Aplikacje Mobilne – podstawy interfejsu użytkownika

rezultat projektu

Nowe Technologie wsparciem dla edukacji

nr umowy - POWERSE-2018-1-PL01-KA101-049291

realizowanego ze środków POWER na zasadach programu Erasmus+

sektor Edukacja szkolna

„Ponadnarodowa mobilność kadry edukacji szkolnej"

Zespół Szkół Łączności w Gliwicach

44-100 Gliwice, ul. Warszawska 35 tel. 32 231 36 12, **www.zsl.gliwice.pl**

APLIKACJE MOBILNE

PODSTAWY INTERFEJSU UŻYTKOWNIKA

Android Visualizer

- Podstawy interfejsu użytkownika.
- Widoki, od angielskiego „View”.

TextView

Displays text

```
<TextView
    android:id="@+id/title_text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/my_photos"
    android:textAppearance="?android:textAppearanceLarge"
    android:textColor="#4689C8"
    android:textStyle="bold" />
```

My Photos

ImageView

Displays Image

```
<ImageView
    android:id="@+id/photo_image_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scaleType="centerCrop"
    android:src="@drawable/beach" />
```

Layouts | Android Develo

Bezpieczna | https://developer.android.com/guide/topics/ui/declaring-layout.html

Developers DESIGN DEVELOP DISTRIBUTE Search DEVELOPER CONSOLE

API Guides

- Overview
- Layouts
  - Linear Layout
  - Relative Layout
  - Recycler View
  - List View
  - Grid View
- Input Controls
- Input Events
- Menus

Common Layouts

Each subclass of the `ViewGroup` class provides a unique way to display the views you nest within it. Below are some of the more common layout types that are built into the Android platform.

> **Note:** Although you can nest one or more layouts within another layout to acheive your UI design, you should strive to keep your layout hierarchy as shallow as possible. Your layout draws faster if it has fewer nested layouts (a wide view hierarchy is better than a deep view hierarchy).

Linear Layout

Relative Layout

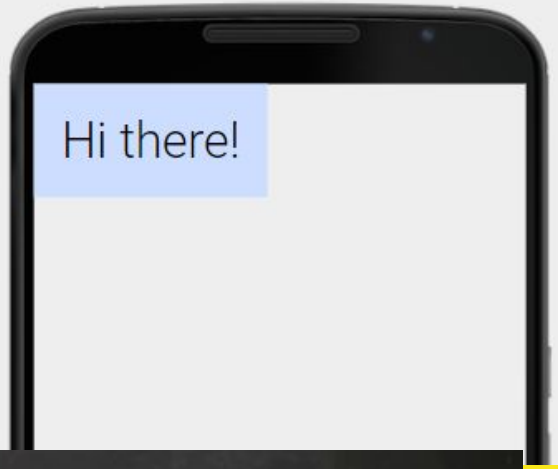
Web View

https://labs.udacity.com/android-visualizer/

XML Layout Code

UNDO REDO Code has been saved RESET CODE

```
1 <TextView
2     android:text="Hi there!"
3     android:layout_width="wrap_content"
4     android:layout_height="wrap_content"
5     android:textSize="36sp"
6     android:fontFamily="sans-serif-light"
7     android:textColor="@android:color/black"
8     android:background="#ccddff"
9     android:padding="20dp"/>
```



- Interfejs użytkownika buduje się w języku XML;
- XML – extensible markup language, czyli rozszerzalny język znaczników;
- XML służy do przechowywania oraz reprezentowania danych;
- Jest czytelny zarówno dla maszyn, jak i dla człowieka;

What is XML?

The Extensible Markup Language (XML) is a simple text-based format for representing structured information: documents, data, configuration, books, transactions, invoices, and much more. It was derived from an older standard format called SGML (ISO 8879), in order to be more suitable for Web use.

What is XML Used For?

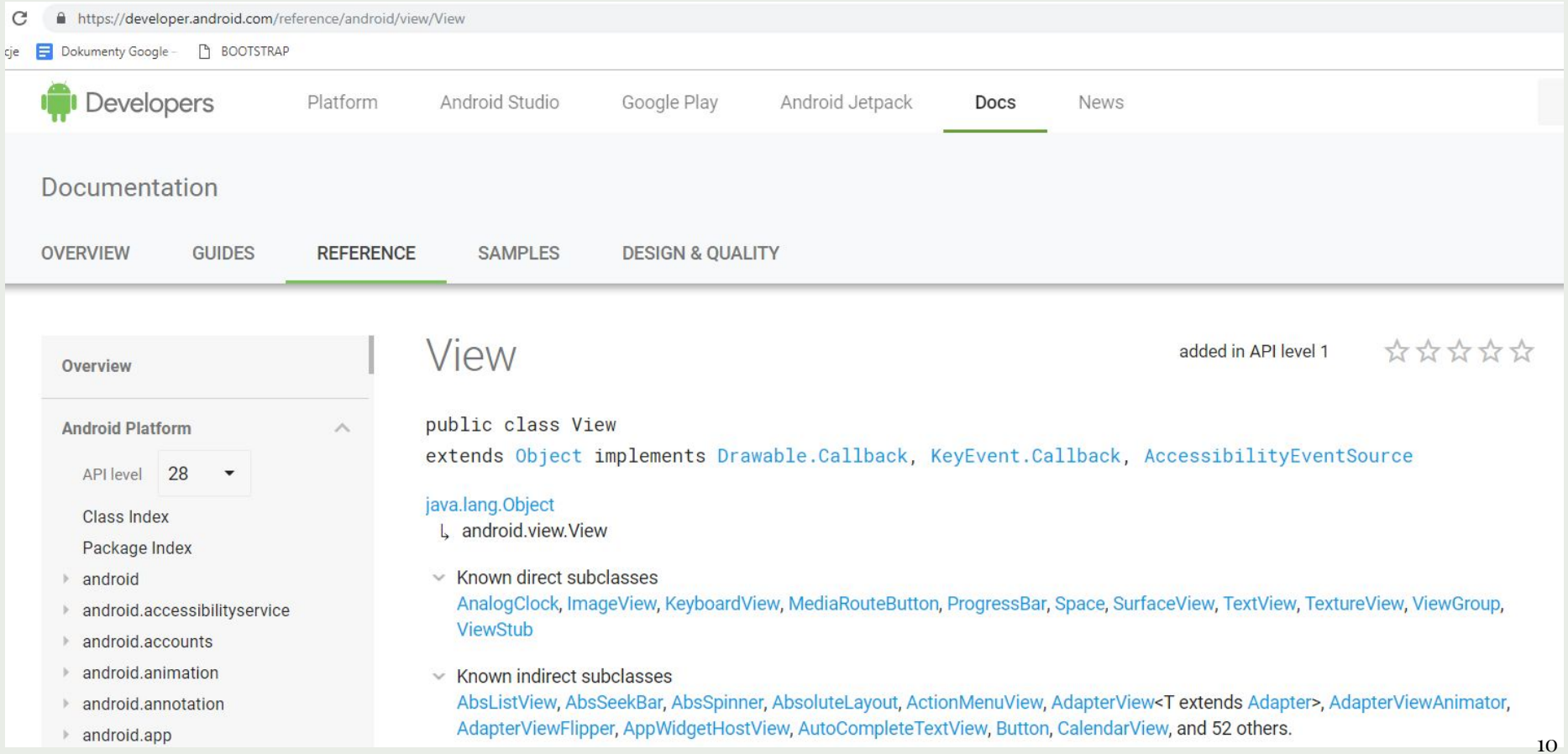
XML is one of the most widely-used formats for sharing structured information today: between programs, between people, between computers and people, both locally and across networks.

