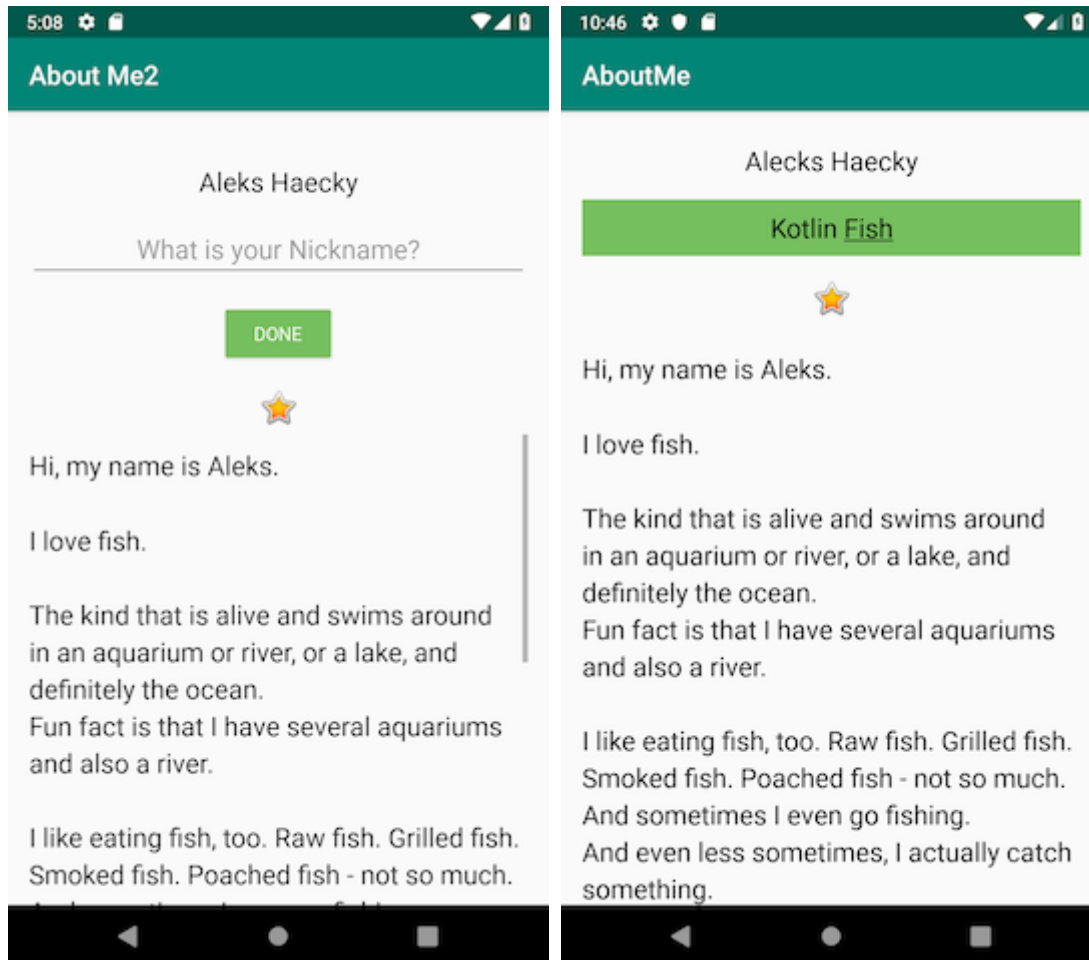


2.

W tym ćwiczeniu rozszerzasz aplikację AboutMe, aby dodać interakcję użytkownika. Dodaj pole pseudonimu, przycisk GOTOWE i widok tekstowy, aby wyświetlić pseudonim. Gdy użytkownik wpisze pseudonim i stuknie przycisk GOTOWE, widok tekstu zostanie zaktualizowany, aby pokazać wprowadzony pseudonim. Użytkownik może ponownie zaktualizować pseudonim, dotykając widoku tekstowego.



3. Zadanie: dodaj EditText do wprowadzania tekstu

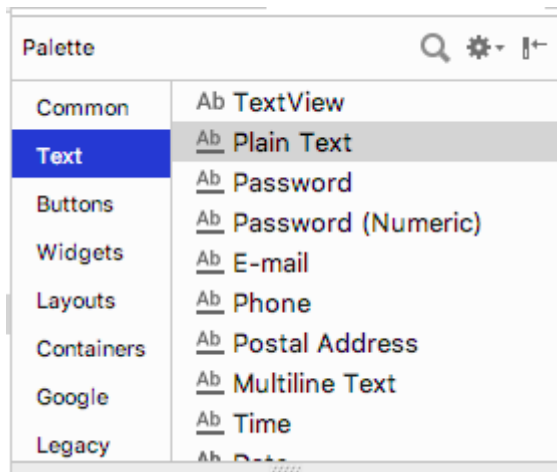
W tym zadaniu dodajesz pole wprowadzania EditText, aby umożliwić użytkownikowi wpisanie pseudonimu..

