Android Studio - Java: Suwak

© Copyright by 3bird Projects 2026, http://edukacja.3bird.pl

Opis

Aplikacja prezentuje suwak, który zwiększa / zmniejsza wielkość czcionki. Wersja rozbudowana dodatkowo zawiera suwak zmieniający kolor kwadratu oraz przełącznik motywów *Light/Dark*.

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii, twarde odstępy, odmienne cudzysłowy, odmienne apostrofy, odmienne myślniki. To godziny dodatkowej pracy na wykrywanie błędów i ich poprawianie. Kod należy przepisać ze zrozumieniem.

Kod - wersja podstawowa

Plik „activity_main.xml”

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

    <TextView
        android:id="@+id/przykladowyTekst"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Twoje Nazwisko"
        android:textSize="20sp"
        android:gravity="center"
        android:layout_marginTop="60dp"
        android:layout_marginBottom="40dp"/>

    <SeekBar
        android:id="@+id/suwak"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="30"
    android:progress="10"/>
</LinearLayout>
```

Plik „MainActivity.java”

```
package com.example.nazwaAplikacji;
// import ...
public class MainActivity extends AppCompatActivity {
  // Konwencja: W Java stałe piszemy wielkimi literami:
  private static final int MINIMALNA_WIELKOSC_CZCIONKI = 10;
  private TextView przykladowyNapis;
  private SeekBar suwakDoNapisu;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
      Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
      v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
      return insets;
    });
    przykladowyNapis = findViewById(R.id.przykladowyTekst);
    suwakDoNapisu = findViewById(R.id.suwak);

    suwakDoNapisu.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
    {
      @Override
      public void onProgressChanged(SeekBar suwak, int progressSuwaka, boolean
      czyZmianaPrzezUzytkownika) {
```

```
            int obecnyRozmiarCzcionki = MINIMALNA_WIELKOSC_CZCIONKI + progressSuwaka;
            przykladowyNapis.setTextSize(TypedValue.COMPLEX_UNIT_SP,
            obecnyRozmiarCzcionki);
        }
        @Override
        public void onStartTrackingTouch(SeekBar suwak) {
        }
        @Override
        public void onStopTrackingTouch(SeekBar suwak) {
        }
    });
  }
}
```

Kod - wersja rozbudowana

Plik „activity_main.xml”

Info: Plik odpowiada za wygląd aplikacji i ułożenie elementów na ekranie smartfona. Lokalizacja pliku: *nazwaProjektu / app / src / main / res / layout / activity_main.xml*

```
<?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"
  android:id="@+id/main"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <!-- Pasek tytułu -->
  <com.google.android.material.appbar.MaterialToolbar
    android:id="@+id/pasekTytulu"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:background="@android:color/holo_blue_dark"
```

```
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent">

    <TextView
        android:id="@+id/tytul"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Suwak"
        android:textColor="@android:color/white"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_gravity="center"/>
</com.google.android.material.appbar.MaterialToolbar>

<!-- Przełącznik motywu. Zamiast Switch, lepiej użyć SwitchMaterial -->
<com.google.android.material.switchmaterial.SwitchMaterial
    android:id="@+id/zmianaMotywu"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Motyw Jasny / Ciemny"
    android:layout_marginTop="24dp"
    android:layout_marginEnd="16dp"
    app:layout_constraintTop_toBottomOf="@id/pasekTytulu"
    app:layout_constraintEnd_toEndOf="parent" />

<TextView
    android:id="@+id/przykladowyTekst"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Imię Nazwisko"
    android:textColor="#FFA500"
    android:textSize="24sp"
    android:layout_marginTop="40dp"
```

```
    app:layout_constraintTop_toBottomOf="@id/zmianaMotywu"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"/>