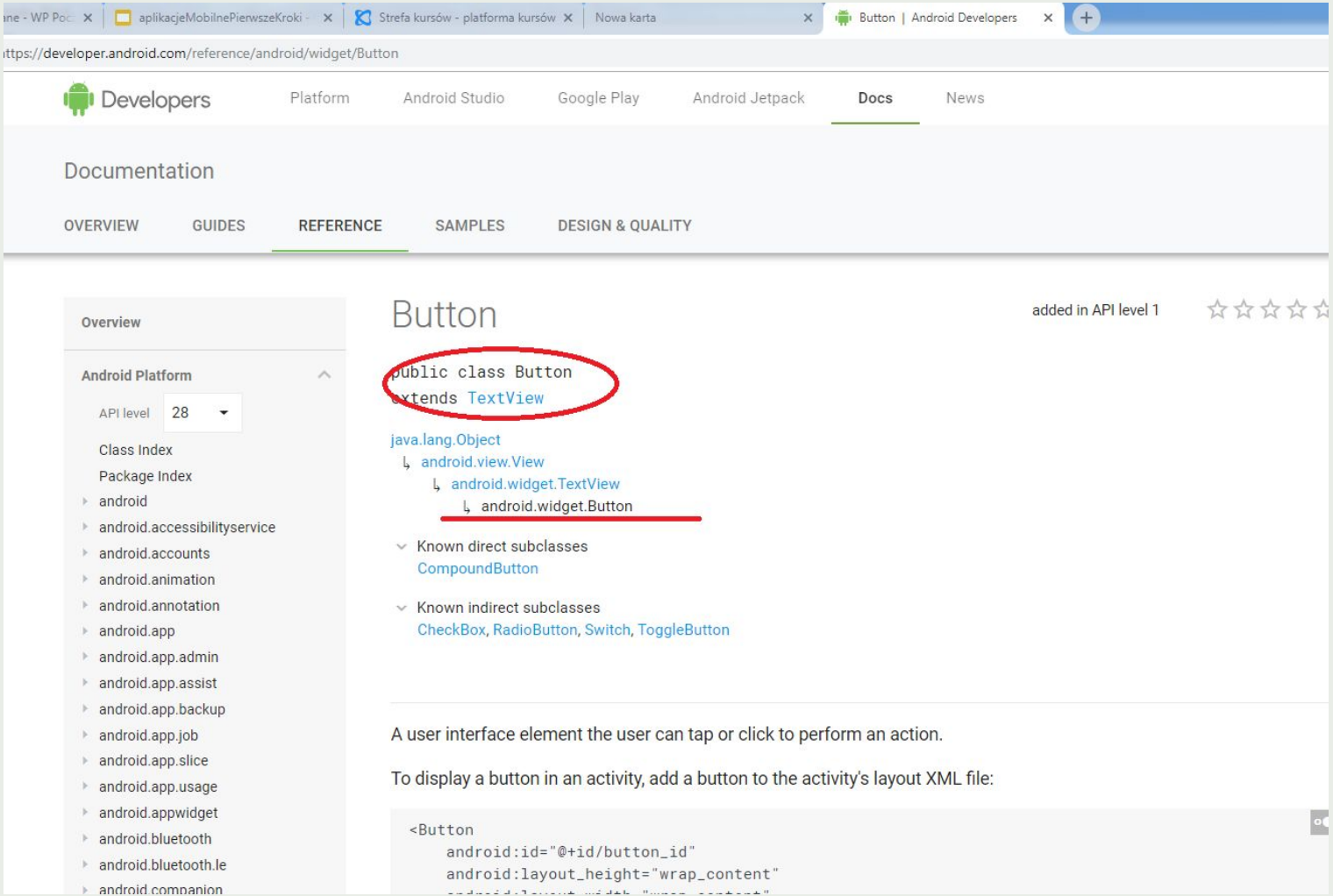
```
<?xml version="1.0" encoding="UTF-8"?>
<katalog_nowych_aut_forda>
	<auto>
		<nazwa_modelu>Fiesta ST</nazwa_modelu>
		<cena_bazowa>75850</cena_bazowa>
		<rok_produkcji>2017</rok_produkcji>
	</auto>

	<auto>
		<nazwa_modelu>Focus ST</nazwa_modelu>
		<cena_bazowa>117400</cena_bazowa>
		<rok_produkcji>2017</rok_produkcji>
	</auto>

	<auto>
		<nazwa_modelu>Mondeo</nazwa_modelu>
		<cena_bazowa>91900</cena_bazowa>
		<rok_produkcji>2017</rok_produkcji>
	</auto>
</katalog_nowych_aut_forda>
```

- Stwórz plik XML w postaci katalogu aut wybranego producenta.
- Uwzględnij maksymalnie trzy auta.
- Uwzględnij takie ich parametry, jak **nazwa modelu**, **cena bazowa** oraz **rok produkcji**.

W XML definiujemy interface użytkownika, ale za jego budowanie i wyświetlanie odpowiedzialna jest JAVA
Celem XML jest przechowywanie danych z naciskiem na ich reprezentację.
HTML ma za zadanie wyświetlanie danych z naciskiem na ich wygląd.

Kurs Android - podstawy tworzenia aplikacji Wszystko jest widokiem




aplikacjeMobilnePierwszeKroki
Strefa kursów - platforma kursów
Nowa karta
Button | Android Developers
https://developer.android.com/reference/android/widget/Button
Developers
Platform
Android Studio
Google Play
Android Jetpack
Docs
News
Documentation
OVERVIEW
GUIDES
REFERENCE
SAMPLES
DESIGN & QUALITY
Overview
Android Platform
API level 28
Class Index
Package Index
android
android.accessibilityservice
android.accounts
android.animation
android.annotation
android.app
android.app.admin
android.app.assist
android.app.backup
android.app.job
android.app.slice
android.app.usage
android.appwidget
android.bluetooth
android.bluetooth.le
android.companion
Button
added in API level 1
public class Button
extends TextView
java.lang.Object
↳ android.view.View
↳ android.widget.TextView
↳ android.widget.Button
Known direct subclasses
CompoundButton
Known indirect subclasses
CheckBox, RadioButton, Switch, ToggleButton
A user interface element the user can tap or click to perform an action.
To display a button in an activity, add a button to the activity's layout XML file:
<Button
android:id="@+id/button_id"
android:layout_height="wrap_content"

Use density-independent pixels

The first pitfall you must avoid is using pixels to define distances or sizes. Defining dimensions with pixels is a problem because different screens have different pixel densities, so the same number of pixels may correspond to different physical sizes on different devices.

