МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП’ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з курсу: «Програмування мобільних пристроїв»

для студентів освітньо-кваліфікаційного рівня «бакалавр»
спеціальності 151 – «Автоматизація та комп’ютерно-інтегровані технології»

Тернопіль - 2019

Методичні вказівки до виконання лабораторних робіт з дисципліни «Програмування мобільних пристроїв» для освітнього ступеня – бакалавр, спеціальності 151 – «Автоматизація та комп'ютерно-інтегровані технології» галузі знань 15 – «Автоматизація та приладобудування» / Укл.: Заставний О.М. – Тернопіль: ТНЕУ, 2019. – 89с.

Методичні рекомендації до виконання лабораторних робіт складаються з частин, що рекомендовані освітньо-професійної програми підготовки бакалавра галузі знань 15 Автоматизація та приладобудування, спеціальність 151 – «Автоматизація та комп'ютерно-інтегровані технології»

Укладачі: Заставний Олег Михайлович к.т.н.,
Сегін Андрій Ігорович, к.т.н., доцент

Рецензенти: Сабадаш І.О. к.т.н., директор інституту мікропроцесорних систем керування об'єктами електроенергетики

Івасьєв С.В. к.т.н., старший викладач кафедри кібербезпеки Тернопільського національного економічного університету

Затверджено на засіданні кафедри СКС
протокол №1 від 28.08.2019.

Розглянуто та схвалено групою забезпечення з автоматизації та комп'ютерно-інтегрованих технологій
протокол №1 від 28.08.2019.

Розглянуто та схвалено вченою радою факультету комп'ютерних інформаційних технологій, протокол №2 від 18.09.2019 р.

ЗМІСТ

Лабораторна робота №1

Тема: Основні етапи розробки додатку з використанням Android Studio
Мета: Розробка простого додатку, що допомагає зрозуміти структуру додатку, освоїти основні оператори, користуватися середовищем розробки

Хід виконання роботи

Для досягнення поставленої цілі в лабораторній роботі створіть додаток у середовищі розробки Android Studio, докладно розглянемо структуру отриманого проекту і розберемо призначення основних його елементів.

Щоб подальші дії придбали якийсь сенс, сформулюємо завдання, яке буде вирішувати наш додаток, назвемо його "Відгадай число". Суть додатку в тому, що програма випадковим чином "загадує" число від 0 до 100, а користувач повинен відгадати це число. При кожному введенні числа програма повідомляє користувачеві результат: введене число більше загаданого, менше або "Ура, перемога!" число відгадано.

Розроблюваний додаток виконує свої функції тільки тоді, коли видимий на екрані, коли він не видимий його робота призупиняється, тобто маємо справу з додатком переднього плану. Для виконання всієї роботи досить визначити одну активність у додатку, фонові процеси не передбачені.

Далі в роботі розглянемо найпростіші елементи інтерфейсу користувача і додамо їх в додаток, а також розглянемо питання, пов'язані безпосередньо з програмуванням: навчимося обробляти події, що виникають при взаємодії програми з користувачем; реалізуємо логіку перевірки числа на співпадіння із заданим.

1.1 Створення додатку та вивчення його структури

Створіть новий проект в середовищі Android Studio.

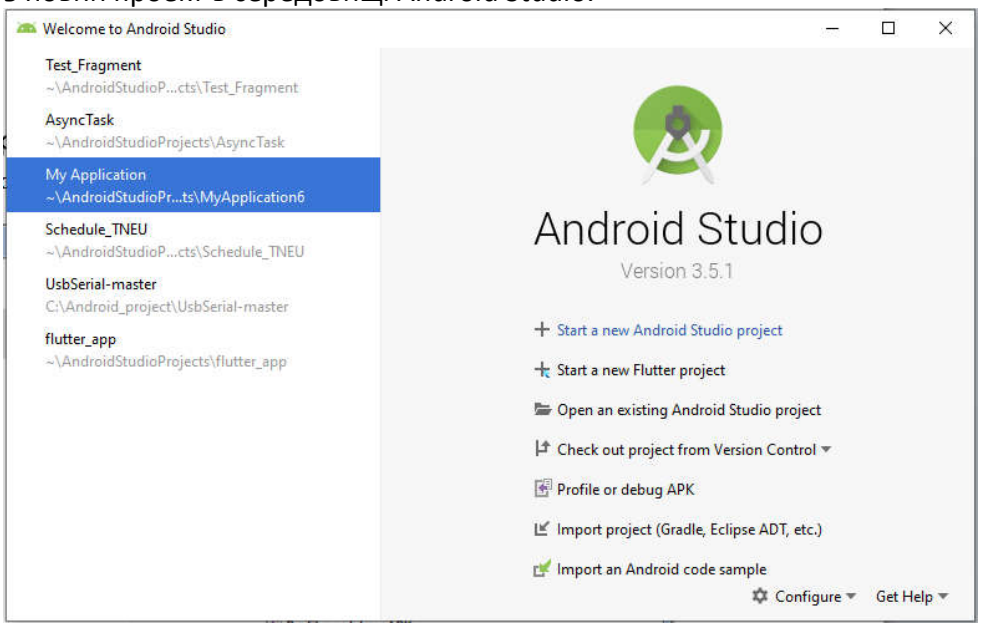

Рис. 1.1 – Головне вікно створення додатку

Вкладка Configure –> SDK Manadger дозволяє налаштувати SDK (Software Development Kit) для підтримки різних версій Android (рис 1.2).

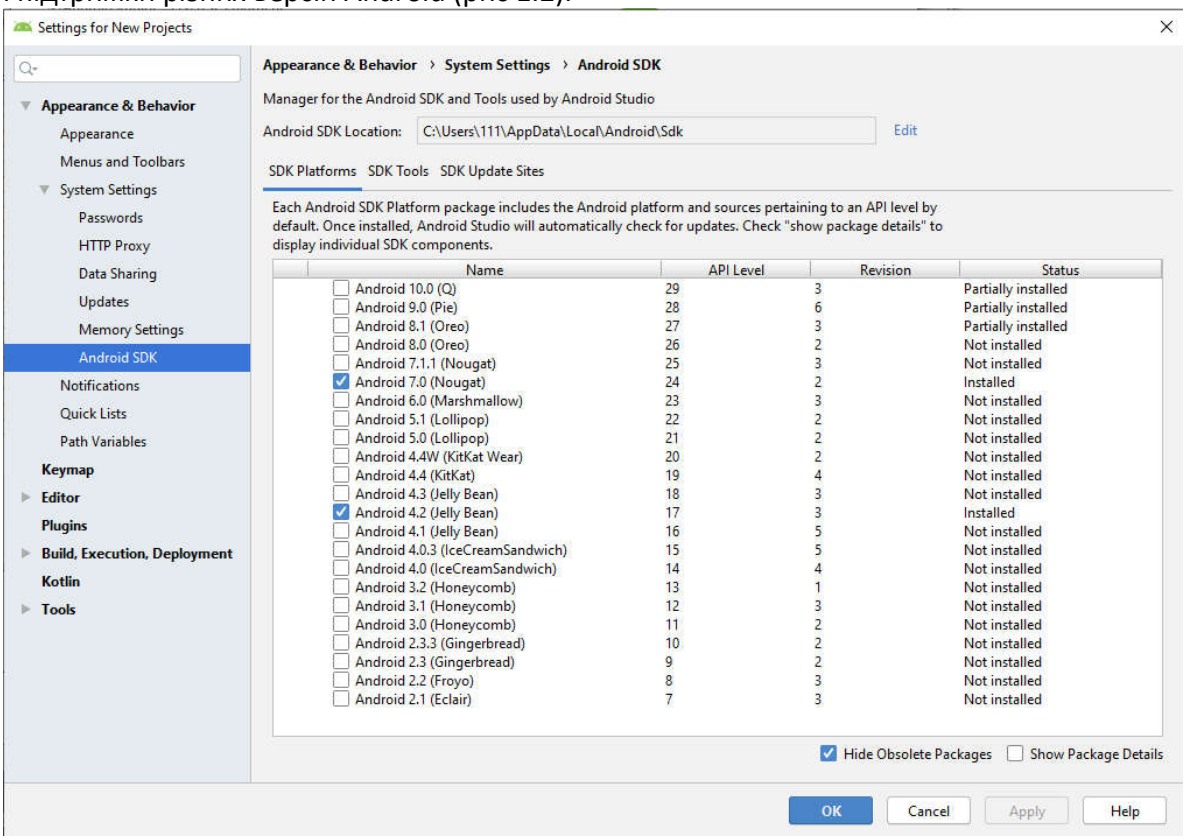

Рис. 1.2 – Вікно налаштувань версій SDK

Після завершення налаштувань повертаємось в головне вікно і вибираємо Start new Android Studio project (Рис. 1.3), та вибираємо тим проекту Empty Activity

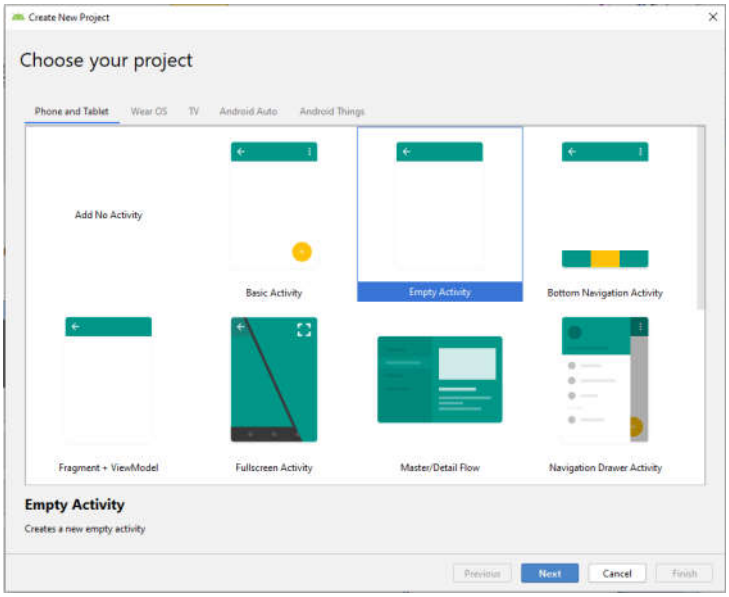

Рис 1.3 – Створення нового додатку

Далі вводимо ім’я проекту, мову програмування та мінімальний рівень API (рис.1.4)

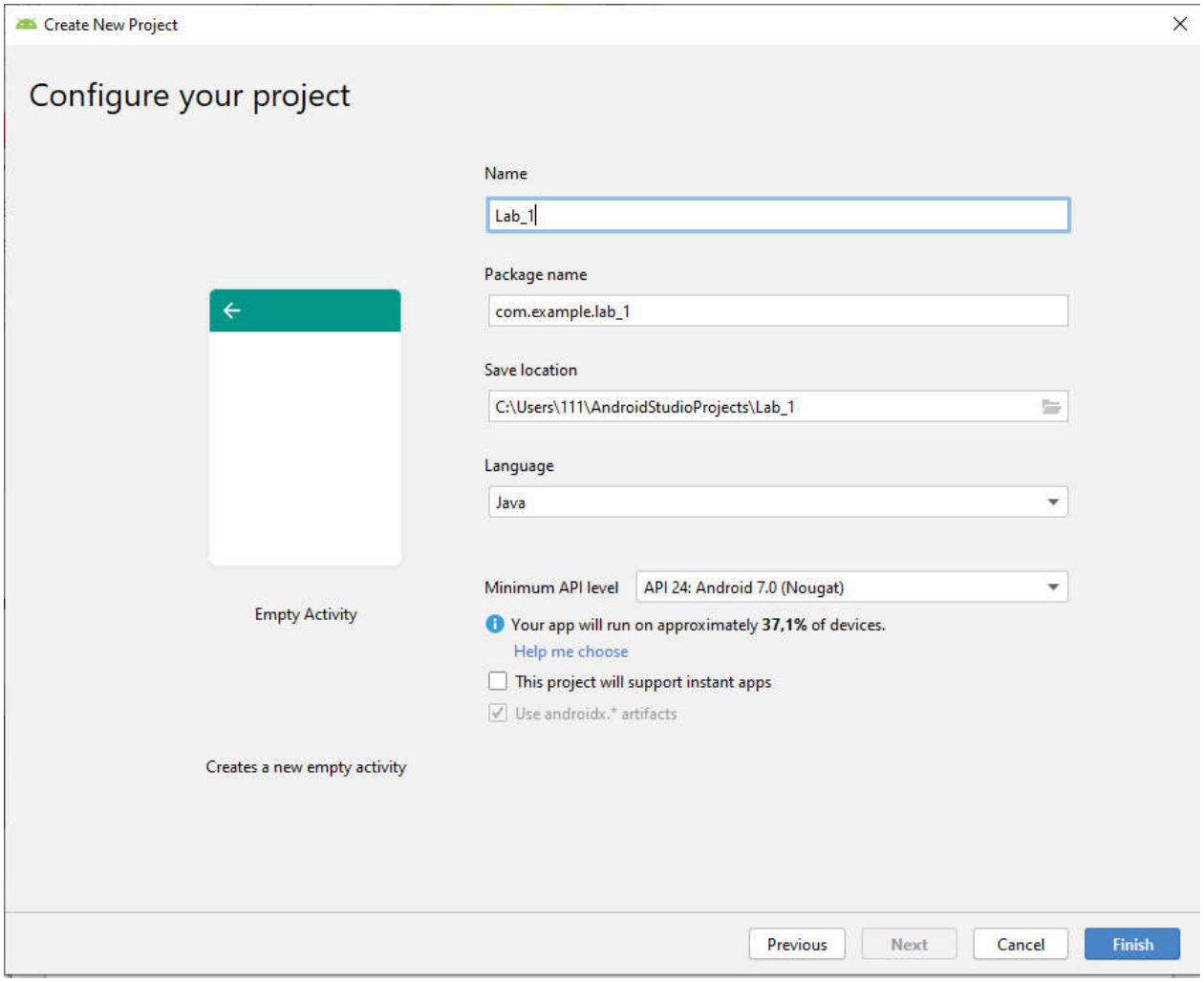

Рис. 1.4 – Ввід назви додатку, вибір мови програмування та мінімального рівня API

У процесі створення проекту, ми назвали його Lab_1, середовище розробки готує необхідні папки і файли. Повний ієрархічний список обов'язкових елементів проекту можна побачити на вкладці Project, ієрархія отриманих папок і файлів для нашого проекту зображена на рис. 1.5.

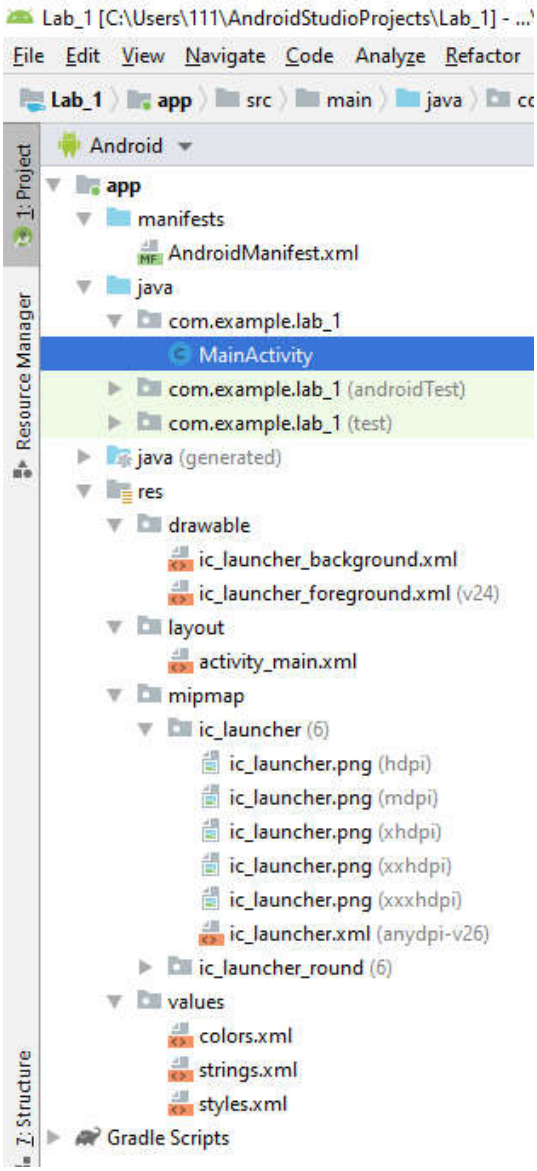

Рис. 1.5 – Структура проекту

В даний час нас буде цікавити призначення декількох файлів і папок.

Розглянемо папки:

Папка **manifests** – містить файл AndroidManifest.xml - файл у форматі xml, який описує основні властивості проекту, дозвіл на використання ресурсів пристрою та ін.

Папка **java** - містить файли з вихідним кодом на мові Java. Саме в цій папці розміщуються всі класи, створювані в процесі розробки програми. Зараз в цій папці в пакеті com.example.lab_1 розміщується єдиний клас MainActivity.java. Цей клас визначає головну і єдину активність в цьому додатку.

Примітка 1: Ім'я пакету присвоюється в процесі створення програми в полі Package Name, використовувати com.example не рекомендується, так як пакет з таким ім'ям не можна завантажити в Google Play. Часто рекомендують використовувати в якості імені пакета назву сайту програміста, записане в зворотному порядку, можна просто використовувати свої ім'я

та прізвище. Останнє слово в імені пакету формується автоматично і збігається з ім'ям проекту.

Примітка 2: Ім'я файлу присвоюється в процесі створення програми на етапі налаштування активності. Ім'я визначається в поле Activity Name.

Папка **res** - містить структуру папок ресурсів програми, розглянемо деякі з них:

- layout - в цій папці містяться xml-файли, які описують зовнішній вигляд форм і їх елементів, поки там знаходиться тільки activity_main.xml;
- values - містить XML файли, які визначають прості значення, таких ресурсів як, рядки, числа, кольори, теми, стилі, які можна використовувати в цьому проекті;
- menu - містить XML файли, які визначають все меню програми.

Розглянемо файл AndroidManifest.xml - файл у форматі xml, який описує основні властивості проекту, дозвіл на використання ресурсів пристрою та ін. Відразу після створення програми файл AndroidManifest.xml виглядає так:

```
      <?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.lab_1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        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>
```

Приклад фрагмету файлу маніфесту з додатковими елементами

```
<manifest  xmlns:android="http://schemas.android.com/apk/res/android"
         package="string"
         android:sharedUserID="string"
         android:sharedUserLabel="string resource"
         android:versionCode="integer"
         android:versionName="string"
         android:installLocation=["string" | "internalOnly" | "preferExternal"]
>
    . . .
</manifest>
```

Розглянемо докладно файл маніфесту.

Перший обов'язковий елемент <manifest> є кореневим елементом файлу, повинен містити обов'язковий елемент <application> і всі інші елементи за потребою. Розглянемо основні атрибути цього елемента:

Таблиця 1.1

xmlns:android	визначає простір імен Android, завжди повинен мати значення: "http://schemas.android.com/apk/res/android". Обов'язковий атрибут.
package	повне ім'я пакета, в якому розташовується додаток. Обов'язковий атрибут. Ім'я повинно бути унікальним, може містити великі і малі латинські букви, числа і символ підкреслення. Однак починатися повинно тільки з букви. Для уникнення конфліктів з іншими розробниками рекомендується використовувати ім'я вашого сайту (якщо він є) записаний в зворотньому порядку. У нашому випадку пакет має ім'я "**com.example.lab_1**". Увага: якщо Ви опублікували свій додаток, Ви не можете змінювати ім'я пакета, тому що ім'я пакета є унікальним ідентифікатором для програми та в разі його зміни додаток буде розглядатися, як зовсім інший і користувачі попередньої версії не зможуть його оновити.
android:versionCode	внутрішній номер версії програми не видно користувачеві. Цей номер використовується тільки для визначення чи є одна версія більш сучасною в порівнянні з іншого, більший номер показує більш пізню версію.
android:versionNumber	номер версії, є рядком і використовується тільки для того, щоб показати користувачеві номер версії програми.
android:shareUserID, android:sharedUserLabel, android:installLocation	ці атрибути в нашому файлі маніфесту не представлені, про їх призначення можна почитати за посиланням: http://developer.android.com/guide/topics/manifest/manifest-element.html.

Розглянемо елемент <application>, який є обов'язковим елементом маніфесту, повністю визначає склад додатку. Являє собою контейнер для елементів <activity »,« service »,« receiver »,« provider> (і не тільки), кожен з яких визначає відповідний компонент програми. Містить набір атрибутів, дія яких поширюється на всі компоненти програми. Розглянемо атрибути елемента <application>, представлені в маніфесті:

Таблиця 1.2

android:allowBackup	визначає дозвіл для додатку брати участь в резервному копіюванні та відновленні. Якщо значення цього атрибута false, то для додатку не може бути створена резервна копія, навіть якщо проводиться резервне копіювання всієї системи цілком. За замовчуванням значення цього атрибута дорівнює true.

Продовження табл. 1.2

android:icon	визначає іконку для додатку в загальному, а також іконку за замовчуванням для компонентів додатку, яка може бути перевизначена атрибутом android: icon кожного компоненту. Здається як посилання на графічний ресурс, що містить зображення, в нашому випадку значення цього атрибута дорівнює "@mipmap/ ic_launcher".
android:label	визначає видимий для користувача заголовок додатку в загальному, а також заголовок за замовчуванням для компонентів додатку, який може бути перевизначений атрибутом android: label кожного компонента. Здається як посилання на стрічковий ресурс, в нашому випадку значення атрибута дорівнює "Lab_1".
android:theme	визначає тему за замовчуванням для всіх активностей додатку, може бути перевизначений атрибутом android: theme кожної активності. Здається як посилання на стильовий ресурс, в нашому випадку значення атрибута дорівнює "@style/AppTheme".

Насправді у елемента <application> набагато більше атрибутів, ніж нам вдалося розглянути, знайти повний список атрибутів з описами можна за посиланням: http://developer.android.com/guide/topics/manifest/application-element.html.

У додатку всього одна активність, інших компонентів немає, в зв'язку з цим елемент <application> в маніфесті містить рівно один елемент <activity> і більше ніяких інших елементів не містить. Розглянемо елемент <activity>, який визначає активність. Для кожної активності обов'язково необхідний свій елемент <activity> в маніфесті. Розглянемо атрибути елемента <activity>, представлені в маніфесті:

Таблиця 1.3

android:name	визначає ім'я класу, який задає активність. Значення атрибута повинно повністю визначати ім'я класу із зазначенням пакета, в якому розташовується клас. Наприклад "com.tneu.scs.lab_1.MainActivity". Можна використовувати скорочений запис ".MainActivity", в цьому випадку додається ім'я пакета, визначену відповідним атрибутом елемента <manifest>.
android:configChanges	перераховує зміни конфігурації, якими може керувати активність. Якщо конфігурація змінюється під час роботи, то за замовчуванням активність зупиняється і перезапускається. Якщо ж зміна конфігурації вказано в цьому атрибуті, то при появі цієї зміни активність не перезапускається, замість цього вона продовжує працювати і викликає метод onConfigurationChanged(). Наприклад: атрибут має значення "orientation | keyboardHidden | screenSize", тобто при зміні орієнтації екрану, зміні розміру екрану і зміні доступності клавіатури не відбудеться перезапуск активності.
android:label	визначає видимий користувачеві заголовок активності, якщо він відрізняється від загального заголовка програми. Здається як посилання на строковий ресурс.

Продовження табл. 1.3

android:theme	визначає тему активності, якщо вона відрізняється від загальної теми програми, заданої відповідним атрибутом елемента <application>. Здається як посилання на стильової ресурс, наприклад значення атрибута може дорівнювати "@style/FullscreenTheme".

Насправді у елемента <activity> набагато більше атрибутів, ніж нам вдалося розглянути, знайти повний список атрибутів з описами можна за посиланням: http://developer.android.com/guide/topics/manifest/application-element.html.

У маніфесті для нашого застосунку елемент <activity> містить рівно один елемент: <intent-filter>, що визначає типи намірів, які може приймати активність. Цей елемент містить два елементи: <action> і <category>.

Перший елемент визначає дії, які проходять в фільтр намірів, при цьому <intent-filter> повинен містити хоча б один елемент <action>, в іншому випадку жоден об'єкт-намір не зможе пройти через фільтр і активність не можливо буде запустити. Елемент <action> має єдиний атрибут android: name = "android.intent.action.MAIN".

Другий елемент визначає ім'я категорії в фільтрі намірів. Має єдиний атрибут android: name = "android.intent.category.LAUNCHER".

На цьому розбір маніфесту додатки закінчимо, докладно з описом всіх елементів цього файлу можна ознайомитися за посиланням: http://developer.android.com/guide/topics/manifest/manifest-intro.html.

Найчастіше, при створенні програми доводиться мати справу з папками java, res/layout і res/values, тому що там знаходяться основні файли проекту.

1.2 Налаштування інтерфейсу додатку

До того, як почнемо формувати інтерфейс, є сенс підготувати можливість перевірки додатку на помилки. Щоб не завантажувати кожен раз додаток на реальний пристрій, в Android SDK передбачена можливість використання віртуального пристрою (AVD або Android Virtual Device), котрий емулює роботу реального смартфона.

Для створення зовнішнього виду програми необхідно визначити які елементи графічного інтерфейсу нам потрібні, як ці елементи будуть розташовуватися на формі і яким чином буде реалізовано взаємодію з користувачем.

