Carmen Delessio, Lauren Darcey, Shane Conder

Android Studio w 24 godziny

Wygodne programowanie dla platformy Android

SAMS

Helion

Tytuł oryginału: Sams Teach Yourself Android™ Application Development in 24 Hours, Fourth Edition

Tłumaczenie: Grzegorz Kowalczyk
Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

ISBN: 978-83-283-2150-2

Authorized translation from the English language edition: ANDROID APPLICATION DEVELOPMENT IN 24 HOURS, SAMS TEACH YOURSELF, ISBN 0672337398; by Carmen Delessio; and Lauren Darcey; and Shane Conder; published by Pearson Education, Inc, publishing as SAMS Publishing.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem: *ftp://ftp.helion.pl/przyklady/as24w4.zip*

Spis treści

Godzina 7

Tworzenie układów interfejsów użytkownika

W tej godzinie:

- **Rozpoczynamy pracę z układami interfejsów użytkownika**
- **Zastosowanie układu `RelativeLayout`**
- **Zastosowanie różnych rodzajów układów interfejsu**
- **Tworzenie efektywnych układów interfejsu użytkownika**

W systemie Android układy (ang. *layouts*) są wykorzystywane do prawidłowego wyświetlania interfejsu użytkownika na różnych urządzeniach. Każda aktywność, którą utworzyłeś w aplikacji, posiada powiązany z nią plik XML układu interfejsu, wykorzystujący komponent `RelativeLayout`. Pliki układu to szablony opisujące wygląd ekranu aplikacji. W tej godzinie poznasz różne rodzaje układów interfejsów użytkownika i dowiesz się, jak ich używać.

Rozpoczynamy pracę z układami interfejsów użytkownika

Pliki układu są wykorzystywane do opisywania tego, jak będzie wyglądał ekran aplikacji (lub jego fragment), lub inaczej mówiąc, jak będzie wyglądał interfejs użytkownika aplikacji. W plikach układu przechowywane są definicje formantów interfejsu użytkownika, takich jak na przykład `EditText` czy `Button`. Poszczególne formanty interfejsu użytkownika mogą odwoływać się do innych zasobów, takich jak ciągi znaków, kolory, wymiary czy komponenty typu *drawable*.

W godzinie 6. „Korzystanie z podstawowych formantów interfejsu użytkownika" dowiedziałeś się, że widoki to najbardziej podstawowe elementy składowe interfejsów użytkownika w systemie Android. Każdy formant, taki jak `Button`, `TextView` czy `Layout`, to rodzaj widoku. Obiekty `ViewGroup` to kontenery dla innych widoków. Kiedy przy użyciu Android Studio umieszczasz w interfejsie użytkownika przycisk `Button`, to tym samym dodajesz przycisk `Button` do układu `RelativeLayout`. W takim przypadku przycisk `Button` jest traktowany jako widok układu `RelativeLayout`.

Oprócz układu `RelativeLayout` istnieje również kilka innych rodzajów układów, takich jak `LinearLayout`, `FrameLayout`, `TableLayout` czy `GridLayout`.

Każda klasa układu jest kontenerem dla innych widoków. W poszczególnych klasach układów obowiązują różne reguły dodawania widoków podrzędnych (formantów). Atrybuty widoków podrzędnych często zależą od tego, w jakim układzie został umieszczony dany widok. Jeśli na przykład korzystasz z układu `RelativeLayout`, to możesz umieszczać przyciski `Button` jeden pod drugim, używając atrybutu `android:layout_below`. Ten atrybut nie jest używany w układach `LinearLayout` czy `FrameLayout`.

Używanie plików XML układów do definiowania widoków dla interfejsu użytkownika jest powszechnie stosowaną praktyką. Widoki możesz również tworzyć w sposób programowy. Układ zawsze spełnia rolę kontenera dla innych widoków, niezależnie od tego, czy został zdefiniowany w pliku XML, czy też został utworzony dynamicznie w czasie działania programu.

Jak się już zapewne zdążyłeś przekonać, pliki układów są przechowywane w odpowiedniej strukturze katalogu */res/layout*.

Uwaga

Zastosowanie znacznika `include` w pliku układu

W plikach XML opisujących układ interfejsu użytkownika możemy używać znacznika `<include />`, pozwalającego na dołączanie innych układów do układu macierzystego. Aby skorzystać z takiego znacznika, powinieneś najpierw utworzyć plik układu pomocniczego, który osadzany jest w innych układach, a następnie w kodzie XML pliku możesz odwołać się do niego tak, jak się odwołujemy do innych zasobów. Na przykład: aby w danym pliku XML odwołać się do pliku układu o nazwie *basicHeader.xml*, powinieneś użyć znacznika `<include layout="@layout/basicHeader"/>`.