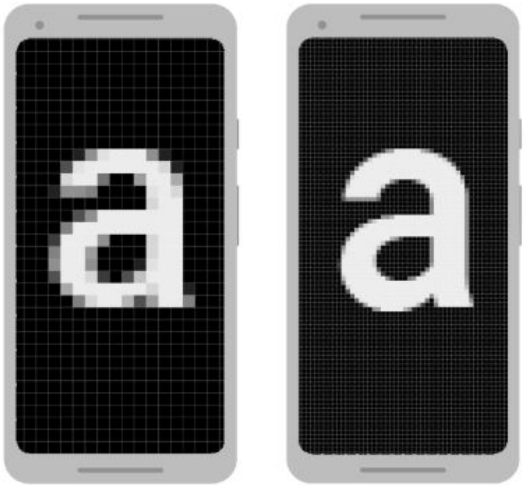


Figure 1. Two screens of the same size may have a different number of pixels

To preserve the visible size of your UI on screens with different densities, you must design your UI using *density-independent pixels* (dp) as your unit of measurement. One dp is a virtual pixel unit that's roughly equal to one pixel on a medium-density screen (160dpi; the "baseline" density). Android translates this value to the appropriate number of real pixels for each other density.

For example, consider the two devices in figure 1. If you were to define a view to be "100px" wide, it will appear much larger on the device on the left. So you must instead use "100dp" to ensure it appears the same size on both screens.

When defining text sizes, however, you should instead use *scalable pixels* (sp) as your units (but never use sp for layout sizes). The sp unit is the same size as dp, by default, but it resizes based on the user's preferred text size.

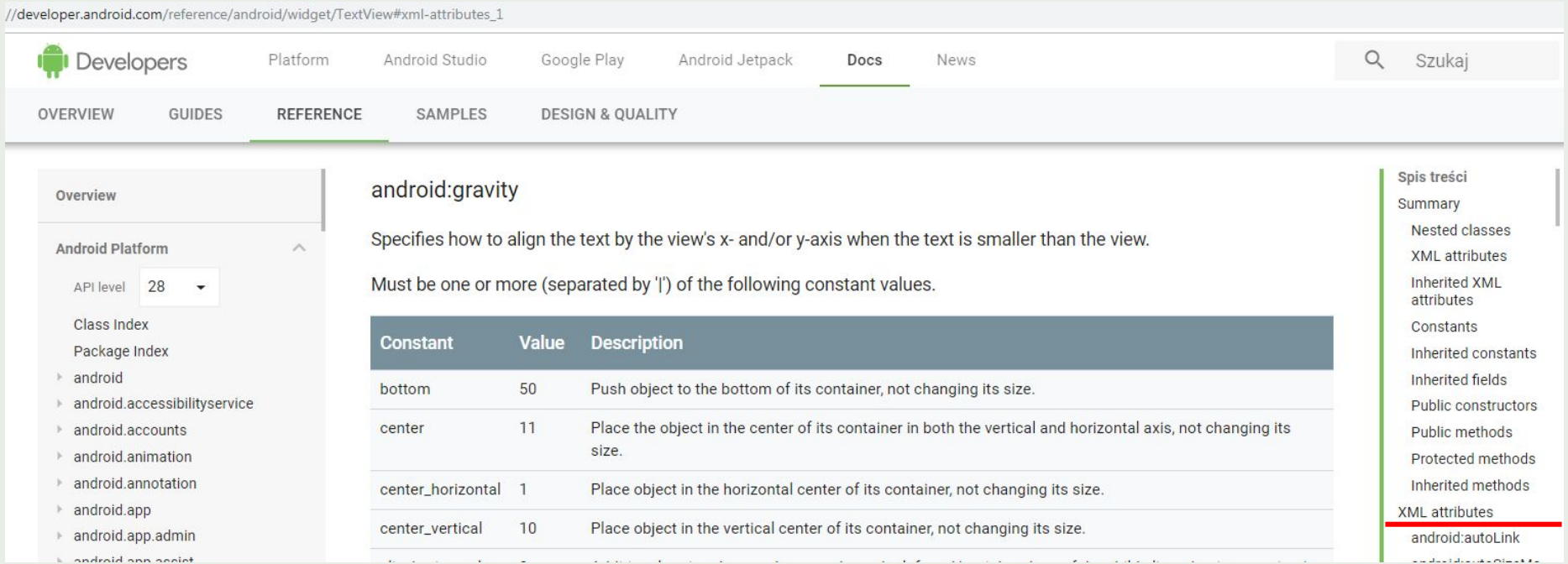
For example, when you specify spacing between two views, use `dp`:

```
<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/clickme"
    android:layout_marginTop="20dp" />
```

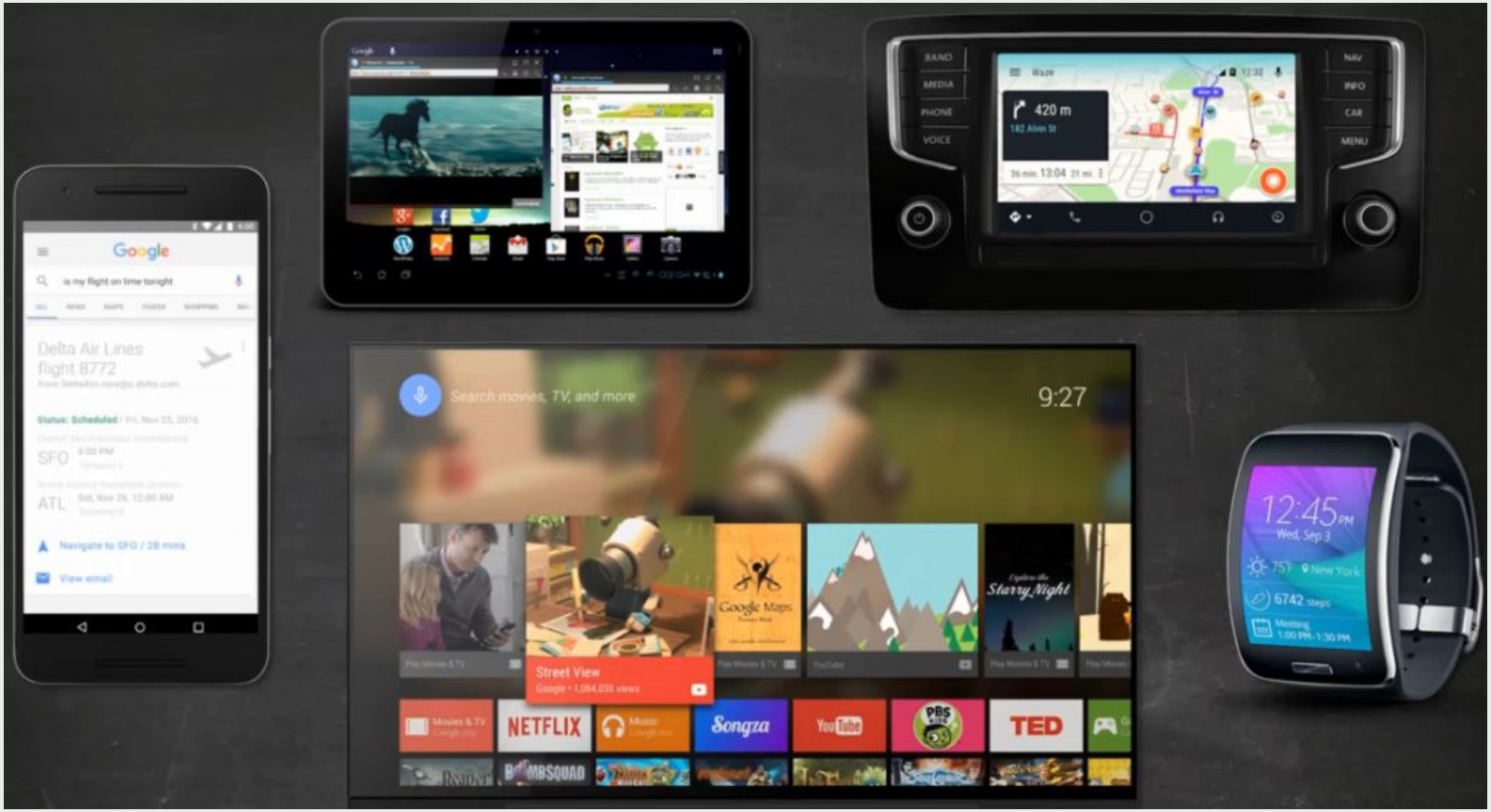
When specifying text size, always use `sp`:

```
<TextView android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="20sp" />
```

