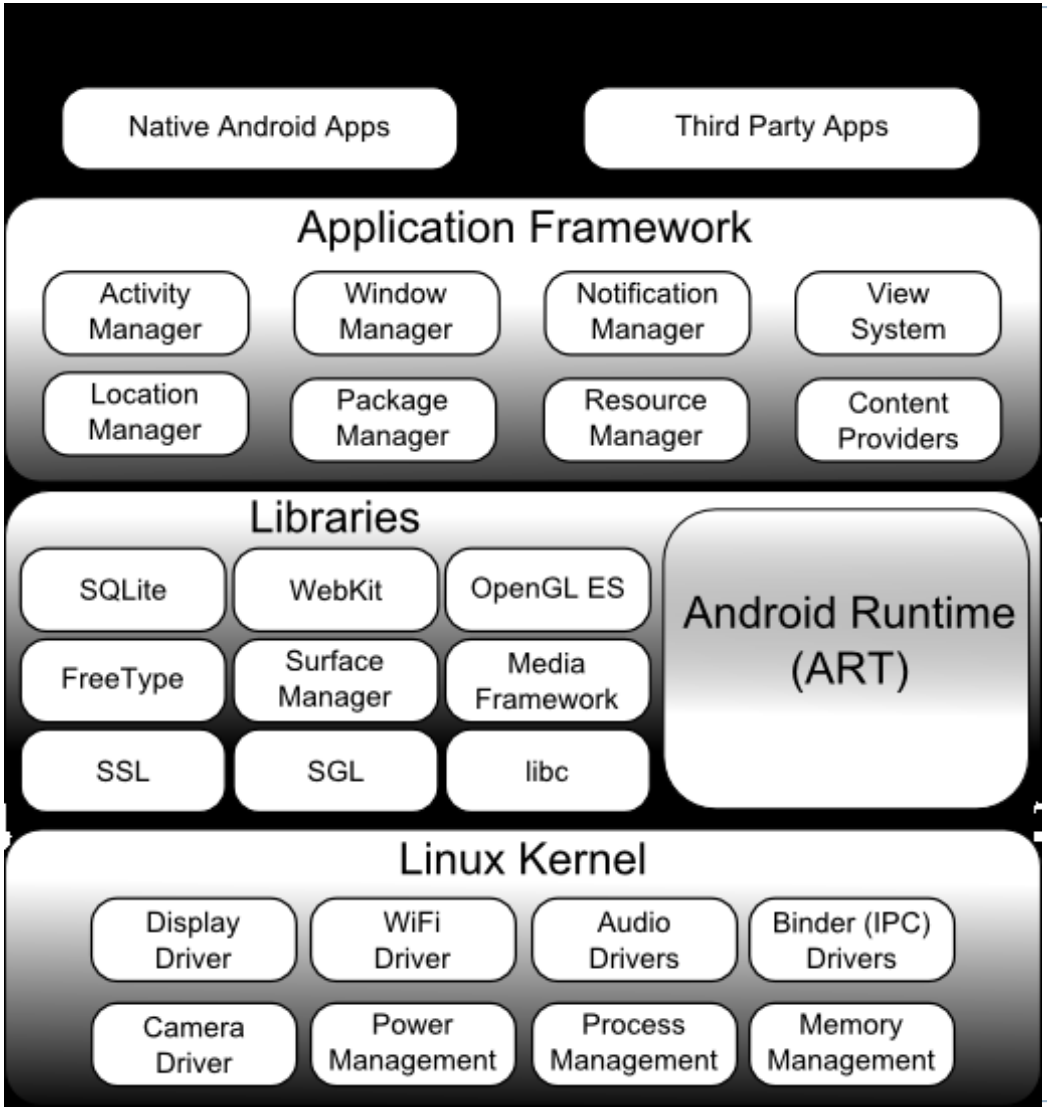
Сучасні технології мобільного програмування

Розробка Android-застосунків на основі технології Jetpack Compose

Слайди до лекцій (1 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.

Компоненти Android-застосунку



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 одними з основних компонентів застосунку.
- ***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.

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. Є одними з основних компонентів застосунку.
- ***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. Є одними з основних компонентів застосунку.

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. Є одними з основних компонентів застосунку.
- ***Маніфест застосунку* (*Application Manifest*)** – у цьому XML-файлі описуються активності, служби, широкомовні отримувачі, постачальники вмісту і дозволи відповідного застосунку.



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 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

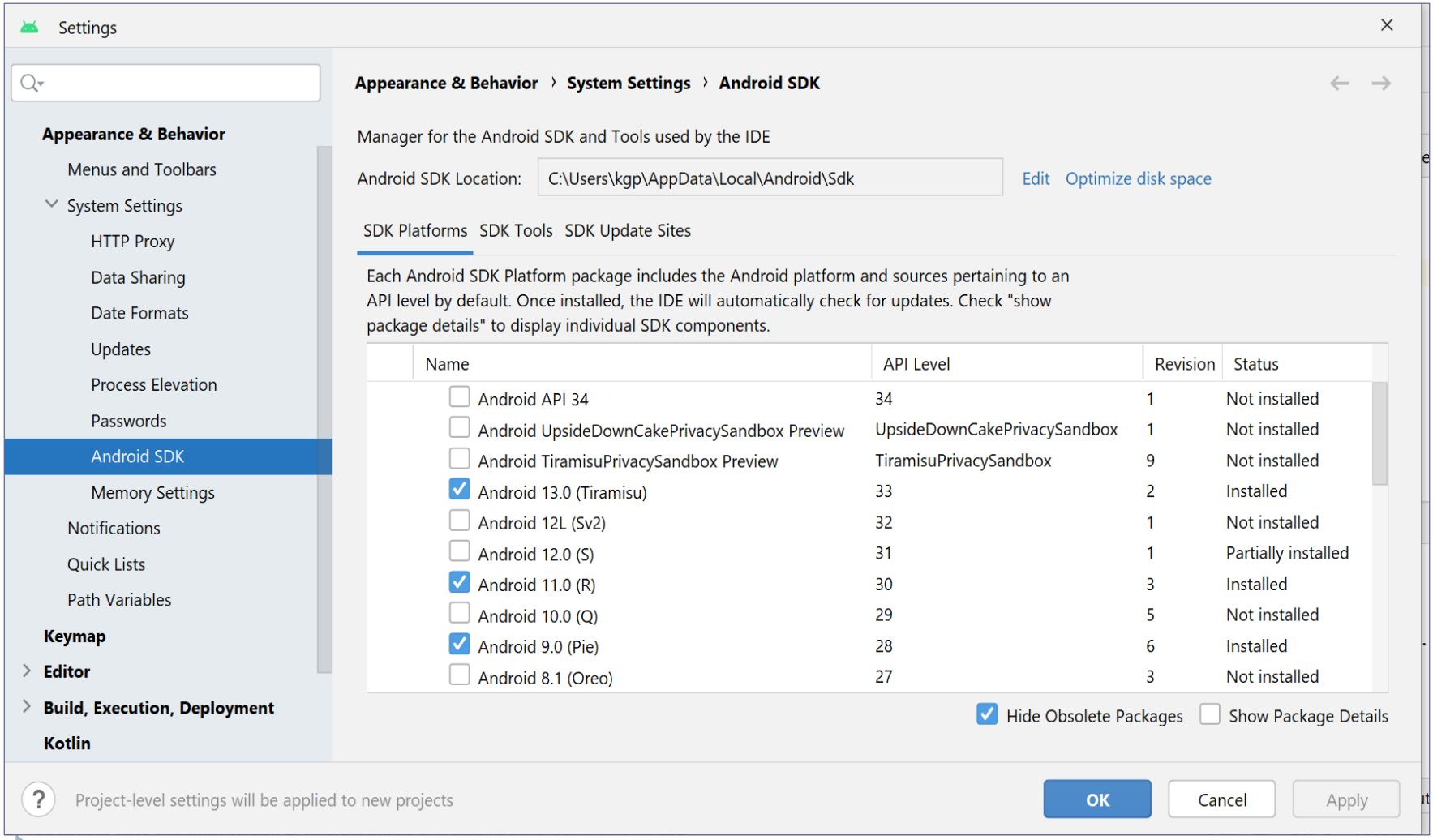
AppData\Local\Android\Sdk ← ANDROID_HOME

