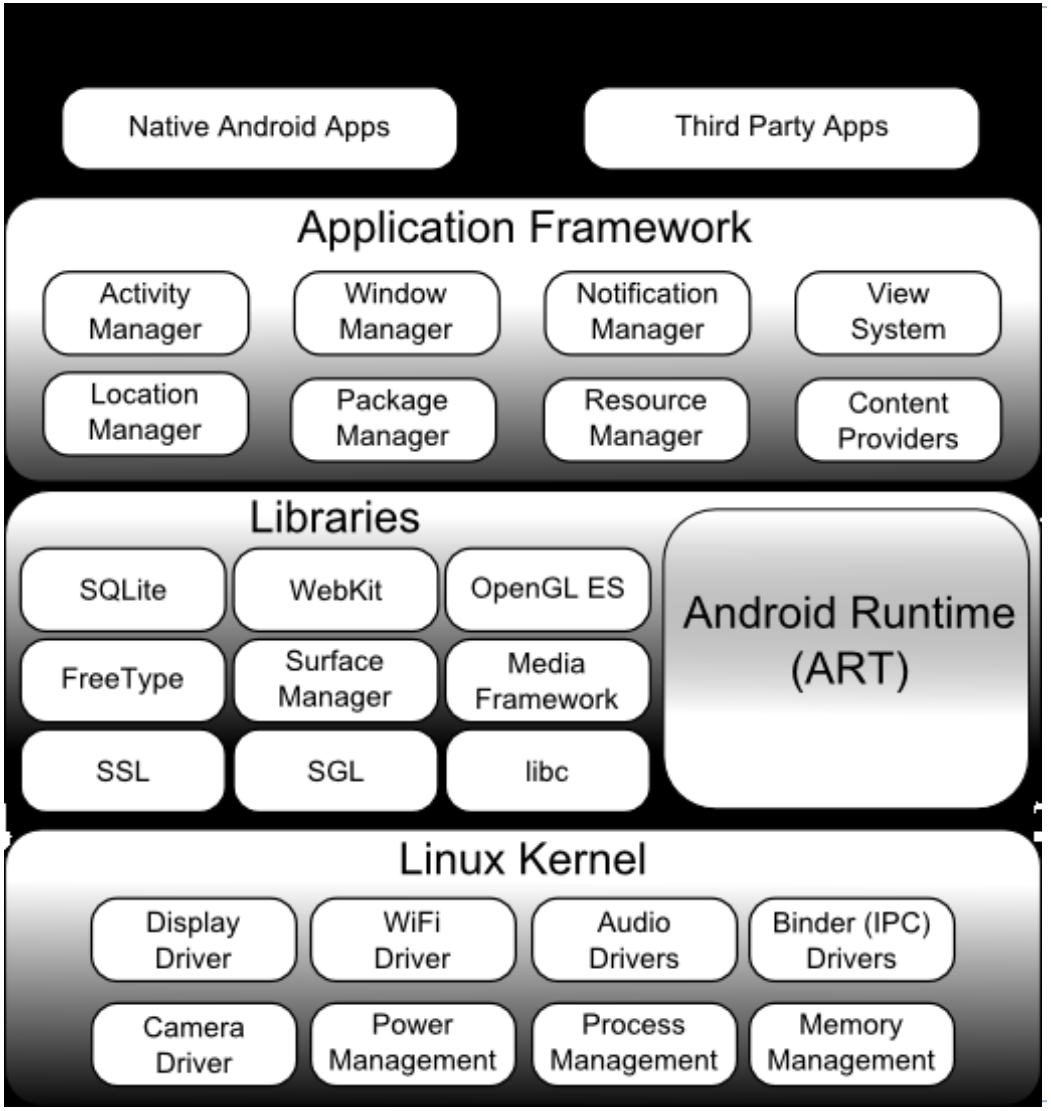
Програмування мобільних пристроїв

Налаштування середовища розробки мобільних застосунків та створення простого Android-застосунка

Слайди до лекцій (2 змістовий модуль)

Програмний стек платформи Android



Програмний стек платформи Android

- ***Ядро Linux* (*Linux Kernel*)** - забезпечує рівень абстракції між апаратним забезпеченням пристрою та верхніми рівнями програмного стеку Android. Ядро забезпечує превентивну багатозадачність, низькорівневі базові системні служби, такі як керування пам'яттю, процесами та живленням, а також мережевий стек і драйвери для дисплею, WiFi, аудіо тощо.
- ***Android Runtime* (*ART*)**. Коли програма Android створюється в Android Studio, вона компілюється у формат проміжного байт-коду (відомий як формат *DEX – Dalvik Executable*). Коли програма згодом завантажується на пристрій, *Android Runtime* (*ART*) використовує процес, званий компіляцією Ahead-of-Time (AOT), для перетворення файлу з DEX байт-кодом до файлу з інструкціями процесора пристрою у форматі Executable and Linkable Format (ELF). Кожного разу після запуску програми запускається ELF-файл, що забезпечує швидшу роботу програми та подовжує час роботи батареї.

Програмний стек платформи Android

- ***Бібліотеки Android (Libraries).*** На додаток до стандартних бібліотек Java, платформа Android містить *бібліотеки Android, основні:*
  - *android.app* – надає доступ до моделі програми та є базісом для усіх програм Android;
  - *android.content* – забезпечує доступ до вмісту, публікацію та обмін повідомленнями між програмами та компонентами програми;
  - *android.database* – використовується для доступу до даних, опублікованих постачальниками вмісту, і включає засоби керування базою даних SQLite;
  - *android.graphics* – API малювання двовимірної графіки низького рівня, включаючи кольори, точки, фільтри, прямокутники та полотна;
  - *android.hardware* –API, що надає доступ до апаратного забезпечення, такого як акселерометр, датчик освітлення тощо;
  - *android.opengl* – інтерфейс Java для API відтворення 3D-графіки OpenGL ES (OpenGL for Embedded Systems);

Програмний стек платформи Android

- *android.os* – надає застосункам доступ до стандартних служб операційної системи, включаючи повідомлення, системні служби та зв’язок між процесами;
- *android.media* – надає засоби для відтворення аудіо та відео;
- *android.net* – набір API, що надають доступ до мережевого стеку, включає *android.net.wifi*, який забезпечує доступ до бездротового мережевого стека пристрою;
- *android.print* – містить засоби, які дозволяють виконувати друк з застосунків Android;
- *android.provider* – набір допоміжних засобів, які надають доступ до стандартних баз даних постачальників вмісту Android, таких як ті, що підтримуються програмами календаря та контактів;
- *android.text* – засоби для візуалізації та обробки тексту на дисплеї пристрою;
- *android.util* – набір службових класів для виконання таких завдань, як перетворення рядків і чисел, обробка XML та маніпулювання датою й часом;

Програмний стек платформи Android

- *android.view* – основні будівельні блоки інтерфейсу користувача програми;
- *android.widget* – колекція компонентів інтерфейсу користувача, таких як кнопки, написи, списки, перемикачі, менеджери макетів тощо;
- *android.webkit* – набір класів для реалізації у застосунах можливості веб-перегляду.

основні бібліотеки по суті, є "обгортками" Java навколо набору бібліотек на основі C/C++.

- ***Application Framework*** – це набір служб, які разом утворюють середовище виконання Android-застосунків. Цей фреймворк реалізує концепцію багаторазового використання взаємозамінних компонентів у Android-застосунках. Платформа Android включає такі ключові служби:
  - *Activity Manager* – контролює всі аспекти життєвого циклу програми та стек виклику активностей;
  - *Content Providers* – дозволяє застосункам публікувати та обмінюватися даними з іншими застосунками;

Програмний стек платформи Android

- *Resource Manager* – надає доступ до таких ресурсів як рядки, кольори і макети інтерфейсу користувача без використання коду;
- *Notifications Manager* – дозволяє програмам відображати попередження та сповіщення для користувача;
- *View System* – розширюваний набір представлень екрану, який використовується для створення інтерфейсу користувача;
- *Package Manager* – система, за допомогою якої програми можуть отримувати інформацію про інші встановлені програми;
- *Telephony Manager* – надає застосунку інформацію про телефонні послуги, доступні на пристрої, наприклад, інформацію про абонента.
- *Location Manager* – надає доступ до служб визначення місцезнаходження, дозволяючи програмі отримувати оновлення при зміні розташування пристрою.

