Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1
з дисципліни «Розробка мобільних застосувань під Android»

Виконала:
Одинюк О.М.
Група ІО-34
Номер у списку: 88
Перевірив:
Орленко С. П.

Київ – 2026

Тема: Дослідження роботи з елементами керування
Мета роботи: дослідити створення простого застосунку під платформу Андроїд та набути практичні навички з використання елементів керування інтерфейсу, мов програмування Java чи Kotlin.

Завдання:
Написати програму під платформу Андроїд, яка має інтерфейс для введення або/та вибору даних згідно варіанту (таблиця) і відображає результат взаємодії з цим інтерфейсом у деяке текстове поле цього інтерфейсу. Передбачити наступне: якщо не всі дані введені або обрані, а користувач натискає кнопку для отримання результату, то відобразити вікно, що спливає, з повідомленням завершити введення всіх даних.
Варіант:

3.	Вікно містить групу опцій (радіо-батонів) для вибору шрифту, текстове поле для введення інформації та кнопки «ОК» і «Cancel». Вивести введений рядок обраним шрифтом при натисканні на кнопку «ОК» у деяке текстове поле та очистити при натисканні кнопки «Cancel».

Скріншоти виконання:

Спершу робота на емуляторі:

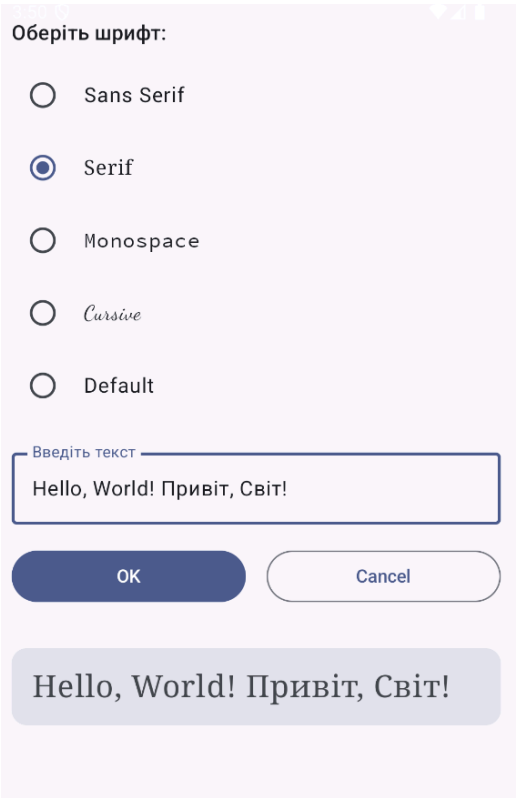

Рис. 1 – шрифт Serif

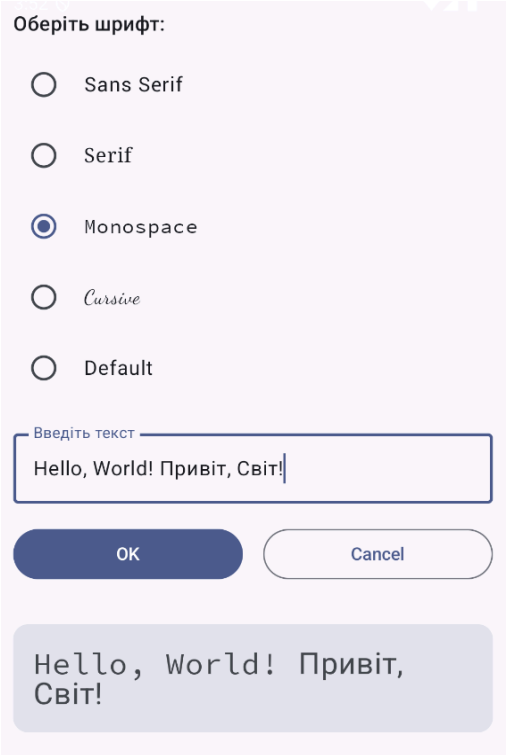

Рис. 2 – шрифт Monospace

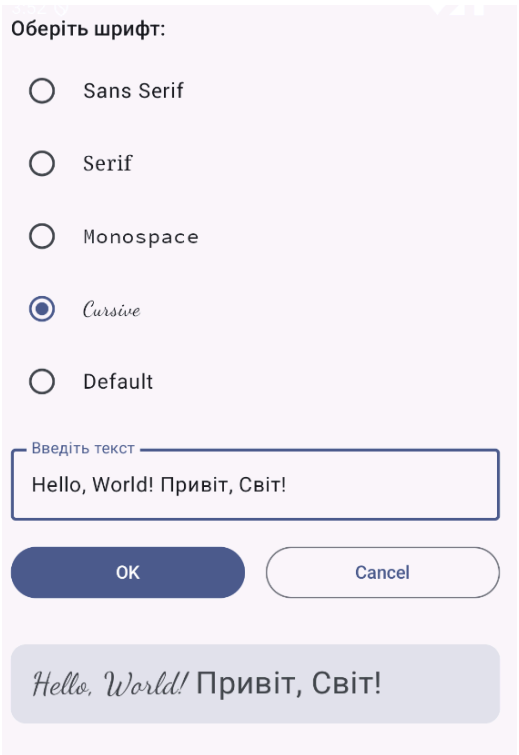

Рис. 3 – шрифт Cursive