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

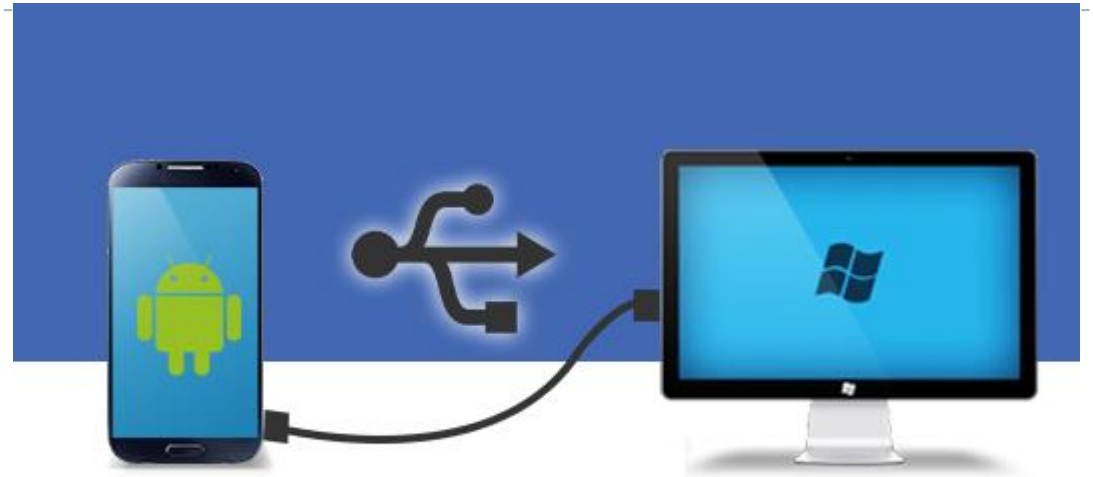


Налаштування середовища розробки Android-застосунків

Tools–SDK Manager



Налаштування середовища розробки Android-застосунків

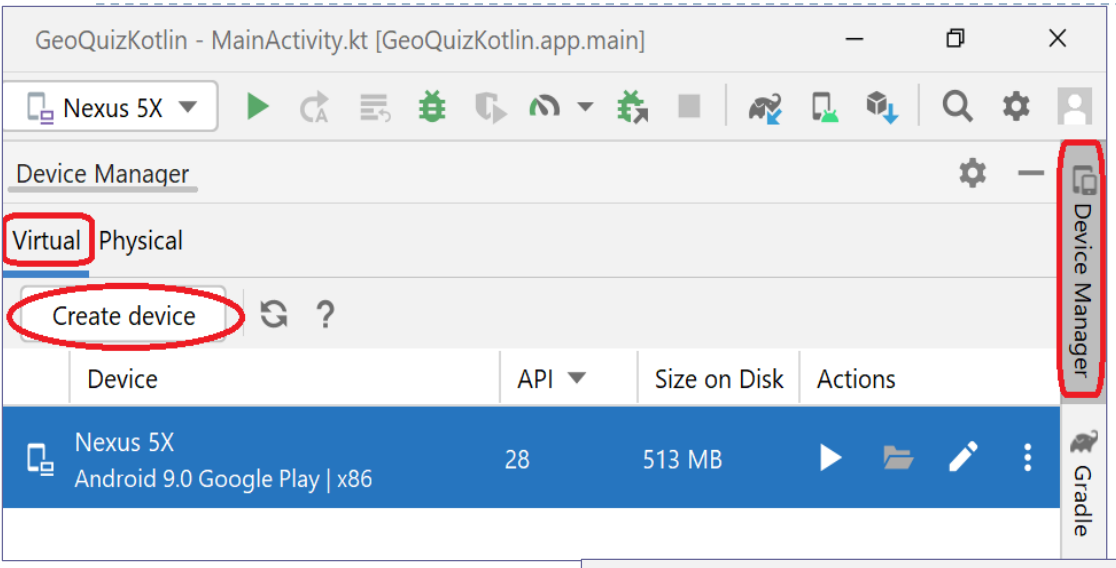


Налаштування–Про телефон–Відомості про ПЗ → Версія Android

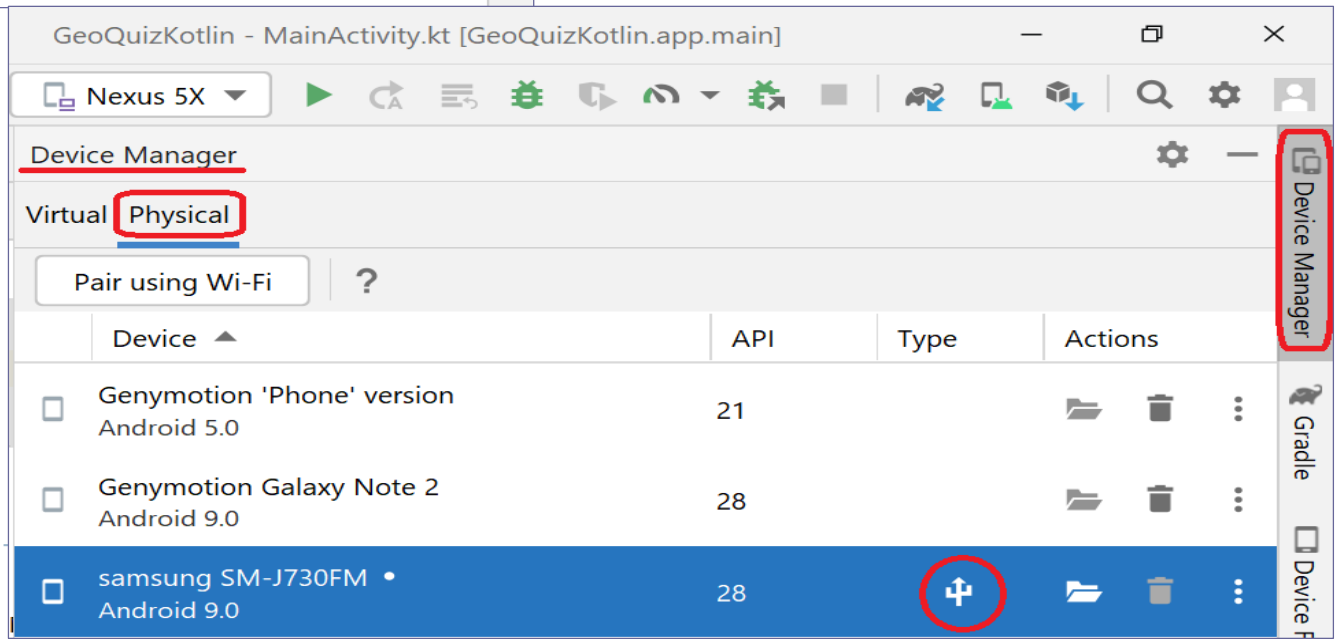
Параметри розробника на смартфоні → *Налаштування–Про телефон–Відомості про ПЗ 7 разів тапнути по Номер версії* (Samsung) або Версія MUI (Xiaomi)