<!-- Suwak 1 - nieruchomy, jego pozycja jest stała i niezależna od wielkości tekstu -->
<!-- Atrybut max=30, gdyż od 10sp zaczyna, więc na końcu będzie 40sp -->
<!-- Atrybut progress="14" oznacza wartość początkową suwaka... ale + 10 daje 24sp -->
<!-- Później wartość "progress" jest nadpisywana z SharedPreferences -->
<SeekBar
    android:id="@+id/suwak1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:max="30"
    android:progress="14"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="8dp"
    android:layout_marginTop="190dp"
    android:progressDrawable="@drawable/seekbar_wlasny"
    android:thumb="@drawable/seekbar_kciuk_wlasny"
    app:layout_constraintTop_toBottomOf="@id/pasekTytulu"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toStartOf="@id/wielkoscCzcionki" />

<!-- Bieżąca wielkość czcionki -->
<TextView
    android:id="@+id/wielkoscCzcionki"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="24sp"
    android:textSize="16sp"
    android:layout_marginEnd="16dp"
    app:layout_constraintTop_toTopOf="@id/suwak1"
    app:layout_constraintBottom_toBottomOf="@id/suwak1"
    app:layout_constraintEnd_toEndOf="parent"/>
```

```
<View
    android:id="@+id/kolorowyKwadrat"
    android:layout_width="140dp"
    android:layout_height="140dp"
    android:background="#FF0000"
    android:layout_marginTop="60dp"
    app:layout_constraintTop_toBottomOf="@id/suwak1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"/>

<!-- Suwak 2 -->
<!-- Atrybut "max" oznacza pełną paletę kolorów HUE w zakresie 0-360*, -->
<!-- czyli od koloru czerwonego do czerwonego. Atrybut "progress" to wartość -->
<!-- początkowa (liczba rzeczywista), czyli kolor czerwony. -->
<SeekBar
    android:id="@+id/suwak2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:max="360"
    android:progress="0"
    android:layout_marginTop="24dp"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="62dp"
    android:progressDrawable="@drawable/seekbar_wlasny"
    android:thumb="@drawable/seekbar_kciuk_wlasny"
    app:layout_constraintTop_toBottomOf="@id/kolorowyKwadrat"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Plik „seekbar_wlasny.xml”

Info: Własny projekt suwaka (koloru żółtego). Położenie pliku: *nazwaProjektu / app / src / main / res / drawable / seekbar_wlasny.xml*

```
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
```

```
    <!-- Tło paska suwaka -->
    <item android:id="@android:id/background">
        <shape android:shape="rectangle">
            <size android:height="6dp"/>
            <solid android:color="#444444"/>
            <corners android:radius="30dp"/>
        </shape>
    </item>

    <!-- Pasek -->
    <item android:id="@android:id/progress">
        <clip>
            <shape android:shape="rectangle">
                <size android:height="6dp"/>
                <solid android:color="#FFD700"/>
                <corners android:radius="30dp"/>
            </shape>
        </clip>
    </item>
</layer-list>
```

Plik „seekbar_kciuk_wlasny.xml”

Info: Własny projekt elementu suwaka zwanego „kciukiem”. Położenie pliku: *nazwaProjektu / app / src / main / res / drawable / seekbar_kciuk_wlasny.xml*

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <size
        android:width="15dp"
        android:height="35dp"/>
    <solid android:color="#FF4500"/>
</shape>
```

Plik „MainActivity.java”

```
package com.example.nazwaAplikacji;


import android.content.SharedPreferences;
```

```
import android.os.Bundle;
import android.util.TypedValue;
import android.view.View;
import android.widget.SeekBar;
import android.widget.TextView;
import com.google.android.material.switchmaterial.SwitchMaterial;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatDelegate;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
    // Deklaracje prywatnych zmiennych:
    // Stała (final) wskazująca na plik "settings.xml" tworzony na smartfonie w chwili zapisywania
    // ustawień. Fizycznie plik znajduje się w takiej lokalizacji:
    // /data/user/0/com.example.javasuwak/shared_prefs/settings.xml
    // Jest ona jednak prywatna (sandbox), widoczna dopiero po zrootowaniu.
    private static final String nazwaPlikuSharedPreferences = "settings";
    // Zmienna obiektowa, interfejs (uchwyt) do odczytu i zapisu ustawień w SharedPreferences:
    private SharedPreferences uchwytSharedPreferences;
    private static final String kluczMotywCiemny = "dark";
    private static final String kluczPaletaBarwKwadratu = "hue";
    // Dodajemy klucz dla czcionki w SharedPreferences:
    private static final String kluczWielkoscCzcionki = "font_size";
    private static final int kluczMinimalnaWielkoscCzcionki = 10;
    private TextView przykladowyNapis;
    private TextView rozmiarCzcionki;
    private SeekBar suwakDoNapisu;
    // Zamiast "Switch przelacznikMotywow;", lepiej tak:
    private SwitchMaterial przelacznikMotywow;
    private SeekBar suwakDoKwadratu;
```