- *Рівень застосунків* (*Applications*) знаходиться на вершині програмного стеку Android і містить як програми, що надаються з конкретною реалізацією Android (наприклад, веб-браузер і програми електронної пошти), так і сторонні програми, встановлені користувачем після придбання пристрою.

Основні компоненти Android-застосунку

Основні компоненти Android-застосунку

- ***Активності (Activity)*** – це окремі модулі, що реалізують частину (або всі) функціональні можливості програми, зазвичай пов'язані безпосередньо з одним екраном інтерфейсу користувача. Є підкласами *android.app.Activity* і мають бути реалізовані повністю незалежними від інших активностей.
- ***Фрагменти* (*Fragments*)** - альтернатива активностям керують частинами екрану. При цьому активність стає контейнером, у який вбудовано один або декілька фрагментів – нащадків класу *androidx.fragment.app.Fragment*.
- ***Інтенти* (*Intents*)** – це механізм, за допомогою якого одна активність може запускати іншу та реалізовувати потік дій, які складають програму. Інтенти складаються з опису операції, яку потрібно виконати, і, за бажанням, даних, на основі яких операція має бути виконана.
- ***Широкомовні інтенти* (*Broadcast Intents*)** надсилаються всім програмам, які зареєстрували "зацікавлений" Broadcast Receiver. Система Android зазвичай надсилає Broadcast Intents, щоб вказати зміни стану пристрою, такі як завершення запуску системи, підключення зовнішнього джерела живлення до пристрою тощо.

Основні компоненти Android-застосунку

- ***Широкомовні отримувачі* (*Broadcast Receivers*)** – це механізм, за допомогою якого програми можуть відповідати на широкомовні інтенти. Широкомовний отримувач має бути зареєстрований програмою та налаштований за допомогою фільтра інтентів, щоб вказати типи широкомовної комунікації, які його цікавлять. Коли передається відповідний інтент, отримувач буде викликаний середовищем виконання Android незалежно від того, чи запущено зараз програму, яка зареєструвала отримувача.
- ***Служби* (*Services*)** – це процеси, які працюють у фоновому режимі та не мають інтерфейсу користувача. Їх можна запустити та керувати ними з активностей, широкомовних отримувачів або інших служб. Хоча служби не мають інтерфейсу користувача, вони все одно можуть сповіщати користувача про події за допомогою сповіщень і тостів (невеликі сповіщення, які з’являються на екрані, не перериваючи поточну активність), а також можуть створювать інтенти. Середовище виконання Android надає службам вищий пріоритет, аніж іншим процесам, і система зупиняє їх лише у крайньому випадку, щоб звільнити ресурси.

Основні компоненти Android-застосунку

- ***Постачальники вмісту* (*Content Providers*)** – реалізують механізм для обміну даними між програмами. Будь-яка програма може надавати іншим програмам доступ до своїх даних через реалізацію постачальника вмісту, включаючи можливість додавати, видаляти та запитувати дані (за наявності дозволів). Доступ до даних надається за допомогою універсального ідентифікатора ресурсу (URI), визначеного постачальником вмісту. Даними можна ділитися у формі файлу або цілої бази даних SQLite. Наітивні програми Android включають низку стандартних постачальників вмісту, які дозволяють програмам отримувати доступ до таких даних, як контакти та медіафайли. Постачальники вмісту, які наразі доступні в системі Android, можна знайти за допомогою Content Resolver.
- ***Маніфест застосунку* (*Application Manifest*)** – у цьому XML-файлі описуються активності, служби, широкомовні отримувачі, постачальники вмісту і дозволи відповідного застосунку.

Основні компоненти Android-застосунку

- ***Ресурси застосунку* (*Application Resources*)** – окрім файлу маніфесту та файлів Dex з байт-кодом, Android-застосунок, зазвичай, містить колекцію файлів ресурсів. Ці файли зберігають такі ресурси, як XML-представлення макетів інтерфейсу користувача, рядки, зображення, шрифти та кольори, що складають інтерфейс користувача. За замовчуванням ці файли зберігаються у підкаталозі */res* проєкту.
- ***Контекст застосунку* (*Application Context*)** – під час компіляції програми створюється клас під назвою *R*, який містить посилання на ресурси програми. Файл маніфесту програми та ці ресурси об'єднуються, щоб створити так званий контекст застосунку. Цей контекст, представлений класом *android.content.Context*, може використовуватися для отримання доступу до ресурсів застосунку під час виконання. Також цей клас містить багато методів, які можуть використовуватися для збору інформації та внесення змін до програмного оточення застосунку під час його виконання.