Під час тестування розробленого додатка було виявлено, що при виборі певних шрифтових сімей (наприклад, **Cursive** або **Monospace**) візуальний стиль тексту для кириличних символів може залишатися стандартним, тоді як для латиниці він змінюється коректно.

Причина: Стандартні Android-шрифти (наприклад, FontFamily.Cursive), які базуються на системному сімействі Roboto або Dancing Script, часто не містять набору символів (гліфів) для кириличного алфавіту.

Можливе рішення: Для забезпечення стабільного відображення специфічних накреслень (курсив, моноширинний шрифт) для української мови, необхідно використовувати користувацькі шрифти. Їх потрібно завантажувати у форматі .ttf у папку res/font.

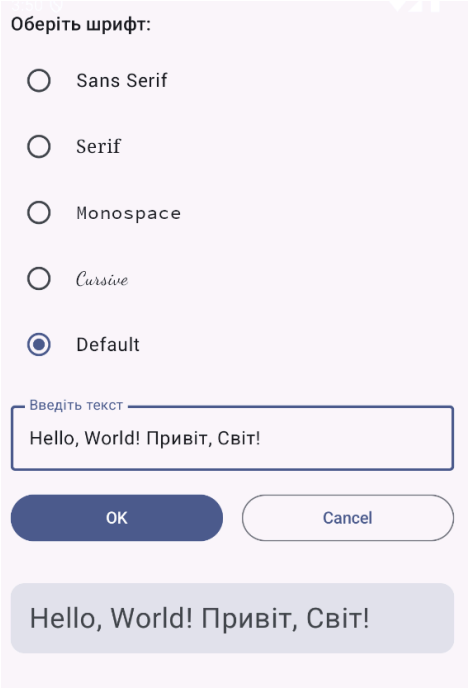

Рис. 4 – шрифт Default

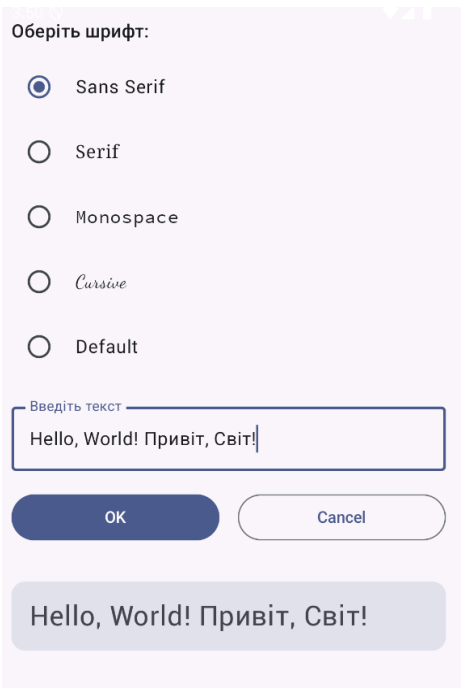

Рис. 5 – шрифт Sans Serif

Під час виконання лабораторної роботи також було помічено, що при перемиканні між варіантами FontFamily.SansSerif та FontFamily.Default візуальний вигляд тексту (як латиниці, так і кирилиці) залишається незмінним.

Причина: Починаючи з версії Android 4.1, основним системним шрифтом є **Roboto**. Цей шрифт належить до класу **Sans Serif** (гротески або шрифти без засічок). Оскільки FontFamily.Default в ОС Android зазвичай вказує саме на головний системний шрифт, він автоматично посилається на ту саму гарнітуру, що й FontFamily.SansSerif.

