Moja pierwsza aplikacja mobilna w Android Studio (Java+XML)



Warunki wstępne

1. Smartfon: włączyć *Ustawienia / Opcje programisty**:

✔ Debugowanie USB;

✔ Debugowanie bezprzewodowe (*Wireless Debugging*) / *Pair Device using QR Code (Android > 11, dostęp do Wi-Fi)*

✔ Zezwalaj na instalacje z nieznanych źródeł.

2. BIOS/UEFI: włączyć opcję *Virtual Technology* (*VT / VT-x / VMX / SVM*).

3. Windows (tylko dla AMD): instalacja funkcji *hypervizor'a WHPX* (*Panel sterowania / Programy i funkcje / Windows Hypervisor Platform*). Funkcja Hyper-V może być włączona.

* Jeśli „*Opcje programisty*" są niewidoczne, należy kliknąć 7 razy w: *Ustawienia / Telefon - informacje (About phone) / Numer wersji kompilacji (Build Number)*.

Warunki wstępne

4. Windows: instalacja *Microsoft Visual C++ Redistributable* (odpowiada za emulator).

5. Windows: instalacja *JDK* (lub *openJDK*).

6. Windows: *Panel sterowania / Programy i funkcje / Włącz lub wyłącz funkcje systemu Windows: Platforma funkcji Hipervisor systemu Windows, Platforma maszyny wirtualnej*.

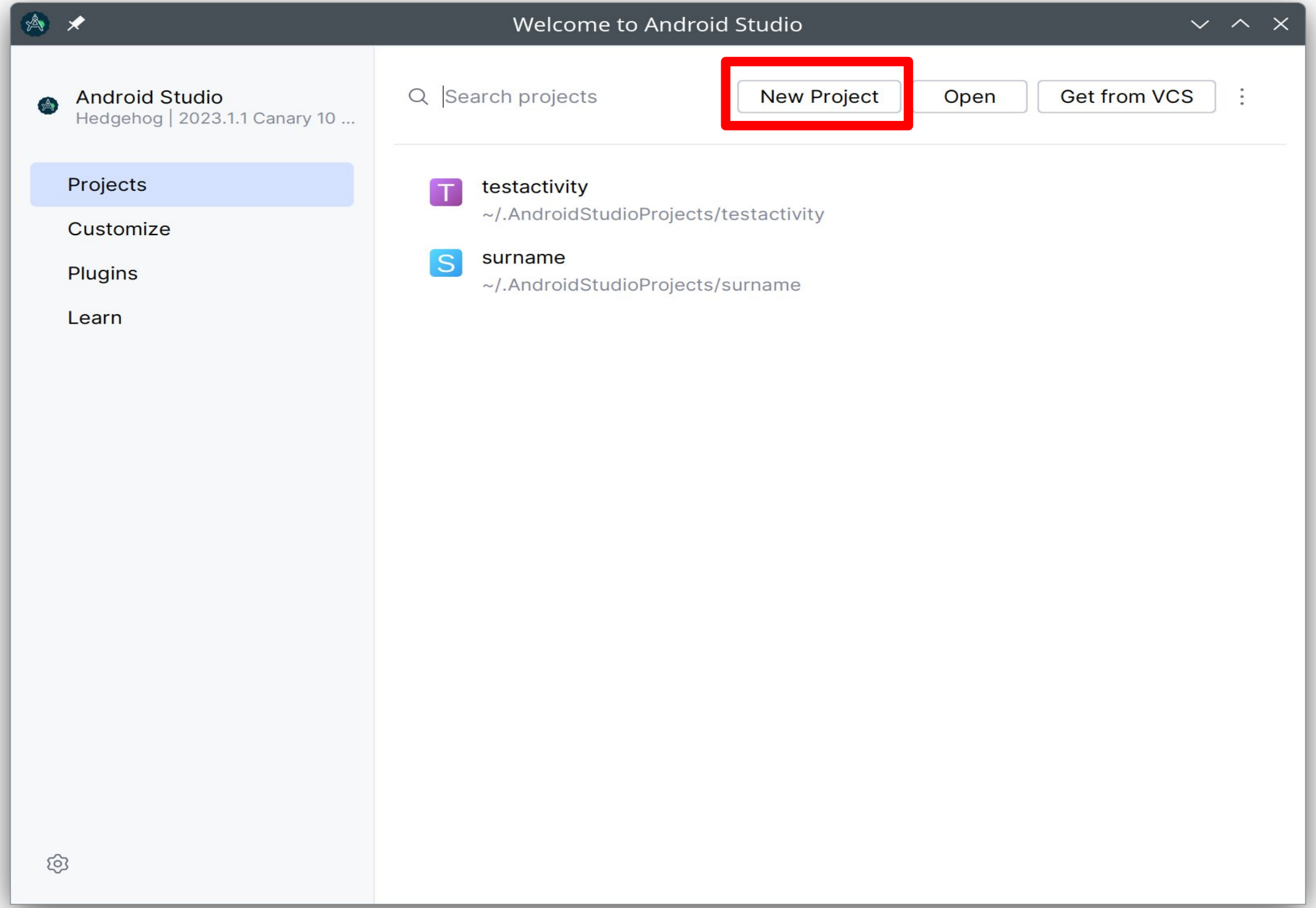
7. Android Studio (tylko Intel i Windows): instalacja funkcji hypervisor'a Androida AEHD (*Tools / SDK Manager / SDK Tools: Android Emulator Hypervisor Driver*). Funkcja Hyper-V musi być wyłączona.