```
private View kwadrat;

// Poniżej, metoda pomocnicza (musi być poza onCreate). Tworzy kolor w palecie ARGB z koloru
// w palecie HSV i zwraca go jako typ integer (liczbę szesnastkową).
// float barwa - parametr wejściowy, gdzie "barwa" oznacza odcień/barwę w modelu HSV (0-
// 360) i przekazywana jest tutaj jako wartość suwaka (progress).
// Paleta HSV składa się z: HUE (barwa), Saturation (nasycenie), Value (jasność).
// Color.HSVToColor - konwertuje paletę HSV na ARGB (Alpha, Red, Green, Blue) jako wartość
// integer, ale w postaci liczby szesnastkowej, bo taką paletę ma Android (np. 0xFFFF0000, czyli
// czerwony).
// Tworzy tablicę float[] z trzema składnikami palety HSV: {barwaHue, pełneNasycenie,
// pełnaJasność}. Wartość "1f" oznacza liczbę typu float.
private int kolorPaletyHue(float barwa) {
    return android.graphics.Color.HSVToColor(new float[]{barwa, 1f, 1f});
}
// Alternatywny sposób do powyższego, ale mniej elegancki (umieszczany po onCreate):
// float[] hsv = new float[]{savedHue, 1f, 1f};
// int barwa = android.graphics.Color.HSVToColor(hsv);
// kwadrat.setBackgroundColor(barwa);

@Override
protected void onCreate(Bundle savedInstanceState) {
    uchwytSharedPreferences = getSharedPreferences(nazwaPlikuSharedPreferences,
    MODE_PRIVATE);
    boolean czyMotywCiemny = uchwytSharedPreferences.getBoolean(kluczMotywCiemny,
    false);
    int zapisanaWielkoscCzcionki =
    uchwytSharedPreferences.getInt(kluczWielkoscCzcionki, 24); // domyślnie 24sp
    int zapisanaBarwaHUE = uchwytSharedPreferences.getInt(kluczPaletaBarwKwadratu, 0);
    // KLUCZOWE: przed setContentView!
    // Aktywujemy motyw ciemny:
    AppCompatDelegate.setDefaultNightMode(
        // Jeśli wartość "czyMotywCiemny" jest "true" ustawia motyw ciemny (powoduje to
        // rekreację Activity). W przeciwnym wypadku ustawia motyw jasny.
```

```
        // Znak ? jest operatorem trójargumentowym (warunek ? wartość1 : wartość2).
        // Zastępuje on konstrukcję "if - else".
        // Czyli: Android wybiera tu odpowiednie zasoby: "res/values/themes.xml" lub
        // "res/values-night/themes.xml":
        czyMotywCiemny ? AppCompatDelegate.MODE_NIGHT_YES :
        AppCompatDelegate.MODE_NIGHT_NO
);
super.onCreate(savedInstanceState);
EdgeToEdge.enable(this);
setContentView(R.layout.activity_main);
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
    Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
    return insets;
});

przelacznikMotywow = findViewById(R.id.zmianaMotywu);
przykladowyNapis = findViewById(R.id.przykladowyTekst);
rozmiarCzcionki = findViewById(R.id.wielkoscCzcionki);
suwakDoNapisu = findViewById(R.id.suwak1);
suwakDoKwadratu = findViewById(R.id.suwak2);
kwadrat = findViewById(R.id.kolorowyKwadrat);
kwadrat.setBackgroundColor(kolorPaletyHue(zapisanaBarwaHUE));
// Poniżej, jawna deklaracja, że wielkość czcionki podana będzie w "sp" ("scale-independent
// pixels"), choć setTextSize(24) - też byłoby interpretowane jako jednostka "sp":
przykladowyNapis.setTextSize(TypedValue.COMPLEX_UNIT_SP, zapisanaWielkoscCzcionki);
rozmiarCzcionki.setText(zapisanaWielkoscCzcionki + "sp");