Так як додаток дуже простий, то і інтерфейс особливою складністю відрізнятися не буде. Нам буде потрібне поле для введення чисел (TextEdit), текстова мітка для виведення інформації (TextView) і кнопка для підтвердження введеного числа (Button). Розташовувати елементи інтерфейсу будемо один під одним, зверху інформаційна частина, нижче поле вводу, кнопку розмістимо в самому низу програми. Взаємодія програми з користувачем організовується дуже просто: користувач вводить число в поле вводу і натискає кнопку, читає результат в інформаційному полі і, або радіє перемозі, або вводить нове число.

Якщо користувач вводить правильне число, то додаток пропонує йому зіграти знову при цьому кнопка буде грати роль підтвердження, а в інформаційне поле буде виведено запрошення до повторної гри.

Схематично інтерфейс програми зображений на рис. 1.6

Рис. 1.6 – Схема інтерфейсу додатку «Відгадай число»

Нам необхідно додати на форму три елементи: інформаційне поле (TextView), поле вводу (TextEdit) і кнопку (Button).

Android IDE підтримує два способи для виконання дій по формуванню інтерфейсу програми: перший базується на XML-розмітці, другий належить до візуального програмування і дозволяє перетягувати об'єкти інтерфейсу і розміщувати їх на формі за допомогою мишки. Вважається, що візуальний спосіб підходить для новачків, а більш просунуті розробники можуть писати код вручну, однак найчастіше використовується комбінований підхід.

Для формування інтерфейсу будемо працювати з файлом res/layout/activity_main.xml. На рис. 1.7 можна побачити редактор, відповідний візуальному способу формування інтерфейсу, цьому режиму відповідає вкладка Design

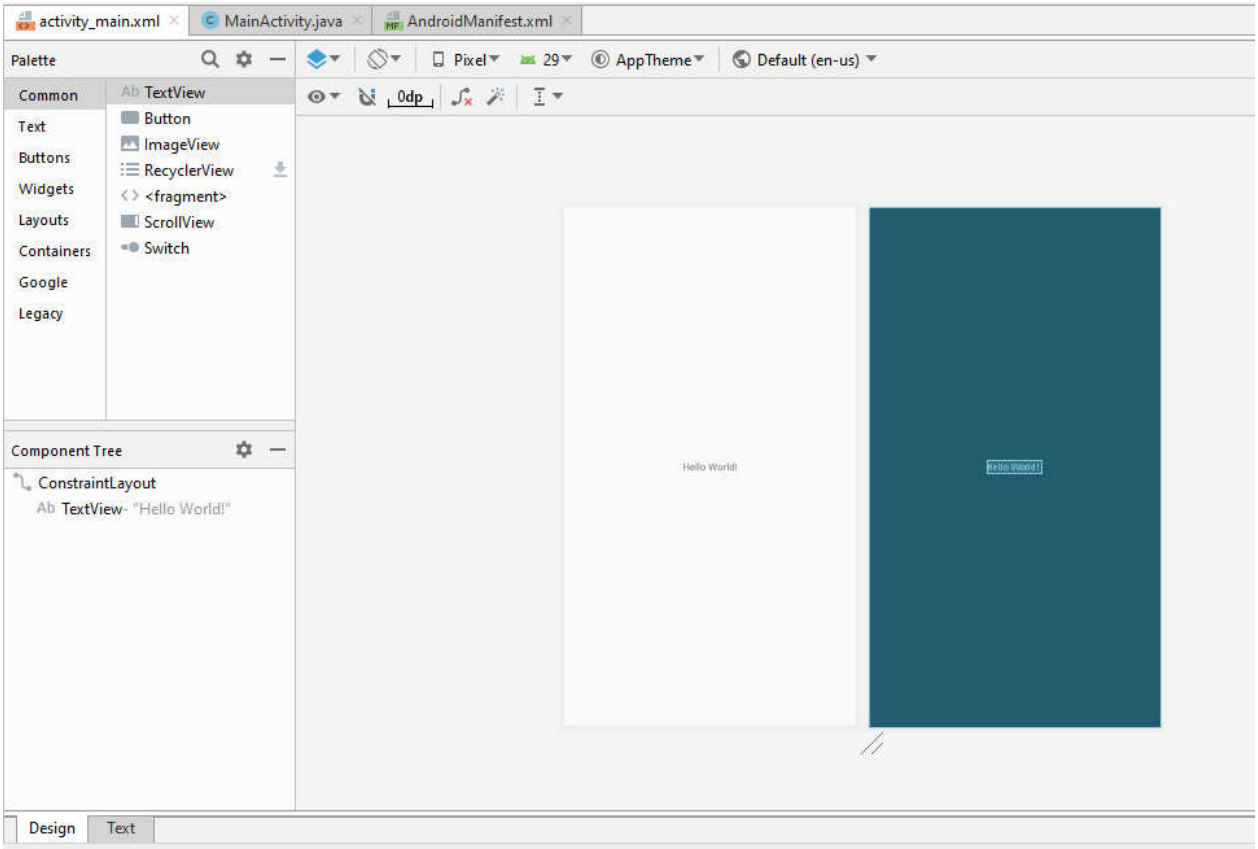

Рис. 1.7 – Графічне зображення активності додатку

На рис. 1.7 поруч з вкладкою Design розташована вкладка Text. Вона відповідає режиму редагування інтерфейсу шляхом формування XML файлу. На рис. 1.8 можна побачити редактор XML файлу.

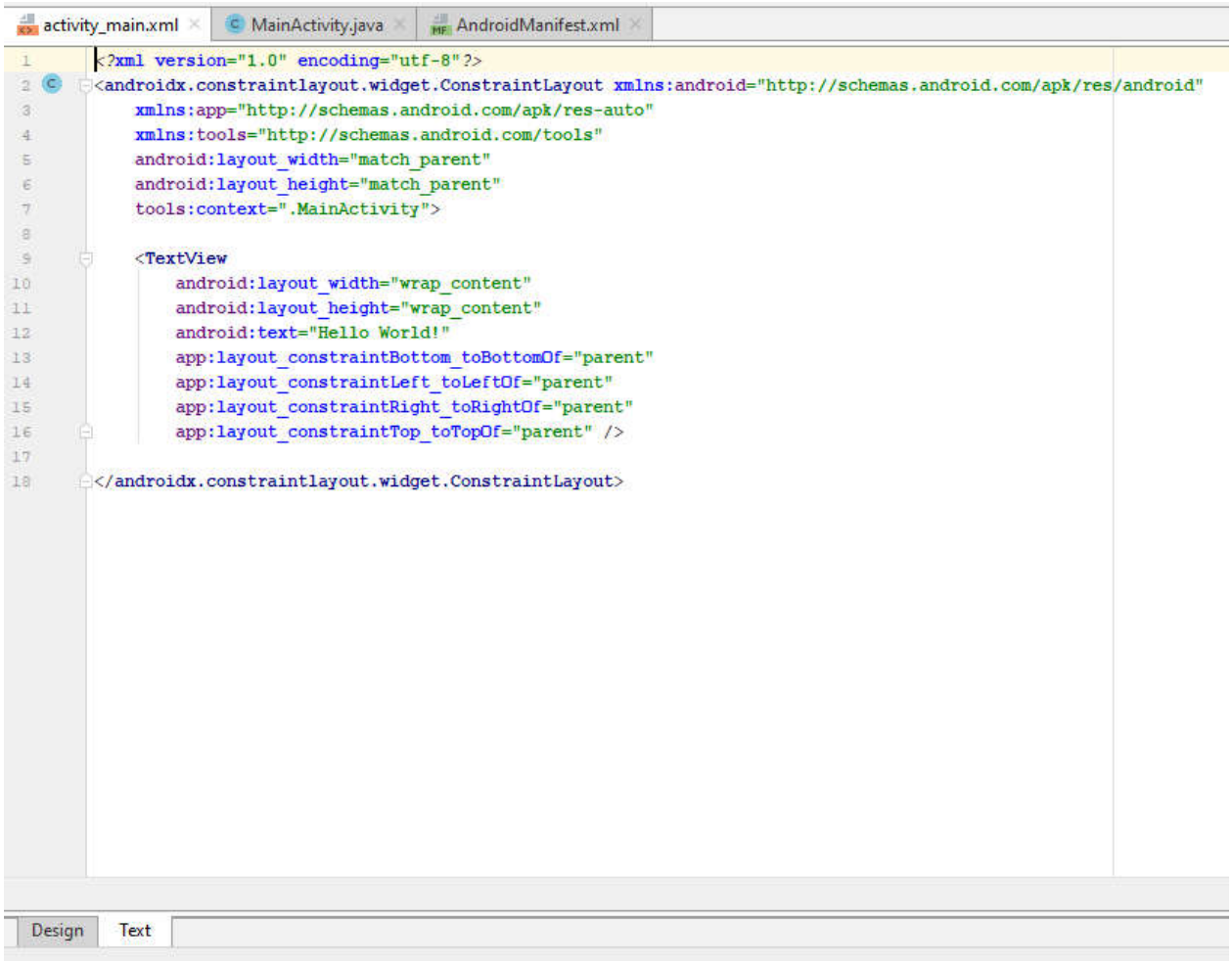

Рис. 1.8 – Опис активності в XML форматі

Задамо лінійне розташування компонентів на формі, для цього виберемо вкладку Layouts, знайдемо там LineralLayout і додамо його на форму. На рис. 1.9 можна побачити результат цих дій.

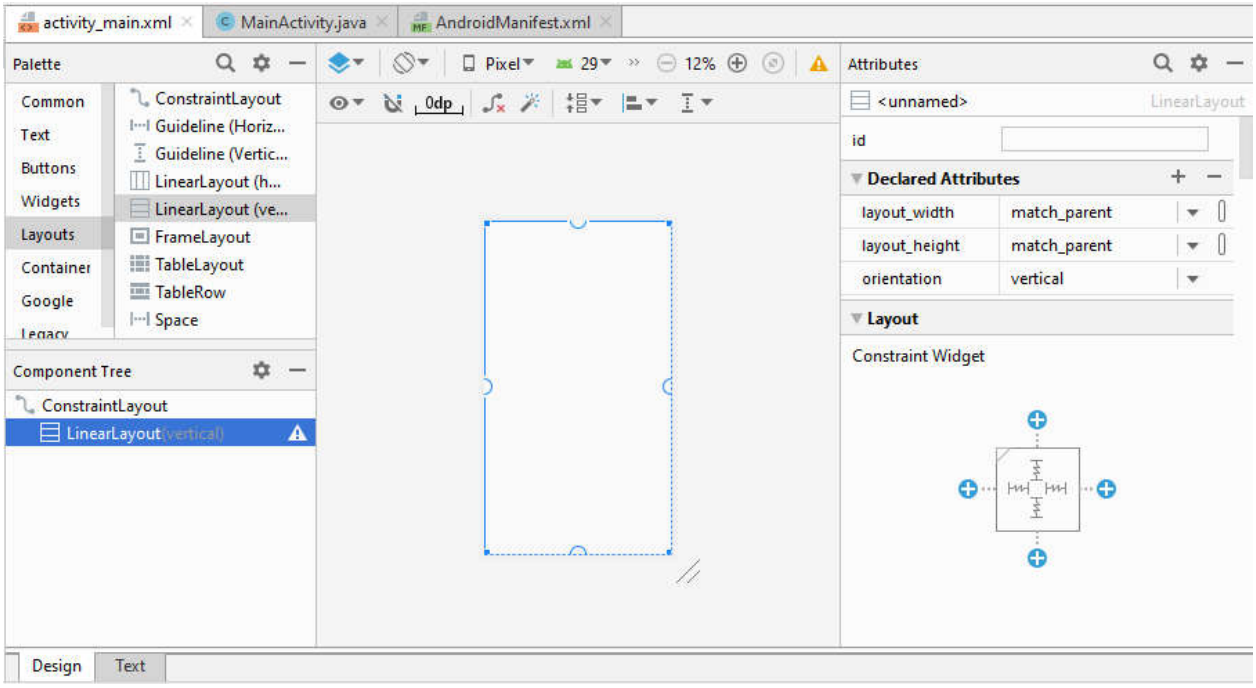

Рис. 1.9 – Налаштування інтерфейсу

Тепер почнемо додавати елементи інтерфейсу, будемо використовувати графічний режим редагування.

Для початку необхідно додати інформаційне поле. Для цього на панелі Palette вибираємо вкладку Text, на цій вкладці знайдемо поле TextView, перенесемо у вікно програми, розмістимо в першому рядку макету (LinearLayout).

Наступним потрібно розмістити поле вводу інформації, на вкладці Text знайдемо текстове поле Number(Decimal) і розмістимо у другому рядку макету.

І останнім на вкладці Button, виберемо там елемент Button і додамо в третій рядок макету рис. 1.10.

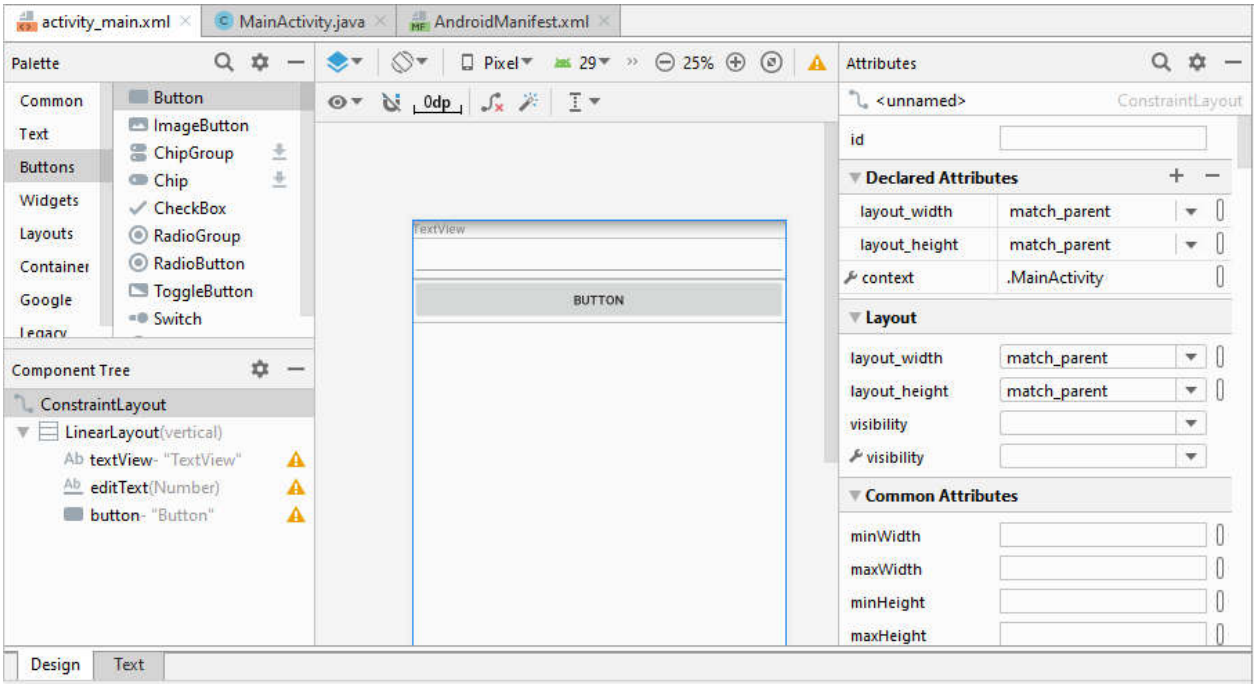

Рис.1.10 – Інтерфейс програми

Після настройки інтерфейсу можна заглянути в файл activity_main.xml, в цьому файлі прописано, що використовується LinearLayout і дано опис кожного з трьох рядків.

Тепер необхідно наповнити наші елементи інтерфейсу змістом, нам знадобиться текст для спілкування з користувачем, при програмуванні під Android існує практика розділяти ресурси і код програми. Для зберігання будь-яких рядків, які можуть знадобитися для роботи з додатком, використовується файл strings.xml. Зберігання всіх строкових ресурсів в цьому файлі серйозно полегшує локалізацію додатку на інші мови. Цей файл можна знайти в Project Explorer в папці res/values. Відкриємо його і подивимося, що там є. Приберемо зайві рядки і додамо нові:

```
<resources>
    <string name="app_name">Lab_1</string>
    <string name="behind">Більше</string>
    <string name="ahead">Менше</string>
    <string name="hit">Відгадано</string>
    <string name="input_value">Ввести значення</string>
    <string name="play_more">Зіграти ще</string>
    <string name="try_to_guess">Спробуйте відгадати число (1 &amp;#8211;
100)</string>
    <string name="error">Невірний ввід!</string>

</resources>
```

Дані змінні будуть виконувати такі завдання:

- app_name встановить "видиму" назву додатку;
- behind, ahead, hit сповістять користувача про його успіхи в грі;
- play_more і try_to_guess встановить назву кнопки, яка пояснить її функції;
- input_value запросить користувача до введення числа;
- error повідомить при невірному вводі.

Після зміни strings.xml, при переході на іншу вкладку, не забудьте зберегти зміни (найшвидший спосіб - натиснути Ctrl + S).

Налаштуємо текст в інформаційному полі. Для цього виберемо елемент textView (або в вікні дизайну або в списку елементів Component Tree) і на вкладці Attributes в правій частині вікна (це і є наше інформаційне поле, є сенс придумати йому більш осмислене ім'я) знайдемо властивість Text, підставимо в нього значення рядка з ім'ям try_to_guess, див. Рис. 1.11.

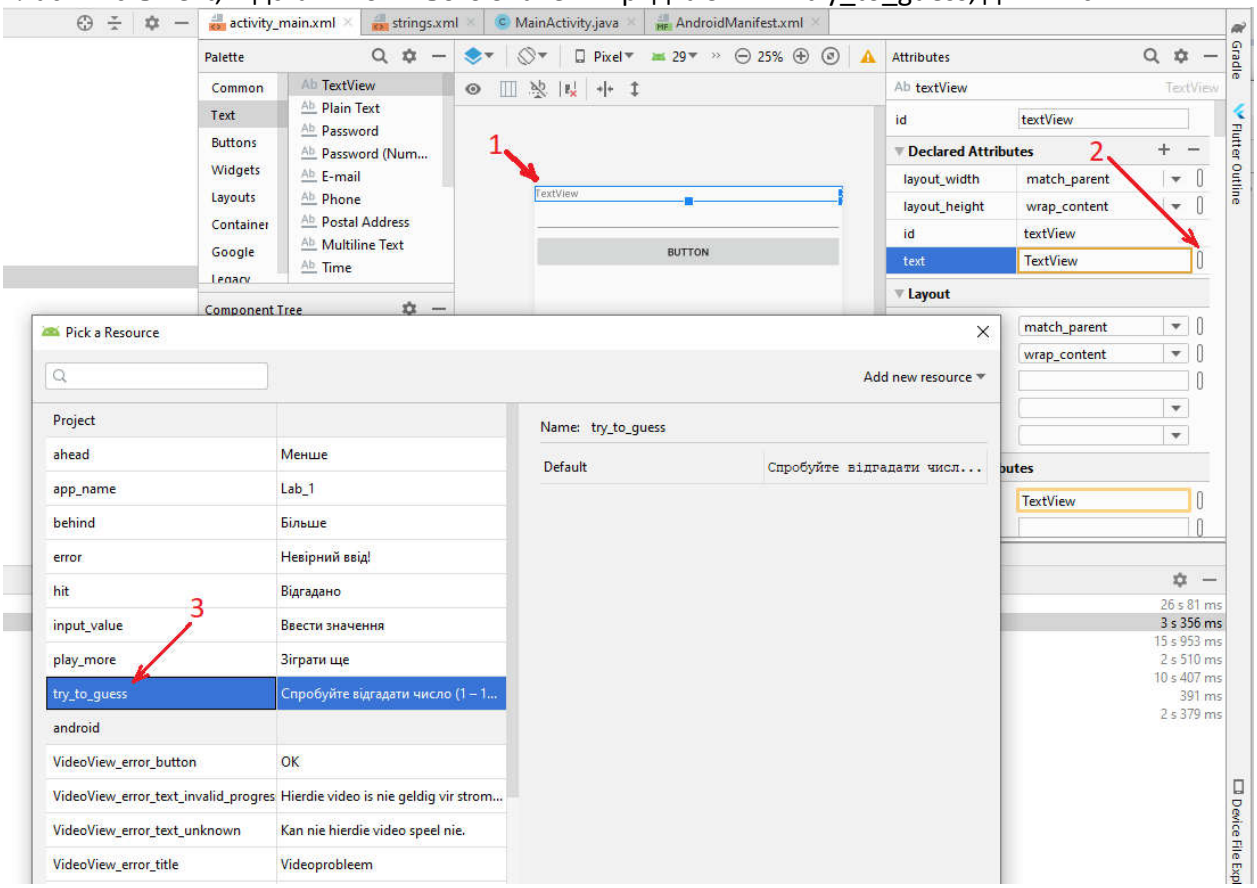

Рис. 1.11 – Налаштування текстової інформації для TextView

Аналогічно можна налаштувати текст, яким нас вітатиме кнопка, тільки в цьому випадку треба працювати з елементом button.

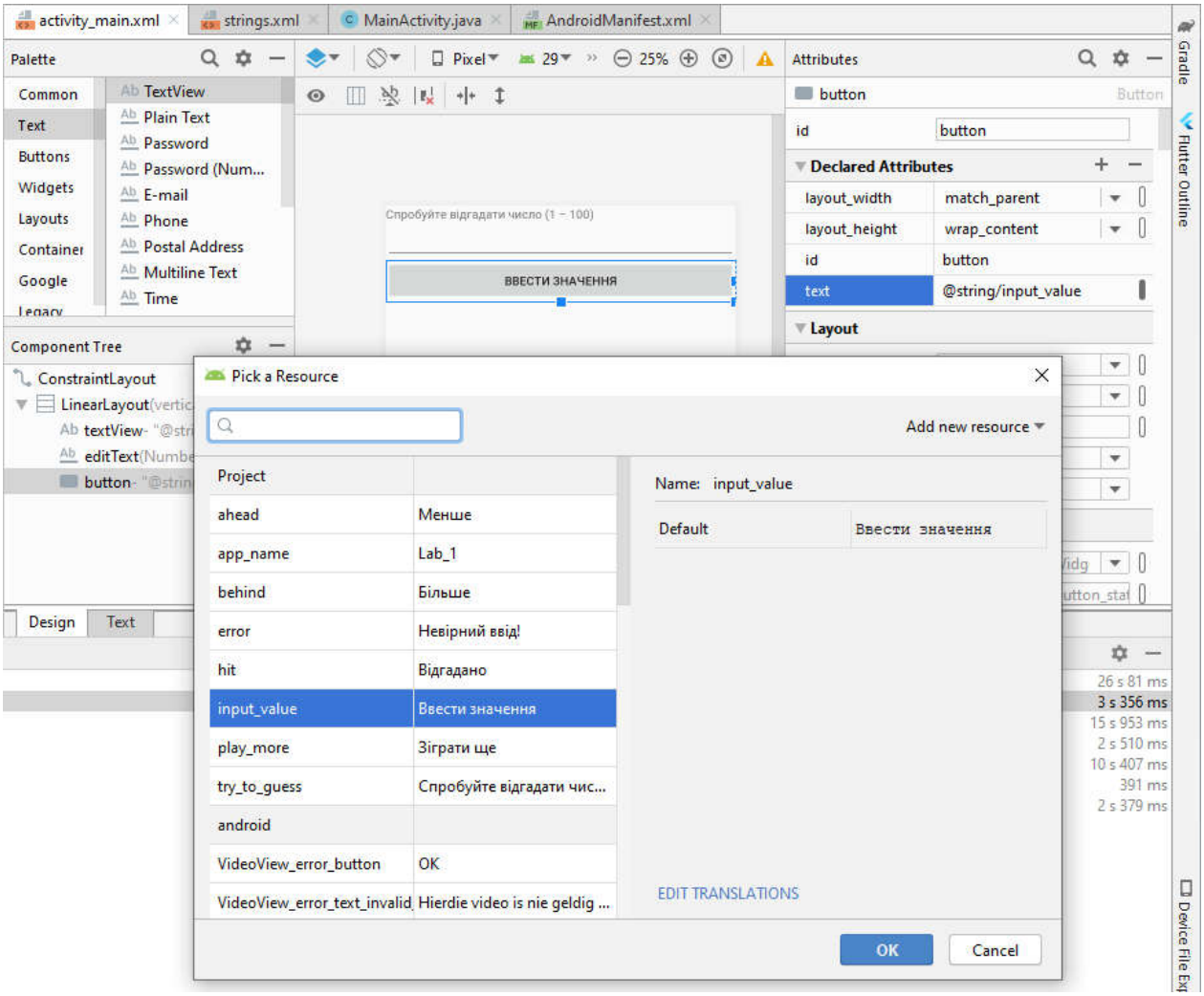

Рис.1.12 - Налаштування тексту для кнопки button

Настав час згадати про віртуальний пристрій, якщо додаток працює, вже можна запустити проект і подивитися, як додаток буде виглядати на екрані пристрою, а виглядати він може як показано на рис. 1.13.