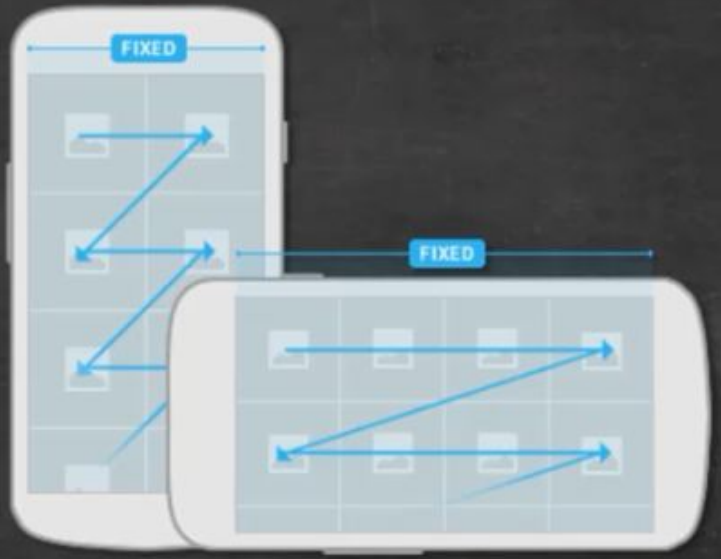
https://developer.android.com/reference/android/widget/TextView#xml-attributes_1

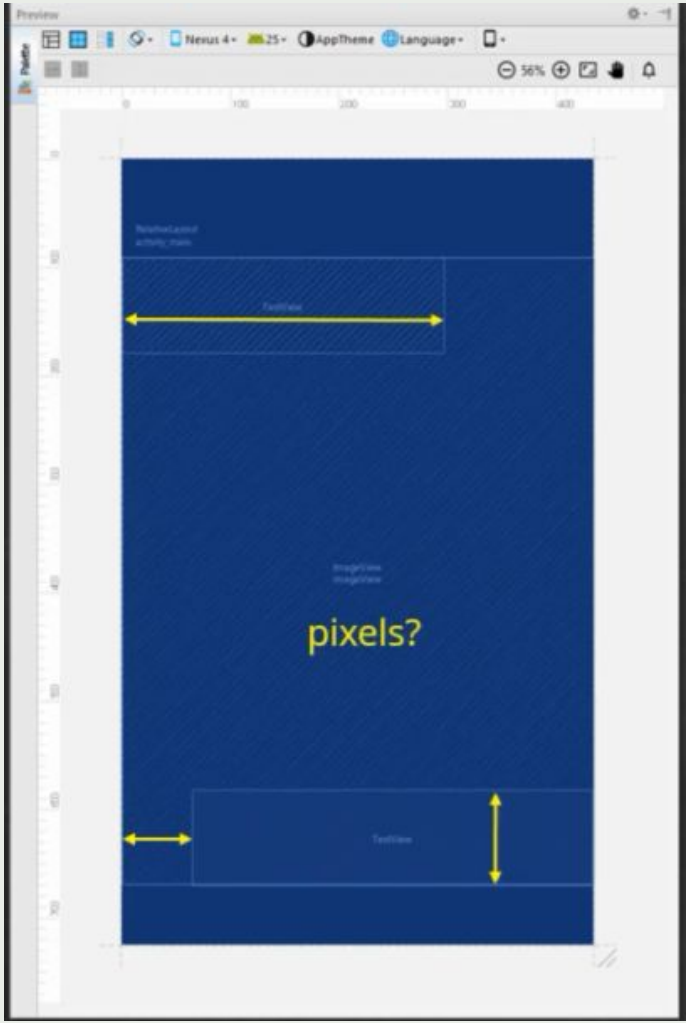


Tworzenie interfejsu dla różnych ekranów



- Tworzenie uniwersalnego interfejsu użytkownika jest trudne (też mi odkrycie).
- Nie tylko z uwagi na rozmaitość urządzeń, ale również na ich dwa tryby wyświetlania zawartości – pionowy oraz poziomy.

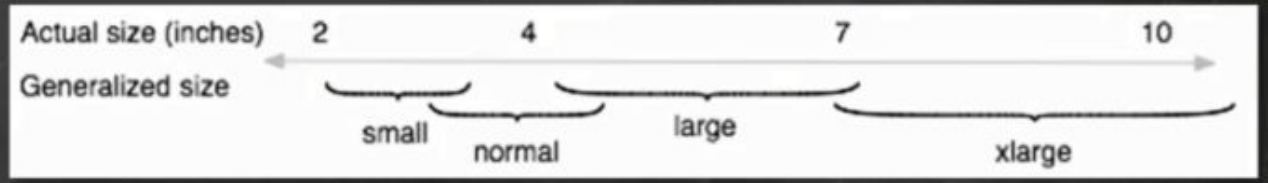




- Rozmiar i położenie elementów wyrażane w pikselach?
- OK, ale jak sobie poradzisz z tworzeniem UI na wiele różnych urządzeń?
- Będziesz tworzyć kilkadziesiąt różnych wersji UI?
- Nawet o tym nie myśl...
- ...bo istnieją normalne sposoby, aby sobie z tym poradzić.

- Wielkość ekranu.
- Rozdzielczość ekranu.
- Gęstość ekranu.

- Wielkość ekranu, to inaczej jego przekątna mierzona w calach (ang. „inches”, skrót „in”).
- Google grupuje dostępne wielkości w cztery grupy: small, normal, large i extra-large.

