МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

ШТЕФАН Н. З.

МОБІЛЬНІ ТЕХНОЛОГІЇ

(частина 1)

Конспект лекцій

Одеса
Одеський державний екологічний університет
2023

УДК 004.4(075.034)
Ш91

Штефан Н. З.
Ш91. Мобільні технології: конспект лекцій (частина 1). Одеса: Одеський державний екологічний університет, 2023. 89 с.

Конспект лекцій з дисципліни «Мобільні технології»(Частина 1) призначений для здобувачів вищої освіти (бакалаврів), які навчаються за науковою спеціальністю 122 – комп'ютерні науки.

У конспекті подаються основи з розробки мобільних додатків, необхідних для успішного засвоєння цієї дисципліни. Кожна тема лекції містить: план лекції, основні пояснення до теми, методичні поради щодо вивчення кожної теми, питання для самоконтролю. Наприкінці конспекту подається термінологічний словник за матеріалами кожного змістовного модуля, а у кінці усіх тем – предметний покажчик.

Рекомендовано методичною радою Одеського державного екологічного університету Міністерства освіти і науки України як конспект лекцій (протокол №2 від 27.10.2023 р.)

ISBN 978-966-186-283-7

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

AOSP – Android Open Source Project
CDC – Connected Device Configuration
CLDC – Connected Limited Device Configuration
JDK – Java Development Kit
J2ME – Java 2 Micro Edition
MIDP – Mobile Information Device Profile
OHA – Open Handset Alliance
ОС – операційна система

AndroidStudio – середа розробки
Dalvik – віртуальна машина Java
C# – мова програмування
Java – мова програмування
iOS – операційна система
Python – мова програмування
Eclipse – середа розробки

ВСТУП

Конспект лекцій розроблено відповідно до силлабусу «Мобільні технології» для студентів рівня вищої освіти бакалавр за спеціальністю 122 «Комп'ютерні науки». Матеріал лекційного курсу присвячено вивченню теоретичних основ розробки додатків, призначених для функціонування на мобільних пристроях під управлінням ОС Android.

В результаті засвоєння лекційного матеріалу студенти буде знати:

- загальну характеристику найпопулярніших мобільних платформ;
- архітектуру та особливості платформи Android;
- засоби та можливості середовища Android Studio, зокрема, щодо відлагодження розроблюваних програм з використанням емулятора;
- основи проектування та розробки рішень під платформу Android на мові програмування Java включаючи, використання апаратних особливостей мобільних пристроїв.

В результаті засвоєння лекційного матеріалу студенти буде вміти:

- використовувати основні класи і типи даних мови Java та можливості SDK, призначені для розробки додатків під мобільну платформу;
- застосовувати середовище Android Studio та відповідний емулятор;
- проектувати інтерфейс програми, в тому числі із застосуванням різноманітних елементів управління;

Для успішного засвоєння матеріалу студенти повинні володіти базовими знаннями з теорії алгоритмів, програмування та методології об'єктно-орієнтованого програмування. Необхідним є володіння на базовому рівні об'єктно-орієнтованою мовою програмування Java.

Освоєння дисципліни є однією із передумов для підготовки та захисту студентами кваліфікаційної роботи.

ТЕМА 1 «МОБІЛЬНІ ТЕХНОЛОГІЇ: ЕВОЛЮЦІЯ, РИНОК, СУЧАСНИЙ СТАН»

План лекції.

1. Сучасний стан ринку мобільних додатків.
2. Операційні системи для мобільних пристроїв.
3. Історія виникнення Android.
4. Переваги та недоліки ОС Android.

Пояснення до теми.

1.1 Сучасний стан ринку мобільних додатків

У 2022 році кількість завантажень мобільних додатків у всьому світі зросла на 11% порівняно з 2021 роком. Це означає, що інтерес до мобільних додатків зростає з кожним роком.

Провідним ринком мобільних додатків за кількістю завантажень був Китай – у 2022 році він перевищив 110 мільярдів завантажень. Це майже половина всіх глобальних завантажень разом. Індія та Сполучені Штати були на другому та третьому місцях з 28 мільярдами та 12 мільярдами відповідно.

Лише на шість категорій додатків припадає більше половини загальних витрат споживачів на мобільні додатки? Що ще дивніше, так це те, що більшість часу, проведеного в додатках, можна віднести лише до трьох категорій.

У 2022 році додатки для потокового відео були першими як за завантаженнями, так і за споживчими витратами з часткою 16%. Disney+ був провідним додатком у цьому піджанрі. Додатки в піджанрах знайомств і коротких відео були другим і третім за споживчими витратами. Люди витрачали найбільше часу на комунікаційні програми, і WeChat був додатком номер 1 у цій категорії.

Як і у будь-якого комп’ютерного пристрою, у мобільних пристроїв є операційні системи. Мобільна операційна система (мобільна ОС) – це операційна система для смартфонів, планшетів, КПК або інших мобільних пристроїв. Мобільні операційні системи поєднують в собі функціональність ОС для ПК з функціями для мобільних і кишенькових пристроїв: сенсорний екран, стільниковий зв’язок, Bluetooth, Wi-Fi, GPS-навігація, камера,

відеокамера, розпізнавання мови, диктофон, музичний плеєр, NFC (зв’язок ближнього поля) та інфрачервоне дистанційне управління.

1.2 Операційні системи для мобільних пристроїв

1.2.1 Android

Сучасні операційні системи для мобільних пристроїв: Android, iOS, CyanogenMod, Cyanogen OS, Fire OS, Flyme OS, Firefox OS, Sailfish OS, Tizen, Ubuntu Touch. Android – операційна система для мобільних пристроїв: смартфонів, планшетних комп'ютерів. В даний час саме Android є широко використовуваною операційною системою для мобільних пристроїв.

Android - це операційна система з відкритим вихідним кодом для мобільних пристроїв і відповідний проект з відкритим вихідним кодом, очолюваний Google. Цей сайт і репозиторій Android Open Source Project (AOSP) пропонують інформацію та вихідний код, необхідні для створення варіантів ОС Android, перенесення пристроїв і аксесуарів на платформу Android, а також забезпечення відповідності пристроїв вимогам сумісності, що підтримують екосистему Android. здорове та стабільне середовище для мільйонів користувачів.

Як проект із відкритим вихідним кодом, мета Android полягає в тому, щоб уникнути будь-якої центральної точки відмови, в якій один гравець галузі може обмежувати або контролювати інновації будь-якого іншого гравця. З цією метою Android являє собою повнофункціональну операційну систему виробничої якості для споживчих товарів з вихідним кодом, що настроюється, який можна портувати практично на будь-який пристрій, і загальнодоступною документацією, доступною для всіх

Перша версія Android була представлена 23 вересня 2008 року, версії було дано назву Apple Pie (можна помітити співзвуччя із прямим конкурентом). Операційна система і платформа для мобільних телефонів та планшетних комп’ютерів, створена компанією Google на базі ядра Linux. Підтримується альянсом Open Handset Alliance (OHA). Хоча Android базується на ядрі Linux, він стоїть дещо осторонь Linux-спільноти та Linux-інфраструктури. Базовим елементом цієї операційної системи є реалізація Dalvik – віртуальної машини Java, і все програмне забезпечення і застосування спираються на цю реалізацію Java.

З найбільш відомих на вітчизняному ринку – це планшети і смартфони Samsung, HTC, Huawei, SONY, LG, Lenovo. Перша версія ОС вийшла у світ в 2008 році на смартфоні HTC, і з тих пір невпинно оновлюється. Основою

популярності Android є те, що Android можна встановлювати на пристрої різних виробників.

Переваги ОС Android:

- різноманітність програм та ігор на Android;
- швидка інтеграція з сервісами Google;
- абсолютна незалежність від апаратної начинки мобільного пристрою;
- Android є екосистемою з відкритим кодом;
- багатозадачність – це означає, що без проблем працює одночасно кілька програм;
- легкість встановлення програм із різних ресурсів;
- широкі можливості індивідуалізації;
- зручне оновлення через Інтернет, причому оперативне – ведеться безперервна робота над поліпшенням функціоналу ОС, виправляються баги, вносяться зміни в інтерфейс;
- можливість заміни / видалення дефолтних програм;
- максимально спрощено виробництво програм, ігор, плагінів, оновлень. Стрімко набирає обертів нова професія – розробник додатків під Андроїд. Побутує думка, що краща мова програмування для Android – Java, далі C#, Python.

Недоліки ОС Android:

- пристрій під управлінням ОС Android доводиться досить часто заряджати/підзаряджати;
- існують деякі проблеми сумісності нових версій операційної системи зі знятими з виробництва, але присутніми в обігу застарілими пристроями, або ж із пристроями, які випущені маловідомими китайськими виробниками;
- користувачі, у яких на першому місці комфорт від швидкості роботи, можуть виявитися не задоволеними як нею в цілому, так і кількістю налаштувань.

1.2.2 iOS

Для смартфонів, планшетів та ін. пристроїв, розроблена легендарною компанією Apple, виключно під себе, на відміну від Windows Phone (Microsoft) і Android (Google). Вперше операційна система була представлена в 2007 році разом з телефоном iPhone. Спершу створювалася для iPhone і IPod Touch, пізніше для таких пристроїв, як iPad і Apple TV.

Ядро iOS майже ідентичне ядру операційної системи Apple MacOS (раніше іменувалася OS X), тому використовує такий самий набір основних компонентів.

Переваги iOS:

- зручність в роботі;
- шикарний дизайн;
- акцент на надійність і якість ОС;
- оптимізація сприяє збільшенню тривалості роботи гаджета при повному його завантаженні;
- відсутність будь-яких сповільнень, «глюків», як у програмах, так і на робочому екрані пристроїв;

величезний вибір додатків;

- у програмах, на відміну від Google Play, рідко зустрічається реклама;
- доступність оновлень відбувається відразу після випуску нової версії ОС для всіх пристроїв одночасно;
- розробники щоразу анонсують свої програми для IOS;
- можливість об'єднати роботу оновлених до останньої ОС мобільних пристроїв;
- великий плюс – тривала підтримка старих пристроїв;
- сімейний доступ для покупок в App Store;
- багатозадачність. Передбачена спеціальна панель-меню багатозадачності.

Недоліки iOS:

- закрита, досить консервативна система – все програмне забезпечення, за кількома винятками, є платним. Причому, такий варіант характерний для всіх пристроїв Apple – від смартфонів iPhone до планшетів iPad;
- інколи користувачеві доведеться переплачувати за додатки;
- пряме копіювання файлів чи передача з'явилася лише нещодавно (переміщення можливе було тільки за допомогою iTunes);
- в iOS все зав'язано на інтернеті (коли немає підключення, то ви втрачаєте багато функцій);
- працює тільки на пристроях компанії Apple, що значно зменшує поширення iOS по всьому світі. iPad, iPhone, iPod, Macbook, iMac і iWatch найбільш поширені у заможних країнах – на території США, Європи. Як операційна система iOS була представлена з iPhone на Macworld Conference

& Expo 9 січня 2007 року і випущена в червні того ж року. Спершу, Apple не вказувала її ім’я, просто заявивши, що «iPhone використовує OS X».

Питання для самоконтролю за темою

Для самостійної перевірки знань доцільно сформулювати розширені відповіді на поставлені питання і перевірити їх повноту та правильність за допомогою матеріалів запропонованих літературних джерел.

1 Які ОС для мобільних пристроїв ви знаєте? Дайте характеристику кожному з них.

2 Чому Android швидко набирає оберти серед споживачів?

3 Наведіть основні переваги Android.

4 Які недоліки ОС iOS ви можете назвати?

ТЕМА 2 «ПІДГОТОВКА СЕРЕДОВИЩА РОЗРОБКИ ДОДАТКІВ ПІД ANDROID. ОСНОВИ СТВОРЕННЯ ІНТЕРФЕЙСУ»

План лекції.

Огляд середовищ розробки мобільних додатків
Налаштування Android Studio
Перший проект
Вибір емулятору

Пояснення до теми.

2.1 Огляд середовищ розробки мобільних додатків

Індустрія розробки мобільних додатків переживає етап трансформації. З розвитком технологій мікропроцесорів ви зможете запускати мобільні програми на багатьох платформах. Наприклад, програми, створені для мобільних пристроїв, у найближчі роки безперебійно працюватимуть на комп'ютерах. Крім того, мобільні програми, створені за допомогою Flutter або React Native, працюватимуть на телефонах Android, iPhone, Mac, а також ПК. Але перш ніж приступати до подробиць, давайте покопаємося в статистиці використання мобільних додатків, щоб зрозуміти, як користувачі проводять час за допомогою мобільних телефонів і які можливості є у створенні власного мобільного додатку.

Проаналізуємо основні технології, які використовуються для розробки додатків для мобільних пристроїв. Перша технологія-це Java 2 Micro Edition (J2ME). Це набір специфікацій і технологій, призначених для різних типів портативних пристроїв. Існують два основні напрями: Connected Device Configuration (CDC) і Connected Limited Device Configuration (CLDC). Напрям визначає тип конфігурації центральних бібліотек Java, а так само параметрів віртуальної машини Java (в якій будуть використовуватися додатки).

Пристрої, які використовують технологію CDC будуть більш розвиненими, в якості прикладу можна навести комунікатори. До пристроїв CLDC відносяться звичайні мобільні телефони, які апаратно володіють більш обмеженими можливостями (ресурсами).

Спеціальні режими дозволяють визначати функціональність конфігурацій для різних типів пристроїв. Режим Mobile Information Device

Profile (MIDP) призначений для CLDC портативних пристроїв з можливістю спілкування. Режим MIDP визначає функціональність – роботу користувальницького інтерфейсу, збереження налаштувань, роботу в мережі і модель додатка. CLDC і MIDP закладають основу реалізації J2ME. Java-код інтерпретується безпосередньо самим пристроєм за допомогою так званої Java Virtual Machine. Цей механізм робить можливим вільне розповсюдження Java - додатків, так як вони працюють на всіх пристроях з аналогічною Java –платформою.

Програмування Java-додатків і на сьогоднішній день займає більшу частину, так як більшість мобільних пристроїв (в основному мобільні телефони) в світі мають вже встановлену Java - систему.

Наступна популярна на сьогоднішній день технологія – це технологія Qt. Вона в основному використовується в якості крос-платформного середовища, яке дозволяє використовувати написані з її допомогою додатки на різних пристроях і операційних системах, у тому числі і Android, а також Windows, Mac OS X, Linux, Symbian.

Оскільки фрагментація (різниця у версіях операційної системи) Android має великий вплив на стабільну роботу мобільного додатку, перевірка повного функціоналу на цільових операційних системах вимагає великої кількості ресурсів та часу. В такому випадку використовується система моніторингу роботи мобільного додатку з інфраструктури Google.

Сьогодні Android SDK можна використовувати з різними IDE:

· Eclipse;

· NetBeans;

· AndroidStudio.

Eclipse – середовище для розробки додатків під Android з встановленим плагіном ADT. Розробка ведеться на мові програмування Java. Є можливість налагодження з використанням емулятора вбудованого в ADT або безпосередньо на мобільному пристрої з ОС Android. Тут можливість більш низькорівневої розробки з використанням Android NDK (Native Development Kit) на мові C/C++. Середовище розробки Eclipse надає користувачам можливість виконувати UNIT тести на мобільних пристроях та на емуляторі . Для виконання тестів створюється окремий проект , що зручно дозволяє залишати основний проект без суттєвих змін.

Android Studio - середовище розробки на основі IntelliJ IDEA, що надає інтегровані інструменти для розробки та налагодження додатків для платформи Android. Android Studio містить Android SDK, інструменти для

розробки дизайну, тестування і налагодження, останню версію платформи Android для компіляції.

Основні властивості Android Studio:

· Редактор WYSIWYG. Рендеринг додатків в реальному часі.

·Консоль розробника: підказки для оптимізації, помічник для перекладу, відстеження рефералів, кампанії і просування.

· Підтримка білдів на основі Gradle.

· ProGuard і можливості підпису додатків.

NetBeans IDE також підтримує інтеграцію Android SDK через плагін, який розроблено третіми особами. Деяка функціональність плагіна стає доступною лише після оформлення платної підписки. Через це NetBeans IDE використовувати недоцільно.

Для розробки додатку бакалаврської роботи буде використане середовище Android Studio. Адже воно має всі необхідні інструменти для комфортної розробки і тестування. Відсутність низькорівневої розробки NDK Support не є суттєвою, а нова система складання проекту Gradle є більш ефективною.

2.2 Налаштування Android Studio

З метою прискорення виконання робіт, Android Studio надає колекцію елементів для створення інтерфейсу і вбудований візуальний редактор. Крім цього в ній є можливість попереднього перегляду елементів управління в різних дозволах. Однак стандартними наборами справа не обмежується, ніхто не забороняє вам зробити свій індивідуальний дизайн.

2.2.1 Установка JDK

Перед тим як створити додаток для Android, потрібно встановити Android Studio або будь-який інший компілятор. Однак жоден такий продукт не буде правильно працювати без попередньої інсталяції JDK (Java Development Kit). Поширюється цей компонент абсолютно безкоштовно і може бути завантажений з офіційного сайту компанії Oracle. Будьте уважні при виборі розрядності: програма для 64-бітної ОС не може працювати в 32-розрядної Windows.

Установка не викличе проблем навіть у початківця користувача. Майже всі кроки можна пропустити, натиснувши кнопку “Next”. Винятком стане лише вікно вибору місця для інсталяції. Ви можете вказати будь-який

каталог, проте варто віддати перевагу шлях, в якому не буде пропусків і кириличних символів.

2.2.2 Установка середовища розробки

Перед тим як створити додаток для Android, необхідно встановити і середовище розробки. Завантажити Android Studio можна абсолютно безкоштовно на офіційному сайті розробників операційної системи Android. Інсталяційний файл включає в себе ще і Android SDK, тому його розмір дорівнює приблизно 500 Мб.

Запустіть файл, завантажений з інтернету.

Далі піде уточнення: встановити програму для одного користувача або для всіх?

Після натискання на кнопку “Next” буде запропоновано ввести бажане розташування системних файлів програми.

Натисніть ще кілька разів на кнопку “Далі”, щоб перейти до процесу розпакування всіх необхідних для Android Studio компонентів.

Після завершення інсталяції натисніть кнопку “Finish”, відразу ж після цього відкриється головне вікно середовища розробки.