Aby zaakceptować wprowadzanie tekstu, system Android udostępnia widget interfejsu użytkownika (UI) zwany *edit text*. Tekst edytowalny definiujemy przy pomocy [EditText](#) podklasy `TextView`

Krok 2: Dodaj EditText

1. W Android Studio otwórz plik układu `activity_main.xml` na karcie **Design**.

2. W okienku **Palette** kliknij opcję **Text**.

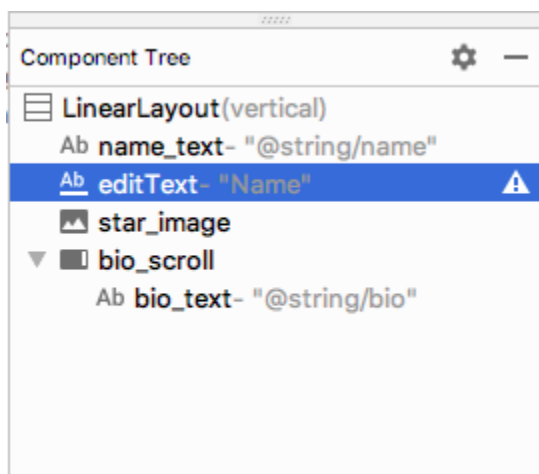


Ab TextView, który jest TextView, pokazuje się na górze listy elementów tekstowych w panelu **Palette**. Poniżej Ab TextView znajduje się wiele widoków EditText.

W panelu **Palette** zwróć uwagę, jak ikona TextView pokazuje litery Ab bez podkreślenia. Ikony EditText pokazują Ab podkreślone. Podkreślenie wskazuje, że widok można edytować.

Dla każdego z widoków EditText system Android ustawia różne atrybuty, a system wyświetla odpowiednią metodę miękkiego wprowadzania (np. Klawiaturę ekranową).

3. Przeciągnij **PlainText** do **Component Tree** umieść go poniżej name_text i nad star_image.



4. Użyj panelu Atrybuty, aby ustawić następujące atrybuty w widoku EditText.

Attribute	Value
id	nickname_edit
layout_width	match_parent (default)

`layout_height wrap_content (default)`

5. Uruchom aplikację. Nad obrazem gwiazdy widoczny jest tekst edycji z domyślnym tekstem "Name".

4. Zadanie: nadaj styl EditText

W tym zadaniu stylizujesz widok EditText, dodając podpowiedź, zmieniając wyrównanie tekstu, zmieniając styl na NameStyle i ustawiając typ wprowadzania.


Krok 1: Dodaj tekst podpowiedzi (hint text)

1. Add Dodaj nowy zasób łańcucha dla podpowiedzi w pliku. `string.xml`

```
<string name="what_is_your_nickname">What is your Nickname?</string>
```

Wskazówka: Dobrą praktyką jest wyświetlanie podpowiedzi w każdym widoku EditText, aby pomóc użytkownikom w określeniu, czego można się spodziewać w przypadku pól edytowalnych..

2. Użyj panelu **Attributes** aby ustawić następujące atrybuty widoku EditText:

Attribute	Value
style	NameStyle
textAlignment	 (center)
hint	@string/what_is_your_nickname

3. W panelu **Attributes** usuń wartość Name. Wartość atrybutu tekstowego musi być pusta, aby podpowiedź była wyświetlana.

Krok 2: Ustaw atrybut inputType

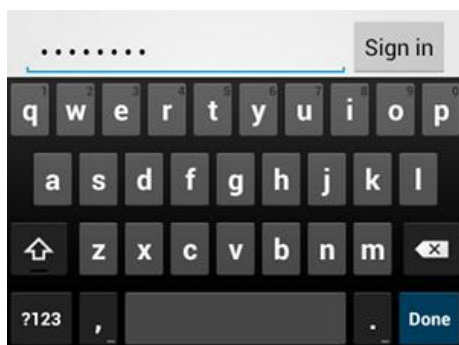
Atrybut `inputType` określa typ danych wejściowych, które użytkownicy mogą wprowadzać w widoku EditText. System Android wyświetla odpowiednie pole wprowadzania i klawiaturę ekranową, w zależności od ustawionego typu wejścia.

Aby zobaczyć wszystkie możliwe typy wprowadzania, w panelu **Attributes** kliknij pole `inputType` lub kliknij trzy kropki ... obok pola. Otwiera się lista, która pokazuje wszystkie typy danych wejściowych, których możesz użyć, z zaznaczonym aktualnie aktywnym typem

danych wejściowych. Możesz wybrać więcej niż jeden typ wejścia.

▼ inputType	[textPersonName]
date	<input type="checkbox"/>
textUri	<input type="checkbox"/>
textShortMessage	<input checked="" type="checkbox"/>
textLongMessage	<input type="checkbox"/>
textAutoCorrect	<input type="checkbox"/>
none	<input type="checkbox"/>
numberSigned	<input type="checkbox"/>
textVisiblePassword	<input type="checkbox"/>
textWebEditText	<input type="checkbox"/>
textMultiLine	<input type="checkbox"/>
textNoSuggestions	<input type="checkbox"/>
textFilter	<input type="checkbox"/>
number	<input type="checkbox"/>
datetime	<input type="checkbox"/>
textWebEmailAddress	<input type="checkbox"/>
textPersonName	<input checked="" type="checkbox"/>
text	<input checked="" type="checkbox"/>
textPhonetic	<input type="checkbox"/>
textCapSentences	<input type="checkbox"/>
textPassword	<input type="checkbox"/>
textAutoComplete	<input type="checkbox"/>
textImeMultiLine	<input type="checkbox"/>
textPostalAddress	<input type="checkbox"/>
numberDecimal	<input type="checkbox"/>
textEmailAddress	<input checked="" type="checkbox"/>
numberPassword	<input type="checkbox"/>

Na przykład w przypadku haseł użyj wartości textPassword. Pole tekstowe ukrywa dane wejściowe użytkownika.



W przypadku numerów telefonów użyj wartości telefonu. Wyświetlana jest klawiatura numeryczna, a użytkownik może wprowadzać tylko cyfry.