Налаштування–Параметри розробника–Налагодження–Налагодження USB

Налаштування середовища розробки Android-застосунків



Tools–Device Manager



Jetpack Compose

Jetpack Compose — це набір інструментів для розробки інтерфейсу користувача, який забезпечує:

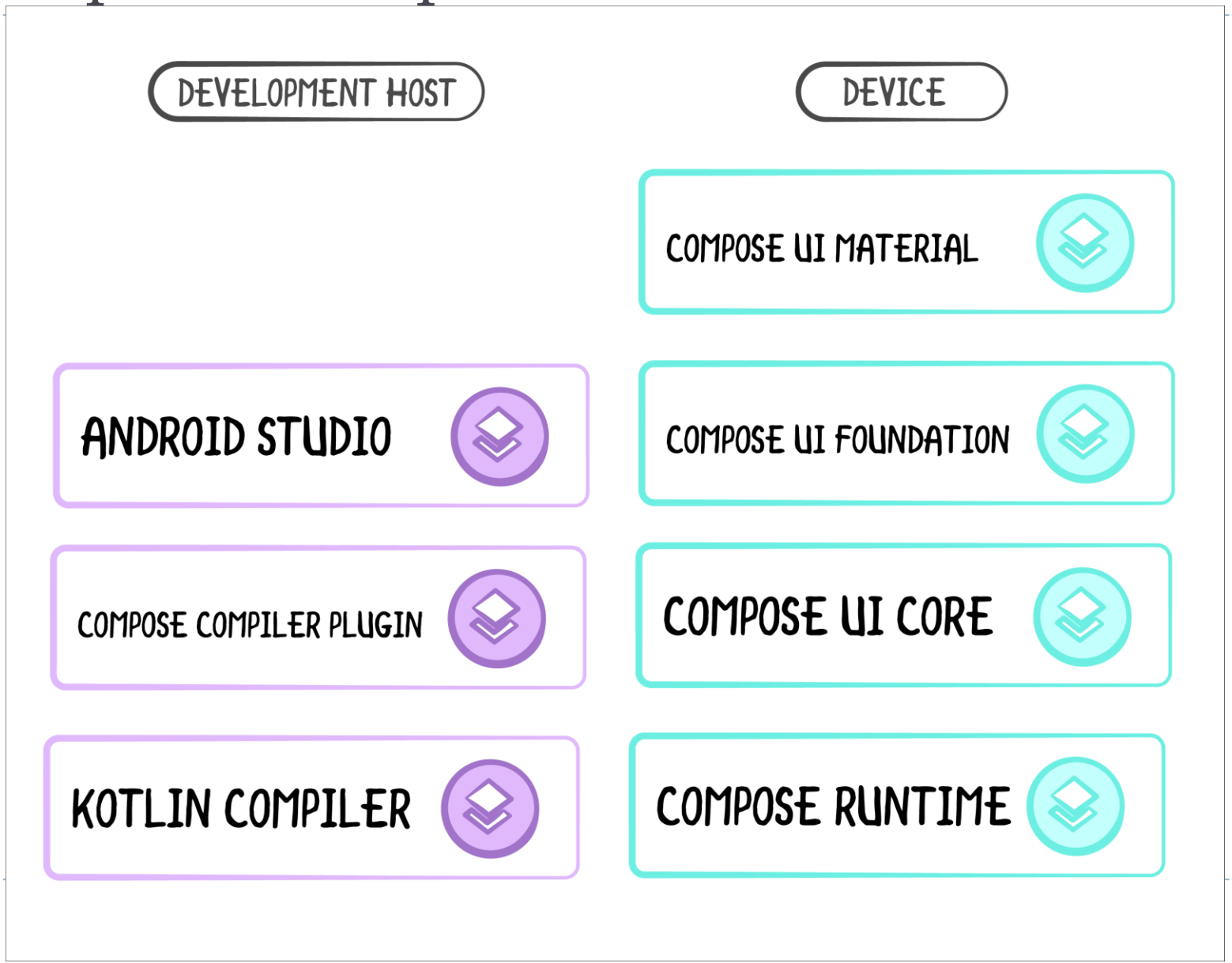
- API на базі Kotlin;
- Декларативні способи побудови інтерфейсу користувача мовою Kotlin;
- Відслідковування стану застосунку з автоматичним оновленням інтерфейсу у разі його зміни;
- Великий набір компонентів, які можна комбінувати, стилізувати та анімувати.