Projektowanie układów w Android Studio

Kiedy przy użyciu Android Studio tworzysz nową aktywność, automatycznie tworzony jest powiązany z nią plik układu wykorzystujący kontener `Relative` ↪`Layout`, w którym domyślnie tworzony jest pojedynczy formant `TextView` będący widokiem podrzędnym kontenera `RelativeLayout`.

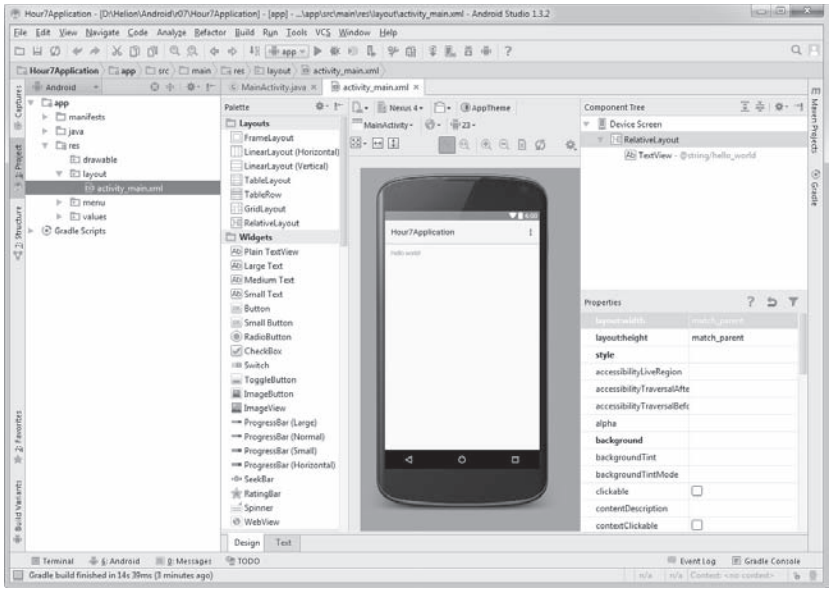
Na listingu 7.1 przedstawiono początkową, domyślną zawartość pliku *activity_* ↪*main.xml*. Zauważ, że nie ma w nim żadnej informacji wskazującej, gdzie formant `TextView` powinien zostać umieszczony. W układzie `RelativeLayout` domyślną pozycją widoku podrzędnego jest lewy górny róg obszaru układu. Na rysunku 7.1 przedstawiono formant `TextView` umieszczony w takiej domyślnej lokalizacji.

Listing 7.1. Plik XML z układem RelativeLayout (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:paddingLeft="@dimen/activity_horizontal_margin"
      android:paddingRight="@dimen/activity_horizontal_margin"
      android:paddingTop="@dimen/activity_vertical_margin"
      android:paddingBottom="@dimen/activity_vertical_margin"
      tools:context=".MainActivity">

   <TextView android:text="@string/hello_world"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

</RelativeLayout>
```



RYSUNEK 7.1. Układ RelativeLayout w Android Studio

W Android Studio możesz edytować pliki w trybie wizualnym (karta *Design*) bądź w trybie tekstowym (karta *Text*). Widoczny na rysunku 7.1 edytor układu pracuje w trybie wizualnym (zwróć uwagę na karty *Design* i *Text* w dolnej części okna).

Edytowanie plików układu bezpośrednio w trybie tekstowym może być bardzo użyteczne. Używając układu `RelativeLayout`, rozmieszczasz widoki podrzędne względem innych widoków lub względem krawędzi kontenera nadrzędnego. Nowy widok może zostać umieszczony powyżej, poniżej, z lewej lub z prawej

strony innego widoku. Często okazuje się, że w takiej sytuacji łatwiej będzie rozmieszczać elementy interfejsu, edytując plik XML bezpośrednio w trybie tekstowym, niż korzystając z edytora wizualnego.

W praktyce zazwyczaj będziemy pracować w trybie mieszanym, przełączając w miarę potrzeb edytor układów Android Studio na przemian do trybu tekstowego i do trybu wizualnego.

Tworzenie układów bezpośrednio w języku XML

W razie potrzeby możesz bezpośrednio edytować pliki XML opisujące układ elementów interfejsu. W miarę jak będziesz nabierał doświadczenia, tworząc kolejne aplikacje, bezpośrednie edytowanie plików XML będzie coraz bardziej naturalnym procesem. Poprzez częste przełączanie Android Studio do widoku XML szybko zorientujesz się, jaki kod XML jest generowany poprzez poszczególne rodzaje formantów. Kiedy zmieniasz właściwości wybranego formantu z poziomu edytora wizualnego, odpowiednie zmiany są automatycznie wprowadzane w pliku XML. Po nabraniu doświadczenia i praktyki w tworzeniu interfejsów użytkownika wprowadzanie odpowiednich zmian w plikach XML opisujących układ stanie się po prostu szybsze i wygodniejsze.

Ponieważ pliki XML możesz edytować jak zwykłe pliki tekstowe, możesz korzystać z takich udogodnień jak wyszukiwanie określonych fraz i znaczników czy szybkie wprowadzanie zmian w odpowiednich miejscach.

Korzystanie z zasobów układu z poziomu kodu programu

W godzinie 6. budowaliśmy interfejs użytkownika składający się z formantów `TextView`, `Button` i `EditText`. Poszczególne widoki były zdefiniowane w pliku *activity_main.xml*, zlokalizowanym w katalogu */res/layout*. W kodzie XML definicje formantów zostały umieszczone wewnątrz elementu `RelativeLayout`.

Aby użyć takiego pliku układu w danej aktywności, musisz wywołać metodę `setContentView()`. Wiersz kodu przedstawiony poniżej jest automatycznie dołączany do kodu aktywności podczas generowania projektu.

