Układy Layouts w Android

mgr inż. Stanisław Lota

Wprowadzenie

Układ (ang. *layout*) to podstawowy element każdego interfejsu użytkownika. To na nim umieszczamy kontrolki i inne elementy, które będzie widział użytkownik i z których będzie korzystał i ich położenie na ekranie aplikacji.

Układy mają **postać zbiorów definicji obiektów zapisanych w postaci kodu XML**. Poszczególne definicje są wykorzystywane do tworzenia obiektów pojawiających się na ekranie, takich jak przyciski czy tekst.

Wprowadzenie

Podstawowym blokiem konstrukcyjnym interfejsu użytkownika jest obiekt **View,** który jest tworzony z klasy View. Zajmuje on prostokątny obszar na ekranie i jest odpowiedzialny za rysowanie i obsługę zdarzeń.

View to klasa bazowa dla widżetów, które służą do tworzenia interaktywnych komponentów interfejsu użytkownika, takich jak przyciski, pola tekstowe itp.

Wprowadzenie

Poniżej przedstawiono niektóre typowe podklasy View, które będą używane w aplikacjach na Androida.

- TextView
- EditText
- Button
- CheckBox
- RadioButton
- ImageButton
- Progress Bar
- Spinner

Wprowadzenie

ViewGroup jest podklasą **View. W ViewGroup obiekty są zwykle nazywane układami np. LinearLayout czy ConstraintLayout.**

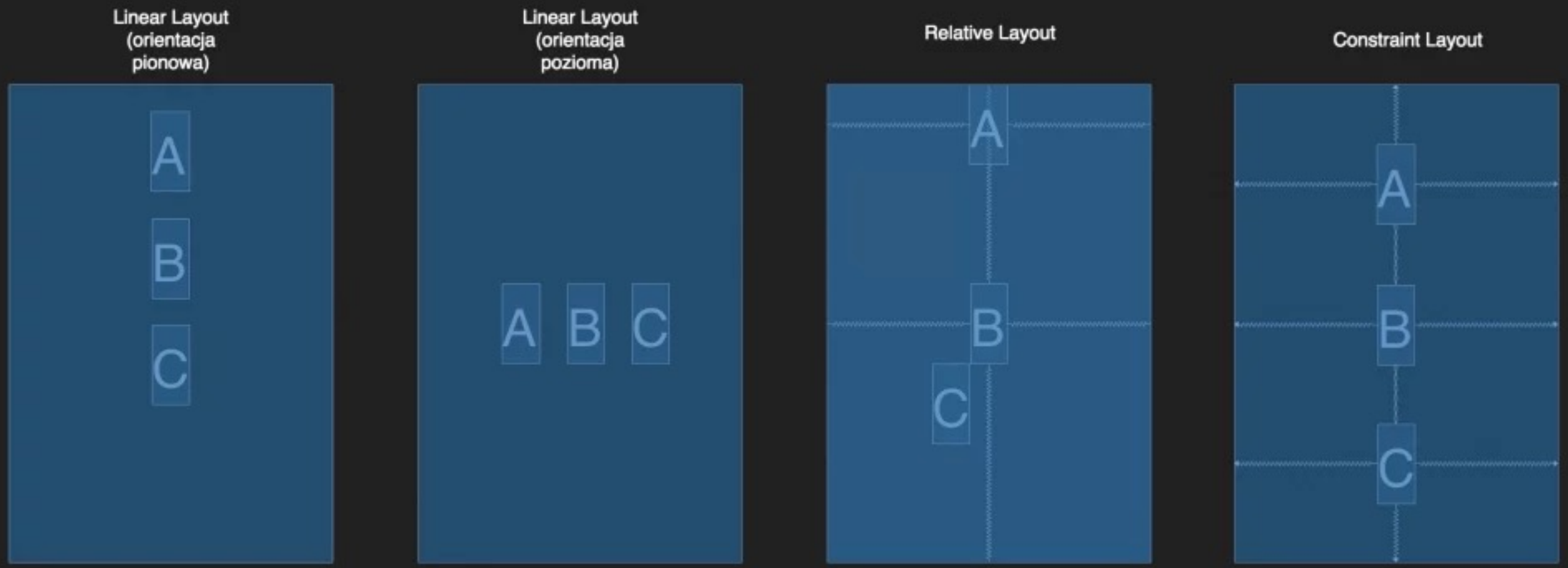
Typowy układ definiuje wizualną strukturę interfejsu użytkownika Androida i może być tworzony w czasie wykonywania przy użyciu obiektów. Zapewnia niewidoczny pojemnik do przechowywania widoków lub układów.