Uwaga: **HAXM** przestał być wspierany przez Intel. Zamiast niego procesory *Intel* używają wbudowanej w *„Android Studio"* funkcji *„Android Emulator Hypervisor"* (*Tools / SDK Manager / SDK Toos*).

Warunki wstępne

8. Windows: instalacja *Android Studio* (ścieżka dostępu do konta systemowego musi spełniać zasadę 8.3). Dostęp do Internetu nie może przebiegać przez proxy ani przez OSE (w przypadku Polski).

9. Instalacja *Android Studio* jest ukończona wtedy, gdy na koncie administratora uruchomiono w emulatorze domyślne projekty w Kotlin i Java, a następnie to samo na koncie z ograniczeniami.

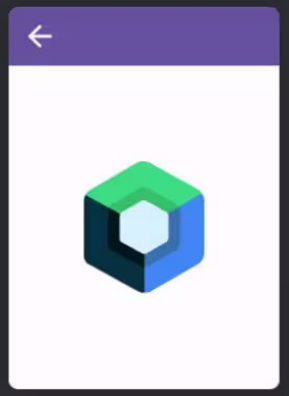

Welcome to Android Studio
Android Studio
Hedgehog | 2023.1.1 Canary 10 ...
Projects
Customize
Plugins
Learn
Search projects
New Project
Open
Get from VCS
testactivity
~/.AndroidStudioProjects/testactivity
surname
~/.AndroidStudioProjects/surname

Templates

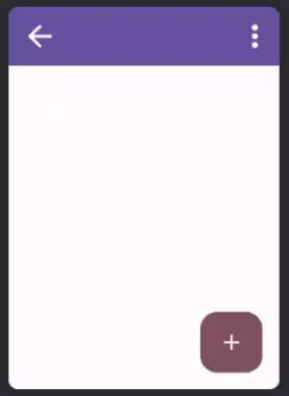
- Phone and Tablet
- Wear OS
- Television
- Automotive



No Activity



Empty Activity



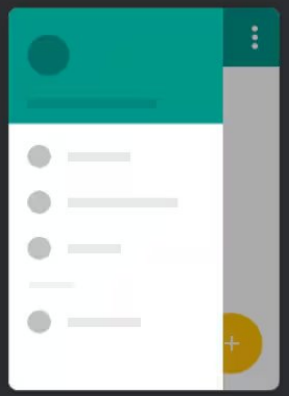
Basic Views Activity



Bottom Navigation Views A...

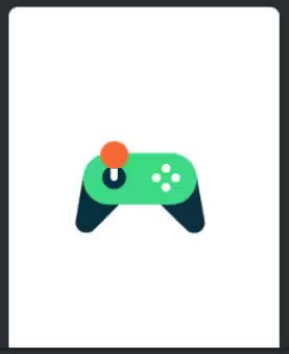


Empty Views Activity



Navigation Drawer Views A...







Previous | Next | Cancel | Finish

Empty Views Activity

Creates a new empty activity

Zapamiętaj!