```
setContentView(R.layout.activity_main);
```

Aktywność wykorzystuje metodę `setContentView()` do powiązania aktywności z układem. Kiedy chcesz uzyskać dostęp do formantów zdefiniowanych w pliku układu, musisz użyć metody `findViewById()`. Aby utworzyć nowy obiekt klasy `Button` reprezentujący przycisk o identyfikatorze `id`, zdefiniowany w pliku układu, a następnie przypisać go do zmiennej `myButton`, powinieneś użyć następującego wiersza kodu:

```
Button myButton = (Button)findViewById(R.id.button1);
```

Metoda `findViewById()` jest dostępna zarówno z poziomu aktywności, jak i z poziomu dowolnego widoku. Widoki możesz tworzyć dynamicznie z układów XML. Plik układu jest rozwijany do obiektu `View` za pomocą klasy `LayoutInflater`.

Po rozwinięciu układu możesz korzystać z jego formantów. Na listingu zamieszczonym poniżej przedstawiono przykład zastosowania klasy `Layout ↪Inflater` i tworzenia widoku:

```
LayoutInflater inflater = LayoutInflater.from(context);
View exampleView = inflater.inflate(R.layout.example, container, false);
Button myExampleButton = (Button)exampleView.findViewById(R.id.button1);
```

RelativeLayout dla zaawansowanych

Układ `RelativeLayout` zapewnia dużą kontrolę nad rozmieszczaniem widoków podrzędnych oraz elastyczność pozwalające na projektowanie interfejsów użytkownika efektywnie działających na urządzeniach o różnych wielkościach ekranów.

W układzie `RelativeLayout` poszczególne widoki podrzędne są rozmieszczane względem innych widoków lub względem krawędzi kontenera nadrzędnego.

Wyrównywanie do krawędzi kontenera nadrzędnego

Kiedy widok podrzędny jest rozmieszczany w odniesieniu do kontenera nadrzędnego, może być wyśrodkowany, położony względem krawędzi kontenera bądź położony względem początku lub końca kontenera.

Jeżeli widok jest położony względem krawędzi kontenera, możesz używać takich atrybutów jak `layout_alignParentLeft` i `layoutAlignParentStart`. Różnica pomiędzy tymi atrybutami polega na tym, że ten drugi atrybut jest przeznaczony do obsługi języków, w których tekst jest zapisywany od prawej do lewej (ang. *right-to-left languages; RTL languages*). W językach RTL pisanie tekstu rozpoczyna się od prawego marginesu. Jeżeli użyjesz atrybutu `layout ↪AlignParentStart`, widok zostanie umieszczony przy prawej krawędzi ekranu.

Poniżej przedstawiamy zestawienie atrybutów, które są używane do pozycjonowania widoków podrzędnych względem krawędzi kontenera nadrzędnego:

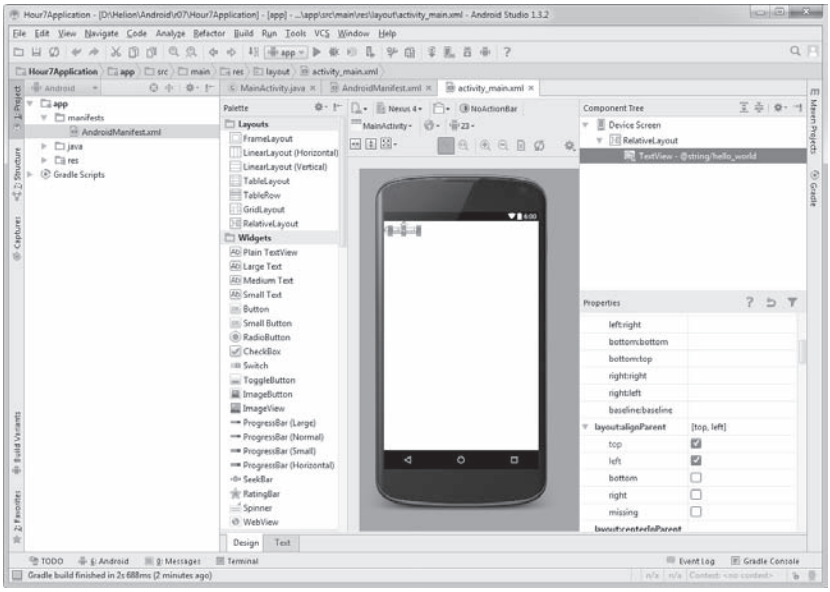
- `android:layout_alignParentStart`
- `android:layout_alignParentEnd`
- `android:layout_alignParentBottom`
- `android:layout_alignParentRight`
- `android:layout_alignParentTop`
- `android:layout_alignParentLeft`

W kolejnym zestawieniu przedstawiamy atrybuty, które są używane do pozycjonowania widoków podrzędnych względem środka kontenera nadrzędnego:

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

Na rysunku 7.2 przedstawiono formant `TextView` wyrównany do lewej, górnej krawędzi kontenera. W pliku układu nasz formant został opisany w następujący sposób:

```
<TextView android:text="@string/hello_world"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_alignParentLeft="true" />
```



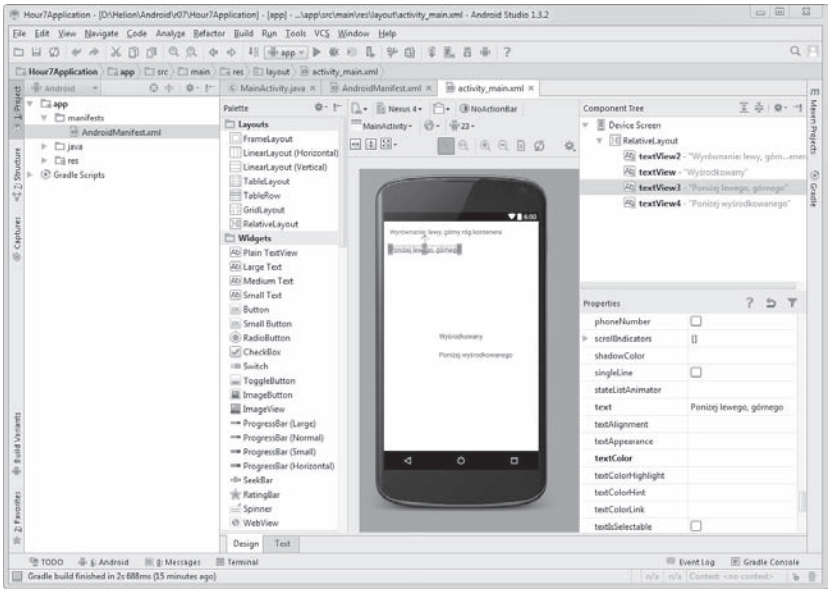
RYSUNEK 7.2. Wyrównywanie formantu TextView do kontenera nadrzędnego

Wyrównywanie widoków względem siebie

Widoki podrzędne mogą być umieszczane powyżej, poniżej, po prawej lub po lewej stronie innych widoków. Na ekranie musi się znajdować co najmniej jeden widok, który będzie spełniał rolę punktu odniesienia. Bardzo często zdarza się, że najpierw jeden bądź więcej widoków zostaje wyrównane względem kontenera, a następnie pozostałe widoki są rozmieszczane względem tych widoków bazowych.

Podobnie jak dzieje się w przypadku wyrównywania względem kontenera nadrzędnego, klasa `RelativeLayout` również posiada atrybuty pozwalające na odpowiednie pozycjonowanie dla języków RTL. Na przykład zamiast atrybutu `android:layout_toLeftOf` możesz w takiej sytuacji używać atrybutu `android:layout_toStartOf`.

Na rysunku 7.3 dwa formanty `TextView` — pierwszy znajdujący się w lewym górnym rogu ekranu oraz drugi zlokalizowany centralnie na środku ekranu — wyrównane są do kontenera nadrzędnego. Pozostałe dwa formanty są rozmieszczone względem tych dwóch formantów bazowych.



RYSUNEK 7.3. Wyrównywanie formantów TextView względem siebie

Zawartość pliku XML odpowiadająca układowi przedstawionemu na rysunku 7.3 wygląda następująco:

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

   <TextView android:text="@string/center" android:
   ↪layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:layout_centerInParent="true"
      android:id="@+id/textViewCenter" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="@string/top_left"
      android:id="@+id/textViewTopLeft"
      android:layout_alignParentTop="true"
      android:layout_alignParentStart="true" />
```