ViewGroup

Poniżej przedstawiono niektóre często używane podklasy dla ViewGroup:

- LinearLayout
- RelativeLayout
- FrameLayout
- GridView
- ListView

ViewGroup


Linear Layout (orientacja pionowa)
A
B
C
Linear Layout (orientacja pozioma)
A B C
Relative Layout
A
B
C
Constraint Layout
A
B
C

Układ

Layout możemy tworzyć zarówno w sposób wizualny (zakładka *Design*), jak i w sposób programistyczny (XML). Wszystkie zmiany dokonywane w jednym z tych widoków są automatycznie odzwierciedlane w drugim.

Atrybuty układu w systemie Android

Atrybuty dostosowywania układu podczas jego definiowania:

- **android: id:** jednoznacznie identyfikuje układ Androida.
- **android: hint:** Pokazuje podpowiedź, co należy wypełnić wewnątrz EditText.
- **android: layout_height:** Ustawia wysokość układu.
- **android: layout_width:** Ustawia szerokość układu.
- **android: layout_gravity:** Ustawia pozycję widoku potomnego.

Atrybuty układu w systemie Android

- **android: layout_marginTop:** Ustawia margines od góry układu.
- **android: layout_marginBottom:** Ustawia margines od dołu układu.
- **android: layout_marginLeft:** Ustawia margines z lewej strony układu.
- **android: layout_marginRight:** Ustawia margines z prawej strony układu.
- **android: layout_x:** określa współrzędne osi x układu.
- **android: layout_y:** określa współrzędne osi y układu.

ConstraintLayout

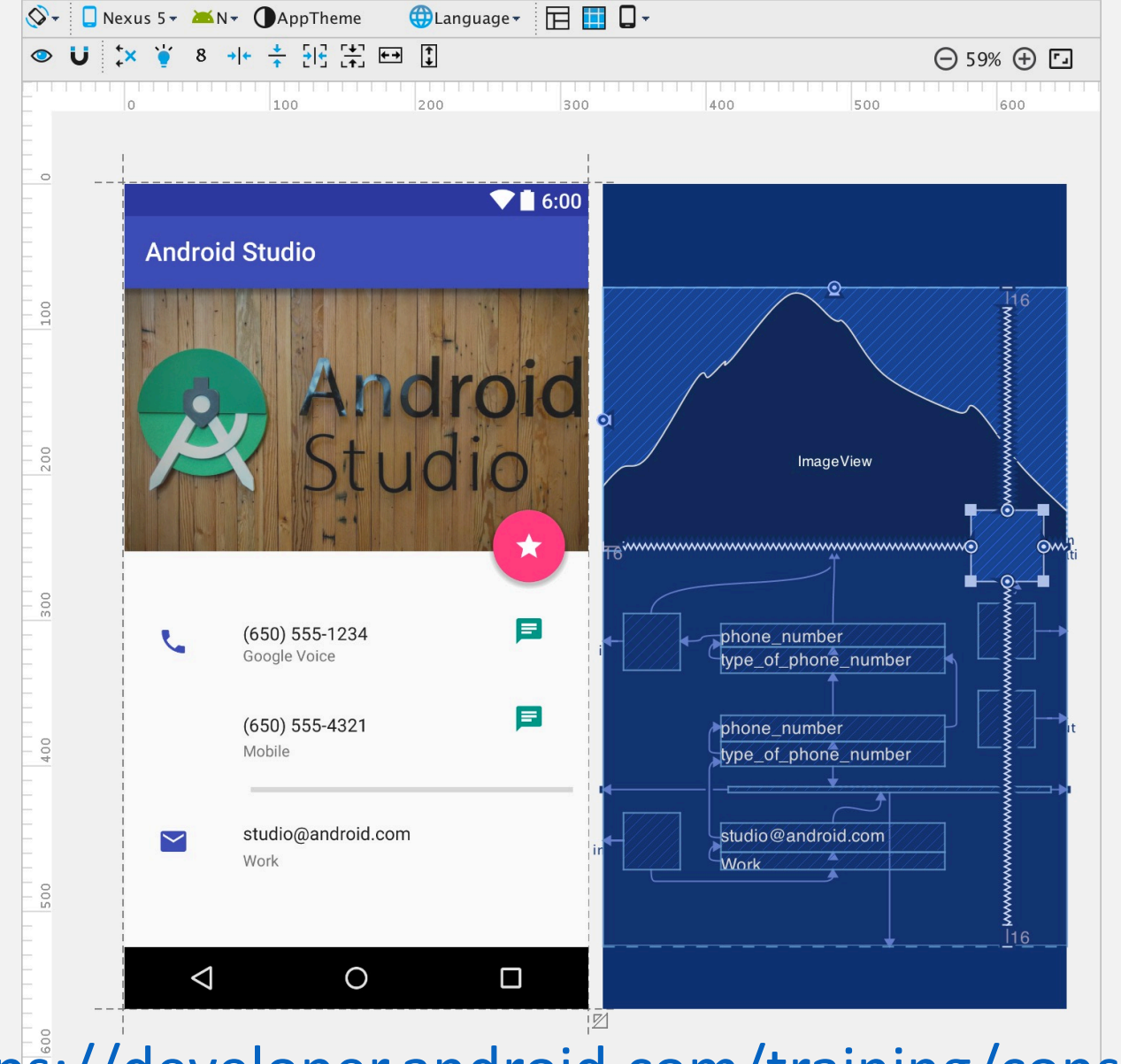
ConstraintLayout to układ w systemie Android, który pozwala na tworzenie złożonych układów o płaskiej strukturze, czyli bez zagnieżdżania w nim innych układów. Pozwala też na dodawanie animacji.