Name: Gra w kosci Java

Package name: com.example.grawkoscijava

Save location: /home/robert/.AndroidStudioProjects/GrawkosciJava

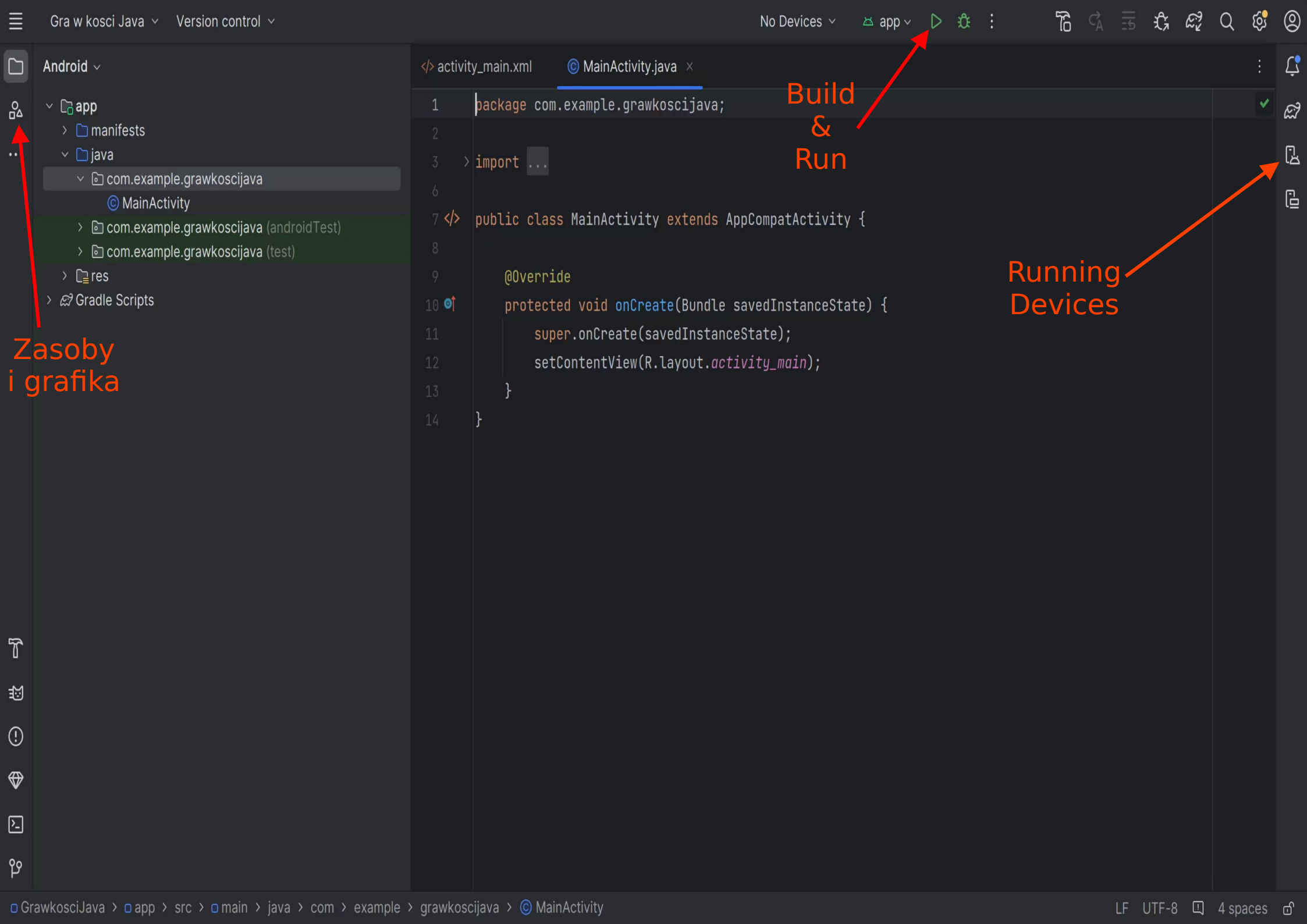
Language: Java

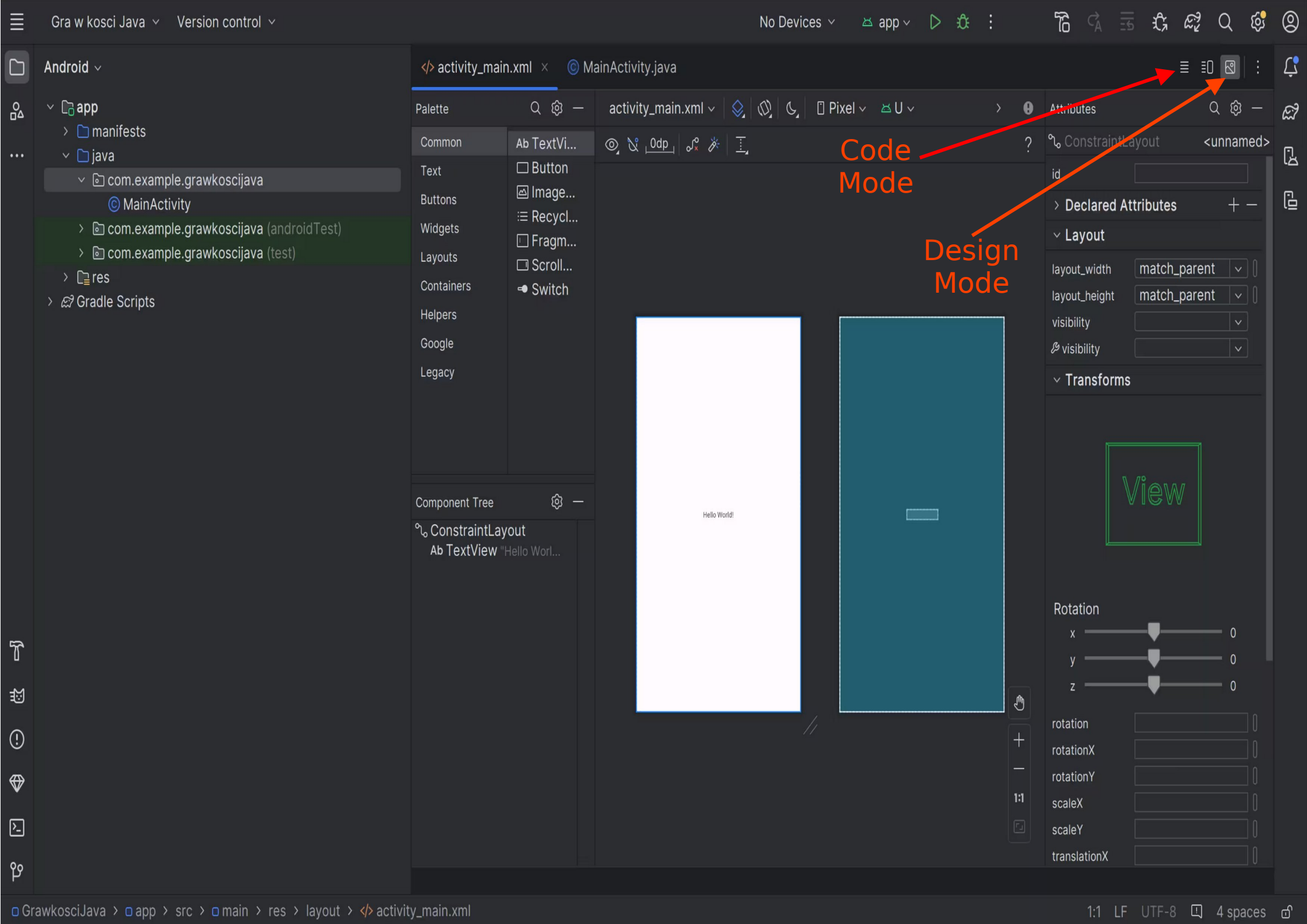
Minimum SDK: API 26 ("Oreo"; Android 8.0)