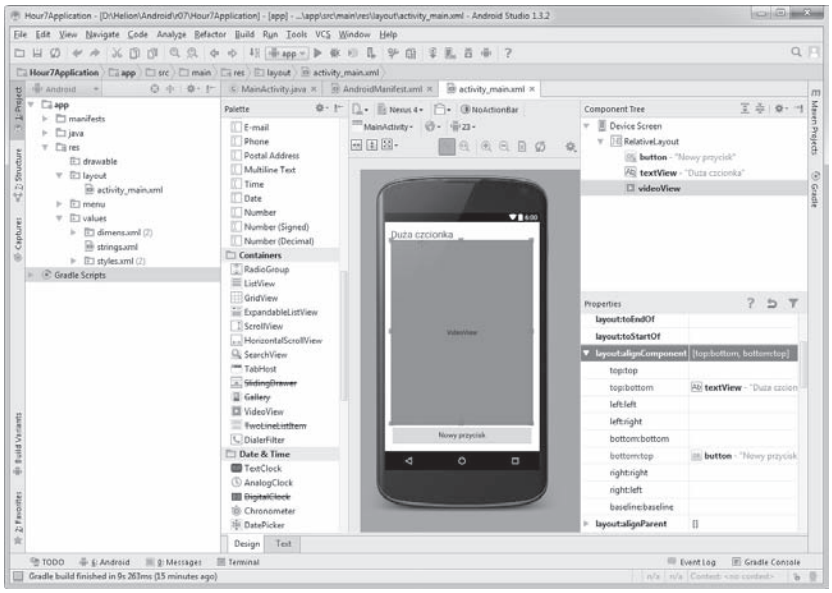
```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/below_top_left"
        android:id="@+id/textViewBelowTopLeft"
        android:layout_below="@+id/textViewTopLeft" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/below_center"
        android:id="@+id/textViewBelowCenter"
        android:layout_below="@+id/textViewCenter"
        android:layout_alignStart="@+id/textViewCenter" />
</RelativeLayout>
```

Jednym ze sposobów wykorzystania układu `RelativeLayout` jest umieszczenie w środku interfejsu widoku, który odwołuje się do czegoś znajdującego się nad nim i pod nim. Na rysunku 7.4 przedstawiono układ interfejsu w Android Studio, w którym znajdują się następujące elementy:

- Przycisk `Button` wyrównany do dolnej krawędzi kontenera:

  `android:layout_alignParentBottom="true"`
- Pole `TextView` umieszczone w lewym górnym rogu.
- Formant `VideoView`, który znajduje się poniżej pola `TextView` i powyżej przycisku `Button`.



RYSUNEK 7.4. Projekt z wykorzystaniem układu RelativeLayout

Aby ustawić pozycję formantu `VideoView`, powinieneś użyć następujących właściwości:

```
android:layout_below="@+id/textView1"
android:layout_above="@+id/loadPhotosButton"
```

Wspólne atrybuty

Niektóre atrybuty są wspólne dla układów i umieszczonych w nich widoków podrzędnych. Na przykład wszystkie układy posiadają właściwości `android:layout_width` i `android:layout_height`, wykorzystywane do kontrolowania odpowiednio szerokości i wysokości elementów. Wartości takich atrybutów mogą być wyrażane zarówno w pikselach logicznych niezależnych od gęstości ekranu (na przykład `20dp`), jak i w postaci predefiniowanych stałych, posiadających określone znaczenie (na przykład `match_parent` czy `wrap_content`).

Stała `match_parent` powoduje, że dany widok zostanie przeskalowany do rozmiarów układu nadrzędnego, a stała `wrap_content` powoduje, że rozmiary widoku zostają dostosowane do zawartości elementu.

Marginesy układów

Marginesy układów określają odległość pomiędzy widokiem podrzędnym a krawędzią kontenera nadrzędnego. Aby przesunąć przycisk w dół i w prawo, musisz dla niego określić margines od górnej i od lewej krawędzi układu. Na przykład dodając do definicji przycisku w pliku układu dwa wiersze zamieszczone poniżej, przesuniesz go o 40 pikseli w dół i 120 pikseli w prawo.

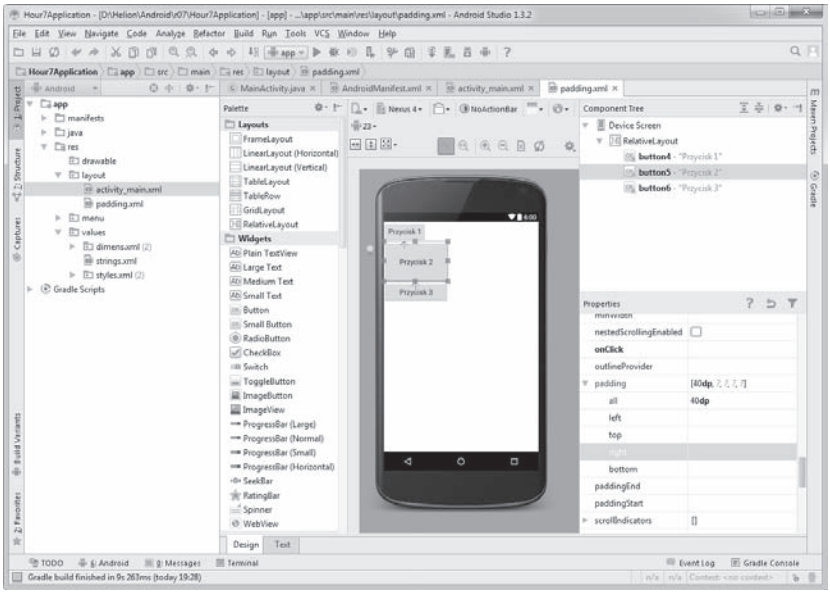
```
android:layout_marginTop="40dp"
android:layout_marginLeft="120dp"
```