2.3 Інструменти розробника

Як правило, розробка Android-додатків здійснюється на мові Java. Тому, в першу чергу, необхідно встановити Java Development Kit.

Java – це об’єктно-орієнтована мова програмування. Програми на ній транслюються в байт-код, що виконується віртуальною машиною Java, яка обробляє байт-код і передає інструкції обладнанню як інтерпретатор. Перевага подібного способу виконання програм полягає в повній незалежності байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-додатки на будь-якому пристрої, для якого існує відповідна віртуальна машина. Іншою важливою особливістю Java є гнучка система безпеки завдяки тому, що виконання програми повністю контролюється віртуальною машиною.

Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з’єднання з іншим комп’ютером) викликають негайне їх переривання. Слід зауважити, що фактично, більшість архітектурних рішень, прийнятих при створенні Java, було продиктовано бажанням надати синтаксис, схожий з C/C++.

Android Software Development Kit (SDK) містить багато інструментів і утиліт для створення і тестування додатків. Наприклад, за допомогою SDK Manager можна встановити Android API будь-якої версії (рис. 2.1), а також перевірити репозиторій на наявність доступних, але ще не встановлених пакетів і архівів

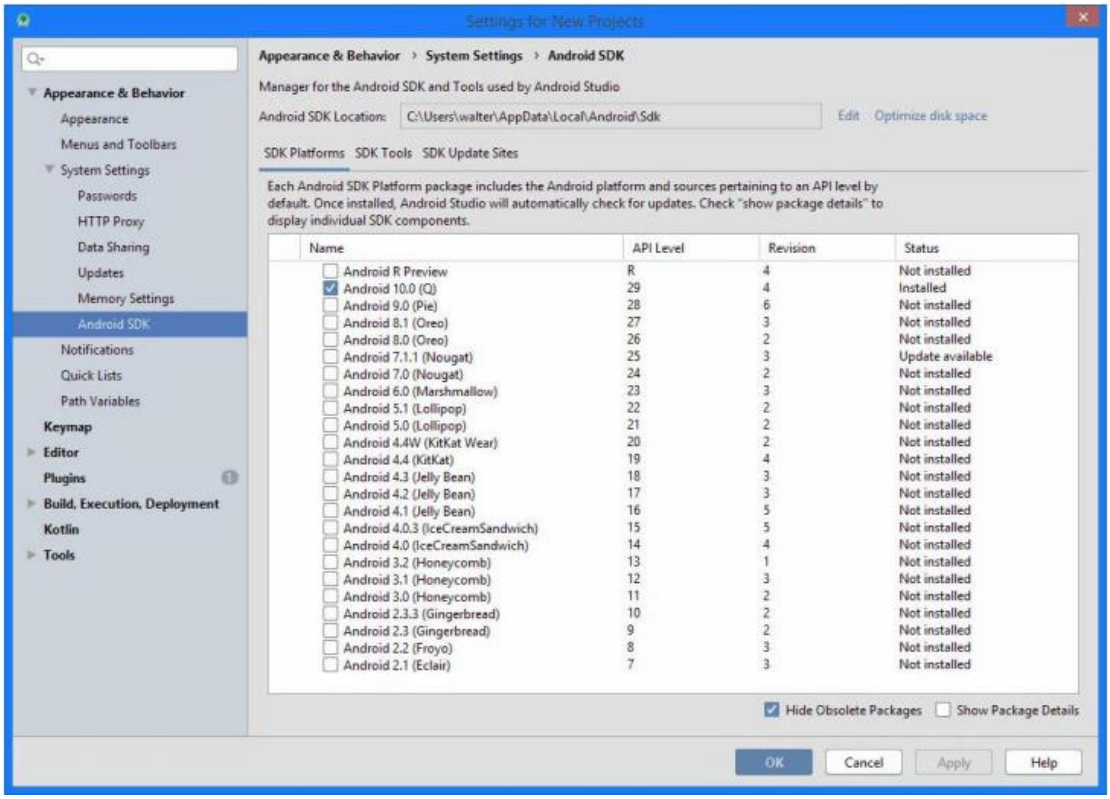


Рисунок 2.1– Вікно Android SDK

JDK (Java Development Kit – це безкоштовно розповсюджуваний комплект розробки додатків на мові Java, що включає в себе компілятор Java, стандартні бібліотеки класів Java, зразки коду, документацію, різні утиліти і систему Java Runtime Environment (JRE). До складу JDK не входить інтегроване середовище розробки IDE(Integrated Development Environment). Тому після того, як буде встановлено JDK, слід встановити IDE.

2.4 Емулятори

Емуляція (англ. Emulation) в обчислювальній техніці – комплекс програмних, апаратних засобів або їх поєднання, призначений для копіювання (або емуляції) функцій однієї обчислювальної системи (гостя)

на інший, відмінній від першої, обчислювальної системі (хост) таким чином, щоб поведінка, яка емулюється, як можна ближче відповідала поведінці оригінальної системи (гостя).

Метою є максимально точне відтворення поведінки на відміну від різних форм комп’ютерного моделювання, в яких імітується поведінка деякої абстрактної моделі. Емулятор – це віртуальний мобільний пристрій, який запускається на комп'ютері. За допомогою емулятора можна розробляти і тестувати програми без використання реальних пристроїв. На рис. 2.2 наведено приклад запущеного стандартного емулятора.

Рисунок 2.2– Стандартний емулятор Android

До переваг використання емуляторів можна віднести простоту їх використання і низьку вартість. Розробнику не потрібно купувати величезну кількість пристроїв з різними характеристиками, щоб перевірити працездатність програми на різних смартфонах.

Досить створити кілька емуляторів з необхідними характеристиками і запустити на них додаток. На жаль, емулятори мають і ряд недоліків:

1) вимагають багато системних ресурсів;

2) через відмінності в архітектурі процесорів комп’ютера і смартфона повільно запускаються. Сучасні персональні комп’ютери побудовані на

архітектурі x86 і x64, а більшість процесорів смартфонів на Android – ARM. Процес емуляції однієї архітектури на інший надзвичайно складний і відбувається досить повільно;

3) у деяких випадках стандартного емулятора недостатньо. Йдеться про можливості смартфонів, якими звичайні комп'ютери не володіють (наприклад, наявність датчика GPS або акселерометра).

У таких випадках повноцінне відлагодження можна провести тільки з використанням реального пристрою

2.4.1 Альтернативні емулятори

Стандартний емулятор, що поставляється разом з Android SDK, не влаштовує багатьох. Існують проекти, що підтримують розробку та розвиток альтернативних емуляторів. Як приклад можна привести Genymotion (https://www.genymotion.com/) – швидкий емулятор Android (на думку його розробників). Він містить попередньо налаштовані образи Android (x86 з апаратним прискоренням OpenGL). Genymotion доступний для Linux, Windows і MacOS X і вимагає для своєї роботи VirtualBox. Іншими словами, Genymotion є віртуальною машину з встановленою ОС Android, яку користувач запускає так само, як і інші віртуальні машини.

2.4.2 Можливість відлагодження на реальних пристроях

Розроблений додаток можна запустити на реальному пристрої, наприклад, на смартфоні. Для цього необхідно виконати попередню роботу.

Для відлагодження додатків на реальних пристроях, як правило, необхідно:

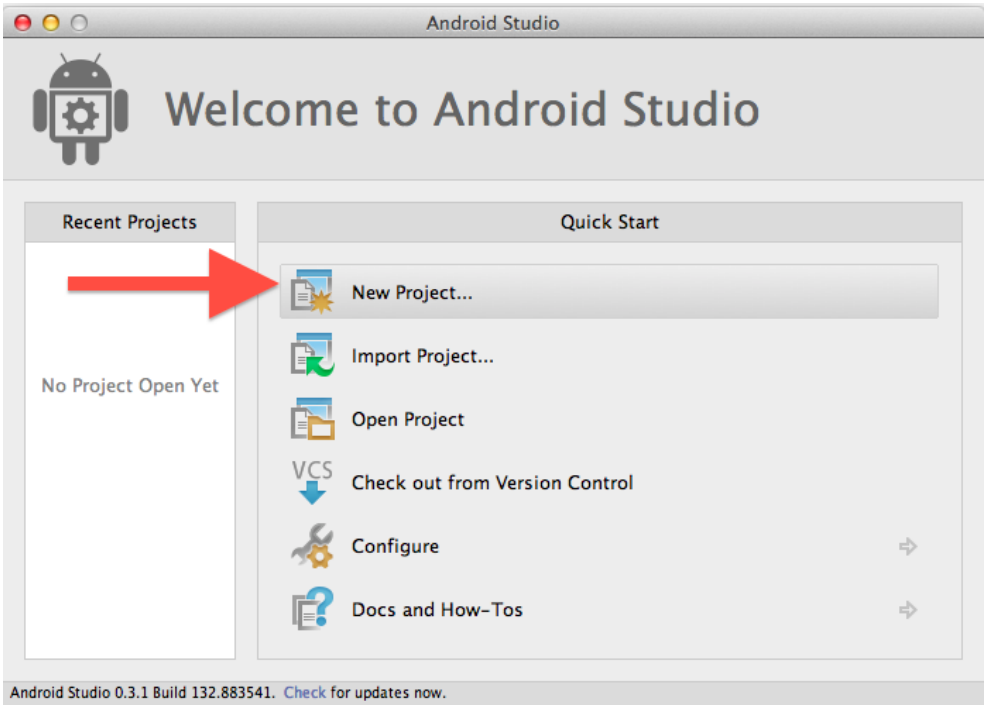
1. налаштувати пристрій (включити режим налагодження по USB);
2. налаштувати комп'ютер (для Windows необхідно встановити потрібний драйвер вручну, потрібні права адміністратора);
3. налаштувати середовище розробки і запустити додаток на пристрої.

2.5 Перший додаток

Після завершення установки "Студії" можна перейти безпосередньо до розробки своєї першої програми. Класика жанру – створення програми "Hello, World".

Відкрийте Android Studio, якщо середовище не запустилася автоматично. Після цього користувачеві буде запропоновано ряд пунктів:

відкрити проект, імпортувати з іншого середовища або створити новий. Так як попередніх робіт ще немає, а розробка додатків для Android тільки починається, слід клацнути по напису “Start a new project”. У різних версіях “Студії” написи можуть відрізнятися, тому бажано хоча б базове знання англійської мови.

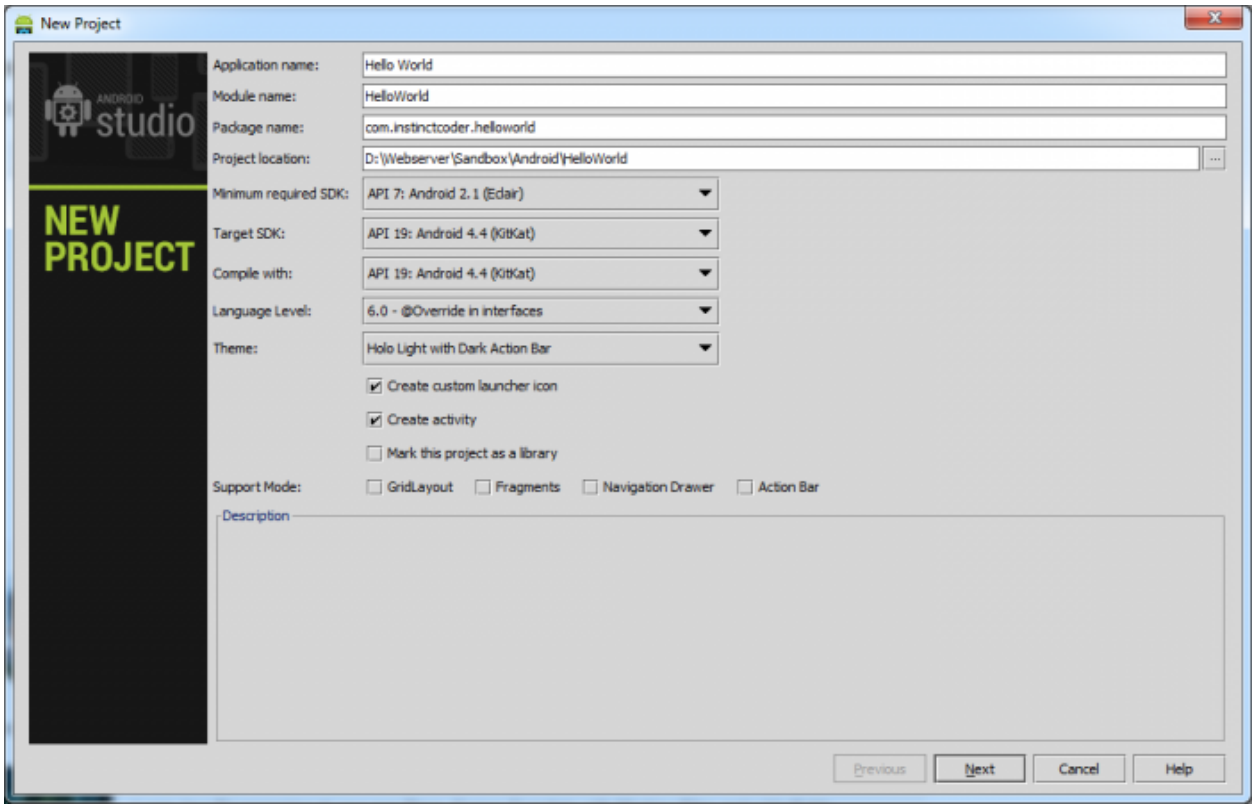


Наступне вікно програми запропонує зробити початкові налаштування:

1. Application name – ввести майбутнє ім’я програми.
2. Company Domain, або домен програми – розташування головного класу.
3. Project Location вкаже Android Studio, де повинні зберігатися файли проекту.
4. Після виконання конфігурації клацніть на кнопку “Next”.

Далі вам буде поставлено питання: яка мінімальна версія операційної системи буде використовуватися в процесі роботи? Тут можна залишити значення, встановлене Android Studio за замовчуванням.

Услід за цим необхідно вибрати “активіті”, тобто зовнішній вигляд інтерфейсу, і його розташування робочих елементів, так як створити додаток для Android максимально швидко без цього не вийде. Тут представлений значний список, який поповнюється від версії до версії, однак зараз варто зупинитися на найпростішому варіанті – Blank Activity.



Після вибору середовище розробки відобразить на екрані нове вікно налаштувань. У ньому буде кілька текстових полів:

1. Activity Name: ім’я класу.
2. Layout Name: ім’я файлу, в якому буде зберігатися розмітка інтерфейсу.
3. Title: назва головного вікна.
4. Resource Name: ім’я файлу для зберігання ресурсів вибраного “Activity”.

Клікнувши по кнопці “Готово”, ви завершите попередню конфігурацію, що призведе до відкриття головного вікна проекту.

2.5.1 Структура проекту

Як максимально складні й корисні програми для Android, так і найпростіші софти мають подібну структуру. Для її перегляду клацніть по назві проекту у верхній частині редактора. Зліва з’явиться список файлів і каталогів: AndroidManifest.xml – тут зберігаються опису фундаментальних характеристик програми і список всіх компонентів.

Директорія “java” включає в себе всі вихідні коди програми. Зараз в ній знаходиться тільки один файл – “MainActivity”, але навіть якщо б тут

була велика кількість документів, що саме цей все одно запускався б першим після “тапа” по іконці програми.

В каталозі “res” розташовані додаткові підпапки з ресурсами:

- “drawable” – включає в себе всі зображення, які використовуються в проекті;
- “layout” – зберігає файли графічного інтерфейсу. Зараз тут можна бачити єдиний документ “activity_main.xaml”, логічно, що і “актівіті” в додатку теж буде представлений в однині;
- у “menu” розташовані xml-документи, що визначають всі налаштування відображення користувальницьких меню;
- “mipmap” – зберігає зображення, на основі яких створюються іконки готової програми для різних розмірів екрану;
- в підкаталозі “values” також можна знайти xml-файли, але в них розташовані опису колекцій ресурсів.

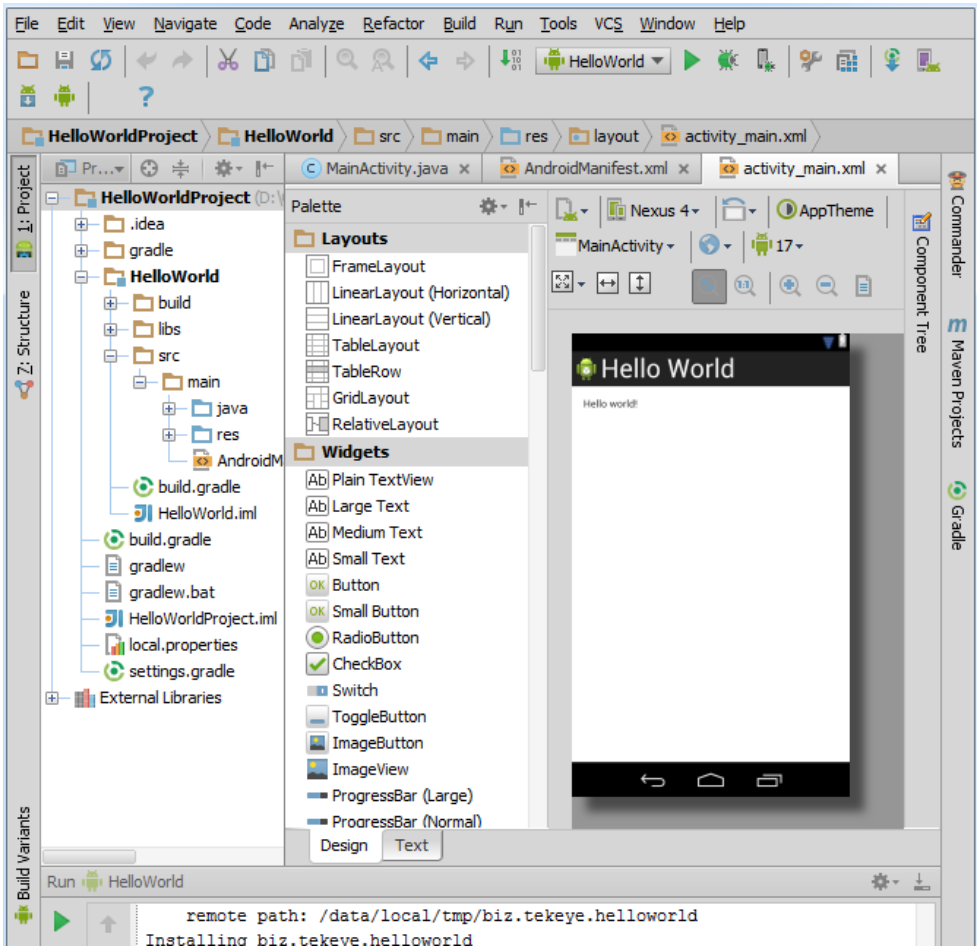
2.5.2 Створення програми

Розробка додатків для Android вже може бути розпочато, більш того, створений проект вже цілком реально запустити; ось тільки функціонал його практично нульовий – висновок на екран рядок “Hello, World”. Зараз в Android Studio відображається вміст документа “activity_main.xml”.