Налаштування середовища розробки Android-застосунків

https://developer.android.com/studio

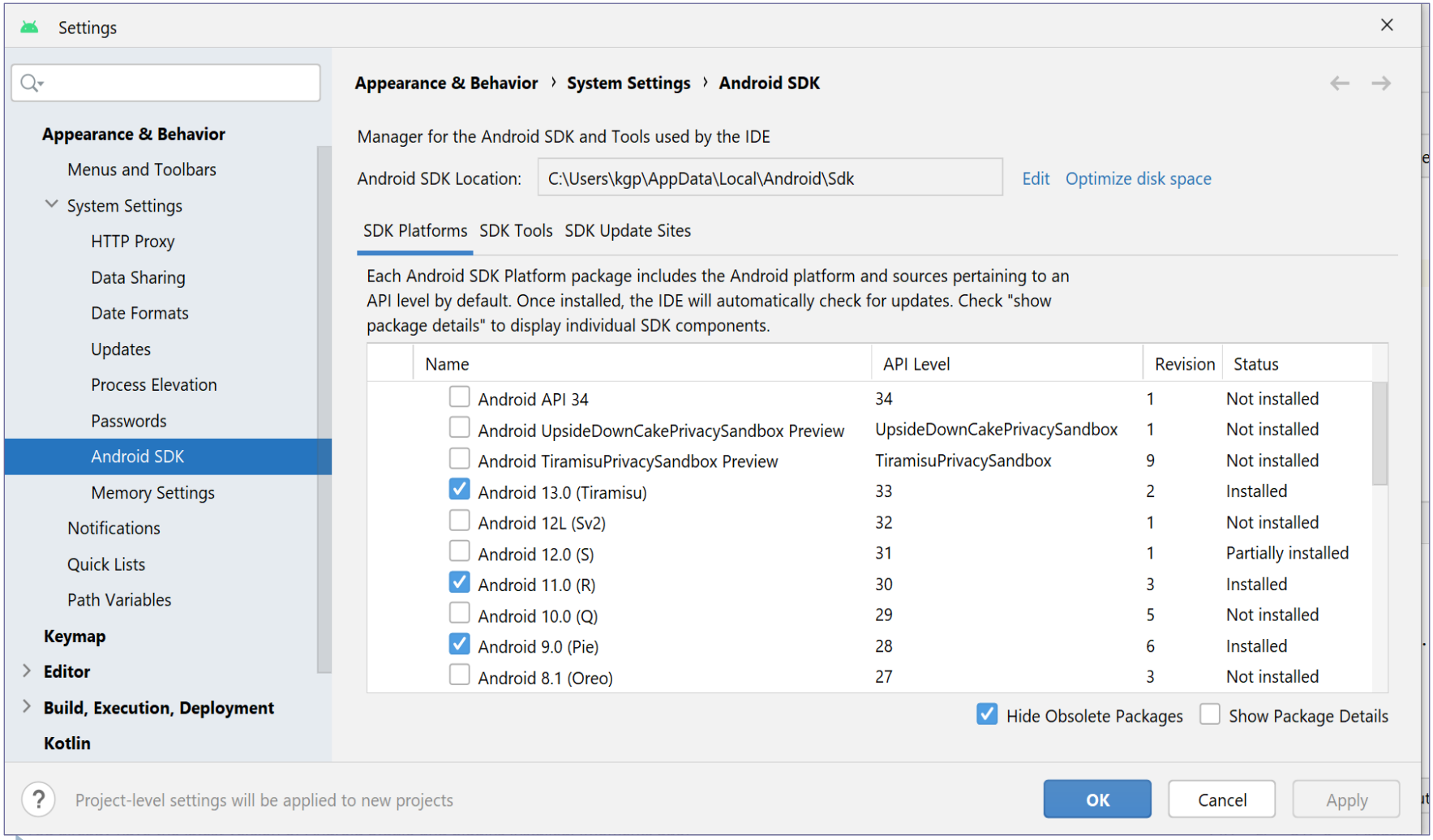
C:\Program Files\Android\Android Studio

AppData\Local\Android\Sdk ← ANDROID_HOME

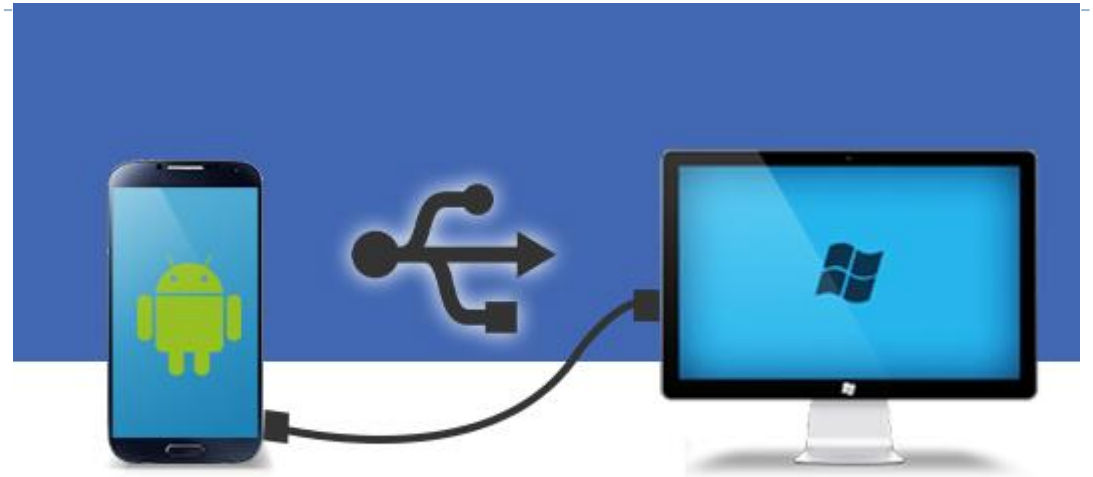
%ANDROID_HOME%\tools
%ANDROID_HOME%\tools\bin
%ANDROID_HOME%\platform-tools
← PATH

Налаштування середовища розробки Android-застосунків

Tools–SDK Manager



Налаштування середовища розробки Android-застосунків

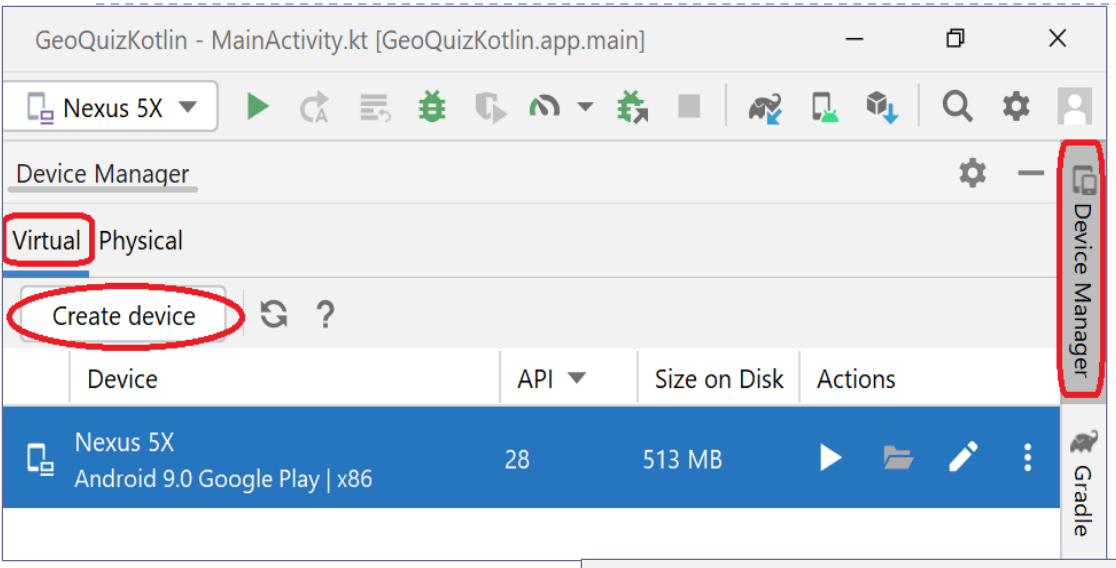


Налаштування–Про телефон–Відомості про ПЗ → Версія Android

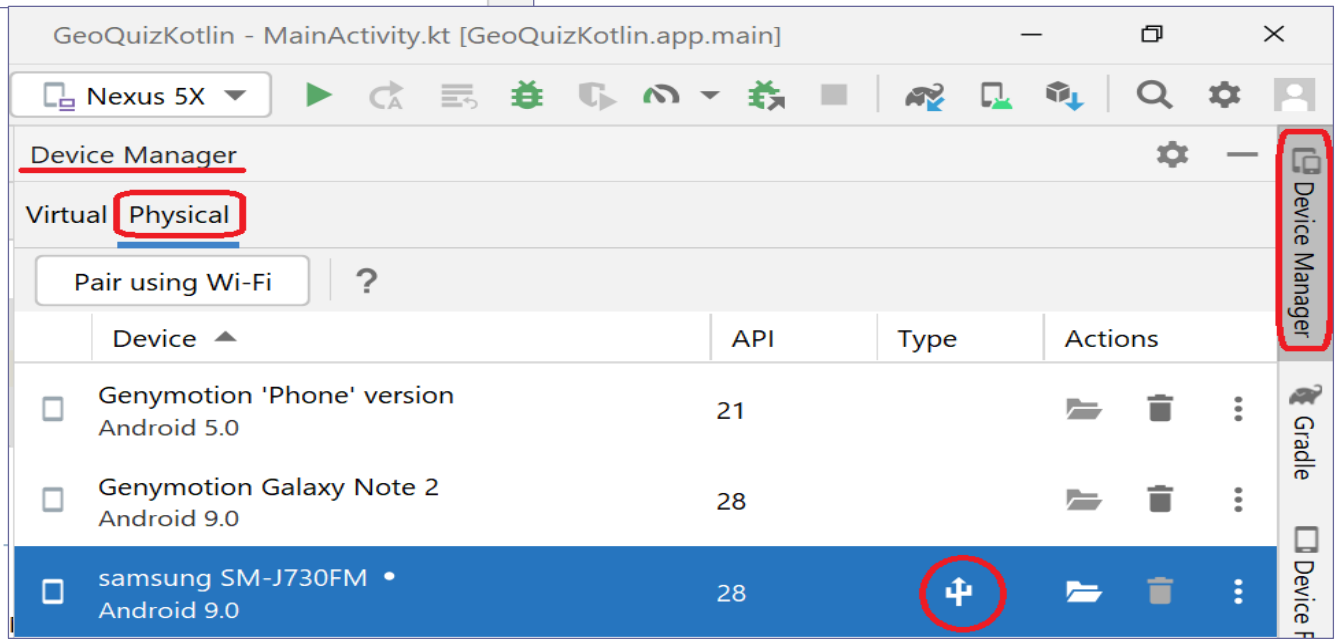
Параметри розробника на смартфоні → *Налаштування–Про телефон–Відомості про ПЗ 7 разів тапнути по Номер версії* (Samsung) або Версія MUI (Xiaomi)