ConstraintLayout jest bardzo podobny do RelativeLayout, ponieważ widoki są rozmieszczone zgodnie z relacjami między widokami rodzeństwa a układem nadrzędnym. Został wydany na Google I / O 2016.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent" >

<TextView

android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Hello World!"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintBottom_toBottomOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



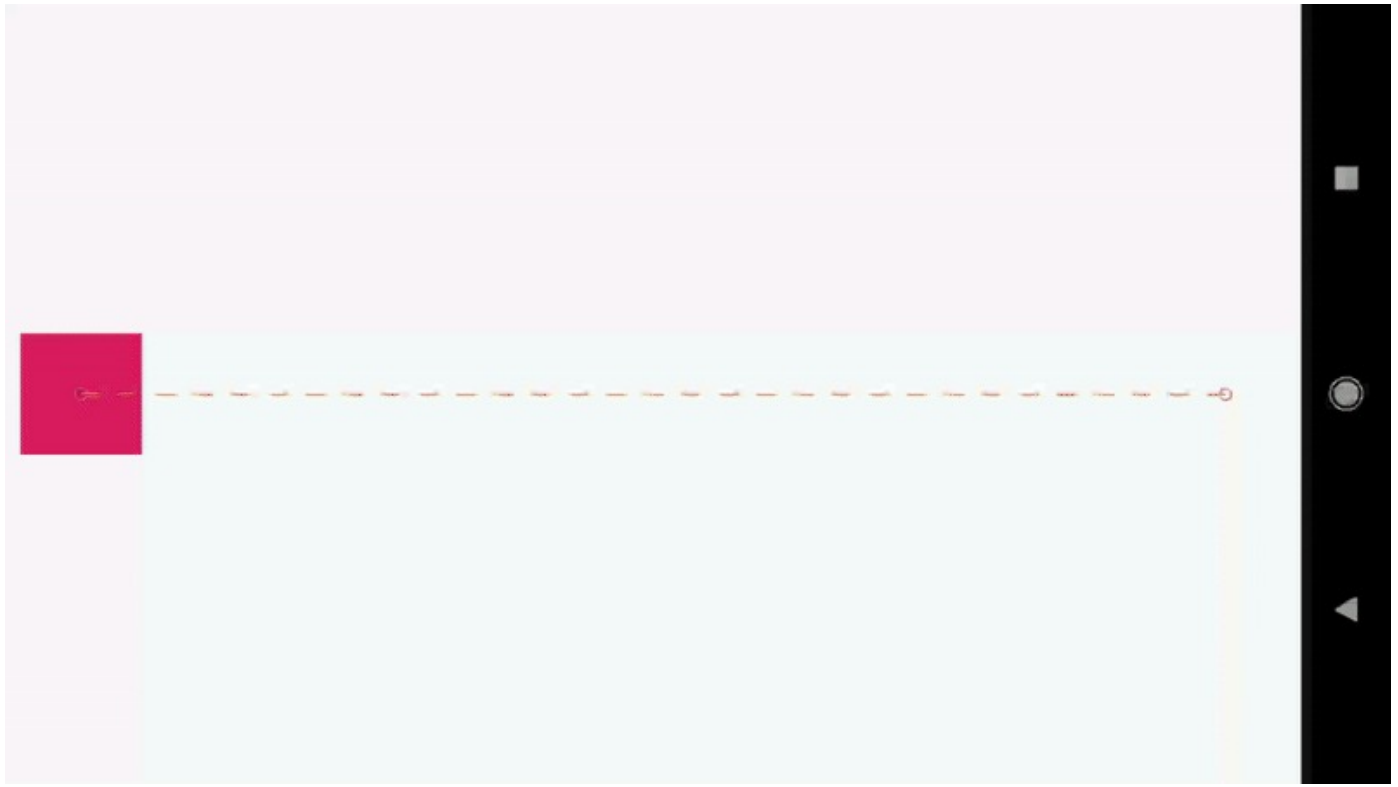
Tutorial : https://developer.android.com/training/constraint-layout