Ustaw typ danych wejściowych dla pola pseudonimu:

1. Ustaw atrybut `inputType` na `textPersonName` dla `nickname_edit`.
2. W panelu **Component Tree** pane, zwróć uwagę na ostrzeżenie `autoFillHints` To ostrzeżenie nie dotyczy tej aplikacji i jest poza zakresem tego kodu, więc możesz je zignorować.



3. W panelu **Attributes** sprawdź wartości następujących atrybutów widoku `EditText`:

Attribute	Value
<code>id</code>	<code>nickname_edit</code>
<code>layout_width</code>	<code>match_parent (default)</code>
<code>layout_height</code>	<code>wrap_content (default)</code>
<code>style</code>	<code>@style/NameStyle</code>
<code>inputType</code>	<code>textPersonName</code>
<code>hint</code>	<code>"@string/what_is_your_nickname"</code>
<code>text</code>	<code>(empty)</code>

5. Zadanie: dodaj przycisk i nadaj mu styl.

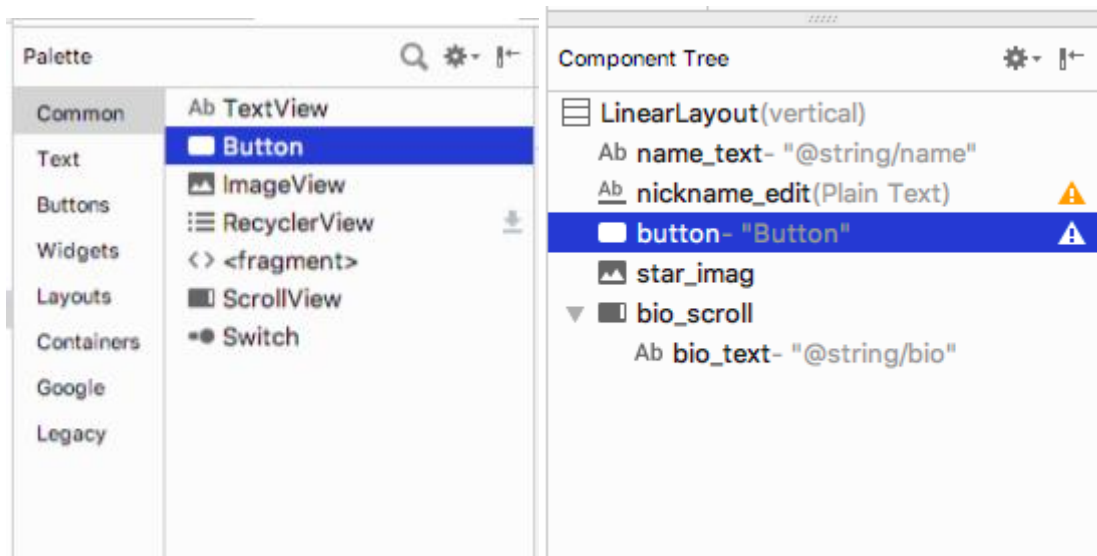
Przycisk to element interfejsu użytkownika, który użytkownik może dotknąć, aby wykonać akcję. Przycisk może składać się z tekstu, ikony lub zarówno tekstu, jak i ikony.



W tym zadaniu dodajesz przycisk GOTOWE, który użytkownik stuka po wprowadzeniu pseudonimu. Przycisk zamienia widok `EditText` na widok `TextView`, który wyświetla pseudonim. Aby zaktualizować pseudonim, użytkownik może dotknąć widoku `TextView`.

Krok 1: Dodaj przycisk DONE

1. Przeciągnij przycisk z panelu **Palette** do **Component Tree**. Umieść przycisk poniżej tekstu edycji pseudonimu..



2. Utwórz nowy string resource nazwij go done.

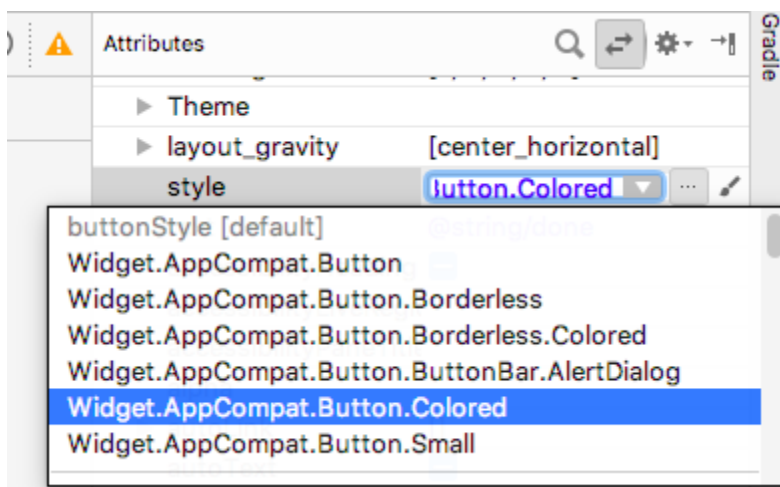
```
<string name="done">Done</string>
```