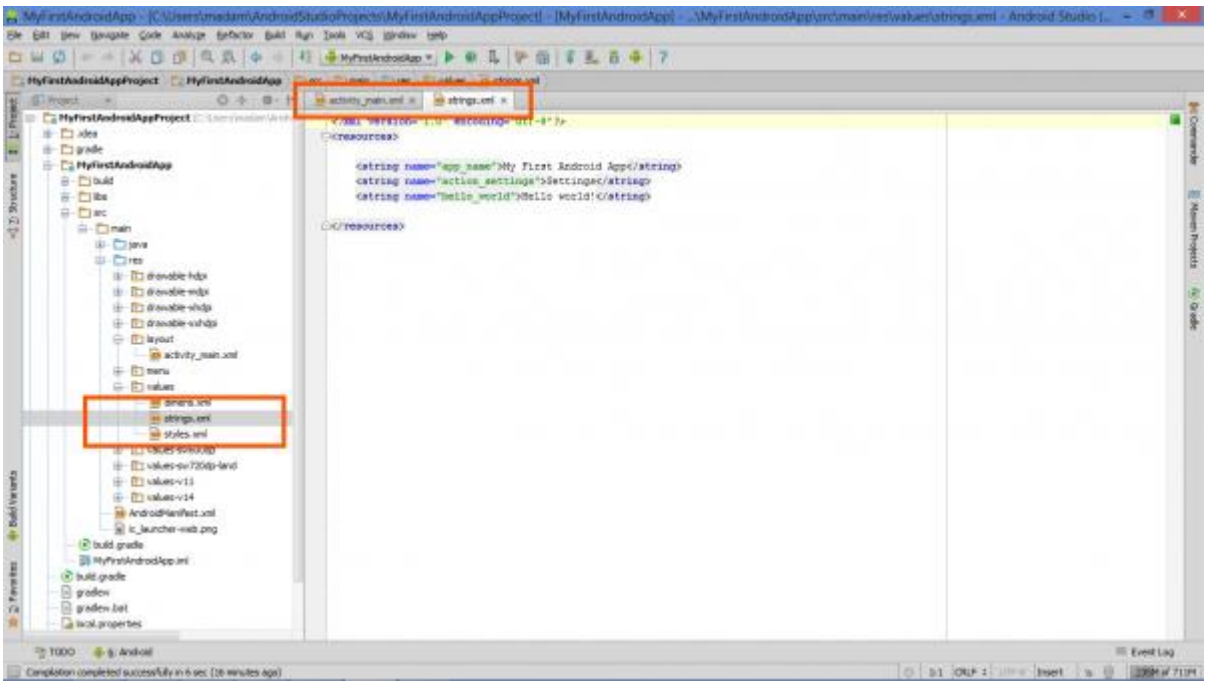
Також його налаштування показані на екрані віртуального телефону. Звичайно, орієнтуватися на це зображення можна тільки приблизно, але для отримання загального уявлення про майбутню програмі його цілком достатньо.

Щоб змінити класичний “Hello, World”, потрібно додати трохи текстового коду. Для цього клацніть по файлу “strings.xml”, розташованого в директорії “res/values”.

Відшукайте в ньому фрагмент “<stringname=”hello_world”>Hello world!” і змініть текст “Hello, world!” на будь-який інший.



Тепер збережіть проект і подивіться графічне вікно із зображенням смартфона. Рядок у ньому стане такою, яку ви визначили.



2.5.3 Запуск програми

Щоб запустити створений додаток, можна використовувати як емулятор, що поставляється разом з Android Studio, так і реальний телефон. У другому випадку в смартфоні потрібно поставити галочку біля пункту “Налагодження USB”. Він знаходиться у налаштуваннях розробника. Тестування програми на цьому смартфоні краще, емулятор не дасть стовідсоткової гарантії, що додаток запуститься поза комп’ютера.

Крім запуску, в процесі буде виконана установка додатків на Android, представлених у проекті. Зараз воно одне, тому нічого додаткового не встановиться.

Підключіть пристрій до ПК і запустити проект. Для цього слід натиснути на іконку із зображенням стрілки, яка розташовується у верхньому меню “студії”. Потім середовище розробки попросить вибрати, на якому саме пристрої слід тестувати проект.

Розробка першого додатка завершена, і якщо все зроблено вірно, на екрані телефону з’явиться програма з єдиною рядком тексту. Якщо ж під час запуску ви побачите повідомлення: “В додатку Android сталася помилка, спробуйте пройтися по всіх пунктах з самого початку. Якщо проблема виникне знову, змініть версію ОС на одному з перших кроків конфігурації проекту.

2.5.4 Вибір середовища розробки

Android Studio – досить громіздкий інструмент. Не всі звикли використовувати подібні рішення, тому намагаються знайти щось більш відповідне і зручне. До того ж з його допомогою можна створити щось серйозне без довгого вивчення азів програмування.

Перед тим як зробити вибір на користь якоїсь середовища розробки, потрібно ознайомитися з її описом на офіційному сайті і вивчити всі технічні можливості. Наприклад, деякі набори редакторів і компіляторів не дають широкого функціоналу, а здатні лише змінювати графічний інтерфейс. Відповідно, створити новий продукт за допомогою подібних утиліт проблематично.

Щоб вибрати хороший інструмент, в якому створення додатків для Android буде найбільш простим і продуктивним, потрібно оцінити його по наступним критеріям:

- прості і інтуїтивні елементи управління.
- зрозуміла і задокументована логіка роботи.

- можливість використання як графічного редагування, так і перегляду вихідного коду.
- велика документація або активна служба підтримки.

2.5.5 Запуск додатку з метою відлагодження на фізичному пристрої

Для запуску додатку на фізичному пристрої необхідно здійснити два кроки:

1. Підключити пристрій до комп'ютера за допомогою кабелю USB. Якщо ви розробляєте під управлінням ОС Windows, то вам може знадобитися встановити відповідний драйвер USB для вашого пристрою.

2. Увімкнути режим відлагодження через USB. Для здійснення другого кроку необхідно перейти на екран налаштувань параметрів розробника (Developer options screen).

На ОС Android 4.1 і новіших версій екран налаштувань параметрів розробника доступний за замовчуванням. На ОС Android 4.2 та новіших версій потрібно увімкнути цей екран. Щоб увімкнути параметри розробника, торкніться опції номер збірки (Build number) 7 разів.

Цю опцію можна знайти в одному з наступних місць, залежно від версії Android:

- Android 9 (API level 28) та вище: Settings>About Phone>Build Number
- Android 8.0.0 (API level 26) та Android 8.1.0 (API level 26): Settings>System > About Phone>Build Number;
- Android 7.1 (API level 25) та нижче: Settings>About Phone>Build Number.

У верхній частині екрану є перемикач для ввімкнення/вимкнення параметрів розробника (рис. 2.3). Щоб увімкнути режим відлагодження через USB, увімкніть відповідну опцію у меню параметрів розробника. Цю опцію можна знайти в одному з наступних місць, залежно від версії Android:

- Android 9 (API level 28) та вище: Settings>System>Advanced>Developer Options>USB debugging;
- Android 8.0.0 (API level 26) та Android 8.1.0 (API level 26): Settings>System>Developer Options>USB debugging;
- Android 7.1 (API level 25) та нижче: Settings>Developer Options>USB debugging.

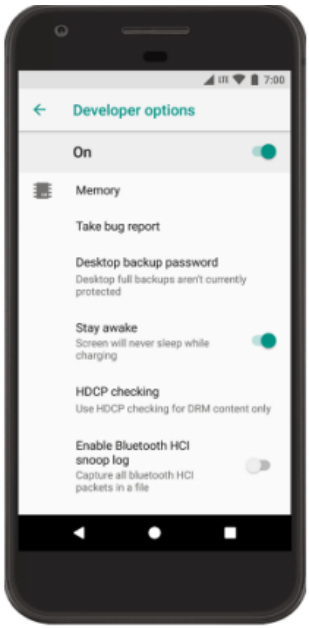


Рисунок 2.3 – Екран налаштувань параметрів розробника

2.6 Види Android-додатків

Виділяють такі основні види додатків під ОС Android:

1. ***Додатки переднього плану***, які виконують свої функції лише тоді, коли видимі на екрані, в іншому ж випадку їх виконання призупиняється. Такими додатками є, наприклад, ігри, текстові редактори, відеопрогравачі. При розробці цих програм потрібно дуже уважно вивчити життєвий цикл активності, щоб перемикання в фоновий режим і назад відбувалося плавно. Тобто, йдеться про коректне збереження стану додатку.

Ще одним важливим моментом, на який обов'язково треба звернути увагу при розробці додатків переднього плану є зручний і інтуїтивно зрозумілий інтерфейс для користувача.

2. ***Фонові додатки,*** які після налаштування не потребують взаємодії з користувачем а більшу частину часу перебувають і працюють в прихованому стані.

Прикладами таких додатків можуть бути служби екранування дзвінків, SMS-автовідповідачі. Здебільшого фонові програми націлені на відстеження подій, породжуваних апаратним забезпеченням, системою або іншими додатками. Можна створювати абсолютно невидимі сервіси, але тоді вони будуть некерованими. Мінімум дій, які необхідно дозволити користувачеві це: санкціонування запуску сервісу, налаштування,

припинення і переривання його роботи при необхідності.

2. ***Змішані додатки,*** які більшу частину часу працюють у фоновому режимі, проте допускають взаємодію з користувачем і після свого налаштування. Зазвичай взаємодія з користувачем зводиться до повідомлення про які-небудь події.

Прикладами таких додатків можуть бути мультимедіа-програвачі, додатки для обміну текстовими повідомленнями (чати), поштові клієнти. Можливість реагувати на введення користувачем інформації без втрати працездатності у фоновому режимі є характерною особливістю змішаних додатків. Такі програми зазвичай містять як видимі активності, так і приховані (фонові) сервіси, і при взаємодії з користувачем повинні враховувати свій поточний стан. Можливо їм буде необхідно оновлювати графічний інтерфейс, якщо додаток знаходиться на передньому плані, або ж посилати користувачеві повідомлення з фонового режиму, щоб тримати його в курсі того, що відбувається. Такі особливості необхідно враховувати при розробці подібних додатків.

3. **Віджети –** це невеликі додатки, які відображаються у вигляді графічного об’єкта на робочому столі. Прикладами можуть бути додатки для відображення динамічної інформації, такої як заряд батареї, прогноз погоди, дата і час.

Зрозуміло, що складні додатки можуть містити елементи кожного з розглянутих вище видів. Плануючи розробку конкретної програми необхідно визначити спосіб її використання і тільки після цього переходити до проектування і подальшої розробки

Питання для самоконтролю за темою

Для самостійної перевірки знань доцільно сформулювати розширені відповіді на поставлені питання і перевірити їх повноту та правильність за допомогою матеріалів запропонованих літературних джерел.

1. Що таке фонові додатки? Наведіть приклад.
2. Яким чином можна відлагоджити додаток на реальних пристроях?
3. Які файли зберігаються у проекті? Як вони організовані по директоріям?

ТЕМА 3 «АРХІТЕКТУРА ОС ANDROID»

План лекції.

Архітектура ОС Андроід.
Огляд рівнів.
Бібліотеки.
Взаємодія верхніх рівнів з розробником.

Пояснення до теми.

Розглянемо основні компоненти операційної системи Android:

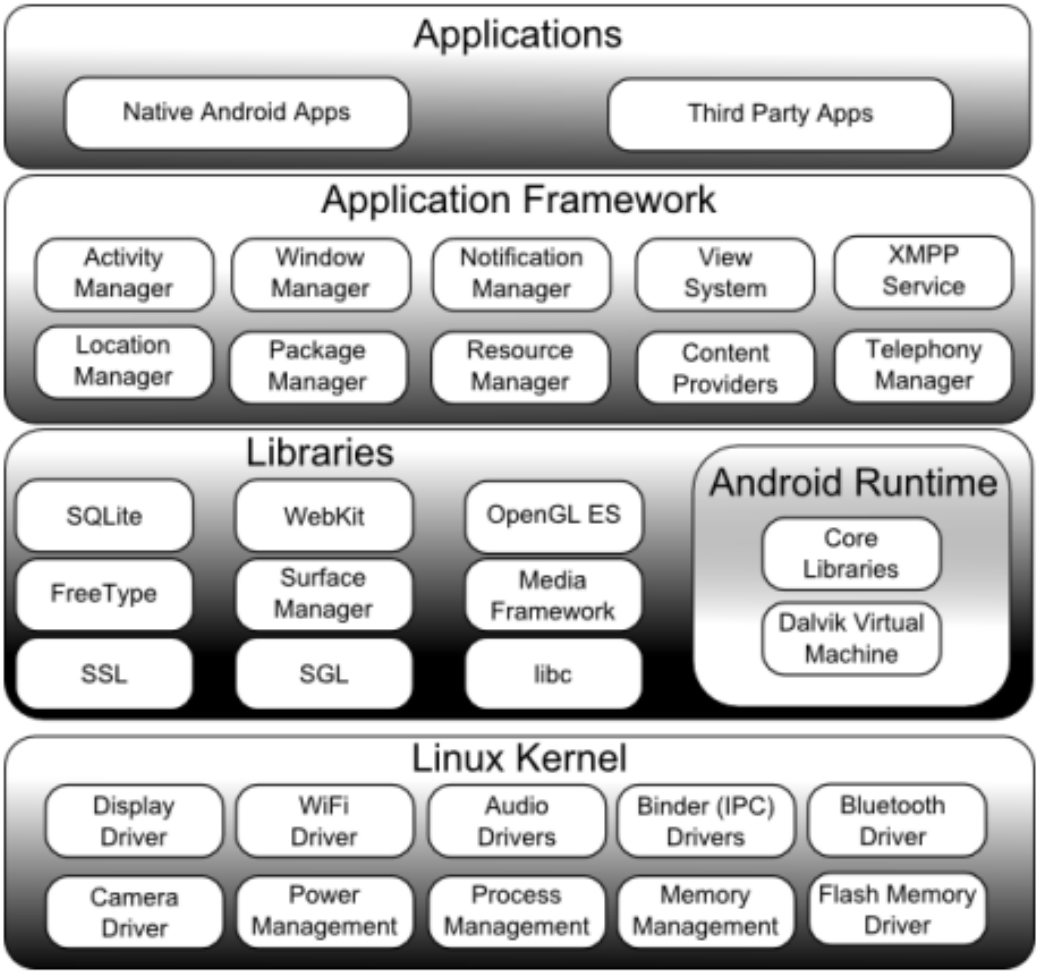


Рисунок 3.1 – Архітектура ОС Android

З точки зору архітектури, система Android представляє собою повний програмний стек, в якому можна виділити такі рівні:

- Рівень додатків (Applications) – набір встановленого прикладного програмного забезпечення;

- Рівень каркасу додатків (Application Framework) забезпечує розробникам доступ до API, наданих компонентами системи рівня бібліотек;
- Набір бібліотек і середовище виконання (Libraries&Android Runtime) забезпечує найважливіший базовий функціонал для додатків, містить віртуальну машину Dalvik і базові бібліотеки Java необхідні для запуску Android додатків;
- Базовий рівень (Linux Kernel) – рівень абстракції між апаратним рівнем і програмним стеком.

Розглянемо детальніше ці рівні.

Applications. Android постачається з набором основних додатків, що включає в себе календар, карти, браузер, менеджер контактів і ін. Всі перераховані програми написані на Java.

Application Framework. Будучи відкритою платформою, Android дає розробникам можливість створювати гнучкі та інноваційні програми. Розробники можуть використовувати апаратні можливості пристрою, отримувати інформацію про місцезнаходження, виконувати завдання у фоновому режимі, встановлювати оповіщення та багато іншого. Розробники мають повний доступ до тих же API, що використовуються в основних додатках.

Архітектура додатків розроблена з метою спрощення повторного використання компонентів; будь-який додаток може «публікувати» свої можливості і будь-яке інший додаток може потім використовувати ці можливості (з урахуванням обмежень безпеки).

Цей же механізм дозволяє замінювати стандартні компоненти на призначені користувацькі.

Libraries. Android включає в себе набір C/C++ бібліотек, які використовуютьсярізними компонентами системи. Ці можливості доступні розробникам в контексті застосування Android Aplication Framework. Деякі основні бібліотеки, перераховані нижче:

- Медіа бібліотеки – ці бібліотеки надають підтримку відтворення і запису багатьох популярних аудіо, відео форматів і форматів зображень, в тому числі MPEG4, MP3, AAC, AMR, JPG, PNG і інших;
- Surface Manager – управляє доступом до підсистеми відображення 2D і 3D графічних шарів;
- LibWebCore – сучасне веб-ядро, на якому побудований браузер

Android;

- SGL – основне графічне ядро 2D;
- 3D бібліотеки – реалізовані на основі OpenGL; бібліотеки використовують або апаратне 3D-прискорення (при його наявності), або вмикаються програмно;
- FreeType – підтримка растрових і векторних шрифтів;
- SQLite – механізм бази даних, доступний для всіх додатків.

Android Runtime. Android включає в себе набір основних бібліотек, які забезпечують більшість функцій, доступних в бібліотеках Java. Кожна програма Android працює в своєму власному процесі, зі своїм власним екземпляром віртуальної машини Dalvik.

Dalvik була написана так, що пристрій може працювати ефективно з декількома віртуальними машинами одночасно. Dalvik проектувалася спеціально під платформу Android. Віртуальна машина оптимізована для низького споживання пам’яті та роботи на мобільному апаратному забезпеченні.