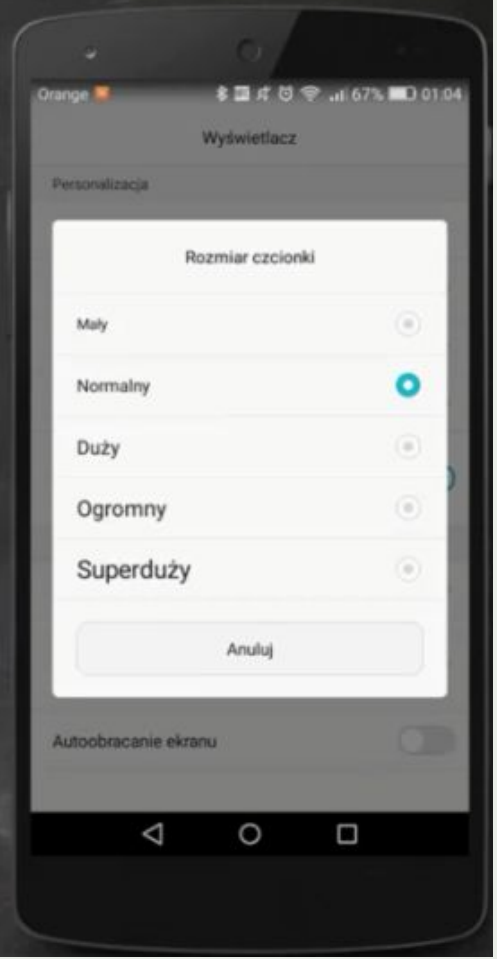


- Rozdzielczość – całkowita ilość wyświetlanych pikseli na ekranie urządzenia na szerokość oraz na wysokość.
- Ile pikseli ma ekran Full HD (1920 na 1080 pikseli)?
- 2 073 600 pikseli (1920*1080).
- Piksel, to najmniejszy, jednolity element wyświetlany na ekranie (ang. „pixel”, skrót „px”).

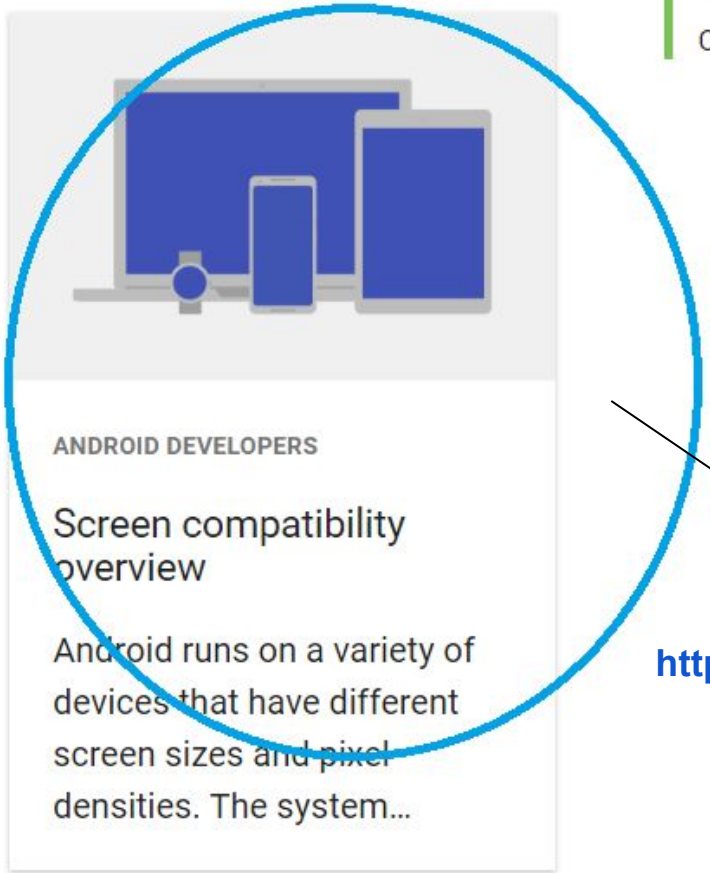
- Gęstość ekranu, to ilość pikseli w określonej przestrzeni ekranu.
- 3 px na 1 in, to 3 dpi (ang. „dots per inch”) lub też 3 ppi (ang. „pixels per inch”).
- Google grupuje dostępne gęstości w sześć grup: low (ldpi ~120 dpi), medium (mdpi ~160 dpi), high (hdpi ~240 dpi), extra-high (xhdpi ~320 dpi), extra-extra-high (xxhdpi ~480 dpi) oraz extra-extra-extra-high (xxxhdpi ~640 dpi).

- Density-independent pixel (skrót „dp”).
- 1dp = 1 px na ekranie 160 dpi (czyli gęstość „medium”, „mdpi”).
- px = dp * (dpi / 160)
- 3 = 1 * (480 / 160)
- Przy tworzeniu UI używaj jednostki „dp”.

- Scale-independent pixel (skrót „sp”).
- Rozmiar czcionki „normalny” -> 14sp -> 14dp...
- ...czyli dp * 1.
- Gdy użytkownik zmieni ustawienia -> dp * współczynnik ustawień. Czyli np. 14 sp -> 14 dp * 1.2.



https://developer.android.com/guide/



https://material.io/tools/devices/

https://developer.android.com/guide/practices/screens_support

Google Material Design - rekomendacje dla wyglądu aplikacji

Element ImageView

ImageView

```
public class ImageView
extends View
```

java.lang.Object
↳ android.view.View
↳ android.widget.ImageView

Known direct subclasses
ImageButton, QuickContactBadge

Known indirect subclasses
ZoomButton

XML attributes	
android:adjustViewBounds	Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
android:baseline	The offset of the baseline within this view.
android:baselineAlignBottom	If true, the image view will be baseline aligned with based on its bottom edge.
android:cropToPadding	If true, the image will be cropped to fit within its padding.
android:maxHeight	An optional argument to supply a maximum height for this view.
android:maxWidth	An optional argument to supply a maximum width for this view.
android:scaleType	Controls how the image should be resized or moved to match the size of this ImageView.
android:src	Sets a drawable as the content of this ImageView.
android:tint	The tinting color for the image.
android:tintMode	Blending mode used to apply the image tint.