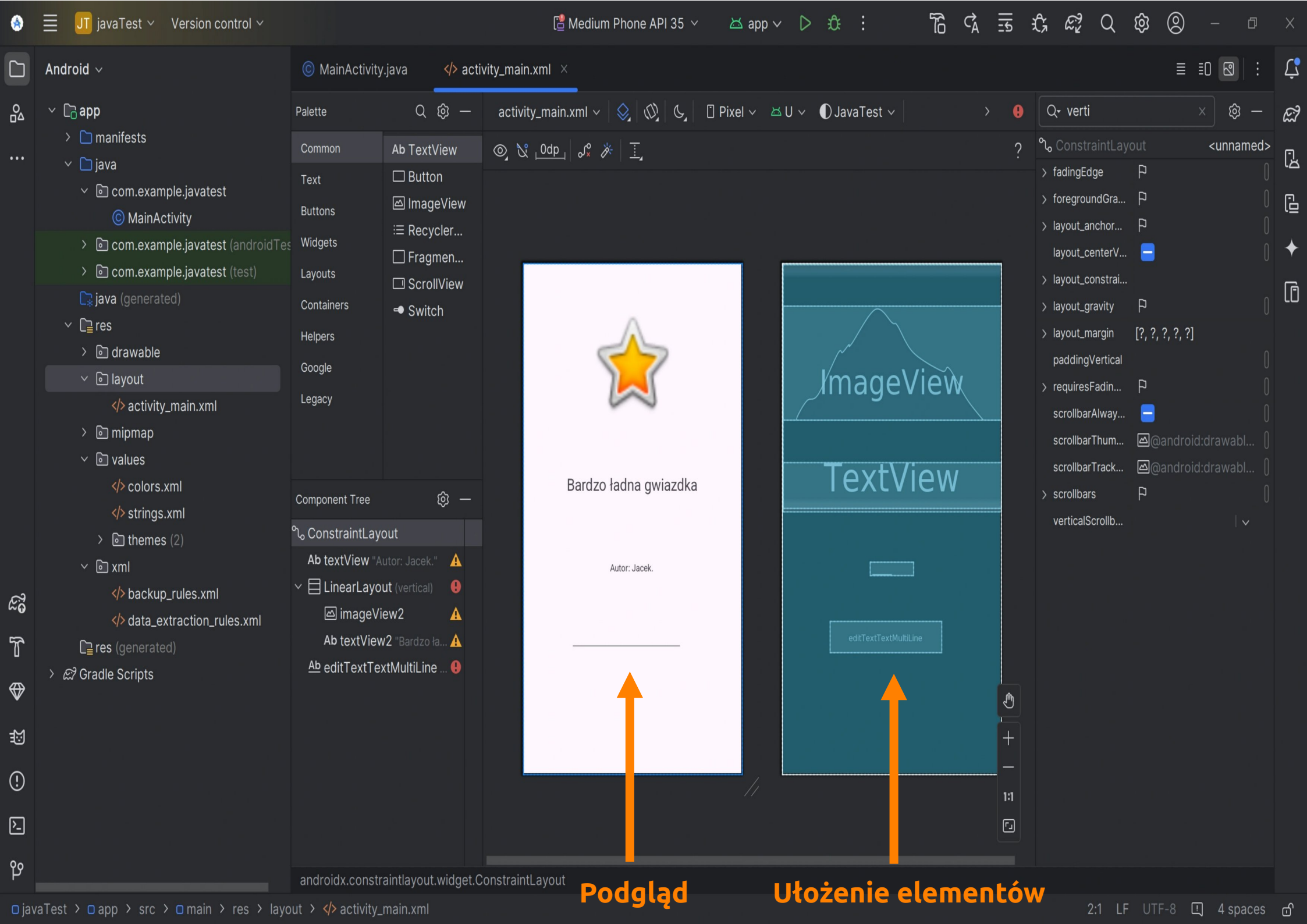
Your app will run on approximately **95,4%** of devices.
Help me choose

Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

Previous | Next | Cancel | Finish


Gra w kosci Java
Version control
No Devices
app
Android
app
manifests
java
com.example.grawkoscijava
MainActivity
com.example.grawkoscijava (androidTest)
com.example.grawkoscijava (test)
res
Gradle Scripts
activity_main.xml
MainActivity.java
package com.example.grawkoscijava;
import ...
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
}
Build & Run
Running Devices
Zasoby i grafika
GrawkosciJava
app
src
main
java
com
example
grawkoscijava
MainActivity
LF
UTF-8
4 spaces


Gra w kosci Java
Version control
No Devices
app
Android
activity_main.xml
MainActivity.java
app
manifests
java
com.example.grawkoscijava
MainActivity
com.example.grawkoscijava (androidTest)
com.example.grawkoscijava (test)
res
Gradle Scripts
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
TextVi...
Button
Image...
Recycl...
Fragm...
Scroll...
Switch
Component Tree
ConstraintLayout
TextView "Hello Worl...
Pixel
Hello World!
Code Mode
Design Mode
Attributes
ConstraintLayout
<unnamed>
id
Declared Attributes
Layout
layout_width
match_parent
layout_height
match_parent
visibility
visibility
Transforms
View
Rotation
x
y
z
rotation
rotationX
rotationY
scaleX
scaleY
translationX
GrawkosciJava
app
src
main
res
layout
activity_main.xml
1:1
LF
UTF-8
4 spaces