MotionLayout

MotionLayout to typ układu, który pomaga zarządzać animacjami widżetów w aplikacji. MotionLayout stanowi podklasę ConstraintLayout i jest przeznaczony do przesuwania, zmiany rozmiaru i animacji elementów interfejsu użytkownika, z którymi użytkownicy wchodzą w interakcje, takich jak przyciski i paski tytułu.

Ruch w aplikacji nie powinien być zbędnym efektem specjalnym w aplikacji.

MotionLayout



Tutorial : https://developer.android.com/training/constraint-layout/motionlayout

RelativeLayout

Układ RelativeLayout, jak sama nazwa wskazuje, zbudowany jest na podstawie idei ustalania położenia komponentów w odniesieniu do położenia innych komponentów. Słowo Relative możemy przetłumaczyć jako „względny” lub „zależny”

Możemy podać lokalizację naszego komponentu w odniesieniu do innych komponentów leżących na tym samym rodzicu bądź podać lokalizację naszego komponentu w odniesieniu do samego rodzica.

RelativeLayout

W układzie **RelativeLayout** możliwe jest wyrównanie pozycji elementów względem kontenera nadrzędnego. Aby tak to zdefiniować, piszemy:

- android: layout_alignParentTop = „true”
- android: layout_alignParentLeft = „true”

