

## Zagadnienia

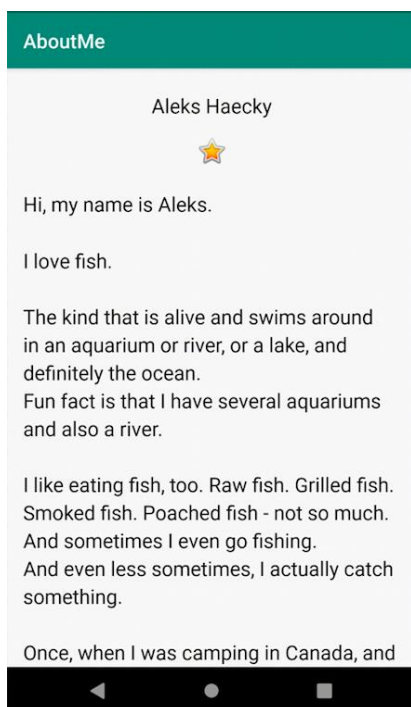
- Jak pracować z View i ViewGroup.
- Jak zorganizować widoki w działaniu, używając LinearLayout.
- Jak używać ScrollView do wyświetlania przewijalnej zawartości.
- Jak zmienić widoczność Widoku.
- Jak tworzyć i wykorzystywać zasoby łańcuchów i wymiarów.
- Jak utworzyć układ liniowy za pomocą Edytora układów w Android

## Do zrobienia

- Utwórz aplikację AboutMe.
- Dodaj TextView do układu, aby wyświetlić swoje imię.
- Dodaj ImageView.
- Dodaj ScrollView, aby wyświetlić przewijalny tekst.

W aplikacji AboutMe możesz zaprezentować interesujące fakty na swój temat lub dostosować aplikację do potrzeb znajomego, członka rodziny lub zwierzaka.

Aplikacja wyświetla nazwę, przycisk GOTOWE, gwiazdkę i przewijalny tekst.

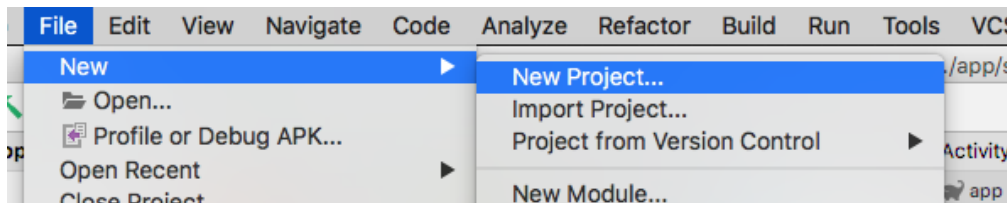


## 3. Zadanie: Utwórz projekt AboutMe

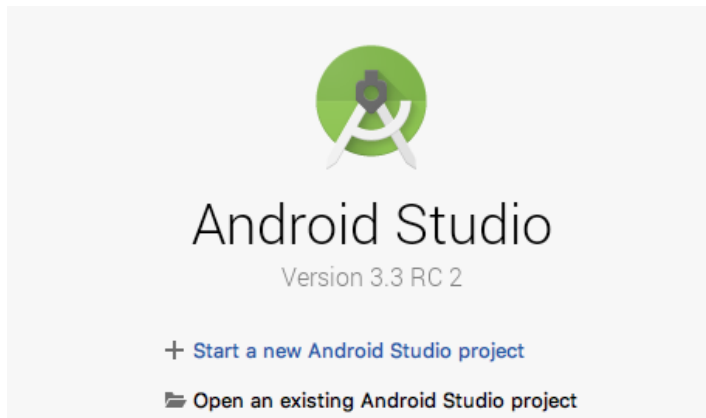
Tworzymy projekt AboutMe w Android Studio..

1. Otwórz Android Studio, jeśli nie jest jeszcze otwarty.

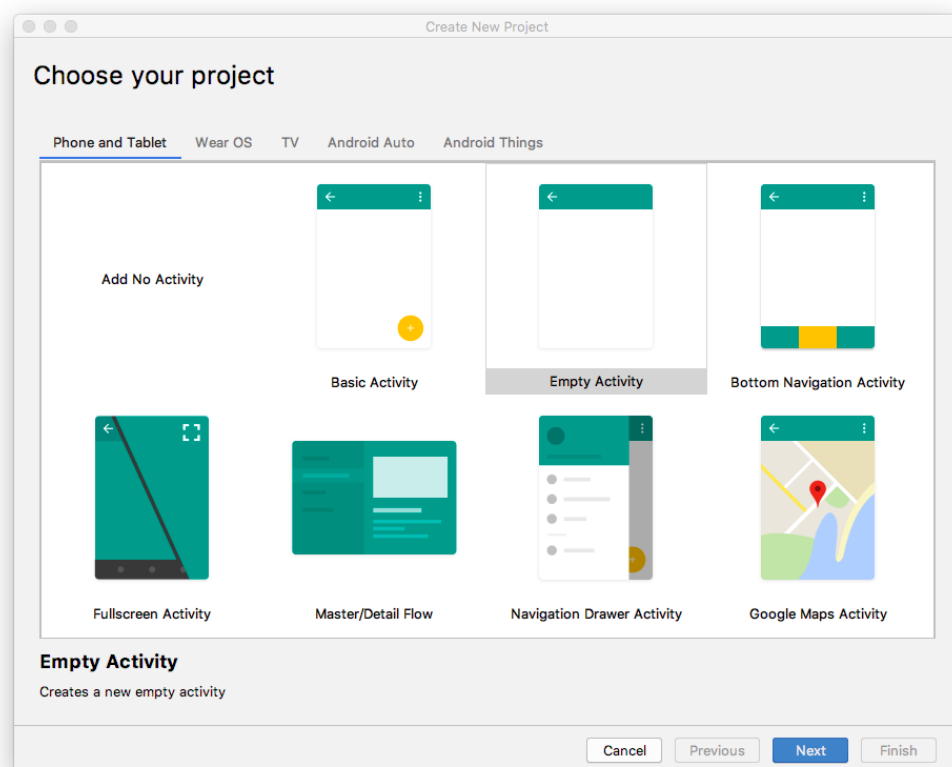
2. Nowy projekt jeśli jakiś jest otwarty **File > New > New Project**.



3. Jeśli nie + **Start a new Android Studio project** w **Welcome to Android Studio**



4. W oknie **Create New Project** w zakładce **Phone and Tablet** wybieramy **Empty Activity** template. Klikamy **Next**.



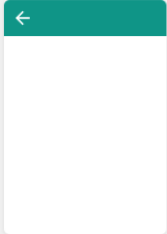
5. Dalej w **Create New Project** konfigurujemy projekt, klikamy **Finish**.





