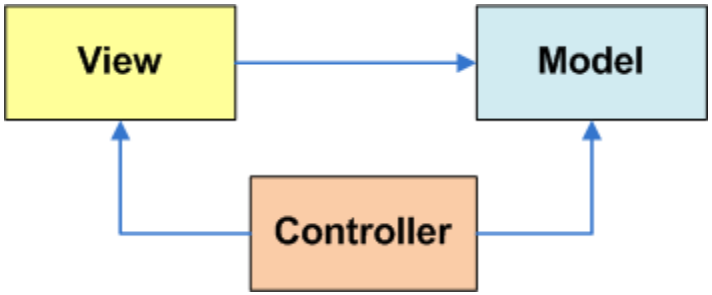
Pracownia aplikacji mobilnych

Graficzny interfejs użytkownika

Wzorzec MVC

Wzorzec Model-Widok-Kontroler *(MVC)* jest istotnym wzorcem projektowym, którego głównym zadaniem jest odseparowanie (1) interfejsu użytkownika, (2) biznesowej oraz (3) operacyjnej logiki.



Jak to wygląda z punktu widzenia programisty Android?

- **Model**. Składa się z kodu w języku JAVA i odwołań API, które zarządzają zachowaniem danych aplikacji.
- **Widok**. Zestaw ekranów, które użytkownik widzi i wchodzi w interakcję.
- **Kontroler**. Implementowany m. in. poprzez system operacyjny, odpowiedzialny za interpretację zdarzeń użytkownika i systemowych. Zdarzenia mogą pochodzić z różnorakich źródeł: trackball, klawiatura, ekran dotykowy, moduł GPS, usługi działające w tle. Nakazuje modelowi i/lub widokowi (zwykle przez wywołania zwrotne i nasłuchiwaczy) by dokonały odpowiednich zmian.

Wzorzec MVC c. d.

Graficzne interfejsy użytkownika zwykle implementowane są jako pliki XML (aczkolwiek mogą one być tworzone również w sposób dynamiczny poprzed kod języka Java).

Zachowanie kontrolera:

- Bezpośrednia interakcja

  Gdy użytkownik dotknie określonego miejsca na ekranie, kontroler dokonuje interpretacji tego zdarzenia i określa, jaki dokładnie fragment ekranu oraz gest miał miejsce. Na podstawie tej informacji, przekazywane jest do modelu wywołanie określonej funkcji (callback) lub konieczność zmiany stanu aplikacji.

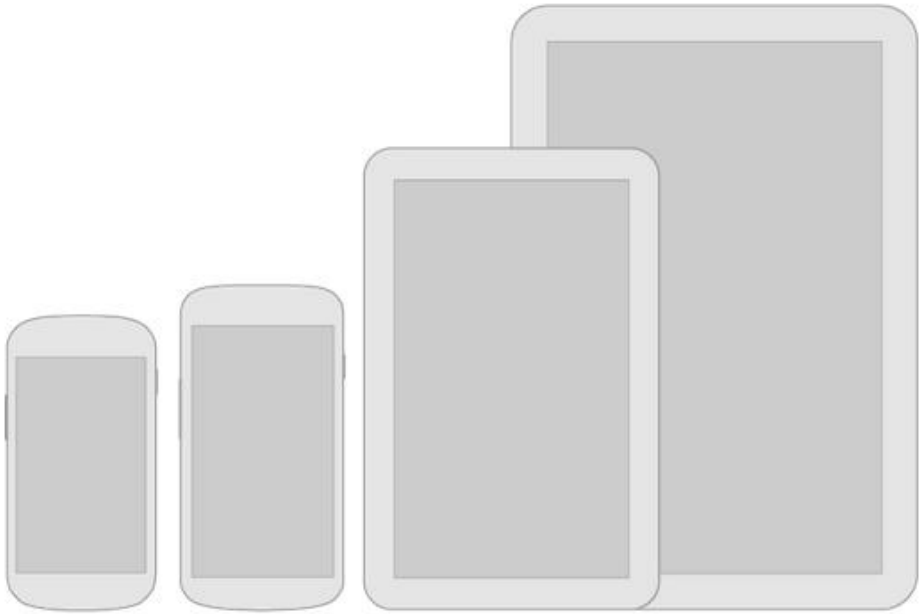
- Niejawna interakcja

  Usługa działająca w tle, może bez ingerowania użytkownika powiadomić kontroler odnośnie zmian stanu (np. osiągnięto miejsce docelowe w nawigacji), co w konsekwencji może spowodować zmianę widoku.

Wzorce projektowe GUI

Devices and Displays

Android powers more than a billion phones, tablets, and other devices in a wide variety of screen sizes and form factors. By taking advantage of Android's flexible layout system, you can create apps that gracefully scale from large tablets to smaller phones.



Be flexible

Stretch and compress your layouts to accommodate various heights and widths.

Optimize layouts

On larger devices, take advantage of extra screen real estate. Create compound views that combine multiple views to reveal more content and ease navigation.

Assets for all

Provide resources for different screen densities (DPI) to ensure that your app looks great on any device.

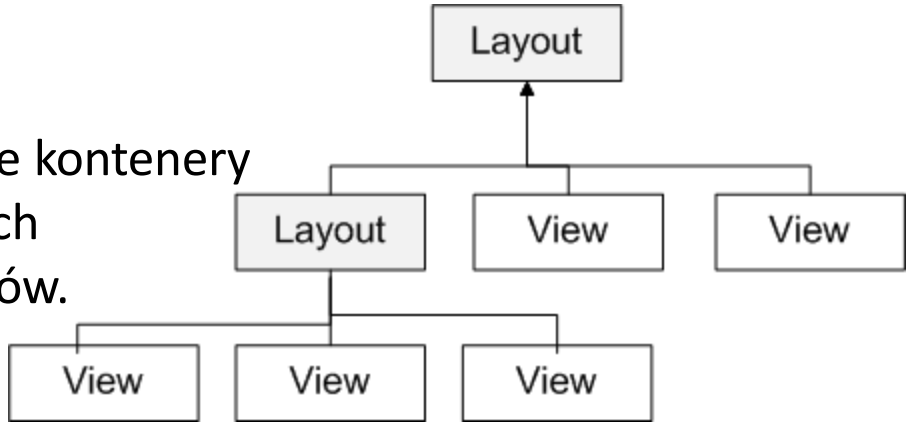
Klasa View

- Klasa **View reprezentuje podstawowy komponent Androida** za pomocą którego można tworzyć graficzne interfejsy użytkownika. Stanowi kontener dla wszystkich elementów, które można wyświelić.

- **Widok** zajmuje prostokątną przestrzeń na ekranie i jest odpowiedzialny za *rysowanie* jak również *przetwarzanie zdarzeń*.

- **Widżety (ang. widgets)** są potomkami klasy View. Używane są do stworzenia interaktywnych komponentów graficznych takich jak przyciski, etykiety, pola tekstowe itp.

- **Układy (ang. layouts)** to niewidzialne kontenery umożliwiające pozycjonowanie innych widoków oraz zagnieżdżonych układów.



Interfejs GUI ↔ Układ XML

Rzeczywisty interfejs aplikacji

Wersja tekstowa: *activity_main.xml* →

```
<RelativeLayout
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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="35dp"
        android:layout_marginTop="35dp"
        android:ems="10"
        android:hint="Enter your NAME here" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText1"
        android:layout_below="@+id/editText1"
        android:layout_marginLeft="54dp"
        android:layout_marginTop="26dp"
        android:text="Go" />

</RelativeLayout>
```

Wykorzystanie Widoków

- Plik **XML** z widokiem składa się z układem (**layout)** tworzącym hierarchiczną strukturę zawartych w nim elementów.
- Wewnętrzne elementy mogą być zwykłymi widżetami bądź zagnieżdżonymi widokami zawierające skomplikowane hierarchie elementów.
- Aktywność używa `setContentView(R.layout.xmlfilename)` by wyświetlić widok na ekranie urządzenia.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    Widżety i zagnieżdżone układy