Налаштування–Параметри розробника–Налагодження–Налагодження USB

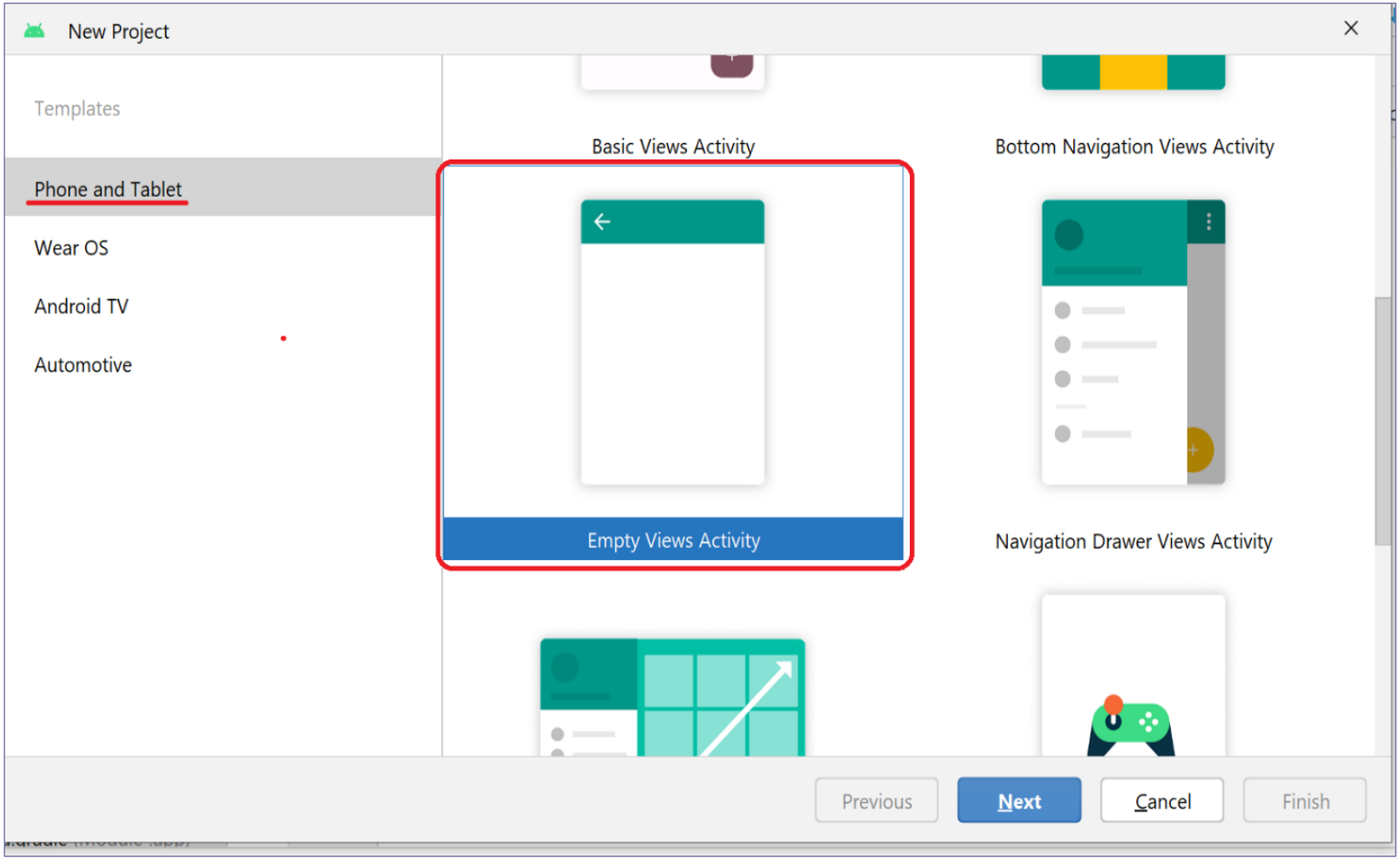
Налаштування середовища розробки Android-застосунків