Create New Project

### Configure your project



Empty Activity

Creates a new empty activity

Name  
AboutMe

Package name  
android.example.com.aboutme

Save location  
/Users/jocelyn/AndroidStudioProjects/AboutMe

Language  
Kotlin

Minimum API level  
API 19: Android 4.4 (KitKat)

**i** Your app will run on approximately **95.3%** of devices.  
[Help me choose](#)

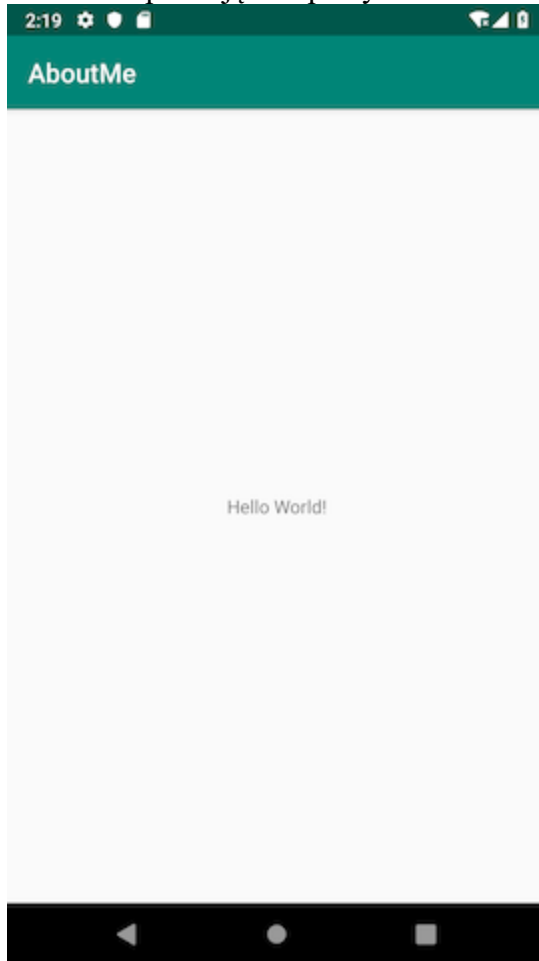
☐ This project will support instant apps

☒ Use AndroidX artifacts

Cancel Previous Next Finish

Czekamy na wygenerowanie projektu

6. Uruchom aplikację. Na pustym ekranie zobaczysz ciąg „Hello World”.



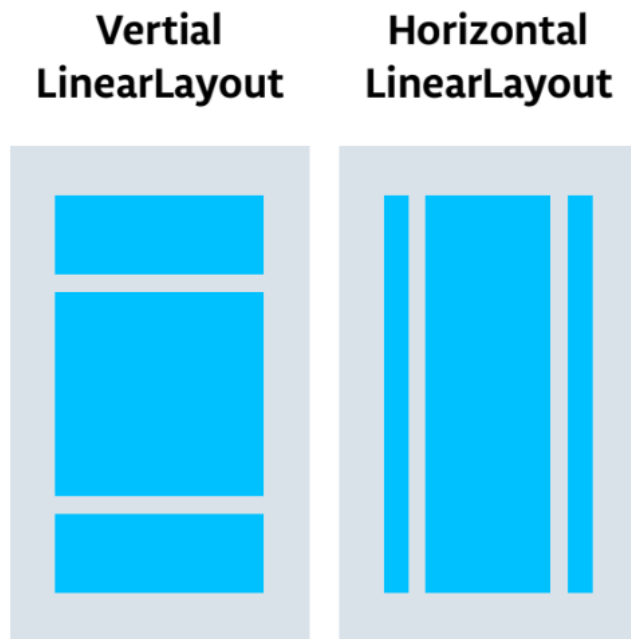
Szablon Pustej aktywności tworzy pojedyncze puste działanie, `Mainactivity.kt`. Szablon tworzy również plik układu o nazwie `activity_main.xml`. Plik układu ma `ConstraintLayout` jako swoją główną grupę `ViewGroup` i ma pojedynczy `TextView` jako swoją zawartość..

## **4. Zadanie: Zadanie: zmień układ główny, aby użyć LinearLayout**

### **View groups**

Grupa widoków jest widokiem, który może zawierać widoki potomne, które są innymi widokami i grupami widoków. Widoki tworzące układ są zorganizowane w hierarchię widoków z grupą widoków jako katalogiem głównym.

W grupie widoków `LinearLayout` elementy interfejsu są ułożone poziomo lub pionowo.



Zmień układ główny, aby korzystał z grupy widoków `LinearLayout`:

1. **Project > Android** w folderze `app/res/layout` otwórz `activity_main.xml`.
2. Wybierz **Text** tab, zmień `ConstraintLayout` to `LinearLayout`.
3. Usuń `TextView`.
4. `android:orientation` ustaw na `vertical`.

przed:

```
<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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

po:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```

xmlns:tools="http://schemas.android.com/tools"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:context=".MainActivity">

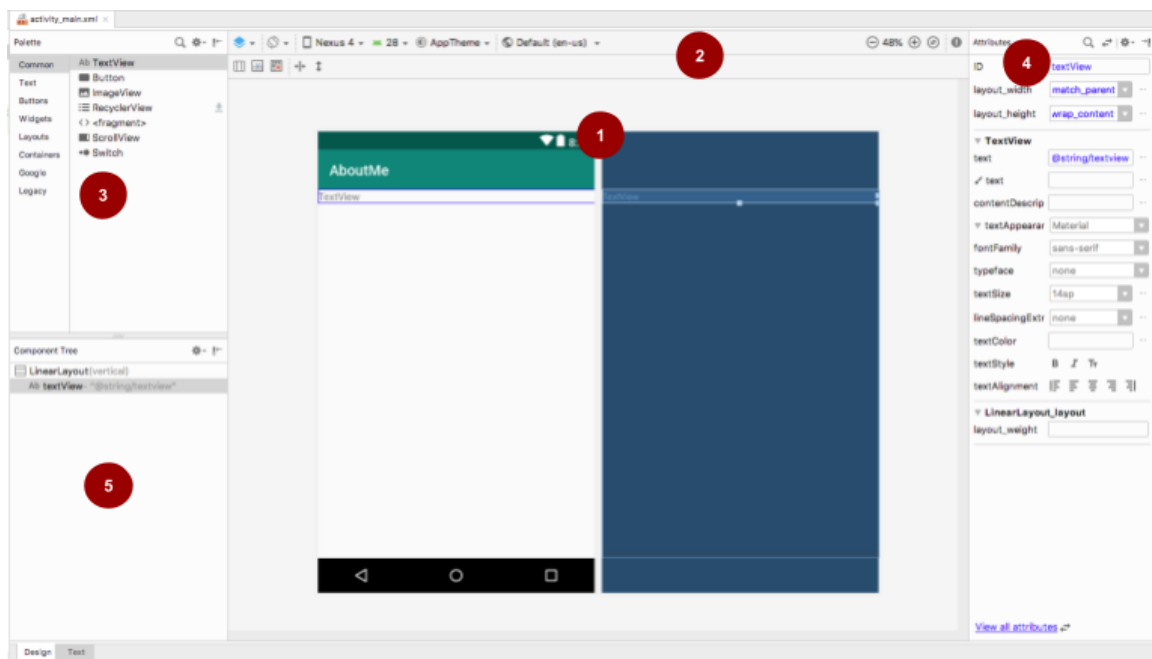
```

</LinearLayout>

## 5. Zadanie: Dodaj TextView używając Layout Editor

Edytor układu to narzędzie do projektowania wizualnego w Android Studio. Zamiast pisać kod XML ręcznie, aby zbudować układ aplikacji, możesz użyć Edytora układów, aby przeciągnąć elementy interfejsu użytkownika do edytora projektu.