Właściwość padding

Padding to ilość miejsca dodawanego z wybranej strony formantu po to, aby przesunąć jego zawartość. Właściwość `padding` odnosi się do wewnętrznych marginesów formantu. W przypadku przycisków `Button` będzie to dodatkowa przestrzeń dodawana pomiędzy tekstem etykiety a krawędziami przycisku. Padding możesz ustawić dla całego formantu lub dla każdej krawędzi osobno.

Na rysunku 7.5 przedstawiono nowy układ XML z trzema przyciskami, ilustrujący zastosowanie paddingu.

Dla pierwszego przycisku właściwość `padding` nie została ustawiona. Drugi przycisk ma właściwość `padding` ustawioną na wartość `40dp` ze wszystkich stron. Możesz to zrobić za pomocą polecenia `android:padding="40dp"`. Dzięki takiemu poleceniu z każdej strony przycisku zostanie dodane 40 pikseli dp. Trzeci przycisk ma właściwość `padding` ustawioną na 40dp z lewej i z prawej strony. Prawy padding został ustawiony za pomocą polecenia `android:paddingRight="40dp"`, a lewy padding za pomocą polecenia `android:paddingLeft="40dp"`.



RYSUNEK 7.5. Zastosowanie paddingu

Inne rodzaje układów

Powszechnie stosowanym rozwiązaniem jest zawieranie całego ekranu aplikacji w jednym, dużym układzie nadrzędnym. Najczęściej wykorzystywane są układy `RelativeLayout` i `LinearLayout`. W tabeli 7.1 przedstawiono zestawienie najczęściej wykorzystywanych układów.

TABELA 7.1. Najpopularniejsze formanty układów

Nazwa formantu układu	Opis	Najważniejsze atrybuty
`LinearLayout`	Układ liniowy, w którym każdy kolejny widok podrzędny jest umieszczany za poprzednim, w jednym wierszu bądź w jednej kolumnie.	Orientacja (pionowa lub pozioma).
`RelativeLayout`	Układ względny, w którym każdy kolejny widok podrzędny jest pozycjonowany względem innych widoków lub względem krawędzi układu nadrzędnego.	Wiele atrybutów wyrównania odpowiadających za położenie widoku podrzędnego względem innych widoków.

TABELA 7.1. Najpopularniejsze formanty układów — ciąg dalszy

Nazwa formantu układu	Opis	Najważniejsze atrybuty
`FrameLayout`	Układ ramowy, w którym kolejne widoki podrzędne są ustawiane jeden na drugim i tworzą stos.	Kolejność umieszczania widoków podrzędnych jest bardzo istotna. Poszczególne widoki mogą się na siebie nakładać.
`TableLayout`	Układ tabeli, w którym poszczególne widoki podrzędne są rozmieszczane w wierszach i kolumnach (tak jak w tabeli).	Każdy wiersz wymaga elementu `TableRow`.
`GridLayout`	Układ rastrowy, w którym poszczególne widoki podrzędne są umieszczane w prostokątnych obszarach tworzących raster.	Wiersze i kolumny z atrybutami `RowSpec` i `ColumnSpec`.

Zastosowanie układu LinearLayout

W układach `LinearLayout` widoki rozmieszczane są w wierszach lub kolumnach. Kierunek rozmieszczania widoków jest określany za pomocą atrybutu `orientation`. Jeżeli korzystamy z orientacji pionowej, widoki podrzędne są rozmieszczane w kolumnie jeden pod drugim. Jeżeli korzystamy z orientacji poziomej, widoki podrzędne rozmieszczane są w wierszu obok siebie. Choć układy posiadają bardzo wiele wspólnych atrybutów, najważniejsze różnice między nimi polegają na kolejności i regułach rozmieszczania formantów na ekranie. Atrybut `orientation` jest unikatowy dla układu `LinearLayout`:

```
android:orientation="vertical"
```

Właściwość gravity

Widoki podrzędne posiadają właściwość `layout_gravity`, która może przyjmować wartości takie jak `left`, `right`, `center` i tak dalej:

```
android:layout_gravity="right"
```

Właściwość `gravity` odnosi się do widoków podrzędnych, ale może być także używana dla całych układów. Na przykład poprzez ustawienie właściwości `gravity` układu `LinearLayout` na wartość `right` spowodujemy, że wszystkie widoki podrzędne zostaną przesunięte do prawej krawędzi układu:

```
android:gravity="right"
```

Właściwość weight

Właściwość `layout_weight` daje więcej miejsca na ekranie widokom posiadającym większą „wagę". Znaczenie tej właściwości można najłatwiej zauważyć, tworząc układ `LinearLayout` w orientacji poziomej.