3. Użyj panelu **Attributes** aby ustawić następujące atrybuty w nowo dodanym widoku przycisku:

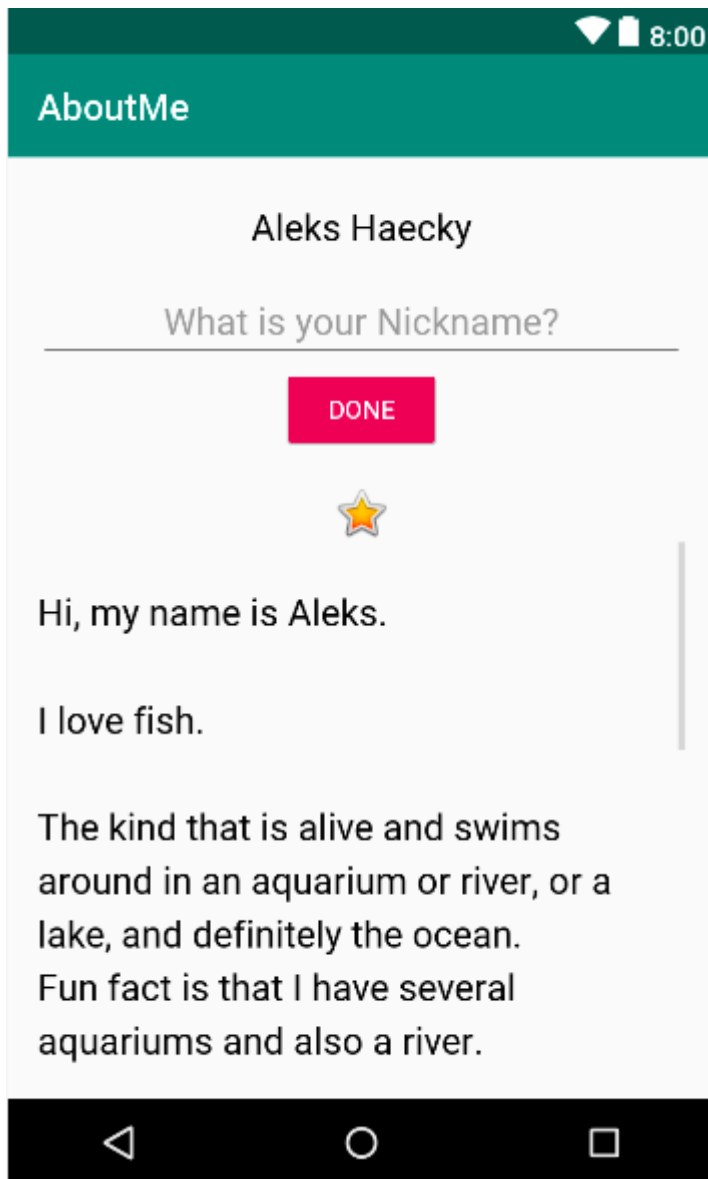
Attribute	Values
id	done_button
text	@string/done
layout_gravity	center_horizontal
layout_width	wrap_content

layout_gravity wyśrodkowuje widok w układzie nadrzędnym, LinearLayout.

4. Zmień styl na Widget.AppCompat.Button.Colored, który jest jednym ze wstępnie zdefiniowanych stylów udostępnianych przez system Android. Możesz wybrać styl z menu rozwijanego lub z okna **Resources**



Ten styl zmienia kolor przycisku na `colorAccent`. `colorAccent` jest zdefiniowany w pliku `res/values/colors.xml`



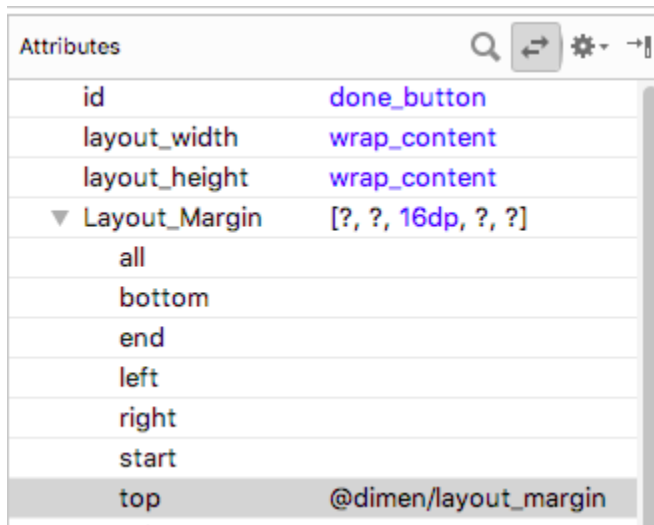
Plik `colors.xml` zawiera domyślne kolory dla Twojej aplikacji. Możesz dodać nowe zasoby kolorów lub zmienić istniejące zasoby kolorów w swoim projekcie, w zależności od wymagań aplikacji.

Przykładowy plik `colors.xml`:

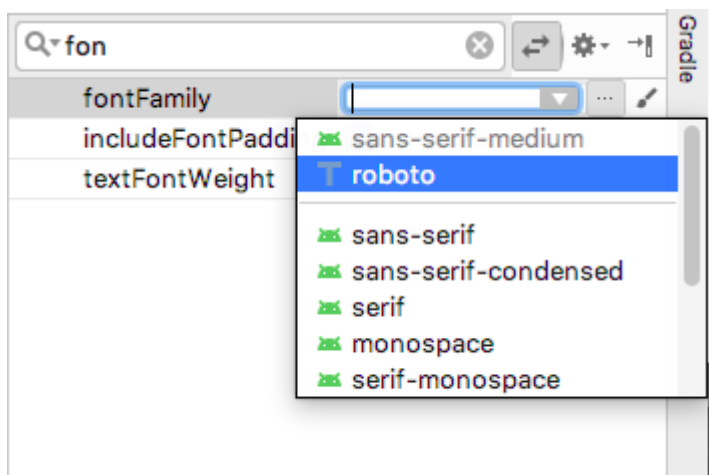
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
</resources>
```

Krok 2: Styl przycisku DONE

1. W panelu **Attributes** dodaj górny margines, wybierając **Layout_Margin > Top**. Ustawiamy górny margines na `layout_margin`, zdefiniowany w pliku `dimens.xml` file.



2. Ustaw atrybut `fontFamily` na `roboto` z menu rozwijanego.



3. Przejdź do zakładki Tekst i sprawdź wygenerowany kod XML dla nowo dodanego przycisku.