javaTest
Version control
Medium Phone API 35
app
Android
MainActivity.java
activity_main.xml
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
Ab TextView
Button
ImageView
Recycler...
Fragmen...
ScrollView
Switch
Component Tree
ConstraintLayout
Ab textView "Autor: Jacek."
LinearLayout (vertical)
imageView2
Ab textView2 "Bardzo ła...
Ab editTextTextMultiLine ...
Pixel
JavaTest
Bardzo ładna gwiazdka
Autor: Jacek.
ImageView
TextView
editTextTextMultiLine
verti
ConstraintLayout
<unnamed>
fadingEdge
foregroundGra...
layout_anchor...
layout_centerV...
layout_constrai...
layout_gravity
layout_margin
[?, ?, ?, ?, ?]
paddingVertical
requiresFadin...
scrollbarAlway...
scrollbarThum...
@android:drawabl...
scrollbarTrack...
@android:drawabl...
scrollbars
verticalScrollb...
androidx.constraintlayout.widget.ConstraintLayout
Podgląd
Ułożenie elementów
javaTest
app
src
main
res
layout
activity_main.xml
2:1
LF
UTF-8
4 spaces

Layouts

<ConstraintLayout> - **domyślny**, bez zagnieżdżania (połączenie linkami);

<LinearLayout> - wyświetla elementy w jednym wierszu lub kolumnie;

<GridLayout> - układ siatki, zawiera komórki do umieszczania elementów (jak w arkuszu kalkulacyjnym), możliwe jest łączenie komórek (*rowSpan, columnSpan*);

Przestarzałe:

<FrameLayout> - umożliwia zachodzenie elementów na siebie (np. tekst na obraz, wskaźnik postępu na tle innych elementów);

<RelativeLayout> - elementy umieszczane względem innych elementów (przestarzały, zastąpiony przez ConstraintLayout);

<TableLayout> - forma tabeli, każdy wiersz reprezentowany jest przez *<TableRow>…</TableRow>*;

Layouts

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout

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

   xmlns:app="http://schemas.android.com/apk/res-auto"

   xmlns:tools="http://schemas.android.com/tools"

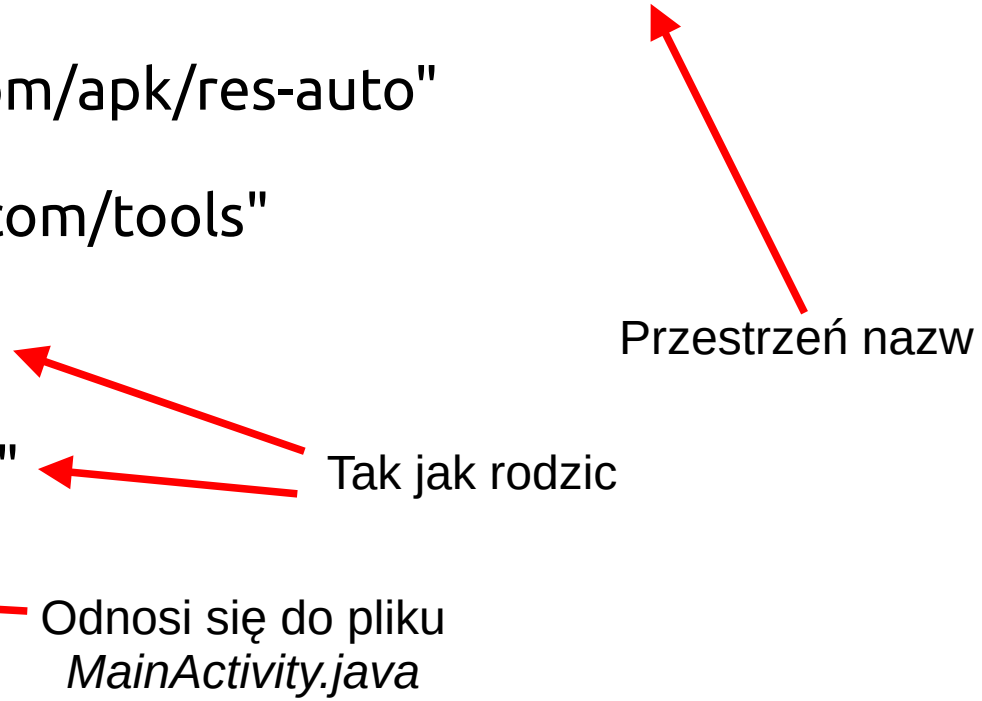
   android:layout_width="match_parent"

   android:layout_height="match_parent"

   tools:context=".MainActivity">

...

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



```
// Każdy element w <ConstraintLayout> powinien być zaczepiony:
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

<TextView
    android:id="@+id/napis"
    android:text="Hello Kitty!"
    // Górna krawędź tekstu zaczepiona do górnej krawędzi rodzica:
    app:layout_constraintTop_toTopOf="parent"
    // lub
    app:layout_constraintTop_toBottomOf="@id/jakiśElement"
    // Margines od tej górnej krawędzi:
    android:layout_marginTop="16dp"
    // Lewa krawędź elementu zaczepiona do lewej krawędzi rodzica:
    app:layout_constraintStart_toStartOf="parent"
    // Prawa krawędź elementu zaczepiona do prawej krawędzi rodzica:
    app:layout_constraintEnd_toEndOf="parent" />

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

Wielkość elementów

wrap_content - dopasowanie wielkości do zawartości

match_parent - dopasowanie wielkości do nadrzędnego elementu, wypełnia całą dostępną przestrzeń rodzica

gravity - położenie obiektu w układzie

layout_marginTop / **layout_marginBottom** - odstęp między layout'ami

Style

Plik dla stylów:

*nazwaProjektu/app/src/main/res/values/themes/**themes.xml***

(lub stary sposób: *nazwaProjektu/app/src/main/res/values/* PPM: *New/Value Resource file: **styles.xml***)

Zawartość pliku:

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <style name="nazwaNowegoStylu">
        <item name="android:textSize">16sp</item>
        <item name="android:textStyle">bold</item>
    </style>
</resources>
```

Zastosowanie stylów:

```
<TextView
    style="@style/nazwaNowegoStylu" />
```

Style - dziedziczenie

Kiedy stosować dziedziczenie?