Тест на Samsung Galaxy A25:

Під час виконання роботи було проведено додаткове тестування додатка на власному мобільному пристрої. Метою було порівняння відображення системного шрифту за замовчуванням (Default) та шрифту без засічок (Sans Serif). Для наочності в налаштуваннях операційної системи Android було встановлено користувацький шрифт **Rosemary,** як основний системний шрифт.

Що призвело до цікавої проблеми:

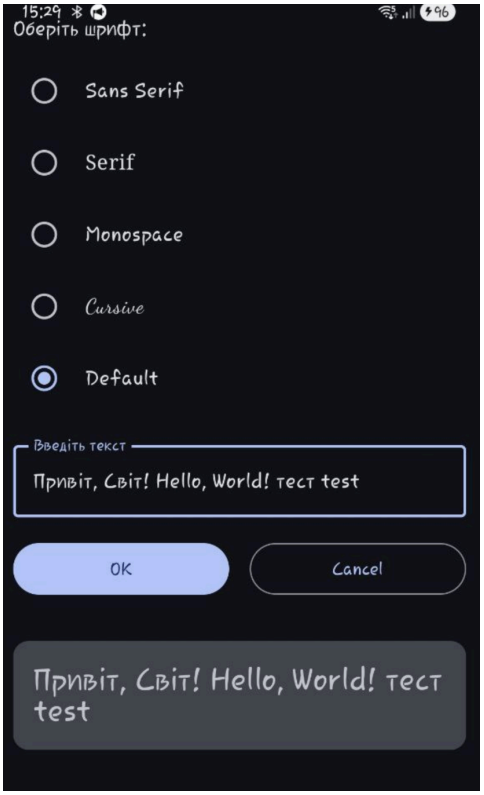

Рис. 6 – шрифт Default

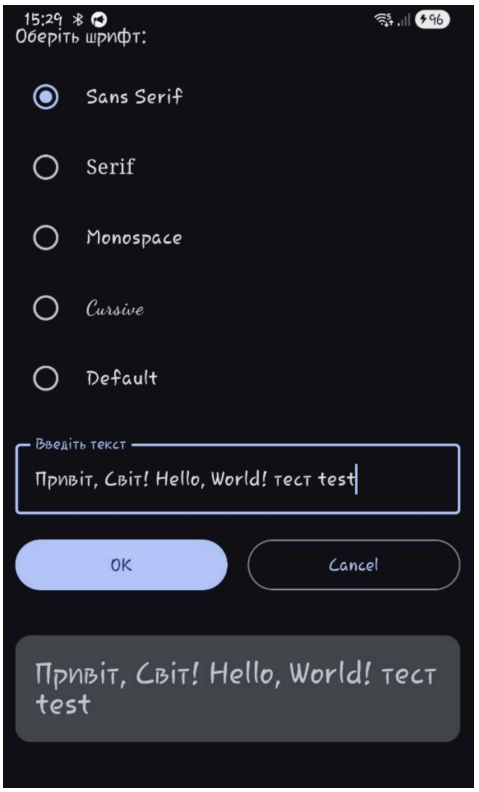

Рис. 7 – шрифт Sans Serif

Обидва варіанти демонструються як заданий Default шрифт **Rosemary.**

Причина: Дана поведінка зумовлена архітектурними особливостями ОС Android та бібліотеки Jetpack Compose:

1. В Android назви сімейств, такі як sans-serif, є логічними посиланнями (псевдонімами). Коли користувач змінює шрифт у налаштуваннях системи (наприклад, на Rosemary), ОС перепризначає це логічне посилання sans-serif на новий файл шрифту.
2. У Jetpack Compose константа FontFamily.Default зазвичай прямо вказує на поточне системне сімейство sans-serif. Оскільки користувацьке налаштування змістило стандартний шрифт (Roboto) на Rosemary для всієї системи, обидві змінні в коді почали звертатися до одного й того самого ресурсу в пам’яті пристрою.
3. Виробники смартфонів (Samsung, Xiaomi тощо) реалізують зміну шрифту так, щоб забезпечити цілісність інтерфейсу. Тому система ігнорує запит на «стандартний Sans Serif», підставляючи замість нього вибраний користувачем стиль, щоб додаток не вибивався з загального дизайну системи.

Моє рішення: Для розв’язання проблеми ідентичності шрифтів та забезпечення передбачуваного вигляду інтерфейсу, було прийнято рішення відмовитися від використання виключно системних констант на користь локальних ресурсів додатка.