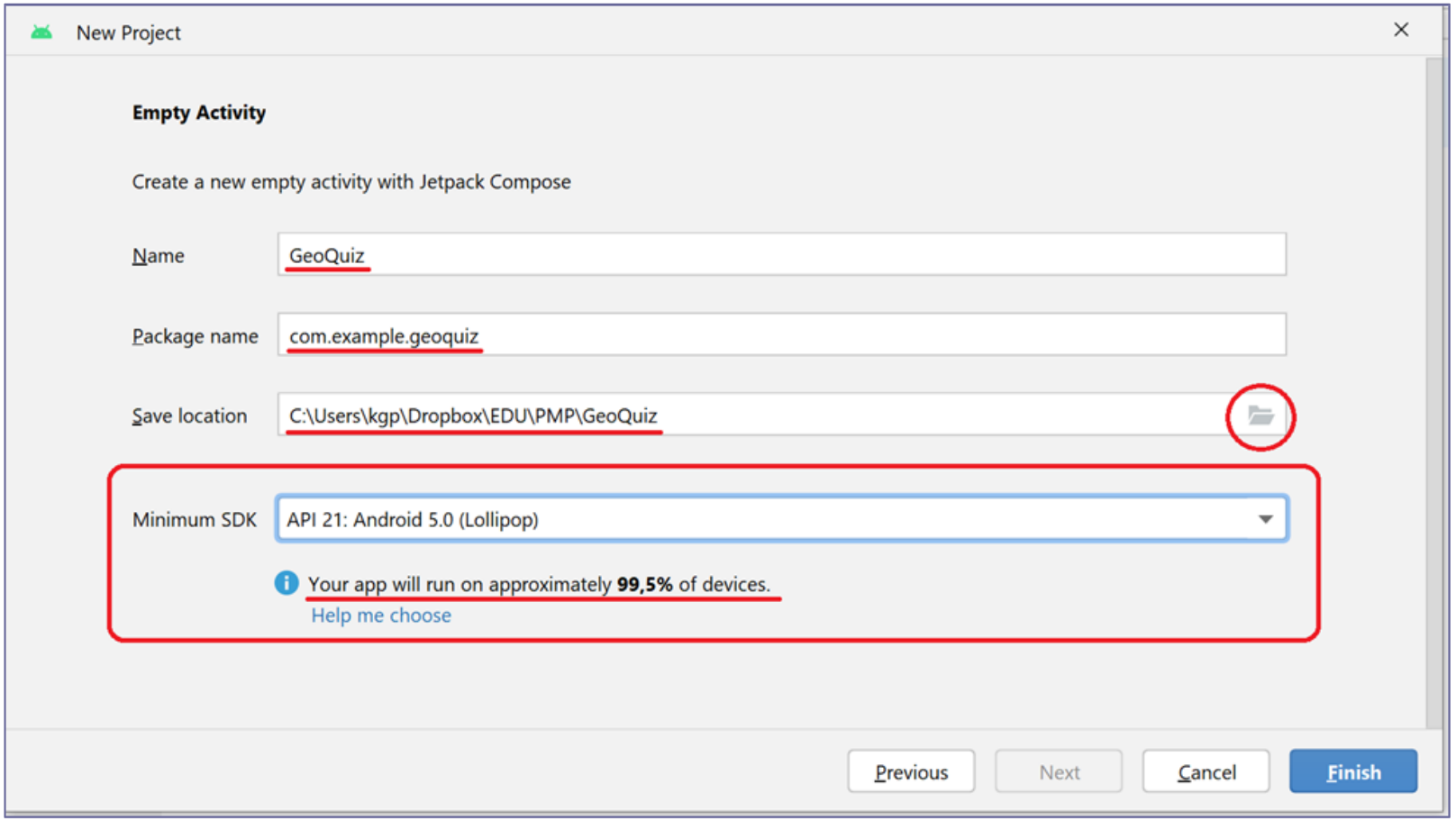
Tools–Device Manager



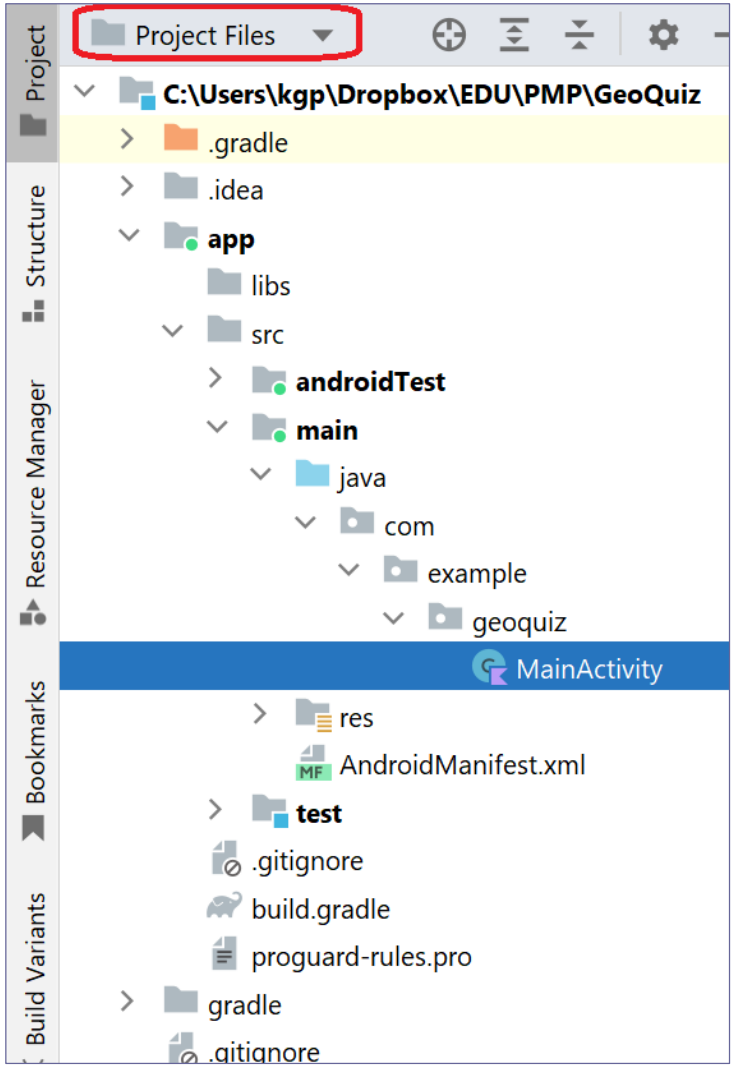
Створення проєкту Android-застосунку

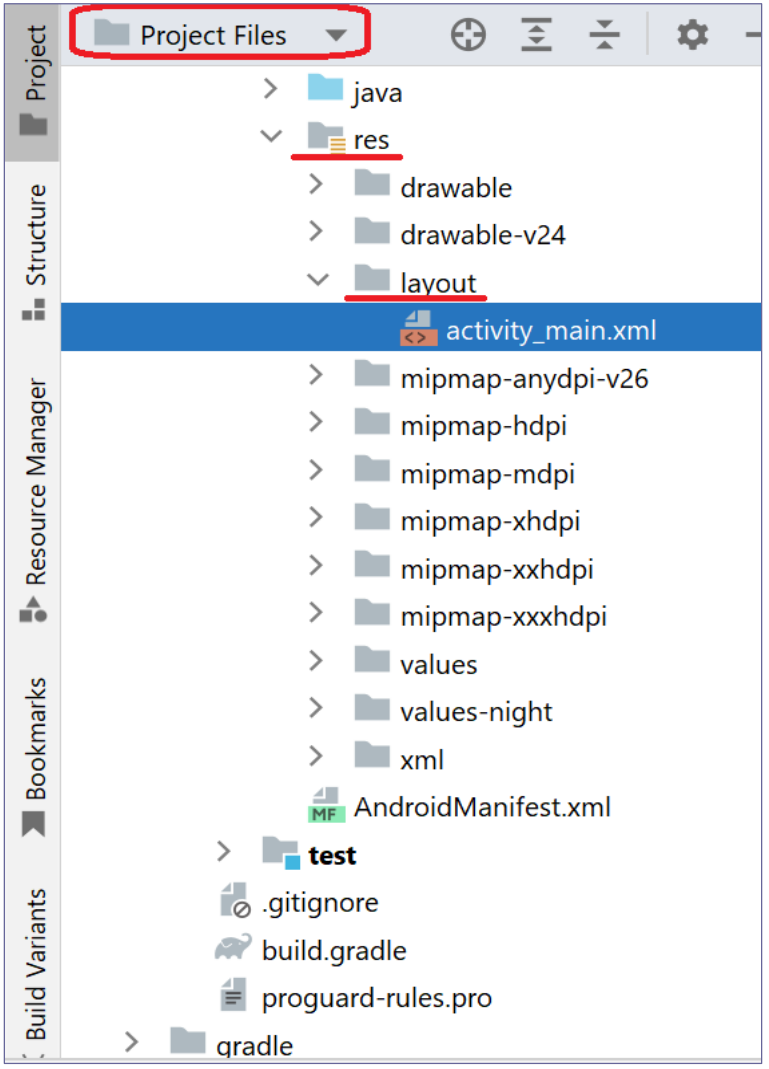


Створення проєкту Android-застосунку

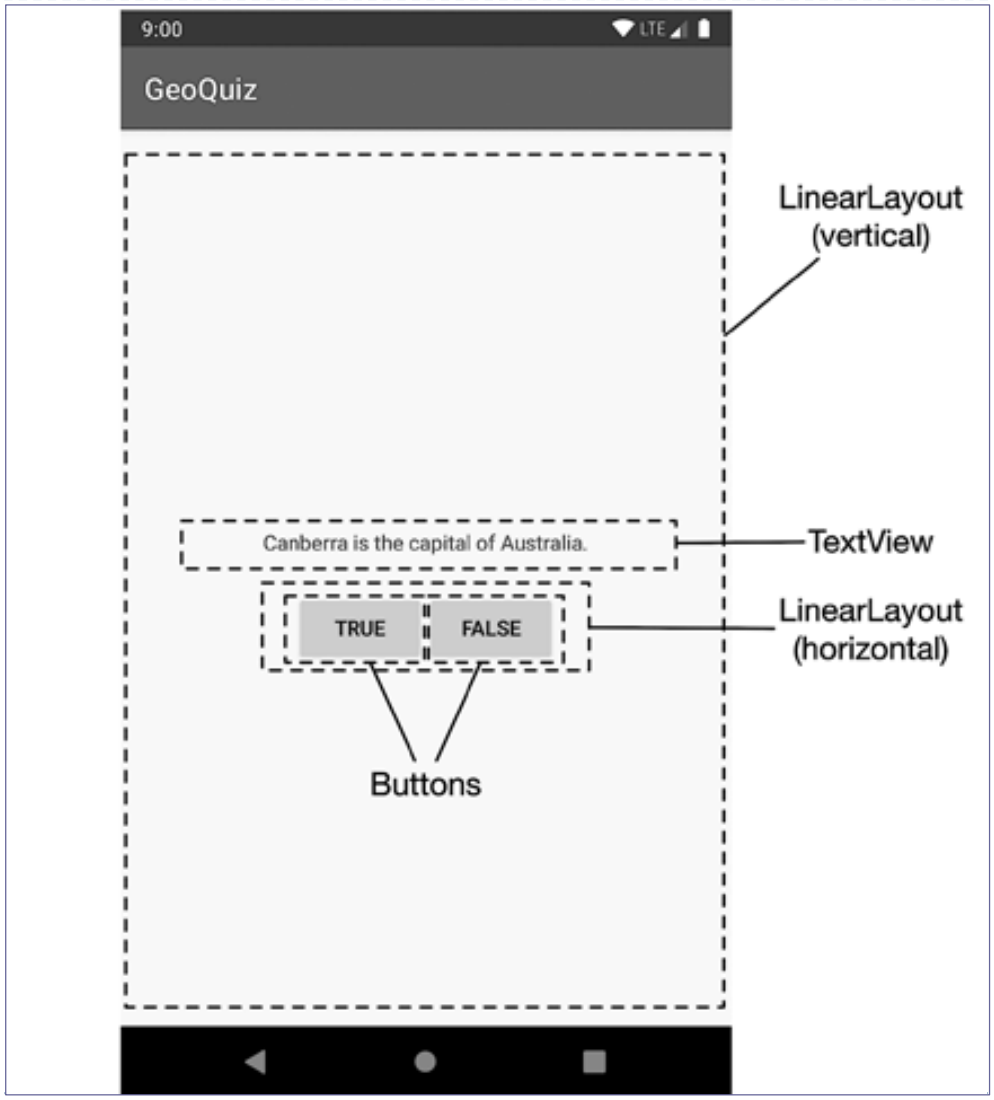


Створення проєкту Android-застосунку

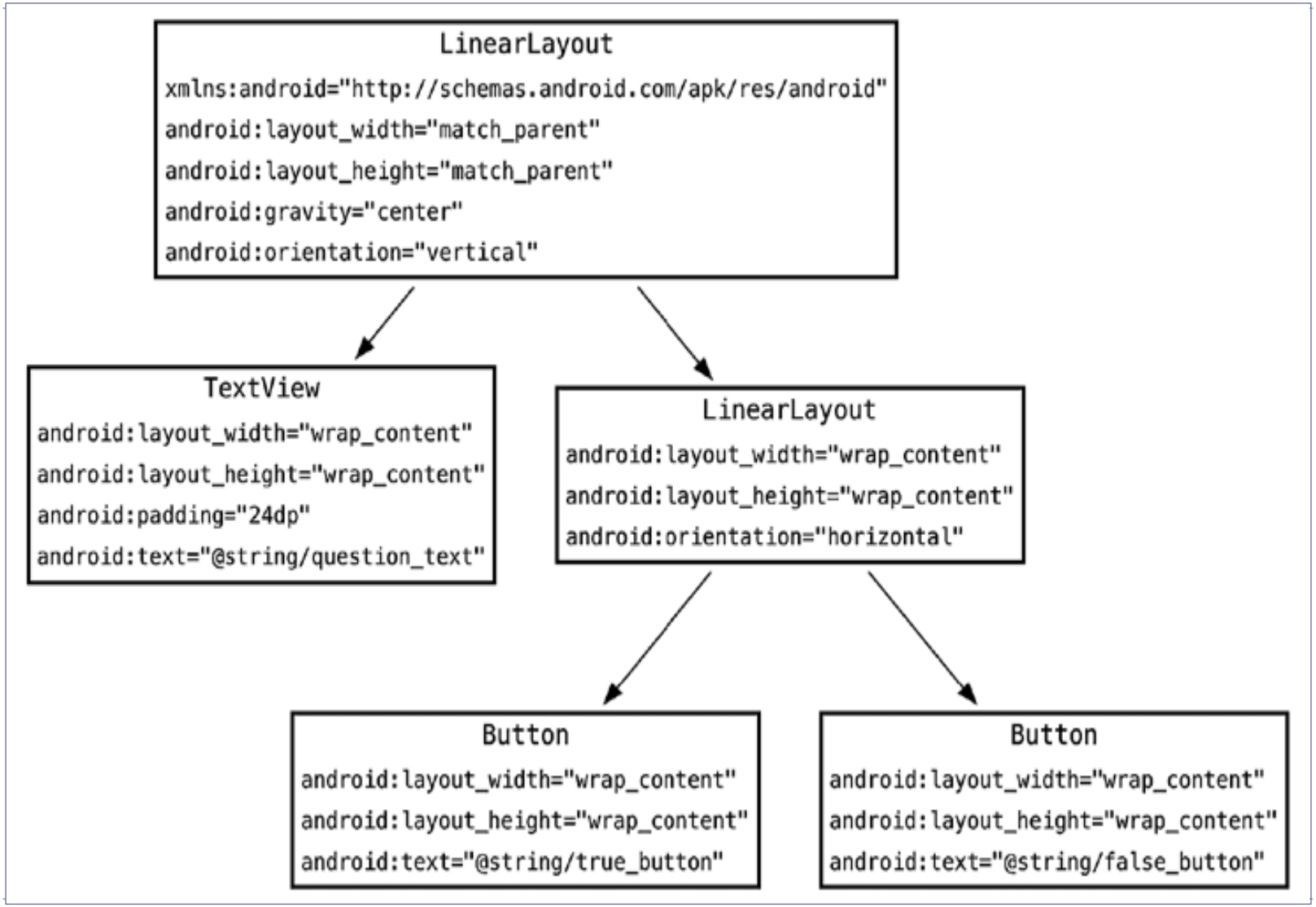




Створення макету користувацького інтерфейсу активності



Створення макету користувацького інтерфейсу активності



Component Tree -> ConstaraintLayout - Convert to - LinearLayout

Створення макету користувацького інтерфейсу активності

Основні атрибути віджетів:

- **`android:id`** - ідентифікатор віджету, за яким він може бути знайдений у файлі ресурсів;
- **`android:orientation`** - визначає, як будуть розміщені вкладені представлення – по вертикалі чи по горизонталі;
- **`android:gravity`** - визначає позиціонування вмісту представлення ();
- **`android:layout_width`** та **`android:layout_height`** - визначають ширину та висоту представлень (**`match_parent`** -– коли розміри визначаються розмірами батьківського представлення, або **`wrap_content`** - коли розміри представлення визначаються розмірами його вмісту);
- **`android:layout_weight`** – визначає "вагу" – важливість віджету: віджет з більшим значенням цього атрибуту зможе зайняти більшу площину у батьківському представленні;
- **`android:text`** - зазначає текст, що повинен відображатися у віджеті.

Створення макету користувацького інтерфейсу активності

Одиниці розміру, які використовуються для представлення:

- ***пікселі (pixels – px)*** – розмір зазначається кількістю екранних пікселів;
- ***міліметри (millimeters – mm)***;
- ***дюйми (inches – in) – 25,4 mm***;
- ***пункти (points – pt)*** – 1/72 дюйма;
- ***незалежні від щільності піксели (density-independent pixels – dp промовляється "діпс")*** – одиниця виміру, яка базується на фізичній щільності екрана: один dp дорівнює одному пікселю на екрані 160 пікселів на дюйм. При зазначенні розмірів у dp реальний розмір елементів на екранах різної щільності буде приблизно однаковим;
- ***незалежні від масштабу пікселі (scale-independent pixels) – sp*** – визначається аналогічно dp, але з урахуванням коефіцієнту масштабування, який базується на розмірі шрифту, який користувач обирає в системних налаштуваннях пристрою.

Рекомендується розмір шрифту визначати у sp, а усе інше – у dp.