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/ic_launcher"
    />
```

Musimy zdefiniować te dwa atrybuty, mimo, że nie ma ich w xml

Wyjaśnienie:

Atrybuty:
`android:layout_width`
`Android:layout_height`
są statycznymi metodami klasy LayoutParams

https://developer.android.com/reference/android/view/ViewGroup.LayoutParams

Klasa ta posiada jedynie dwa atrybuty XML:

XML attributes	
`android:layout_height`	Specifies the basic height of the view.
`android:layout_width`	Specifies the basic width of the view.

Użycie tych atrybutów w jakimkolwiek widoku
Naszego UI mówi rodzicom tych widoków
Jak mają one być wyświetlane: czy mają mieć specyficzne wysokości i szerokości, czy mają wypełniać całą, dostępną wolną przestrzeń, czy może powinny ograniczać swoje rozmiary tylko do swojej zawartości. Użycie tych atrybutów jest kluczowe.

Wartości dla atrybutów:

`android:layout_width`
`Android:layout_height`

Constants	
int	FILL_PARENT Special value for the height or width requested by a View.
int	MATCH_PARENT Special value for the height or width requested by a View.
int	WRAP_CONTENT Special value for the height or width requested by a View.

przestarzałe

Zamiast FILL_PARENT

Special value for the height or width requested by a View. WRAP_CONTENT means that the view wants to be just large enough to fit its own internal content, taking its own padding into account.

https://developer.android.com/guide/topics/resources/providing-resources

Grouping resource types

You should place each type of resource in a specific subdirectory of your project's `res/` directory. For example, here's the file hierarchy for a simple project:

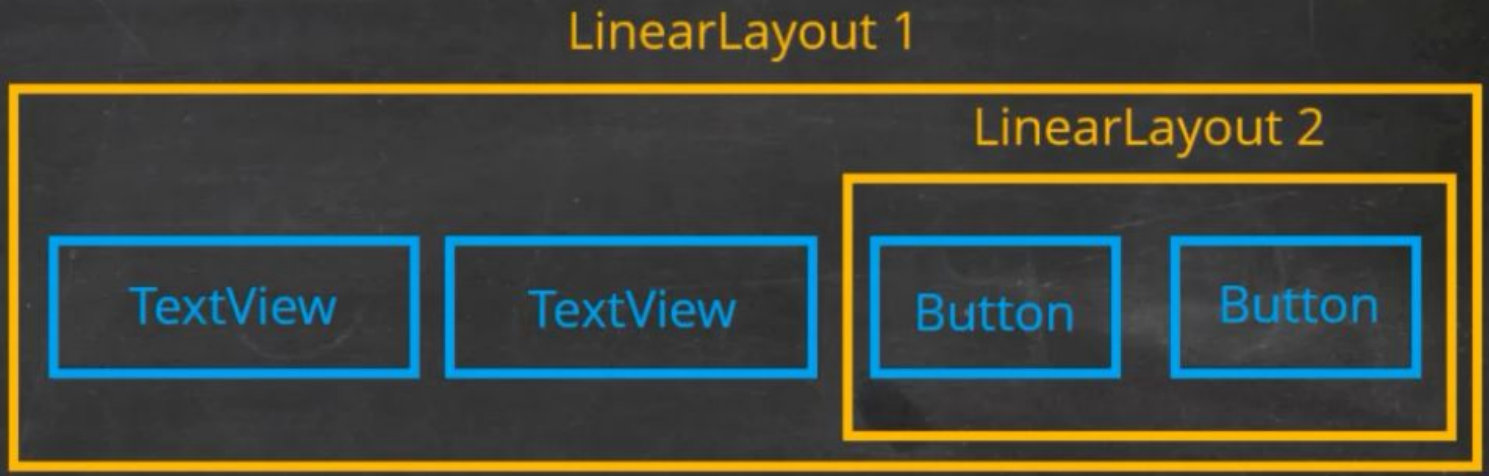
```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            graphic.png
        layout/
            main.xml
            info.xml
        mipmap/
            icon.png
        values/
            strings.xml
```

As you can see in this example, the `res/` directory contains all the resources (in subdirectories): an image resource, two layout resources, `mipmap/` directories for launcher icons, and a string resource file. The resource directory names are important and are described in table 1.

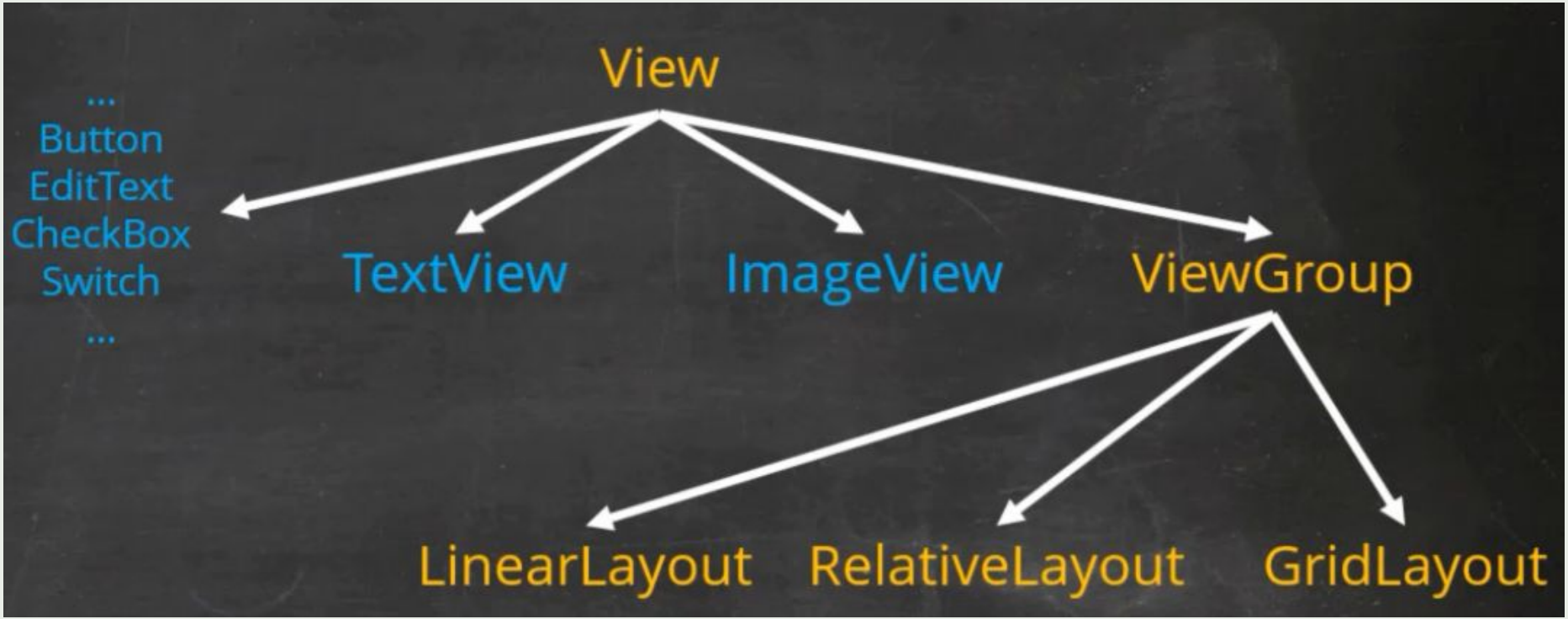
https://labs.udacity.com/android-visualizer/

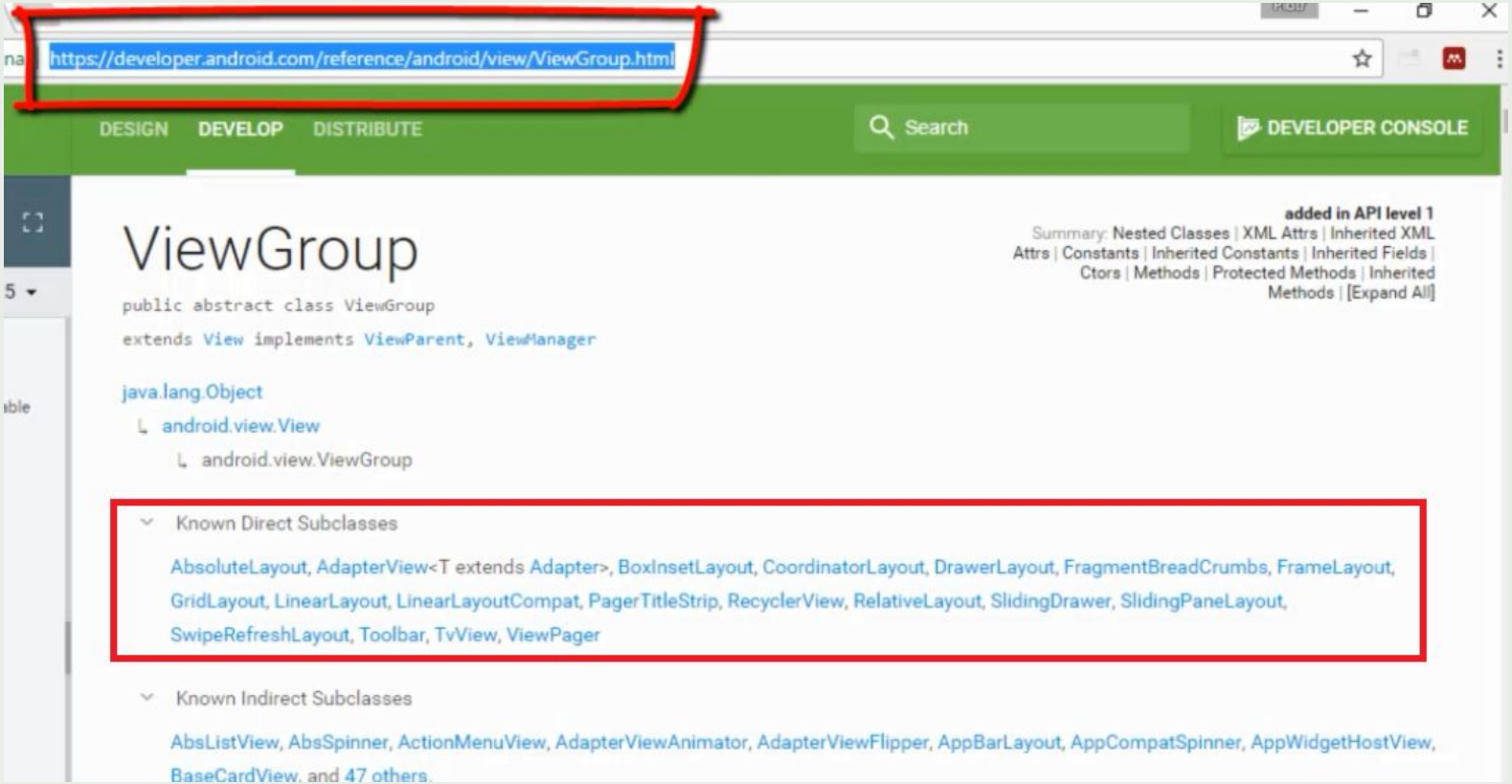
ViewGroup - sposób na wyświetlanie więcej, niż jednego elementu na ekranie

- ViewGroup, to klasa abstrakcyjna. Nie możemy utworzyć obiektu tej klasy.
- Klasy LinearLayout, RelativeLayout oraz GridLayout dziedziczą po klasie ViewGroup, ale są już „zwykłymi" klasami, więc dla nich MOŻEMY utworzyć obiekt. Przez to mogą być już kontenerami dla innych widoków.
- Rozkład może być kontenerem dla innego rozkładu.



ViewGroup - sposób na wyświetlanie więcej, niż jednego elementu na ekranie

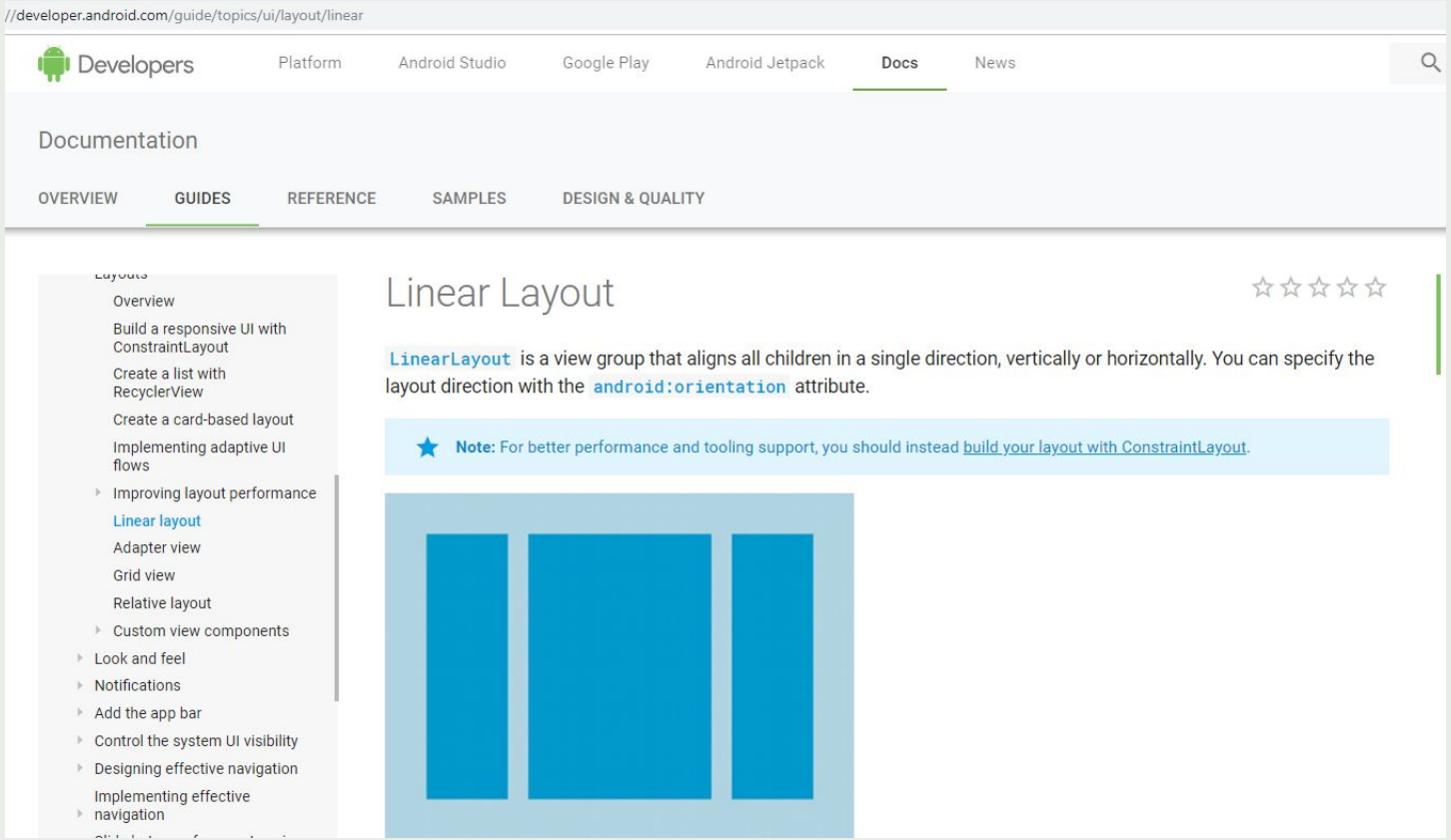



https://developer.android.com/reference/android/view/ViewGroup.html
DESIGN
DEVELOP
DISTRIBUTE
Search
DEVELOPER CONSOLE
ViewGroup
public abstract class ViewGroup
extends View implements ViewParent, ViewManager
added in API level 1
Summary: Nested Classes | XML Attrs | Inherited XML Attrs | Constants | Inherited Constants | Inherited Fields | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]
java.lang.Object
android.view.View
android.view.ViewGroup
Known Direct Subclasses
AbsoluteLayout, AdapterView<T extends Adapter>, BoxInsetLayout, CoordinatorLayout, DrawerLayout, FragmentBreadCrumbs, FrameLayout, GridLayout, LinearLayout, LinearLayoutCompat, PagerTitleStrip, RecyclerView, RelativeLayout, SlidingDrawer, SlidingPaneLayout, SwipeRefreshLayout, Toolbar, TvView, ViewPager