</LinearLayout>
```

Wykorzystanie widoków

Wykorzystanie widoków i układów z reguły składa się z następującego szeregu czynności:

1. **Ustaw właściwości:** Przykładowo kolor tła, tekstu, czcionki i wielkości komponentów.

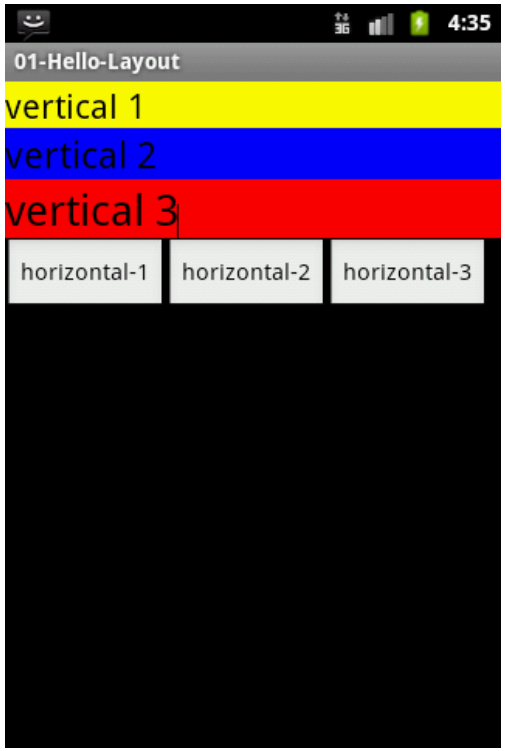
2. **Ustaw nasłuchiwacze (ang. listeners):** Przykładowo obraz może być skonfigurowany by reagował na zdarzenia kliknięcia, dłuższego przytrzymania palca itp.

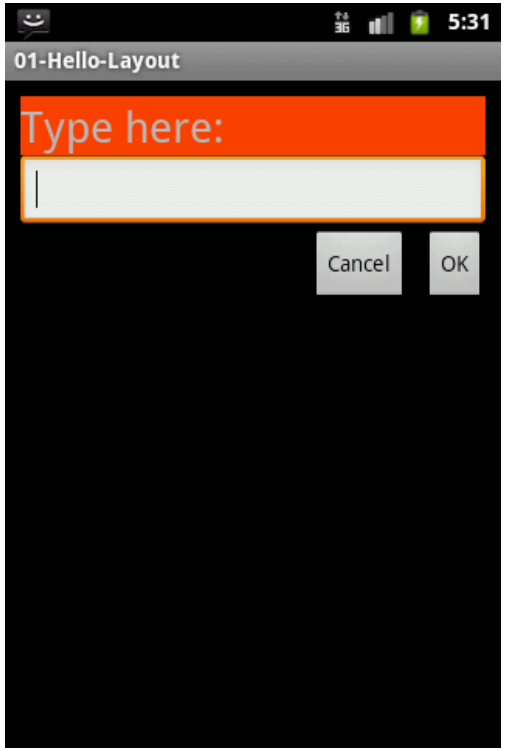
3. **Ustaw focus:** By ustawić focus na określonym komponencie należy użyć metody requestFocus() lub znacznika XML <requestFocus/>

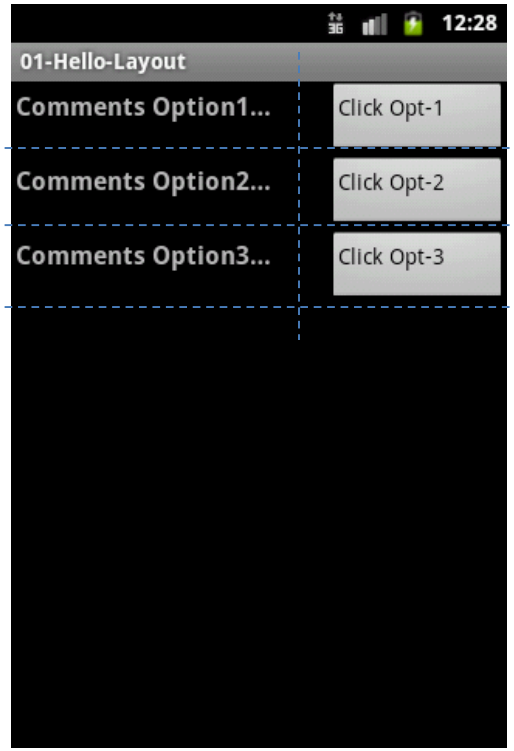
4. **Ustaw widoczność:** Można pokazywać lub ukrywać elementy wykorzystując metodę setVisibility(...).

Przykładowe komponenty GUI

Układy







Linear Layout
LinearLayout rozmieszcza widoki horyzontalnie lub wertykalnie.

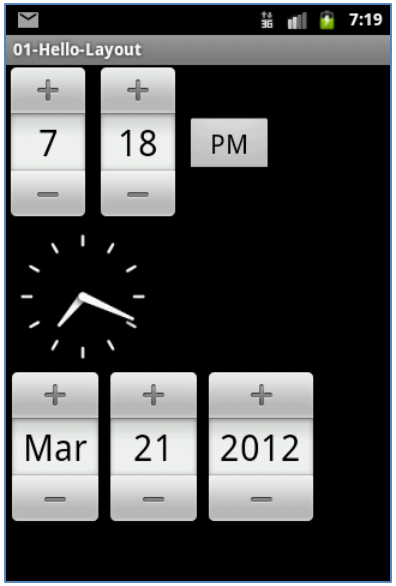
Relative Layout
RelativeLayout jest grupą elementów, która umożliwia rozmieszczenie widoków w sposób względny.

Table Layout
TableLayout jest grupą elementów, która rozmieszcza elementy w wierszu bądź kolumnie wirtualnej tabeli.

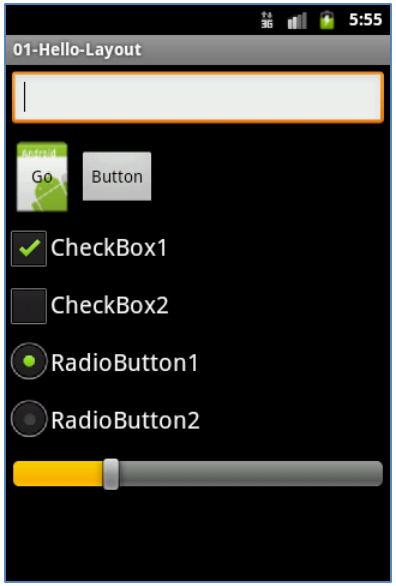
Reference: http://developer.android.com/guide/topics/ui/layout-objects.html

Przykładowe komponenty GUI

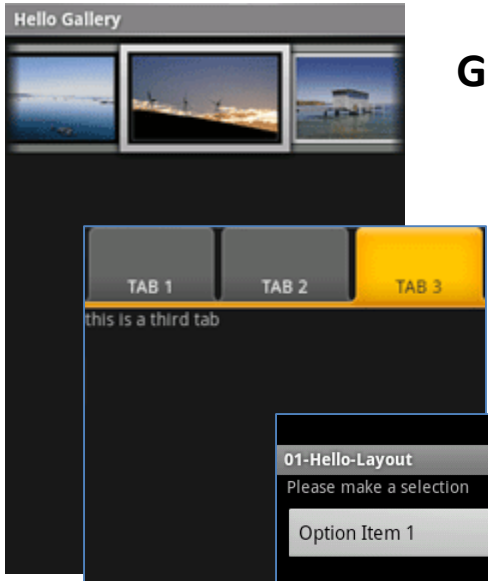
Widżety



TimePicker
AnalogClock
DatePicker
DatePicker jest komponentem umożliwiającym wybór miesiąca, dnia i roku.



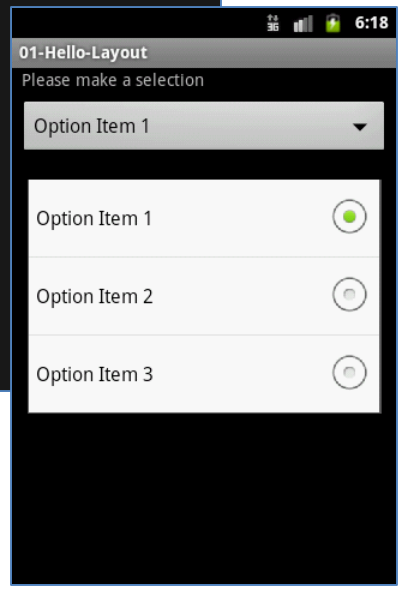
Kontrolki formularza
Grupa komponentów do wpisania danych: *przyciski z grafiką, pola jednokrotnego i wielokrotnego wyboru itp.*



GalleryView

TabWidget

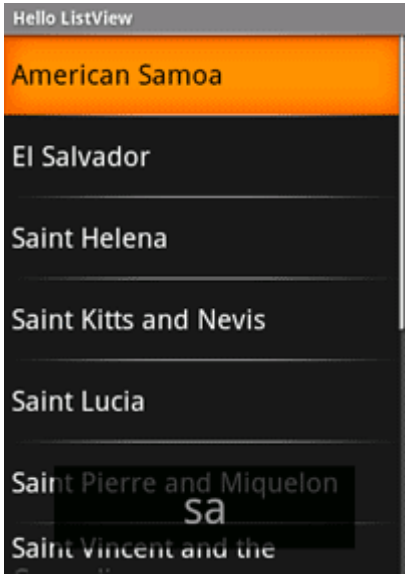
Spinner



Reference: http://developer.android.com/guide/topics/ui/layout-objects.html

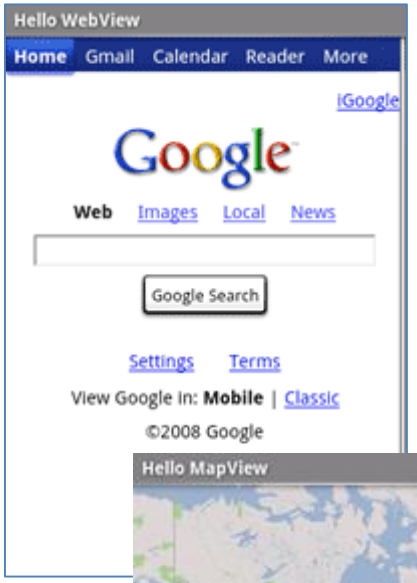
Przykładowe komponenty GUI

AutoCompleteTextView
Jest specyficzną wersją pola tekstowego (*EditText), który pokazuje sugestie użytkownikowi podczas pisania*. Sugestie pochodzą z kolekcji typu tablica ciągów.



ListView
ListView jest widokiem który pokazuje dane w formie pionowej, przewijalnej listy. Dane pochodząc z obiektu typu *ListAdapter*.

WebView

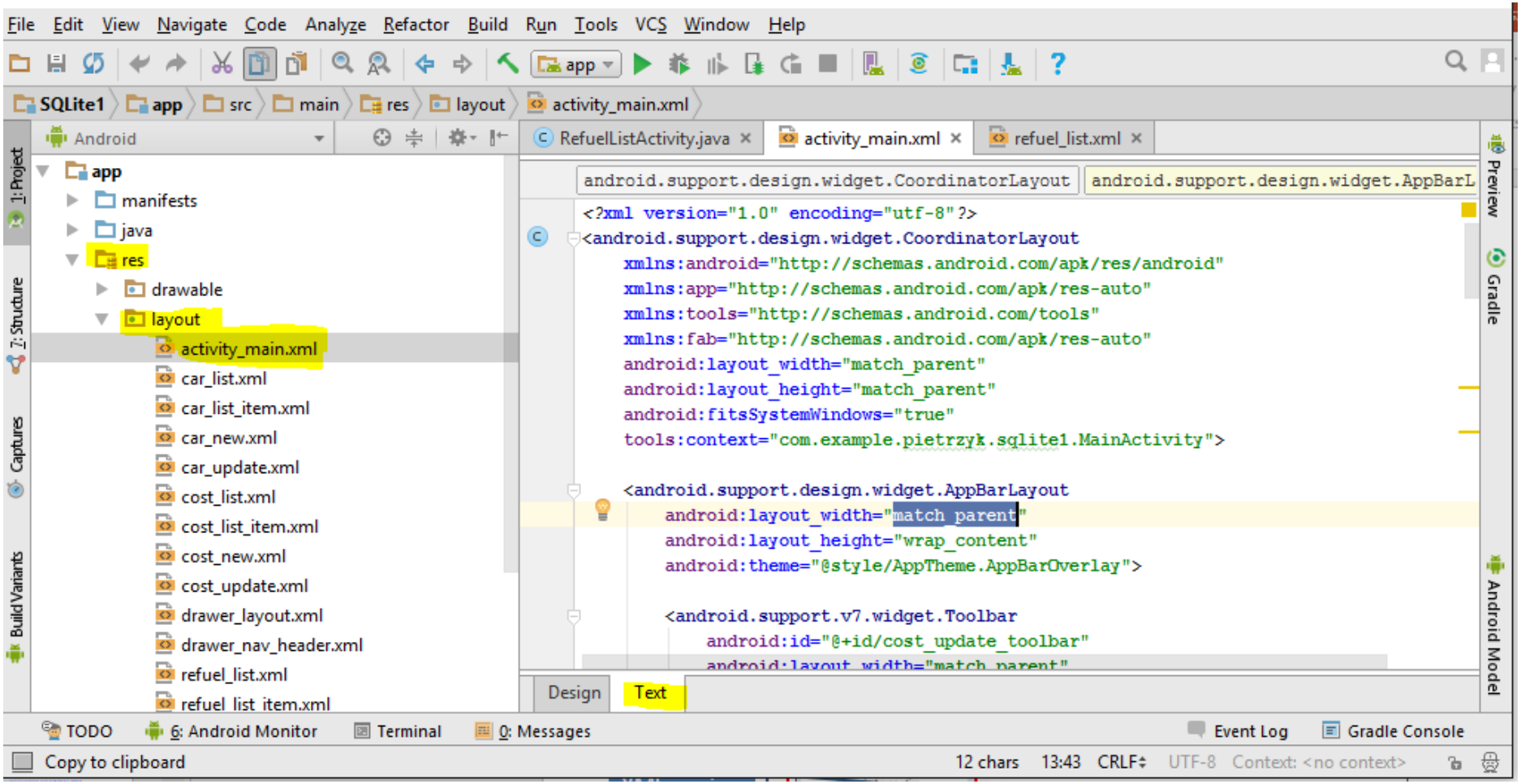


MapView

Reference: http://developer.android.com/guide/topics/ui/layout-objects.html