Опис внесених змін:

1. Додавання шрифтових ресурсів: У папку res/font було додано зовнішні файли шрифтів у форматі .ttf: roboto_regular та sourcecodepro_regular.
2. Створення об'єктів FontFamily: У коді було оголошено власні змінні Roboto та SourceCodePro через функцію FontFamily(Font(R.font.name)).
3. Перепризначення мапінгу:
    - Пункт Sans Serif посилається на конкретний об'єкт Roboto.
    - Пункт Monospace посилається на SourceCodePro.
    - Пункт Default залишився системним (FontFamily.Default), щоб продемонструвати різницю з користувацьким налаштуванням пристрою.

Першочергово, код виглядав так:

```
val fonts = listOf(

    "Sans Serif" to FontFamily.SansSerif,
    "Serif" to FontFamily.Serif,
    "Monospace" to FontFamily.Monospace,
    "Cursive" to FontFamily.Cursive,
    "Default" to FontFamily.Default
```

Тепер вигляд має такий:

```
val fonts = listOf(
    "Sans Serif" to Roboto,
    "Serif" to FontFamily.Serif,
    "Monospace" to SourceCodePro,
    "Cursive" to FontFamily.Cursive,
    "Default" to FontFamily.Default
)
```

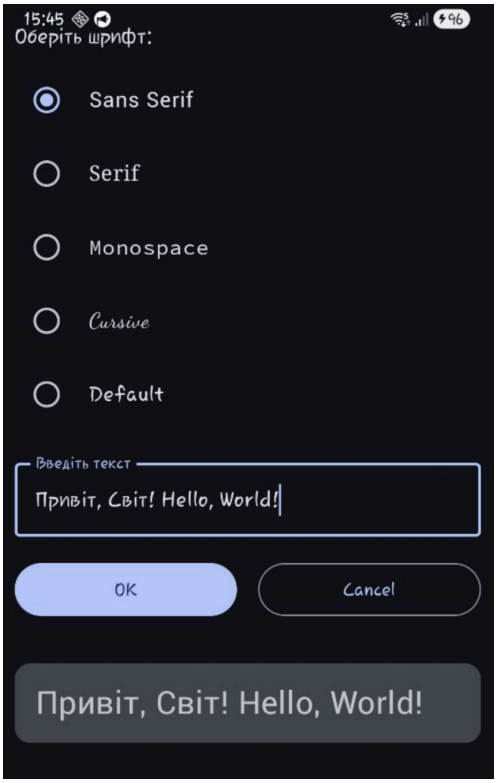

Рис. 8 – шрифт Sans Serif

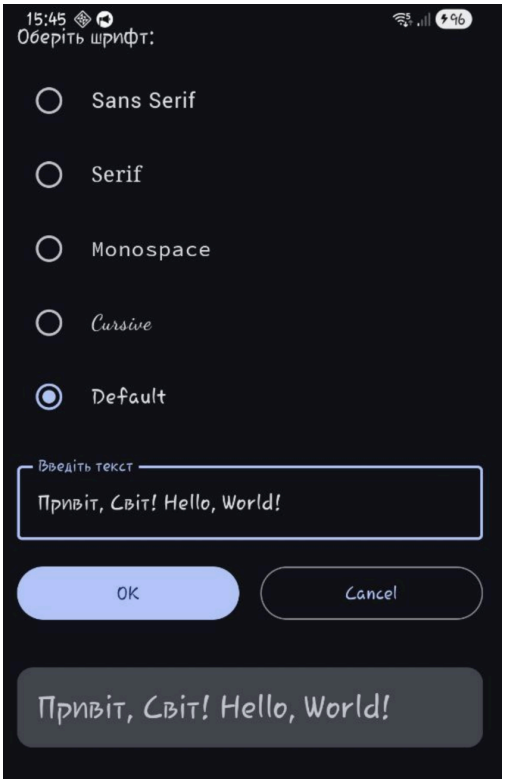

Рис. 9 – шрифт Default

Після введених змін, текст відображається коректно демонструючи шрифти.