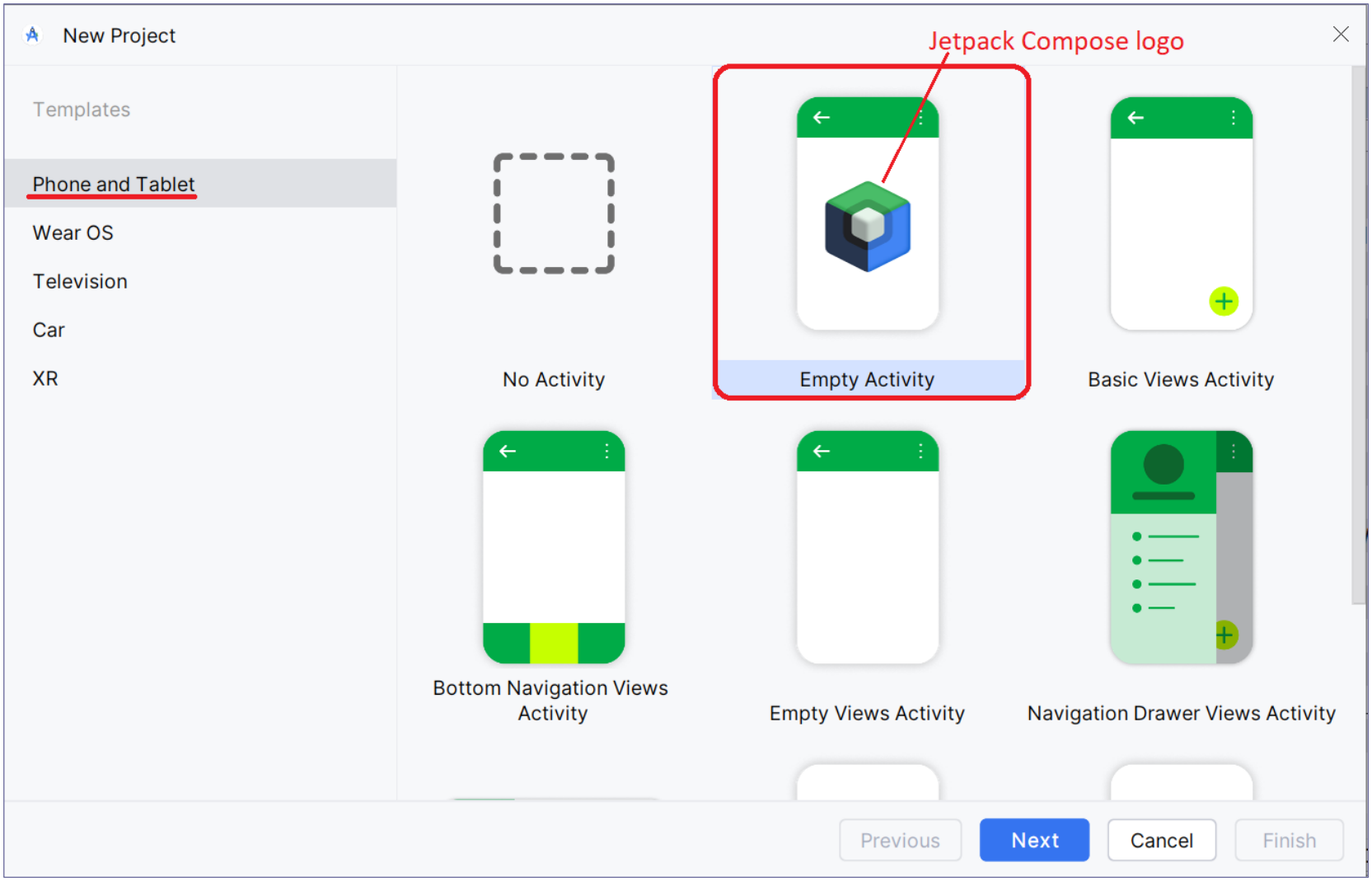
jetpack
compose



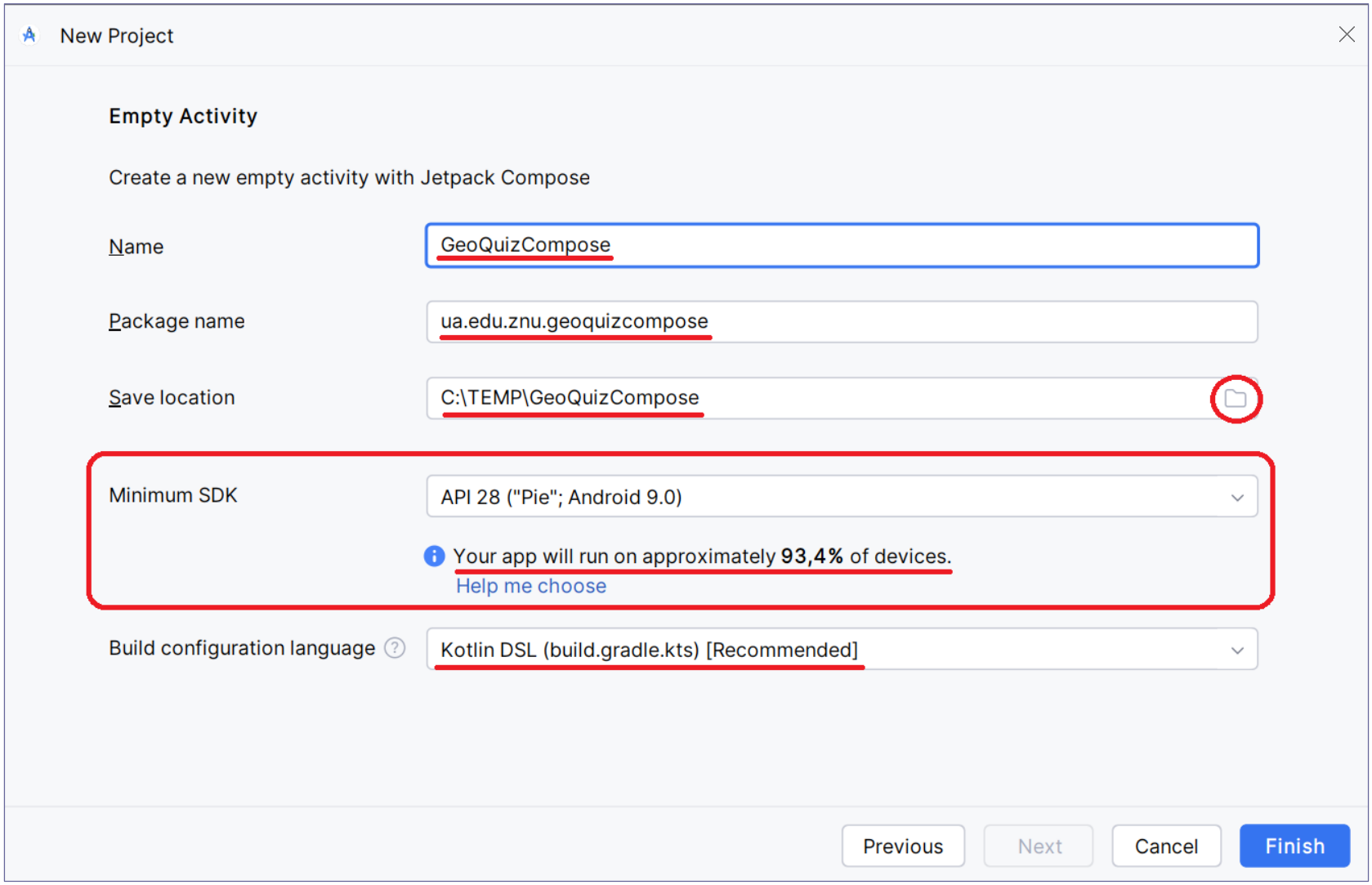
Jetpack Compose - стек технологій



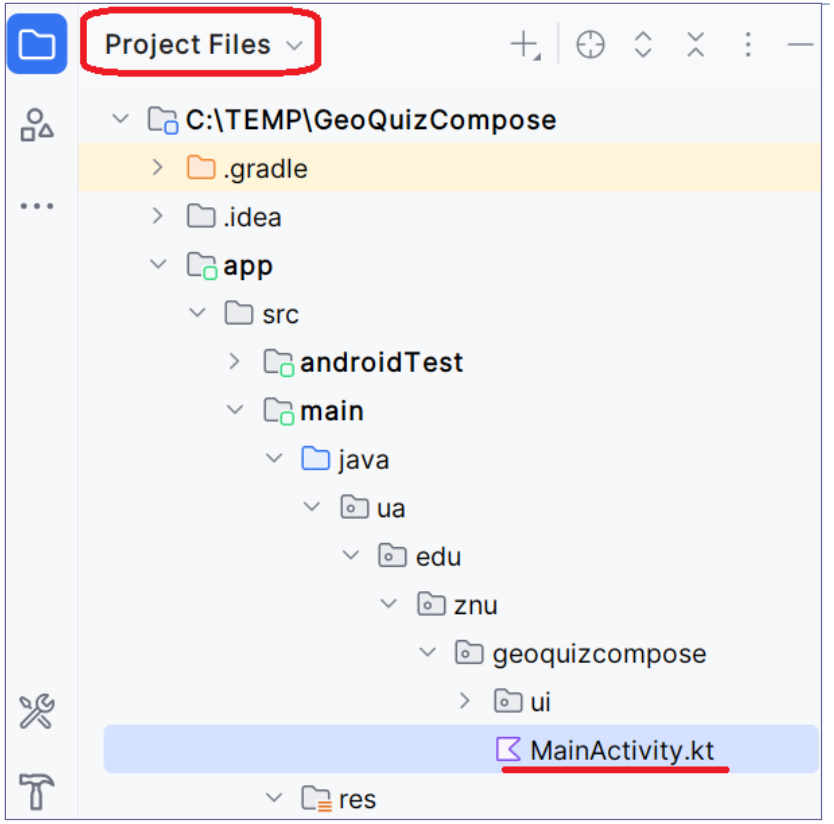
Створення проєкту Android-застосунку

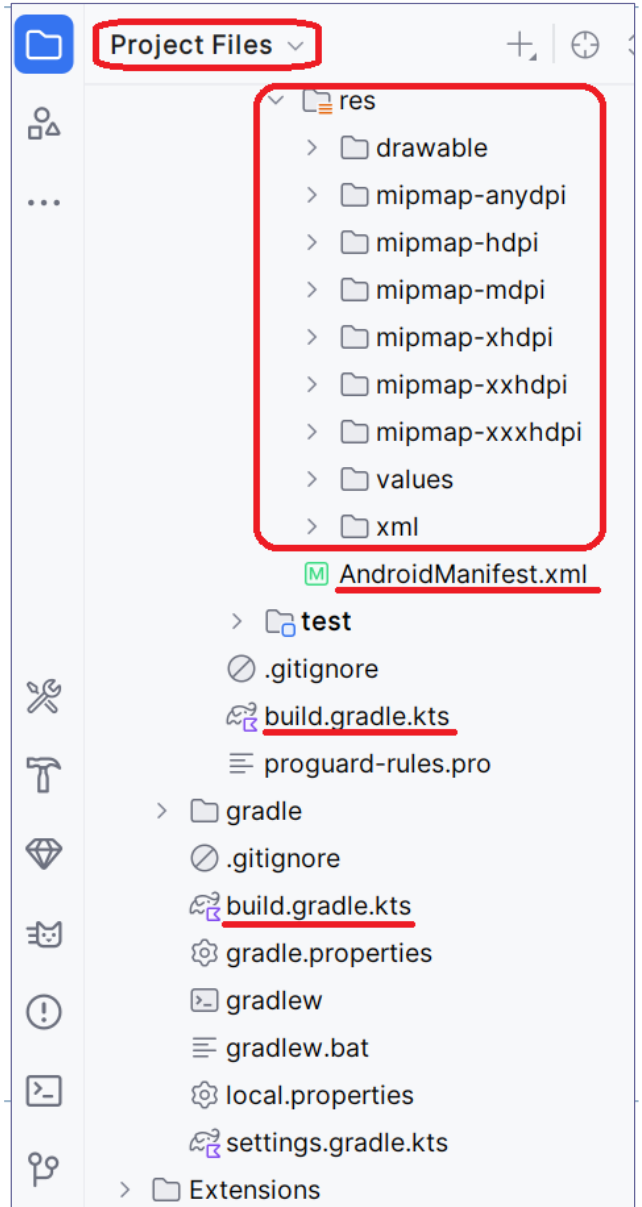


Створення проєкту Android-застосунку



Створення проєкту Android-застосунку





Створення проєкту Android-застосунку

```
package ua.edu.znu.geoquizcompose

import ...

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            GeoQuizComposeTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        name = "Android",
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
} ...
```

Composable functions

Set use full screen

Invoke custom Composable function with passing args to it

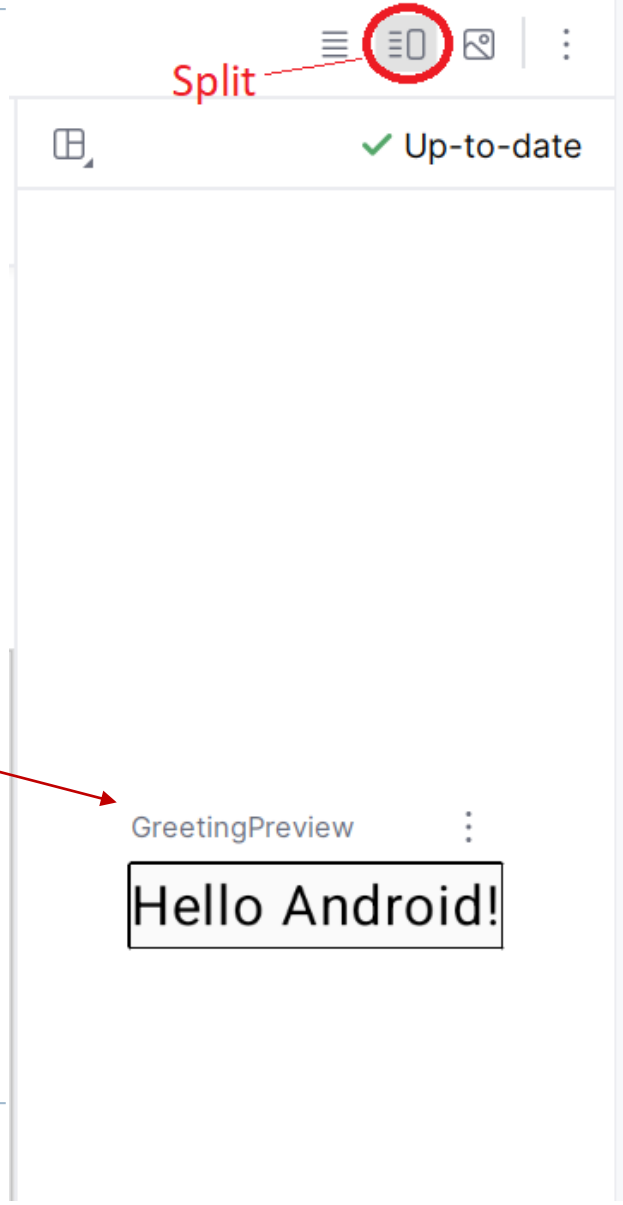
Створення проєкту Android-застосунку

```
...
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    GeoQuizComposeTheme {
        Greeting("Android")
    }
}
```

Composable functions declaration

Preview function declaration



Створення проєкту Android-застосунку

```
@Composable  2 Usages
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
    Text(
        text = "and",
        modifier = modifier
    )
    Text(
        text = "Jetpack Compose",
        modifier = modifier
    )
}
```

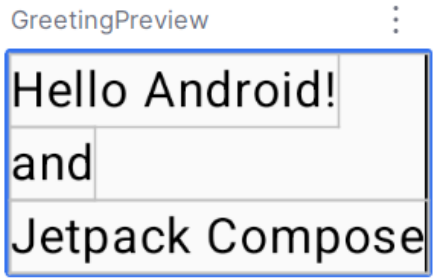


We need a Layout Composable



Створення проєкту Android-застосунку

```
@Composable  2 Usages
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Column {
        Text(
            text = "Hello $name!",
            modifier = modifier
        )
        Text(
            text = "and",
            modifier = modifier
        )
        Text(
            text = "Jetpack Compose",
            modifier = modifier
        )
    }
}
```