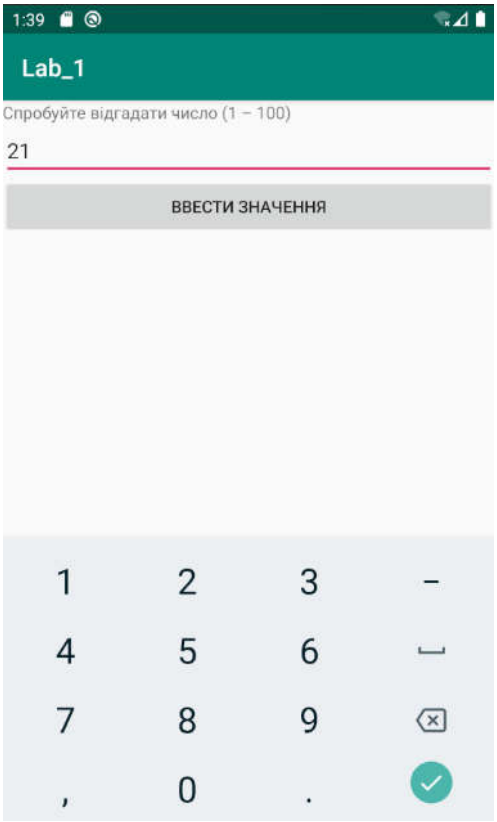

Рис. 1.13 - Запуск програми на реальному пристрої

Додаток виглядає досить просто, але ми на багато і не розраховували. Головне, що нас цікавить, це наявність всіх елементів на екрані, вірний текст в кожному елементі, де він передбачений і можливість вводити числа в полі введення. На рис. 1.13 видно, що всі вимоги виконані. Додаток є, його можна запустити на віртуальному чи реальному пристрої, але він нічого не робить. Наступним кроком буде реалізація логіки додатку, тобто обробка події натискання на кнопку, як було прописано в завданні.

1.3. Реалізація логіки додатку

Приступимо безпосередньо до програмування, працювати будемо з файлом java/com.example.lab_1/MainActivity.java. Знайдемо цей файл в Project Explorer див. Рис. 1.14, відкриємо і почнемо редагувати.

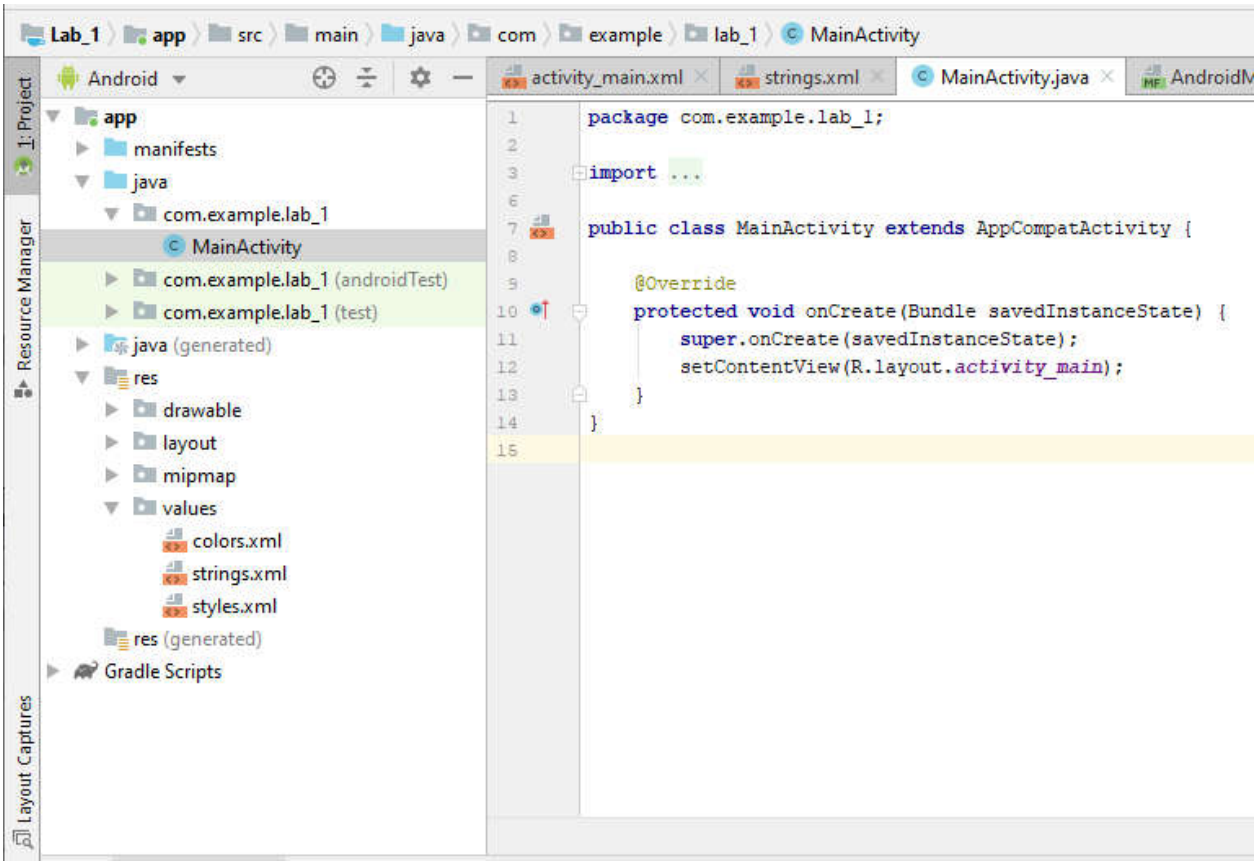

Рис. 1.14 - Файл MainActivity.java

Можна помітити, що клас MainActivity є спадкоємцем класу Activity і в ньому вже реалізований метод onCreate(), який запускається при початковому створенні активності, нам буде потрібно його доповнити, але про це трохи пізніше.

Ми припускаємо програмно змінювати інформацію в поле TextView, отримувати значення з поля EditText і обробляти події натискання на кнопку Button, тому необхідно оголосити відповідні змінні, як поля класу MainActivity:

- TextView tvInfo;
- EditText etInput;
- Button bControl;

Як можна побачити на рис.1.15, дані класи виділені червоним кольором, це тому що їх немає в проекті і необхідно додати відповідні бібліотеки для їх використання, проте якщо навести курсор мишки на назву класу виводиться підказка, що можна поставити курсор на цю назву і відповідна бібліотека буде додана до проекту, це стосується стандартних бібліотек, при використанні сторонніх чи власних бібліоте їх необхідно додавати вручну.

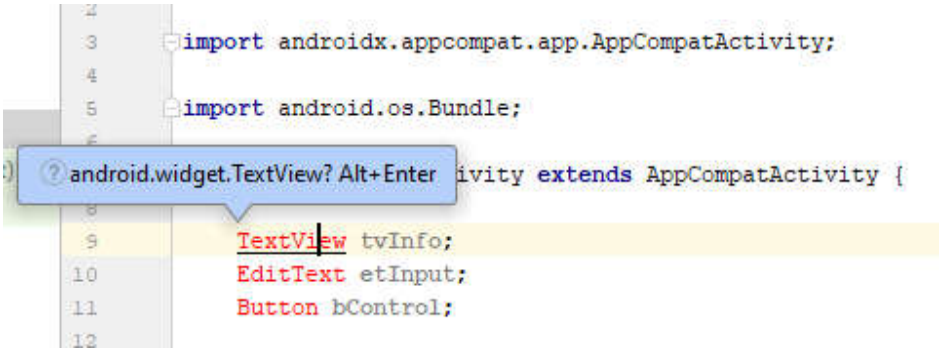

Рис.1.15 – Помилки при відсутності необхідних класів чи бібліотек

Тепер необхідно пов'язати ці змінні з елементами інтерфейсу, вже доданими нами в activity_main.xml, зробити це необхідно в методі **onCreate()**, а для отримання вже створеного елементу інтерфейсу скористаємося методом **findViewById()**. Отже в метод **onCreate()** додамо наступні рядки:

```
tvInfo = (TextView)findViewById(R.id.textView1);
etInput = (EditText)findViewById(R.id.editText1);
bControl = (Button)findViewById(R.id.button1);
```

Метод **findViewById()** повертає об'єкт класу View, який є спільним предком для всіх компонентів для користувацького інтерфейсу, для того щоб уникнути можливих помилок в дужках перед викликом методу вказуємо до якого конкретно компонента необхідно звузити можливості об'єкту **View**.

Прийшов час виконати обробку натискання на кнопку. Повернемося до файлу activity_main.xml в графічний режим редагування, виберемо елемент **Button** і на вкладці з властивостями елементу знайдемо властивість **OnClick** і запишемо в нього **onClick** - ім'я методу, який буде обробляти натискання на кнопку. Як це виглядає показує рис. 1.16.

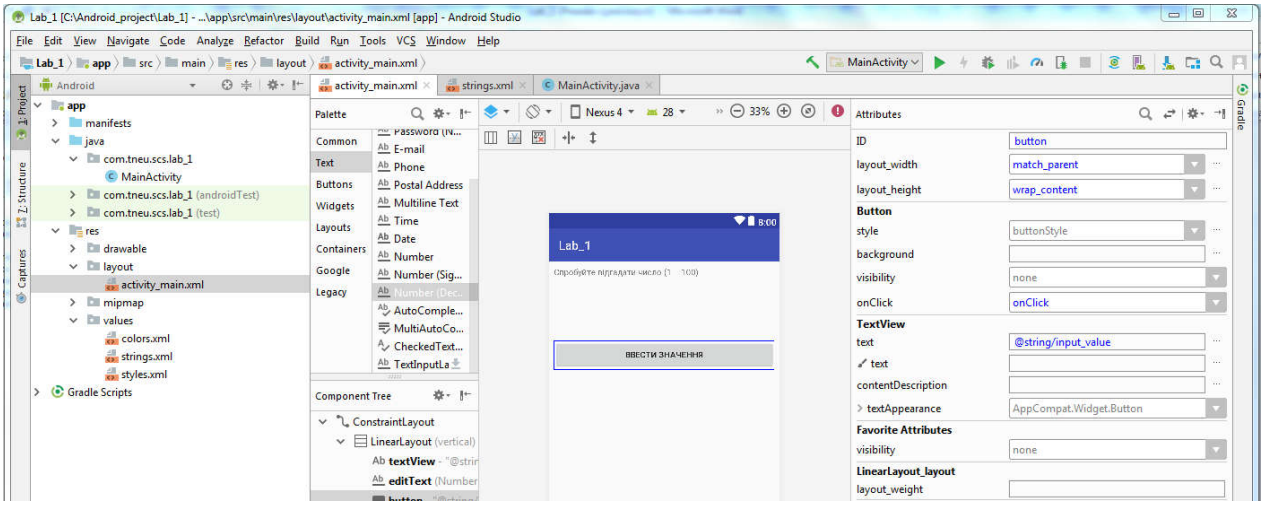

Рис. 1.16 - Налаштування властивості OnClick для кнопки

Ці ж дії можна виконати в файлі activity_main.xml, досить дописати виділений рядок:

```
<Button
android:id="@+id/button"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:onClick="onClick"
android:text="@string/input_value" />
```

Для налаштування властивостей елементів інтерфейсу потрібно використовувати будь-який спосіб: графічний або редагування XML файлу.

Повернемося в файл MainActivity.java, в клас активності необхідно додати метод:

public void onClick(View v){...}

Ім'я методу не обов'язково має бути onClick(), головне, щоб воно збігалося з ім'ям, зазначеним у властивості OnClick. У цьому методі і буде відбуватися все наше програмування в цій лабораторній роботі.

Нам будуть потрібні дві змінні:

• цілочисельна для зберігання задуманого числа (випадкове число від 1 до 100);

• логічна для зберігання стану закінчена гра чи ні.

Обидві ці змінні слід оголосити як поля класу активності, початкові значення привласнити в методі onCreate.

Отримати цілочисельне значення з поля вводу, можна за допомогою наступної конструкції:

Integer.parseInt(etInput.getText().toString())

змінити значення тексту в інформаційному полі можна за допомогою наступної конструкції:

tvInfo.setText(getResources().getString(R.string.ahead));

в даному випадку в інформаційному полі з'явиться значення строкового ресурсу з ім'ям ahead.

Залишилося реалізувати логіку додатку в методі onClick(). Пропонуємо написати код цього методу самостійно, для контролю в додатку запропонований лістинг, який містить один з варіантів коду описаного додатку.

Завдання для самостійної роботи:

Зрозуміло, що запропонована в додатку реалізація не ідеальна, вимагає доопрацювань:

- Що станеться, якщо кнопка буде натиснута до введення будь-якого числа? Швидше за все додаток буде зупинено, необхідно якось відстежувати цей варіант розвитку подій і адекватно реагувати. Пропонуємо попрацювати самостійно над цією проблемою.
- Що станеться, якщо користувач введе число менше нуля або більше 100? Швидше за все додаток обробить цей ввід, але було б краще, якби з'явилося повідомлення про те, що введене число не відповідає умовам завдання.
- Як завершити додаток? І чи треба це робити?

Робота з емулятором

При виконанні даної роботи ми використовували емулятор для перевірки працездатності програми. Розглянемо деякі можливості, що полегшують і прискорюють взаємодію з віртуальним пристроєм.

По-перше, існує набір корисних комбінацій клавіш для управління віртуальним пристроєм:

- Alt + Enter - розгортає емулятор до розмірів екрану;
- Ctrl + F11 - змінює орієнтацію емулятора з портретної на альбомну і назад;
- F8 - вмикає / вимикає мережу.

Повний список комбінацій клавіш для роботи з емулятором можна знайти за посиланням:

http://developer.android.com/tools/help/emulator.html.

По-друге, хто б не працював з емулятором, той на собі відчув наскільки терплячим треба бути, щоб взаємодіяти з ним, так повільно він працює. Існує рішення для прискорення роботи емулятора Android і цим рішенням є Intel Hardware Accelerated Execution Manager (Intel® HAXM).

Intel Hardware Accelerated Execution Manager (Intel® HAXM) - це додаток з підтримкою апаратної віртуалізації (гіпервізор), який використовує технологію віртуалізації Intel для прискорення емуляції додатків Android на комп'ютері для розробки. (http://software.intel.com/ru-ru/android/articles/intel-hardware-accelerated-execution-manager)

Intel HAXM здатний прискорити роботу емулятора для x86 пристроїв. При цьому емулятор буде працювати зі швидкістю, наближеною до швидкості роботи реального пристрою, що допоможе скоротити час на запуск і налагодження програми. Детально ознайомитися з установкою Intel HAXM і налаштуванням емулятора на роботу з прискорювачем можна в статті за посиланням: : http://habrahabr.ru/company/intel/blog/146114/

По-третє, не варто переривати процес запуску віртуального пристрою, наберіться терпіння, на старих комп'ютерах перший запуск AVD може зайняти до 10 хвилин, на сучасних - від однієї до трьох хвилин. Після того, як створили і запустили віртуальний пристрій є сенс залишити його відкритим, при всіх наступних запусках програми буде використовуватися вже відкритий віртуальний пристрій, що дозволить заощадити час.

Висновок

У лабораторній роботі розглянуто процес розробки простого додатку переднього плану. Описано створення активності, налаштування інтерфейсу і реалізація логіки додатку. Інших компонентів в додатку не передбачено. У наступних роботах будуть розглядатися додатки, що містять кілька активностей. Змішані програми, що працюють на передньому плані і при цьому підтримують сервіси, що працюють у фоновому режимі.

Лістинг 1.1

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.*;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {
    TextView tvInfo;
    EditText etInput;
    Button bControl;

    int guess;
    boolean gameFinished;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tvInfo = (TextView)findViewById(R.id.textView);
        etInput = (EditText)findViewById(R.id.editText);
        bControl = (Button)findViewById(R.id.button);

        guess = (int)(Math.random()*100);
        gameFinished = false;
    }

    public void onClick(View v){
        if (!gameFinished){
            int inp=Integer.parseInt(etInput.getText().toString());

            if (inp > guess)
                tvInfo.setText(getResources().getString(R.string.ahead));
            if (inp < guess)
                tvInfo.setText(getResources().getString(R.string.behind));
            if (inp == guess)
            {
                tvInfo.setText(getResources().getString(R.string.hit));
                bControl.setText(getResources().getString(R.string.play_more));
                gameFinished = true;
            }
        }
        else
        {
            guess = (int)(Math.random()*100);
            bControl.setText(getResources().getString(R.string.input_value));
            tvInfo.setText(getResources().getString(R.string.try_to_guess));
            gameFinished = false;
        }
        etInput.setText("");
    }