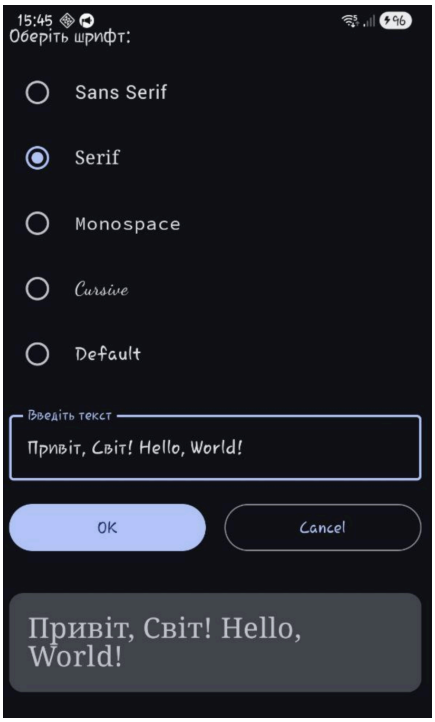

Рис. 10 – шрифт Serif

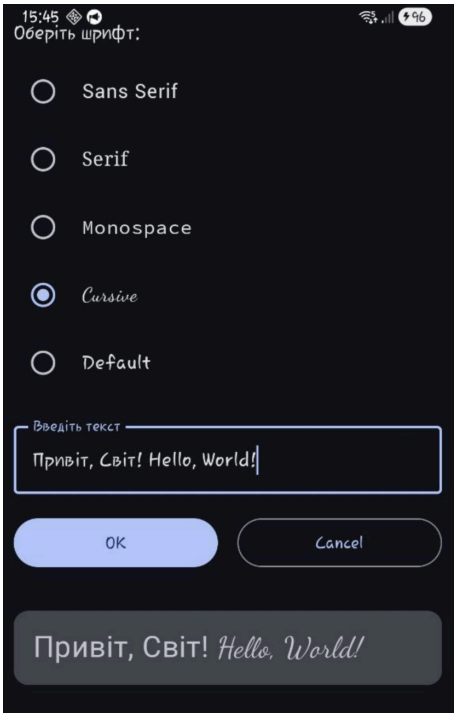

Рис. 11 – шрифт Cursive

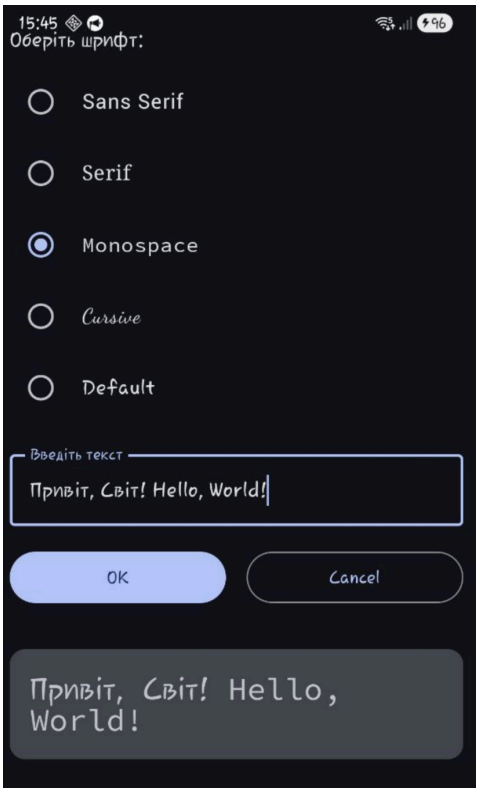

Рис. 12 – шрифт Monospace

Висновок:
Під час виконання лабораторної роботи було розроблено мобільний застосунок на базі Jetpack Compose для дослідження роботи з елементами керування, зокрема RadioButton, OutlinedTextField та Button.

У результаті проведених досліджень було встановлено:

- *Проблема локалізації системних шрифтів***:** Стандартні сімейства шрифтів Android (наприклад, Cursive) часто не мають підтримки кириличних гліфів, що призводить до автоматичної заміни (fallback) на стандартний системний шрифт при відображенні українського тексту.
- *Глобальні налаштування ОС:* Експеримент із встановленням користувацького шрифту **Rosemary** на реальному пристрої продемонстрував, що Android підміняє логічні посилання (aliases) на сімейство sans-serif. Через це FontFamily.Default та FontFamily.SansSerif візуально стають ідентичними.
- *Переваги кастомних шрифтів:* Використання локальних ресурсів (.ttf файлів у папці res/font) дозволило обійти обмеження системи,

забезпечивши стабільне відображення обраних гарнітур (Roboto, Source Code Pro) незалежно від налаштувань користувача та мови введення.