- gdy chcemy zastosować **predefiniowany** (domyślny) styl i zmienić w nim tylko wybrane elementy (metoda zalecana).

- gdy chcemy stworzyć swój rozbudowany styl i podzielić go na części dla wygody używania.

Style - dziedziczenie

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Poniżej: Wczytaj styl od rodzica (parent), a następnie nadpisz jego niektóre elementy
    poniższymi definicjami. Aby ten styl został zastosowany, należy się do niego odwołać w pliku
    AndroidManifest.xml -->
    <style name="Base.Theme.nazwaNowegoStylu" parent="Theme.Material3.DayNight">
        <item name="android:textColor">#ffff00</item> <!-- Domyślny kolor tekstu w aplikacji -->
        <item name="android:textSize">16sp</item> <!-- Domyślny rozmiar czcionki w aplikacji -->
        <!-- Poniżej, styl, który dotyczy tylko menu. Nazwa "popupTheme" jest predefiniowana -->
        <item name="popupTheme">@style/CzerwoneMenu</item>
    </style>

    <!-- Tworzymy styl wyłącznie dla menu -->
    <style name="CzerwoneMenu">
        <item name="android:textColor">#ff0000</item>
    </style>
</resources>
```

Style - aktywacja nowego stylu

Plik „AndroidManifest.xml”:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools">

    <!-- Poniżej, domyślnym szablonem aplikacji jest: android:theme="@style/Theme.Material3.Dark.NoActionBar"
    Można go jednak zastąć naszym własnym zmodyfikowanym szablonem, zdefiniowanym w pliku values-night/themes.xml:
    android:theme="@style/Base.Theme.JavaPasekTytulu" -->
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Base.Theme.nazwaNowegoStylu"
        tools:targetApi="34">
        ...
    </application>

</manifest>
```

Style - Tworzenie klasy (sposób 1)

```
<resources xmlns:tools="http://schemas.android.com/tools">

    <style name="nazwaNowegoStylu.nazwaKlasy">

        <item name="android:textSize">18sp</item>

        <item name="android:textStyle">normal</item>

    </style>

</resources>
```

Style - Tworzenie klasy (sposób 2)

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <style name="nazwaKlasy" parent="nazwaNaszegoStylu">
        <item name="android:textSize">18sp</item>
        <item name="android:textStyle">normal</item>
    </style>
</resources>
```

```
<TextView
    style="@style/nazwaNaszegoStylu.nazwaKlasy" />
```

Theme (szablon)

Plik: *nazwaProjektu/app/src/main/**AndroidManifest.xml***

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:theme="@style/Theme.Dark"> (styl całej aplikacji)
        <activity android:name=".MainActivity"
            android:theme="@style/Theme.Light"> (styl pojedynczego Activity)
        </activity>
    </application>
</manifest>
```

AVD (*Android Virtual Device*)

=

wirtualny smartfon

Emulatory smartfonów

Działające, przetestowane:

- **Nexus S**
- **Pixel**
- **Pixel XL**
- **Pixel 2**
- **Pixel 3**
- **Pixel 3a**
- **Pixel 4**
- **Pixel 4 XL**
- **Pixel 5**
- **Pixel 6**
- **Pixel 6 Pro**
- **Pixel 7**
- **Pixel 7 Pro**

Emulatory smartfonów

- API do emulatora powinna „ważyć” ok. **8**-**10**GB. Jeśli mniej - to coś jest nie tak.