```
<Button
    android:id="@+id/done_button"
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="@dimen/layout_margin"
    android:fontFamily="@font/roboto"
    android:text="@string/done" />
```

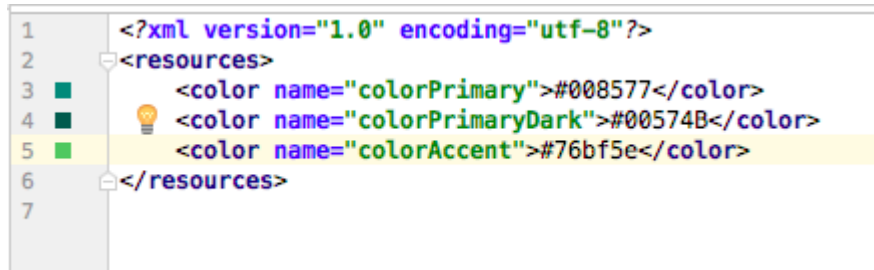
Krok 3: Zmień zasób koloru

W tym kroku zmieniasz kolor akcentu przycisku, tak aby pasował do paska aplikacji Twojej aktywności.

1. Otwórz `res/values/colors.xml` zmień wartość `colorAccent` na `#76bf5e`.

```
<color name="colorAccent">#76bf5e</color>
```

Kolor odpowiadający kodowi HEX można zobaczyć na lewym marginesie edytora plików.



Zwróć uwagę na zmianę koloru przycisku w edytorze projektu.

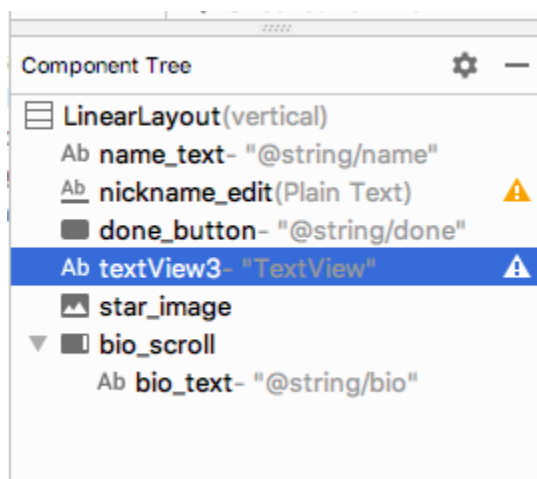
2. Uruchom aplikację. Powinieneś zobaczyć stylizowany przycisk.

6. Zadanie: Dodaj TextView, aby wyświetlić pseudonim „nickname”

Po wprowadzeniu przez użytkownika pseudonimu i dotknięciu przycisku **DONE** pseudonim jest wyświetlany w widoku TextView. W tym zadaniu dodasz widok tekstu z kolorowym tłem. Widok tekstu wyświetla pseudonim użytkownika nad obrazem. `star_image`.


Krok 1: Dodaj TextView dla pseudonimu nickname

1. Przeciągnij text view z panelu **Palette** do **Component Tree**. Umieść widok tekstu poniżej przyciskue `done_button` i nad `star_image`.



2. Użyj panelu **Attributes** aby ustawić następujące atrybuty dla nowego widoku TextView:

Attribute	Value
id	nickname_text

```
style           NameStyle
textAlignment   (center)
```

Krok 2: Zmień widoczność TextView

Możesz wyświetlać lub ukrywać widoki w aplikacji za pomocą atrybutu `visibility` attribute. Ten atrybut przyjmuje jedną z trzech wartości:

- `visible`: widok jest widoczny.
 - `Invisible`: ukrywa widok, ale widok nadal zajmuje miejsce w layoutcie.
 - `gone`: ukrywa widok, a widok nie zajmuje miejsca w layoutcie.
1. W panelu **Attributes** ustaw `visibility` widoku `nickname_text` na `gone`, ponieważ nie chcesz, aby aplikacja wyświetlała ten widok tekstowy na początku.



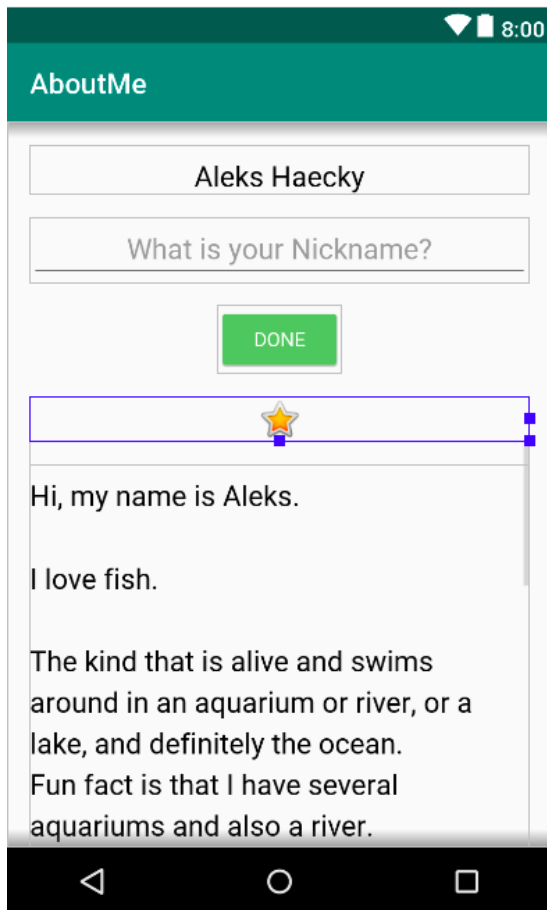
Zauważ, że po zmianie atrybutu w panelu Atrybuty widok pseudonimu tekst znikną z edytora projektu. Widok jest ukryty w podglądzie layout.

2. Zmień wartość atrybutu tekstowego w widoku pseudonimu na pusty ciąg znaków.

Your Wygenerowany kod XML dla tego TextView powinien wyglądać podobnie do tego:

```
<TextView
    android:id="@+id/nickname_text"
    style="@style/NameStyle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textAlignment="center"
    android:visibility="gone"
    android:text="" />
```

layout preview powinien wyglądać mniej więcej tak:



7. Zadanie: Dodaj detektor kliknięć (click listener) do przycisku DONE button

Moduł obsługi kliknięć na obiekcie przycisku (lub w dowolnym widoku) określa akcję, która ma zostać wykonana po stuknięciu przycisku (widoku). Funkcja, która obsługuje zdarzenie kliknięcia, powinna zostać zaimplementowana w Aktywności, które obsługuje układ za pomocą przycisku (widok).

Odbiornik kliknięć ma generalnie ten format, w którym przekazany widok jest widokiem, który otrzymał kliknięcie lub dotknięcie.

```
private fun clickHandlerFunction(viewThatIsClicked: View) {  
    // Add code to perform the button click event  
}
```

Możesz dołączyć funkcję detektora kliknięć (click-listener) do zdarzeń kliknięcia przycisku na dwa sposoby:

- W pliku XML layout możesz dodać atrybut `android: onClick` do elementu `<Button>`. Na przykład::

```
<Button  
    android:id="@+id/done_button"  
    android:text="@string/done"  
    ...  
    android:onClick="clickHandlerFunction"/>
```

Albo

- Możesz to zrobić programowo w czasie wykonywania, w `onCreate()` w Activity, wywołując [setOnClickListener](#). Przykład:

```
myButton.setOnClickListener {  
    clickHandlerFunction(it)  
}
```

W tym zadaniu programowo dodajesz detektor kliknięć dla przycisku. `done_button`. w pliku `MainActivity.kt`.

Twoja funkcja nasłuchiwanie kliknięć, o nazwie `addNickname`, wykona następujące czynności:

- Pobierze tekst z `nickname_edit`.
- Ustawi tekst widoku `nickname_text`.
- Ukryje `nickname_edit` i `done_button`.
- Wyświetli widok `nickname_text`.

Krok 1: Dodaj detektor kliknięć click listener

```
private fun addNickname(view: View) {  
}  
...  
val editText = findViewById<EditText>(R.id.nickname_edit)  
val nicknameTextView = findViewById<TextView>(R.id.nickname_text)  
...  
nicknameTextView.text = editText.text  
...  
editText.visibility = View.GONE  
...
```

Ukryj przycisk **DONE**, ustawiając właściwość widoczności na `View.GONE`. Masz już odniesienie do przycisku jako parametr wejściowy funkcji, `view`.

```
view.visibility = View.GONE  
...  
nicknameTextView.visibility = View.VISIBLE  
...
```

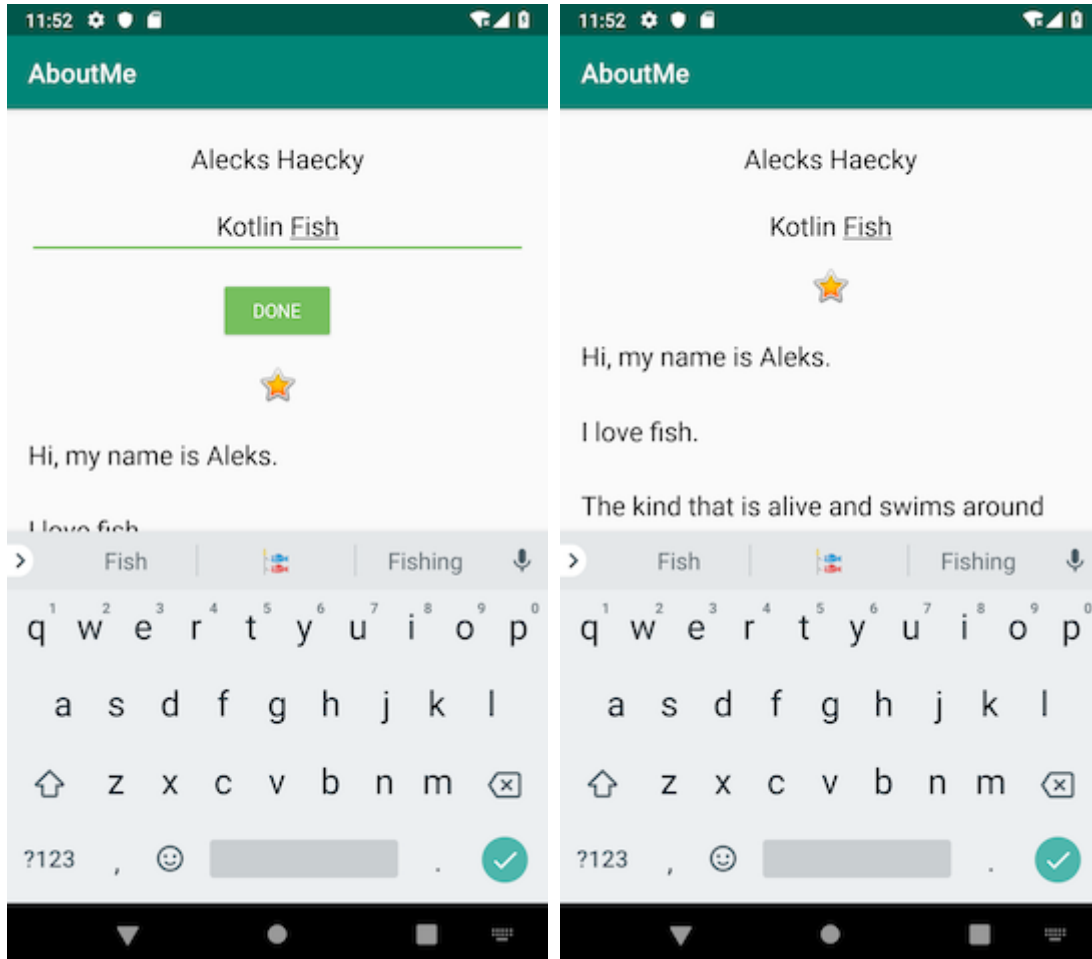
Krok 2: Dołącz detektor kliknięć do przycisku DONE Button

Teraz, gdy masz funkcję, która określa akcję, która ma być wykonana po stuknięciu przycisku **DONE**, musisz dołączyć tę funkcję do widoku przycisku.