Dalvik використовує власний байт-код. Android-додатки перекладаються компілятором в спеціальний машинно-незалежний низькорівневий код. І саме Dalvik інтерпретує і виконує таку програму при виконанні на платформі. Крім того, за допомогою спеціальної утиліти, що входить до складу Android SDK, Dalvik здатна перекладати байт-коди Java в коди власного формату і також виконувати їх у своїй віртуальному середовищі.

Linux Kernel. Android базується на Linux 2.6 з основними системними службами – безпека, управління пам’яттю, управління процесами і модель драйверів. Розробник зазвичай взаємодіє з двома верхніми рівнями архітектури Android для створення нових додатків. Бібліотеки, система виконання і ядро Linux приховані за каркасом додатків та абстракцій. Для того, щоб встановити програму на пристроях з ОС Android створюється файл з розширенням *.apk (Android package), який містить виконувані файли, а також допоміжні компоненти, наприклад, файли з даними і файли ресурсів. Після установки на пристрій кожен додаток «живе» в своєму власному ізольованому екземплярі віртуальної машини Dalvik.

Питання для самоконтролю за темою

Для самостійної перевірки знань доцільно сформулювати розширені відповіді на поставлені питання і перевірити їх повноту та правильність за допомогою матеріалів запропонованих літературних джерел.

1. Що забезпечує Application Framework?
2. Що включає в себе Android Runtime?
3. Що забезпечує набір бібліотек і середовище виконання?

ТЕМА 4 «ФАЙЛ МАНІФЕСТУ. РОБОТА З ACTIVITY»

План лекції.

Файл маніфесту.

Огляд компонентів додатку.

Поняття Актівіті.

Управляння Актівіті.

Пояснення до теми.

4.1 Файл маніфесту AndroidManifest.xml

Кожна програма містить файл маніфесту AndroidManifest.xml. Файл AndroidManifest.xml містить інформацію про ваш пакет, включаючи такі компоненти програми, як дії, служби, приймачі трансляції, постачальники вмісту тощо.

Він також виконує деякі інші завдання:

- він відповідає за захист програми від доступу до будь-яких захищених частин, надаючи дозволи.
- він також оголошує android api, який програма збирається використовувати.
- у ньому перераховані класи приладів. класи приладів надають профільну та іншу інформацію. Ця інформація видаляється безпосередньо перед публікацією програми тощо.
- це необхідний xml-файл для всіх програм android і розташований у кореневому каталозі.

Файл маніфесту може виглядати так:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.eugene.viewsapplication">
 <application
 android:allowBackup="true"
 android:icon="@mipmap/ic_launcher"
 android:label="@string/app_name"
 android:supportsRtl="true"
 android:theme="@style/AppTheme">
 <activity android:name=".MainActivity">
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
```

```
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
 </application>
</manifest>
```

Кореневим елементом є вузол manifest. В даному випадку тільки визначається пакет додатку – *package="com.example.eugene.viewsapplication".*

Більшість налаштувань рівня додатку визначаються елементом application. Наприклад, через атрибут *android:icon="@mipmap/ic_launcher"* задається іконка програми, яка знаходиться в каталозі *res /mipmap-xxxx.*

Також тут задається назва додатку, яка буде відображатися на мобільному пристрої в списку додатків і в заголовку: android:label="@string/app_name". В даному випадку вона зберігається в стрічковому ресурсі.

Вкладені елементи activity визначають всі використовувані в додатку activity. В даному випадку видно, що в додатку є тільки одна activity – MainActivity.

Елемент intent-filter в MainActivity вказує, як дана activity буде використовуватися. Зокрема, за допомогою вузла action android:name="android.intent.action.MAIN" вказано, що дана activity буде вхідною точкою в додаток і не повинна отримувати будь-які дані ззовні.

Елемент category android:name="android.intent.category.LAUNCHER" вказує, що MainActivity представлятиме стартовий екран, який відображається під час запуску програми.

4.2 Компоненти Android-додатку

Кожен додаток під управлінням ОС Android запускається в своєму власному процесі. Тому додаток ізольовано від інших запущених додатків, і некоректно працюючий додаток не може нашкодити іншим запущеним на виконання додаткам.

Проте, важливою властивістю додатку є можливість використовувати компоненти інших додатків, якщо вони дають на це відповідні права. Припустимо, нам потрібен якийсь компонент з прокруткою для відображення тексту, і схожий компонент вже реалізований в іншому додатку. Тоді у нас є можливість використовувати реалізований компонент. У цьому випадку наш додаток не копіює необхідний код до себе і не створює

посилання на нього. Замість цього додаток робить запит на виконання частини коду іншої програми, де є потрібний нам компонент.

Виділяють чотири типи компонентів Android-програми*: Activities, Services, Broadcast receivers і Content providers.*

Також важливо відзначити об'єкти Intents, в Android-додатках майже все працює завдяки їм. **Intent** - це механізм для опису однієї операції (вибрати фотографію, відправити лист, зробити дзвінок, запустити браузер і перейти за вказаною адресою тощо). Найбільш поширений сценарій використання Intent – запуск іншого Activity в додатку.

4.1.1 Services

Service – це певний процес, який запускається у фоновому режимі. Наприклад, Service може отримувати дані по мережі, виконувати будь-які тривалі обчислення.

Хорошим прикладом Service служить програвач музики. Користувач може вибрати будь-яку пісню в програвачі, включити її і закрити плеєр. Музика буде програватися в фоновому процесі. Service для програвання музики буде працювати, навіть якщо Activity плеєра буде закритою.

Подібно до Activity, Service має свої методи життєвого циклу:

- *void onCreate();*
- *void onStart(Intent intent);*
- *void onDestroy().*

У повному життєвому циклі Service існує два вкладених цикли:

1. Повне життя Service – це проміжок між моментом виклику методу onCreate() і моментом повернення onDestroy(). Подібно до Activity, для Services здійснюють початкову ініціалізацію в onCreate() і звільняють всі зайняті ним ресурси в onDestroy();

2. Активне життя Service – починається з виклику методу onStart(). Цьому методу передається об'єкт Intent, який передавався в startService().

Як і Activities, Services запускаються в головному потоці процесу додатку. Томуїх слід запускати в окремому потоці, щоб вони не блокували ні інші компоненти ні інтерфейс користувача.

4.1.2 Broadcast receivers

Broadcast receiver – це компонент, який розсилає і реагує на широкомовні повідомлення. Прикладом широкомовних компонентів

можуть бути: повідомлення про перехід на літній/зимовий час, повідомлення про мінімальний заряді батареї і т. д.

Broadcast receiver не відображує на екрані користувацький інтерфейс, але може запустити Activity при отриманні певного повідомлення або використовувати NotificationManager для залучення уваги користувача. Привернути увагу користувача можна, наприклад, вібрацією пристрою, програванням звуку або миготінням спалаху.

Приймач широкомовних повідомлень має єдиний метод життєвого циклу: *onReceive()*. Коли широкомовне повідомлення прибуває до одержувача, Android викликає його методом *onReceive()* і передає в нього об'єкт Intent, що містить саме повідомлення. Приймач широкомовних повідомлень є активним тільки під час виконання цього методу. Процес, який в даний час виконує Broadcast receiver, є пріоритетним процесом і буде збереженим, окрім випадків гострої нестачі пам'яті в системі.

Коли програма повертається з *onReceive()*, приймач стає неактивним і система вважає, що робота об'єкта Broadcast receiver закінчена. Процес з активним широкомовним одержувачем захищений від знищення системою. Однак процес, що містить неактивні компоненти, може бути знищений системою в будь-який момент, якщо пам'ять, яку він використовує, буде необхідна для інших процесів.

4.1.3 Content providers

Content providers надають доступ до даних (читання, додавання, оновлення). Content provider може надавати доступ до даних не тільки свого додатку, а й інших.

Дані можуть розміщуватися в файловій системі чи в базі даних.

4.2 Activity

4.2.1 Поняття Activity

Ключовим компонентом створення візуального інтерфейсу в додатку є activity (активність). Часто activity асоціюється з окремим екраном або вікном додатку, а перемикання між вікнами буде відбуватися як переміщення від однієї activity до іншої.

Додаток може включати одну або декілька activity (рис. 4.1).

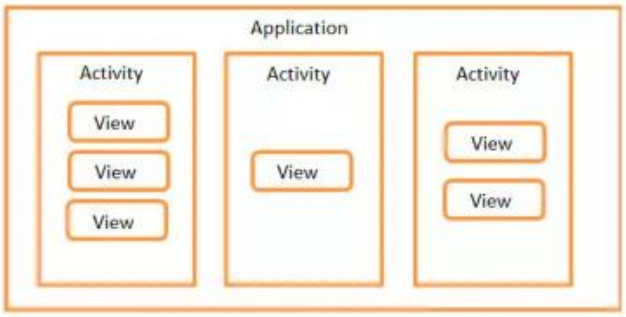


Рисунок 4.1 – Представлення додатку як набору activity

Всі об’єкти activity є об’єктами класу android.app.Activity, який містить базову функціональність для всіх activity. Як правило activity успадковуються від класу AppCompatActivity який, успадковує базовий клас Activity:

```
public class MainActivity extends AppCompatActivity {
 // вміст класу
}
```

4.2.2 Життєвий цикл Activity

Всі додатки Android мають строго визначений життєвий цикл. При запуску користувачем додатку операційна система надає цьому додатку високий пріоритет.

Кожна програма запускається у вигляді окремого процесу, що дозволяє системі надавати одним процесам вищий пріоритет, на відміну від інших. Завдяки цьому, наприклад, при роботі з одними додатками не блокуються вхідні дзвінки. Після припинення роботи додатку, система звільняє всі пов’язані з ним ресурси, переводить додаток у розряд низькопріоритетного і закриває його.

Всі об’єкти activity, які є в додатку, управляються системою у вигляді стека activity, який називається back stack. При запуску нової activity вона поміщається на вершину стеку і виводиться на екран пристрою, поки не з’явиться нова activity. Коли поточна activity закінчує свою роботу (наприклад, користувач закриває програму), то вона видаляється із стеку, і відновлює роботу та activity, яка раніше була другою в стеці.

Після запуску activity проходить через ряд подій, які обробляються системою і для обробки яких існує ряд зворотних викликів:

- protected void onCreate(Bundle saveInstanceState);
- protected void onStart();

- protected void onRestoreInstanceState(Bundle saveInstanceState);
- protected void onRestart();
- protected void onResume();
- protected void onPause();
- protected void onSaveInstanceState(Bundle saveInstanceState);
- protected void onStop();
- protected void onDestroy();

Схематично взаємозв'язок між усіма цими зворотними викликами можна представити так, як це зроблено на рис 4.2.

Тут ***onCreate()*** – перший метод, з якого починається виконання activity. Це перший метод, який викликається, коли ми запускаємо дію з головного екрана або наміру. Іншими словами, це стандартний зворотний виклик, який автоматично створюється, коли ви створюєте нову дію.

Це єдиний метод, потрібний розробникам для реалізації логіки діяльності, яка має відбуватися лише один раз, наприклад, ініціалізація ViewModel.

Android Studio автоматично створює клас під назвою файл MainActivity.java. Цей клас містить зворотний виклик onCreate(). Він викликається, коли користувач вперше відкриває програму.

Коли програма встановлена на пристрої, вона перебуває в стані «не існує». Це означає, що діяльність мертва.

Як тільки користувач відкриває програму, починається життєвий цикл. Діяльність виводиться на перший план. У цьому випадку негайно викликається onCreate(), щоб запустити програму. Він може містити такі компоненти, як інтерфейс користувача активності.

В методі ***onStart()*** відбувається підготовка до виведення activity на екран пристрою. Як правило, цей метод не вимагає перевизначення, а всю роботу робить вбудований код. Після завершення роботи методу activity відображається на екрані, викликається метод ***onResume()***, а activity переходить в стан Resumed.

Після завершення методу ***onStart()*** викликається метод ***onRestoreInstanceState(),*** який призначений для відновлення збереженого стану з об'єкта Bundle, який передається в якості параметра. Але слід враховувати, що цей метод викликається тільки тоді, коли Bundle не

дорівнює null і містить раніше збережений стан. Так, при першому запуску програми цей об’єкт Bundle матиме значення null, тому і метод ***onRestoreInstanceState()*** не викликатиметься.

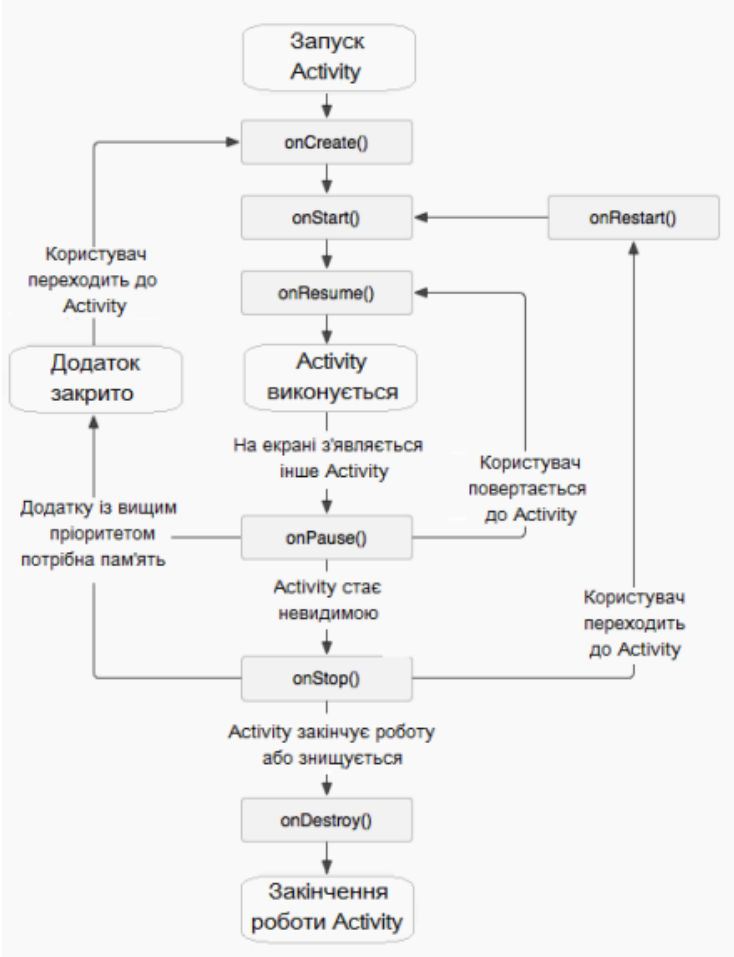


Рисунок 4.2 – Життєвий цикл activity

При виклику методу ***onResume()*** activity переходить в стан Resumed, а користувач може з нею взаємодіяти. І власне activity залишається в цьому стані доти, поки вона не втратить фокус, наприклад, внаслідок перемикання на іншу activity або просто через вимкнення екрану пристрою.

Якщо користувач вирішить перейти до іншої activity, то система викликає метод ***onPause().*** У цьому методі можна звільняти використовувані ресурси, припиняти процеси, наприклад, відтворення аудіо, анімацій, зупиняти роботу камери (якщо вона використовується) і т.д., для того, щоб вони менше позначалися на продуктивності системи. Але треба враховувати, що на роботу даного методу відводиться дуже мало часу, тому не варто тут зберігати якісь дані, особливо якщо при цьому потрібно

звернення до мережі, наприклад, відправка даних по інтернету, або звернення до бази даних.

Після виконання цього методу activity стає невидимою, не відображається на екрані, але вона все ще активна. І якщо користувач вирішить повернутися до цієї activity, то система викличе знову метод ***onResume(),*** і activity знову з'явиться на екрані. Інший варіант роботи може виникнути, якщо раптом система бачить, що для роботи активних додатків необхідно більше пам'яті. І система може сама завершити повністю роботу activity, яка невидима і знаходиться в фоні. Або користувач може натиснути на кнопку Back (Назад).

В цьому випадку у activity викликається метод ***onStop().*** Метод ***onSaveInstanceState()*** викликається після методу onPause(), але до виклику ***onStop().*** У ***onSaveInstanceState*** проводиться збереження стану програми в переданий в якості параметра об'єкт Bundle. В методі onStop() activity переходить в стан Stopped. Тут слід звільнити використовувані ресурси, які не потрібні користувачеві, коли він не взаємодіє з activity. Тут також можна зберігати дані, наприклад, в базу даних. При цьому в стані Stopped activity залишається в пам'яті пристрою, зберігається стан всіх елементів інтерфейсу.

Наприклад, якщо в текстове поле EditText був введений якийсь текст, то після відновлення роботи activity і переходу її в стан Resumed ми знову побачимо в текстовому полі раніше введений текст. Якщо після виклику методу ***onStop()*** користувач вирішить повернутися до колишньої activity, тоді система викличе метод ***onRestart()***. Якщо ж activity повністю завершила свою роботу, наприклад, внаслідок закриття програми, то викликається метод ***onDestroy().***

Завершується робота activity викликом методу ***onDestroy()***, який відбувається або якщо система вирішить завершити роботу activity, або при виклику методу ***finish()***.

Також слід зазначити, що при зміні орієнтації екрану система завершує activity і потім створює її заново, викликаючи метод ***onCreate()***. В цілому перехід між станами activity можна представити наступною схемою (рис 4.3):

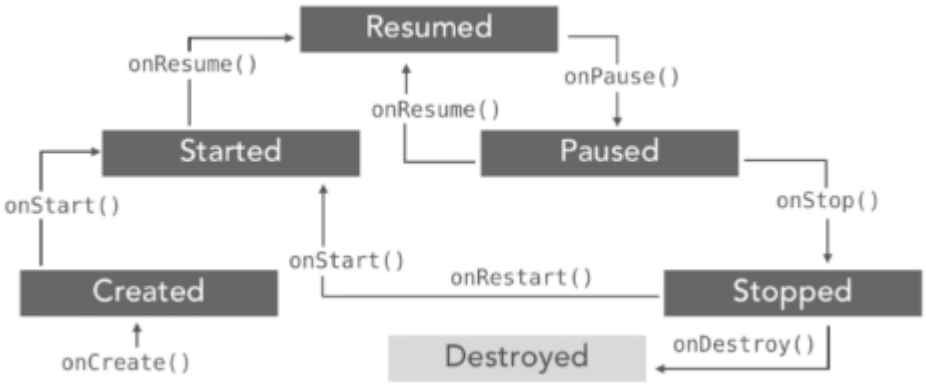


Рисунок 4.3 – Діаграма переходів між станами Activity

Розглянемо декілька типових ситуацій. Якщо ми працюємо з Activity і потім перемикається на іншу програму, або натискаємо на кнопку Home, то у Activityвикликається наступний ланцюжок методів: *onPause()->onStop()*. Activity переходить в стан Stopped.

Якщо користувач вирішить повернутися до Activity, то викликається наступний ланцюжок методів*: onRestart()->onStart()->onResume()*.

Інша типова ситуація, коли користувач натискає на кнопку Back (Назад), то викликається наступний ланцюжок *onPause()->onStop()->onDestroy()*. В результаті Activity знищується. Якщо ми раптом захочемо повернутися до Activity через диспетчер задач або заново відкривши додаток, то activity буде заново перестворенопшляхом виклику методів *onCreate()->onStart()-> onResume()*.

4.2.3 Управління життєвим циклом Activity

Ми можемо керувати подіями життєвого циклу activity, перевизначивши відповідні методи. Наприклад:

```
public class MainActivity extends AppCompatActivity {
 private final static String TAG="MainActivity";

@Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 Log.d(TAG, "onCreate");
 }

 @Override
 protected void onDestroy(){
```

```
 super.onDestroy();
 Log.d(TAG, "onDestroy");
 }
 @Override
 protected void onStop(){
 super.onStop();
 Log.d(TAG, "onStop");
 }

@Override
 protected void onStart(){
 super.onStart();
 Log.d(TAG, "onStart");
 }

@Override
 protected void onPause(){
 super.onPause();
 Log.d(TAG, "onPause");
 }

@Override
 protected void onResume(){
 super.onResume();
 Log.d(TAG, "onResume");
 }

 @Override
protected void onRestart(){
 super.onRestart();
 Log.d(TAG, "onRestart");
 }

@Override
 protected void onSaveInstanceState(Bundle outState){
 super.onSaveInstanceState(outState);
 Log.d(TAG, "onSaveInstanceState");
 }

@Override
 protected void onRestoreInstanceState(Bundle
savedInstanceState){
 super.onRestoreInstanceState(savedInstanceState);
 Log.d(TAG, "onRestoreInstanceState");
 }
}
```

Для логування подій тут використовується клас **android.util.Log.**

В даному випадку обробляються всі ключові методи життєвого циклу. Обробка зводиться до виклику методу *Log.d()*, в який передається TAG –

випадкове значення рядка і рядок, який виводиться в консолі logcat внизу середовища Android Studio у вікні Android Monitor. Якщо ця консоль за замовчуванням прихована, то її можна відкрити через пункт меню View->Tool Windows->Android Monitor. Під час запуску програми ми зможемо побачити у вікні logcat відлагоджувальну інформацію (рис. 4.4).

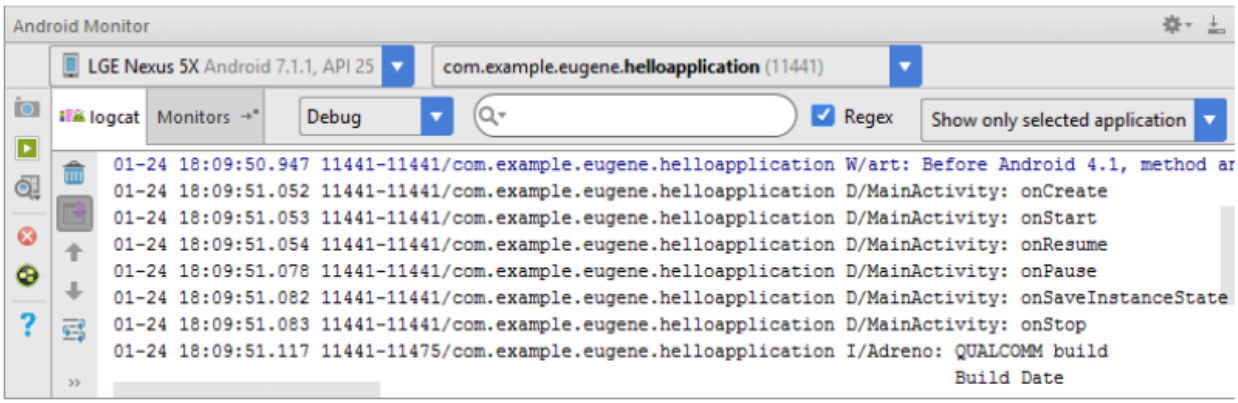


Рисунок 4.4 – Відлагоджувальна інормація у вікні logcat середовища Android Studio

4.2.4 Запуск Activity

При організації взаємодії між різними об’єктами activity ключовим класом є android.content.Intent. Він представляє собою задачу, яку повинен виконати додаток.

Нехай ми добавили в проект нову пусту activity з ім’ям SecondActivity. Після цього в файлі маніфесту AndroidManifest.xml ми зможемо знайти відповідний рядок:

```
        <activity
         android:name=".MainActivity">
         <intent-filter>
         <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER"
/>
         </intent-filter>
        </activity>
        <activity android:name=".SecondActivity"></activity>
```

Тут для MainActivity в елементі intent-filter визначено інтент-фільтр, в якому елемент <action android:name="android.intent.action.MAIN" представляє головну точку входу в додаток. Тобто MainActivity залишається основною і запускається додатком за замовчуванням.

Для SecondActivity просто вказано, що вона є в проекті, і ніяких intentфільтрів для неї не задано.

Для того, щоб з MainActivity запустити SecondActivity, слід викликати метод startActivity():

```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```

Як параметр в метод startActivity() передається об’єкт Intent, який приймає два параметри: action (дію або завдання) і data (передані в задачу дані).

В якості параметра action може виступати багато можливих дій. В даному випадку використовується дія ACTION_MAIN, якf задається константою "android.intent.action.MAIN".

Тепер розглянемо реалізацію переходу від однієї Activity до іншої. Для цього в файлі activity_main.xml (тобто в інтерфейсі для MainActivity) визначимо кнопку:

```
<Button
 android:id="@+id/navButton"
 android:textSize="20sp"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Перейти до SecondActivity"
android:onClick="onClick" />
```

Для запуску SecondActivity із MainActivity при натисканні на цю кнопку необхідно написати наступний код:

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

 public void onClick(View view) {
 Intent intent = new Intent(this, SecondActivity.class);
 startActivity(intent);
 }
}
```

4.3 Передача даних між Activity

Для передавання даних між двома Activity використовується об’єкт Intent. Через його метод putExtra() можна передати ключ і пов’язане з ним значення.

Наприклад, передача з поточної activity в SecondActivity стрічки "Hello World" з ключем "hello" може виглядати так:

```
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("hello", "Hello World");
startActivity(intent);
```

Для передавання даних застосовується метод putExtra(), який дозволяє передати дані найпростіших типів – string, int, float, double, long, short, byte, char, масиви цих типів, або об’єкт інтерфейсу Serializable.

Щоб отримати відправлені дані при завантаженні SecondActivity, можна скористатися методом get(), в який передається ключ об’єкта:

```
Bundle arguments = getIntent().getExtras();
String name = arguments.get("hello").toString();
```

Залежно від типу даних, що передаються, при отриманні ми можемо використовувати ряд методів об’єкта Bundle. Всі вони в якості параметра приймають ключ об’єкта.

Основні з них:

- get(): універсальний метод, який повертає значення типу Object. Відповідно поле отримання дане значення необхідно перетворити до потрібного типу;
- getString(): повертає об’єкт типу string;
- getInt(): повертає значення типу int;
- getByte(): повертає значення типу byte;
- getChar(): повертає значення типу char;
- getShort(): повертає значення типу short;
- getLong(): повертає значення типу long;
- getFloat(): повертає значення типу float;
- getDouble(): повертає значення типу double;
- getBoolean(): повертає значення типу boolean;
- getCharArray(): повертає масив об’єктів char;
- getIntArray(): повертає масив об’єктів int;

- getFloatArray(): повертає масив об’єктів float;
- getSerializable(): повертає об’єкт інтерфейсу Serializable.

Для передачі даних складних типів використовується механізм серіалізації.

Нехай в проекті є клас Product, який реалізує інтерфейс Serializable:

```
public class Product implements Serializable {
 private String name;
 private String company;
 private int price;
 public Product(String name, String company, int price){
 this.name = name;
 this.company = company;
 this.price = price;
 }
…
}
```

Щоб передати його в іншу activity слід написати код:

```
Product product=new Product(name, company, price);
Intent intent=new Intent(this, SecondActivity.class);
intent.putExtra(Product.class.getSimpleName(), product);
startActivity(intent);
```

В якості ключа тут використовується результат виклику методу Product.class.getSimpleName(), який по суті повертає назву класу. Щоб отримати об’єкт типу Product слід написати:

```
public class SecondActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 TextView textView = new TextView(this);
 textView.setTextSize(20);
 textView.setPadding(16, 16, 16, 16);
 Bundle arguments = getIntent().getExtras();
 final Product product;
 if(arguments !=null){
 product = (Product)
arguments.getSerializable(Product.class.getSimpleName());
 textView.setText("Name:    "    +    product.getName()    +
"\nCompany: " +
product.getCompany() +
 "\nPrice: " + String.valueOf(product.getPrice()));
 }
```

```
 setContentView(textView);
 }
}
```

Питання для самоконтролю за темою

Для самостійної перевірки знань доцільно сформулювати розширені відповіді на поставлені питання і перевірити їх повноту та правильність за допомогою матеріалів запропонованих літературних джерел.

1. Дайте визначення Інтенту.
2. Для чого потрібні Content providers?
3. З яких епатів складається життєвий цикл активіті?
4. Перелічите основні методи роботи з активіті.
5. Як виконується запуск Активіті?

ТЕМА 5 «ОСНОВИ ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ ПРОГРАМИ»

План лекції.

Графічний інтерфейс користувача.

Визначення інтерфейсу у файлі xml.

Варіанти компонування елементів інтерфейсу

Пояснення до теми.

5.1 Компоненти екрану

Графічний інтерфейс користувача є ієрархією об’єктів android.view.View і android.view.ViewGroup. Кожен об’єкт ViewGroup представляє собоє контейнер, який містить і впорядковує дочірні об’єкти View. Зокрема, до контейнерів відносять такі елементи, як RelativeLayout, LinearLayout, GridLayout, ConstraintLayout і ряд інших.

Прості об’єкти View є елементами управління та інші віджети, наприклад, кнопки, текстові поля і т. д., через які користувач взаємодіє з програмою.Більшість візуальних елементів успадковуються від класу View, такі як кнопки, текстові поля та інші, і розташовуються в пакеті android.widget.

Розмітка визначає візуальну структуру користувацького інтерфейсу. Встановити розмітку можна декількома способами:

1. Створити елементи управління програмно в коді Java;
2. Оголосити елементи інтерфейсу в XML
3. Поєднання обох способів – базові елементи розмітки визначити в XML, а решта додавати під час виконання додатку програмним способом.

Найкращим є підхід, за якого візуальний інтерфейс описується в файлах xml, а вся пов’язана з ним логіка – в класі activity. Таким чином ми досягаємо розмежування інтерфейсу і логіки додатка, їх легше розробляти і модифікувати.

5.2 Визначення інтерфейсу у файлі xml

Файли layout: у додатках під Android візуальний інтерфейс часто завантажується із спеціальних файлів xml, які зберігають розмітку. Ці файли

є ресурсами розмітки. Подібний підхід нагадує створення веб-сайтів, коли інтерфейс описується в файлах html, а логіка програми – в коді javascript.

Файли розмітки графічного інтерфейсу розташовуються в проекті в каталозі *res/layout*. При створенні розмітки в XML слід дотримуватися деяких правил: кожен файл розмітки повинен містити один кореневий елемент, який повинен представляти об'єкт **View або ViewGroup**.

При компіляції кожен XML-файл розмітки компілюється в ресурс View. Завантаження ресурсу розмітки здійснюється в методі *Activity.onCreate()*. Щоб встановити розмітку для поточного об'єкта activity, треба в метод *setContentView()* як параметр передати посилання на ресурс розмітки.

Для отримання посилання на ресурс в коді Java необхідно застосувати вираз R.layout.[Назва_ресурсу]. Назва ресурсу layout повинна збігатися з ім'ям файлу.

Наприклад:

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

Зазвичай в проекті є декілька різних ресурсів layout (рис. 5.1). Як правило, кожен окремий клас Activity використовує свій файл layout. Але для одного класу Activity можна також використовуватися декілька різних файлів layout.

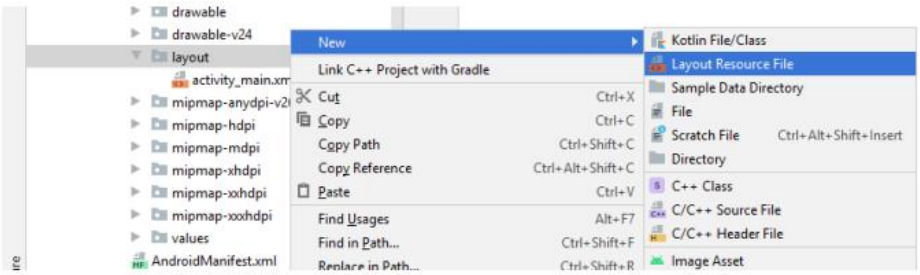


Рисунок 5.1 – Додавання нового файла layout в проект

Для доступу до елементів по атрибуту id клас Activity має метод *findViewById()*. У цей метод передається ідентифікатор ресурсу у вигляді R.id.[ідентифікатор_елементу]. Цей метод повертає об'єкт View – об'єкт базового класу для всіх елементів, тому результат методу ще необхідно привести до відповідного типу.

Далі ми можемо маніпулювати цим елементом.

Наприклад, для текстового поля:

```
<TextView android:id="@+id/header"
 android:text="Second Activity"
 android:textSize="26dp"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content" />
```

можна змінити текст:

```
public class MainActivity extends AppCompatActivity {
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);

 // встановлюємо в якості інтерфейсу файл second_layout.xml
 setContentView(R.layout.second_layout);

 // отримуємо елемент textView
 TextView textView = (TextView) findViewById(R.id.header);
 // змінюємо текст
 textView.setText("Hello Android!");
 }
}
```

Тут важливо, що пошук елемента відбувається після того, як в методі setContentView() була встановлена розмітка, в якій цей візуальний елемент був визначений.

5.3 Графічні можливості Android Studio

Android Studio має потужний інструментарій, який полегшує розробку графічного інтерфейсу (рис. 5.2).

Ми можемо відкрити файл activity_main.xml і внизу за допомогою кнопки Design переключитися в режим дизайнера із графічним представленням інтерфейсу у вигляді ескізу смартфона.

Зліва буде знаходитися панель інструментів, із якої ми можемо переносити потрібний елемент мишкою на ескіз смартфона. При цьому всі перенесені елементи будуть автоматично додаватися в файл activity_main.xml. За допомогою миші ми можемо змінювати позиціонування вже доданих елементів.

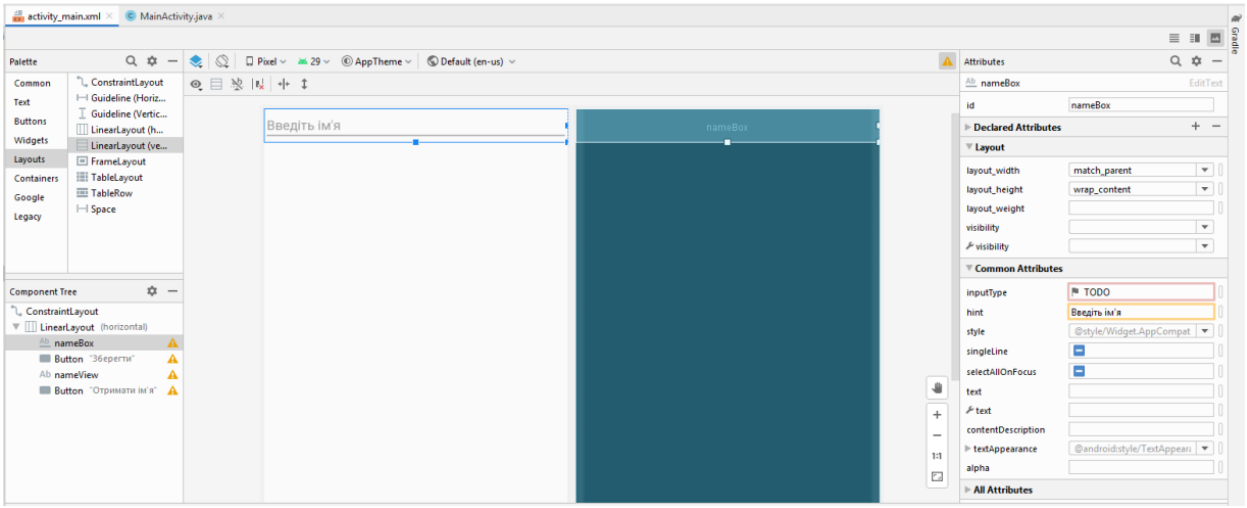


Рисунок 5.2 – Частина Android Studio, призначена для розробки інтерфейсу програми

Справа буде вікно Properties – панель властивостей виділеного елемента. Тут ми можемо змінити значення властивостей елемента. І знову ж таки після зміни властивостей зміниться і вміст файлу activity_main.xml.

Тобто при будь-яких змінах в режимі дизайнера відбуватиметься синхронізація з айлом activity_main.xml. Це все одно, що ми вручну змінювали б код безпосередньо в файлі activity_main.xml.

5.4 Різні варіанти компонування елементів інтерфейсу (Layout)

5.4.1 LinearLayout

Контейнер LinearLayout представляє об’єкт ViewGroup, який впорядковує всі дочірні елементи в одному напрямку: по горизонталі або по вертикалі. Все елементирозташовуються один за одним. Напрямок розмітки вказується за допомогою атрибута *android:orientation*

Якщо, наприклад, орієнтація розмітки вертикальна (android:orientation = "vertical"), то всі елементи розташовуються в стовпчик – по одному елементу на кожен рядок.

Якщо орієнтація горизонтальна (android:orientation="horizontal"), то елементи розташовуються в один рядок:

```
     <?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:orientation="horizontal" >

      <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello"
      android:textSize="26sp" />

      <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Android"
      android:textSize="26sp" />

     <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Nougat"
      android:textSize="26sp" />
     </LinearLayout>
```

LinearLayout підтримує таку властивість, як вага елемента, яка задається атрибутом android:layout_weight. Це властивість вказує, яку частину залишку вільного місця контейнера по відношенню до інших об’єктів займе даний елемент.

Наприклад, якщо один елемент у нас буде мати для властивості android:layout_weight значення 2, а інший – значення 1, то в сумі вони дадуть 3, тому перший елемент буде займати 2/3 простору контейнера, а другий – 1/3.

Якщо всі елементи мають значення android:layout_weight="1", то всі вони будуть рівномірно розподілені по всій площі контейнера.

5.4.2 RelativeLayout

RelativeLayout представляє об’єкт ViewGroup, який розміщує дочірні елементи відносно щодо позиції інших дочірніх елементів розмітки, або щодо області самої розмітки RelativeLayout. Використовуючи відносне

позиціонування, ми можемо встановити елемент по правому краю або в центрі або іншим способом, який надає даний контейнер.

Для установки елемента в файлі xml ми можемо застосовувати атрибути групи android:

```
layout_(android:layout_above,
             android:layout_below,
              android:layout_centerHorizontal,
             android:layout_alignTop,
            android:layout_alignParentLeft
тощо).
```

Наприклад:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/activity_main"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView android:text="Left Top"
 android:layout_height="wrap_content"
 android:layout_width="wrap_content"
 android:textSize="26sp"
 android:layout_alignParentLeft="true"
 android:layout_alignParentTop="true" />

 <TextView android:text="Right Top"
 android:layout_height="wrap_content"
 android:layout_width="wrap_content"
 android:textSize="26sp"
 android:layout_alignParentRight="true"
 android:layout_alignParentTop="true" />

<TextView android:text="Left Bottom"
 android:layout_height="wrap_content"
 android:layout_width="wrap_content"
 android:textSize="26sp"
 android:layout_alignParentLeft="true"
 android:layout_alignParentBottom="true" />

 <TextView android:text="Right Bottom"
 android:layout_height="wrap_content"
android:layout_width="wrap_content"
 android:textSize="26sp"
 android:layout_alignParentRight="true"
 android:layout_alignParentBottom="true" />

</RelativeLayout>
```

На екрані ми побачимо:

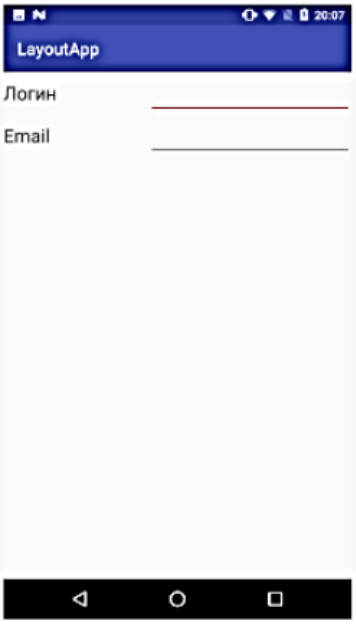
5.4.3 TableLayout

Контейнер TableLayout впорядковує дочірні елементи управління по стовпцях і рядках. Визначимо в файлі activity_main.xml елемент TableLayout, який буде включати два рядки і два стовпці:

```
     <TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

     <TableRow>

      <TextView
      android:layout_weight="0.5"
      android:text="Логин"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

     <EditText
      android:layout_weight="1"
      android:layout_width="match_parent"
     android:layout_height="wrap_content" />
      </TableRow>

     <TableRow>
      <TextView
```

```
 android:layout_weight="0.5"
 android:text="Email"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content" />
 <EditText
 android:layout_weight="1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content" />

</TableRow>
</TableLayout>
```

На екрані ми побачимо:



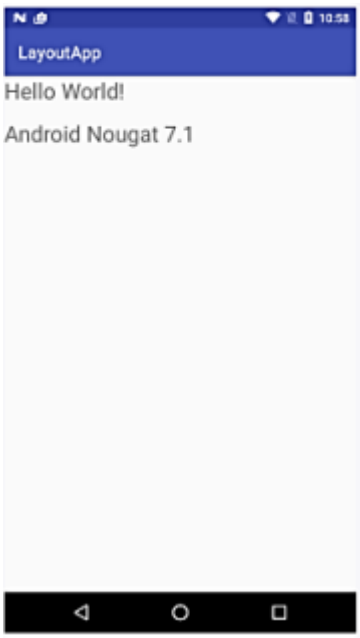
5.4.4 FrameLayout

Контейнер FrameLayout призначений для виведення на екран одного поміщеного в нього візуального елемента. Якщо ж ми помістимо декілька елементів, то вони будуть накладатися один на одного. Припустимо ми вклали в FrameLayout два елементи TextView:

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

<TextView
 android:layout_width="wrap_content"
```

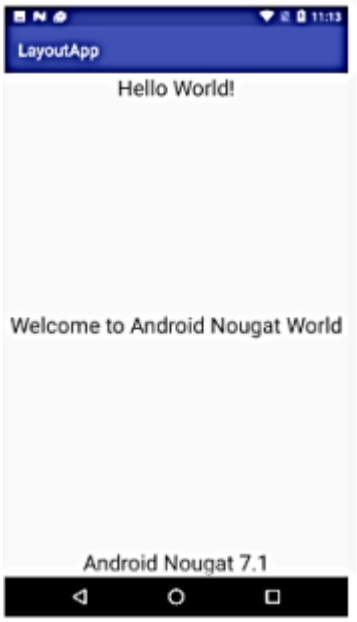
```
 android:layout_height="wrap_content"
 android:text="Hello World!"
 android:textSize="26sp"/>

<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Android Nougat 7.1"
 android:textSize="26sp"
 android:layout_marginTop="50dp"/>

</FrameLayout>
```

Тут обидва елементи позиціонуються в одне і той же місце – в лівий верхній кут контейнера FrameLayout. Щоб уникнути накладання, в даному випадку в другого TextView встановлюється відступ зверху в 50 одиниць:



Елементи управління всередині FrameLayout можуть встановлювати своє позиціонування за допомогою атрибута *android: layout_gravity*. При заданні значення цього атрибуту ми можемо комбінувати ряд значень, розділяючи їх вертикальною лінією: *bottom | center_horizontal*.

В результаті можна отримати таку розмітку:



5.4.5 ConstraintLayout

ConstraintLayout представляє собою новий тип контейнерів, який є розвитком elativeLayout і дозволяє створювати гнучкі та масштабовані інтерфейси. Починаючи з версії Android Studio 2.3 ConstraintLayout був доданий в список стандартних компонентів і навіть є контейнером, який використовується в файлах layout за замовчуванням.

Розглянемо приклад його застосування:

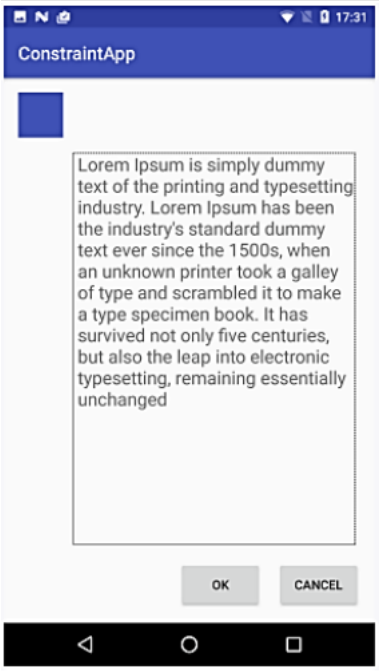
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout

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:id="@+id/activity_main"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

      <ImageView
      android:text="TextView"
      android:background="#3F51B5"
      android:layout_width="50dp"
      android:layout_height="50dp"
      android:id="@+id/imageView"
      app:layout_constraintLeft_toLeftOf="parent"
      android:layout_marginLeft="16dp"
      app:layout_constraintTop_toTopOf="parent"
     android:layout_marginTop="16dp" />
```

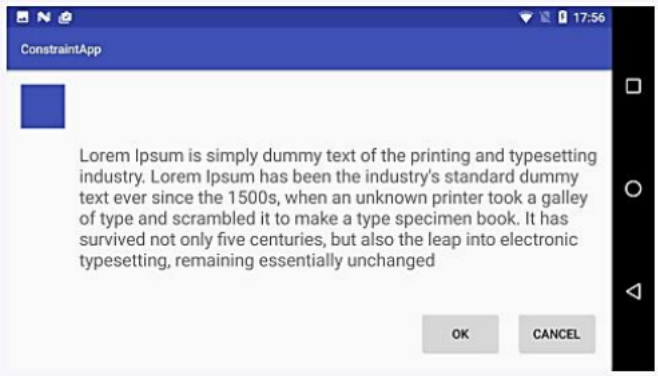
```
     <TextView
      android:layout_width="0dp"
      android:layout_height="0dp"
      android:textSize="20sp"
      android:id="@+id/textView"
      android:text="Lorem Ipsum is simply dummy text of the
printing and typesetting
     industry ... remaining essentially unchanged"
      app:layout_constraintLeft_toRightOf="@+id/imageView"
      android:layout_marginLeft="16dp"
      app:layout_constraintTop_toBottomOf="@+id/imageView"
      android:layout_marginTop="16dp"
      app:layout_constraintRight_toRightOf="parent"
      android:layout_marginRight="16dp"
      app:layout_constraintBottom_toTopOf="@+id/button2"
      android:layout_marginBottom="16dp" />

     <Button
      android:text="Cancel"
      android:layout_width="93dp"
      android:layout_height="53dp"
      android:id="@+id/button1"
      app:layout_constraintRight_toRightOf="parent"
      android:layout_marginRight="16dp"
      app:layout_constraintBottom_toBottomOf="parent"
      android:layout_marginBottom="16dp" />

      <Button
      android:text="OK"
      android:layout_width="93dp"
      android:layout_height="53dp"
      android:id="@+id/button2"
      app:layout_constraintRight_toLeftOf="@+id/button1"
      android:layout_marginRight="16dp"
      app:layout_constraintBaseline_toBaselineOf="@+id/button1"
/>

     </android.support.constraint.ConstraintLayout>
```

На екрані це виглядатиме так:

По-перше, тут елемент позиціонується відносно контейнера ConstraintLayout: від верхньої і лівої межі контейнера до відповідних меж ImageView по 16 dip.

По-друге, відносно контейнера позиціонується також кнопка з id=button1: від правої і нижньої межі контейнера до відповідних меж Button також по 16 dip.





По-третє, друга кнопка з id=button2 позиціонується відносно першої кнопки: від правої межі другої кнопки до лівої межі першої кнопки (app:layout_constraintRight_toLeftOf="@+id/button1") також 16 dip. І щоб обидві кнопки знаходилися на одному рівні, у них задано вирівнювання по базовій лінії: app:layout_constraintBaseline_toBaselineOf="@+id/button1".

Нарешті елемент TextView із фрагментом тексту позиціонується одразу відносно як до контейнера, так і до елемента ImageView і до другої кнопки.

5.5 Одиниці вимірювання розміру елементів екрану

Розрізняють декілька одиниць вимірювання розміру елементів екрану:

- px: пікселі поточного екрану. Дана одиниця вимірювання не

рекомендується, оскільки реальне представлення зовнішнього вигляду екрану програми може змінюватися залежно від конкретного пристрою; кожен пристрій має індивідуально визначену кількість пікселів на один дюйм. Відповідно, кількість пікселів на екрані є різною для різних пристроїв;

- dp: (device-independent pixels) – пікселі, незалежні від щільності екрану. Це абстрактна одиниця вимірювання,

яка базується на фізичній щільності екрану з роздільною здатністю 160 dpi (точок на дюйм).

У цьому випадку 1dp=1px. Якщо ж розмір екрана більший або менший, ніж 160dpi, кількість пікселів, які застосовуються для представлення 1dp, відповідно? збільшується або зменшується.

Наприклад, на екрані з 240 dpi 1dp=1,5px, а на екрані з 320dpi матимемо: 1dp=2px. Загальна формула для обчислення кількості фізичних пікселів з dp така: px=dp*(dpi/160);

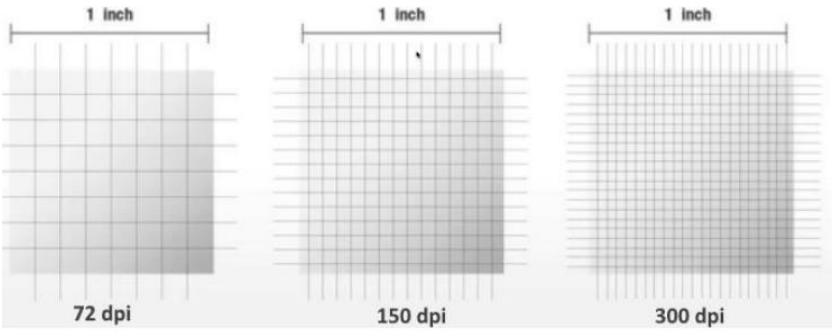
- sp: (scale-independent pixels) – пікселі, незалежні від масштабування.

Допускає налаштування розмірів користувачем. Рекомендується для роботи із шрифтами;

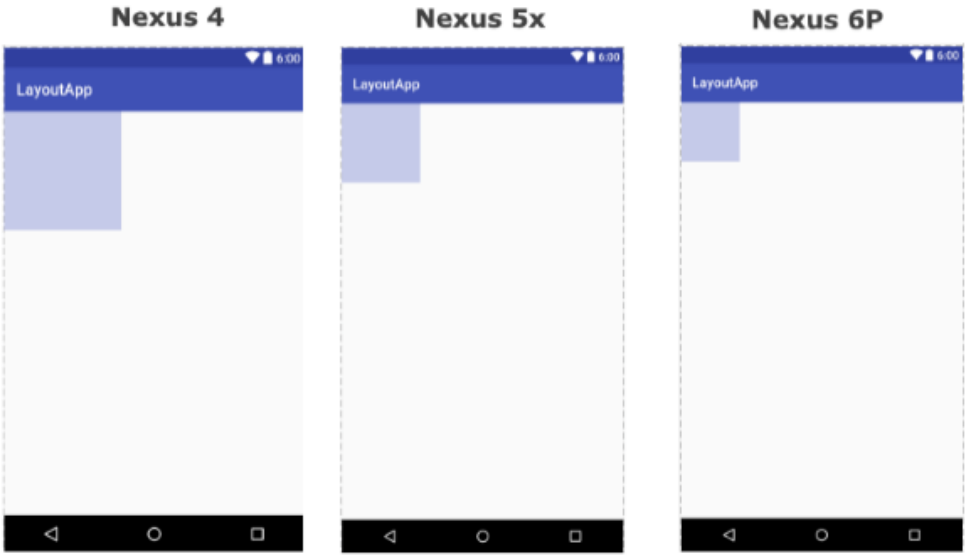
- pt: 1/72 дюйма, базується на фізичних розмірах екрану;
- mm: міліметри;
- in: дюйми.

Рекомендованою одиницею для використання є dp. Це пов’язано з тим, що світ мобільних пристроїв на Android досить сильно фрагментований в контексті широкого різноманіття розширення та розмірів екрану. І чим більшою є щільність (кількість пікселів на один дюйм), тим більше пікселів буде на екрані:

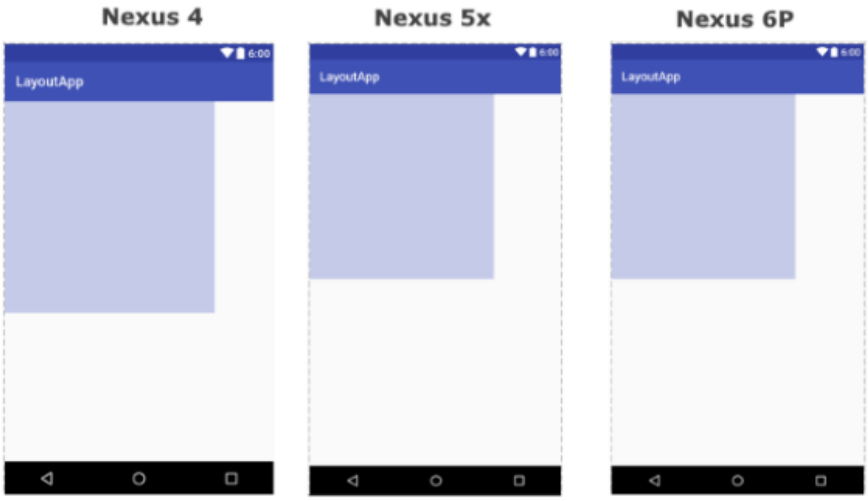
Використовуючи стандартні фізичні пікселі, ми неминуче стикнемося із тою проблемою, що розміри елементів сильно варіюватимуться для різних екранів.



Наприклад, для 3 пристроїв з різними характеристиками екрану Nexus 4, Nexus 5X та Nexus 6P, квадрат розмірами 300px на 300px на екрані виглядатиме так:



Тепер візьмемо квадрат із сторонами 300dp на 300dp. Як бачимо, тепер квадрат виглядає більш-менш коректно.



Для спрощення роботи всі розміри розбито на декілька груп:

- ldpi (low): ~120dpi;
- mdpi (medium): ~160dpi;
- hdpi (high): ~240dpi (Nexus One);
- xhdpi (extra-high): ~320dpi (Nexus 4);
- xxhdpi (extra-extra-high): ~480dpi (Nexus 5/5X, Samsung Galaxy S5);
- xxxhdpi (extra-extra-extra-high): ~640dpi (Nexus 6/6P, Samsung Galaxy S6).

Питання для самоконтролю за темою

Для самостійної перевірки знань доцільно сформулювати розширені відповіді на поставлені питання і перевірити їх повноту та правильність за допомогою матеріалів запропонованих літературних джерел.

1 Які компоненти екрану ви знаєте?
2 Як працювати з контейнером ConstraintLayout?
3 В яких випадках зручно використовувати RelativeLayout?
4 Як отримати посилання на ресурс в коді Java?
5 На які групи розбиті розміри ектанів?