Створення макету користувацького інтерфейсу активності

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        android:text="Hello World!" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
...
```

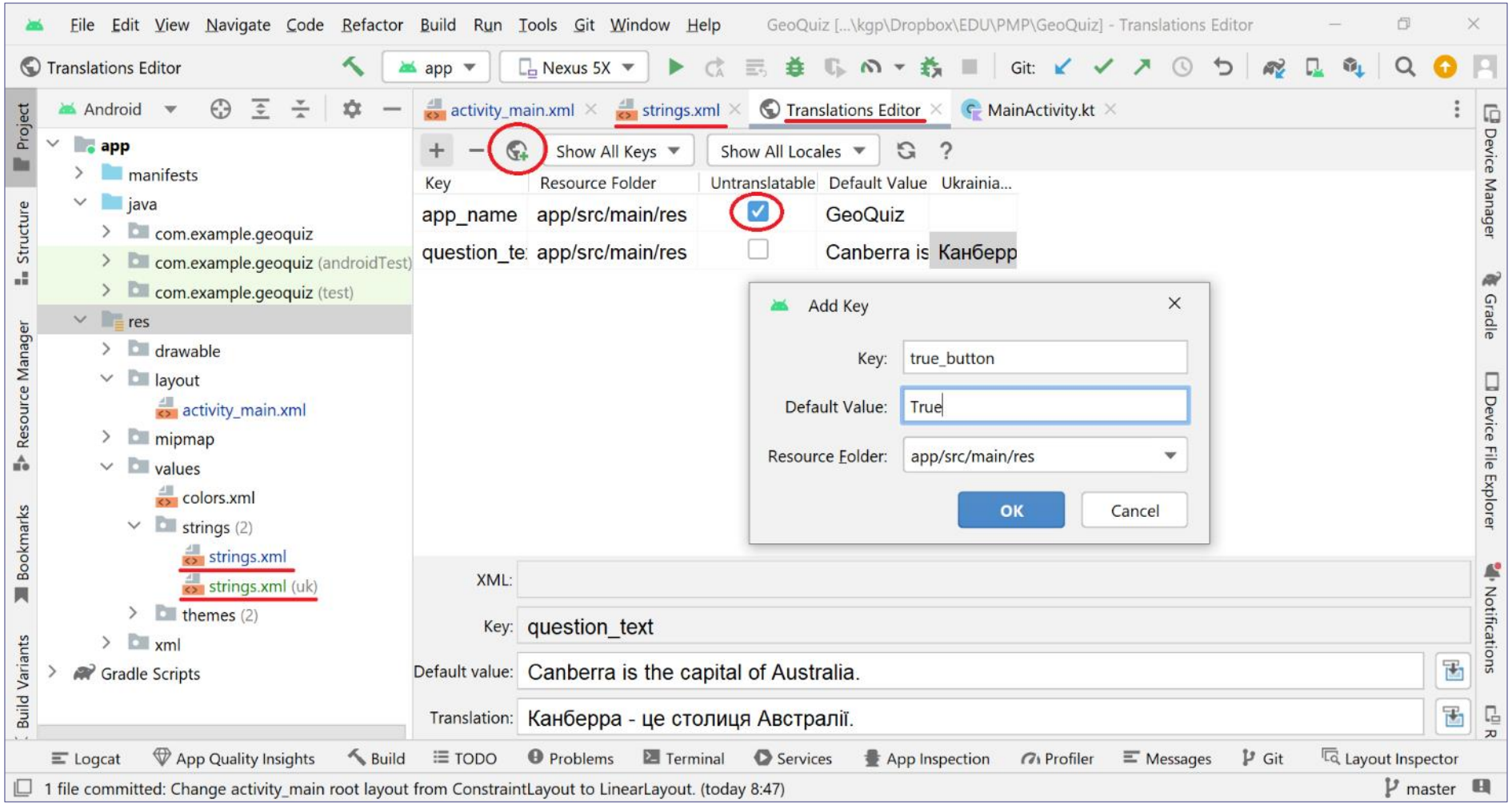
activity_main.xml

Створення макету користувацького інтерфейсу активності

```
...

    <Button
        android:id="@+id/true_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/false_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
  </LinearLayout>