Aby wyświetlić Edytor układu, kliknij kartę Projekt. Poniższy zrzut ekranu pokazuje części Edytora układu.



**1 Design editor:** Wyświetla wizualną reprezentację układu ekranu w widoku projektu, widoku planu lub w obu. Edytor projektu jest główną częścią edytora układu..


**2 Toolbar:** Zawiera przyciski do konfigurowania wyglądu układu w edytorze projektu i do zmiany niektórych atrybutów układu. Na przykład, aby zmienić sposób wyświetlania

układu w edytorze projektu, użyj **Select Design Surface**  drop-down menu:

- Use **Design** aby uzyskać rzeczywisty podgląd swojego układu.
- Use **Blueprint** aby zobaczyć tylko kontury dla każdego widoku.

- Use **Design** + **Blueprint** aby zobaczyć oba wyświetlacze obok siebie.


**3 Palette:** Zawiera listę widoków i grup widoków, które można przeciągnąć do swojego układu lub do panelu **Component Tree**.

**4 Attributes:** pokazuje atrybuty dla aktualnie wybranego widoku lub grupy widoków. Aby przełączać się między pełną listą atrybutów a często używanymi atrybutami, użyj ikony  u góry panelu.

**5 Component Tree:** wyświetla hierarchię układu jako drzewo widoków. Drzewo komponentów jest przydatne, gdy masz małe, ukryte lub nakładające się widoki, których inaczej nie można byłoby wybrać w edytorze projektu.

## Krok 1: Dodaj TextView

1. Otwórz `res/layout/activity_main.xml`.

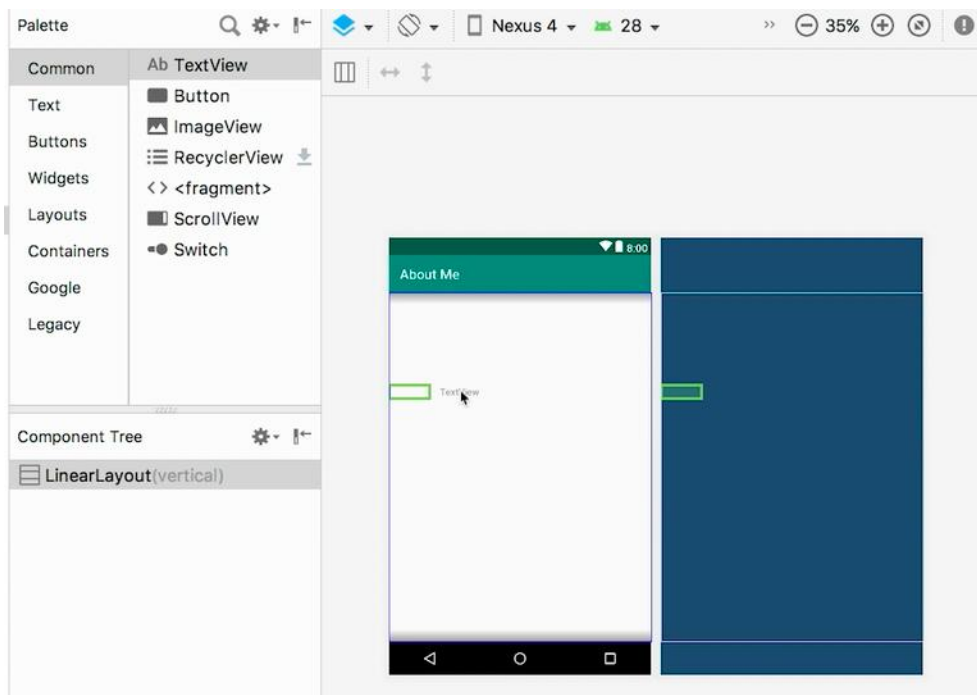
2. Przejdź do zakładki Tekst  i sprawdź kod. Kod ma LinearLayout jako swoją główną grupę widoków. (Grupy widoków to widoki zawierające inne widoki).

LinearLayout ma wymagane atrybuty `layout_height`, `layout_width` i orientacja, która jest domyślnie pionowa.

3. Przejdź do karty **Design** aby otworzyć Layout Editor.

**Note:** Uwaga: Karta **Design** i karta **Text** pokazują ten sam Layout, tylko w inny sposób. Zmiany dokonane w jednej karcie są odzwierciedlone w drugiej.

4. Przeciągnij widok tekstu z panelu **Palette** do edytora projektu



5. Zwróć uwagę na panel **Component Tree** Nowy widok tekstu jest umieszczony jako element potomny nadrzędnej grupy widoków, którą jest `LinearLayout`.




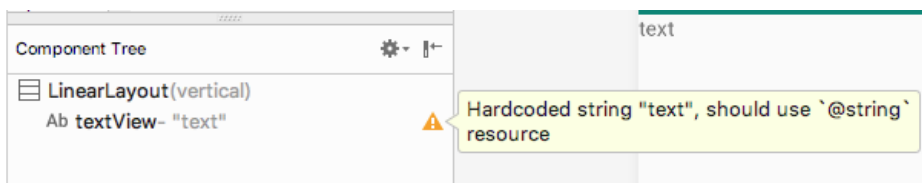
6. Otwórz panel **Attributes**. Aby otworzyć panel, kliknij dwukrotnie nowo dodany TextView w edytorze projektu.)
7. Ustaw następujące atrybuty w panelu **Attributes**:

Attribute	Value
ID	name_text
text	Ustaw na swoje imię. (Jedno z pól tekstowych pokazuje ikonę klucza wskazującą, że jest to <code>tools</code> namespace. Jedno bez klucza dotyczy <code>android</code> namespace— to jest pole tekstowe, które zmieniamy)
textAppearance > textSize	20sp
textAppearance > textColor	@android:color/black
textAppearance > textAlignment	Center

## Krok 2: Utwórz string resource



W **Component Tree**, obok TextView, zobaczysz ikonę ostrzeżenia. . Aby zobaczyć tekst ostrzeżenia, kliknij ikonę lub wskaż ją, jak pokazano na zrzucie ekranu poniżej.

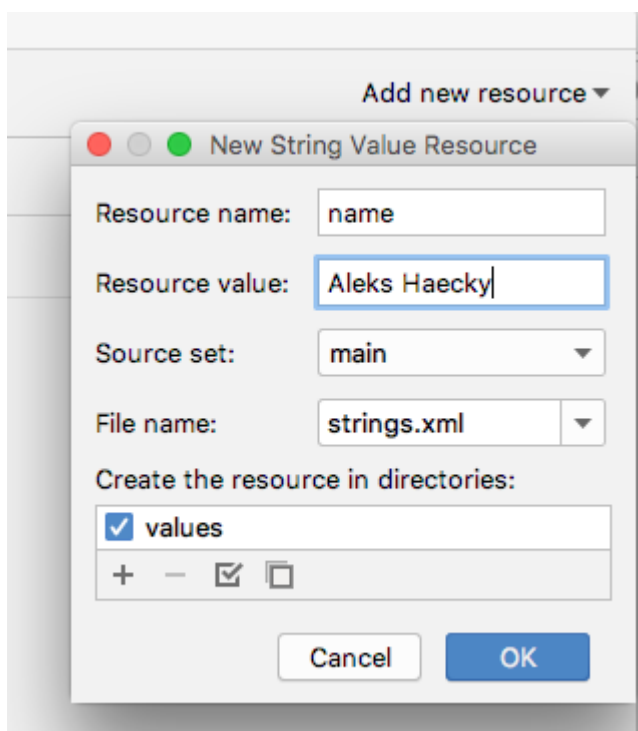


Aby usunąć ostrzeżenie, utwórz zasób:

1. W panelu **Attributes** kliknij trzy kropki obok atrybutu tekstowego ustawionego na swoje imię. Zostanie otwarty edytor zasobów.