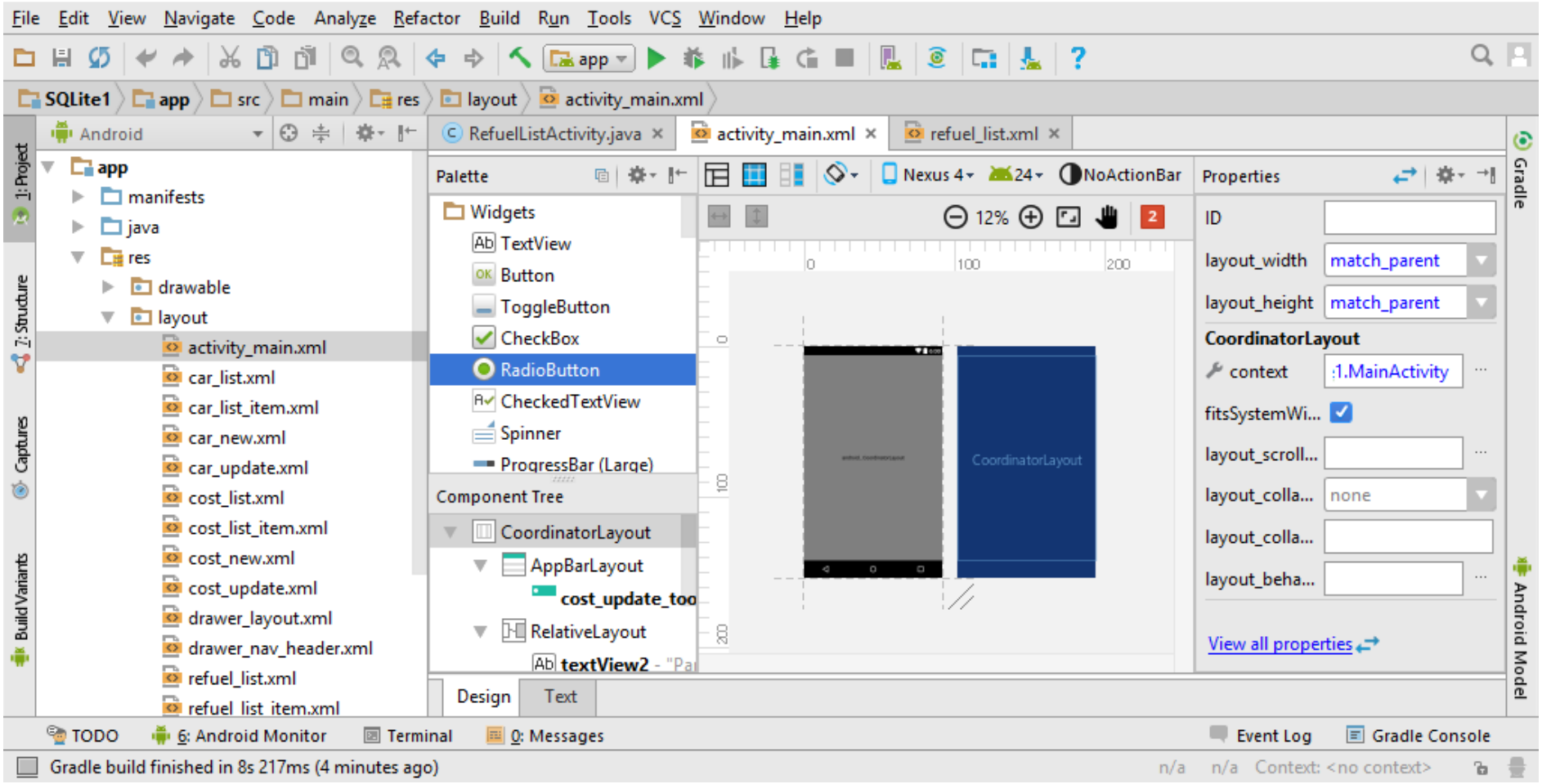
Układy XML w Android Studio

W kontekście platformy Android, pliki XML wchodzą w skład **zasobów (ang. *resources)***, więc pliki z interfejsem graficznym znajdują się w katalogu **res/layout** projektu.



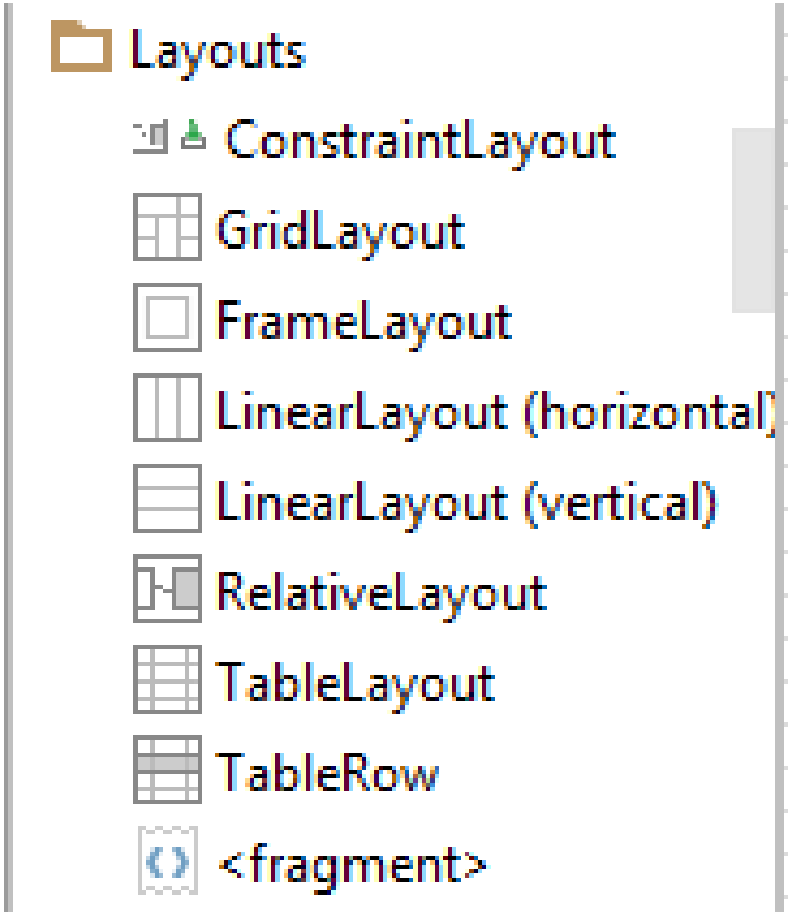
Układy XML w Android Studio

Android Studio zawiera również edytor typu WYSIWIG, który umożliwia tworzenie interfejsów metodą przeciągij-i-upuść.



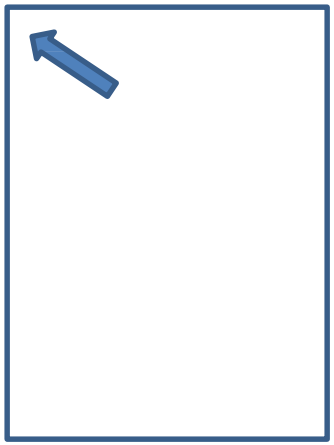
Jak tworzyć GUI w Androidzie

- Układy w Androidzie są kontenerami GUI, posiadające ściśle zdefiniowaną strukturę oraz właściwości dotyczące rozłożenia elementów.
- **Układy mogą być zagnieżdżane**, dlatego komórka, wiersza albo kolumna może być początkiem dla innego widoku bądź układu.
- W Android Studio dostępne są następujące typy układów (co ewoluuje z każdą wersją Androida):



Podstawowe układy

FrameLayout

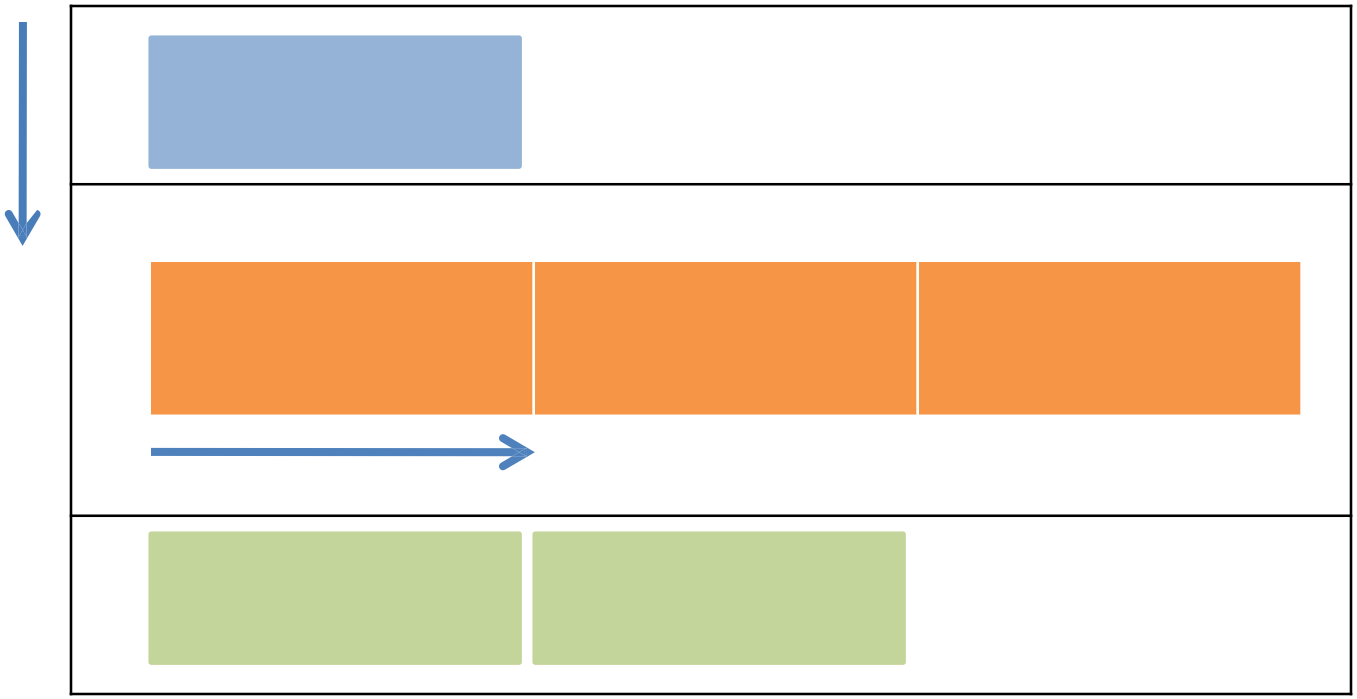


- FrameLayout jest najprostszym typem układu.
- Najczęściej wykorzystywany jest jako najbardziej zewnętrzny układ.
- Umożliwia zdefiniowanie jak duża część ekranu (wysokość, szerokość) ma zostać wykorzystana.
- Wszystkie jego elementy potomne pozycjonowane są względem górnego lewego końca ekranu.

Linear Layout

1. Linear Layout

- **LinearLayout** wspiera strategię wypełnienia według której elementy są ułożone w sposób horyzontalny lub wertykalny.
- Jeżeli orientacja układu ustawiona jest na pionową nowe wiersze (widoki) układane są jeden na drugim.
- Orientacja horyzontalna wykorzystuje rozmieszczenie jeden obok drugiego.



Linear Layout

1. LinearLayout: Ustawianie właściwości

Konfiguracja **LinearLayout** sprowadza się do ustawienia następujących właściwości:

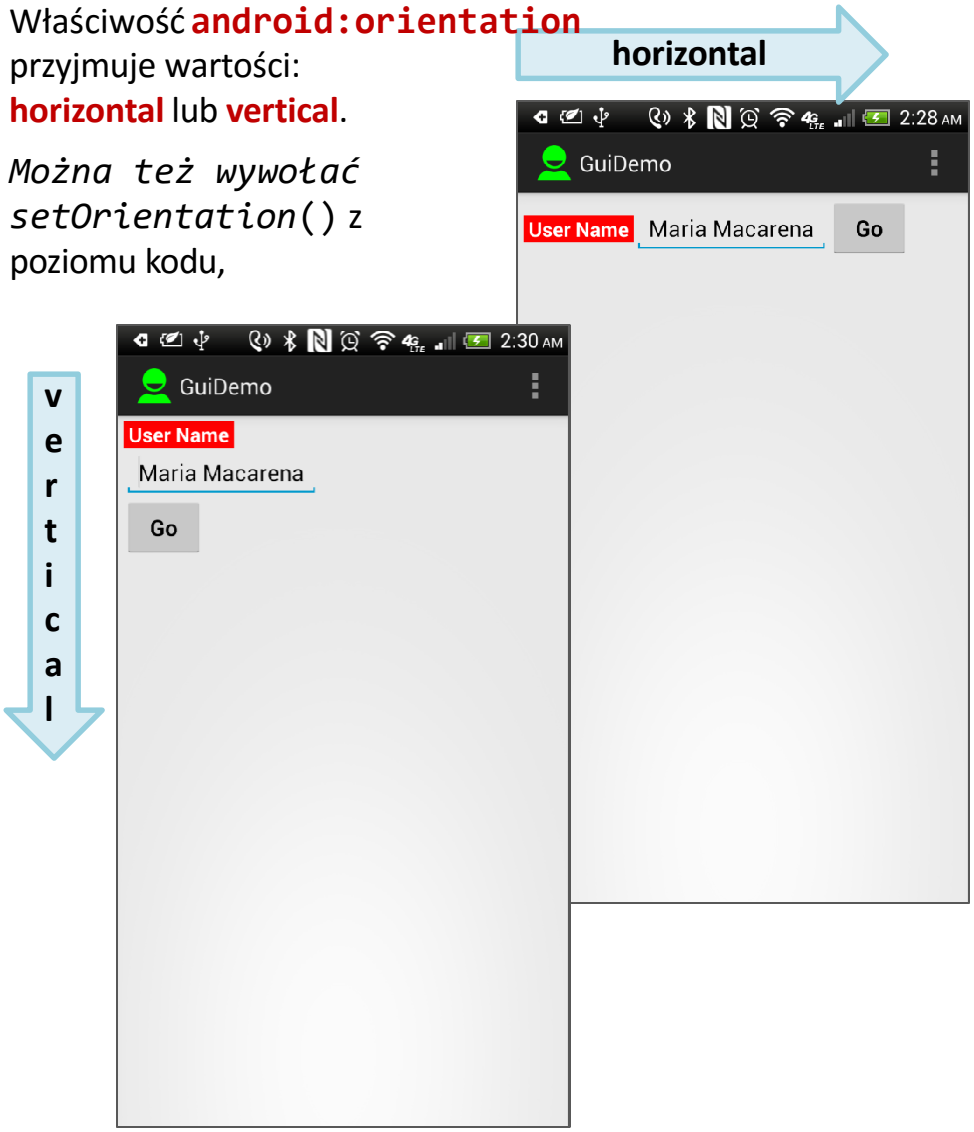
- orientation (*vertical, horizontal)*
- fill model (*match_parent, wrap_contents*)
- weight (*0, 1, 2, …n*)
- gravity (*top, bottom, center,…*)
- padding (*dp – dev. independent pixels*)
- margin (*dp – dev. independent pixels*)

LinearLayout - Orientation

1.1 Atrybut Orientation

Właściwość **android:orientation** przyjmuje wartości: **horizontal** lub **vertical**.

Można też wywołać setOrientation() z poziomu kodu,

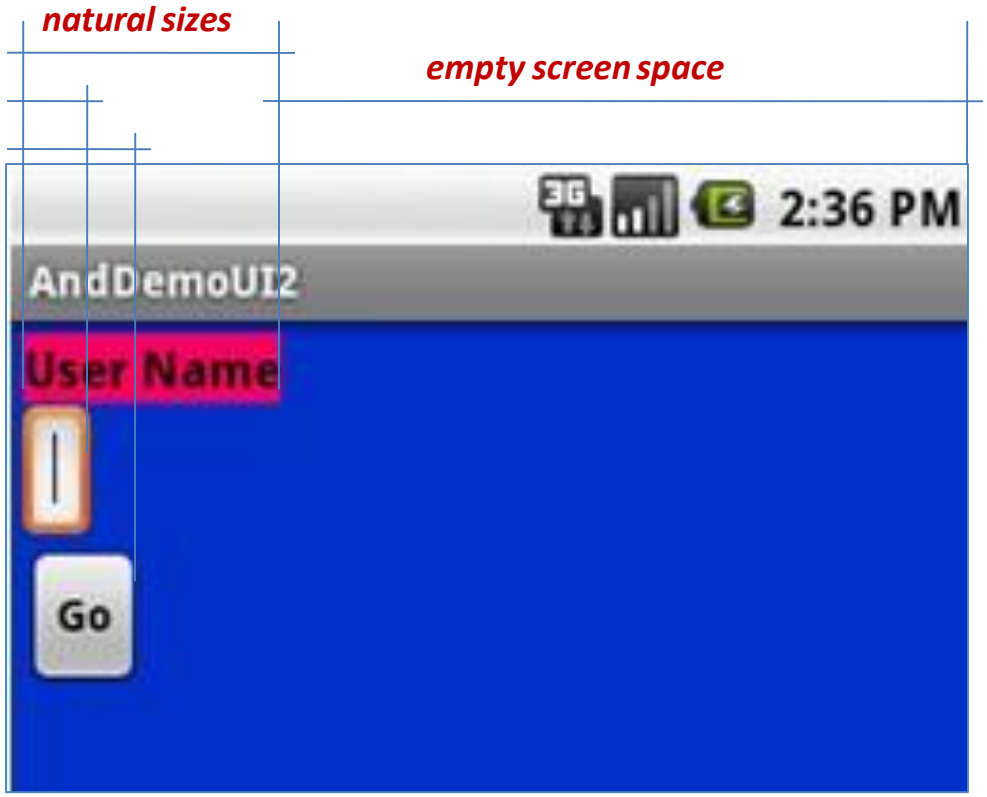