```
findViewById<Button>(R.id.done_button).setOnClickListener {  
    addNickname(it)  
}
```

W powyższym kodzie `it` odnosi się do przycisku `done_button`, czyli widoku przekazanego jako argument.

2. Uruchom aplikację, wprowadź pseudonim i dotknij przycisku **DONE**. Zwróć uwagę, w jaki sposób tekst edycji i przycisk są zastępowane widokiem tekstu pseudonimu..



Zauważ, że nawet po tym, jak użytkownik dotknie przycisku **DONE**, klawiatura jest nadal widoczna. To zachowanie jest domyślne.

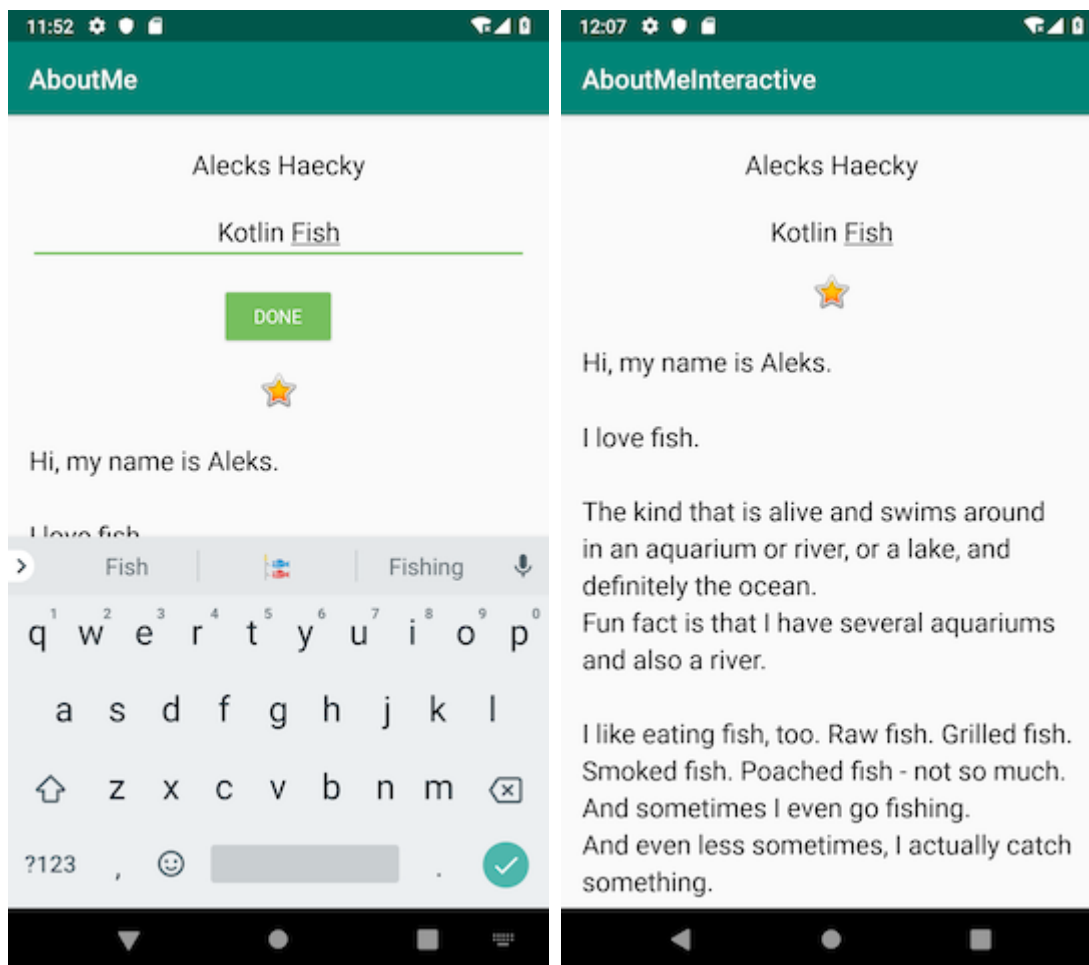
Krok 3: Ukryj klawiaturę

In this step, you add code to hide the keyboard after the user taps the **DONE** button.

1. W `MainActivity.kt` na końcu funkcji `addNickname()` dodaj następujący kod.