2. W oknie **Resources** wybierz **Add new resource > New string Value**.
3. W oknie **New String Value Resource** Ustaw pole **Resource name** na `name`. Ustaw **Resource value**. Kliknij **OK**.



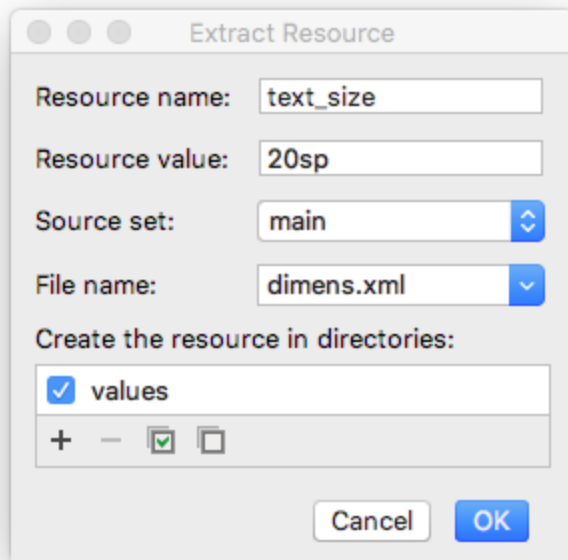
4. `res/values/strings.xml` w zasobach pojawił się nowy string `name`.

```
<string name="name">Aleks Haecky</string>
```

### Krok 3: Utwórz zasób wymiaru (dimension resource)

Właśnie dodałeś zasób za pomocą edytora zasobów. Możesz także wyodrębnić zasoby w edytorze kodu XML, aby utworzyć nowe zasoby:

1. W `activity_main.xml` przejdź do **Text** tab.
2. W lini `textSize` kliknij liczbę (20sp) i naciśnij `Alt+Enter`. Wybierz **Extract dimension resource** z wyskakującego menu.
3. W oknie **Extract Resource**, wprowadź `text_size` w polu **Resource name**.
4. Zatwierdź **OK**.



5. Otwórz plik `res/values/dimens.xml`:

```
<dimen name="text_size">20sp</dimen>
```

**Uwaga:** jeśli plik `dimens.xml` nie był już obecny w folderze `res / values`, Android Studio go utworzy.

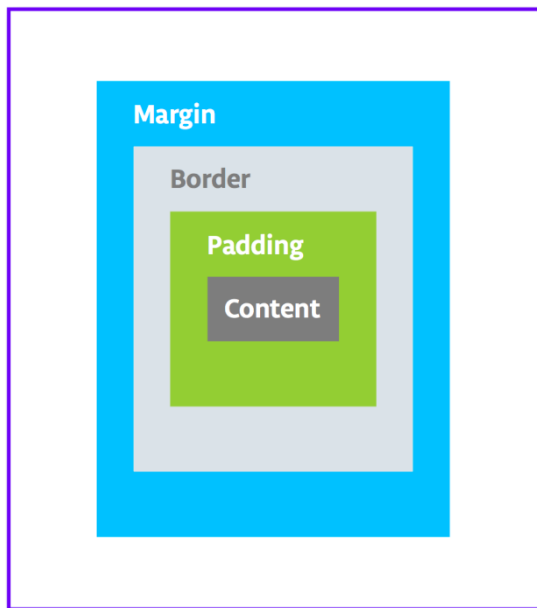
5. Uruchom aplikację

## 6. Zadanie: nadawanie stylu `TextView`

Kiedy patrzysz na ekran aplikacji, twoje imię jest dociśnięte do górnej części ekranu. Teraz powinniśmy dodać odpowiednie padding and a margin

### Padding versus margin

[Padding](#) to przestrzeń wewnątrz granic widoku lub elementu. Jest to przestrzeń między krawędziami widoku i jego treścią, jak pokazano na poniższym rysunku.



Rozmiar widoku obejmuje jego padding. Najczęściej używane „atrybuty wypełniania” padding attributes to:

- [android:padding](#) określa wypełnienie dla wszystkich czterech krawędzi widoku.
- [android:paddingTop](#) określa wypełnienie górnej krawędzi.
- [android:paddingBottom](#) określa wypełnienie dolnej krawędzi.
- [android:paddingStart](#) specifies padding for the "starting" edge of the view.
- [android:paddingEnd](#) specifies padding for the "ending" edge of the view.
- [android:paddingLeft](#) specifies padding for the left edge.
- [android:paddingRight](#) specifies padding for the right edge.

Margin to przestrzeń dodana poza granicami widoku. Jest to przestrzeń od krawędzi widoku do jego obiektu nadrzędnego, jak pokazano na powyższym rysunku. Najczęściej używane są atrybuty marginesów:

- [android:layout\\_margin](#) specifies a margin for all four sides of the view.
- [android:layout\\_marginBottom](#) specifies space outside the bottom side of this view.
- [android:layout\\_marginStart](#) specifies space outside the "starting" side of this view.
- [android:layout\\_marginEnd](#) specifies space on the end side of this view.
- [android:layout\\_marginLeft](#) specifies space on the left side of this view.
- [android:layout\\_marginRight](#) specifies space on the right side of this view.

### Right/left versus start/end

"Right" and "left" zawsze odnoszą się do prawej i lewej strony ekranu, niezależnie od tego, czy aplikacja używa przepływu od lewej do prawej (LTR), czy od prawej do lewej (RTL).

"Start" and "end" zawsze odnoszą się do początku i końca przepływu::

- W przypadku przepływu LTR początek = w lewo i koniec = w prawo.
- W przypadku przepływu RTL początek = prawo i koniec = lewo.

Jeśli API level jest minimum 17 (Android 4.2) :

- Używamy "start" i "end" zamiast "left" i "right".
- Na przykład, `android:layout_marginLeft` trzeba zamienić na `android:layout_marginStart` aby obsługiwać języki RTL.

Jeśli nie używamy jednocześnie `android:paddingLeft` i `android:paddingStart`.

## Step 1: Dodawanie padding

Aby wstawić spację między swoim nazwiskiem a górną krawędzią text view, dodajemy top padding.

1. Otwórz plik `activity_main.xml` w zakładce **Design**.
2. W **Component Tree** lub w edytorze projektu kliknij widok tekstu, aby otworzyć panel **Attributes**.



3. W górnej części okienka **Attributes** kliknij ikonę podwójnej strzałki, aby zobaczyć wszystkie dostępne atrybuty.
4. Wyszukaj **Padding**, rozwiń go i kliknij trzy kropki ... obok **top** atrybutu. Pojawi się okno dialogowe **Resources**.
5. W oknie dialogowym **Resources** wybierz **Add new resource > New dimen Value**.
6. W oknie dialogowym **New Dimension Value Resource** utwórz nowy zasób dimen o nazwie `small_padding` o wartości 8dp.