```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:padding="4dp" >

    <TextView
        android:id="@+id/labelUserName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ffff0000"
        android:text=" User Name "
        android:textColor="#ffffffff"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/ediName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Maria Macarena"
        android:textSize="18sp" />

    <Button
        android:id="@+id/btnGo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go"
        android:textStyle="bold" />

</LinearLayout>
```

LinearLayout – Fill Model

1.2 Fill Model

- Widżety posiadają "naturalny rozmiar“ określony na podstawie ich zawartości.
- W pewnych okolicznościach widżet powinien posiadać określony rozmiar (szerokość, wysokość) nawet jeżeli nie wprowadzono żadnego tekstu (tak jak na poniższym rysunku).



LinearLayout – Fill Model

1.2 Fill Model

Wszystkie widżety wewnątrz LinearLayout **muszą** posiadać nadane atrybuty ‘width’ i ‘height’.

```
android:layout_width
android:layout_height
```

Wartości używane do określenia szerokości czy wysokości to:

1. Określony rozmiar jak **125dp** (device independent pixels, a.k.a. **dip**)

2. **wrap_content** wskazuje, że widżet zachowuje swój naturalny rozmiar.

3. **match_parent** (kiedyś ‘**fill_parent**’) wskazuje, że widżet powinien być tak samo wielki jak jego rozdzic

LinearLayout – Fill Model

1.2 Fill Model

Medium resolution is: 320 x 480 dpi.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff0033cc
    "
    android:orientation="vertical
    "  android:padding="4dp" >

    <TextView
        android:layout_width="_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0066"
        android:text="User Name"
        android:textColor="#ff000000"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/ediName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp" />

    <Button
        android:id="@+id/btnGo"
        android:layout_width="125dp"
        android:layout_height="wrap_content"
        android:text="Go"
        android:textStyle="bold" />

</LinearLayout>
```

Row-wise

Use all the row

Specific size: 125dp

LinearLayout – Weight

1.2 Weight

Określa ile miejsca w LinearLayout powinien mieć zaalokowany widok. Należy użyć **0** jeżeli widok nie ma się rozciągać. Im większa waga, tym więcej miejsca posiada dany widżet na swoim poziomie hierarchii.

Przykład

Specyfikacja XML okna jest podobna jak w poprzednim przykładzie.

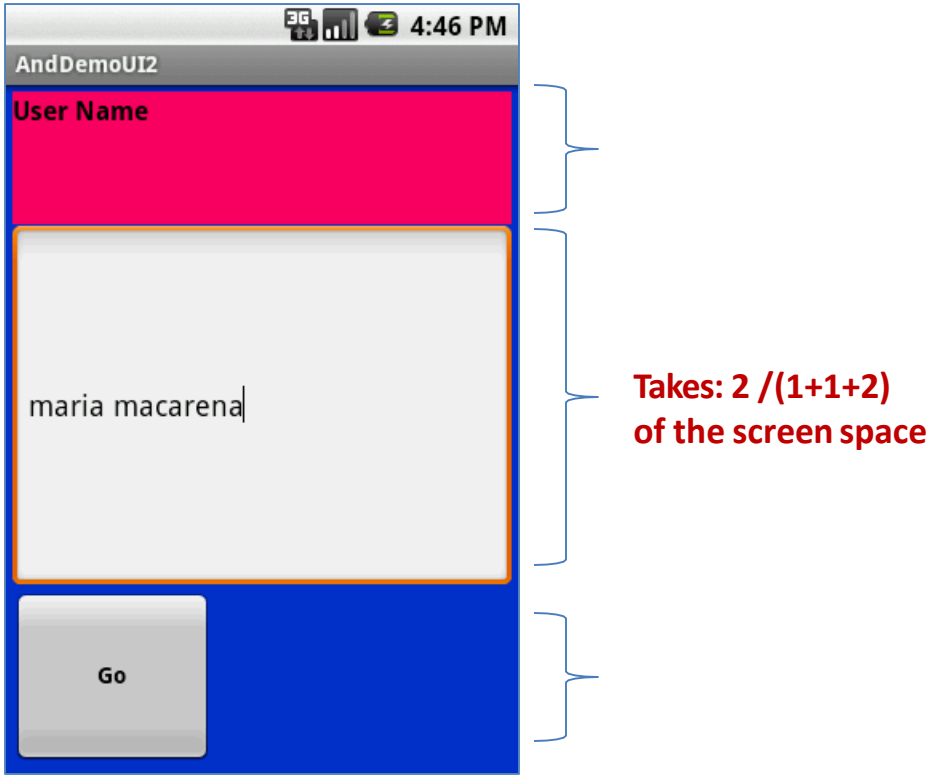
Kontroki TextView oraz Button mają dodatkowo właściwość:

```
android:layout_weight="1"
```