We use a **Column** Layout Composable

All Views and Layouts are Composable functions

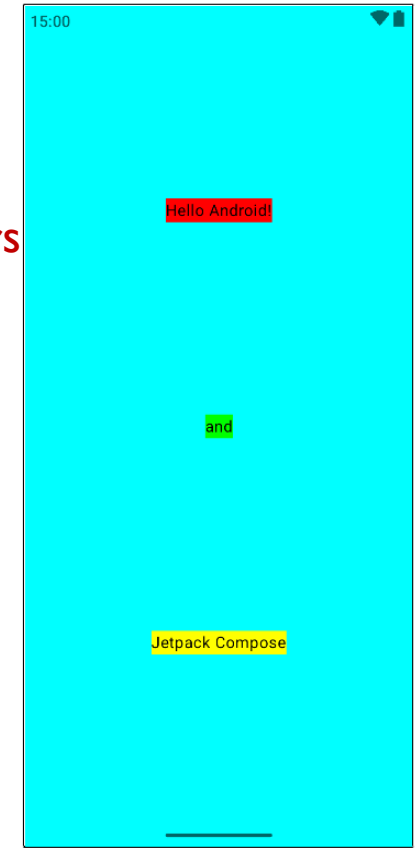


Створення проєкту Android-застосунку

```
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Column(
        modifier = Modifier.fillMaxSize().background(color = Color.Cyan),
        horizontalAlignment = Alignment.CenterHorizontally,
//          verticalArrangement = Arrangement.Center
        verticalArrangement = Arrangement.SpaceEvenly
    ) {
        Text(
            text = "Hello $name!",
            modifier = modifier.background(color = Color.Red)
        )
        Text(
            text = "and",
            modifier = modifier.background(color = Color.Green)
        )
        Text(
            text = "Jetpack Compose",
            modifier = modifier.background(color = Color.Yellow)
        )
    }
}

@Preview(showBackground = true, showSystemUi = true)
```

Column parameters



You can use Row Layout Composable as horizontal container

Jetpack Compose

- Усі компоненти (віджети) та компонувальники - це Composable-функції, їх імена зазвичай пишуться з великої літери;
- І розмітка екранів, і код логіки, і навіть конфігураційні скрипти пишуться на мові Kotlin;
- Composable-функції, як і буть які функції можуть мати параметри;
- Зазвичай Composable-функції нічого не повертають (повертають Unit)
- Компонувальники мають параметр content, який повинен бути Composable-функцією.

```
@Composable
inline fun Column(
    modifier: Modifier = Modifier,
    verticalArrangement: Arrangement.Vertical = Arrangement.Top,
    horizontalAlignment: Alignment.Horizontal = Alignment.Start,
    content: @Composable ColumnScope.() -> Unit
) { ... )
}
```

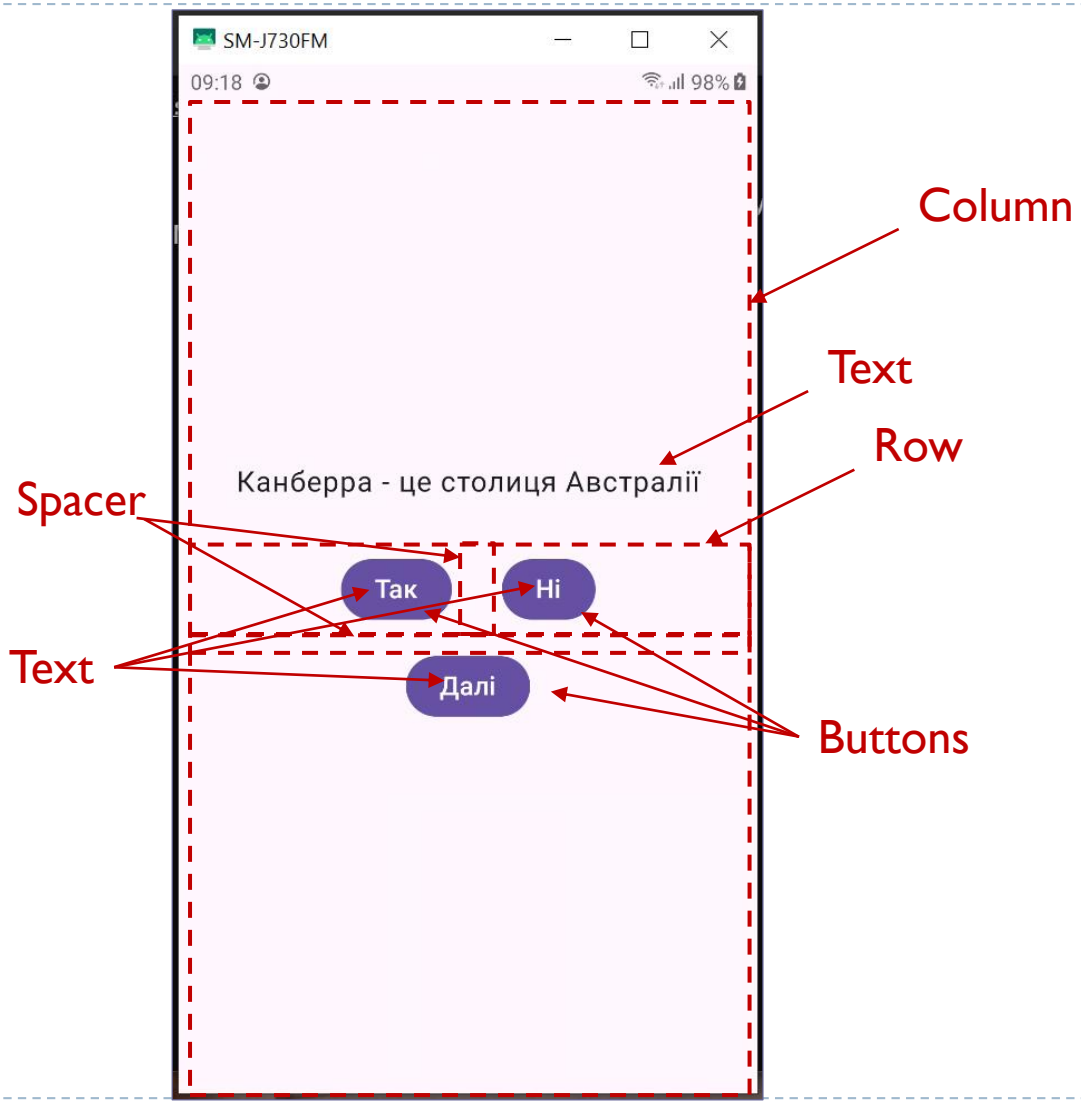
Jetpack Compose - Modifier

- Кожний компонент (віджет) має ***специфічні для нього параметри***, наприклад, text, fontSize, fontStyle, textAlign для компонента Text
- Водночас є ***параметри, загальні для усіх компонентів***, які можуть бути використані, або ні при налаштуванні компонента та які обробляються компонувальниками або системою. Вони підключаються через параметр modifier типу Modifier.
- Такі параметри є функціями-розширеннями з типом-отримувачем Modifier, які зберігаються у найбільш підходящих файлах Kotlin (наприклад,
fun Modifier.align(alignment: Alignment.Horizontal): Modifier
зберігається у Column.kt).
- Кожна функція-розширення повертає об'єкт Modifier, що дозволяє будувати ланцюжки з необхідних модифікаторів, наприклад:

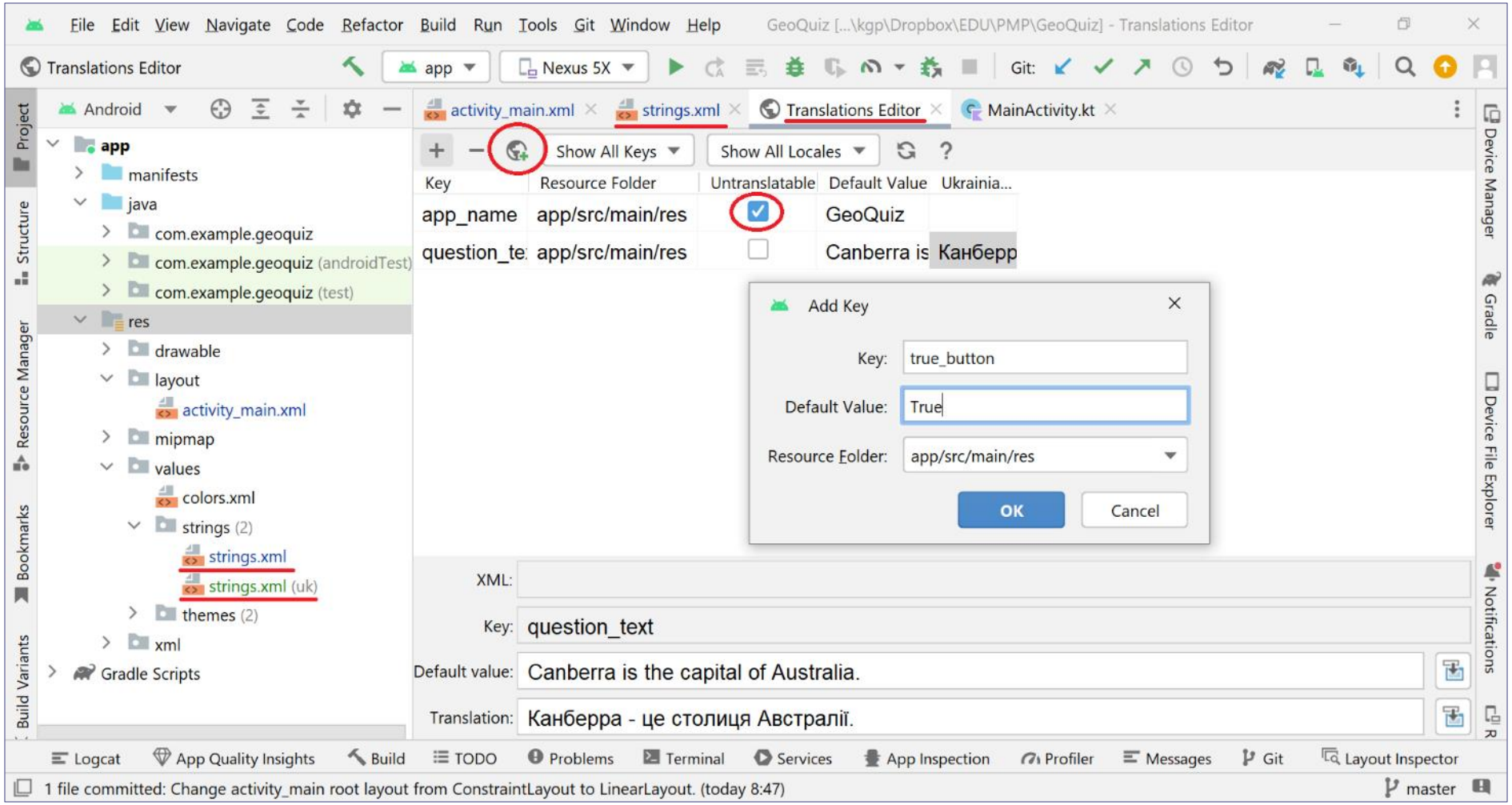
```
Text(
    text = "Hello $name!",
    modifier = modifier
        .background(color = Color.Red)
        .align(BiasAlignment.Horizontal(0.5f))
        .weight(2f)
)
```

Порядок модифікаторів важливий

Створення макету користувацького інтерфейсу активності



Створення рядкових ресурсів



res/values/strings.xml & res/values-uk/strings.xml

Створення рядкових ресурсів

```
<resources>
   <string name="app_name" translatable="false">GeoQuizComposeEdu</string>
   <string name="true_button">True</string>
   <string name="false_button">False</string>
   <string name="next_button">Next</string>
   <string name="question_australia">Canberra is the capital of Australia.</string>
   <string name="question_oceans">The Pacific Ocean is larger than the Atlantic
Ocean.</string>
   <string name="question_mideast">The Suez Canal connects the Red Sea and the Indian
Ocean.</string>
   <string name="question_africa">The source of the Nile River is in Egypt.</string>
   <string name="question_americas">The Amazon River is the longest river in the
Americas.</string>
   <string name="question_asia">Lake Baikal is the world\'s oldest and deepest freshwater
lake.</string>
</resources>
```