Skrót dp oznacza niezależny od gęstości. Jeśli chcesz, aby element interfejsu wyglądał tak samo na ekranach o różnych gęstościach, użyj dp jako jednostki miary. Jednak przy określaniu rozmiaru tekstu zawsze używaj sp (skalowalne piksele).

Kliknij **OK**.

## Krok 2: Dodaj margines

Aby odsunąć nazwę text view od krawędzi elementu nadrzędnego, dodaj górny margines.

1. W okienku **Attributes** wyszukaj "margin" aby znaleźć **Layout\_Margin**.
2. Rozwiń **Layout\_Margin**, i kliknij trzy kropki ... obok atrybutu **top**.
3. Utwórz nowy zasób dimen o nazwie `layout_margin` i ustaw go na 16dp. Kliknij **OK**.

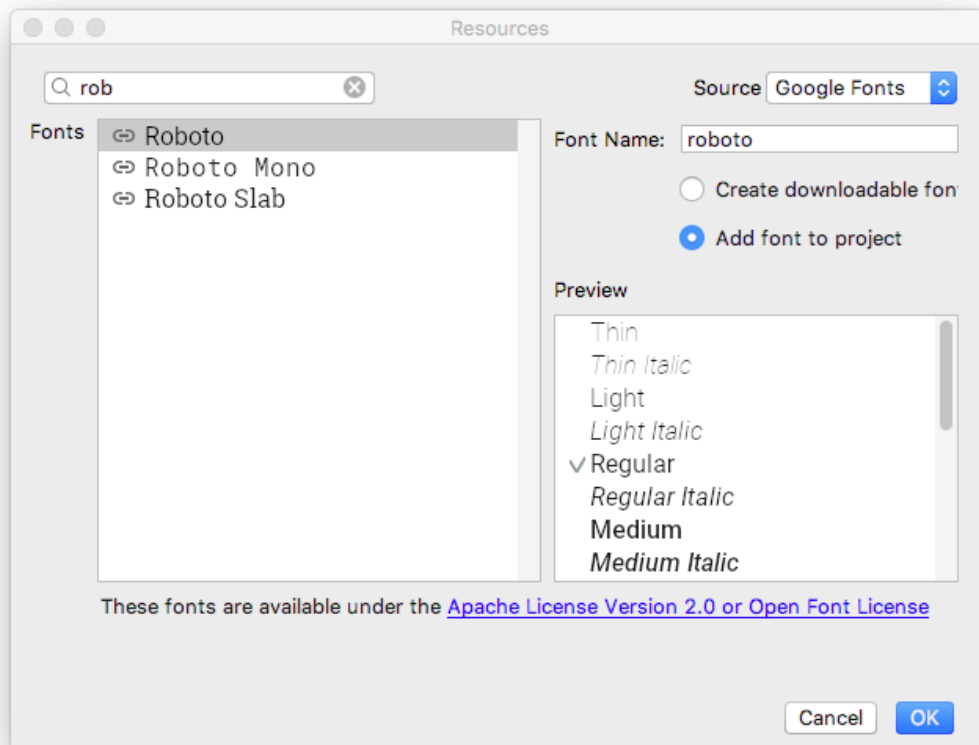
## Krok 3: Dodaj czcionkę

Aby widok tekstu nazwy wyglądał lepiej, użyj czcionki Android Roboto. Ta czcionka jest częścią biblioteki pomocniczej i dodajesz ją jako zasób.

1. W panelu **Attributes** wyszukaj „fontFamily”.
2. W polu **fontFamily** kliknij strzałkę listy rozwijanej, przewiń na dół listy i wybierz opcję **More Fonts**.
3. W oknie dialogowym **Resources** wyszukaj `rob` i wybierz **Roboto**. Z listy **Preview** wybierz

opcję **Regular**.

4. Wybierz przycisk **Add font to project**. Kliknij OK



Folder `res` ma teraz folder `font` zawierający plik czcionek `roboto.ttf`. Atrybut `@font/roboto` jest dodawany do `TextView`.

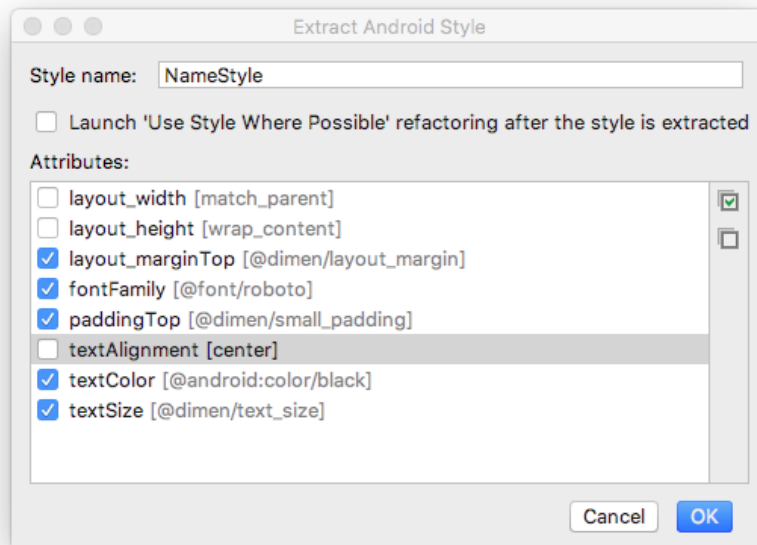
## Krok 4: Wyodrębniij styl

([style](#)) Styl to zbiór atrybutów określających wygląd i format widoku. Styl może obejmować kolor czcionki, rozmiar czcionki, kolor tła, wypełnienie, margines i inne typowe atrybuty.

Możesz wyodrębnić formatowanie widoku tekstowego nazwy do stylu i ponownie użyć stylu dla dowolnej liczby widoków w aplikacji. Ponowne użycie stylu zapewnia spójny wygląd aplikacji w przypadku wielu widoków. Korzystanie ze stylów pozwala również zachować te wspólne atrybuty w jednym miejscu.

1. Kliknij prawym przyciskiem myszy `TextView` w **Component Tree** i wybierz **Refactor > Extract Style**.
2. W oknie dialogowym **Extract Android Style** wyczyść pole wyboru `layout_width`, `layout_height` i `textAlignment`. Te atrybuty są zwykle różne dla każdego widoku, więc nie chcesz, aby były częścią stylu.
3. W polu **Style name** field, wpisz `NameStyle`.