Gdzie komponent EditText ma

```
android:layout_weight="2"
```

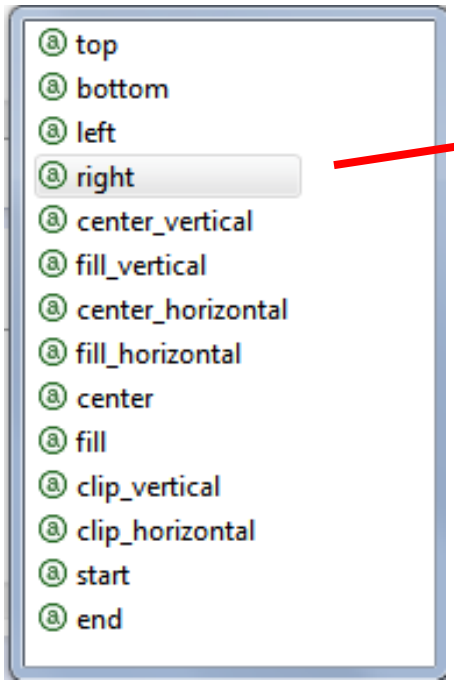
Domyślna wartość to 0

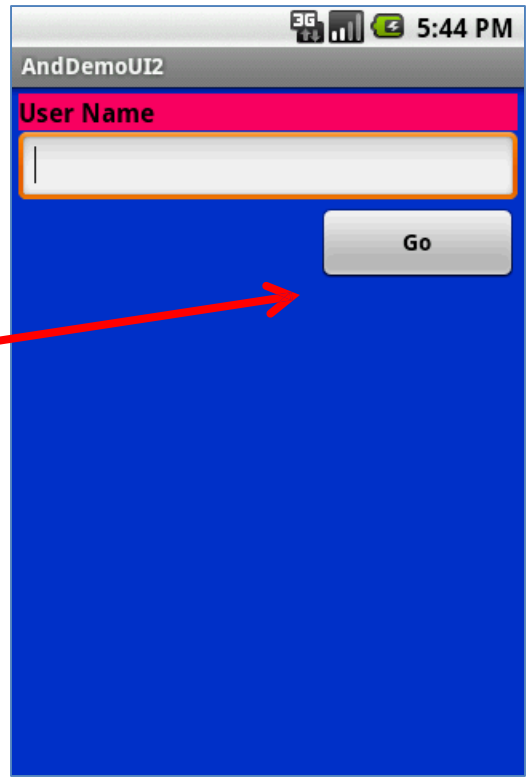


LinearLayout – Gravity

1.3 Layout_Gravity

- Używana by wskazać, jak wyrównane są elementy.
- Zwykle widżety pozycjonowane są do *lewej, górnej strony*.
- Wykorzystując w pliku XML właściwość
  `android:layout_gravity="..."`
  można ustawić inne wyrównanie:
  left, center, right, top, bottom, itp.





Button has **right** layout_gravity

The LinearLayout – Gravity

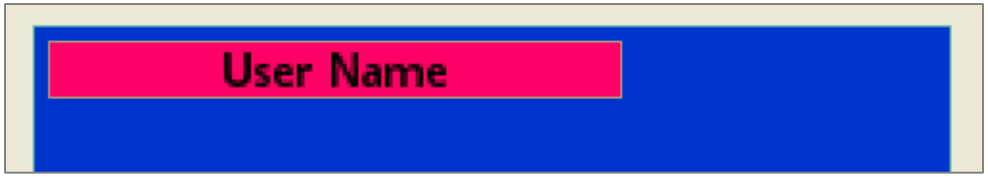
1.3 UWAGA: gravity vs. layout_gravity

Różnica względem:

android:gravity

Wskazuje jak zachowuje się obiekt wewnątrz kontenera. Tutaj tekst jest wycentrowany

```
android:gravity="center"
```



android:layout_gravity

Pozycjonuje widżet względem rodzica:

```
android:layout_gravity="center"
```

LinearLayout – Padding

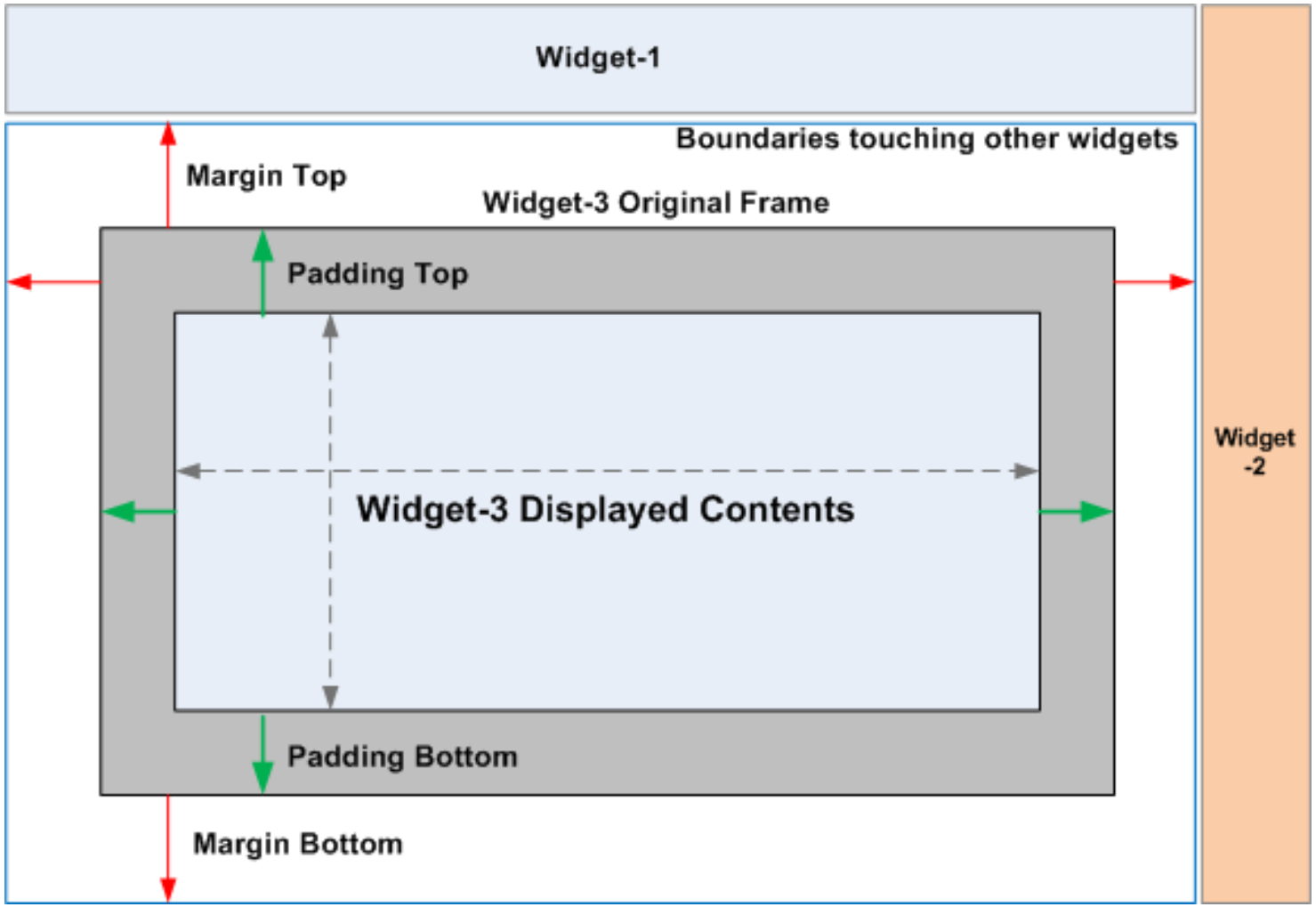
1.4 Padding

- Atrybut **padding** określa wewnętrzny margines widżetu (w **dp**).
- Wewnętrzny margines dodaje dodatkowe miejsce między obramowaniem widżetu a jego faktyczną zawartością.
- Używaj
  - `android:padding` (właściwości)
  - lub metody `setPadding()` z poziomu kodu

LinearLayout – Padding

1.3 Padding i Margin

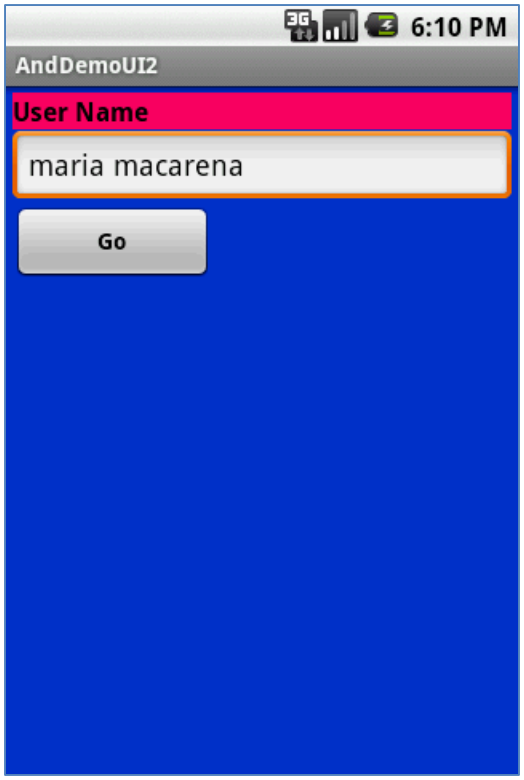
Różnice między wewnętrznym i zewnętrznym marginesem