}
```

Лабораторна робота №2

Тема: Основи розробки інтерфейсів мобільних додатків
Мета: Вивчення основ розробки інтерфейсів мобільних додатків

Хід виконання роботи

Лабораторна робота присвячена розробці інтерфейсів мобільних додатків. Робота містить докладний опис побудови гармонійного зрозумілого для користувача інтерфейсу для головної активності додатку і опис основних елементів інтерфейсу. Лабораторна робота допоможе вибрати концепцію свого застосунку і почати розробку його інтерфейсу.

Більш детальну інформацію про розробку інтерфейсів мобільних додатків під Android можна дізнатися на сайті https://developer.android.com/design/index.html

2.1. Створення прототипу інтерфейсу

Розглянемо приклад розробки інтерфейсу додатку, який шукає в мережі Інтернет зображення за запитом користувача, дозволяє оцінювати їх, завантажувати, і відвідувати інтернет-сторінки сайтів, на яких було знайдено зображення.

Виглядати головне вікно буде приблизно так:

Рис. 2.1 – Інтерфейс головної активності

На ньому присутні: поле вводу тексту для запиту користувача і кнопка, що починає пошук зображень. Внизу екрана дві кнопки: "like" і "dislike", з їх допомогою користувач зможе оцінити зображення. Після того, як користувач зробить оцінку зображення, поточне зображення закривається і завантажується наступне.

Отже, почнемо з створення нового проекту. Назвемо його "RatingImages" ("Рейтинг зображень").

На даному етапі вивчення програмування під Android не обов'язково міняти іконку запуску, але ця лабораторна робота спрямована на розробку дизайну інтерфейсу, і тому, приступимо.

Створіть іконку на свій смак (File -> New -> Image Asset).

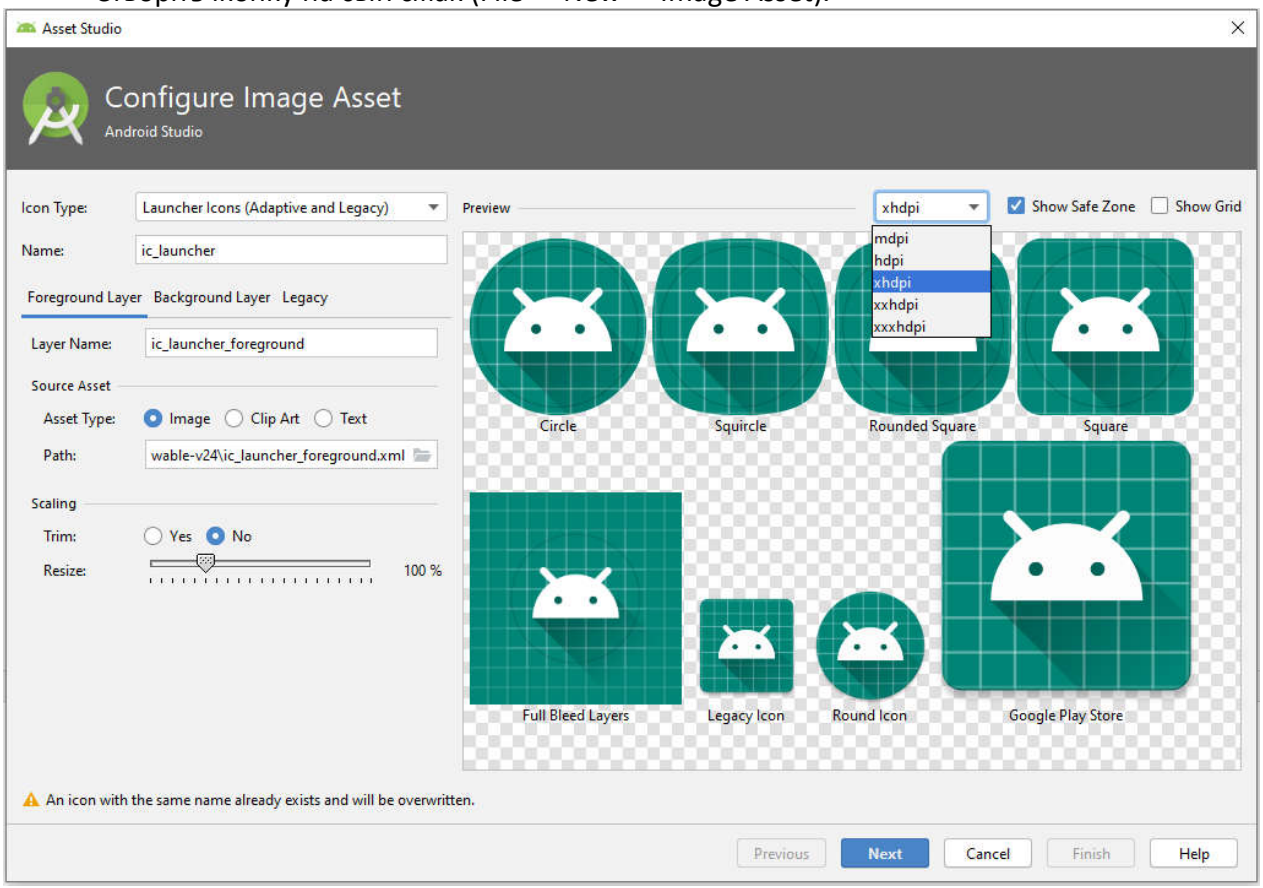

Рис. 2.2 - Створення іконки

Після створення проекту відкрийте activity_main.xml з каталогу res/layout/.

Коли ви відкриєте файл activity_main.xml, ви побачите графічний редактор макету. Завдяки цьому редактору створення інтерфейсів стало ще цікавіше, оскільки додати елемент на форму можна за допомогою перетягування мишею, до того ж, завдяки графічному редактору, не обов'язково запускати емулятор, щоб побачити результат своєї праці.

Тепер клацніть по вкладці activity_main.xml в нижній частині екрана. Відкрився XML-редактор коду. Цей спосіб редагування стандартний, але всі зміни, що вносяться до цього документу, можна так само відчути візуально, перейшовши на графічний редактор.

Повернемося на вкладку з графічним редактором. По-перше, підготуємо документ до початку роботи, для цього видаліть <TextView>.

Результат виглядає так:

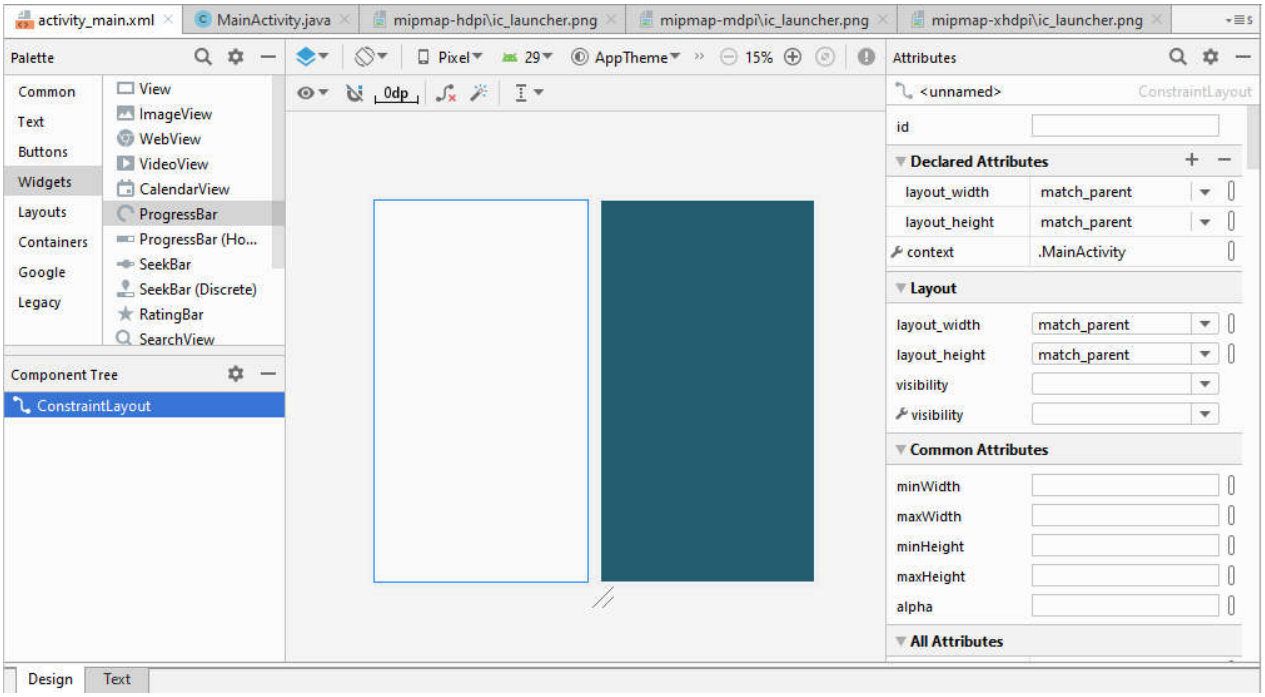

Рис. 2.3 - Проект, готовий до початку розробки

На робочій області екрана залишився один елемент. Це макет <ConstraintLayout>. У ньому позиція дочірніх елементів може бути описана по відношенню один до одного або до батька чи матері.

Два атрибути, ширина і висота (android: layout_width і android: layout_height), потрібні для всіх елементів для того, щоб вказати їх розмір.

Так як <ConstraintLayout> - це корінь в макеті, то потрібно, щоб він заповнював всю область екрану. Це досягається за допомогою встановлення параметру "match_parent" для ширини і висоти. Це значення вказує, що ширина і висота елементу буде дорівнює ширині і висоті батьківського елементу.

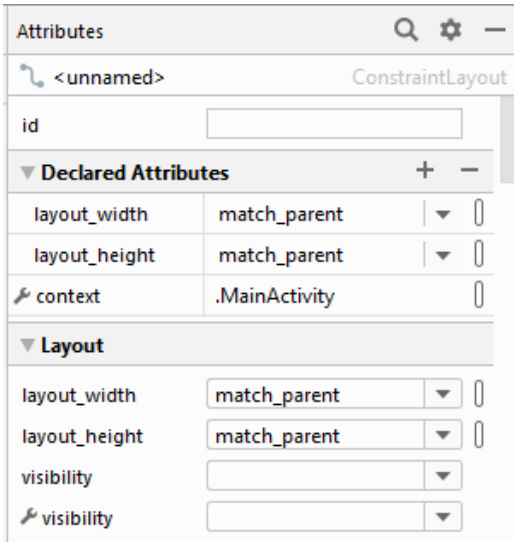

Рис. 2.4 - Властивості <ConstraintLayout> елементу

2.2 Додавання текстового поля

Для початку додайте елемент <LinearLayout> з горизонтальною орієнтацією в <ConstraintLayout>, і вкажіть для ширини і висоти параметр "wrap_content". Тепер, для створення користувацького редагованого текстового поля, додайте елемент <EditText> з параметром "wrap_content" для ширини і висоти в <LinearLayout>.

Зараз повинно вийти приблизно наступне:

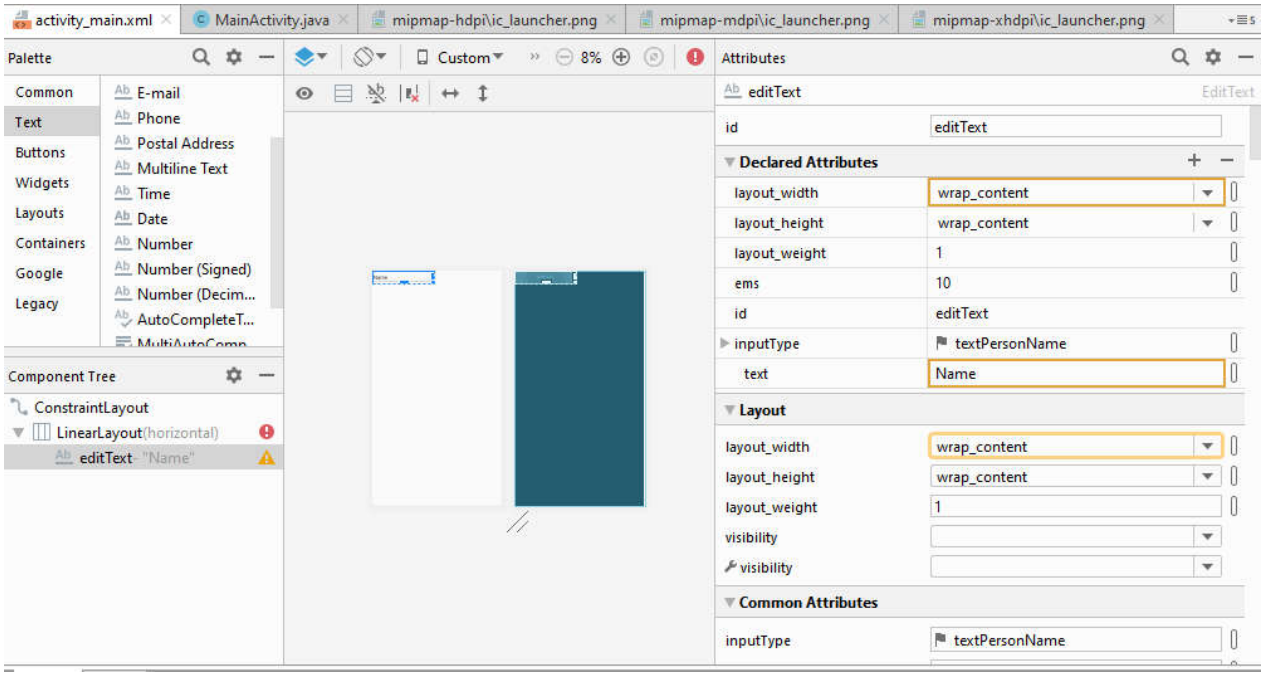

Рис. 2.5 - Додавання текстового поля

Можливо, з'явився жовтий застережливий знак, але зараз це не важливо, з часом він зникне. Наявність таких попереджень ніяк не впливає на компілювання проекту.

Тепер переходимо до налаштування доданих нами елементів.

Для багатьох елементів потрібно призначати id, він забезпечує унікальний ідентифікатор, який можна використовувати як посилання на об'єкт з коду вашої програми для управління ним.

При вказівці id, знак (@) потрібно в тому випадку, якщо ви маєте на увазі будь-який ресурс об'єкту з XML-файлу. За ним слідують тип ресурсу (в даному випадку ID), коса риска (слеш) і ім'я ресурсу (editText1).

Знак плюс (+) перед типом ресурсів необхідний тільки тоді, коли ви вперше визначаєте ідентифікатор ресурсу.

По суті, id, який створюється автоматично, вже унікальний, але грамотніше перейменовувати id відповідно до призначенням елемента.

Задамо id для текстового поля. Для цього прямо в коді рядок android: id = "@+id/editText1" замінюємо на android: id = "@+id/edit_message", тиснемо CTRL+S і відкриваємо графічний редактор. Якщо все добре, то у властивостях текстового поля в графі id буде наступне:

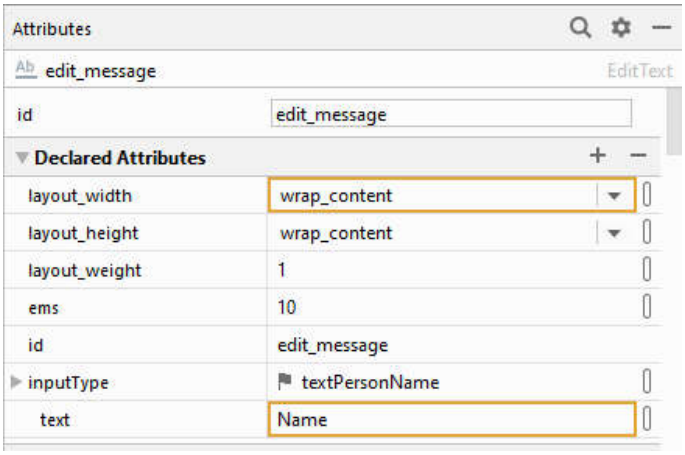

Рис. 2.6 - Ідентифікатор текстового поля

Додамо в код ще два рядки:

android: ems = "10" - задає відповідності для симетричного відображення шрифтів, android: hint = "@string/edit_message" - зміст тестового поля "за замовчуванням", тобто поки користувач не почав вводити в поле текст. Замість того, щоб використовувати просто слово (наприклад android: hint = "message"), що вкрай не зручно при зміні основної мови додатку, використовується посилання на значення, яке зберігається в файлі strings.xml. Оскільки це відноситься до конкретного ресурсу (а не тільки до id), знак плюс не потрібен.

Для того, щоб посилання на ресурс почало працювати, потрібно цей ресурс створити.

Відкрийте файл res/values/strings.xml. Очевидно, що його теж можна редагувати двома способами: графічно і вручну.

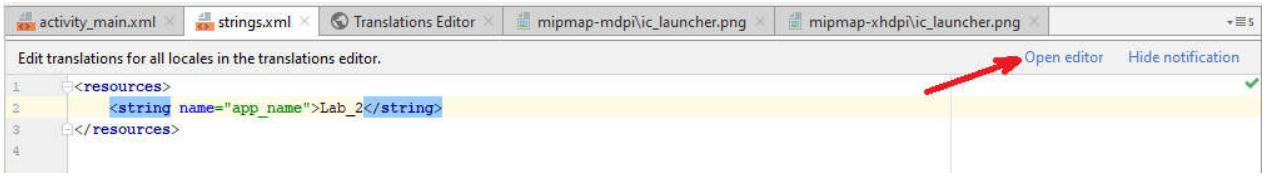

Рис. 2.7 – Редагування стрічкових ресурсів

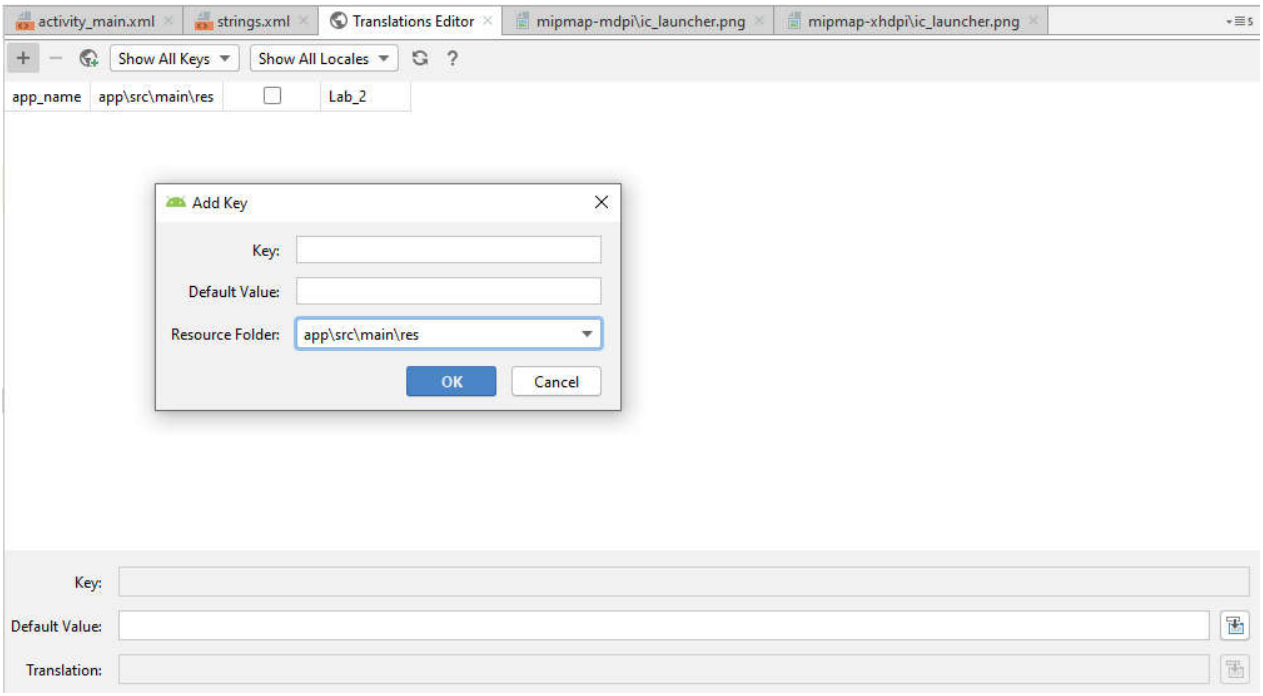

Рис. 2.8 - Редагування файлу ресурсів графічним способом

Тепер додайте <Button> в макет після елемента <EditText>:

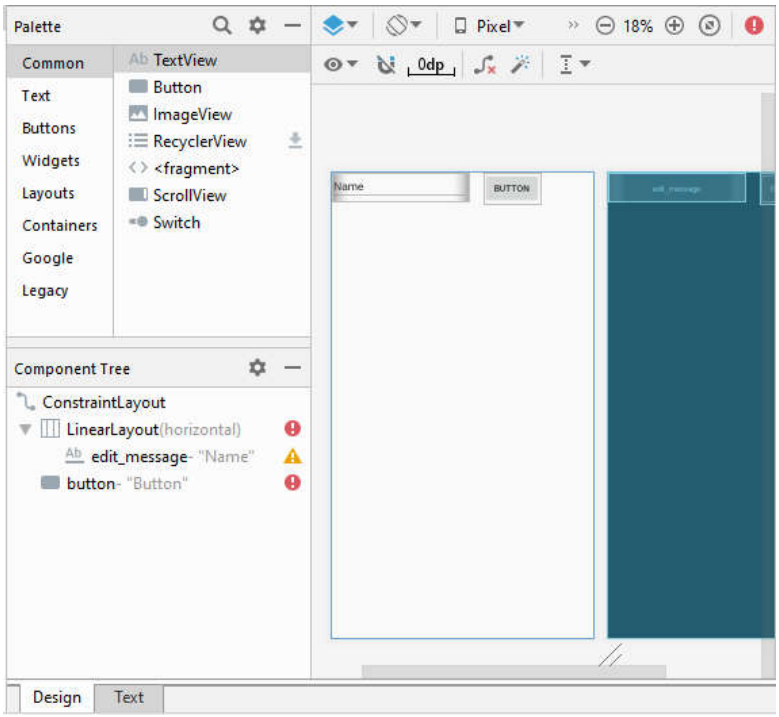

Рис. 2.9 - Нова кнопка

Щоб кнопка трансформувалася відповідно до тексту кнопки, ширина і висота повинні бути встановлені у "wrap_content".

Тепер поміняємо напис на кнопці на "Go!" за допомогою посилання на ресурс в XML-коді головної активності і додавання одного ресурсу в файл strings.xml:

```
    <EditText
android:id="@+id/editText"
android:layout_width="50dp"
android:layout_height="35dp"
android:layout_marginEnd="8dp"
android:layout_marginRight="8dp"
android:layout_marginBottom="36dp"
android:autofillHints=""
android:ellipsize="end"
android:inputType="text"
android:maxLines="1"
android:paddingEnd="35dp"
android:paddingRight="35dp"
android:scrollHorizontally="true"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent" />
```

Рис. 2.10 - Редактор XML-коду файлу strings.xml

Збережіть.

В графічному редакторі головної активності будуть зміни:

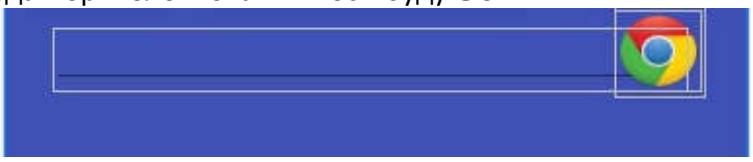

Рис. 2.11 - Кнопка прийняла нову форму, відповідно до напису на ній

Тепер, коли ми помістили два головних представлення на елемент <LinearLayout>, настав час додати ще два параметри для цього елемента.

Йдеться про "збільшенні" правого і лівого країв лейаута до правого і лівого країв <RelativeLayout> елемента відповідно. А зробити це простіше простого - просто потягніть мишкою один край до іншого!

Рис. 2.12 - "Приріст" лівого краю

Зелені стрілки по краях дають зрозуміти, до якого елементу вдалося причепити край:

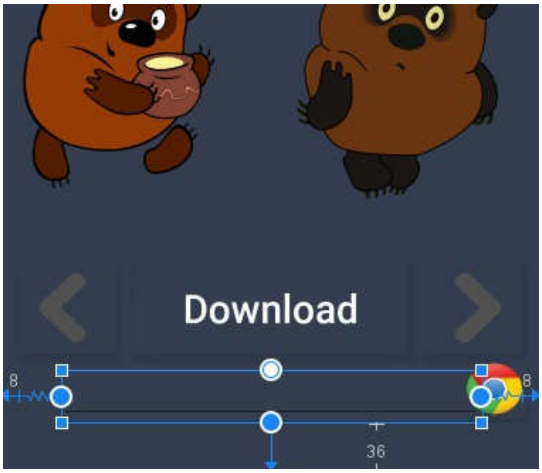

Рис. 2.13 - <LinearLayout> після вирівнювання

І, звичайно, це відобразиться у властивостях:

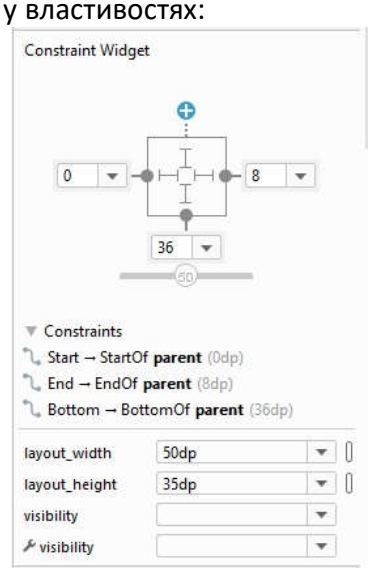

Рис. 2.14 - Властивості <EditText>

2.3 Зміна фону

Йдемо далі - спробуємо поміняти фон.

Щоб змінити колір фону на чорний, потрібно в XML-коді головної активності написати один рядок в блоці <RelativeLayout> елемента:

android: background = "# 000".

Збережіть і перевірте результат, відкривши графічний редактор.

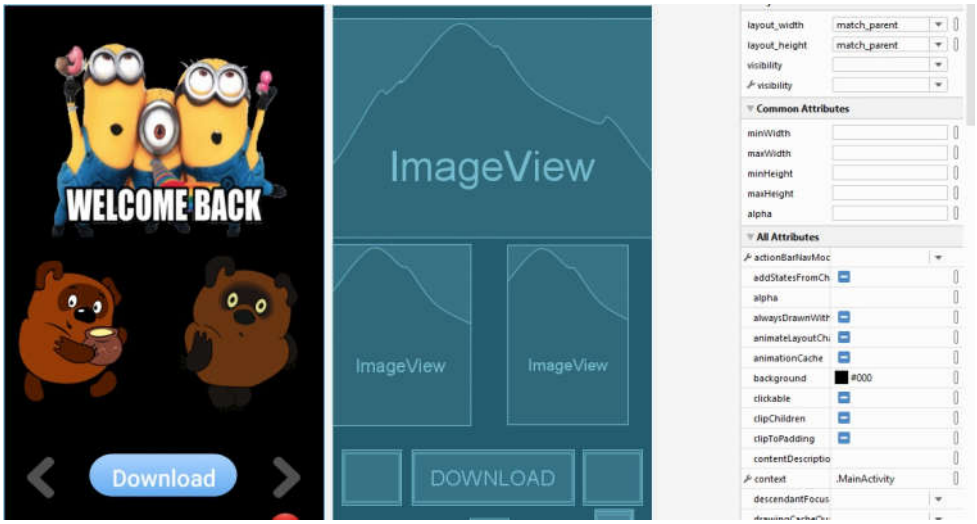

Рис. 2.15 - Фон став чорним

Красиво, але нудно. Як зробити фон ще цікавішим? Помістити на нього малюнок! Для цього спочатку в папці res/ створимо папку drawable/

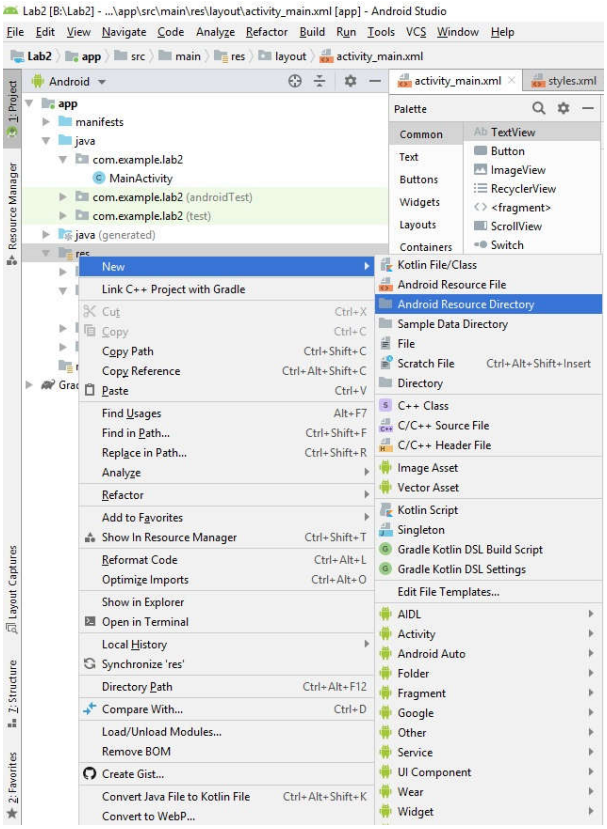

Рис. 2.16 - Створення папки

Після того, як папка створена, потрібно покласти в цю папку зображення - картинка називається got.png:

Після цього в папці drawable/ потрібно створити файл background.xml, важливо при створенні вибрати параметр bitmap.

Як тільки новий файл відкрився, пропишемо в нього одну сходинку, з вказівкою на те, звідки і який файл використовувати:

<? Xml version = "1.0" encoding = "utf-8"?>
<Bitmap xmlns: android = "http://schemas.android.com/apk/res/android"
android: src = "@ drawable/got">
</ Bitmap>

Повернемося в редактор XML-коду, туди, де прописували колір фону.

Замість рядка android: background = "# 000000" напишемо посилання на XML-файл android: background = "@drawable/background".

Зберігаємо і бачимо результат:

Рис. 2.17 - Новий фон

Безсумнівно, фон виглядає добре, але очевидно, що кнопка і поле введення просто загубилися, а це значить, що для цього додатку такий фон не підходить. Можна продовжити підбирати зображення на фон, але краще створити черепичну заливку невеликим зображенням. На цьому сайті можна знайти візерунок на будь-який смак!

Коли ви вибрали візерунок і завантажили його, скопіюйте зображення в папку drawable/.

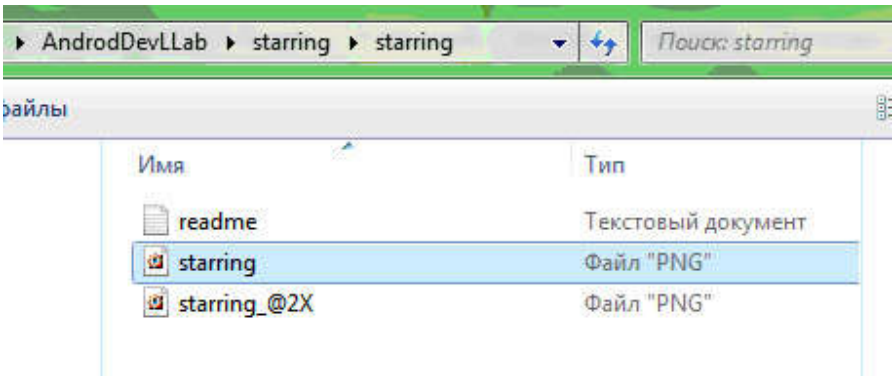

Рис. 2.18 - Копіювання зображення

Тепер трохи змінимо файл background.xml.
По-перше, потрібно змінити ім'я зображення зі старого на нове.
android: src = "@drawable/starring"

По-друге, додамо таку стрічку:
android: tileMode = "repeat"

Зберігаємо.

Атрибут android: tileMode задає тип заповнення, в даному випадку просте повторення вихідного зображення. Крім repeat можливі варіанти clamp і mirror. Пам'ятайте, що даний прийом можна застосувати тільки до bitmap, до фігур, створеним за допомогою XML, застосувати цю операцію не можна.

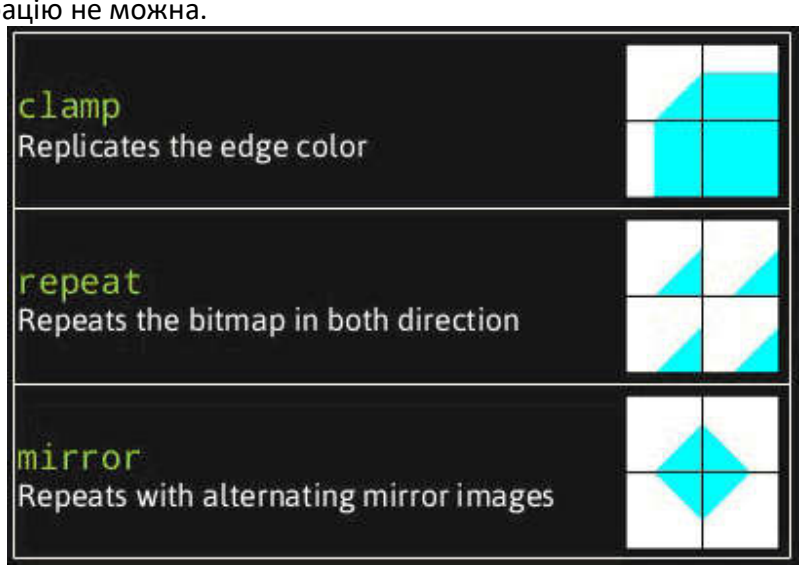

Рис. 2.19 - Варіанти заповнення

Настав час подивитися, що з цього вийшло:

Рис. 2.20 - Фон з зірочок

У <RelativeLayout> варто додавати android: background тільки в тому випадку, якщо ви хочете нерухомий фон, а в <ScrollView> щоб фон прокручувався разом з контентом.

Якщо ви вибрали темний фон, то варто поміняти колір тексту, що вводиться в поле введення, наприклад на білий.

Для цього в блок <EditText> додамо рядок

android: textColor = "# ffffff"

2.4 Область перегляду зображень

Тепер займемося створенням області відображення зображень, які користувач буде оцінювати.

Додайте "рамку" <FrameLayout> в <RelativeLayout>:

Рис. 2.21 - Додавання «рамки»

Зазначимо цьому елементу ширину, висоту і id. Попередження повинне пропасти.

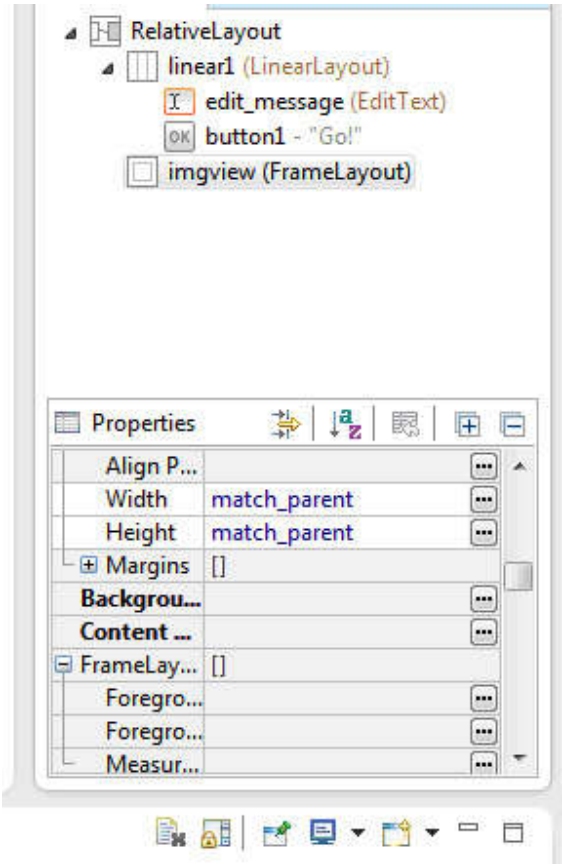

Рис. 2.22 - Властивості "рамки"

Зараз FrameLayout "заїжджає" на поле введення і кнопку. Щоб виправити це, потрібно задати для "рамки" параметр, який буде стежити, щоб "рамка" перебувала нижче, ніж поле введення і кнопка.

Виберіть зі списку Outline "рамку" і кліком правої кнопки миші викличте контекстне меню: Other Properties -> Layout Parametrs -> Layout Below

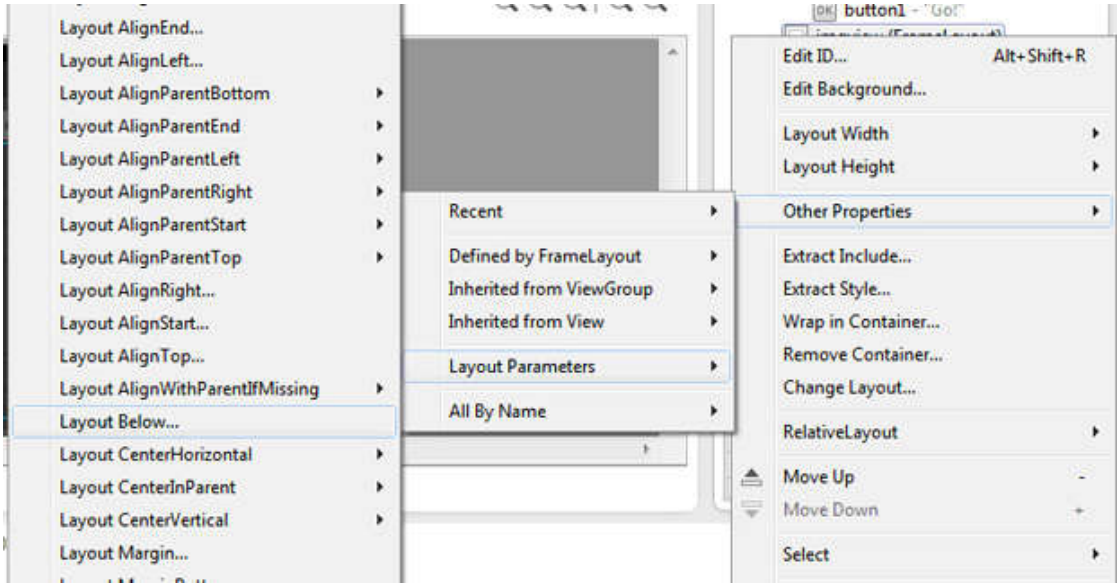

Рис. 2.23 - Контекстне меню "рамки"

З'явиться вікно. У ньому вибираємо зі списку "ID" ім'я лейаута, на якому знаходяться поле вводу і кнопка:

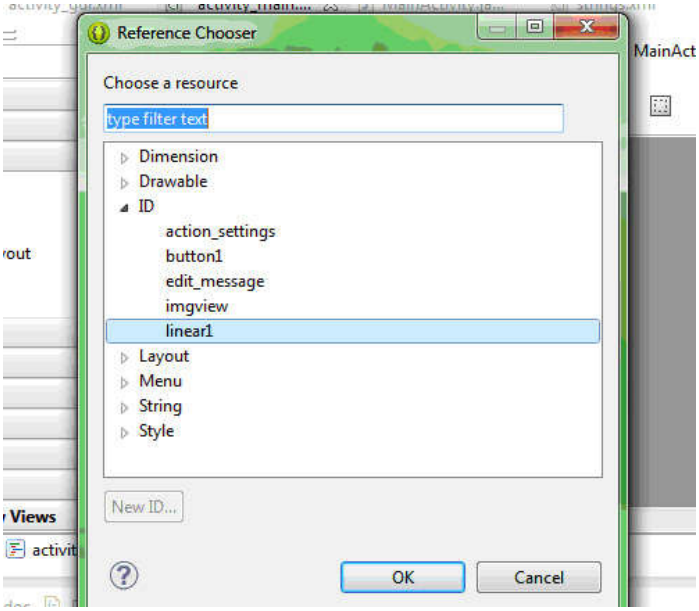

Рис. 2.24 - Вибір елементу, нижче котрого повинна бути "рамка"

Натискаємо **ОК** і "рамка" прийме вид:

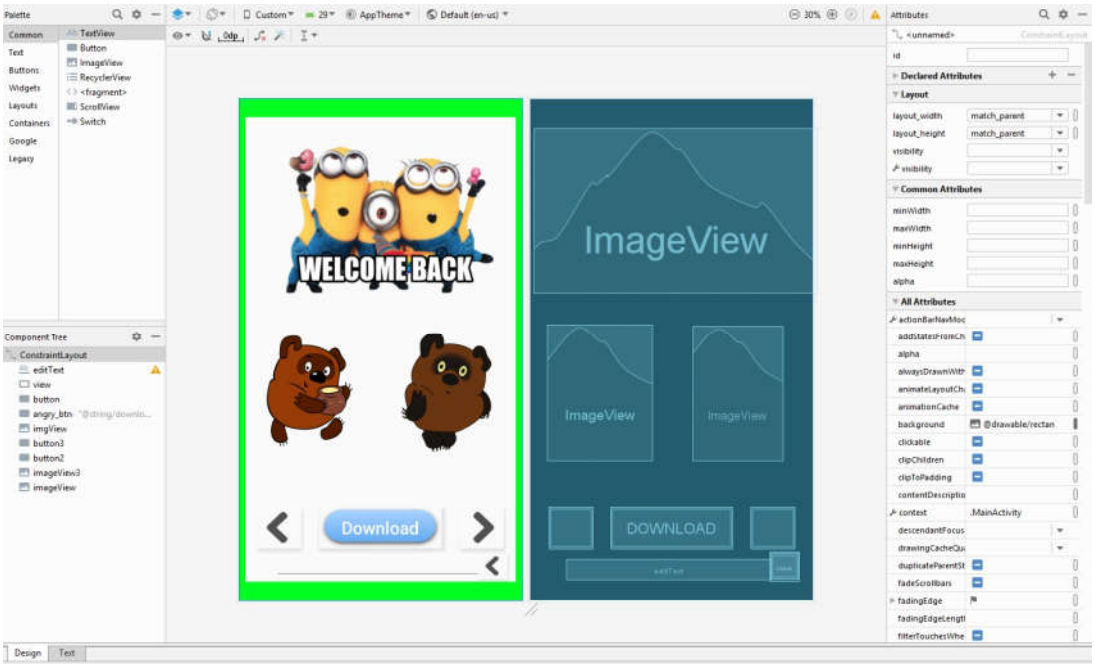

Рис. 2.25 - Нове розташування "рамки"

Тепер для наочності помістимо туди зображення.
Спочатку помістимо саме зображення в папку res/drawable/ і оновимо її.
Це потрібно для того, щоб нові дані завантажилися в проект.

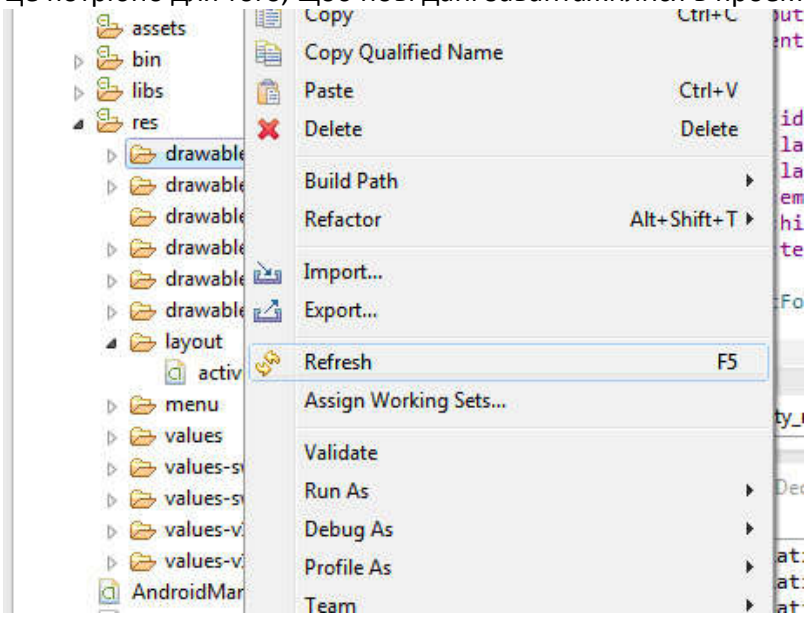

Рис. 2.26 - Обновлення папки

Тепер помістимо всередину "рамки" <ImageView>

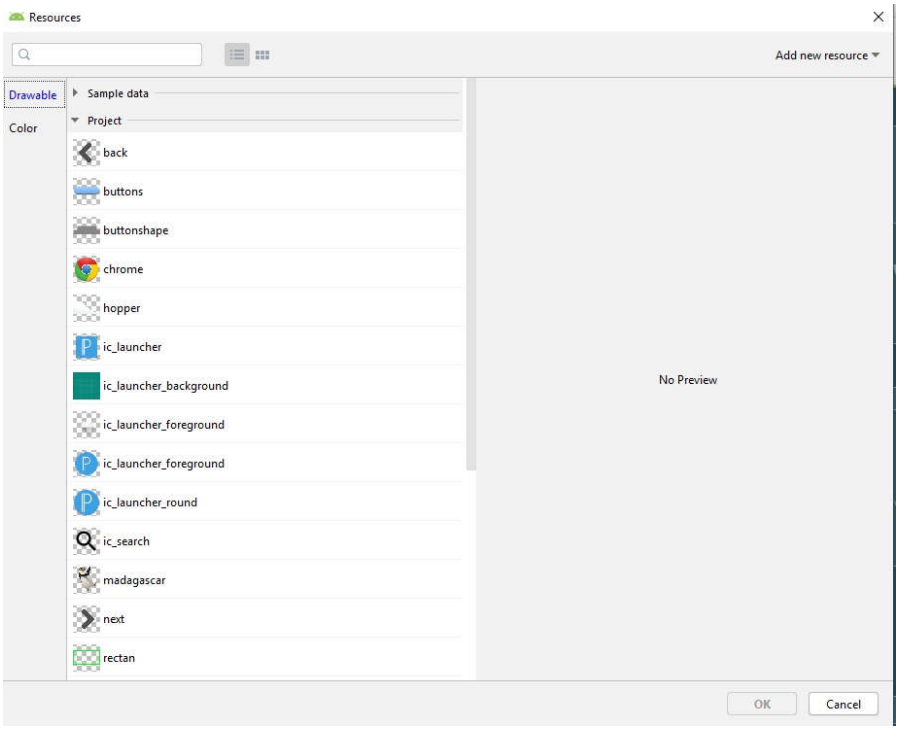

Рис. 2.27 - Вибір зображення <ImageView>

У вікні вибираємо потрібне нам зображення і тиснемо OK

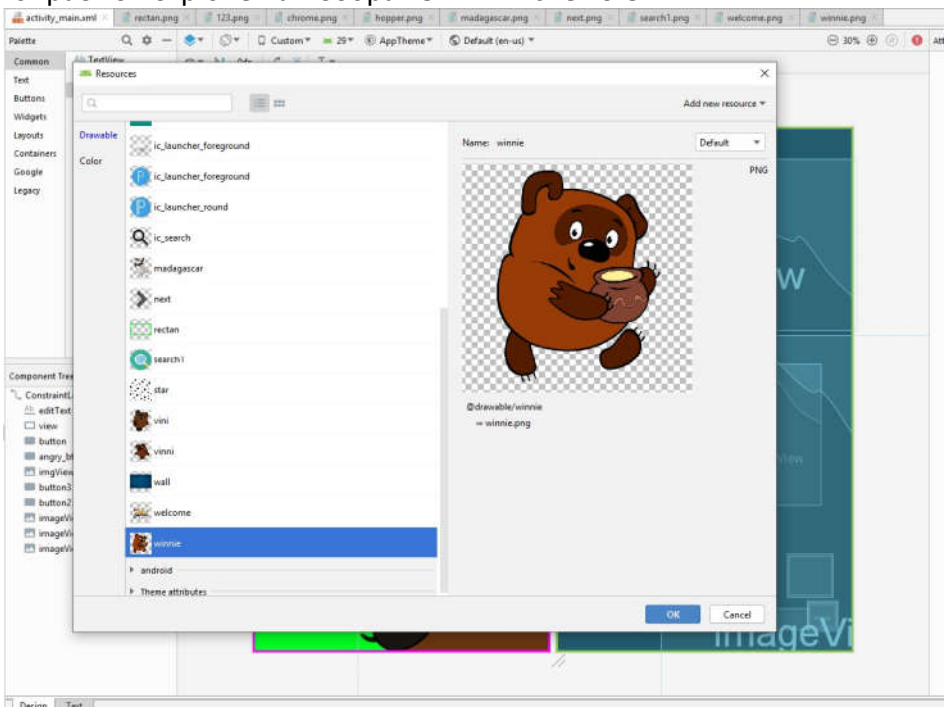

Рис. 2.28 - Вибір файла

Зображення повинно появитися на активності:

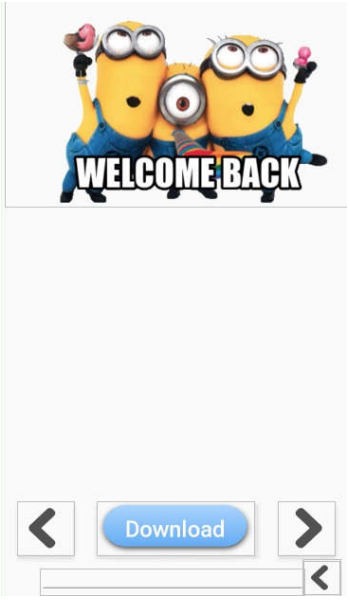

Рис. 2.29 - Візуалізація зображення

Разом із зображенням з'явився новий попереджувальний знак.

В описі попередження сказано: "[Accessibility] Missing contentDescription attribute on image". Давайте розберемося, що це означає.

Атрибут android: contentDescription використовується в програмах, які підтримують більш доступного режиму для людей з вадами зору та слуху. Тобто, текст, прописаний в цьому атрибуті, що не буде показаний на екрані, але при включеній в "Спеціальних можливостях" послуги "звукові підказки" цей текст буде проговорюватися, коли користувач переходить до цього елементу управління.

Цей атрибут можна задати для ImageButton, ImageView і CheckBox, і задавати його чи ні - справа кожного.

Властивості у зображення повинні бути наступними:

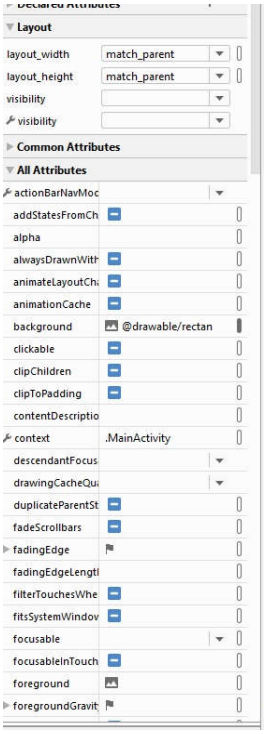

Рис. 2.30 - властивості зображення

2.5 Кнопки "like" і "dislike"

Прийшов час створити кнопки оцінювання.

Для цього додайте на форму <RelativeLayout>, задайте для нього ширину і висоту wrap_content, і вкажіть id.

Знову в папку res/drawable/ потрібно додати файли. Знайдіть зображення "Палець вгору" і "Палець вниз", і помістіть їх в цю папку, після чого поновіть її.

Зображення та інші корисні файли можна завантажити тут.

Додайте <ImageButton>, виберіть зображення "Палець вгору", і перемістіть <RelativeLayout> в таке становище:

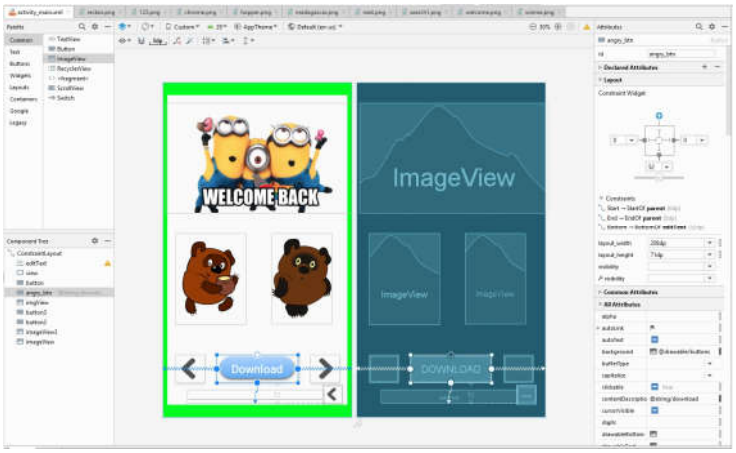

Рис. 2.31 - Нова кнопка

Вкажіть "рамці" бути вище, ніж поле з кнопкою оцінки:

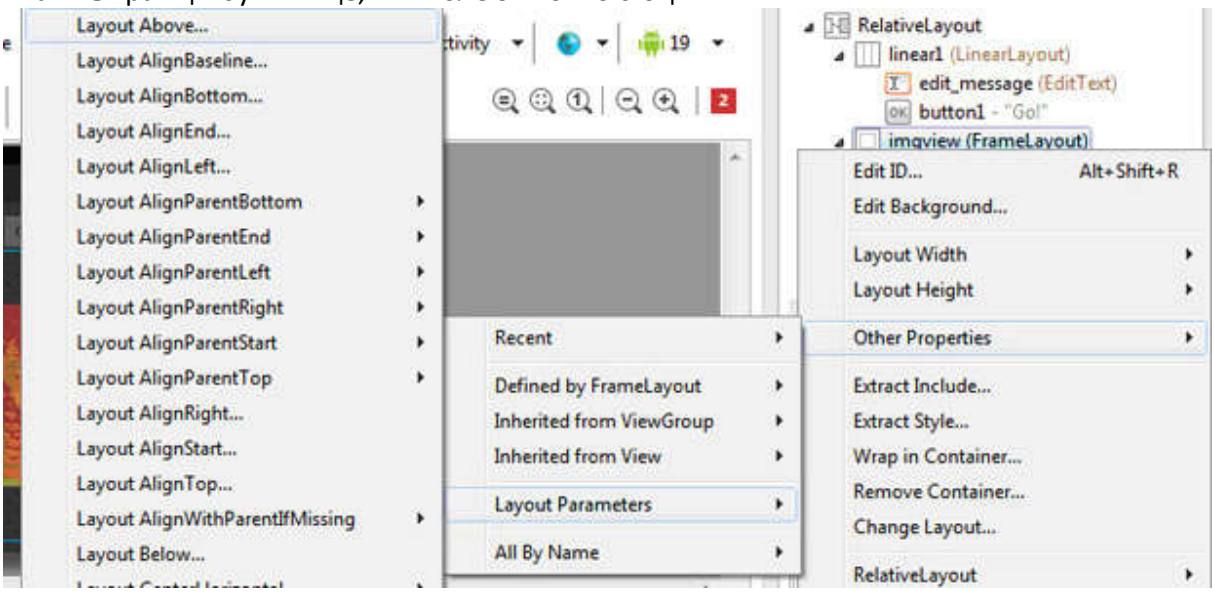

Рис. 2.32 - Контекстне меню "рамки"

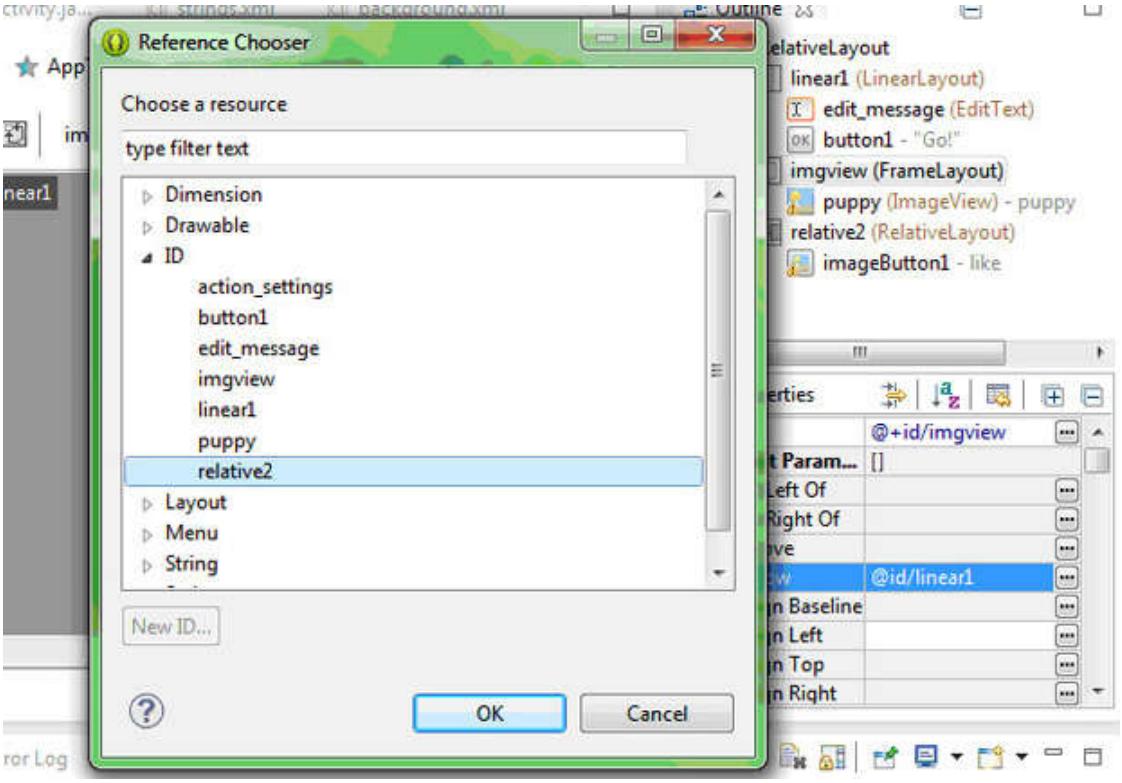

Рис. 2.33 - Вибір елемента, вище якого повинна бути "рамка"

В результаті має вийти наступне:

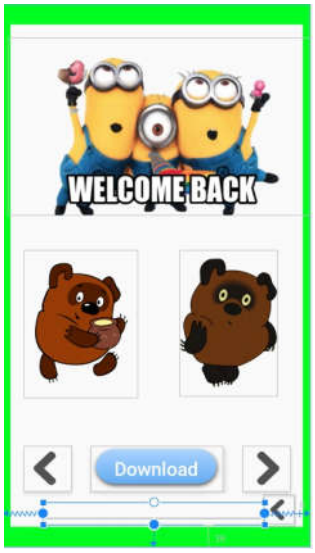

Рис. 2.34 - Нове розташування елементів

Додайте ще одну кнопку - кнопку "Палець вниз". Вона "наклалася" на першу кнопку. Щоб це виправити, виконайте з <RelativeLayout> те ж саме, що і з <LinearLayout>: розтягніть елемент вліво і вправо, до отримання такого результату:

Рис. 2.35 - Вирівнювання області з кнопками

Тепер розставте кнопки по краях так, щоб вони "прикріпилися" до країв.

Рис. 2.36 - вирівнювання кнопок

Готово! Тепер можна запустити емулятор і подивитися, що вийшло.

Рис. 2.37 - Головна активність

Завдання

1. Створіть в додатку на головній активності рядок, в якій буде виводитися адреса сайту, звідки завантажено зображення, і кнопку для переходу на цей сайт.

2. Подумайте над інтерфейсом власного додатку. Які елементи управління воно може містити? Спробуйте втілити його інтерфейс на практиці.

Лістинг 2.1

```
<RelativeLayout
  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:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity"
  android:background="@drawable/background">

  <LinearLayout
          android:id="@+id/linear1"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_alignParentLeft="true"
          android:layout_alignParentRight="true"
          android:orientation="horizontal" >
          <EditText
                  android:id="@+id/edit_message"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:ems="10"
                  android:hint="@string/edit_message"
                  android:textColor="#ffffff" >
                  <requestFocus />
          </EditText>

          <Button
                  android:id="@+id/button1"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="@string/button_send"
          />
  </LinearLayout>

  <FrameLayout
          android:id="@+id/imgview"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:layout_above="@+id/relative2"
          android:layout_below="@id/linear1" >
          <ImageView
                  android:id="@+id/puppy"
                  android:layout_width="wrap_content"
                  android:layout_height="match_parent"
```

```
                android:src="@drawable/puppy"
        />
  </FrameLayout>

  <RelativeLayout
        android:id="@+id/relative2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/imgview"
        android:layout_alignParentBottom="true"
        android:layout_alignRight="@+id/imgview" >
        <ImageButton
                android:id="@+id/imageButton2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_alignParentTop="true"
                android:src="@drawable/dislike"
        />
        <ImageButton
                android:id="@+id/imageButton1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:layout_alignParentTop="true"
                android:src="@drawable/like"
        />
  </RelativeLayout>