Набуто практичних навичок у створенні інтерфейсу, обробці станів (mutableStateOf) та управлінні шрифтовими ресурсами в Android-розробці.

Додатки:
Посилання на репозиторій з кодом програми:
https://github.com/wyrais/android_lab1

Посилання, з яких було завантажено шрифти:
https://github.com/tpn/fonts/blob/master/SourceCodePro-Regular.ttf

https://ua.fonts2u.com/roboto-regular.%D1%88%D1%80%D0%B8%D1%84%D1%82

Відповіді на контрольні запитання:

1. Архітектура застосунку під платформу Андроїд.

- **Linux Kernel:** Нижній рівень (керування пам'яттю, драйвери).
- **Hardware Abstraction Layer (HAL):** Стандартні інтерфейси для заліза.
- **Android Runtime (ART):** Виконує код програми (базується на DEX-файлах).
- **Java API Framework:** Набір сервісів, які розробники використовують для створення програм (Window Manager, Resource Manager тощо).
- **Applications:** Самі застосунки (системні та користувацькі).

2. Загальний огляд компонентів застосунку під Андроїд.

1. **Activities (Діяльності):** Візуальний інтерфейс користувача.
2. **Services (Служби):** Фонова робота без інтерфейсу.
3. **Broadcast Receivers (Приймачі оголошень):** Реагують на системні події (наприклад, низький заряд батареї).

4. **Content Providers (Постачальники вмісту):** Керування доступом до даних (своїх або системних).

3. Життєвий цикл компоненту «Діяльність».

Це послідовність станів, через які проходить екран. Основні методи:

- onCreate(): Створення вікна, ініціалізація даних.
- onStart(): Діяльність стає видимою.
- onResume(): Діяльність доступна для взаємодії (фокус).
- onPause(): Втрата фокусу (частково перекрита іншим вікном).
- onStop(): Діяльність більше не видима.
- onDestroy(): Повне знищення (вивільнення пам'яті).

4. Життєвий цикл компоненту «Служба».

Залежить від типу запуску:

- **Started Service (startService()):** onCreate() -> onStartCommand() -> Працює -> onDestroy(). Працює, доки сама не зупиниться або її не вб'є система.
- **Bound Service (bindService()):** onCreate() -> onBind() -> Клієнт взаємодіє -> onUnbind() -> onDestroy(). Працює, поки є прив'язані клієнти.

5. Опис процесів платформи Андроїд.

За замовчуванням кожен застосунок працює у власному Linux-процесі. Android керує ними за пріоритетністю (якщо пам'яті мало, система вбиває процеси з нижчим пріоритетом):

1. **Foreground process:** Те, що зараз на екрані.
2. **Visible process:** Видимий, але не активний (наприклад, діалогове вікно зверху).
3. **Service process:** Працює фонова служба (музика).
4. **Cached process:** Невидимий процес (у стеку перемикання), зберігається для швидкого повторного запуску.

6. Яким чином активуються компоненти застосунку.

Більшість компонентів (Activity, Service, Broadcast Receiver) активуються через **Intent** (Намір). Це асинхронне повідомлення, яке каже системі «я хочу щось зробити».

- **Явні Intent:** Вказують конкретний клас (наприклад, MainActivity).
- **Неявні Intent:** Вказують дію (наприклад, «відкрити посилання у браузері»). Content Providers активуються через запити до ContentResolver.

7. Призначення файлу маніфесту та його структура.

AndroidManifest.xml – це паспорт вашого додатка.

- **Призначення:** Опис компонентів, дозволів (Permissions), мінімальної версії Android та апаратних вимог.
- **Структура:**
  - <manifest>: Корінь, вказує package name.
  - <uses-permission>: Дозволи (інтернет, камера).
  - <application>: Містить описи компонентів (<activity>, <service>, <receiver>, <provider>).
  - <intent-filter>: Описує, на які події реагує компонент всередині тегів компонентів.

8. Поняття ресурсу та яким чином визначаються ресурси.

Ресурси – це всі зовнішні елементи, які не є кодом (рядки, картинки, макети інтерфейсу, кольори). Вони зберігаються в папці res/.

- **Визначення:** Кожен ресурс отримує унікальний ID у згенерованому класі R.java.
- **Доступ:**
  - У коді Java/Kotlin: R.string.hello_world.
  - У XML: @string/hello_world. Це дозволяє легко робити локалізацію (різні мови для різних папок values-uk, values-en).