```
// Hide the keyboard.  
val inputMethodManager = getSystemService(Context.INPUT_METHOD_SERVICE) as  
InputMethodManager  
inputMethodManager.hideSoftInputFromWindow(view.windowToken, 0)
```

2. Uruchom ponownie aplikację.



Użytkownik nie może zmienić pseudonimu po dotknięciu przycisku **DONE**. W następnym zadaniu uczynisz aplikację bardziej interaktywną i dodasz funkcjonalność, aby użytkownik mógł zaktualizować pseudonim.

8. Zadanie: dodaj funkcjonalność, aby użytkownik mógł zaktualizować pseudonim.

Step 1: Add a click listener

1. W `MainActivity`, dodaj funkcję detektora kliknięć o nazwie `updateNickname` dla widoku tekstowego pseudonimu..

```
private fun updateNickname (view: View) {
}
```

2. Wewnątrz funkcji `updateNickname` uzyskaj odniesienie do edytowanego tekstu pseudonimu oraz odwołanie do przycisku **DONE**.

```
val editText = findViewById<EditText>(R.id.nickname_edit)
val doneButton = findViewById<Button>(R.id.done_button)
```

3. Dodaj

```

editText.visibility = View.VISIBLE
doneButton.visibility = View.VISIBLE
view.visibility = View.GONE

```

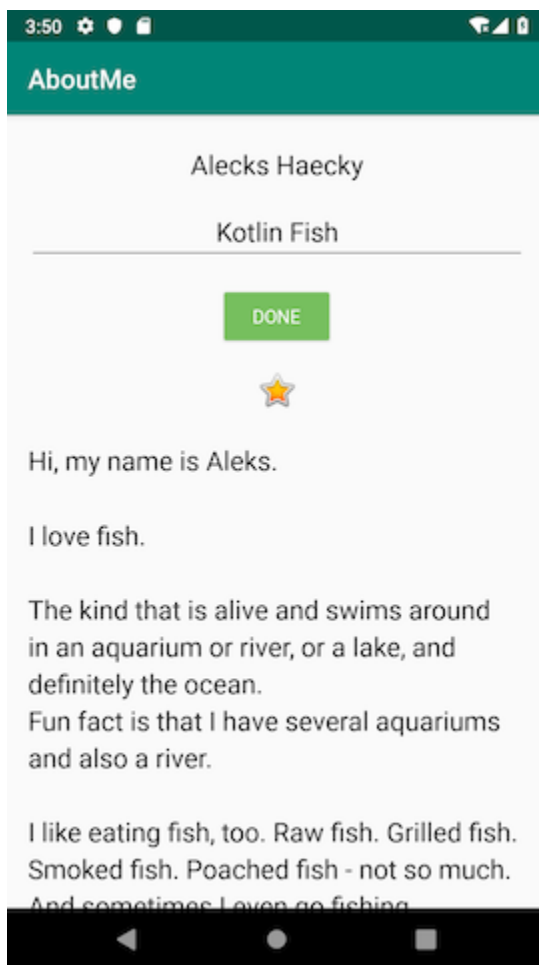
4. W `MainActivity.kt`, na końcu `onCreate()` ustaw `setOnClickListener` `nickname_text` I przełącz referencje do `updateNickname()`.

```

findViewById<TextView>(R.id.nickname_text).setOnClickListener {
    updateNickname(it)
}

```

5. Uruchom aplikację. Wprowadź pseudonim, dotknij przycisku GOTOWE, a następnie dotknij pseudonimu Widok `TextView`. Widok pseudonimu znika, a tekst edycji i przycisk GOTOWE stają się widoczne..



Zauważ, że domyślnie widok `EditText` nie jest aktywny, a klawiatura nie jest widoczna. Użytkownikowi trudno jest zorientować się, że widok tekstu pseudonimu można kliknąć. W następnym zadaniu dodasz fokus i styl do widoku tekstu pseudonimu.

Krok 2: Ustaw fokus w widoku `EditText` i pokaż klawiaturę

1. Na końcu funkcji `updateNickname` ustaw fokus w widoku `EditText`. Użyj metody [`requestFocus\(\)`](#)

```
// Set the focus to the edit text.  
editText.requestFocus()
```

2. Na końcu `updateNickname` dodaj kod, aby klawiatura była widoczna..

```
// Show the keyboard.  
val imm = getSystemService(Context.INPUT_METHOD_SERVICE) as  
InputMethodManager  
imm.showSoftInput(editText, 0)
```

Krok 3: Dodaj kolor tła do widoku pseudonimu `TextView`

1. Ustaw kolor tła widoku tekstu pseudonimu na `@color/colorAccent`, i dodaj bottom padding of `@dimen/small_padding`. Zmiany te będą służyć jako wskazówka dla użytkownika, że można kliknąć tekst widoku pseudonimu.

```
android:background="@color/colorAccent"  
android:paddingBottom="@dimen/small_padding"
```

2. Uruchom swoją ostateczną aplikację. Tekst edycji ma fokus, pseudonim jest wyświetlany w tekście edycji, a widok tekstu pseudonimu jest stylizowany.