Known Indirect Subclasses
AbsListView, AbsSpinner, ActionMenuView, AdapterViewAnimator, AdapterViewFlipper, AppBarLayout, AppCompatSpinner, AppWidgetHostView, BaseCardView, and 47 others.

- ViewGroup, to klasa abstrakcyjna, która stanowi bazę dziedziczenia dla wszystkich rozkładów (ang. „layouts”) w Androidzie.
- Rozkład, to widok, który jest kontenerem dla innych widoków.
- Przykładowe rozkłady: LinearLayout, RelativeLayout, GridLayout.

Rozkład LinearLayout (widoki mogą być ułożone poziomo lub pionowo)

https://developer.android.com/guide/topics/ui/layout/linear



Layout Weight

`LinearLayout` also supports assigning a *weight* to individual children with the `android:layout_weight` attribute. This attribute assigns an "importance" value to a view in terms of how much space it should occupy on the screen. A larger weight value allows it to expand to fill any remaining space in the parent view. Child views can specify a weight value, and then any remaining space in the view group is assigned to children in the proportion of their declared weight. Default weight is zero.

Equal distribution

To create a linear layout in which each child uses the same amount of space on the screen, set the `android:layout_height` of each view to `"0dp"` (for a vertical layout) or the `android:layout_width` of each view to `"0dp"` (for a horizontal layout). Then set the `android:layout_weight` of each view to `"1"`.

Unequal distribution

You can also create linear layouts where the child elements use different amounts of space on the screen:

- If there are three text fields and two of them declare a weight of 1, while the other is given no weight, the third text field without weight doesn't grow. Instead, this third text field occupies only the area required by its content. The other two text fields, on the other hand, expand equally to fill the space remaining after all three fields are measured.
- If there are three text fields and two of them declare a weight of 1, while the third field is then given a weight of 2 (instead of 0), then it's now declared more important than both the others, so it gets half the total remaining space, while the first two share the rest equally.

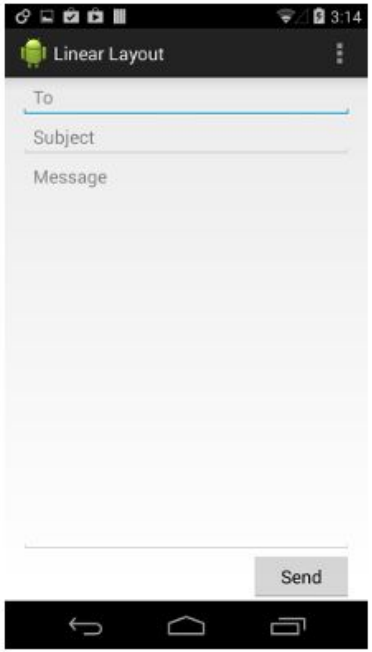
Określenie przestrzeni nazw

Przestrzeń nazw powinna być określana na początku nadrzędnego widoku interfejsu

Poza przestrzenią nazw android mamy również inne, np:
- xmlns: apps
- xmlns: tools

The following code snippet shows how layout weights might work in a "send message" activity. The **To** field, **Subject** line, and **Send** button each take up only the height they need. This configuration allows the message itself to take up the rest of the activity's height.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/androi
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



https://developer.android.com/guide/topics/ui/declaring-layout -obowiązkowa lektura

Zadanie: Wykorzystując poznane elementy należy odtworzyć Przedstawiony układ (kolory, proporcje itp)

Rozkład RelativeLayout:
https://developer.android.com/guide/topics/ui/layout/relative

Relative Layout

☆☆☆☆☆

Spis treści
Positioning Views
Example

`RelativeLayout` is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent `RelativeLayout` area (such as aligned to the bottom, left or center).

Note: For better performance and tooling support, you should instead build your layout with ConstraintLayout.