4. Kliknij OK.

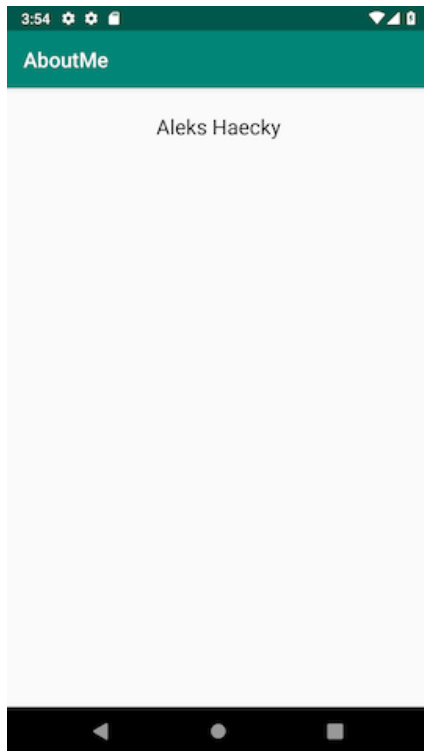


5. Styl jest również zasobem, więc styl jest zapisywany w folderze `res/values/` folder in a `styles.xml` Otwórz `styles.xml` i sprawdź wygenerowany kod dla stylu `NameStyle`, który będzie wyglądał podobnie do tego::

```
<style name="NameStyle">
    <item name="android:layout_marginTop">@dimen/layout_margin</item>
    <item name="android:fontFamily">@font/roboto</item>
    <item name="android:paddingTop">@dimen/small_padding</item>
    <item name="android:textColor">@android:color/black</item>
    <item name="android:textSize">@dimen/text_size</item>
</style>
```

6. Otwórz `activity_main.xml` i przejdź do zakładki Tekst. Zauważ, że wygenerowany styl jest używany w widoku tekstowym jako `style="@style/NameStyle"`.

7. Uruchom aplikację i zauważ zmiany `TextView`.




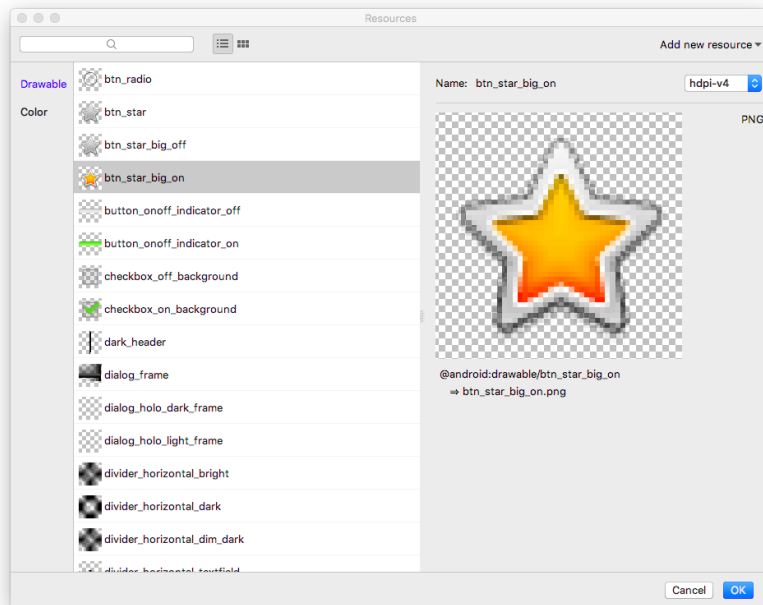
## 7. Zadanie: Dodaj ImageView

Większość rzeczywistych aplikacji na Androida składa się z kombinacji widoków do wyświetlania obrazów, wyświetlania tekstu i akceptowania danych wejściowych od użytkownika w formie zdarzeń tekstowych lub kliknięć. W tym zadaniu dodajesz widok, aby wyświetlić obraz.

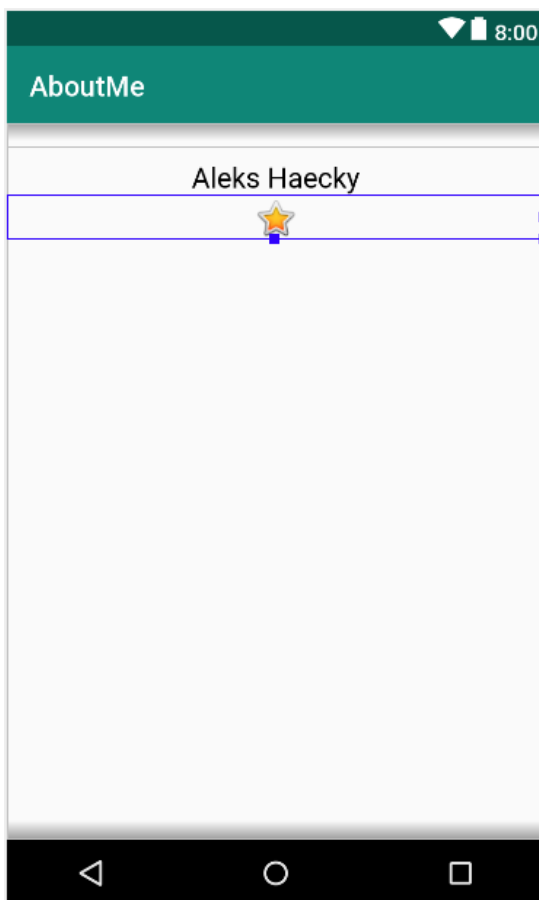
ImageView to widok do wyświetlania zasobów obrazu. Na przykład ImageView może wyświetlać zasoby Bitmapy, takie jak pliki PNG, JPG, GIF lub WebP, lub wyświetlać zasoby do rysowania, takie jak rysunek wektorowy.

Istnieją zasoby graficzne dostarczane z systemem Android, takie jak przykładowe ikony, awatary i tła. Dodaj jeden z tych zasobów do swojej aplikacji.

1. Wyświetl plik layout na karcie **Design** a następnie przeciągnij **ImageView** z panelu **Palette** do miejsca poniżej `name_text` w **Component Tree**. Zostanie otwarte okno dialogowe **Resources**.
2. Wybierz **Drawable** jeśli nie jest jeszcze zaznaczony.
3. Rozwiń **android**, przewiń i wybierz **btn\_star\_big\_on**. To gwiazda. .
4. Kliknij **OK**.



Obraz gwiazdy zostanie dodany do układu pod Twoim nazwiskiem. Ponieważ masz pionowy układ liniowy, dodane widoki są wyrównane w pionie..



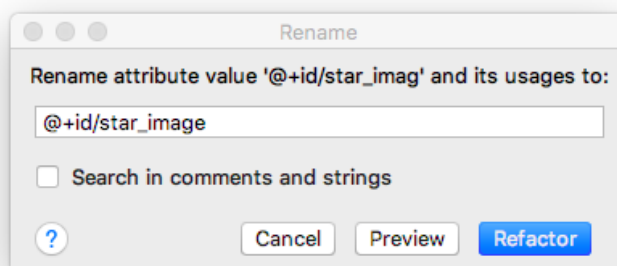
- Przejdź do zakładki Tekst i spójrz na wygenerowany kod ImageView. Szerokość jest ustawiona na match\_parent, więc widok będzie tak szeroki jak jego element




nadrzędny. Wysokość jest ustawiona na `wrap_content`, więc widok będzie tak wysoki jak jej zawartość. `ImageView` odwołuje się do rysunku `btn_star_big_on`.

