “true” – set beginning from right

Gravity = “center” – display this block at center

Layout\_toRightOf=”id” – place this element right from …

Layout\_below=”id” - place this element under the…

Clickable=”true” – can be clicked

android:layout\_gravity="bottom|right" – place item

**.setVisibility(View.GONE)** – don’t display

android:background="@android:color/transparent" – invisible background

android:alpha="…" – opacity

background="@null" – clear background

app:layout\_constraintBottom\_toBottomOf="parent" – place at the bottom of the parent  
app:layout\_constraintStart\_toStartOf="parent" – place at the left of the parent

android:layout\_weight="1" – if block is match parent, this command will move it behind other items

app:layout\_goneMarginBottom="10dp" – minimal margin from main block (if you delate item from what you had constrains)

textView

autoSizeTextType=”uniform” – at TextView it will change textSize filling full space (as more text as lower size)

autoSizeMinTextSize=”…sp” or autoSizeMaxTextSize=”…sp” – set min and max size if autoSizetextType=”uniform”

Animations

Create new resorse file Animation, then there:

<set xmlns:android="http://schemas.android.com/apk/res/android"  
 android:fillAfter="true"> - for avoiding errors

<translate – when activity created move item from… to current position  
 android:duration="1800"  
 android:fromYDelta="100%p"  
 android:fromXDelta="0%p"  
 />

<scale – increase object or decrease  
 android:duration="300"  
 android:fromXScale="80%"  
 android:fromYScale="80%"  
 android:toXScale="0%"  
 android:toYScale="0%"  
 android:pivotX="50%" – where to do it (50% is center)  
 android:pivotY="50%" – where to do it (50% is center)   
 android:interpolator="@android:anim/linear\_interpolator"/> - linear speed   
  
<alpha – change opacity   
 android:duration="300"  
 android:fromAlpha="1.0"  
 android:toAlpha="0.0"  
 android:interpolator="@android:anim/linear\_interpolator"/> - linear speed

<rotate - rotate  
 android:duration="300"  
 android:fromDegrees="0"  
 android:toDegrees="45"  
 android:pivotY="50%" – where to do it (50% is center)  
 android:pivotX="50%" – where to do it (50% is center)  
 android:interpolator="@android:anim/linear\_interpolator"/> - linear speed

Then create variable Animaton animation and set it from resource file:

animation = AnimationUtils.loadAnimation(this, R.anim.frombottom);

and in onCreate method call method setAnimation(animation) from needed block

fast change images

in drawable create xm file with animation-list root and set items:

<item android:drawable="@drawable/ic\_accessibility\_black\_24dp" android:duration="200"/>  
<item android:drawable="@drawable/ic\_account\_balance\_black\_24dp" android:duration="200"/>  
<item android:drawable="@drawable/ic\_add\_to\_photos\_black\_24dp" android:duration="200"/>

Then in main class create ImageView variable and AnimationDreawable, set them and:

imageView.setBackgroundResource(R.drawable.animation); - set this xml, what you created  
animationDrawable = (AnimationDrawable)imageView.getBackground(); - get animation  
animationDrawable.start(); - start

animationDrawable.stop(); - stop

Customize button

Create xml file in drawable:

<shape xmlns:android="http://schemas.android.com/apk/res/android">  
 <solid – set color (for example gradient)   
 android:color="#000fff"/>  
 <corners – set corners   
 android:bottomLeftRadius="10dp"  
 android:radius="40dp"/>  
</shape>

Then set this xml as background of button

Round button like in telegram

(scroll top in menu)

<android.support.design.widget.FloatingActionButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/design\_default\_color\_primary\_dark" – background color  
 android:onClick="…"  
 android:src="@drawable/…" - icon  
 app:fabSize="mini" /> - size mini or normal

Layouts

<android.support.design.widget.CoordinatorLayout

android:layout\_gravity="bottom|right" – allows to set position

collapse toolbar

add design to grades,

after creating toolbar include it in block:

<android.support.design.widget.AppBarLayout – for toolbar stay on top of all items  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
 <include layout="@layout/toolbar"/>  
</android.support.design.widget.AppBarLayout>

Then in toolbar xml file set properties:

app:layout\_scrollFlags="scroll|enterAlways"

Change size dynamically

ViewGroup.LayoutParams params = recyclerView.getLayoutParams();  
params.height = recyclerView.getHeight() - 60;  
recyclerView.setLayoutParams(params);

Changing text’s appearance

TextView textView = findViewById(R.id.textView);  
SpannableString spannableString = new SpannableString("alalalalalaallalalalala"); // - immutable, can't be changed  
SpannableStringBuilder spannableStringBuilder = new SpannableStringBuilder("lalalalaalalala"); // - can be hanged  
  
ForegroundColorSpan fcsRed = new ForegroundColorSpan(Color.RED);  
ForegroundColorSpan fcsGreen = new ForegroundColorSpan(Color.GREEN);  
BackgroundColorSpan bgGray = new BackgroundColorSpan(Color.GRAY);  
ClickableSpan clickableSpan = new ClickableSpan() { // clickable objects  
 @Override  
 public void onClick(View widget) {  
 //code  
 Toast.makeText(getApplicationContext(), "aaaaa", Toast.LENGTH\_SHORT).show();  
 }  
  
 @Override  
 public void updateDrawState(TextPaint ds) {  
 super.updateDrawState(ds);  
 ds.setColor(Color.BLUE);  
 ds.setUnderlineText(true);  
 }  
};  
  
spannableString.setSpan(fcsGreen, 2, 5, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);  
spannableString.setSpan(fcsRed, 6, 9, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);  
spannableString.setSpan(bgGray, 5, 15, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);  
spannableString.setSpan(clickableSpan, 14, 23, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);  
  
  
textView.setText(spannableString);  
textView.setMovementMethod(LinkMovementMethod.getInstance()); // needed for clicking

Animate one layout into another

https://www.youtube.com/watch?v=-sPOtGqd5OA