ТЕМА 6 «ЕЛЕМЕНТИ УПРАВЛІННЯ. РОБОТА З АДАПТЕРАМИ СПИСКІВ»

План лекції.

Огляд елементів управління додатком.

Пояснення до теми.

6.1 Елементи управління TextView, EditView, Button, Checkbox, RadioButton. Створення обробників подій та прив'язка їх до елементів управління

6.1.1 TextView

Для простого перегляду тексту на екрані призначений елемент TextView. Він відображає текст без можливості редагування. Деякі його основні атрибути:

● android:text: встановлює текст елемента;

● android:textSize: встановлює висоту тексту, в якості одиниць виміру для вказівки висоти використовуються sp;

● android:background: задає фоновий колір елемента у вигляді кольору в шістнадцятковому вигляді або у вигляді колірного ресурсу;

● android:textColor: задає колір тексту;

● android:textAllCaps: при значенні true робить всі символи в тексті великими;

● android:textDirection: встановлює напрямок тексту. За замовчуванням використовується напрямок зліва направо, але за допомогою значення rtl можна встановити напрямок справа наліво;

● android:textAlignment: задає вирівнювання тексту. Може приймати наступні значення:

- center : вирівнювання по центру;
- textStart : на початку тексту відносно абзацу;
- textEnd : в кінці тексту;
- viewStart : по лівому краю (відносно екрана);
- viewEnd : по правому краю;

● android:fontFamily: встановлює тип шрифту. Може приймати наступні значення:

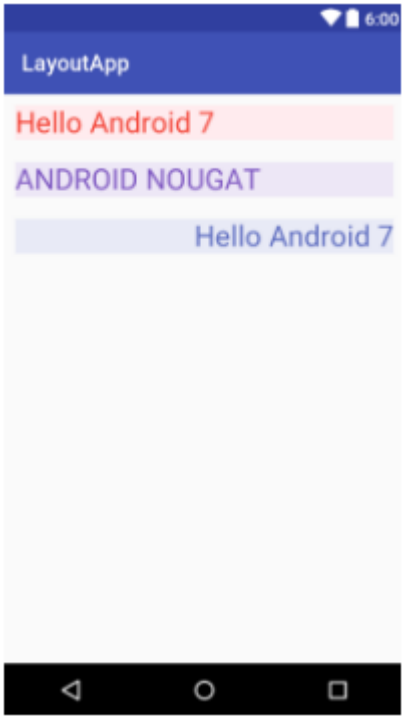
- monospace;
- serif;
- serif-monospace;

- sans-serif;
- sans-serif-condensed;
- sans-serif-smallcaps;
- sans-serif-light;
- casual;
- cursive.

Наприклад, визначимо три текстових поля:

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

<TextView
 android:layout_height="wrap_content"
 android:layout_width="match_parent"
 android:layout_margin="10dp"
 android:text="Hello Android 7"
 android:fontFamily="sans-serif"
 android:textSize="26sp"
 android:background="#ffebee"
 android:textColor="#f44336" />

<TextView
 android:layout_height="wrap_content"
 android:layout_width="match_parent"
 android:layout_margin="10dp"
 android:text="Android Nougat"
 android:textAllCaps="true"
 android:textSize="26sp"
 android:background="#ede7f6"
 android:textColor="#7e57c2" />

<TextView
 android:layout_height="wrap_content"
 android:layout_width="match_parent"
 android:layout_margin="10dp"
 android:text="Hello Android 7"
 android:textAlignment="textEnd"
android:textSize="26sp"
 android:background="#e8eaf6"
 android:textColor="#5c6bc0" />

</LinearLayout>
```

На екрані вони відображуватимуться так:



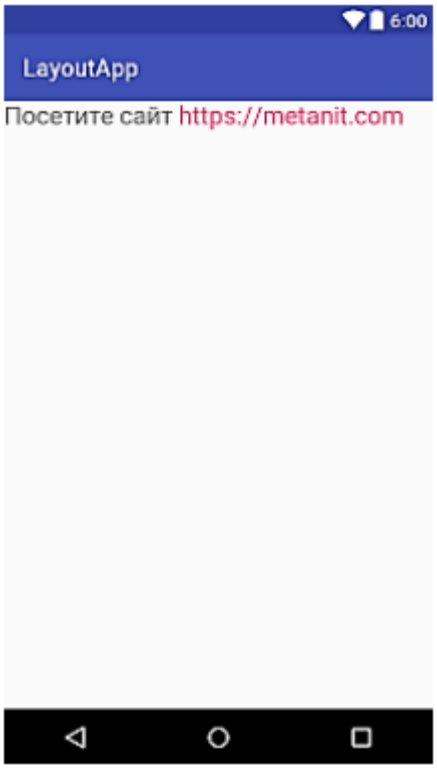
Задання елемента в коді теж не є складним. Наприклад, створимо елемент і виведемо його на екран:

```
public class MainActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 LinearLayout linearLayout=new LinearLayout(this);
 TextView textView1=new TextView(this);
 textView1.setBackgroundColor(0xffe8eaf6); // колір фону
 textView1.setTextColor(0xff5c6bc0); // колір тексту
 textView1.setAllCaps(true); // всі букви великі

textView1.setTextAlignment(TextView.TEXT_ALIGNMENT_CENTER); //
 вирівнювання тексту по центру
 textView1.setText("Android Nougat 7"); // текстове
значення
 textView1.setTypeface(Typeface.create("casual",
Typeface.NORMAL)); // шрифт
 textView1.setTextSize(26); // висота тексту
 LinearLayout.LayoutParams layoutParams = new
LinearLayout.LayoutParams
 (ViewGroup.LayoutParams.MATCH_PARENT,
 ViewGroup.LayoutParams.WRAP_CONTENT);
 layoutParams.setMargins(20,20,20,20); // зовнішні
відступи
 textView1.setLayoutParams(layoutParams); // розміри
 linearLayout.addView(textView1);
```

```
 setContentView(linearLayout);
 }
}
```

Іноді необхідно вивести на екран посилання або телефон, після натискання на які відбувалася б певна дія. Для цього в TextView визначено атрибут android:autoLink :

```
<TextView android:id="@+id/display_message"
 android:text="Посетите сайт https://metanit.com"
 android:textSize="21sp"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:autoLink="web|email" />
```



Властивість *android:autoLink* може приймати декілька значень:

- none: відключає всі посилання;
- web: включає всі веб-посилання;
- email: включає посилання на електронні адреси;
- phone: включає посилання на номери телефонів;
- map: включає посилання на карту;
- all: включає всі перераховані вище посилання.

Тобто при налаштуванні android: autoLink="web" якщо в тексті є адреса url, то ця адреса буде виділятися, а при натисканні на нього буде

здійснено перехід до веббраузера, який відкриє сторінку за цією адресою. За допомогою прямої риски ми можемо об’єднувати умови, як в даному випадку: *android:autoLink="web|email"*.

6.1.2 EditText

Елемент EditText є підкласом класу TextView. Він також представляє текстове поле, але вже з можливістю введення і редагування тексту. Таким чином, в EditText ми можемо використовувати всі ті ж можливості, що і в TextView.

З тих атрибутів, що не розглядалися в підрозділі про TextView, слід зазначити атрибут android:hint. Він дозволяє задати текст, який буде відображатися в якості підказки, якщо елемент EditText порожній. Крім того, ми можемо використовувати атрибут android:inputType, який дозволяє задати клавіатуру для введення.

Зокрема, серед його значень можна виділити наступні:

● text: звичайна клавіатура для введення однорядкового тексту;

● textMultiLine: багаторядкове текстове поле;

● textEmailAddress: звичайна клавіатура, на якій присутній символ @, орієнтована на введення email;

● textUri: звичайна клавіатура, на якій присутній символ /, орієнтована а

введення інтернет-адрес;

● textPassword: клавіатура для введення пароля;

● textCapWords: при введенні перший введений символ слова являє велику літеру, інші – малі;

● number: числова клавіатура;

● phone: клавіатура в стилі звичайного телефону;

● date: клавіатура для введення дати;

● time: клавіатура для введення часу;

● datetime: клавіатура для введення дати і часу.

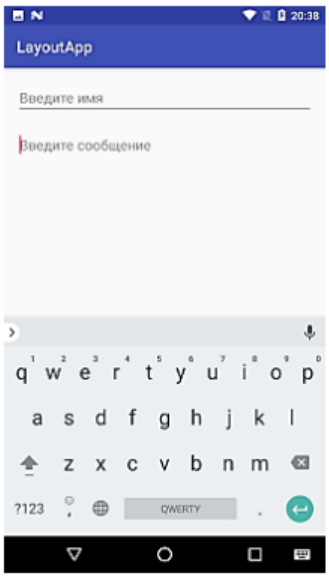
Використовуємо EditText на практиці:

```
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/activity_main"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical"
```

```
    android:padding="16dp">
    <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Введите имя" />
    <EditText
    android:layout_marginTop="16dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:hint="Введіть повідомлення"
    android:inputType="textMultiLine"
    android:gravity="top" />
</LinearLayout>
```

Перше поле тут звичайне однорядкове, а друге – багаторядкове. Щоб в другому олі текст вирівнювався по верху, додатково встановлюється атрибут android:gravity="top".

В результаті отримаємо:



Однією з можливостей елемента EditText також є можливість обробити введені символи під час введення користувача. Для цього визначимо в файлі activity_main.xmlнаступну розмітку:

```
    <LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
```

```
    android:padding="16dp">

    <TextView
    android:id="@+id/textView"
    android:textSize="34sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
    <EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Введіть ім'я"
    android:id="@+id/editText" />
</LinearLayout>
```

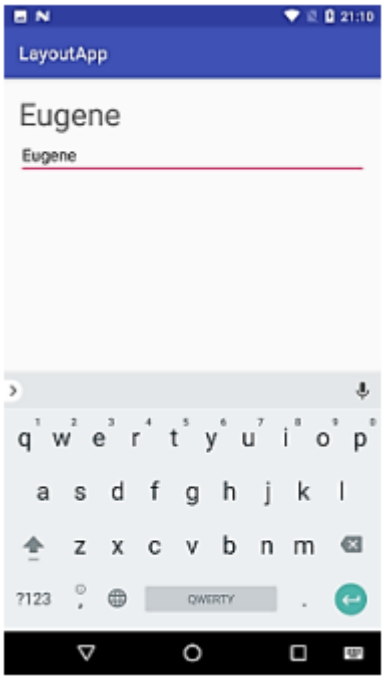
Передбачається, що введені в EditText символи тут же будуть відображатися в елементі TextView. Для цього також змінимо код MainActivity:

```
public class MainActivity extends AppCompatActivity {

@Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 EditText          editText          =          (EditText)
findViewById(R.id.editText);
 editText.addTextChangedListener(new TextWatcher() {
 public void afterTextChanged(Editable s) {}
 public void beforeTextChanged(CharSequence s, int start,
 int count, int after) {
 }

public void onTextChanged(CharSequence s, int start,
 int before, int count) {
 TextView          textView          =          (TextView)
findViewById(R.id.textView);
 textView.setText(s);
 }
 });
 }
}
```

За допомогою методу *addTextChangedListener()* тут до елементу EditText додається слухач введення тексту – об'єкт TextWatcher. Для його використання нам треба реалізувати три методу, але в реальності нам вистачить реалізації методу *onTextChanged*, який викликається при зміні тексту. Введений текст передається в цей метод в якості параметра

CharSequence. У самому методі просто передаємо цей текст в елемент TextView.

В результаті при введенні в EditText все символи також будуть відображатися в TextView:



6.1.3 Button

Одним з часто використовуваних елементів є кнопки, які представлені класом android.widget.Button. Ключовою особливістю кнопок є можливість взаємодії з користувачем через натискання. Деякі ключові атрибути, які можна задати у кнопок:

- text: задає текст на кнопці;
- textColor: задає колір тексту на кнопці;
- background: задає фоновий колір кнопки;
- textAllCaps: при значенні true встановлює текст в верхньому регістрі. За замовчуванням якраз і застосовується значення true;
- onClick: задає обробник натиснення кнопки.

Змінимо код в activity_main.xml наступним чином:

```
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/activity_main"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
```

```
 android:orientation="vertical"
 android:padding="16dp">
 <TextView
 android:id="@+id/textView"
 android:textSize="34sp"
android:layout_width="match_parent"
 android:layout_height="wrap_content" />
 <EditText
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:hint="Введіть ім’я"
 android:id="@+id/editText" />
 <Button
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Click"
 android:onClick="sendMessage" />
</LinearLayout>
```

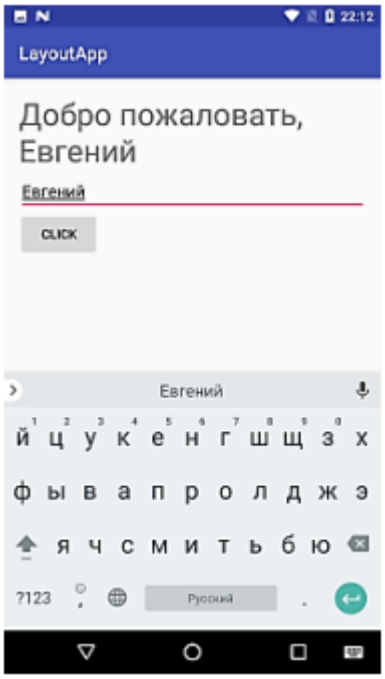
За допомогою атрибута *android:onClick* можна задати метод в коді Java, який буде обробляти натискання кнопки. Так, в наведеному вище прикладі це метод *sendMessage*. Тепер перейдемо до коду MainActivity і пропишемо в ньому такий метод:

```
public class MainActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
 // Обробка натискання на кнопку
 public void sendMessage(View view) {
 TextView textView=(TextView) findViewById(R.id.textView);
 EditText editText=(EditText) findViewById(R.id.editText);
 textView.setText("Ласкаво просимо, "+editText.getText());
 }
}
```

При створенні методу для обробки натискання слід враховувати наступні моменти:

- метод повинен оголошуватися з модифікатором public;
- метод повинен повертати значення void;
- як параметр метод повинен приймати об’єкт View, який представляє натиснуту кнопку.

В даному випадку після натискання на кнопку в TextView виводиться текст з EditText:



Аналогічний приклад повністю в коді MainActivity:

```
public class MainActivity extends AppCompatActivity {
 EditText editText;
 TextView textView;

@Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 LinearLayout linearLayout=new LinearLayout(this);
 linearLayout.setOrientation(LinearLayout.VERTICAL);
 textView=new TextView(this);
 textView.setLayoutParams(new LinearLayout.LayoutParams(
 LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.WRAP_CONTENT
 ));
 linearLayout.addView(textView);
editText=new EditText(this);
 editText.setHint("Введіть ім'я");
 editText.setLayoutParams(new LinearLayout.LayoutParams(
 LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.WRAP_CONTENT
 ));
 linearLayout.addView(editText);
 Button button=new Button(this);
 button.setText("CLICK");
 button.setLayoutParams(new LinearLayout.LayoutParams(
 LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT
 ));
```

```
 linearLayout.addView(button);
 button.setOnClickListener(new View.OnClickListener() {
 public void onClick(View v) {
 textView.setText("Ласкаво просимо, "+editText.getText());
 }
 });
 setContentView(linearLayout);
 }
}
```

При програмному створенні кнопки ми можемо визначити у неї слухач натискання View.OnClickListener і за допомогою його методу onClick також обробити натискання:

```
button.setOnClickListener(new View.OnClickListener() {
 public void onClick(View v) {
 // Обробка натискання
 }
});
```

6.1.4 Checkbox

Елементами Checkbox є прапорці, які можуть перебувати в відміченому і невідміченому стані. Прапорці дозволяють виробляти множинний вибір з кількох значень. Визначимо в файлі розмітки activity_main.xml кілька прапорців:

```
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/activity_main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:padding="16dp">

<TextView android:id="@+id/selection"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:textSize="26sp" />

<CheckBox android:id="@+id/java"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Java"
 android:textSize="26sp"
 android:onClick="onCheckboxClicked"/>

<CheckBox android:id="@+id/javascript"
```

```
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="JavaScript"
 android:textSize="26sp"
 android:onClick="onCheckboxClicked"/>
</LinearLayout>
```

Атрибут android:onClick, як і в випадку з простими кнопками, дозволяє задати обробник натиснення на прапорець. Визначимо обробник натиснення в коді

```
MainActivity:
public class MainActivity extends AppCompatActivity {

@Override
 protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }

public void onCheckboxClicked(View view) {
 CheckBox language=(CheckBox) view; // отримуємо прапорець
 boolean checked=language.isChecked(); // перевіряємо, чи
він відмічений
 TextView                              selection=(TextView)
findViewById(R.id.selection);

 // Перевіряємо, який саме прапорець відмічено
 switch(view.getId()) {
 case R.id.java:
 if (checked){
 selection.setText("Java");
 }
 break;
 case R.id.javascript:
 if (checked)
 selection.setText("JavaScript");
 break;
 }
 }
}
```

В якості параметра в обробник натискання *onCheckboxClicked* передається натиснутий прапорець. За допомогою методу *isChecked()* можна дізнатися, чи виділений прапорець – в цьому випадку метод повертає true. За допомогою конструкції switch ... case можна отримати id натиснутого прапорця і виконати відповідні дії.

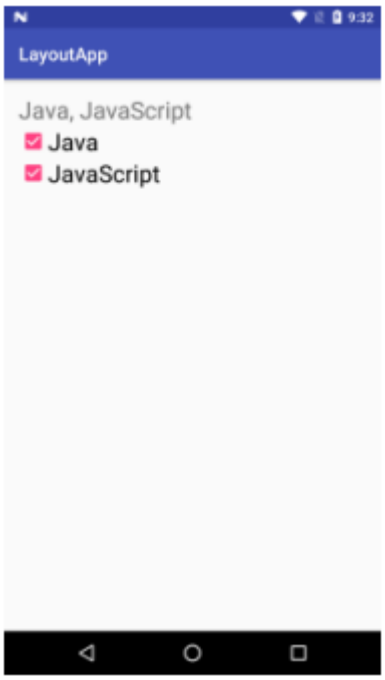
Якщо нам просто треба взяти текст з обраного прапорця, то не обов'язково в даному випадку використовувати конструкцію switch, бо ми можемо скоротити весь код наступним чином:

```
public void onCheckboxClicked(View view) {
 CheckBox language=(CheckBox) view;
 TextView                              selection=(TextView)
findViewById(R.id.selection);
 if(language.isChecked())
 selection.setText(language.getText());
}
```

Але в даному випадку у нас є проблема: в текстовому полі відображається тільки один виділений елемент. Змінимо код MainActivity, щоб показати обидва виділених елементи:

```
public class MainActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
 public void onCheckboxClicked(View view) {
 // отримуємо прапорці
CheckBox java=(CheckBox) findViewById(R.id.java);
```

```
      CheckBox javascript=(CheckBox)
findViewById(R.id.javascript);
      String selectedItems="";
      if(java.isChecked())
      selectedItems+=java.getText() + ", ";
      if(javascript.isChecked())
      selectedItems+=javascript.getText();
      TextView                              selection=(TextView)
findViewById(R.id.selection);
      selection.setText(selectedItems);
      }
     }
```



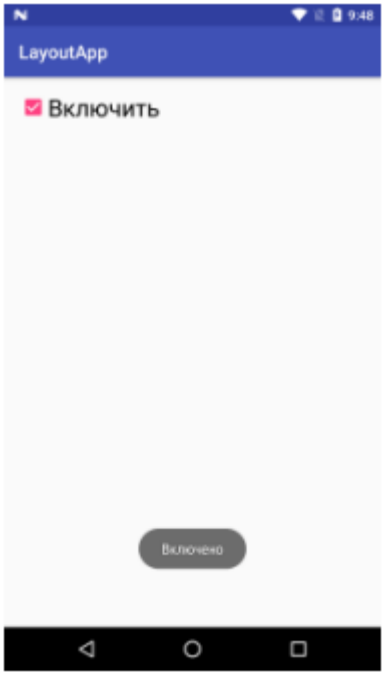
За допомогою слухача *OnCheckedChangeListener* можна відстежувати зміни прапорця. Цей слухач спрацьовує, коли ми встановлюємо або прибираємо позначку на прапорці.

Наприклад, визначимо наступний checkbox:

```
     <LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/activity_main"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical"
      android:padding="16dp">


     <CheckBox android:id="@+id/enabled"
```

```
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:checked="true"
 android:text="Ввімкнути"
 android:textSize="26sp" />
</LinearLayout>

Програмний код:
public class MainActivity extends AppCompatActivity {

@Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 CheckBox enableBox = (CheckBox)
findViewById(R.id.enabled);
 enableBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {

public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
 if (isChecked) {
 Toast.makeText(getApplicationContext(), "Ввімкнено",
Toast.LENGTH_SHORT).show();
 }

else {
 Toast.makeText(getApplicationContext(), "Вимкнено",
Toast.LENGTH_SHORT).show();
 }
 }
 });
 }
}
```

Слухач *OnCheckedChangeListener* визначено в базовому класі CompoundButton і він визначає один метод – *onCheckedChanged*. Перший параметр цього методу buttonView – сам змінений прапорець CheckBox. А другий параметр isChecked вказує, чи відзначений прапорець. При зміні стану прапорця, буде виводитися відповідне повідомлення:



6.1.5 RadioButton

Подібну до прапорців функціональність надають перемикачі, які представлені класом RadioButton. Але на відміну від прапорців одноразово в групі перемикачів ми можемо вибрати тільки один перемикач. Щоб створити список перемикачів для вибору, спочатку треба створити об’єкт RadioGroup , який буде включати в себе всі перемикачі:

```
     <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical" >
      <TextView android:id="@+id/selection"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textSize="26sp"
      />

     <RadioGroup
xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/radios"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical" >

      <RadioButton android:id="@+id/java"
     android:layout_width="wrap_content"
```

```
 android:layout_height="wrap_content"
 android:text="Java"
 android:onClick="onRadioButtonClicked"/>

<RadioButton android:id="@+id/javascript"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="JavaScript"
 android:onClick="onRadioButtonClicked"/>
 </RadioGroup>

</LinearLayout>
```

Оскільки клас RadioGroup є похідним від LinearLayout, то ми також можемо поставити вертикальну або горизонтальну орієнтацію списку, при тому включивши в нього не тільки власне перемикачі, а й інші об’єкти, наприклад, кнопку або TextView.

У класі MainActivity визначимо обробку вибору перемикачів:

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

 public void onRadioButtonClicked(View view) {
 // якщо перемикач ввімкнено

boolean checked = ((RadioButton) view).isChecked();
 TextView selection = (TextView)
findViewById(R.id.selection);

// Отримуємо ввімкнений перемикач
 switch(view.getId()) {
 case R.id.java:
 if (checked){
 selection.setText("Выбран Java");
 }
 break;
 case R.id.javascript:

if (checked){
 selection.setText("Выбран JavaScript");
 }
 break;
 }
 }
```

```
}
```

Окрім обробки натискання на кожен окремий перемикач, ми можемо в цілому визначити для всього RadioGroup з його перемикачами слухач OnCheckedChangeListener і обробляти в ньому натискання. Для цього приберемо з розмітки у перемикачів атрибути android:onClick, а у елемента RadioGroup визначимо id:

```
     <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical" >
      <TextView android:id="@+id/selection"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
     android:textSize="26sp"
      />
      <RadioGroup
xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/radios"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical">
      <RadioButton android:id="@+id/java"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
 android:text="Java"/>
 <RadioButton android:id="@+id/javascript"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="JavaScript"/>
 </RadioGroup>
</LinearLayout>
```

Далі в коді MainActivity призначимо об’єкту RadioGroup слухача:

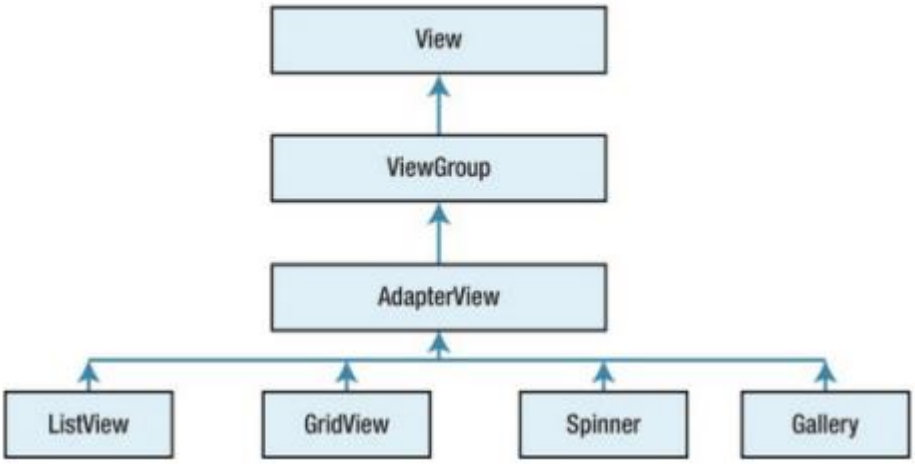
```
public class MainActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 RadioGroup radGrp=(RadioGroup)findViewById(R.id.radios);

// обробка зміни стану премикача
 radGrp.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {

@Override
 public void onCheckedChanged(RadioGroup arg0, int id) {
 TextView                              selection=(TextView)
findViewById(R.id.selection);
 switch(id) {
 case R.id.java:
 selection.setText("Выбран Java");
 break;
 case R.id.javascript:
selection.setText("Выбран JavaScript");
 break;
 default:
 break;
 }
 }});
 }
}
```

Слухач RadioGroup.OnCheckedChangeListener визначає метод onCheckedChanged(), в який передається об’єкт RadioGroup і id виділеного перемикача. Далі також ми можемо перевірити id і виконати певну обробку.

6.2 Адаптери та списки

Android представляє широку палітру елементів, які представляють списки. Всі вони є спадкоємцями класу android.widget.AdapterView. Це такі

віджети як ListView, GridView, Spinner. Вони можуть виступати контейнерами для інших елементів управління:



При роботі зі списками ми маємо справу з трьома компонентами. По-перше, це самі елементи списків (ListView, GridView), які відображають дані. По-друге, це джерело даних – масив, об'єкт ArrayList, база даних і т. д., в якому знаходяться самі відображаються дані. І по-третє, це адаптери – спеціальні компоненти, які пов'язують джерело даних з елементом списку.

Розглянемо зв'язок елемента ListView з джерелом даних за допомогою одного з таких адаптерів – класу ArrayAdapter.

6.2.1 ArrayAdapter

Клас ArrayAdapter є простим адаптером, який пов'язує масив даних з набором елементів TextView, з яких, наприклад, може складатися ListView. Тобто в даному випадку джерелом даних виступає масив об'єктів. ArrayAdapter викликає у кожного об'єкта метод toString()і отриманий рядок встановлює в елемент TextView.

Подивимося на прикладі. Отже, розмітка програми може виглядати так:

```
     <?xml version="1.0" encoding="utf-8"?>
     <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/activity_main"
      android:layout_width="match_parent"
      android:layout_height="match_parent">
      <ListView
      android:id="@+id/countriesList"
      android:layout_width="match_parent"
```

```
 android:layout_height="match_parent">
 </ListView>
</RelativeLayout>
```

Тут також визначено елемент ListView, який буде виводити список об’єктів. Тепер перейдемо до коду activity і пов’яжемо ListView через ArrayAdapter з деякими даними:

```
public class MainActivity extends AppCompatActivity {
 // набор даних, які ми пов’яжемо із списком
 String[] countries={"Бразилия", "Аргентина", "Колумбия",
"Чили", "Уругвай"};
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 // отримуємо ListView
 ListView countriesList=(ListView)
findViewById(R.id.countriesList);
 // створюємо адаптер
 ArrayAdapter<String> adapter=new ArrayAdapter(this,
android.R.layout.simple_list_item_1, countries)
 // задаємо адаптер для списку
 countriesList.setAdapter(adapter);
 }
}
```

Тут спочатку отримуємо по id елемент ListView і потім створюємо для нього адаптер. Для створення адаптера використовувався наступний конструктор ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, countries), де:

- this : поточний об’єкт activity;
- android.R.layout.simple_list_item_1 : файл розмітки списку, який фреймворк являє за замовчуванням. Він знаходиться в папці Android SDK platforms/[androidномер_версії]/data/res/layout.

Якщо нас не задовольняє стандартна розмітка списку, ми можемо створити свою і потім в коді змінити id на id потрібної нам розмітки

- countries : масив даних. Тут необов’язково вказувати саме масив, це може бути список ArrayList<T>.

В кінці необхідно встановити для ListView адаптер за допомогою методу setAdapter().

У підсумку ми отримаємо наступне відображення:

6.2.2 Ресурс string-array і ListView

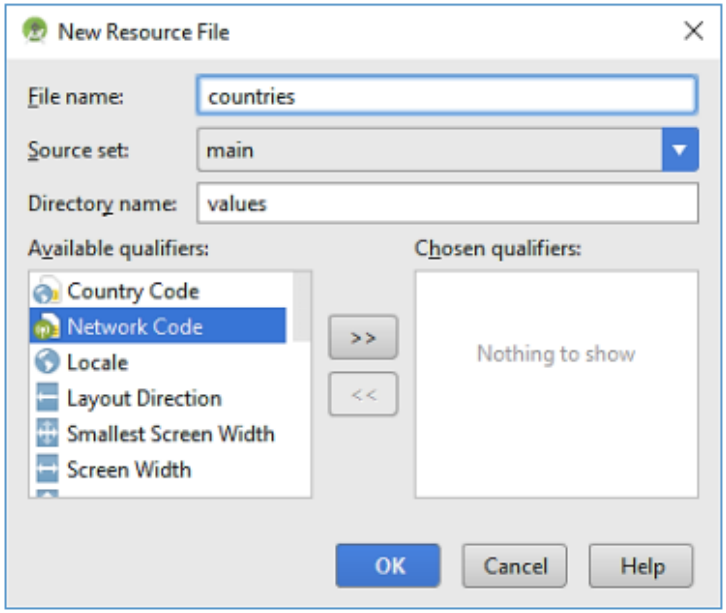
У попередньому підпункті було розглянуто, як виводити масив рядків (стрічок) за допомогою ArrayAdapter в ListView. При цьому масив рядків визначався програмно в коді java. Однак подібний масив рядків набагато зручніше було б зберігати в файлі xml у вигляді ресурсу.

Ресурси типу масивів рядків представляють елемент типу string-array. Вони можуть знаходиться в каталозі res/values в xml-файлі з довільним ім’ям. Визначення масивів рядків мають наступний синтаксис:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string-array name="ім’я_масиве_стрічок">
 <item>елемент</item>
 </string-array>
</resources>
```

Масив рядків задається за допомогою елемента <string-array>, атрибут name якого може мати довільне значення, за яким потім будуть посилатися на цей масив. Всі елементи масиву представляють набір значень <item>.

Наприклад, додамо в папку res/values новий файл. Для цього натиснемо правою кнопкою миші на даний каталог і меню виберемо пункт New -> Value resource file. У вікні назвемо файл як countries:



Після додавання файлу в папку res/values змінимо його вміст на такий:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string-array name="countries">
 <item>Бразилия</item>
 <item>Аргентина</item>
 <item>Колумбия</item>
 <item>Чили</item>
 <item>Уругвай</item>
</string-array>
</resources>
```

В файлі розмітки activity_main.xml залишається визначення елемента ListView:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/activity_main"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <ListView
 android:id="@+id/countriesList"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 </ListView>
</RelativeLayout>
```

Тепер зв’яжемо ресурс і ListView в коді MainActivity:

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

// отримуємо ListView
 ListView countriesList = (ListView)
findViewById(R.id.countriesList);

 // отримуємо ресурс
 String[] countries =
getResources().getStringArray(R.array.countries);

// створюємо адаптер
 ArrayAdapter<String> adapter = new ArrayAdapter(this,
 android.R.layout.simple_list_item_1, countries);

// задаємо адаптер для списку
 countriesList.setAdapter(adapter);
 }
}
```

Для отримання ресурсу в коді java застосовується вираз R.array.назва_ресурсу. Але нам необов’язково додавати список рядків в ListView програмно. У цього елемента є атрибут entries, який в якості значення може приймати ресурс string-array:

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

<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/activity_main"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

<ListView
 android:entries="@array/countries"
 android:id="@+id/countriesList"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 </ListView>

</RelativeLayout>
```

В цьому випадку код MainActivity ми можемо скоротити до стандартного:

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

А результат буде таким же самим:

Питання для самоконтролю за темою

Для самостійної перевірки знань доцільно сформулювати розширені відповіді на поставлені питання і перевірити їх повноту та правильність за допомогою матеріалів запропонованих літературних джерел.

1 Для чого використовують метод addTextChangedListener()?
2 Які елементи управління ви знаєте?
3 Яка властивість задає вирівнювання тексту для елемента TextView?
4 Які значення може приймати властивість android:autoLink?

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Офіційна документація для розробників під ОС Android. URL: https://developer.android.com/docs (дата звернення 12.06.2023)
2. Android Tutorial. URL : https://www.tutorialspoint.com/android/index.htm (дата звернення 12.06.2023)
3. John Horton. Android Programming for Beginners: Build in-depth, full-featured Android 9 Pie apps starting from zero programming experience, 2nd Edition.
4. Аналіз методів і технологій розробки мобільних додатків для платформи Android : навч. посіб. / О. В. Шматко, А. О. Поляков, В. М. Федорченко. – Харків: НТУ «ХПІ», 2018. – 284 с. URL: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/a4786ba9-e3ae-431c-bd93-a1be1a0eb64b/content
5. Android Studio. URL: http://developer.android.com/sdk/index.html (дата звернення 12.06.2023)

ГЛОСАРІЙ

Activity – асоціюється з окремим екраном або вікном додатку.

Android – операційна система для мобільних пристроїв

AndroidManifest.xml – це файл, в якому зберігаються опису фундаментальних характеристик програми і список всіх компонентів.

Broadcast receiver – це компонент Android-програми, який розсилає і реагує на широкомовні повідомлення.

Content providers – це компоненти Android-програми, надають доступ до даних (читання, додавання, оновлення).

Dalvik – віртуальнамашина Java

dp: (device-independent pixels) – пікселі, незалежні від щільності екрану.

Java 2 Micro Edition (J2ME) – це набір специфікацій і технологій, призначених для різних типів портативних пристроїв.

Java Virtual Machine – інтерпритатор Java-коду

Intent – це механізм для опису однієї операції (вибрати фотографію, відправити лист, зробити дзвінок, запустити браузер і перейти за вказаною адресою тощо).

Px – пікселі поточного екрану.

Service – це компонент Android-програми, певний процес, який запускається у фоновому режимі.

sp: (scale-independent pixels) – пікселі, незалежні від масштабування.

Віджети – це невеликі додатки, які відображаються у вигляді графічного об’єкта на робочому столі.

Додатки переднього плану – виконують свої функції лише тоді, коли видимі на екрані, в іншому ж випадку їх виконання призупиняється.

Емулятор – це віртуальний мобільний пристрій, який запускається на комп'ютері.

Емуляція (англ. Emulation) в обчислювальній техніці – комплекс програмних, апаратних засобів або їх поєднання, призначений для копіювання (або емуляції) функцій однієї обчислювальної системи (гостя) на інший, відмінній від першої, обчислювальної системі (хост) таким чином, щоб поведінка, яка емулюється, як можна ближче відповідала поведінці оригінальної системи (гостя).

Змішані додатки – більшу частину часу працюють у фоновому режимі, проте допускають взаємодію з користувачем і після свого налаштування.

Мобільна операційна система (мобільна ОС) – це операційна система для смартфонів, планшетів, КПК або інших мобільних пристроїв.

Файли layout – у додатках під Android візуальний інтерфейс часто завантажується із спеціальних файлів xml, які зберігають розмітку.

Фонові додатки – після налаштування не потребують взаємодії з користувачем а більшу частину часу перебувають і працюють в прихованому стані.

Навчальне електронне видання

ШТЕФАН Наталія Зінов’ївна

МОБІЛЬНІ ТЕХНОЛОГІЇ

(частина 1)

Конспект лекцій

Видавець і виготовлювач
Одеський державний екологічний університет
вул. Львівська, 15, м. Одеса, 65016
тел./факс: (0482) 32-67-35
E-mail: info@odeku.edu.ua
Свідоцтво суб’єкта видавничої справи
ДК № 5242 від 08.11.2016