</RelativeLayout>
```

Лістинг 2.2

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

        <string name="app_name">RatingImages</string>
        <string name="action_settings">Settings</string>
        <string name="hello_world">Hello world!</string>
        <string name="edit_message">Enter your text here</string>
        <string name="button_send">Go!</string>

</resources>
```

Лістинг 2.3

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

<bitmap
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:src="@drawable/starring"
  android:tileMode="repeat" >

</bitmap>
```

Лабораторна робота №3

Тема: Створення багатовіконного додатку

Мета: Навчитися створювати додатки, що складаються з декількох активностей, і діалогових вікон, а також познайомитися з елементами тач-інтерфейсу.

Хід виконання роботи

Дана лабораторна робота присвячена різним способам розробки багатоекранних додатків. Пропонується розробити три додатки, кожен з яких присвячено окремому аспекту цієї тематики. По завершенню роботи над додатками можна розробити власні програми з використанням вивчених компонентів і технологій.

3.1 Створення багатоекранного додатку зі списком

1. Створіть проект MultiScreen. У java-файлі замініть Activity на ListActivity. Клас ListActivity розроблений таким чином, що на екрані є тільки прокручуваний список і йому не потрібна додаткова розмітка. Тому файл activity_main.xml можна видалити. Також слід видалити наступний рядок з класу MultiScreen:

```
setContentView (R.layout.activity_main);
```

так як layout-файл ми тільки що видалили.

2. Тепер нам потрібен посередник, який зв'яже список і назви елементів цього списку. Для цього в Android є інтерфейс Adapter. Нам буде потрібний спадкоємець цього класу ArrayAdapter:

```
new ArrayAdapter (Context context, int textViewResourceId,
String[] objects)
```

В якості аргументів ArrayAdapter потребують поточний контекст, ідентифікатор ресурсу з розміткою для кожного рядка, масив рядків.

Ми можемо вказати в якості поточного контексту ListActivity (можна використовувати ключове слово this), готовий системний ідентифікатор ресурсу (android.R.layout.simple_list_item_1) і створений масив рядків (String[] islands = { "Канари", "Курили", "Мальдіви "," Філіпіни "};). А виглядати це буде так:

```
private ArrayAdapter <String> adapter;
adapter = new ArrayAdapter <String> (this,
android.R.layout.simple_list_item_1, islands);
```

Зверніть увагу на рядок android.R.layout.simple_list_item_1. У ньому вже міститься необхідна розмітка для елементів списку. Якщо вас не влаштовує системна розмітка, то можете створити власну розмітку в xml-файлі і підключити її.

Залишилося тільки підключити адаптер:

```
setListAdapter (adapter);
```

3. Тепер нам потрібно підключити обробку натискання. Для цього необхідно знати, на який пункт списку здійснюється натискання. Існує спеціальний інтерфейс OnItemClickListener з методом onItemClick(). Підключаємо обробник: getListView(). SetOnItemClickListener(itemListener); Набираємо все в тому ж onCreate

```
OnItemClickListener itemListener = new OnItemClickListener() {
}
```

Якщо підкреслити перше слово, то імпортуємо потрібний клас. Далі підведемо курсор до другого слова new OnItemClickListener. Нам буде запропоновано додати метод (Add unimplemented method). Погоджуємося і отримуємо заготовку:

```
OnItemClickListener itemListener = new OnItemClickListener() {

@Override
public void onItemClick (AdapterView <?> arg0, View arg1, int arg2, long arg3){
// TODO Auto-generated method stub
      }
};
```

Міняємо імена змінних arg на більш звичні і зрозумілі

```
public void onItemClick (AdapterView <?> parent,
View v, int position, long id);
```

4. Залишилося описати, що буде відбуватися при натисканні на елемент.

За нашим задумом кожен елемент буде відкривати нове вікно з відповідним вмістом.

Для початку слід створити ще 4 класи: Canary, Curyly, Maldivy, Philipiny, і 4 xml-оболонки до них. Можна просто створити і копіювати по одному файлу в обох директоріях, змінюючи тільки назву і вміст.

Наприклад, створимо файли Canary.java і canary.xml типу Activity. Зверніть увагу, що як тільки ми створили ще один java-файл, потрібно записати його в файл маніфесту, інакше програма не буде бачити цей новий клас. Файл AndtoidManifest.xml знаходиться відразу під папкою res. Додайте код з ім'ям класу в тегах <activity> </ activity>, відразу під головною активністю.

```
<activity android: name = "com.mypackage.multiscreen.Canary"
     android: label = "@ string/can">
</activity>
```

Рядок can потрібно створювати в файлі strings.xml, також як і інші рядки.

Перейдемо в файл canary.xml і створимо на екрані TextView. Екрани будуть відрізнятися один від одного картинками. Візьміть 4 будь-які картинки з вашого комп'ютера (можете також знайти їх в інтернеті) і перетягніть з провідника Windows в папку res\drawable. Тепер ви можете помістити елемент ImageView на екран і, вибравши потрібну картинку з ресурсів проекту, підключити її.

Решта екрани створюються аналогічно.

5. Тепер перейдемо до головного класу і опишемо обробку події onItemClick (). Створимо перемикач, який залежить від номеру елементу.

```
switch (position) {
        case 0:

            break;
        case 1:

            break;
        case 2:

            break;

        case 3:
```

```
                break;
        }
```

Для запуску нового екрану необхідно створити екземпляр класу Intent і вказати клас, на який будемо переходити (у нас їх 4, тому для кожного випадку вибираємо свою). Після цього викликається метод startActivity (), який і запускає новий екран.

```
        Intent intent = new Intent (MultiScreen.MainActivity.this, Canary.class);
            startActivity (intent);
```

6. Тепер залишилося додати спливаюче вікно Toast, до яке буде показувати, який елемент ми вибрали. Цей віджет можна імпортувати так само, як і попередні. Нам буде потрібно метод makeText (), у якого є три параметра: контекст додатку, текстове повідомлення і тривалість часу показу повідомлення.

```
        Toast.makeText      (getApplicationContext(),      "Ви      вибрали"      +
parent.getItemAtPosition(position).toString(),Toast.LENGTH_SHORT).show();
```

Повні листинги файлів проекту, в яких були зроблені зміни, див. Нижче.

Лістинг 3.1 - Файл MultiScreenMainActivity.java

```
import android.os.Bundle;
import android.view.View;
import android.app.ListActivity;
import android.content.Intent;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Toast;

public class MainActivity extends ListActivity{

    String[] islands = { "Канари", "Курили", "Мальдіви", "Філіпіни"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);


        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, islands);
        setListAdapter(adapter);

        OnItemClickListener itemListener = new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position,
long id) {

                switch (position) {
                    case 0:
                        Intent intent = new Intent(MainActivity.this,
CanaryActivity.class);
                        startActivity(intent);
                        break;
                    case 1:
                        Intent intent1 = new Intent(MainActivity.this,
Curyly.class);
                        startActivity(intent1);
                        break;
```

```
                    case 2:
                        Intent intent2 = new Intent(MainActivity.this,
Maldivy.class);
                        startActivity(intent2);
                        break;

                    case 3:
                        Intent intent3 = new Intent(MainActivity.this,
Philipiny.class);
                        startActivity(intent3);
                        break;
                }
                Toast.makeText(getApplicationContext(), "Ви вибрали" +
parent.getItemAtPosition(position).toString(),
                        Toast.LENGTH_SHORT).show();
            }
        };
        getListView().setOnItemClickListener(itemListener);
    }
}
```

Лістинг 3.2. Файл Canary.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class CanaryActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_canary);
    }
}
```

Лістинг 3.3. Файл canary.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dip" >

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="186dp"
        android:src="@drawable/canary" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
```

```
        android:layout_centerHorizontal="true"
        android:layout_marginTop="80dp"
        android:text="@string/enjoy"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

Лістинг 3.4. Файл strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Куди б поїхати в відпустку?</string>
    <string name="action_settings">Settings</string>
    <string name="enjoy">Enjoy yourself!</string>
    <string name="can">Канари</string>
    <string name="phil">Філіпіни</string>
    <string name="cur">Курили</string>
    <string name="mal">Мальдіви</string>
</resources>
```

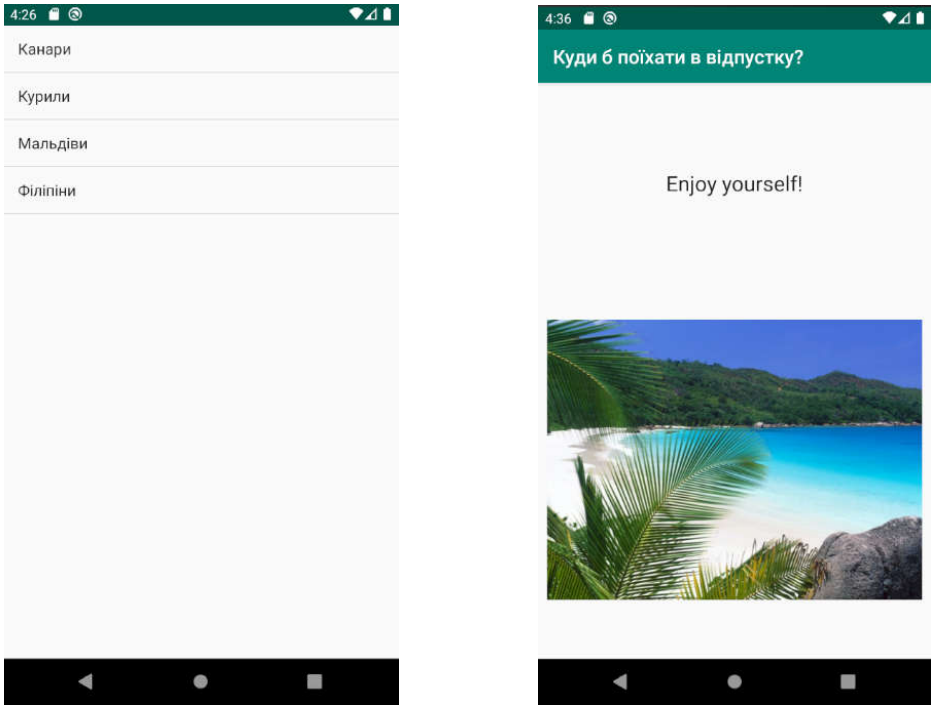

Рис. 3.1 - Додаток "Куди б поїхати у відпустку?", Запущений на пристрої

3.2 Створення діалогового вікна

1. Створіть новий проект Dialog

2. Створіть кнопку на вашій активності та назвіть її "Вибрати фон" (для цього потрібно в файлі strings створити рядок відповідного змісту). Дайте активності і кнопці id.

Лістинг 3.5 - Код файлу activity_main.xml

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

```
<Button
        android:id="@+id/background_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="174dp"
        android:text="@string/bg"/>
</RelativeLayout>
```

3. Наш додаток змінює фон на обраний. Значить, нам потрібно створити кольори і їх назви в файлі strings.xml. Як ви пам'ятаєте, цей файл знаходиться в папці values, яка в свою чергу знаходиться в папці res. Також створимо рядок messages, яка нам знадобиться для діалогового вікна.

Лістинг 3.6 - файл strings.xml

```
<resources>
        <string name = "app_name"> Dialog </string>
        <string name = "action_settings"> Settings </string>
        <string name = "bg"> Вибрати фон </string>
        <string name = "message"> Хочете поміняти фон? </string>
        <string name = "red"> Червоний </string>
        <string name = "yellow"> Жовтий </string>
        <string name = "green"> Зелений </string>
</resources>
```

Лістинг 3.7 - файл colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
    <color name = "redColor"> #FF0000 </color>
    <color name = "yellowColor"> #FFFF00 </color>
    <color name = "greenColor"> #00FF00 </color>>
</resources>
```

4. Перейдемо в файл MainActivity.java. Створіть наступні змінні:

```
private Button bgButton;
public RelativeLayout relativeLayout;
Context context;
```

Якщо компілятор підкреслює тип і повідомляє про помилку, наприклад, підкреслює Context, наведіть курсор на підкреслене слово: має з'явитися контекстне меню, яке пропонує варіанти, як можна виправити помилку. Виберете Import 'Context', щоб імпортувати бібліотеку.

5. Тепер потрібно описати, що буде відбуватися при натисканні на нашу кнопку.

Для початку зв'яжемо об'єкти з activity_main.xml і змінні в MainActivity.java через id (в onCreate):

```
bgButton = (Button) findViewById (R.id.background_button);
relativeLayout = (RelativeLayout) findViewById (R.id.relativelayout);
```

Context - це об'єкт, який надає доступ до базових функцій програми.

Додаємо в код

```
context = MainActivity.this;
```

6. Тепер потрібно додати подію onClick і навісити на кнопку OnClickListener. Додаємо в заголовок класу MainActivity implements OnClickListener. Щоб зв'язати кнопку і Listener в onCreate пишемо

```
bgButton.setOnClickListener (this);
```

Створимо тепер подію onClick

```
@Override
    public void onClick (View v) {
}
```

В цьому об'єкті створюємо власне діалог і називаємо його:

```
AlertDialog.Builder builder = new AlertDialog.Builder (this);
        builder.setTitle (R.string.message);

AlertDialog alert = builder.create ();
        alert.show ();
```

Ми будемо створювати діалогове вікно, яке надає користувачеві вибір зі списку. Для цього буде потрібно ще одна змінна, яка сформує список з наявних рядків.

```
final CharSequence[] items = {
getText     (R.string.red),     getText     (R.string.yellow),     getText
(R.string.green)
};
```

7. Сформуємо власне наш список і поставимо ще один Listener, який буде змінювати колір фону на обраний:

```
builder.setItems (items, new DialogInterface.OnClickListener () {
public void onClick (DialogInterface dialog, int item) {
  switch (item) {
  case 0: {relativeLayout.setBackgroundResource (R.color.redColor);
   break;}
  case 1: {relativeLayout.setBackgroundResource (R.color.yellowColor);
  break;}
  case 2: {relativeLayout.setBackgroundResource (R.color.greenColor);
   break;}
  }
```

8. Залишилося додати в кожен case спливаючі вікна Toast, і додаток повністю готовий!

```
Toast.makeText(context, R.string.green, Toast.LENGTH_LONG).show();
```

Лістинг 3.8. Код файлу MainActivity.java

```
package com.example.i.dialog;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.Toast;

public class Dialog extends Activity implements OnClickListener {

    private Button bgButton;
    public RelativeLayout relativeLayout;
    Context context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dialog);
        bgButton = (Button) findViewById(R.id.background_button);
        bgButton.setOnClickListener(this);
        context = Dialog.this;
        relativeLayout = (RelativeLayout)findViewById(R.id.relativelayout);
    }

    @Override
    public void onClick(View v) {

        final CharSequence[] items={getText(R.string.red),
getText(R.string.yellow), getText(R.string.green)

        };

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle(R.string.message);
        builder.setItems(items, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int item) {
                switch (item) {
                    case 0: {
relativeLayout.setBackgroundResource(R.color.redColor);
                        Toast.makeText(context, R.string.red,
Toast.LENGTH_LONG).show();

                        break;}
                    case 1:
{relativeLayout.setBackgroundResource(R.color.yellowColor);
                        Toast.makeText(context, R.string.yellow,
Toast.LENGTH_LONG).show();

                        break;}
                    case 2:
{relativeLayout.setBackgroundResource(R.color.greenColor);
                        Toast.makeText(context, R.string.green,
Toast.LENGTH_LONG).show();

                        break;}
                }
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }
}
```

Скріншоти працюючого додатку

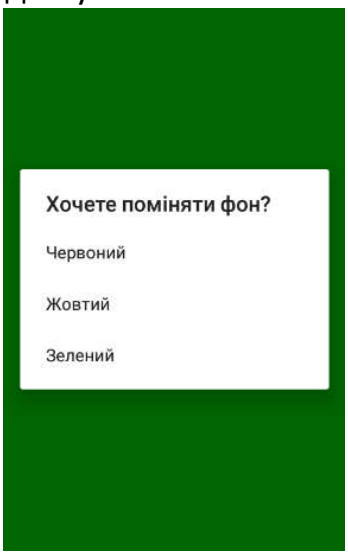

Рис. 3.2 - Додаток "Dialog", запущений на емуляторі

Завдання

Подумайте над власним додатком, що поєднує різні можливості проектування багатовіконних додатків, розглянутих вище. Створіть прототип цього додатку і налаштуйте його користувацький інтерфейс.

Лабораторна робота №4

Тема: Демонстрації розпізнавання стандартних жестів

Мета: Розробити найпростіші додатки для демонстрації розпізнавання стандартних жестів.

Завдання лабораторної роботи:

- розглянути розпізнавання всіх підтримуваних жестів;
- розглянути розпізнавання тільки частини підтримуваних жестів
- створити набір жестів;
- використати створені жести в додатку

Хід виконання роботи

Для роботи зі стандартними жестами Android надає клас GestureDetector. Цей клас містить два вкладених інтерфейсу-слухача: OnGestureListener і OnDoubleTapListener, ці інтерфейси задають методи, які відстежують стандартні жести. А також GestureDetector містить вкладений клас SimpleOnGestureListener, який містить порожні реалізації, які повертають значення false, де це необхідно, всіх методів інтерфейсів: OnGestureListener і OnDoubleTapListener.

У лабораторній роботі розглянемо дві можливості розпізнавання жестів:

- випадок розпізнавання всіх підтримуваних жестів, для цього реалізуємо в класі активності обидва інтерфейсу;
- випадок розпізнавання тільки деякого набору підтримуваних жестів, для цього в класі активності оголосимо внутрішній клас-спадкоємець класу GestureDetector.SimpleOnGestureListener.

4.1 Розпізнавання всіх підтримуваних жестів

Розробимо додаток, в якому продемонструємо розпізнавання всіх підтримуваних жестів. Додаток містить одну активність, одне інформаційне поле для виведення інформації про розпізнаний жесті. Додаток працює наступним чином: користувач виконує один з підтримуваних сенсорних жестів, в інформаційному полі відображається інформація про розпізнаний жест.

1. Створимо простий додаток і додамо на форму TextView для виведення інформації.

2. Настроїмо логіку програми. У java клас, відповідний активності внесемо такі доповнення.

o Клас активності повинен реалізовувати інтерфейси: GestureDetector.OnGestureListener і GestureDetector.OnDoubleTapListener, для цього в оголошення класу додамо конструкцію:

implements GestureDetector.OnGestureListener,
GestureDetector.OnDoubleTapListener

o Нам знадобиться екземпляр класу GestureDetectorCompat тому в якості поля класу активності оголосимо наступну змінну:

```
GestureDetectorCompat mDetector;
```

У методі onCreate() класу активності, створимо екземпляр класу GestureDetectorCompat і дамо його змінної mDetector:

```
mDetector = new GestureDetectorCompat(this, this);
```

одним з параметрів конструктора є клас, який реалізує інтерфейс GestureDetector.OnGestureListener, в нашому випадку використано слово this, т. е. параметром є сам клас активності. Цей інтерфейс повідомляє користувачів коли з'являється певне сенсорне подія.

У методі OnCreate() класу активності, наступний рядок:

```
mDetector.setOnDoubleTapListener(this);
```

встановлює слухач подій, пов'язаних з подвійним торканням, це повинен бути клас, який реалізує інтерфейс GestureDetector.OnDoubleTapListener. У нашому випадку використано слово this, тобто слухачем буде знову сам клас активності.

1. Щоб дозволити вашому об'єкту GestureDetector отримувати події, необхідно перевизначити метод onTouchEvent() для активності або елемента GUI. І передавати в екземпляр детектора всі виявлені події.

2. public boolean onTouchEvent (MotionEvent event) {
3. this.mDetector.onTouchEvent (event);
4. // Be sure to call the superclass implementation
5. return super.onTouchEvent (event);
6.}

7. Після проведеної підготовки прийшов час реалізувати всі методи, оголошені в інтерфейсах, що відповідають за прослуховування сенсорних подій.

Методи інтерфейсу GestureDetector.OnGestureListener:

onDown() - відслідковує появу дотику, тобто палець притиснутий до екрану;

onFling() - відслідковує появу жесту змахування;

onLongPress() - відслідковує утримання пальця притиснутим до екрану тривалий час;

onScroll() - відслідковує появу жесту прокрутки (пролистування);

onShowPress() - відслідковує, что виникла подія дотику і більше ніяких подій не відбувається за короткий час;

onSingleTapUp() - відслідковує появу жесту одиночного натиску(клік).

Методи інтерфейсу GestureDetector.OnDoubleTapListener:

onDoubleTap() - відслідковує появу жесту подвійного натиску("подвійний клік");

onDoubleTapEvent() - відслідковує появу події під час виконинння жесту подвійного натиску, включаючи дотик, переміщення, підйом пальця.

onSingleTapConfirmed() відслідковує появу жесту одиночного натиску (клік).

У лістингу 4.1 представлений код програми, в якому розпізнаються всі підтримувані жести, інформація про який з'явився і розпізнаний жесті видається в інформаційне поле (TextView).

Як практики пропонується відтворити цю програму і перевірити, як система розпізнає той чи інший жест. Дуже корисно для розуміння, як виконуються основні жести.

4.2 Розпізнавання тільки частини підтримуваних жестів

Розробимо додаток, в якому продемонструємо розпізнавання тільки деякої частини підтримуваних жестів за вибором програміста. Ми розглянемо розпізнавання жесту змахування (fling). Додаток містить одну активність, одне інформаційне поле для виведення інформації про розпізнаний жесті. Додаток працює наступним чином: користувач виконує один з підтримуваних сенсорних жестів, в інформаційному полі відображається інформація про розпізнаний жест.

1. Створимо простий додаток і додамо на форму TextView для виведення інформації.

2. Настроїмо логіку програми. У java клас, відповідної активності внесемо такі доповнення.

Нам знадобиться екземпляр класу GestureDetectorCompat тому в якості поля класу активності оголосимо наступну змінну:

GestureDetectorCompat mDetector;

У методі onCreate() класу активності, створимо екземпляр класу GestureDetectorCompat і дамо його змінні mDetector:

mDetector = new GestureDetectorCompat (this, new MyGestListener());

в конструкторі аргументом, що відповідає за відстеження сенсорних подій, служить екземпляр класу MyGestListener() - внутрішній клас, який є спадкоємцем класу GestureDetector.SimpleOnGestureListener.

Є сенс трохи розглянути клас GestureDetector.SimpleOnGestureListener. Цей клас реалізує інтерфейси GestureDetector.OnGestureListener і GestureDetector.OnDoubleTapListener, всі методи заявлені в інтерфейсах в цьому класі мають порожню реалізацію і ті, які повинні повертати значення, повертають false. Тому для розпізнавання якоїсь події або деякого підмножини подій досить написати реалізацію відповідних методів, в класі спадкоємця.

У лістингу 10.2 представлений код програми, в якому розпізнається тільки жест змахування, тобто реалізований метод onFling(), інформація про який з'являється і розпізнаний жест видається в інформаційне поле (TextView). В якості слухача використовується екземпляр класу MyGestListener(), який є спадкоємцем класу GestureDetector.SimpleOnGestureListener.

Отримати більше інформації про розпізнавання жестів можна за посиланням: http://developer.android.com/training/gestures/index.html

Лістинг 4.1. Розпізнавання підтримуваних жестів за допомогою реалізації інтерфейсів

```
package com.example.lab5_1_gestall;

import android.os.Bundle;
import android.app.Activity;
import android.support.v4.view.GestureDetectorCompat;
import android.view.*;
import android.widget.*;
```

```
public class MainActivity extends Activity
       implements GestureDetector.OnGestureListener,
                  GestureDetector.OnDoubleTapListener
{
  TextView tvOutput;
  GestureDetectorCompat mDetector;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    tvOutput = (TextView)findViewById(R.id.textView1);

    mDetector = new GestureDetectorCompat(this,this);
    mDetector.setOnDoubleTapListener(this);
  }

  public boolean onTouchEvent(MotionEvent event){
    this.mDetector.onTouchEvent(event);
    // Be sure to call the superclass implementation
    return super.onTouchEvent(event);
      }

  @Override
  public boolean onDown(MotionEvent event) {
      tvOutput.setText("onDown: " + event.toString());
      return false;
  }

  @Override
  public boolean onFling(MotionEvent event1, MotionEvent event2,
                 float velocityX, float velocityY) {
    tvOutput.setText("onFling: " + event1.toString()+event2.toString());

      return true;
  }

  @Override
  public void onLongPress(MotionEvent event) {
     tvOutput.setText("onLongPress: " + event.toString());
  }

  @Override
  public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,
                    float distanceY) {
     tvOutput.setText("onScroll: " + e1.toString()+e2.toString());
       return true;
  }

  @Override
  public void onShowPress(MotionEvent event) {
     tvOutput.setText("onShowPress: " + event.toString());
  }

  @Override
  public boolean onSingleTapUp(MotionEvent event) {
    tvOutput.setText("onSingleTapUp: " + event.toString());
```

```
      return true;
  }

  @Override
  public boolean onDoubleTap(MotionEvent event) {
    tvOutput.setText("onDoubleTap: " + event.toString());
      return true;
  }

  @Override
  public boolean onDoubleTapEvent(MotionEvent event) {
    tvOutput.setText("onDoubleTapEvent: " + event.toString());
      return true;
  }

  @Override
  public boolean onSingleTapConfirmed(MotionEvent event) {
     tvOutput.setText("onSingleTapConfirmed: " + event.toString());
      return true;
  }
}
```

Лістинг 4.2. Розпізнавання жестів за допомогою класу GestureDetector.SimpleOnGestureListener.

```
package com.example.lab5_1_gestsubset;

import android.os.Bundle;
import android.app.Activity;
import android.support.v4.view.*;
import android.view.*;
import android.widget.*;

public class SubsetGestActivity extends Activity {

private GestureDetectorCompat mDetector;
private TextView tvOut;

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_subset_gest);
  mDetector = new GestureDetectorCompat(this, new MyGestListener());
  tvOut = (TextView)findViewById(R.id.textView1);
}

@Override
public boolean onTouchEvent(MotionEvent event){
this.mDetector.onTouchEvent(event);
return super.onTouchEvent(event);
}

class MyGestListener extends GestureDetector.SimpleOnGestureListener {

@Override
public boolean onFling(MotionEvent event1, MotionEvent event2,
                  float velocityX, float velocityY) {
```

```
tvOut.setText("onFling: " + event1.toString()+event2.toString());
   return true;
  }
}

}
```

Лабораторна робота №5

Тема: Принципи роботи з користувацькими жестами

Мета: Розробка програми, що допомагає зрозуміти принципи роботи с жестами вводяться користувачами.