▼ Spróbuj sam!

Ustawianie wagi formantów w układach

Aby dowiedzieć się, jak działa atrybut `layout_weight` reprezentujący „wagę" widoku i jak należy używać go w swoich projektach, powinieneś wykonać polecenia przedstawione poniżej:

1. W folderze */res/layout* utwórz nowy plik XML układu i nadaj mu nazwę *weight_example.xml*.
2. Użyj układu `LinearLayout` z orientacją poziomą.
3. Dodaj dwa przyciski do układu. Szerokość obu przycisków (atrybut `width`) powinna być ustawiona na stałą `match_parent`.
4. Dla obu przycisków ustaw atrybut `layout_weight` na wartość `1`. Oba przyciski powinny mieć takie same rozmiary i zajmować całą szerokość kontenera.
5. Zmień wagę jednego z przycisków i sprawdź, jaki będzie to miało wpływ na wygląd przycisków w Android Studio.

Układ FrameLayout

Układ `FrameLayout` to kontener, który nie posiada żadnych specjalnych reguł rozmieszczania widoków. Domyślnie wszystkie widoki są rysowane w lewym górnym rogu ekranu. Takie rozwiązanie może się wydawać niezbyt użyteczne, ale w praktyce dzięki zastosowaniu odpowiednich ustawień marginesów pozwala na umieszczanie formantów w dowolnym miejscu. Układ `FrameLayout` jest przydatny w sytuacji, kiedy dwa formanty powinny być umieszczone w tym samym miejscu jeden na drugim. Przykładem może być sytuacja, gdy umieszczamy pole `TextView` na elemencie graficznym lub kiedy używamy kilku nałożonych na siebie obrazów do osiągnięcia efektu cienia i podświetlenia.

Dzięki zastosowaniu układu `FrameLayout` i odpowiednio dobranych atrybutów `layout_margins` możemy tworzyć bardzo efektowne i użyteczne interfejsy użytkownika.

Jeżeli korzystasz z układów `LinearLayout` lub `RelativeLayout`, to system Android podczas renderowania interfejsu użytkownika uwzględnia fizyczne rozmiary ekranu. Ponieważ w takich układach poszczególne widoki są rozmieszczone względem siebie bądź względem krawędzi kontenera, system ma większe możliwości optymalizacji rozłożenia elementów i zwiększenia funkcjonalności interfejsu. Z tego powodu używanie układów `LinearLayout` i `RelativeLayout` jest mocno zalecane. W praktyce w zdecydowanej większości przypadków najbardziej rekomendowanym rozwiązaniem będzie użycie układu `RelativeLayout`.

Podsumowanie

W tej godzinie dowiedziałeś się, czym są układy graficzne interfejsów użytkownika. Bardziej szczegółowo omówiliśmy układy `LinearLayout` i `Relative ↪Layout`, dzięki czemu miałeś możliwość bliżej zapoznać się z zasadami ich działania. Układy XML, pozwalające na definiowanie i rozmieszczanie widoków podrzędnych na ekranie, są wykorzystywane do tworzenia interfejsów użytkownika. Korzystając z atrybutów takich jak `padding`, `margins` i innych, masz pełną kontrolę nad wyglądem interfejsu użytkownika tworzonej aplikacji.

Pytania i odpowiedzi

P: Czym jest orientacja w układzie `LinearLayout`?

O: W układzie LinearLayout atrybut orientacji jest odpowiedzialny za kierunek rozmieszczania widoków. Orientacja pionowa powoduje, że poszczególne widoki są układane w kolumnie, podczas gdy orientacja pozioma tworzy wiersz. Atrybut orientacji nie jest wykorzystywany w układach `FrameLayout` ani `RelativeLayout`. Poszczególne układy mają różne reguły rozmieszczania widoków podrzędnych, co oznacza, że ich zestawy atrybutów również mogą się od siebie różnić.

P: Jaka jest najważniejsza różnica pomiędzy układami `FrameLayout` i `RelativeLayout`?

O: W układzie `RelativeLayout` poszczególne formanty są rozmieszczane względem siebie, czego nie możesz zrobić, korzystając z układu `FrameLayout`.

Warsztaty

Quiz

1. Czym różnią się od siebie marginesy układu i padding?
2. Jaki jest związek między układem `LinearLayout` a klasą `ViewGroup`?
3. W jaki sposób możesz umieścić przycisk `Button` 100 pikseli od górnej krawędzi ekranu urządzenia? Skorzystaj z pikseli `dp` (piksele niezależne od gęstości wyświetlacza).