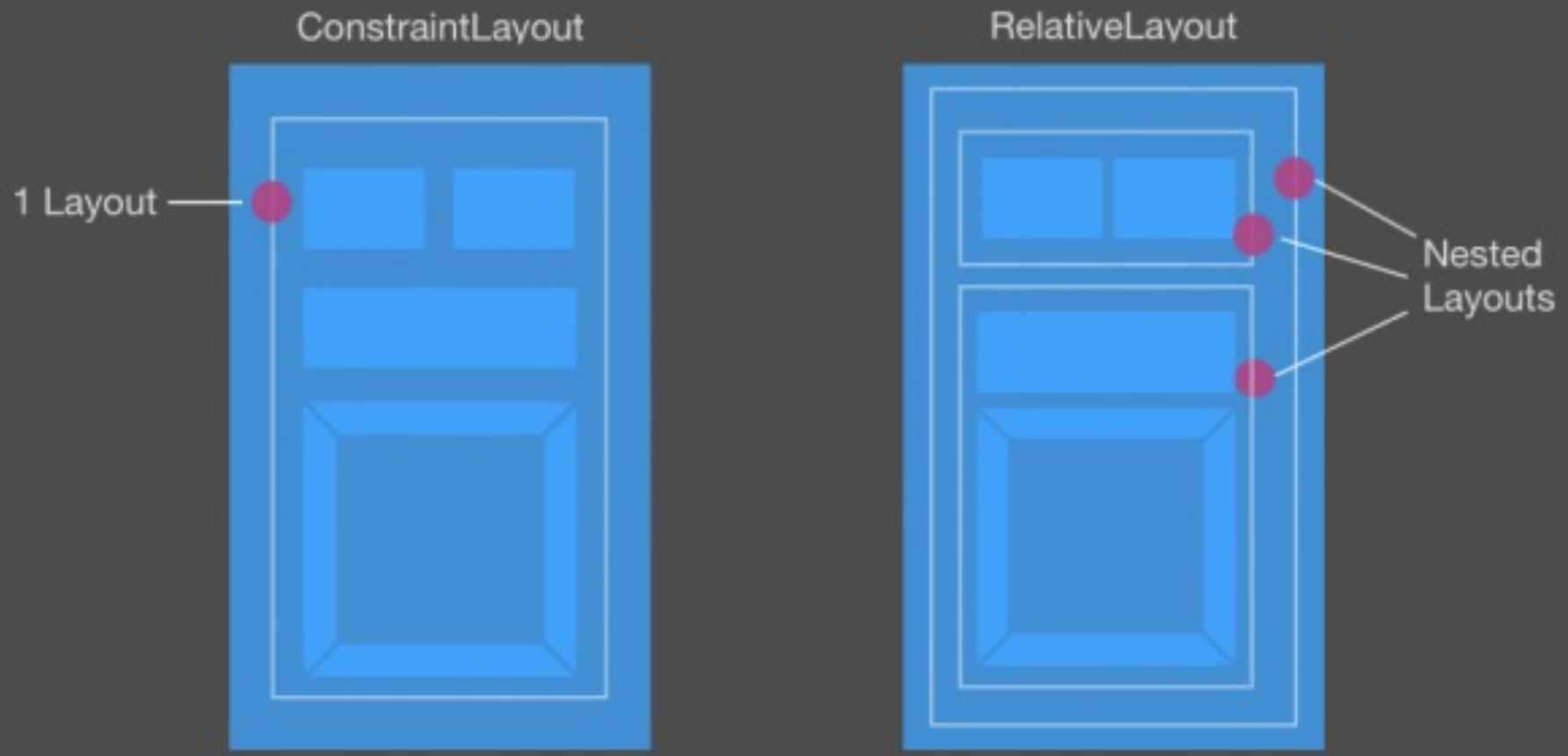
Jeśli napiszemy powyższy kod, element zostanie wyrównany w lewym górnym rogu kontenera nadrzędnego.

RelativeLayout

Jeśli chcemy wyrównać go z jakimś innym elementem w tym samym kontenerze, można to zdefiniować w następujący sposób:

- android: layout_alignLeft = „@ + id / nazwa_elementu”
- android: layout_below = „@ + id / nazwa_elementu”

Spowoduje to wyrównanie elementu poniżej drugiego elementu po jego lewej stronie.


ConstraintLayout
1 Layout
RelativeLayout
Nested
Layouts

RelativeLayout

Dla przykładu jeśli mamy 3 widoki A, B oraz C to, chcąc ułożyć je jeden pod drugim, musimy powiedzieć systemowi coś w stylu: „Przypnij widok A do góry, widok B do widoku A, a widok C do widoku B”. W ten sposób powstanie nam taka wieża. Relative Layout jest używany w sytuacjach, gdzie chcemy osiągnąć efekt niemożliwy do zrealizowania przy użyciu Linear Layout.