// Tu musimy trochę skomplikować, bo chcemy, aby najmniejszą wartością czcionki było 10sp.
// Więc, aby obliczyć progress suwaka, musimy minimalną wielkość odjąć od pożądanej
// wielkości.
// Więc aktualną pozycję suwaka obliczamy jako: 24sp - 10sp = 14. Można by to zrobić tak:
// suwakDoNapisu.setProgress(zapisanaWielkoscCzcionki - 10);
// Ale lepiej jest wstawić zabezpieczenie (liczba nie może być mniejsza od 0). Działa to tak:
```

```
// Math.max(0, -5) - w takim przypadku, wybiera liczbę większą, czyli 0, a nie -5:
suwakDoNapisu.setProgress(Math.max(0, zapisanaWielkoscCzcionki -
kluczMinimalnaWielkoscCzcionki));
// W przypadku tego suwaka, nic nie kombinujemy. Zarówno zakres barw, jak i zakres suwaka
// jest od 0 do 360:
suwakDoKwadratu.setProgress(zapisanaBarwaHUE);

// Obsługa MOTYWÓW
// Zablokowanie na chwilę Listenera, bo chcemy, aby motyw był najpierw
// wczytany z SharedPreferences (zanim uruchomi się Listener i cokolwiek zmieni):
przelacznikMotywow.setOnCheckedChangeListener(null);
// Przywracamy ostatni stan motywu z SharedPreferences:
przelacznikMotywow.setChecked(czyMotywCiemny);
// Ponownie uruchamiamy Listener. Od tej chwili, wszelka zmiana przełącznika uruchomi kod
// wewnątrz lambdy. Parametr "przelacznik1" to obiekt przełącznika typu SwitchMaterial
// i wskazuje na konkretny przełącznik (bo Listener może nasłuchiwać wiele przełączników).
// Innymi słowy, jest to umowna nazwa przełącznika i może być dowolna.
// Nazwa drugiego parametru "czyZmieniony" jest również umowna. Nie można jednak
// zmieniać ilości parametrów ani ich kolejności.
// Podsumowując: "przelacznik1" to źródło zdarzenia, "czyZmieniony" to nowy stan
// przełącznika.
przelacznikMotywow.setOnCheckedChangeListener((przelacznik1, czyZmieniony) -> {
    // Zapis nowego ustawienia w SharedPreferences (włączamy tryb edycji, po czym
    // wkładamy klucz z wartością boolowską (true or false), po czym zapisujemy (apply()):
    uchwytSharedPreferences.edit().putBoolean(kluczMotywCiemny, czyZmieniony).apply();
    AppCompatDelegate.setDefaultNightMode(
            // Operator warunkowy: if (czyZmieniony) { MODE_NIGHT_YES } else
            // { MODE_NIGHT_NO }
            czyZmieniony ? AppCompatDelegate.MODE_NIGHT_YES
                : AppCompatDelegate.MODE_NIGHT_NO
    );
});
```