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:srcCompat="@android:drawable/btn_star_big_on" />
```

6. Aby zmienić nazwę identyfikatora `ImageView`, kliknij prawym przyciskiem myszy `"@+id/imageView"` i wybierz **Refactor > Rename**.
7. W oknie dialogowym **Rename** dialog, ustaw identyfikator na `@+id/star_image`. Kliknij **Refactor**.



**Tip: Refactor > Rename** zmienia nazwy wszystkich wystąpień nazwy atrybutu lub zmiennej w projekcie aplikacji.

8. Na karcie **Design** w **Component Tree**, kliknij ikonę ostrzeżenia  obok `star_image`. Ostrzeżenie dotyczy brakującej zawartości Opis `contentDescription`, którego czytelnicy ekranu używają [screen readers](#) używają do opisywania obrazów użytkownikowi.
9. W panelu **Attributes** kliknij trzy kropki ... obok atrybutu `contentDescription`. Zostanie otwarte okno dialogowe **Resources**.
10. W oknie dialogowym **Resources** wybierz **Add new resource > New string Value**. Ustaw pole **Resource name** na `yellow_star`, Ustaw pole **Resource value** na `Yellow star`. Kliknij **OK**.
11. Użyj panelu **Attributes** aby dodać górny margines `16dp` (czyli `@dimen/layout_margin`) do `yellow_star`, aby oddzielić obraz gwiazdy od imienia.
12. Uruchom aplikację. Twoje imię i gwiazdka są wyświetlane w interfejsie aplikacji.

## 8. Zadanie: dodaj `ScrollView`

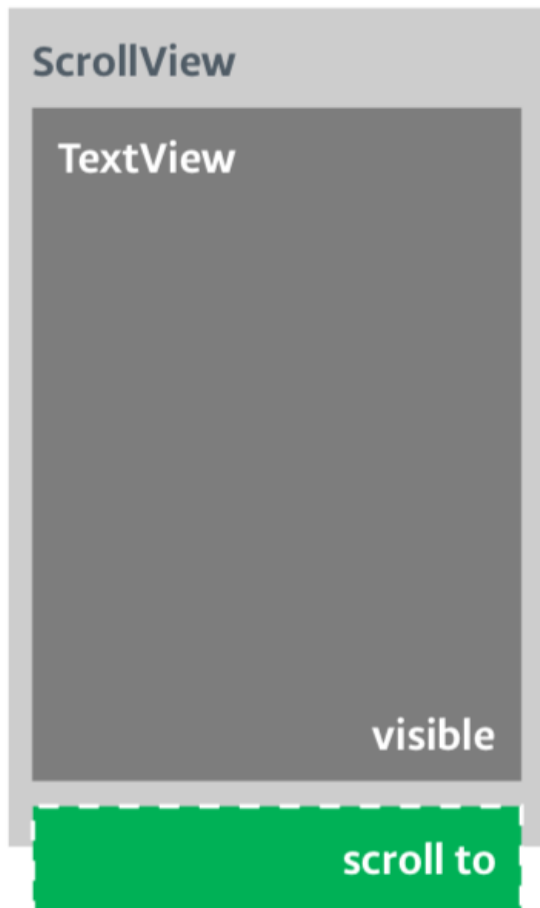
[ScrollView](#) to jest view group (grupa widoków) , która pozwala na przewijanie hierarchii widoków w niej zawartych.

[ScrollView](#) może zawierać tylko jeden inny widok lub grupę widoków jako dziecko. Widok potomny jest zwykle układem liniowym. Wewnątrz układu liniowego możesz dodawać inne widoki.

Poniższy obraz pokazuje przykład ScrollView, który zawiera LinearLayout, który zawiera kilka innych widoków..



W tym zadaniu dodasz ScrollView, który pozwala użytkownikowi przewijać widok tekstu, który wyświetla krótką biografię.



## Krok 1: Dodaj ScrollView, który zawiera TextView

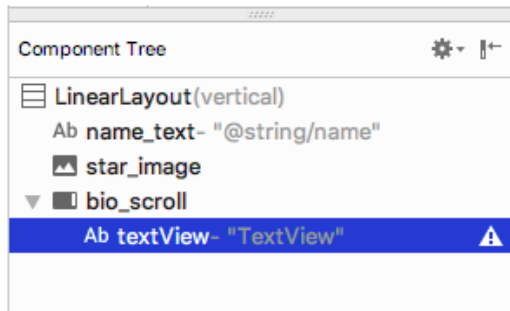
1. Otwórz plik `activity_main.xml` w zakładce **Design** .
2. Przeciągnij scroll view do layoutu przeciągając go do edytora projektu lub do **Component Tree**. Umieść scroll view poniżej obrazu gwiazdy.
3. Przejdź do zakładki **Text** aby sprawdzić wygenerowany kod.

```
// Auto generated code
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

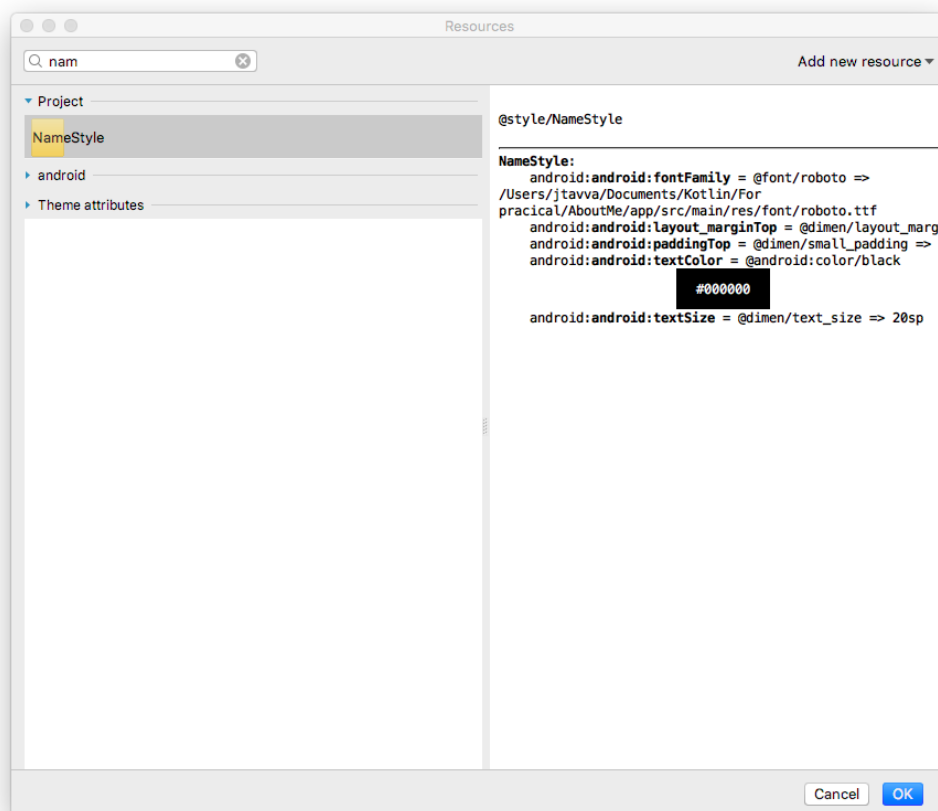
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" />
</ScrollView>
```

Wysokość i szerokość **ScrollView** odpowiadają elementowi nadrzędnemu. Gdy widok `name_text` **text view** i `star_image` **image view** wykorzystają wystarczającą ilość miejsca w pionie do wyświetlenia ich zawartości, system Android ustawi **ScrollView** tak, aby wypełnić resztę dostępnego miejsca na ekranie.

4. Dodaj identyfikator do ScrollView i nazwij go bio\_scroll. Dodanie identyfikatora do ScrollView daje systemowi Android uchwyt do widoku, dzięki czemu gdy użytkownik obraca urządzenie, system zachowuje pozycję przewijania.
5. Wewnątrz ScrollView usuń kod LinearLayout, ponieważ Twoja aplikacja będzie miała tylko jeden widok, który można przewijać - TextView..
6. Przeciągnij TextView z palety do drzewa komponentów. Umieść TextView pod bio\_scroll, jako element potomny bio\_scroll.



7. Ustaw identyfikator nowego widoku tekstu na bio\_text.
8. Następnie dodaj styl dla nowego widoku tekstu. W panelu Atrybuty kliknij trzy kropki ... obok atrybutu stylu, aby otworzyć okno dialogowe. **Resources** dialog.
9. W oknie dialogowym **Resources** wyszukaj nazwę NameStyle. Wybierz NameStyle z listy i kliknij OK. Widok tekstu używa teraz stylu NameStyle, który utworzyłeś w poprzednim zadaniu.



## Krok 2: Dodaj swoją biografię do nowego TextView

Otwórz plik strings.xml, utwórz zasób łańcucha o nazwie bio i wstaw długi tekst np. o sobie lub inny

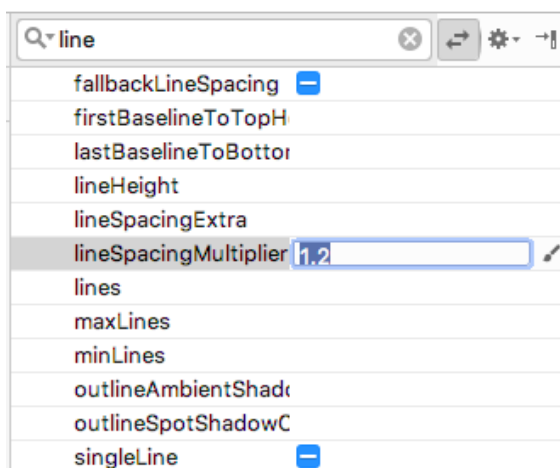
#### Note:

- Użyj \n, aby wskazać przerwanie linii.
- Jeśli używasz apostrofu, musisz użyć backslash. Na przykład::  
"You mustn\'t forget the backslash."
- W przypadku pogrubionego tekstu użyj <b>...</b>, a dla kursywy użyj <i>...</i>. Na przykład:  
"This text is <b>bold</b> and this text is <i>italics</i>."

#### Przykładowa biografia::