Przykład

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"> <ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/imageView"
android:src="@mipmap/ic_launcher" /> <EditText
android:layout_width="match_parent"
android:layout_height="wrap_content" android:id="@+id/editText"
android:layout_toRightOf="@+id/imageView"
android:layout_toEndOf="@+id/imageView" android:hint="@string/hint" />
</RelativeLayout>
```

RelativeLayout

GridLayout

GridLayout to układ siatki podzielony niewidocznymi liniami na wiersze i kolumny. Siatka ta tworzy komórki (ang. cells), w których możemy umieszczać komponenty, co daje nam więcej możliwości niż w układzie tabelarycznym (TableLayout).

Układ GridLayout jest bardzo dobrze zintegrowany z edytorem graficznym, dzięki czemu w łatwy sposób będziemy mogli z niego korzystać.

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
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"
android:columnCount="2" android:rowCount="2"> <TextView
android:layout_width="@dimen/fixed" android:layout_height="wrap_content"
android:text="@string/first" android:background="@color/colorPrimary"
android:layout_margin="@dimen/default_margin" /> <TextView
android:layout_width="@dimen/fixed" android:layout_height="wrap_content"
android:text="@string/second" android:background="@color/colorPrimary"
android:layout_margin="@dimen/default_margin" /> <TextView
android:layout_width="@dimen/fixed" android:layout_height="wrap_content"
android:text="@string/third" android:background="@color/colorPrimary"
android:layout_margin="@dimen/default_margin" />
</GridLayout>
```

GridLayout


10:47
SODocAndroid
First element
Second element
Third element

FrameLayout

Układ FrameLayout rezerwuje dany obszar ekranu w celu wyświetlenia pojedynczego elementu. Jeśli ułożymy na nim kilka elementów, to będą one wyświetlane w formie stosu. Na górze zaś znajdzie się ostatnio dodany element.

FrameLayout powinien być używany do przechowywania pojedynczego widoku podrzędnego, ponieważ organizowanie widoków podrzędnych w sposób skalowalny do różnych rozmiarów ekranu bez nakładania się elementów podrzędnych na siebie może być trudne.