```
// Obsługa SUWAKA nr 1.
// Listener zostaje podłączony do suwaka i czeka na działania użytkownika.
// Gdy użytkownik zmieni pozycję suwaka, zostanie uruchomiony wewnętrzny kod.
// Parametrem metody jest nowo utworzony obiekt anonimowy interfejsu Listenera: new
// SeekBar… Interfejs zawiera 3 metody:
// - onProgressChanged() - uruchamiana po przesunięciu suwaka,
// - onStartTrackingTouch() - dotknięcie suwaka,
// - onStopTrakingTouch() - puszczenie suwaka.
suwakDoNapisu.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
{
  // Override - oznacza, że metoda będzie nadpisywać interfejs nowymi wartościami,
  // czyli będzie nadpisywać metody z klasy:
  @Override
  // Nazwy parametrów metod są dowolne (ważny tylko jest ich typ i kolejność).
  // Nazwy metod zaś - nie są dowolne (są zdefiniowane w klasie).
  public void onProgressChanged(SeekBar suwak1, int progressSuwaka, boolean
  czyZmianaPrzezUzytkownika) {
    // Dodajemy tu tzw. offset, bo suwak działa od 0, a my chcemy w zakresie 10-40sp.
    int obecnyRozmiarCzcionki = progressSuwaka + kluczMinimalnaWielkoscCzcionki;
    przykladowyNapis.setTextSize(TypedValue.COMPLEX_UNIT_SP,
    obecnyRozmiarCzcionki);
    rozmiarCzcionki.setText(obecnyRozmiarCzcionki + "sp");
  }
  // Poniżej, co ma zrobić aplikacja, jak użytkownik dotknie suwak
  // i zacznie przesuwać (np. zatrzymuje inne działania).
  // W poniższym przykładzie nic nie robi.
  @Override
  public void onStartTrackingTouch(SeekBar suwak1) {
  }
  // Co aplikacja ma zrobić, gdy użytkownik ustawi suwak
  // i nie będzie go już dotykał (np. ma zapisać ustawioną wartość w SharedPreferences):
  @Override
  public void onStopTrackingTouch(SeekBar suwak1) {
    // Zapis wielkości czcionki:
```

```
            int obecnyRozmiarCzcionki = suwak1.getProgress() + kluczMinimalnaWielkoscCzcionki;
            uchwytSharedPreferences.edit().putInt(kluczWielkoscCzcionki,
            obecnyRozmiarCzcionki).apply();
        }
    });


    // Obsługa SUWAKA nr 2
    // Podobnie, jak w przypadku poprzedniego suwaka, zostaje utworzony obiekt Listenera
    // zawierający 3 predefiniowane metody (nazwy parametrów są dowolne).
    suwakDoKwadratu.setOnSeekBarChangeListener(new
    SeekBar.OnSeekBarChangeListener() {
        @Override
        // Co aplikacja ma zrobić w przypadku zmiany pozycji suwaka:
        public void onProgressChanged(SeekBar suwak2, int progressSuwaka2, boolean
        czyZmianaPrzezUzytkownika) {
            kwadrat.setBackgroundColor(kolorPaletyHue((float) progressSuwaka2));
        }
        // Co aplikacja ma zrobić w przypadku dotknięcia suwaka przez użytkownika (nic!):
        @Override public void onStartTrackingTouch(SeekBar suwak2) {}
        // Co aplikacja ma zrobić, gdy użytkownik puści suwak (ma zapisać wartość barwy palety
        // HSV). Zapis nastąpi do prywatnego pliku aplikacji "settings.xml" (sharedPreferences):
        @Override public void onStopTrackingTouch(SeekBar suwak2) {
            // Można by to zrobić tak:
            // int progressSuwaka2 = suwak2.getProgress();
            // uchwytSharedPreferences.edit().putInt(kluczPaletaBarwKwadratiu,
            // progressSuwaka2).apply();
            // Ale lepiej zrobić to tak:
            uchwytSharedPreferences.edit().putInt(kluczPaletaBarwKwadratu,
            suwak2.getProgress()).apply();
        }
    });
  }
} // Koniec MainActivity
```

Uwaga: Aby otrzymać ocenę celującą, należy dodatkowo:

- utworzyć własną ikonę aplikacji;
- wykonać cały projekt bez pomocy nauczyciela w ciągu pierwszego tygodnia;
- ustawić nazwę aplikacji na „JavaSuwak” (nazwa znajdująca się pod ikoną w smartfonie).

Ostatnia aktualizacja: 18 maja 2026.