Завдання лабораторної роботи:

- створити набір жестів;
- використати створені жестів в додатку

Хід виконання роботи

Починаючи з версії 1.6, Android надає API для роботи з жестами, який розташовується в пакеті android.gesture і дозволяє зберігати, завантажувати, створювати і розпізнавати жести.

У даній роботі розглянемо процес створення набору жестів, для створення жестів будемо використовувати додаток Gesture Builder, яке встановлено на віртуальні пристрої Android (AVD), починаючи з версії 1.6.

Якщо на віртуальному пристрої Gesture Builder відсутній, можна завантажити цей додаток (або його аналог) через Play Store, якщо він підтримується.

Після цього розробимо програму, в якому передбачається розпізнавання і використання створених жестів.

5.1 Створення набору жестів

Для початку створимо новий додаток.

Далі запустимо емулятор і використовуємо додаток Gesture Builder для створення жестів "1", "2" і "S". Жест завжди пов'язаний з ім'ям, але ім'я не обов'язково повинно бути унікальним, в дійсності, для підвищення точності в розпізнаванні жесту рекомендується зберігати кілька жестів з одним і тим же ім'ям.

На рис. 5.1 можна побачити додаток Gesture Builder в роботі, щоб додати жест, необхідно натиснути на кнопку Add gesture, у вільному просторі зобразити жест (зазвичай він малюється жовтим кольором), в поле введення задати ім'я жесту. Результат послідовного додавання жестів можна побачити на рис. 5.2.

Жести зберігаються на SD карті емулятора, щоб використовувати їх в додатку необхідно імпортувати файл жестів в проект.

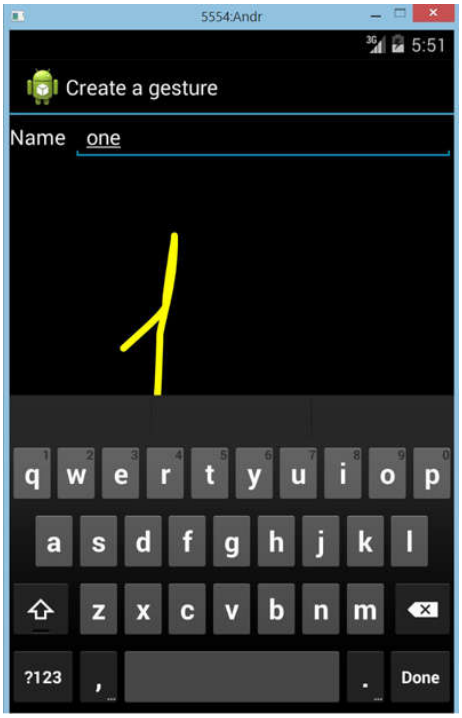

Рис. 5.1 - Створення жестів за допомогою програми Gesture Builder

Рис. 5.2. - Набір жестів, створених в додатку Gesture Builder

Коментар: Gesture Builder може повідомити, що йому нема куди зберегти жести, в цьому випадку необхідно запустити емулятор з образом SD карти.

Спочатку образ потрібно створити за допомогою утиліти mksdcard (розташована в папці <AndrSDK>/sdk/tools, де <AndrSDK> - шлях, куди встановлений Android SDK). У командному рядку напишемо:

mksdcard -l mySdCard 64M gesture.img

Наступним кроком буде створити і запустити емулятор з образом gesture.img, для цього зайдемо в папку <AndrSDK> / sdk / tools і виконаємо команду:

emulator -avd nameEmulator -sdcard gestures.img

nameEmulator - ім'я, яке присвоєно емулятора при створенні.

Цікава стаття на цю тему за посиланням: http://habrahabr.ru/post/120016/

Кожен раз, коли ми створюємо або редагуємо жести за допомогою Gesture Builder, створюється файл gestures на SD карті емулятора (залежно від версії API шлях може відрізнятися). Необхідно імпортувати цей файл в директорію res / raw /, створеного проекту, в якому плануємо використовувати жести.

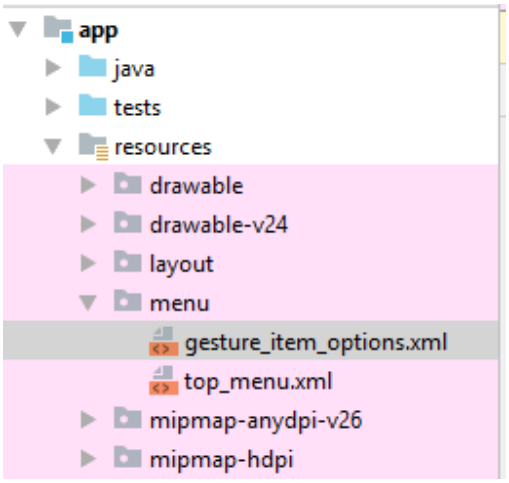

Рис. 5.3. Розташування файлу gestures

Найпростіший спосіб імпортувати жести в проект полягає використанні вкладки File Explorer в компонуванні (perspective) DDMS. Якщо вкладки File Explorer немає, можна знайти: View -> Tool Windows -> Device File Explorer. На вкладці File Explorer знайти директорію sdcard /(в нашому випадку виявилася директорія storage/sdcard/, має сенс при створенні жестів звернути увагу в яку директорію Gesture Builder їх зберігає). На рис. 5.3 показана вкладка File Explorer в компонуванні DDMS.

Щоб скопіювати файл жестів з емулятора в проект, необхідно вибрати його і натиснути кнопку "Pull a file from the device", виділену на рис. 11.3 червоним подобою окружності. Відкриється діалог з пропозицією вибрати папку, в яку необхідно скопіювати жести, тут треба знайти папку проекту, в ній папку res / raw / (якщо папки raw / немає, її необхідно створити) і натиснути кнопку Зберегти. Тепер жести є в нашому проекті і їх можна використовувати.

5.3 Використання створених жестів в додатку

Для розпізнавання жестів необхідно додати елемент GestureOverlayView в XML файл активності. І цей файл може виглядати, наприклад, як показано на рис. 11.4:

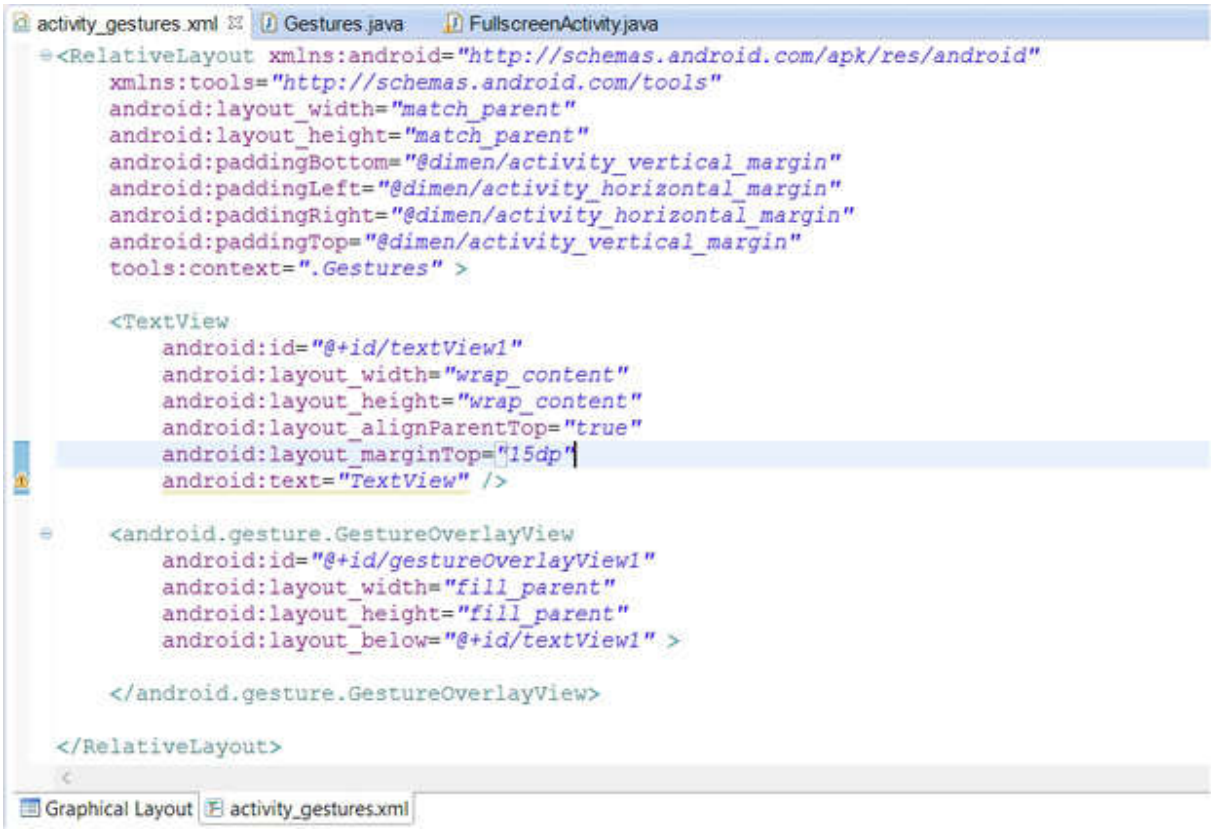

Рис. 5.4. - XML файл активності додатки, елемент GestureOverlayView звичайний компонент інтерфейсу користувача

Можна додати елемент GestureOverlayView поверх всіх компонентів, як прозорий шар, в цьому випадку XML файл активності може виглядати так, як показано на рис. 11.5.

Висновок

В якості практичного завдання пропонуємо реалізувати жестами введення чисел в додатку "Угадайка", розробленому в лабораторній роботі другої теми. Створити жести "0", "1", "2", "3", "4", "5", "6", "7", "8", "9" для введення цифр і жест "S" для зупинки введення числа. В додаток додати розпізнавання цих жестів, перетворення їх в число і порівняння отриманого числа з задуманим.

Ще варіанти для самостійної роботи:

1. розробити простий калькулятор з жестовою введенням чисел і операцій;
2. розробити блокнотик для заміток з рукописним введенням тексту.

Лістинг 5.1. Розпізнавання жестів завантажених в файл res/raw/gestures

```
package com.example.lab5_2_gestures;

import java.util.ArrayList;

import android.os.Bundle;
import android.app.Activity;
```

```
import android.gesture.Gesture;
import android.gesture.GestureLibraries;
import android.gesture.GestureLibrary;
import android.gesture.GestureOverlayView;
import android.gesture.GestureOverlayView.OnGesturePerformedListener;
import android.gesture.Prediction;
import android.view.Menu;
import android.widget.TextView;

public class Gestures extends Activity implements
OnGesturePerformedListener{

  GestureLibrary gLib;
  GestureOverlayView gestures;
  TextView tvOut;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_gestures);
    tvOut=(TextView)findViewById(R.id.textView1);
        //Загрузка жестов (gestures) из res/raw/gestures
        gLib = GestureLibraries.fromRawResource(this, R.raw.gestures);
        if (!gLib.load()) {
      //Якщо жести не завантажені, то вихід з додатку
            finish();
        }

       gestures = (GestureOverlayView)
findViewById(R.id.gestureOverlayView1);
        gestures.addOnGesturePerformedListener(this);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is
present.
    getMenuInflater().inflate(R.menu.gestures, menu);
    return true;
  }

  public void onGesturePerformed(GestureOverlayView overlay, Gesture
gesture) {
    //Створює ArrayList з завантаженими з gestures жестами
        ArrayList<Prediction> predictions = gLib.recognize(gesture);
        if (predictions.size() > 0) {
         //якщо завантажений хоча б один жест з gestures
                    Prediction prediction = predictions.get(0);
           if (prediction.score > 1.0) {
               if (prediction.name.equals("one"))
               tvOut.setText("1");
               else if (prediction.name.equals("stop"))
                 tvOut.setText("stop");
               else if (prediction.name.equals("two"))
                 tvOut.setText("2");
           }else{
             tvOut.setText("Жест невідомий");
           }
        }
```

```
    }

}
```

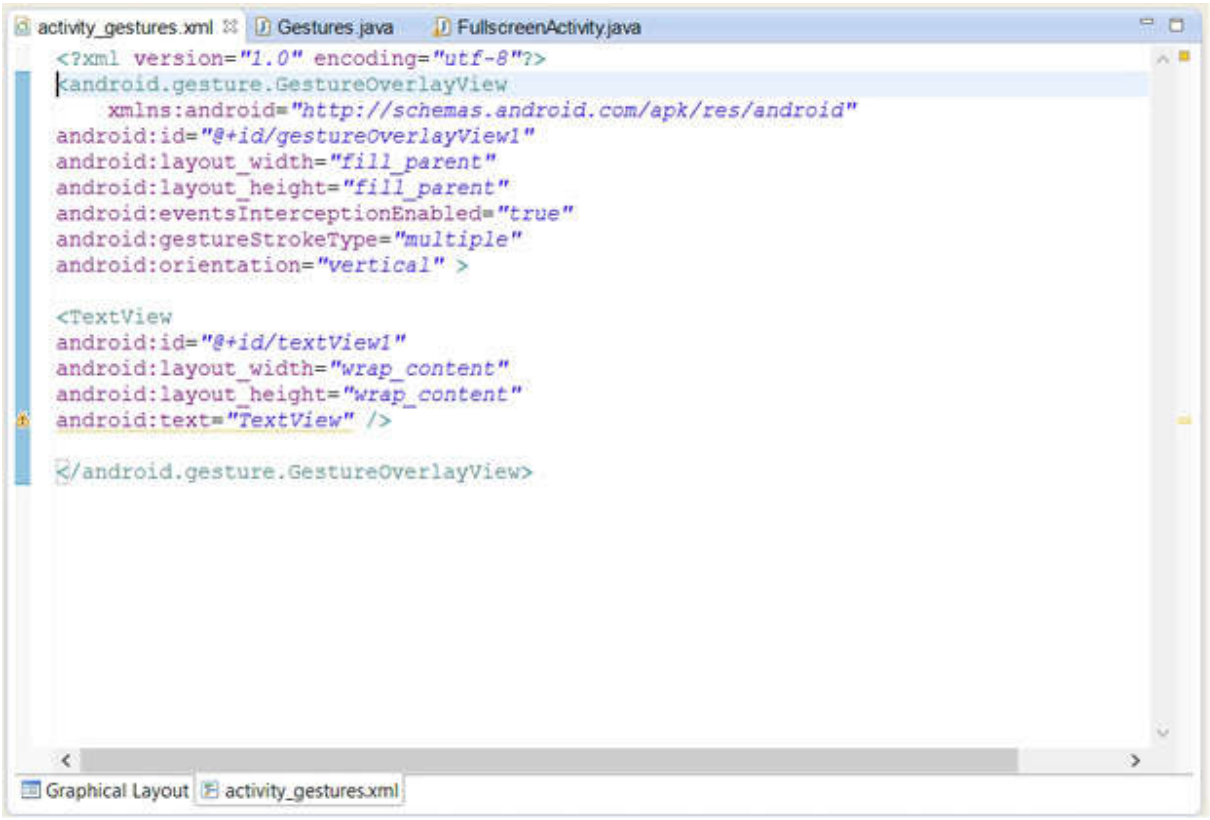

Рис. 5.5. - XML файл активності додатки, елемент GestureOverlayView поверх всіх компонентів інтерфейсу користувача

Далі необхідно обробити введення жесту користувача, порівняти із завантаженими жестами, і або визначити жест, або повідомити користувачеві, що такого жесту немає. Тепер вся робота буде виконуватися в java файлі, що описує головну (і єдину) активність додатки. Внесемо в цей клас такі доповнення:

Клас активності повинен реалізовувати інтерфейс OnGesturePerformedListener, для цього в оголошення класу додамо конструкцію:

implements OnGesturePerformedListener;

Нам знадобляться екземпляри класів GestureLibrary і GestureOverlayView, тому в якості полів класу активності оголосимо наступні змінні:

```
GestureLibrary gLib;
GestureOverlayView gestures;

У методі onCreate () виконаємо наступні дії:
gLib = GestureLibraries.fromRawResource (this, R.raw.gestures);
if (! GLib.load ()) {
finish ();
}
```

У першому рядку виконана ініціалізація змінної gLib жестами, завантаженими з файлу gestures папки res/raw/.

Оператор if виконує перевірку завантажені чи жести, якщо немає, виконується вихід з програми.

Додамо в метод onCreate() ще два рядки:

gestures = (GestureOverlayView) findViewById (R.id.gestureOverlayView1);

gestures.addOnGesturePerformedListener(this);

Для ініціалізації змінної gesture і підключення до неї слухача подій появи жесту.

І нарешті напишемо реалізацію методу OnGesturePerformed(), який і буде викликатися при появі події, відповідного якомусь жесту.

```
      public void onGesturePerformed (GestureOverlayView overlay, Gesture
gesture) {
      // Створює ArrayList с завантаженими з gestures жестами
            ArrayList <Prediction> predictions = gLib.recognize (gesture);
            if (predictions.size ()> 0) {
            // якщо завантажений хоча б один жест з gestures
                  Prediction prediction = predictions.get (0);
                  if (prediction.score> 1.0) {
                        if (prediction.name.equals ( "one"))
                        tvOut.setText ( "1");
                        else if (prediction.name.equals ( "stop"))
                              tvOut.setText ( "stop");
                        else if (prediction.name.equals ( "two"))
                              tvOut.setText ( "2");
                  } else {
                        tvOut.setText ( "Жест невідомий");
                        }
                  }
            }
```

У додатку всього лише розпізнаються жести і в інформаційне поле виводиться інформація про те, що за жест був використаний. У лістингу 5.1 представлений можливий код програми.

Лабораторна робота №6

Тема: Багатовіконні додатки.

Мета: Розробка багатовіконних додатків, що надає можливості: відтворення аудіо та відео файлів, створення і відображення фотознімків.

Завдання лабораторної роботи:

-налаштувати інтерфейс і реалізувати логіку активності для роботи з камерою;

- налаштувати інтерфейс і реалізувати логіку активності для відтворення аудіо та відео;

- налаштувати інтерфейс і реалізувати логіку активності для перегляду зображень;

-налаштувати інтерфейс і реалізувати логіку головної активності додатку.

Хід виконання роботи

Для досягнення мети, поставленої в лабораторній роботі, сформулюємо вимоги до розробленого додатку, назвемо його "Media".

Додаток надає користувачеві можливість вибору роду діяльності:

- робота з камерою для створення знімків;
- відтворення аудіо та відео;
- перегляд зображень.

У додатку передбачається реалізувати чотири активності:

- головна активність, призначена для вибору роду діяльності, містить три кнопки, натискання на кожну кнопку викликає до життя відповідну активність;
- активність для роботи з камерою і створення знімків;
- активність для відтворення аудіо та відео;
- активність для перегляду зображень.

6.1 Створення додатка

Створимо новий додаток, активність, отриману при створенні проекту, залишимо з ім'ям **MainActivity**, вона буде головною активністю додатки. Додамо в додаток ще три активності: **New-> Other ...-> Android Activity.**

У нашому додатку створені активності мають такі імена:

CameraActivity для роботи з камерою і створення знімків;

MediaActivity для відтворення відео і аудіо;

GalleryActivity для перегляду зображень.

Далі продовжимо роботу з цими активностями, налаштуємо інтерфейс і реалізуємо логіку для кожної з них. Почнемо з активностей, що відповідають за той чи інший вид діяльності, головну активність впорядкуємо в самому кінці роботи. .

6.2 Налаштування інтерфейсу та реалізація логіки активності для роботи з камерою.

Налаштуємо інтерфейс активності для роботи з камерою. Нам знадобиться вікно попереднього перегляду, додамо у вікно активності елемент SurfaceView. А так же нам знадобиться кнопка для виконання знімків, додамо у вікно активності елемент ImageButton. Далі пропонуємо налаштувати інтерфейс самостійно. У нашому випадку активність виглядає так, як показано на рис. 6.1.

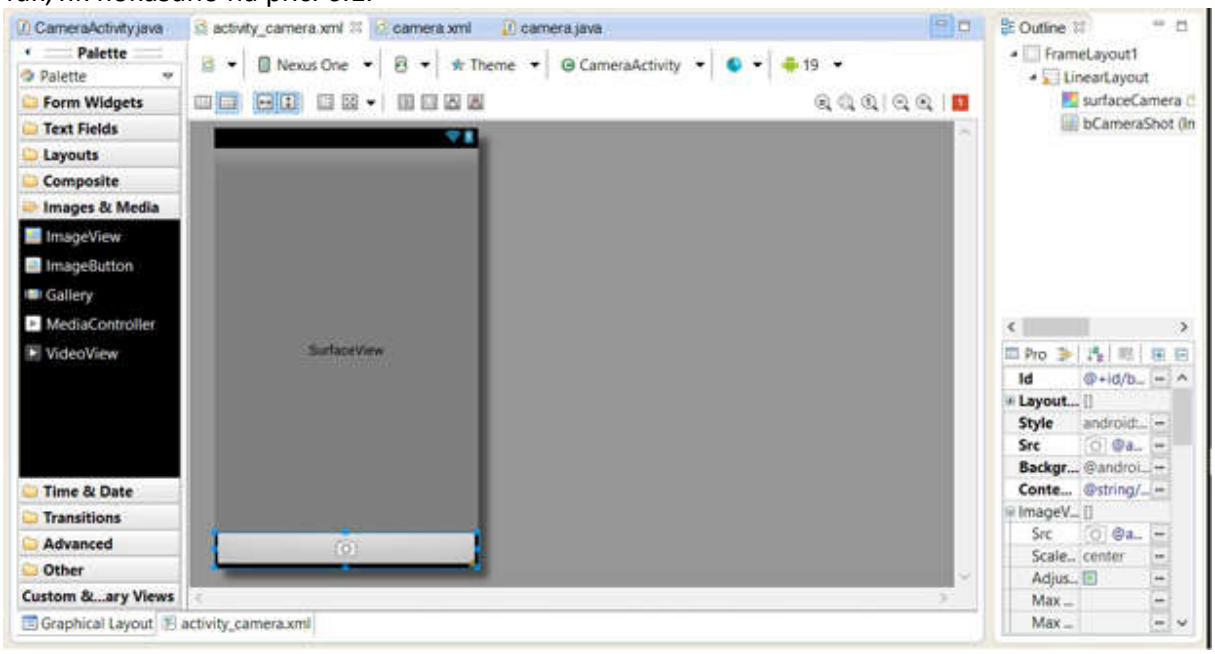

Рис.6.1. Інтерфейс активності для роботи з камерою

Реалізуємо логіку активності, в даному випадку необхідно при натисканні на кнопку виконати знімок. Працюємо з java файлом, що описує відповідний клас активності, **CameraActivity.java.**

Оголосимо наступні поля класу активності:

```
private Camera camera; //для проведення всіх операцій з камерою
private SurfaceHolder surfaceHolder; // для завдання preview
private SurfaceView preview; //для відображення вікна попереднього перегляду
private View shotBtn; //для виконання знімка (кнопка)
```

Для початку роботи з камерою необхідно її форматувати, зробити це краще в методі onResume() класу активності, для ініціалізації використовується наступна конструкція:

```
camera = Camera.open();
```

Після завершення роботи з камерою, необхідно її звільнити для інших додатків, зробити це краще в методі onPause() класу активності, для звільнення камери використовується наступна конструкція:

```
Camera.release();
```

Для активності, що працює з камерою, має сенс відразу задати розташування екрану наступним чином

```
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
```

в іншому випадку, доводиться, наприклад, в методі surfaceCreated () перевіряти розташування екрану і повертати вікно попереднього перегляду, що не дуже зручно, так як поворот екрану займає деякий час.

Виконаємо налаштування вікна попереднього перегляду:

```
preview = (SurfaceView) findViewById (R.id.surfaceCamera);
surfaceHolder = preview.getHolder ();
surfaceHolder.addCallback (new MyCallback ());
```

метод addCallback() використовується для відстеження змін вікна попереднього перегляду. Параметром цього методу служить екземпляр класу реалізує інтерфейс SurfaceHolder.Callback, в нашому випадку для реалізації цього інтерфейсу створений внутрішній клас активності MyCallback.

В цьому класі необхідно реалізувати методи:

```
public void surfaceCreated (SurfaceHolder holder);
public void surfaceChanged (SurfaceHolder holder, int format,
int width, int height);
public void surfaceDestroyed (SurfaceHolder holder);
```

Ми реалізуємо метод surfaceCreated(), що залишилися два методу запишемо з порожньою реалізацією.

У методі surfaceCreated() задамо вікно попереднього перегляду:

```
camera.setPreviewDisplay (holder);
```

Виконаємо необхідні настройки вікна попереднього перегляду (див. Лістинг 6.1).