Przykład

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"> <ImageView
android:src="@drawable/nougat" android:scaleType="fitCenter"
android:layout_height="match_parent"
android:layout_width="match_parent"/> <TextView
android:text="FrameLayout Example" android:textSize="30sp"
android:textStyle="bold" android:layout_height="match_parent"
android:layout_width="match_parent" android:gravity="center"/>
</FrameLayout>
```

FrameLayout

LinearLayout

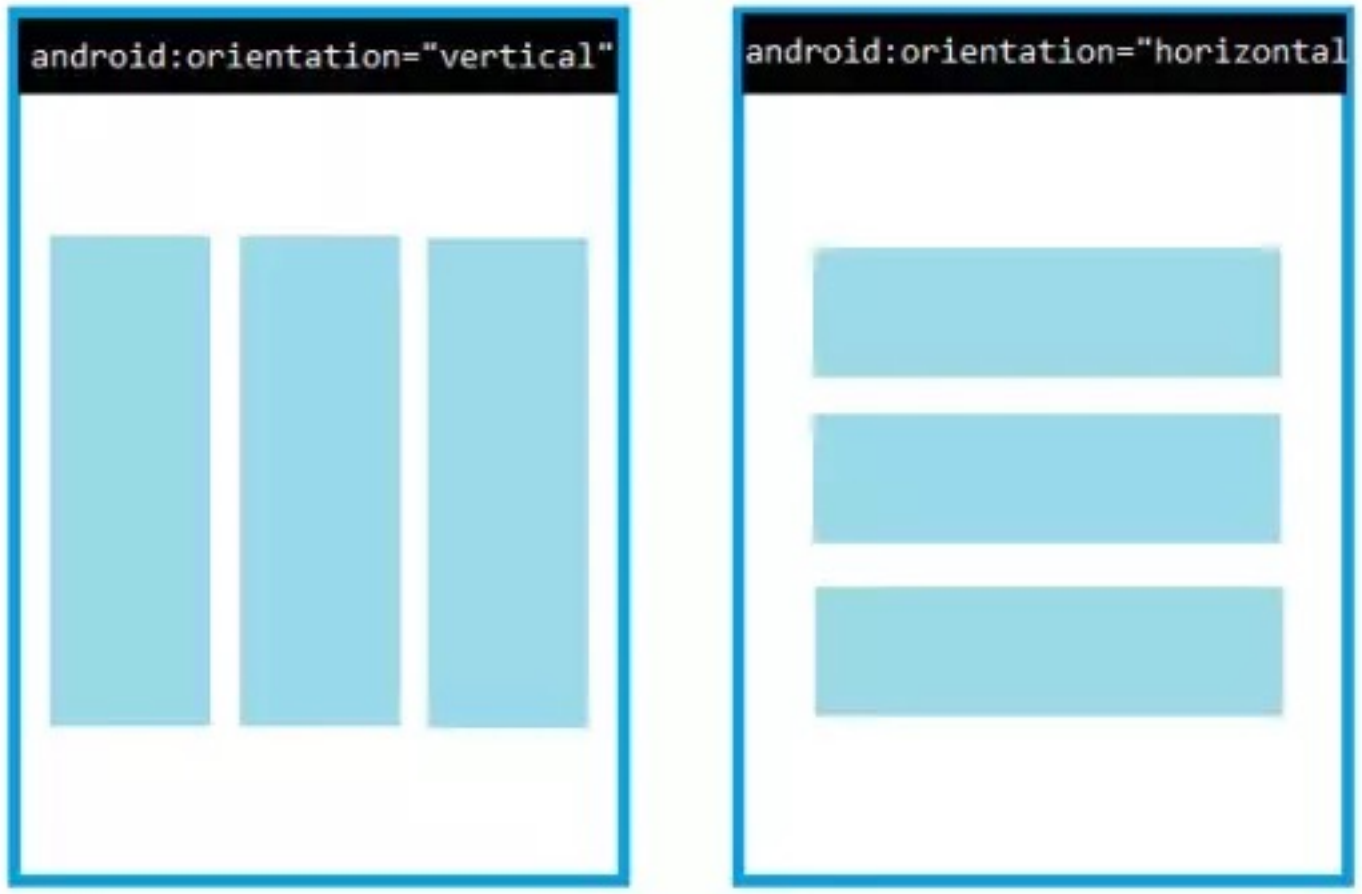
LinearLayout, czyli layout liniowy jest to układ, w którym elementy układane są liniowo. Można je układać na dwa sposoby:

LinearLayout (Horizontal) — wyświetla komponenty w jednym wierszu (jeden obok drugiego).

LinearLayout (Vertical) — wyświetla komponenty w jednej kolumnie (jeden pod drugim).

To, czy wszystkie elementy zostaną ułożone poziomo czy pionowo, zależy od wartości atrybutu android:orientation. Domyślnie orientacja jest **pozioma.**

LinearLayout



Przykład

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical" android:layout_width="match_parent"
android:layout_height="match_parent"> <TextView
android:layout_width="match_parent"
android:layout_height="wrap_content" android:text="@string/app_name"
/> <TextView android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@android:string/cancel" />
</LinearLayout>
```

LinearLayout

TableLayout

TableLayout— wyświetla elementy w układzie tabeli: każdy wiersz jest reprezentowany przez komponent podrzędny TableRow.

Kontenery TableLayout nie wyświetlają linii obramowania dla swoich wierszy, kolumn lub komórek. Tabela będzie miała tyle kolumn, ile wiersz z największą liczbą komórek. Tabela może pozostawić komórki puste. Komórki mogą obejmować wiele kolumn, tak jak w HTML.

ScrollView

Komponent ScrollView jest kontenerem dla innych układów pozwalającym na przewijanie ich zawartości, który przydaje się w sytuacji, kiedy rozmiar układu przekracza fizyczną wielkość naszego ekranu.

Z technicznego punktu widzenia komponent ScrollView dziedziczy po układzie FrameLayout, co oznacza, że jest on przeznaczony do wyświetlania jednego elementu zajmującego cały jego obszar.

Najczęściej wykorzystujemy go wraz z układem LinearLayout w orientacji pionowej zawierającym elementy poukładane jeden pod drugim, dzięki czemu użytkownik może je swobodnie przewijać.

Zadanie

Popracuj nad wykorzystaniem MotionLayout oraz ConstraintLayout.

Tutorial : https://developer.android.com/training/constraint-layout

Tutorial https://developer.android.com/training/constraint-layout/motionlayout