res/values/strings.xml

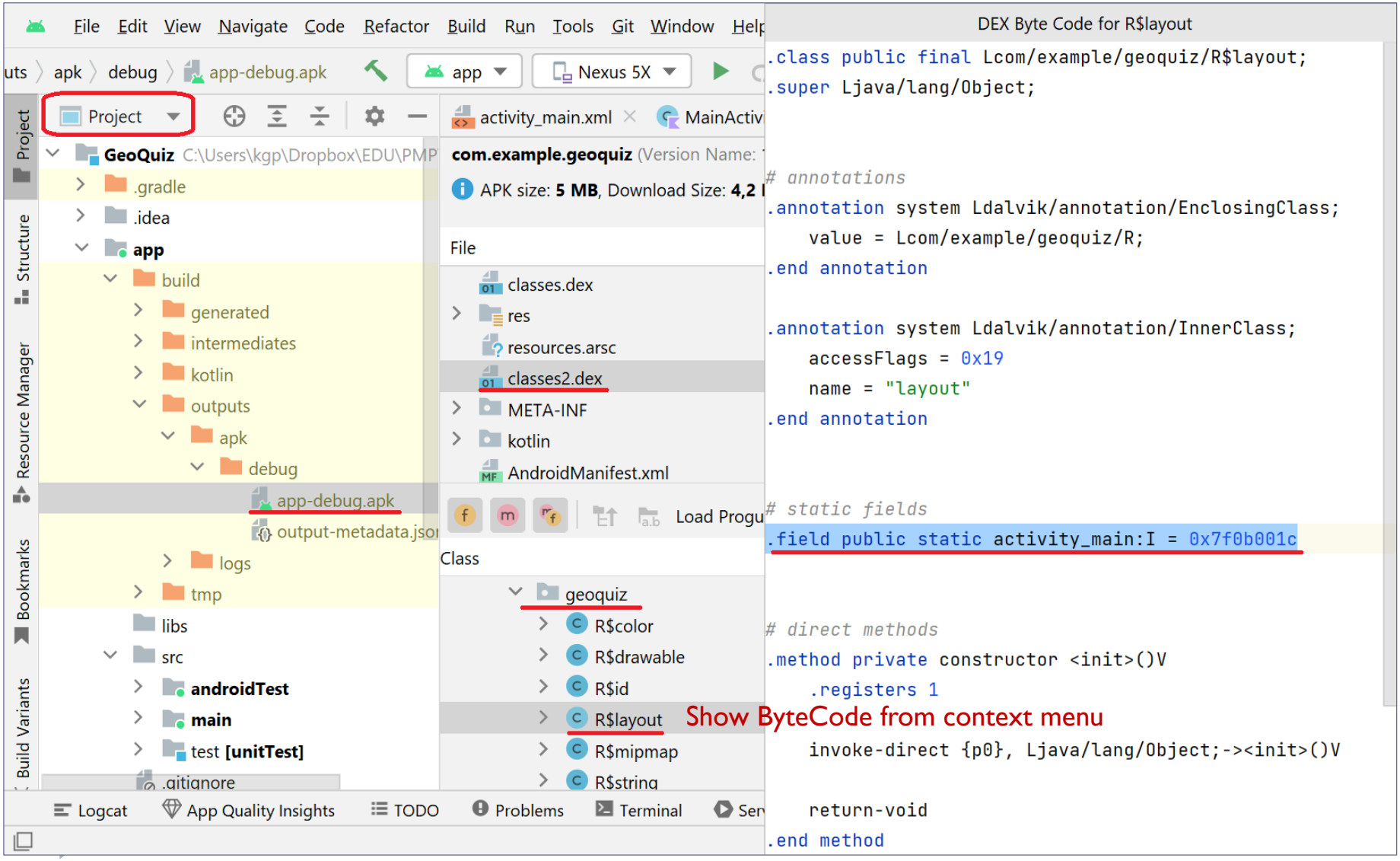
Створення рядкових ресурсів

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="true_button">Так</string>
    <string name="false_button">Ні</string>
    <string name="next_button">Далі</string>
    <string name="question_australia">Канберра - це столиця Австралії</string>
    <string name="question_oceans">Тихий океан більший, аніж Атлантичний
океан</string>
    <string name="question_mideast">Суецький канал поєднує Червоне море та
Індійський океан</string>
    <string name="question_africa">Витік річки Ніл знаходиться в Єгипті</string>
    <string name="question_americas">Амазонка є найдовшою річкою у Америці</string>
    <string name="question_asia">Озеро Байкал є найстарішим та найглибшим
прісноводним озером</string>
</resources>
```



res/values-uk/strings.xml

Перегляд файлу ресурсів R



Створення інтерфейсу головного екрану

- Створити пакет: ui.screens
- Створити файл MainScreen.kt з Composable-функцією MainScreen

```
package ua.edu.znu.geoquizcomposeedu.ui.screens

import ...

@Composable
fun MainScreen(
    innerPadding: PaddingValues,
) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(innerPadding),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { ...
```



Створення інтерфейсу головного екрану

```
...
    Text(
        text = stringResource(id = R.string.question_australia),
        modifier = Modifier.padding(innerPadding)
    )
    Row(
        modifier = Modifier.size(height = 90.dp, width = 200.dp),
        horizontalArrangement = Arrangement.SpaceEvenly,
        verticalAlignment = Alignment.CenterVertically
    ) {
        Button(onClick = { /*TODO*/ }) {
            Text(stringResource(id = R.string.true_button))
        }
        Spacer(modifier = Modifier.width(16.dp))
        Button(onClick = { /*TODO*/ }) {
            Text(stringResource(id = R.string.false_button))
        }

    }
...
```

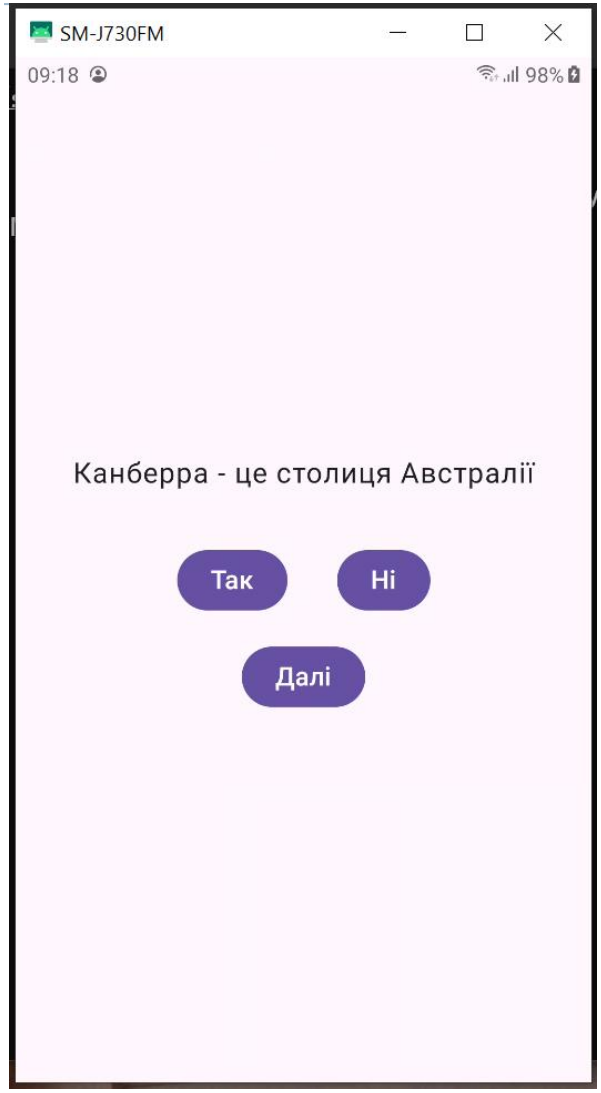
Створення інтерфейсу головного екрану

```
...
        Spacer(modifier = Modifier.width(16.dp))
        Button(onClick = { /*TODO*/ }) {
            Text(stringResource(id = R.string.next_button))
        }
    }
}

@Preview(showBackground = true)
@Composable
fun MainScreenPreview() {

    val innerPadding: PaddingValues = PaddingValues(16.dp)

    MainScreen(innerPadding, questionBank, currentIndex)
}
```



Створення інтерфейсу головного екрану

Одиниці розміру, які використовуються для представлення:

- ***піксели (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.

Додання класу моделі та списку його об'єктів

- Створити пакет: model
- Створити data class Question.kt з текстом питання та правильною відповіддю

```
package
ua.edu.znu.geoquizcomposeedu.model

data class Question(
    @StringRes val textResId: Int,
    val answer: Boolean
)
```



Додання класу моделі та списку його об'єктів

- Додати до функції MainScreen список запитань та індекс поточного питання у списку

```
@Composable
fun MainScreen(
    innerPadding: PaddingValues,
) {
    val questionBank = listOf(
        Question(textResId = R.string.question_australia, answer = true),
        Question(textResId = R.string.question_oceans, answer = true),
        Question(textResId = R.string.question_mideast, answer = false),
        Question(textResId = R.string.question_africa, answer = false),
        Question(textResId = R.string.question_americas, answer = true),
        Question(textResId = R.string.question_asia, answer = true)
    )
    var currentIndex = 0
...
```



Додання класу моделі та списку його об'єктів

- Отримати текст питання для компоненту Text зі списку відповідно до індексу

```
...
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = stringResource(id = questionBank[currentIndex].textResId),
            modifier = Modifier.padding(16.dp)        ) ...

@Preview(showBackground = true)
@Composable
fun MainScreenPreview() {
val innerPadding: PaddingValues = PaddingValues(16.dp)
    MainScreen(innerPadding)
}
```

Розробка класу активності

```
package ua.edu.znu.geoquizcomposeedu

import ...

class MainActivity : ComponentActivity() {

override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            GeoQuizComposeEduTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    MainScreen(innerPadding, questionBank, currentIndex)
                }
            }
        }
}
```

Використати увесь екран