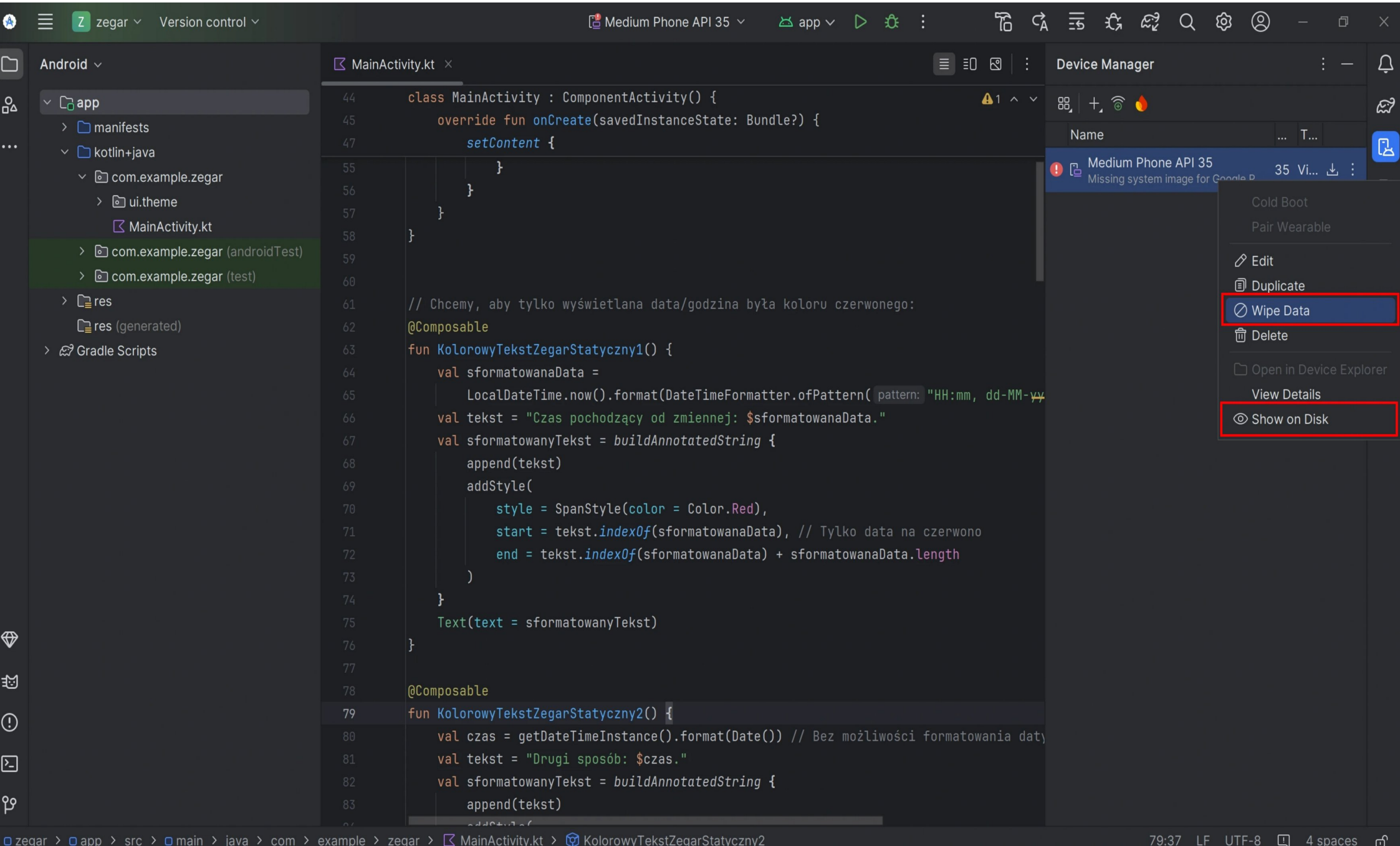
- Uruchamiamy tylko jeden emulator w tym samym czasie. W zakładce „*Running Devices*” sprawdzamy, czy poprzednio uruchamiany emulator został zamknięty.

- Zielona strzałka ► - buduje projekt, wysyła go na smartfon i uruchamia go na smartfonie.

- Gdy wyskoczy okienko z informacją o debuggerze - zamykamy je (*Cancel*).

- Na uruchomienie aplikacji na smartfonie czekamy maksymalnie 5 minut.

Czyszczenie danych cache / Lokalizacja emulatora



Emulatory smartfonów

Co jeśli *Android Studio* nie potrafi uruchomić emulatora?

Wiersz poleceń (cmd):

cd ***AppData\Local\Android\Sdk\emulator***

emulator **-accel-check** (czy akceleracja jest włączona)

emulator **-list-avds** (wykaz wirtualnych smartfonów)

emulator **-avd** ***nazwaEmulatora***

Emulatory smartfonów

Co jeśli emulator jest bardzo powolny?

emulator **-avd** ***nazwaEmulatora*** ***-gpu*** *opcja*

Opcje:

auto - system sam wybiera

host - akceleracja sprzętowa

swiftshader_indirect - akceleracja programowa *OpenGL* (*Linux*)

angle_indirect - akceleracja programowa *Microsoft Direct* (*Windows*)