A `RelativeLayout` is a very powerful utility for designing a user interface because it can eliminate nested view groups and keep your layout hierarchy flat, which improves performance. If you find yourself using several nested `LinearLayout` groups, you may be able to replace them with a single `RelativeLayout`.

Positioning Views

`RelativeLayout` lets child views specify their position relative to the parent view or to each other (specified by ID). So you can align two elements by right border, or make one below another, centered in the screen, centered left, and so on. By default, all child views are drawn at the top-left of the layout, so you must define the position of each view using the various layout properties available from `RelativeLayout.LayoutParams`.

https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams

XML attributes	
`android:layout_above`	Positions the bottom edge of this view above the given anchor view ID.
`android:layout_alignBaseline`	Positions the baseline of this view on the baseline of the given anchor view ID.
`android:layout_alignBottom`	Makes the bottom edge of this view match the bottom edge of the given anchor view ID.
`android:layout_alignEnd`	Makes the end edge of this view match the end edge of the given anchor view ID.
`android:layout_alignLeft`	Makes the left edge of this view match the left edge of the given anchor view ID.

android:layout_alignParentBottom	If true, makes the bottom edge of this view match the bottom edge of the parent.
android:layout_alignParentEnd	If true, makes the end edge of this view match the end edge of the parent.
android:layout_alignParentLeft	If true, makes the left edge of this view match the left edge of the parent.
android:layout_alignParentRight	If true, makes the right edge of this view match the right edge of the parent.
android:layout_alignParentStart	If true, makes the start edge of this view match the start edge of the parent.
android:layout_alignParentTop	If true, makes the top edge of this view match the top edge of the parent.

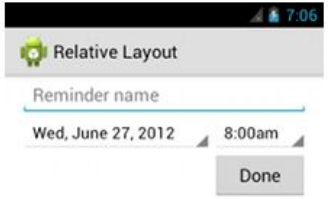
android:layout_centerHorizontal	If true, centers this child horizontally within its parent.
android:layout_centerInParent	If true, centers this child horizontally and vertically within its parent.
android:layout_centerVertical	If true, centers this child vertically within its parent.

https://developer.android.com/guide/topics/ui/layout/relative

Uwaga: zamiast zagnieżdżać układy, lepiej użyć RelativeLayout
Zagnieżdżane rozkłady są zdecydowanie bardziej zasobożerne niż pojedyncze rozkłady

Each of the attributes that control the relative position of each view are emphasized.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/andr
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



Zadanie:
Zadanie: Wykorzystując poznane elementy należy odtworzyć
Przedstawiony układ (kolory, proporcje itp)

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:src="@drawable/rocks"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="20dp"
        android:gravity="center"
        android:text="Photos from Android National Park"
        android:textSize="20sp"
        android:textStyle="bold"
        android:textColor="#000"
        android:background="#FFF" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="BACK"
        android:textSize="20sp"
        android:layout_alignParentBottom="true"
        android:background="#FFF" />
```

```

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="NEXT"
        android:textSize="20sp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:background="#FFF" />

</RelativeLayout>
```