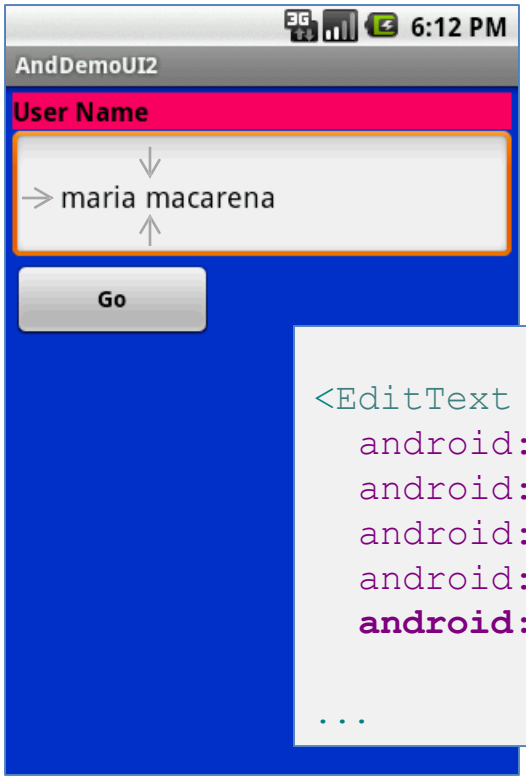


LinearLayout – Padding

1.3 Wewnętrzny margines używając padding

Przykład:
Komponentowi EditText nadano margines o wielkości 30dp



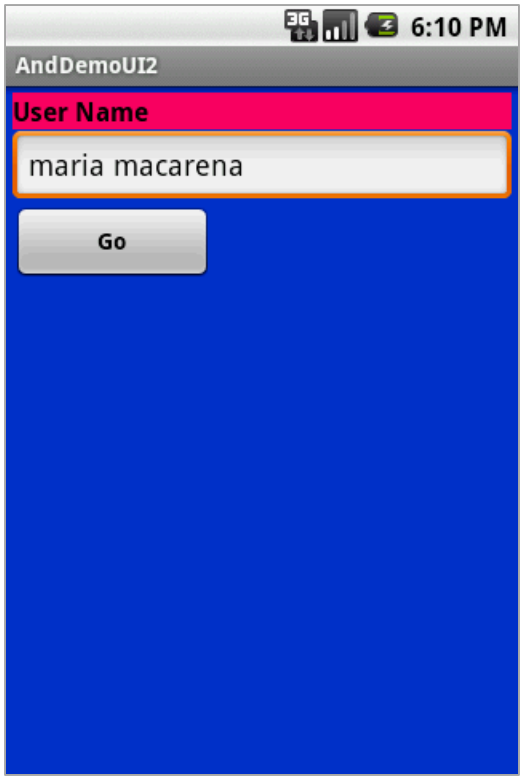


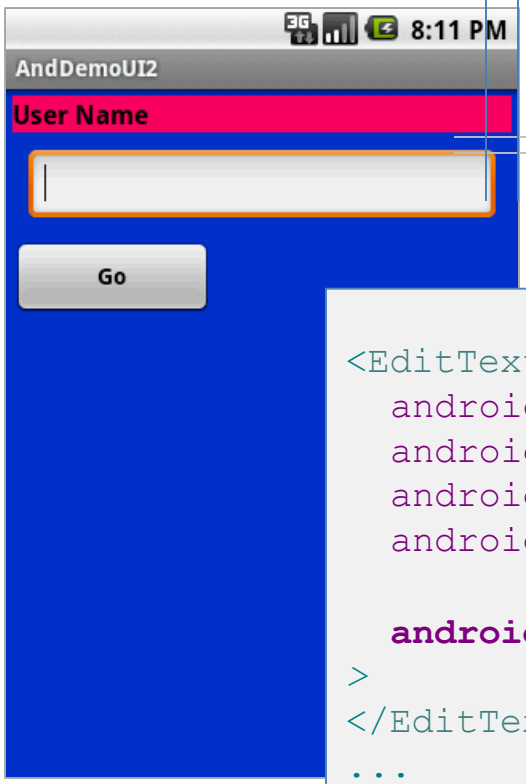

```
<EditText
   android:id="@+id/ediName"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:textSize="18sp"
   android:padding="30dp"   />

...
```

LinearLayout – Margin

1.4 Zewnętrzny margines

- Widżety domyślnie pozycjonowane są tuż obok siebie.
- Atrybut **android:layout_margin** umożliwia zwiększenie tego miejsca





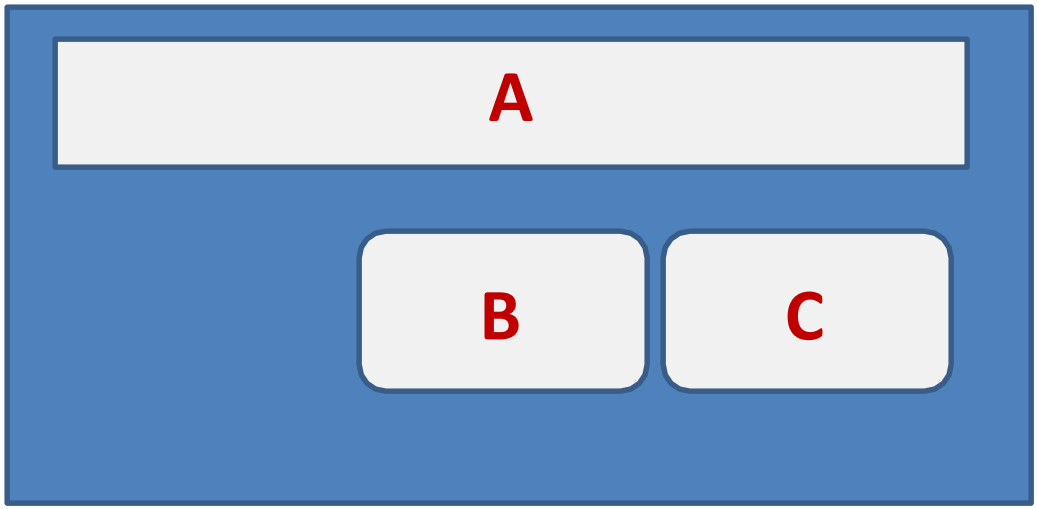
Increased inter-widget space

```
<EditText
   android:id="@+id/ediName"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:textSize="18sp"

   android:layout_margin="6dp"
>
</EditText>
...
```

Relative Layout

2. Relative Layout

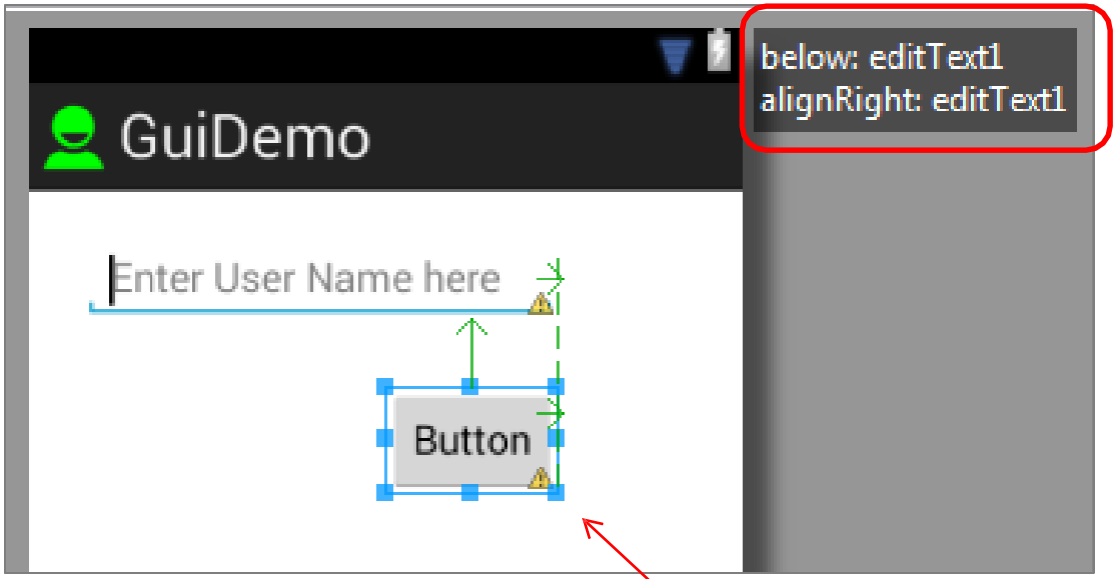
Rozmieszczenie widżetów w **RelativeLayout** uwzględnia relację sąsiedztwa do innych *widżetów w kontenerze* oraz w przodku danej hierarchii.



Przykład:
A jest w górnej części rodzica
C jest pod A, po jego prawej stronie
B jest pod A, na lewo od C

Relative Layout

2. Przykład: Relative Layout



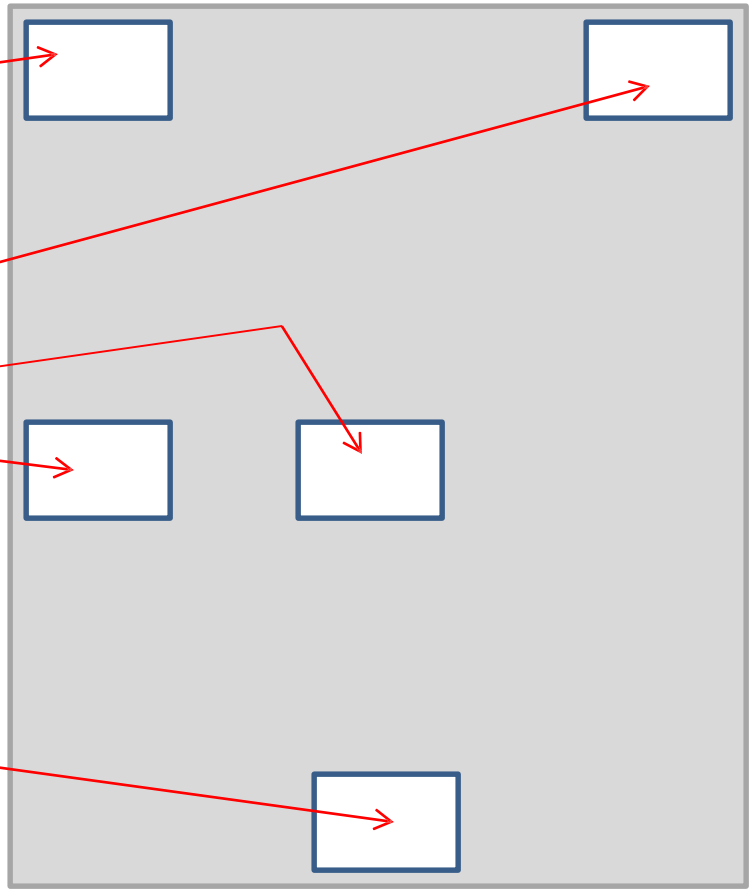
Lokalizacja przycisku jest wyrażona w odniesieniu do (relatywnej) pozycji pola tekstowego (EditText).