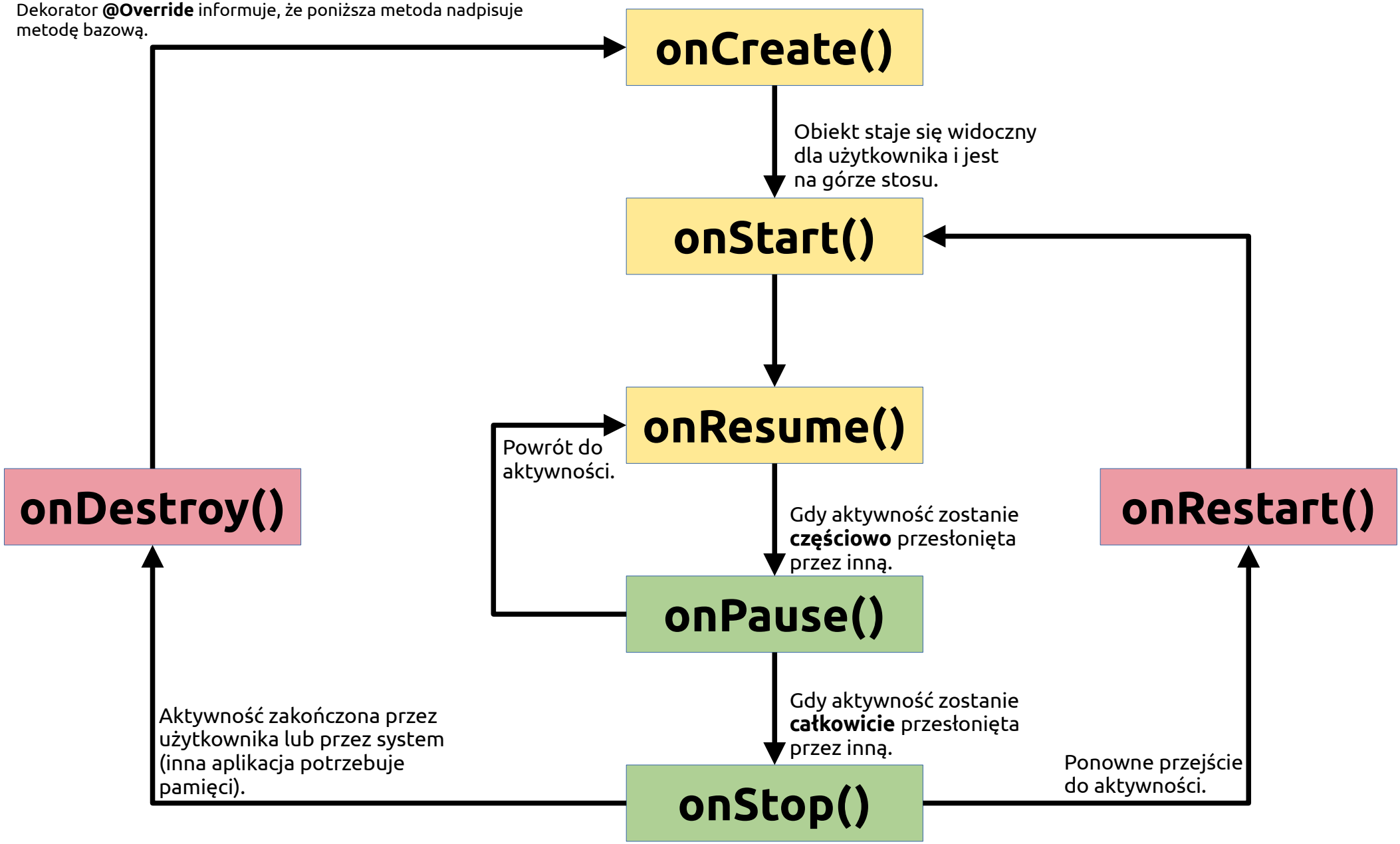
guest - akceleracja programowa po stronie gościa (emulatora smartfona)

Activity = **Okno**

Poszczególnym *activities* możemy nadawać własne nazwy (inne niż *MainActivity*).

Cykle życia Activity

Wyrażenie **super** służy do wywoływania metod z klasy bazowej.
Dekorator **@Override** informuje, że poniższa metoda nadpisuje metodę bazową.



Budowa aplikacji - kod

- Obiekt **Bundle** przechowuje poprzedni stan aktywności i dziedziczy (*extends*) po klasie **AppCompatActivity** (obsługa *action bar*), a ta po klasie **Activity**.
- Wyrażenie **super** służy do wywoływania metod z klasy bazowej.
- Dekorator **@Override** informuje, że poniższa metoda nadpisuje metodę bazową.

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Plik *activity_main.xml*

- Odpowiada za wygląd aplikacji (elementy na ekranie).

- Lokalizacja: *nazwaProjektu / src / main / res / layout / **activity_main.xml*** (Uwaga: W niektórych sytuacjach nazwa pliku to *content_main.xml*. Edycja pliku posiada dwie zakładki: **Design** [tryb WYSIWYG] oraz **Text** [tryb tekstowy]).

- Sekcje:

```
<LinearLayout android:orientation="horizontal">
    <!-- Elementy wewnątrz tej sekcji będą wyświetlane w jednym wierszu (row). -->
</LinearLayout>

<LinearLayout android:orientation="vertical">
    <!-- Elementy wewnątrz tej sekcji będą wyświetlane jedne pod drugimi (column). -->
</LinearLayout>
```

Uwaga: Sekcje <*LinearLayout*> mogą zagnieżdżać się wewnątrz siebie.

Plik *MainActivity.java*

- Odpowiada za to, co aplikacja ma robić (zawiera kod w języku Java).

- Lokalizacja: *nazwaProjektu / app / src / main / java / com / example / nazwaProjektu / **MainActivity.java***

- Aplikacja może mieć wiele *activities* (stron). PPM: *app / New / Activity*

- Nadrzędnym składnikiem kodu jest **klasa** wewnątrz której znajdują się **metody**.

- Jeśli metoda zwraca wynik, musi zawierać typ danych i na końcu **return**. Jeśli nie zwraca, musi zawierać typ **void**.

Plik *MainActivity.java*

Modyfikatory metod:

- **public** - wszystkie klasy mają dostęp do pól danych i metod *public*;
- **private** - dostęp do metod i danych posiadają jedynie inne metody tej samej klasy;
- **protected** - dostęp mają metody tej samej klasy i jej pochodne;
- **static** – wywołanie metody musi być poprzedzone nazwą jej klasy, np. *nazwaKlasy.nazwaMetody()*;
- **@override** – zastąpienie (nadpisanie) poprzedniego działania (poprzedniej *Activity*) lub domyślnego zachowania.

Gdzie jest mój plik *.apk ?

- Lokalizacja: *nazwaProjektu / app / build / outputs / apk / debug /* ***app-debug.apk***

- Budowanie pliku *.apk: menu *Build / Build Bundle(s) / APK(s) / Build APK(s)*

Jeśli działa - niewiele uczysz się!

Jeśli nie działa - tu dopiero zaczyna się edukacja!