Odpowiedzi

1. Marginesy układu definiują dystans pomiędzy komponentem interfejsu a krawędzią układu. Padding ma bezpośredni wpływ na rozmiar komponentu. W zależności od rodzaju komponentu ustawienia paddingu mogą mieć różne efekty.
2. Układ `LinearLayout` jest rozszerzeniem klasy `ViewGroup`. Klasa `ViewGroup` w systemie Android to kontener, który zawiera inne widoki.

3. Aby ustawić górny margines na 100 pikseli dp, powinieneś użyć następującego ustawienia:

```
android:layout_marginTop="100 dp"
```

Ćwiczenia

1. Korzystając z układu `FrameLayout`, zaimplementuj interfejs użytkownika z przyciskami `Button` znajdującymi się w każdym z narożników ekranu. Spróbuj zrobić to samo z użyciem układu `RelativeLayout`.

   Aby to zrobić z użyciem układu `FrameLayout`, powinieneś wykonać polecenia przedstawione poniżej:

   - Utwórz plik układu `FrameLayout`.
   - Utwórz cztery przyciski `Button`.
   - Ustaw atrybuty `topMargin` i `leftMargin` poszczególnych przycisków `Button` tak, aby rozmieścić je w narożnikach ekranu.

2. Spróbuj umieścić jeden układ wewnątrz drugiego. Aby to zrobić, utwórz układ `LinearLayout` z orientacją pionową. Dodaj do niego układ `FrameLayout` i przycisk `Button`, a następnie spróbuj dodać formant `ImageView` do układu `FrameLayout`.

3. Utwórz układ `RelativeLayout`, dodaj do niego formanty `ImageView` oraz `TextView`, a następnie dołącz ten układ do innego układu za pomocą znacznika `include`.

Skorowidz

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

R

S

Ś

T

U

W

Z

Ż

Perspektywy rozwoju Androida są wyjątkowo obiecujące. System ten przebojem zdobywa nowe obszary rynku, a jego zalety, takie jak modułowa architektura, wysoka elastyczność czy otwarty charakter, są cenione zarówno przez programistów, jak i użytkowników aplikacji. Rocznie sprzedaje się miliardy urządzeń pracujących pod kontrolą Androida, a jeśli wziąć pod uwagę rozwój technologii i mnożące się pomysły na wyposażenie w procesor kolejnego przedmiotu codziennego użytku, można przepowiadać temu systemowi świetlaną przyszłość. Podobnie zresztą jak programistom, którzy nauczą się efektywnie pisać świetne, atrakcyjne aplikacje dla Androida.

Książka, którą trzymasz w dłoni, została pomyślana jako podręcznik. Nawet jeśli jesteś bardzo początkującym programistą, błyskawicznie — bo po lekturze 24 rozdziałów, z których każdy zajmie najwyżej godzinę — poznasz podstawy programowania dla Androida i szybko zaczniesz tworzyć w pełni funkcjonalne aplikacje. Najpierw dowiesz się, czym są aktywności, intencje, usługi i odbiorniki rozgłoszeń. Później zagłębisz się w bardziej zaawansowane zagadnienia: nauczysz się obsługi multimediów, poznasz interfejs Material Design firmy Google, sprawdzisz, jak wykorzystywać bazy danych SQLite. Wisienką na tym torcie atrakcji będzie pokazanie szczególnych możliwości systemu Android, wykorzystywanych w bardzo profesjonalnych aplikacjach.

Szczególnie ciekawe dla Ciebie będą:

- **omówienie najnowszych możliwości i mechanizmów systemu Android 5 (Lollipop), pozwalających na tworzenie aplikacji dla Androida TV i urządzeń typu Android Wear**
- **przedstawienie świetnego narzędzia dla programistów — środowiska Android Studio**
- **nauka projektowania responsywnych aplikacji, wykorzystujących zadania działające w tle**
- **wskazówki dotyczące tworzenia wyrafinowanych systemów nawigowania w aplikacji, korzystania z pasków ActionBar oraz z wysuwanych menu**
- **sposoby pobierania danych z chmury i parsowania danych zapisanych w formacie JSON**
- **użycie Google Play Services do lokalizowania położenia urządzenia**

Carmen Delessio — doświadczony deweloper aplikacji. Pracował jako programista, architekt rozwiązań i CTO w dużych i małych firmach. Jest autorem wielu nagradzanych aplikacji dla systemu Android.

Lauren Darcey — uznany autorytet w dziedzinie architektury aplikacji i projektowania komercyjnych aplikacji mobilnych.

Shane Conder — twórca wielu popularnych aplikacji dla systemów Android, iOS, BREW, BlackBerry, J2ME, Palm i Windows Mobile.

Kreatywnych ogranicza tylko wyobraźnia!

Helion

SAMS

42134 numer katalogowy

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-283-2150-2

9 788328 321502

Informatyka w najlepszym wydaniu

cena: 79,00 zł