Relative Layout

2. Odniesienie do kontenera

Poniżej znajduje się lista atrybutów XML typu **boolean** (=“true/false”) określających pozycję widżetów względem jego **rodzica (kontenera)**.

```
android:layout_alignParentTop
android:layout_alignParentBottom

android:layout_alignParentLeft
android:layout_alignParentRight

android:layout_centerInParent
android:layout_centerVertical
android:layout_centerHorizontal
```

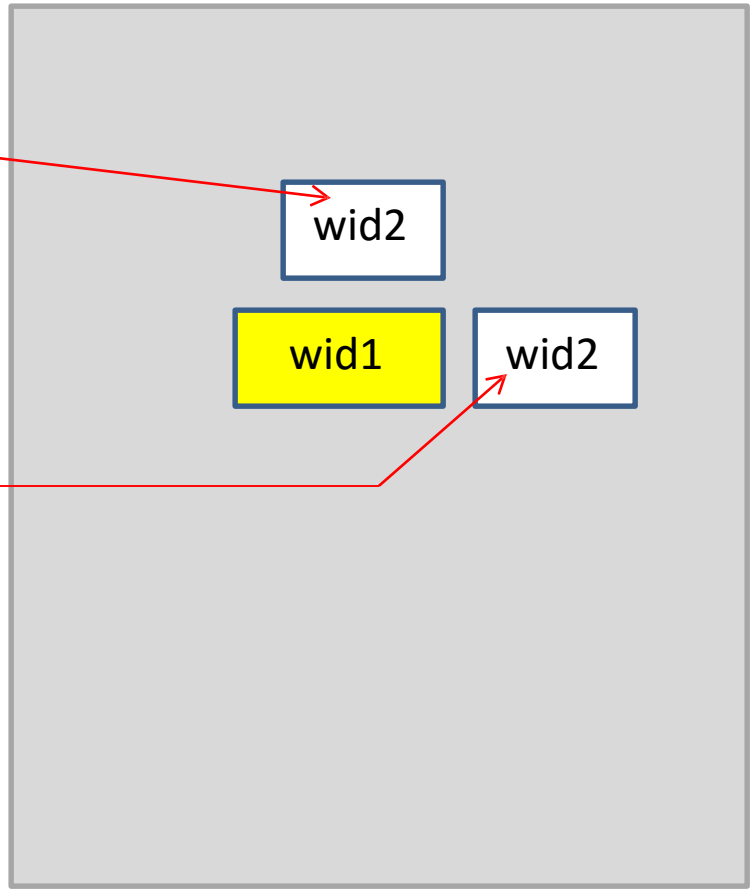


Relative Layout

2. Odniesienie do innych widżetów

Następujące atrybuty określają położenie widżetu **w odniesieniu do innych widżetów:**

```
android:layout_above="@+id/wid1"

android:layout_below

android:layout_toLeftOf

android:layout_toRightOf
```

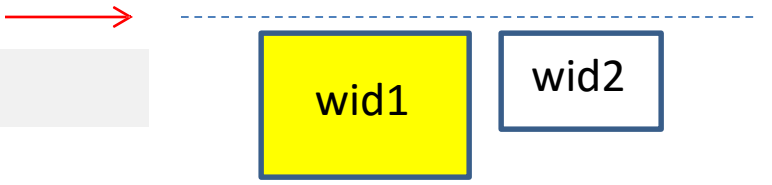


W tym przykładzie “wid2” jest w relacji sąsiedztwa z wid1 (“@+id/wid1”)

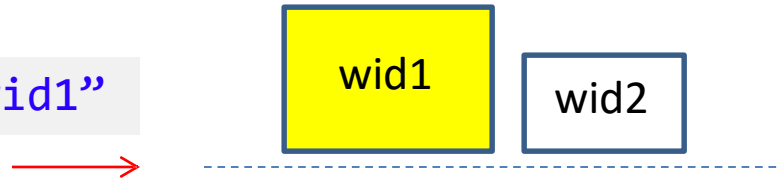
Relative Layout

2. Odniesienie do innych widżetów c.d.

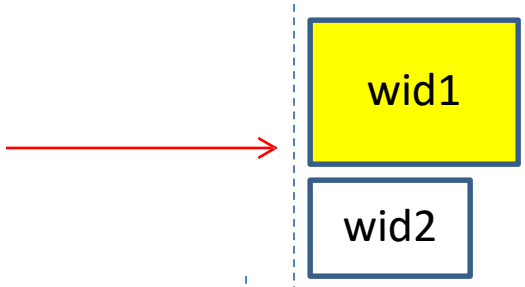
```
android:layout_alignTop=“@+id/wid1”
```



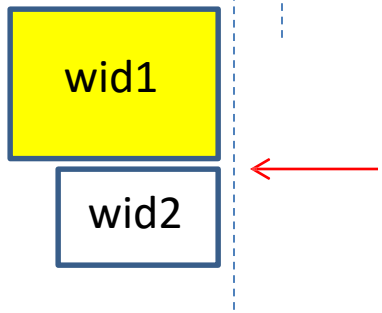

```
android:layout_alignBottom =“@+id/wid1”
```




```
android:layout_alignLeft=“@+id/wid1”
```




```
android:layout_alignRight=“@+id/wid1”
```



Relative Layout

2. Odniesienie do innych widżetów c.d.

Używając pozycjonowania relatywnego należy zadbać o:

1. Nadanie identyfikatorów (atrybut **`android:id`**) dla wszystkich elementów do których będziemy się odnosili.

2. Elementy XML są nazywane poprzez zapis: **`@+id/...`** Przykładowo pole tekstowe może być nazwane: `android:id="@+id/txtUserName"`

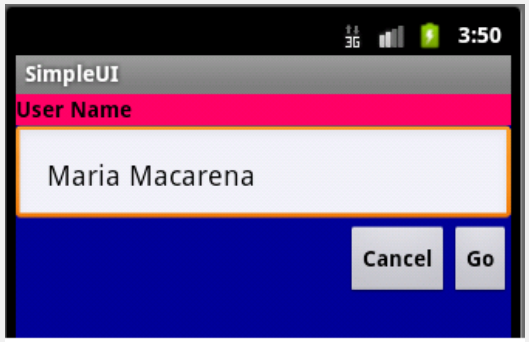
3. Można odwoływać się jedynie do uprzednio zdefiniowanych widzętów. Przykładowo, nowy komponent który znajduje się pod polem tekstowym *txtUserName można wypozycjonować jako*:

   `android:layout_below="@+id/txtUserName"`

Relative Layout

2. Przykład

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myRelativeLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff000099" >

    <TextView
        android:id="@+id/lblUserName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:background="#ffff0066
        "  android:text="User Name"
        android:textColor="#ff000000"
        android:textStyle="bold" >
    </TextView>
```




```
    <EditText
        android:id="@+id/txtUserName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/lblUserName"
        android:padding="20dp" >
    </EditText>

    <Button
        android:id="@+id/btnGo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

android:layout_alignRight="@+id/txtUserName"
        android:layout_below="@+id/txtUserName"
        android:text="Go"
        android:textStyle="bold" >
    </Button>

    <Button
        android:id="@+id/btnCancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/txtUserName"
        android:layout_toLeftOf="@+id/btnGo"
        android:text="Cancel"
        android:textStyle="bold" >
    </Button>

</RelativeLayout>
```

Table Layout

3. Table Layout

1. Układ **TableLayout** wykorzystuje siatkę do pozycjonowania widżetów.
2. Podobnie jak w macierzy, komórki siatki identyfikowane są po numerach wierszy i kolumn.
3. Kolumny są responsywne – mogą się zwężąć bądź rozciągać by dostosować się do ich zawartości.
4. Klasa **TableRow** wykorzystywana jest do stworzenia nowego wiersza w którym widżety mogą zostać umieszczone.
5. Liczba kolumn TableRow jest wyznaczana przez liczbę widżetów umieszczonych w danym wierszu.

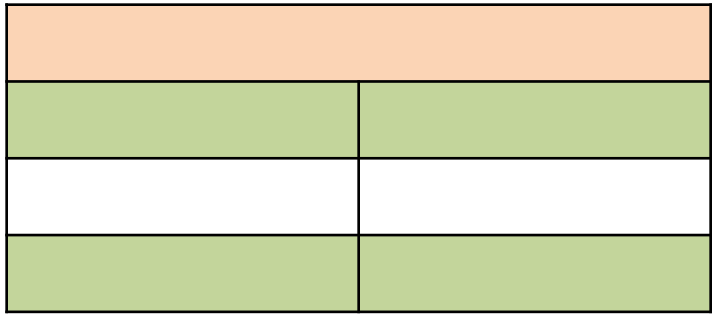


Table Layout

3. Table Layout – Ustawienie liczby kolumn

Liczba kolumn w ramach wiersza wyznacza jest przez system Android.

Przykład: Jeżeli TableLayout ma trzy wiersze, jeden z dwoma widżetami, jeden z trzema i jeden z czterema, zostaną stworzone przynajmniej 4 kolumny.

0		1	
0		1	2
0	1	2	3

Table Layout

3. Table Layout – Rozciąganie kolumn

- Pojedynczy widżet wewnątrz TableLayout może zajmować więcej niż jedną kolumnę.

- Właściwość `android:layout_span` wskazuje na jaką liczbę kolumn widżet może się rozciągnąć.

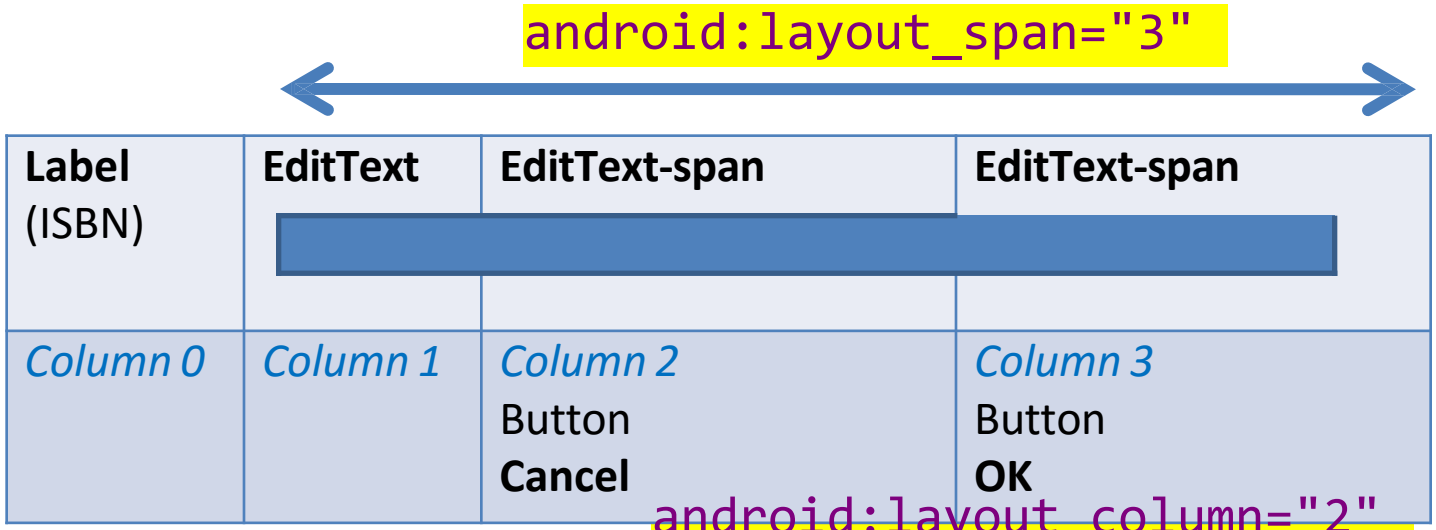
```
<TableRow>
    <TextView android:text="URL:" />
    <EditText android:id="@+id/txtData"
              android:layout_span="3" />
</TableRow>
```

Table Layout

3. Table Layout – Rozciąganie kolumn

Widżety w wierszu tabeli ustawiane są od lewej do prawej, rozpoczynając od pierwszej wolnej kolumny. Każda kolumna w układzie rozciąga się zgodnie z początkową zawartością danego widżetu.

Przykład: Pokazana tabela ma 4 kolumny (*indeksy*: 0,1,2,3). Nazwa (“*ISBN*”) tworzy pierwszą kolumnę (*indeks 0*). Pole tekstowe wykorzystuje atrybut `layout_span` by zostać rozciągniętym na 3 kolumny.



Android - Graphical User Interfaces

Table Layout – przykład 2