```
<string name="bio">Hi, my name is Aleks.  
\n\nI love fish.  
\n\nThe kind that is alive and swims around in an aquarium or river, or a  
lake, and definitely the ocean.  
\nFun fact is that I have several aquariums and also a river.  
\n\nI like eating fish, too. Raw fish. Grilled fish. Smoked fish. Poached  
fish - not so much.  
\nAnd sometimes I even go fishing.  
\nAnd even less sometimes, I actually catch something.  
\n\nOnce, when I was camping in Canada, and very hungry, I even caught a  
large salmon with my hands.  
\n\nI\'ll be happy to teach you how to make your own aquarium.  
\nYou should ask someone else about fishing, though.\n\n</string>
```

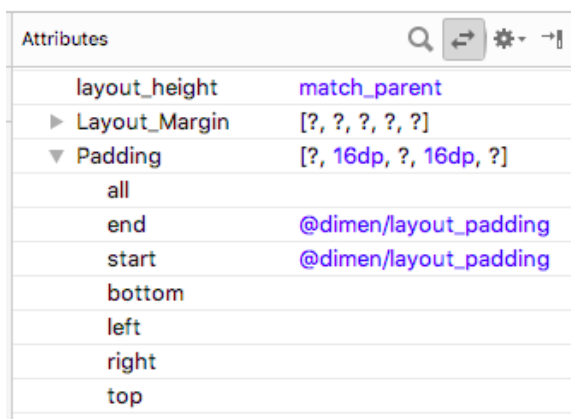
2. W widoku tekstowym bio\_text ustaw wartość atrybutu text na zasób bio string zawierający biografię..
3. Aby ułatwić czytanie tekstu bio\_text, dodaj odstępy między wierszami. Użyj atrybutu. [lineSpacingMultiplier](#) i nadaj mu wartość 1.2.



Zwróć uwagę, że w edytorze projektu biotekst przebiega aż do bocznych krawędzi ekranu. Aby rozwiązać ten problem, możesz dodać atrybuty left, start, right, i end do LinearLayout. Nie trzeba dodawać dolnego dopełnienia, ponieważ tekst, który biegnie aż do dołu, sygnalizuje użytkownikowi, że tekst jest przewijany.

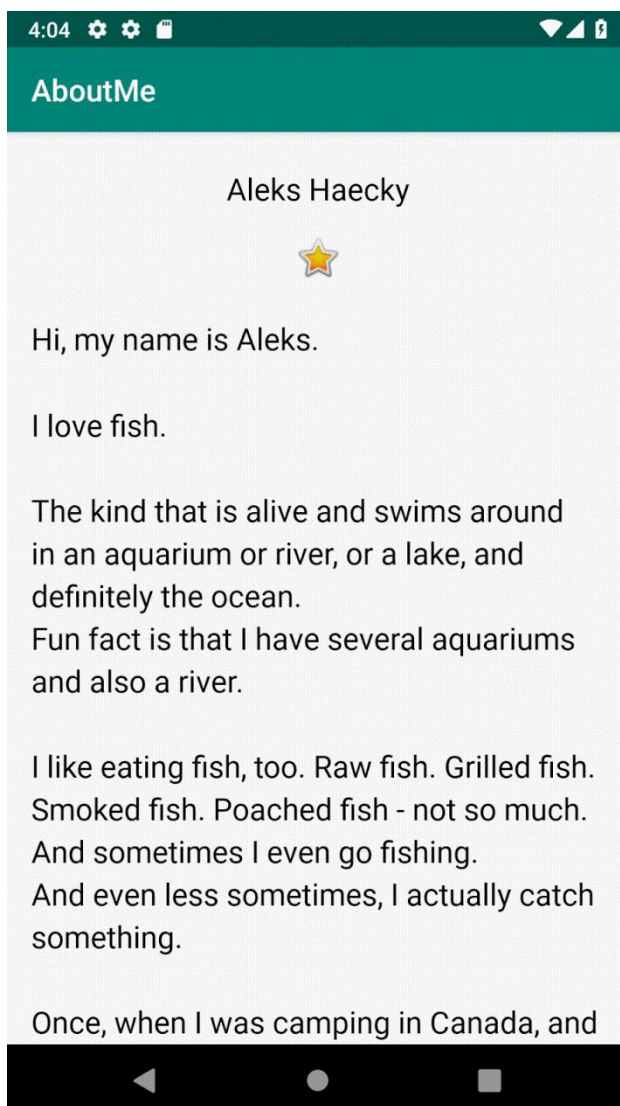
4. Dodaj start i end padding na 16dp do LinearLayout.

5. Przejdź do karty **Text** wyodrębnij zasób wymiaru i nazwij go `layout_padding`.



**Note:** Począwszy od poziomu API 17, użyj „start” i „end” zamiast „left” i „right” dla uzupełnienia i marginesu, aby dostosować swoją aplikację do języków RTL, takich jak arabski..

6. Uruchom aplikację i przewiń tekst.



**10. Wewnątrz ScrollView dodaj inne wudoki np. ImageView**