</LinearLayout>
```

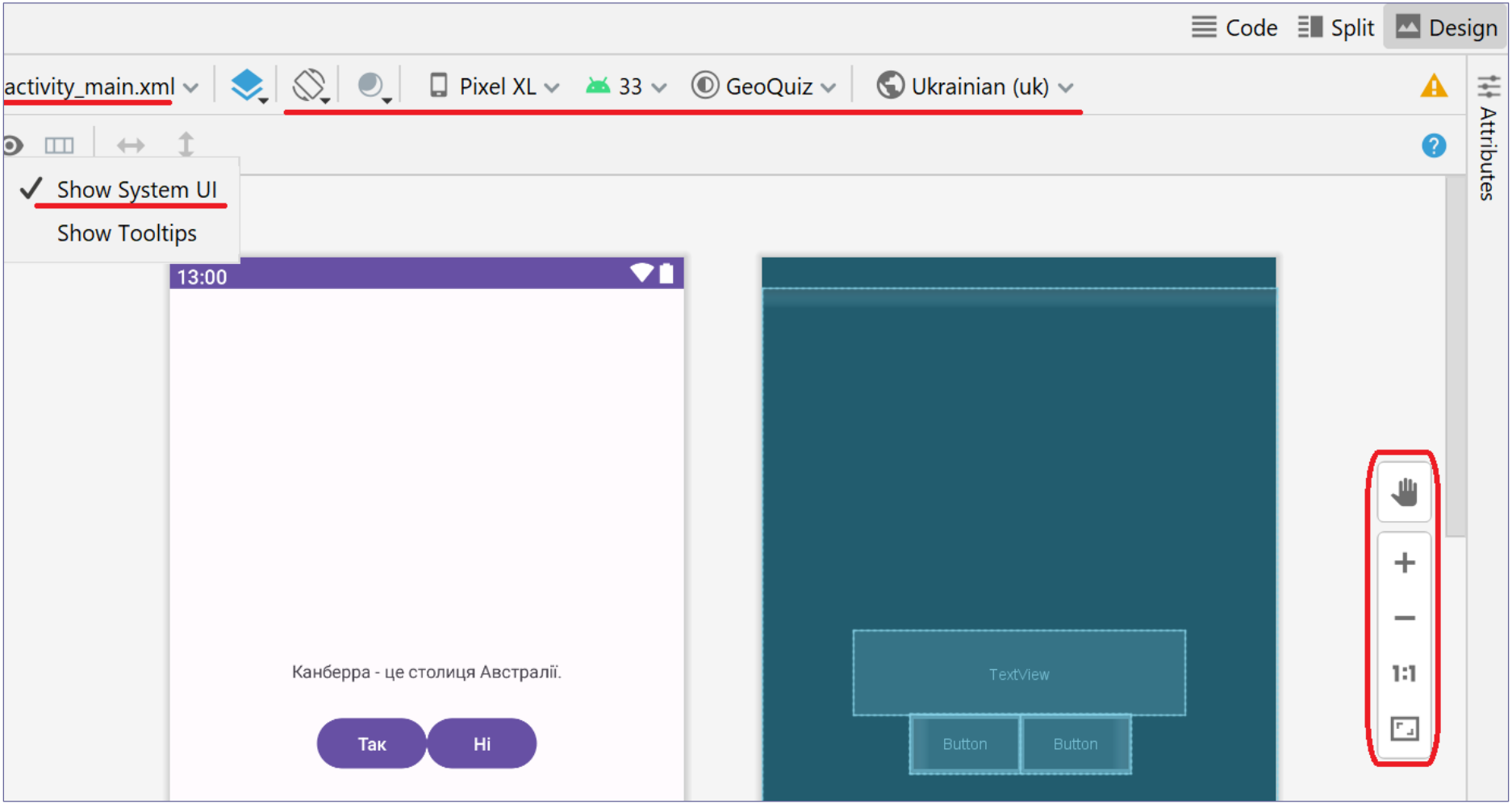
activity_main.xml

Створення рядкових ресурсів



res/values/string.xml

Перегляд макету у вікні Дизайнера



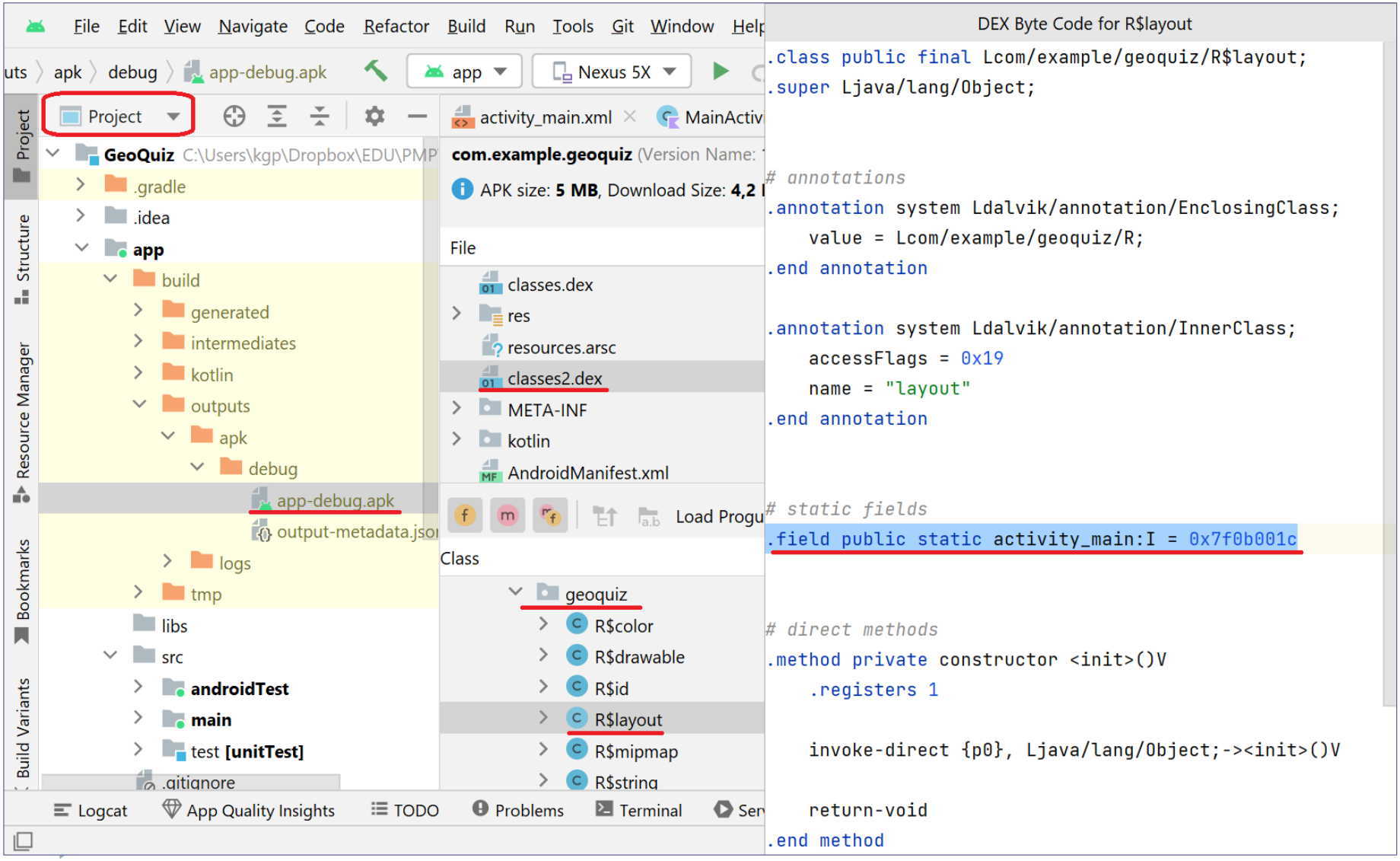
Підкреслені варіанти перегляду

Розробка класу активності

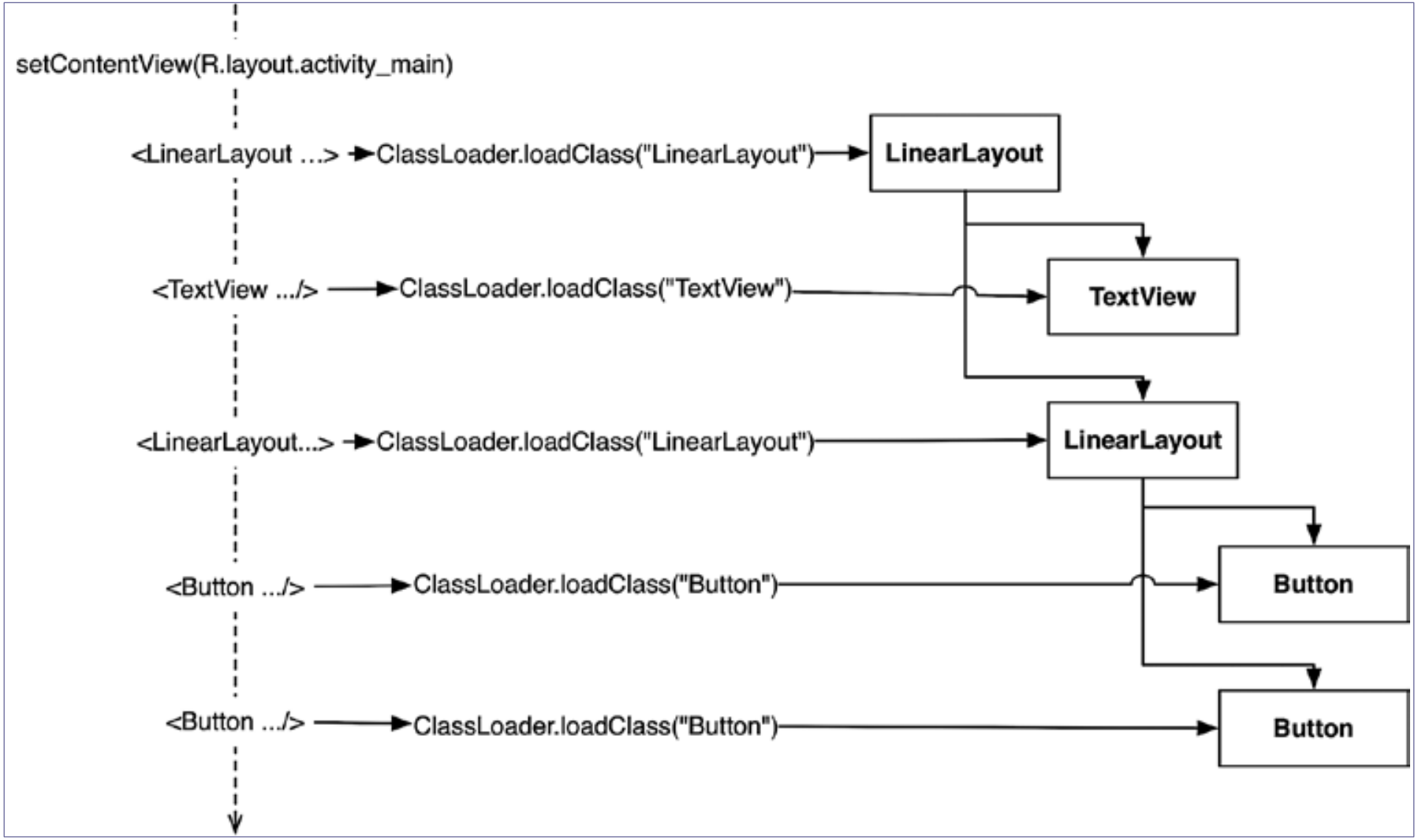
```
package com.example.geoquiz

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

файл ресурсів R

Перегляд файлу ресурсів R



Створення об'єктів-представлень макету активності



android.view.LayoutInflater → View inflate(@LayoutRes int resource, @Nullable ViewGroup root)

Розробка класу активності - додання обробників подій

```
class MainActivity : AppCompatActivity() {

    private lateinit var trueButton: Button
    private lateinit var falseButton: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        trueButton = findViewById(R.id.true_button)
        falseButton = findViewById(R.id.false_button)
...
```

```
<string name="correct_toast">Correct!</string>
<string name="incorrect_toast">Incorrect!</string>
```

Додати до string.xml

Розробка класу активності - додання обробників подій

```
...
    trueButton.setOnClickListener { view:View ->
        Toast.makeText(
            this,                          <-- context
            R.string.correct_toast,        <-- resource id
            Toast.LENGTH_SHORT             <-- duration
        ).show()
    }

    falseButton.setOnClickListener { view:View ->
        Toast.makeText(
            this,
            R.string.incorrect_toast,
            Toast.LENGTH_SHORT
        ).show()
    }      }     }
```

Розробка класу активності - додання обробників подій