```
<TableLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myTableLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp" >

    <TableRow>
        <TextView
            android:background="#FF33B5E5
            "  android:text="Item " />
        <TextView
            android:layout_marginLeft="5dp
            "
            android:background="#FF33B5E5"
            android:text="Calories " />
        <TextView
            android:layout_marginLeft="5dp"
            android:background="#FF33B5E5"
            android:text="Price $ " />
    </TableRow>

    <View
        android:layout_height="1dp"
        android:background="#FF33B5E5" />

    <TableRow>

        <TextView
            android:text="Big Mac" />
        <TextView
            android:gravity="center"
            android:text="530" />
        <TextView
            android:gravity="center"
            android:text="3.99" />
        <Button
            android:id="@+id/btnBuyBigMac"
            android:gravity="center"
            android:text="Buy" />
    </TableRow>

    <View
        android:layout_height="1dp"
        android:background="#FF33B5E5" />

    <!-- other TableRows ommitted --!>


</TableLayout>
```

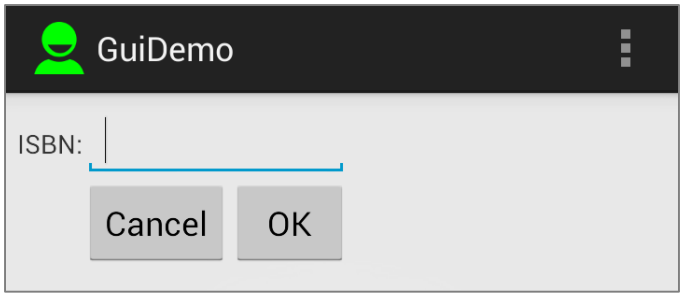
Table Layout

3. Table Layout - przykład

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myTableLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="6dp"
    android:orientation="vertical" >
    <TableRow>
        <TextView android:text="ISBN:" />
        <EditText
            android:id="@+id/ediISBN"
            android:layout_span="3" />
    </TableRow>

    <TableRow>
        <Button
            android:id="@+id/cancel"
            android:layout_column="2"
            android:text="Cancel" />
        <Button
            android:id="@+id/ok"
            android:text="OK" />
    </TableRow>

</TableLayout>
```

Zajmij 3 kolumny

Pomiń kolumny 0, 1



Spróbuj zmienić
layout_span na 1, 2, 3

Table Layout

3. Rozciąganie całej tabeli

- Domyślnie, kolumna jest szeroka jak “naturalny” rozmiar najszerszego widżetu zawartego w tej kolumnie (przykładowo kolumna z przyciskiem “Go” jest węższa niż kolumna z przyciskiem “Cancel”).

- Tabela niekoniecznie musi zajmować całą dostępną przestrzeń w poziomie.

- Jeżeli tabela ma (w poziomie) być zrównana z kontenerem można skorzystać z opcji:

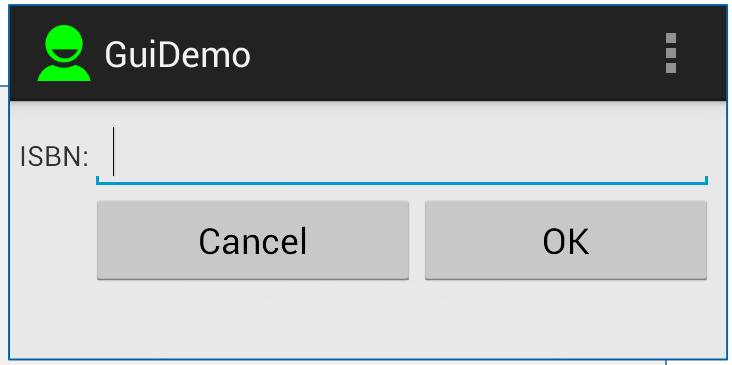
```
android:stretchColumns=“column(s)”
```

Jej wartość to indeks kolumny (bądź lista indeksów kolumn) która ma zostać rozciągnięta by zająć całą dostępną przestrzeń w ramach wiersza.

Table Layout

3. Przykład: Rozciąganie kolumn

W tym przykładzie kolumny 2 i 3 zajmują całą (wolną) dostępną przestrzeń w poziomie.



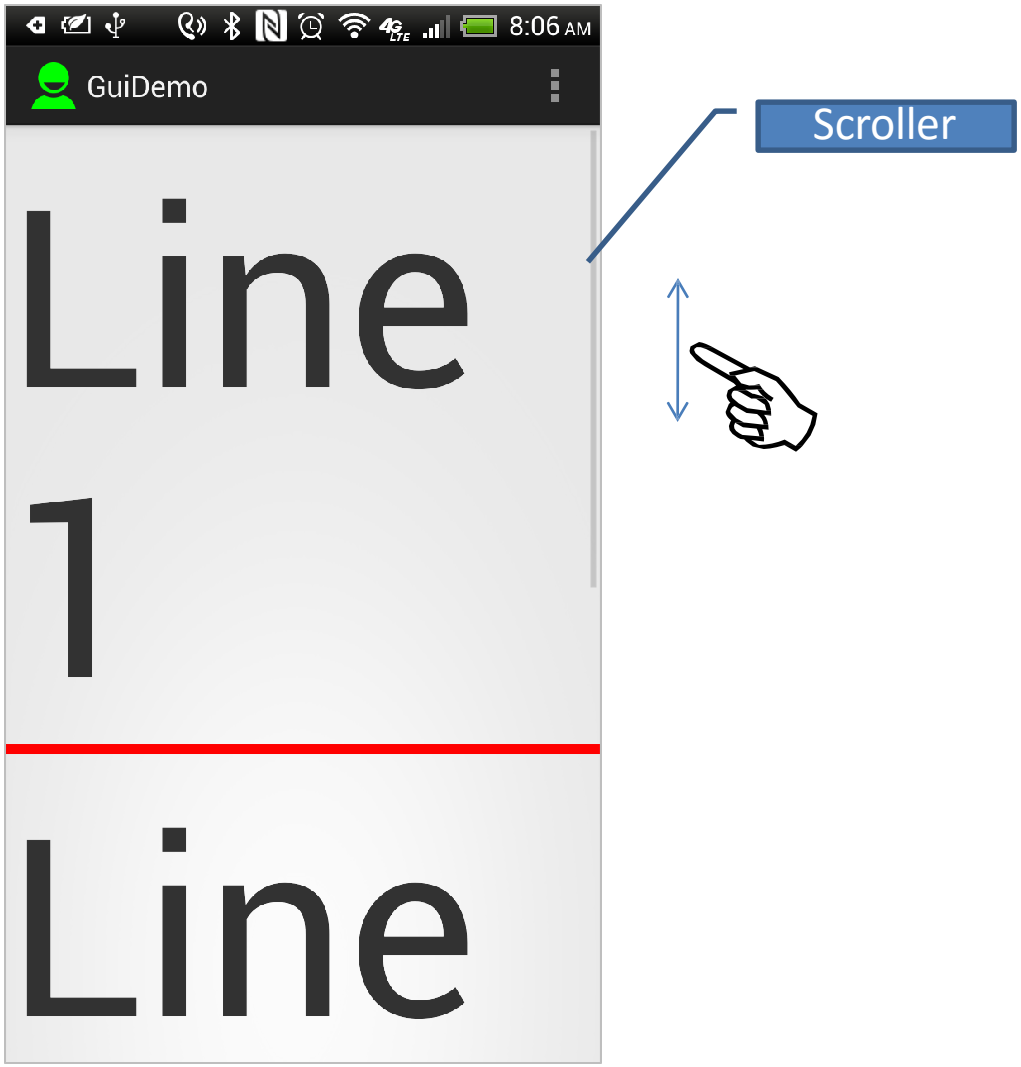

```
...
<TableLayout
  android:id="@+id/myTableLayout"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:stretchColumns="2,3"
  xmlns:android="http://schemas.android.com/apk/res/android"
>

...
```

***Do zrobienia**: spróbuj rozciągania z innymi kolumnami.*

ScrollView

4. ScrollView Layout

- Komponent **ScrollView** jest przydatny w sytuacjach, gdy do wyświetlenia jest więcej danych niż można pomieścić na danym ekranie.
- ScrollView zapewnia dostęp do danych z użyciem pasków przewijania.
- Jedynie porcja danych jest wyświetlana naraz. Pozostała część jest ukryta.



ScrollView

4. Przykład: ScrollView Layout

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

<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myScrollView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:id="@+id/myLinearLayoutVertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Line1"
            android:textSize="150dp" />
        <View
            android:layout_width="match_parent"
            android:layout_height="6dp"
            android:background="#ffff0000" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Line2"
            android:textSize="150dp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="6dp"
            android:background="#ffff0000" />

        <TextView
            android:id="@+id/textView3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Line3"
            android:textSize="150dp" />
    </LinearLayout>

</ScrollView>
```

Przykład HorizontalScrollView Layout

```
<HorizontalScrollView

xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myHorizontalScrollView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <LinearLayout
       android:id="@+id/myLinearLayoutVertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:orientation="horizontal" >

       <TextView
           android:id="@+id/textView1"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:text="Item1"
           android:textSize="75sp" />
        <View
           android:layout_width="6dp"
           android:layout_height="match_parent"
           android:background="#ffff0000" />

       <TextView
           android:id="@+id/textView2"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:text="Item2"
           android:textSize="75sp" />

       <View
         android:layout_width="6dp"
         android:layout_height="match_parent"
         android:background="#ffff0000" />

       <TextView
          android:id="@+id/textView3"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:text="Item3"
          android:textSize="75sp" />
    </LinearLayout>

</HorizontalScrollView>
```

Absolute Layout

5. Absolute Layout

- Układ pozwala podać dokładną pozycje (koordynaty x/y) jego potomków.
- Pozycjowanie absolutne *jest mniej elastyczne i trudniejsze w utrzymaniu* niż w przypadku pozostałych układów opartych o relację między komponentami.

Absolute Layout

5. Przykład Absolute Layout

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
android:id="@+id/myLinearLayout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#ff0033cc"
android:padding="4dp"
xmlns:android="http://schemas.android.com
/apk/res/android"
>

<TextView
android:id="@+id/tvUserName"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="#ffff0066"
android:text="User Name"
android:textSize="16sp"
android:textStyle="bold"
android:textColor="#ff000000"
android:layout_x="0dp"
android:layout_y="10dp"
>
</TextView>
<EditText
android:id="@+id/etName"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:textSize="18sp"
android:layout_x="0dp"
android:layout_y="38dp"
>
</EditText>

<Button
android:layout_width="120dp"
android:text="Go"
android:layout_height="wrap_content"
android:textStyle="bold"
android:id="@+id/btnGo"
android:layout_x="100dp"
android:layout_y="170dp"  />
</AbsoluteLayout>
```