Запустимо відображення вікна попереднього перегляду:

```
camera.startPreview ();
```

Прийшов час обговорити і реалізувати знімки можна робити. Для того, щоб зробити знімок необхідно викликати метод takePicture(). Виклик цього методу можна виконати з методу-обробника події натискання кнопки, тоді при натисканні на кнопку відразу буде отримана фотографія, а можна "розтягнути задоволення" і скористатися попередньою автофокусуванням.

Спочатку инициализируем кнопку і додамо слухача події натискання в методі onCreate() активності:

```
shotBtn = findViewById (R.id.bCameraShot);
shotBtn.setOnClickListener (new MyViewListener ());
```

В даному випадку слухачем є екземпляр внутрішнього класу MyViewListener, який реалізує інтерфейс View.OnClickListener.

```
class MyViewListener implements View.OnClickListener {
    @Override
    public void onClick (View v) {
        if (v == shotBtn) {
            camera.autoFocus (new MyAutoFocusCallback ());
        }
    }
}
```

У методі onClick() викликаємо обробник автофокусу, слухачем події автофокусування в даному випадку є екземпляр внутрішнього класу MyAutoFocusCallback, який реалізує інтерфейс Camera.AutoFocusCallback.

```
class MyAutoFocusCallback implements Camera.AutoFocusCallback {
    @Override
    public void onAutoFocus (boolean paramBoolean, Camera
                                                paramCamera) {
    if (paramBoolean) {
        // якщо вдалося сфокусуватися, робимо знімок
        paramCamera.takePicture (null, null, null,
                                    new MyPictureCallback ());
        }
    }
}
```

Метод takePicture() має чотири параметри:

Camera.ShutterCallback shutter	- викликається в момент отримання зображення з матриці;
Camera.PictureCallback raw	- програмі передаються для обробки raw дані (якщо підтримується апаратно);
Camera.PictureCallback postview	- програмі передаються повністю оброблені дані (якщо підтримується апаратно);
Camera.PictureCallback jpg	- програмі передається зображення в форматі jpg. Тут може бути організована запис зображення на карту пам'яті.

У нашому випадку останнім параметром є екземпляр класу MyPictureCallback(), який реалізує інтерфейс Camera.PictureCallback. В цьому класі реалізований єдиний метод (див. Лістинг 6.1):

```
    public void onPictureTaken (byte [] paramArrayOfByte, Camera paramCamera)
{...}
```

У лістингу 6.1 представлений код активності, що реалізує роботу з камерою і дозволяє виконувати знімки.

Щоб дозволити програмі працювати з камерою і зберігати фотографії на карті пам'яті, в маніфест необхідно додати наступні дозволи:

```
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission   android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
```

6.4 Налаштування інтерфейсу та реалізація логіки активності для відтворення аудіо та відео

Налаштуємо інтерфейс активності для відтворення аудіо та відео. Нам знадобиться вікно попереднього перегляду, додамо у вікно активності елемент SurfaceView. Програма має поле для завдання розташування медіа контенту для програвання, додамо елемент EditText, а також будуть потрібні кілька кнопок, додамо у вікно активності наступні кнопки:

b_Start - для запуску відтворення контенту з початку, задамо властивості On Click цієї кнопки значення onClickStart

b_Pause - для припинення відтворення, задамо властивості On Click цієї кнопки значення onClick;

b_Resume - для продовження відтворення з місця припинення, задамо властивості On Click цієї кнопки значення onClick;

b_Stop - для повної зупинки відтворення, задамо властивості On Click цієї кнопки значення onClick.

Для можливості вибору зацикленого відтворення медіа контенту додамо у вікно активності елемент CheckBox.

Далі пропонуємо налаштувати інтерфейс самостійно. У нашому випадку активність виглядає так, як показано на рис. 6.2.

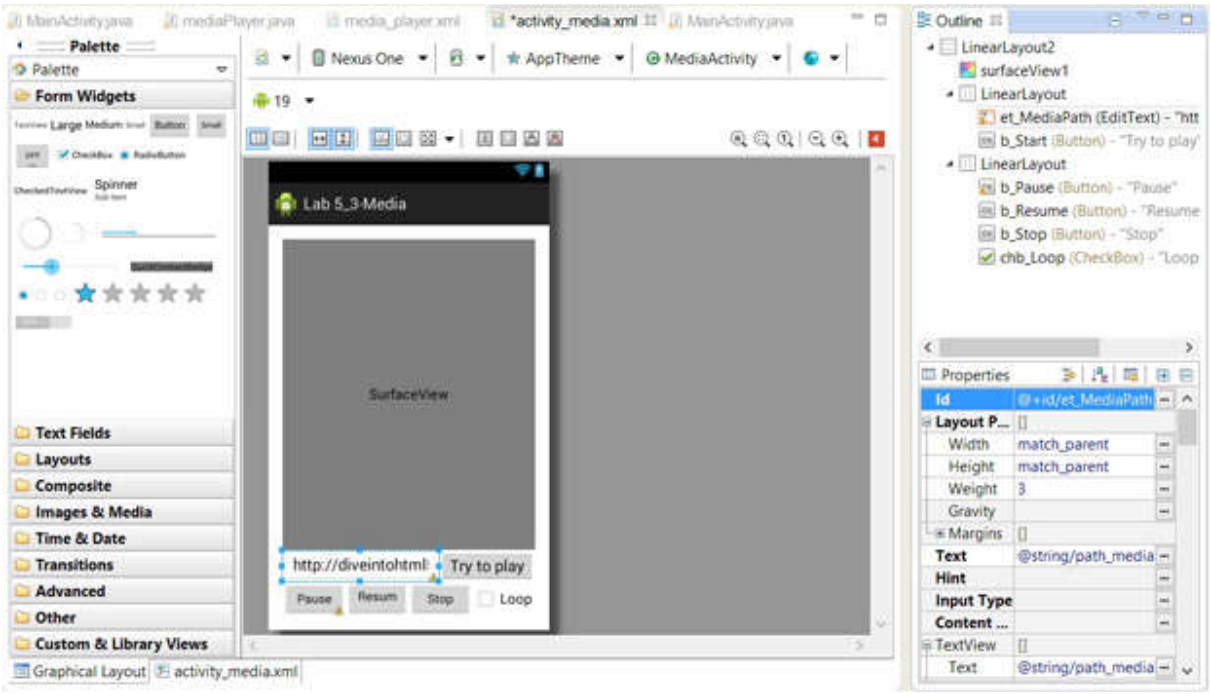

Рис. 6.2. Інтерфейс активності для відтворення аудіо та відео

Реалізуємо логіку активності, в даному випадку необхідно виконувати наступні дії:

-при натисканні на кнопку Try to play, запускається відтворення спочатку контенту, розташованого за адресою, вказаною в поле введення, розташованому зліва від кнопки;

-при натисканні на кнопку Pause, відтворення контенту призупиняється;

-при натисканні на кнопку Resume, відтворення контенту триває з моменту призупинення;

-при натисканні на кнопку Stop, відтворення контенту зупиняється і може бути відновлено тільки з початку, т. Е. Необхідно знову натиснути **Try to play.**

Елемент Loop управляє можливістю повтору відтворення.

Працювати будемо з файлом **MediaActivity.java**, що описує відповідний клас активності.

Оголосимо наступні поля класу активності:

```
MediaPlayer mediaPlayer; // управляє відтворенням
CheckBox chbLoop; // управляє режимом повтору
```

У методі onCreate() активності налаштуємо чек-бокс так, щоб він включав/вимикав режим повтору для плеєра.

```
chbLoop = (CheckBox) findViewById (R.id.chb_Loop);
chbLoop.setOnCheckedChangeListener (new
                          OnCheckedChangeListener () {
    @Override
    public void onCheckedChanged (CompoundButton buttonView,
                            boolean isChecked) {
```

```
        if (mediaPlayer! = null)
        mediaPlayer.setLooping (isChecked);
    }
});
```

Метод onClickStart() використовується для обробки натискань на кнопку Try to play. У цьому методу спочатку звільняємо ресурси поточного програвача, використовуючи виклик методу:

```
releaseMP();
```

Цей метод рекомендується викликати по закінченню використання плеєра, а також при onPause/onStop, якщо немає гострої необхідності тримати об'єкт.

Реалізація методу releaseMP():

```
private void releaseMP() {
    if (mediaPlayer! = null) {
        try {
            mediaPlayer.release();
            mediaPlayer = null;
        } Catch (Exception e) {
          e.printStackTrace();
        }
    }
}
```

Продовжимо розгляд методу onClickStart(). У ньому після звільнення ресурсів, займаємося підготовкою плеєра до відтворення.

```
mediaPlayer = new MediaPlayer();
mediaPlayer.setDataSource (DATA);
mediaPlayer.setDisplay (((SurfaceView)
                findViewById (R.id.surfaceView1)). getHolder ());
mediaPlayer.setOnPreparedListener (this);
mediaPlayer.prepareAsync();
```

Створюємо новий об'єкт класу MediaPlayer.

Використовуємо метод setDataSource() для завдання джерела даних, як параметр передаємо рядок з поля введення, збережену в змінної DATA.

Використовуючи метод setDisplay(), задаємо екран для відтворення, в нашому випадку елемент SurfaceView, доданий у вікно активності на етапі формування інтерфейсу.

Метод prepareAsync() виконує асинхронну підготовку плеєра до відтворення і коли підготовка буде завершена повідомляє про це слухачеві, вказаною в методі setPreparedListener(). У нашому випадку слухачем є сам клас активності, для цього він оголошений, як клас реалізує інтерфейс OnPreparedListener. У разі готовності плеєра викликається метод onPrepared (MediaPlayer mp), оголошений в зазначеному інтерфейсі і реалізований в класі активності, в цьому методі виконується запуск відтворення:

mp.start();

Існує ще метод prepare(), він також виконує підготовку плеєра, але в синхронному режимі. Для прослуховування файлів з інтернету, необхідно використовувати асинхронний режим, що ми і зробили.

Останні два рядки методу onClickStart():

```
mediaPlayer.setLooping (chbLoop.isChecked ());
mediaPlayer.setOnCompletionListener (this);
```

У першій з них визначається можливість циклічного відтворення в залежності від значення чек-боксу.

У другій - задається слухач для отримання повідомлення про досягнення кінця відтвореного контенту. У нашому випадку цим слухачем буде сам клас активності для цього він оголошений, як клас реалізує інтерфейс OnCompletionListener. Повну версію методу onClickStart() можна знайти в лістингу 6.2.

Метод onClick() викликається при натисканні на будь-яку кнопку: **Pause, Resume, Stop**. У цьому методі виконується перевірка яка кнопка була натиснута і після цього виконуються відповідні дії.

Якщо натиснута кнопка Pause, виконується наступна конструкція:

```
if (mediaPlayer.isPlaying ()) mediaPlayer.pause ();
```

Якщо натиснута кнопка Release, виконується наступна конструкція:

```
if (! mediaPlayer.isPlaying ()) mediaPlayer.start ();
```

Якщо натиснута кнопка Stop, виконується наступна конструкція:

```
mediaPlayer.stop();
```

Є сенс звернути увагу на метод активності:

```
protected void onDestroy() {
  super.onDestroy ();
  releaseMP ();
}
```

У цьому методі обов'язково необхідно звільнити ресурси, що і виконується викликом методу releaseMP().

Повний код класу MediaActivity представлений в лістингу 6.2.

6.5 Налаштування інтерфейсу та реалізація логіки активності для перегляду зображень

Налаштуємо інтерфейс активності для перегляду зображень. Нам буде потрібно елемент для перегляду зображень, додамо у вікно активності елемент ImageView. Додамо інформаційне поле, т. Е. Елемент TextView, для відображення інформації про загальну кількість зображень в папці і номері зображення, яке Ви. Додамо дві кнопки, для переміщення від одного зображення до іншого вперед і назад.

Пропонуємо налаштувати інтерфейс самостійно. У нашому випадку активність виглядає так, як показано на рис. 6.3. Але це, зрозуміло, не єдино можливий варіант.

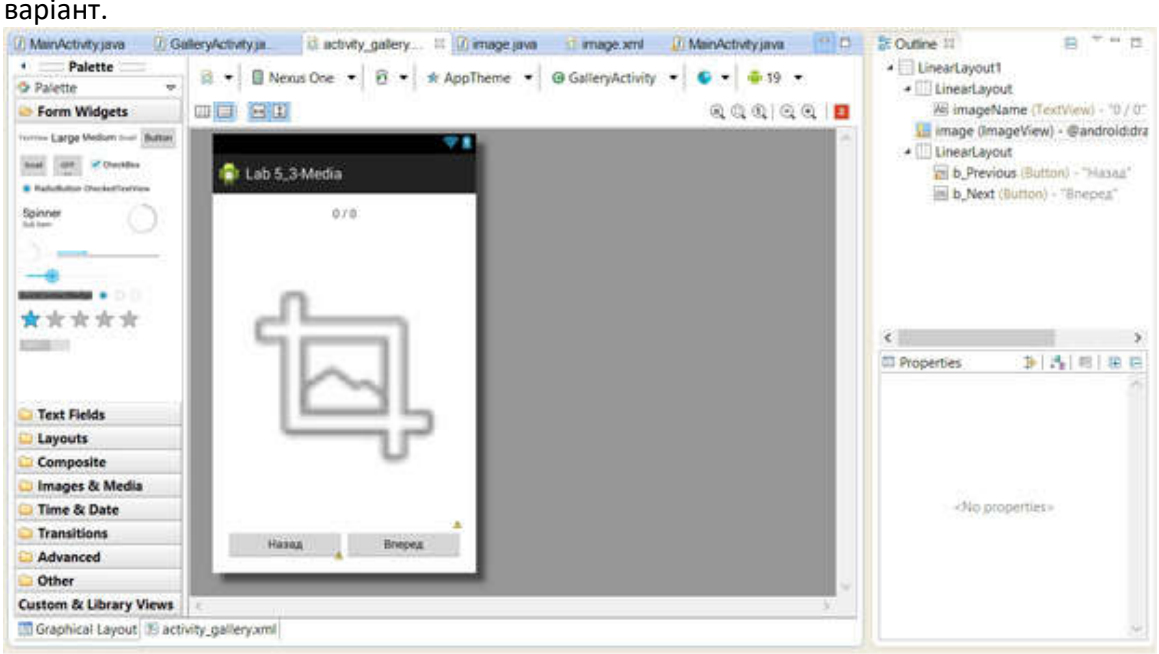

Рис. 6.3. Інтерфейс активності для перегляду зображень

Реалізуємо логіку активності, дана активність орієнтована на перегляд знімків, зроблених в цьому ж додатку і збережених в папці: / sdcard / TrainingMedia /. Кнопка **Назад** виводить у вікно перегляду попереднього зображення, по відношенню до вже відображеного, задамо властивості On Click цієї кнопки значення onPrevious. Кнопка **Уперед** виводить у вікно перегляду наступний знімок, по відношенню до вже відображеного, задамо властивості On Click цієї кнопки значення onNext.

Працювати будемо з файлом **GalleryActivity.java**, що описує клас відповідної активності.

Оголосимо поля класу активності:

```
int currentImage = 0;
ArrayList <String> images;
ImageView imageView;
TextView nameView;
```

Налаштування основних елементів для виведення зображень на екран виконаємо в методі onResume() активності, цей метод викликається кожен раз, перед виведенням активності на передній план.

```
images = new ArrayList <String> ();

imageView = ((ImageView) findViewById (R.id.image));
try {
    File imagesDirectory = new File ( "/ sdcard / TrainingMedia /");
    images = searchImage (imagesDirectory);
    updatePhoto (Uri.parse (images.get (currentImage)));
} Catch (Exception e) {
    nameView.setText ( "Помилка: Папка '/ sdcard / TrainingMedia /" не
                                                        знайдена ");
}
```

Для отримання списку зображень в змінну images використовується метод searchImage(), цей метод, використовуючи переданий в нього в якості параметра адреса директорії з зображеннями, знаходить файл з розширенням .jpg, .png або .jpeg і додає його до списку зображень. Метод повертає список доступних файлів з зображеннями. Код методу представлений в лістингу 6.3.

Метод updatePhoto() виконує оновлення лічильника фотографій в інформаційному полі і виводить на екран зображення, відповідне переданому в метод URI. Код методу представлений в лістингу 6.3.

У методі onPause() активності виконується очищення списку зображень і звільнення пам'яті. Цей метод викликається кожен раз, як активність втрачає фокус введення. Код методу представлений в лістингу 6.3.

Залишилося розглянути ще два методи: onPrevious() і onNext().

Перший метод викликається, коли натиснута кнопка **Назад**. У ньому зменшується номер поточного зображення і викликається метод updatePhoto().

Другий метод викликається, коли натиснута кнопка **Уперед**. У ньому збільшується номер, поточного зображення і викликається метод updatePhoto().

Повний код класу GalleryActivity представлений в лістингу 6.3.

6.6 Налаштування інтерфейсу та реалізація логіки головною активності додатки

Налаштуємо інтерфейс головною активності додатки. Ця активність містить три кнопки (**Image Button**):

bCamera - для виклику активності, що надає можливості роботи з камерою;

bGallery - для виклику активності, що надає можливості перегляду зображень;

bMusic для виклику активності, що надає можливості відтворення аудіо та відео.

Пропонуємо налаштувати інтерфейс самостійно. У нашому випадку активність виглядає так, як показано на рис.6.4. Але це, зрозуміло, не єдино можливий варіант.

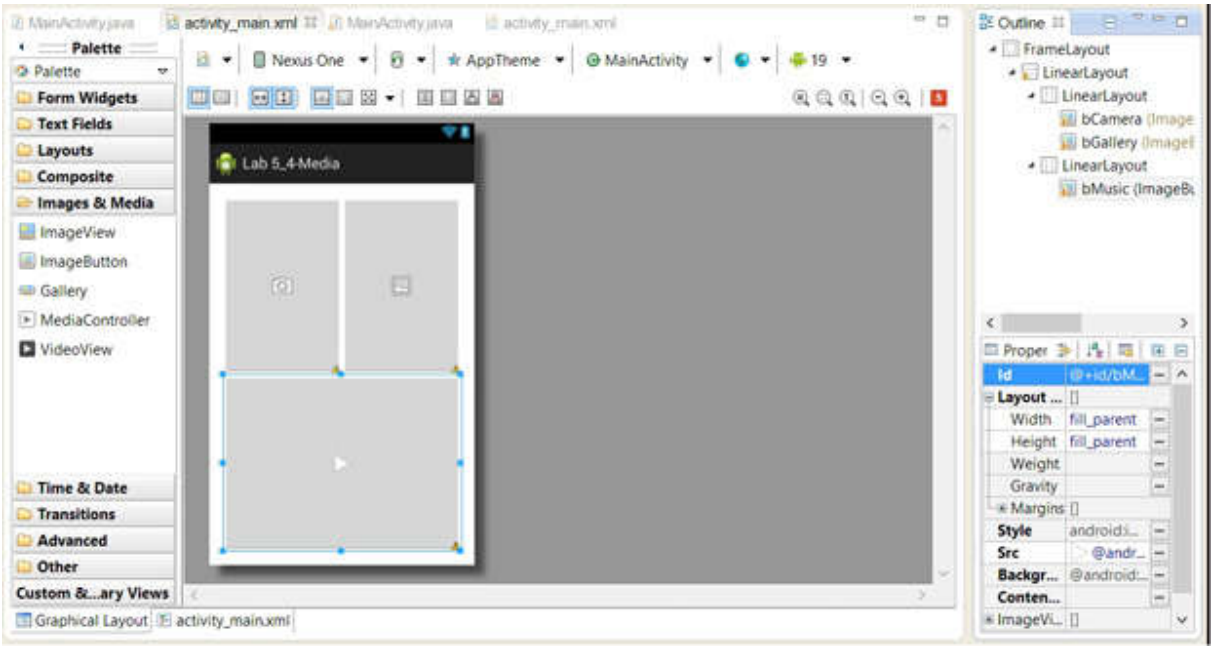

Рис. 6.4. Інтерфейс головної активності додатку

Реалізуємо логіку головною активності. В даному випадку достатньо налаштувати обробку подій натискання на кнопки, таким чином, щоб при натисканні на кнопку запускалася відповідна активність. Працюємо з java файлом, що описує клас активності.

Створимо змінну btnClick, як реалізацію інтерфейсу-слухача

```
OnClickListener:
 OnClickListener btnClick = new OnClickListener () {
 @Override
 public void onClick (View v) {
 Click (v.getId ());
}
};
```

Додамо слухача події натискання до кожної кнопки активності:

```
        ((ImageButton)   findViewById   (R.id.bMusic)).   SetOnClickListener
(btnClick);
       ((ImageButton)   findViewById   (R.id.bCamera)).   SetOnClickListener
(btnClick);
       ((ImageButton)   findViewById   (R.id.bGallery)).   SetOnClickListener
(btnClick);
```

У методі onClick() слухача викликаємо метод Click(), в якому виконується запуск відповідної активності:

```
       protected void Click (int view) {
        Intent intent = null;
        switch (view) {
        case R.id.bMusic: intent = new Intent (this, mediaPlayer.class);
break;
```

```
       case R.id.bGallery: intent = new Intent (this,
GalleryActivity.class); break;
       case R.id.bCamera: intent = new Intent (this,
CameraActivity.class); break;
       default: break;
      }
       if (intent! = Null) {
       startActivity (intent);
      }
      }
```

У лістингу 6.4 представлений код головної активності.

package com.example.lab5_4_media;

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import android.hardware.Camera;
import android.hardware.Camera.Size;
import android.os.Bundle;
import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.ViewGroup.LayoutParams;

public class CameraActivity extends Activity {

  private Camera camera;
  private SurfaceHolder surfaceHolder;
  private SurfaceView preview;
  private View shotBtn;

  @Override
  protected void onCreate (Bundle savedInstanceState) {
    super.onCreate (savedInstanceState);

    // якщо хочемо, щоб додаток завжди мав портретну орієнтацію
    setRequestedOrientation (ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

    // якщо хочемо, щоб додаток був на весь екран
    getWindow (). addFlags (WindowManager.LayoutParams.FLAG_FULLSCREEN);

    // і без заголовку
    requestWindowFeature (Window.FEATURE_NO_TITLE);

    setContentView (R.layout.activity_camera);

    preview = (SurfaceView) findViewById (R.id.surfaceCamera);
    surfaceHolder = preview.getHolder ();
    surfaceHolder.addCallback (new MyCallback (this));
```

```
    surfaceHolder.setType (SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    shotBtn = findViewById (R.id.bCameraShot);
    shotBtn.setOnClickListener (new MyViewListener ());
  }

  @Override
  protected void onResume () {
    super.onResume ();
    camera = Camera.open ();
  }

  @Override
  protected void onPause () {
    super.onPause ();
    if (camera! = null) {
      camera.setPreviewCallback (null);
      camera.stopPreview ();
      camera.release ();
      camera = null;
    }
  }

  class MyCallback implements SurfaceHolder.Callback {

    Activity host;

    MyCallback (Activity act) {
      host = act;
    }

    @Override
    public void surfaceChanged (SurfaceHolder holder, int format, int
width,
    int height) {}
    @Override
    public void surfaceCreated (SurfaceHolder holder) {
      try {
        camera.setPreviewDisplay (holder);
        camera.setPreviewCallback (new MyPreviewCallback ());
      }
      catch (IOException e) {
        Log.d ( "myLogs", "Помилка камери");
        e.printStackTrace ();
      }
      Size previewSize = camera.getParameters (). GetPreviewSize ();
      float aspect = (float) previewSize.width / previewSize.height;

      int previewSurfaceWidth = preview.getWidth ();
      int previewSurfaceHeight = preview.getHeight ();

      LayoutParams lp = preview.getLayoutParams ();

      // тут коригуємо розмір відображуваного preview, щоб не було помилок

      if (host.getResources (). getConfiguration (). orientation! =
            Configuration.ORIENTATION_LANDSCAPE) {
        // портретний вид
        camera.setDisplayOrientation (90);
```

```
            lp.height = previewSurfaceHeight;
            lp.width = (int) (previewSurfaceHeight / aspect);
          }
          else {
            // ландшафтний
            camera.setDisplayOrientation (0);
            lp.width = previewSurfaceWidth;
            lp.height = (int) (previewSurfaceWidth / aspect);
          }
          preview.setLayoutParams (lp);
          camera.startPreview ();
        }

        @Override
        public void surfaceDestroyed (SurfaceHolder holder) {}
      }

      class MyViewListener implements View.OnClickListener {

        @Override
    public void onClick (View v) {
      if (v == shotBtn) {
        // або робимо знімок безпосередньо тут
        // або включаємо обробник автофокусу

        //camera.takePicture(null, null, null, this);
        camera.autoFocus (new MyAutoFocusCallback ());
      }
    }
  }

  class MyAutoFocusCallback implements Camera.AutoFocusCallback {

    @Override
    public void onAutoFocus (boolean paramBoolean, Camera paramCamera) {
      if (paramBoolean) {
        // якщо вдалося сфокусуватися, робимо знімок
        paramCamera.takePicture (null, null, null, new MyPictureCallback ());
      }
    }
  }

  class MyPictureCallback implements Camera.PictureCallback {

    @Override
    public void onPictureTaken (byte [] paramArrayOfByte, Camera paramCamera) {
      // зберігаємо отримані jpg в папці / sdcard / CameraExample /
      // ім'я файлу - System.currentTimeMillis ()
      try {
        File saveDir = new File ( "/ sdcard / CameraExample /");
        if (! saveDir.exists ()) {
          saveDir.mkdirs ();
        }

        FileOutputStream os = new FileOutputStream (String.format ( "/ sdcard /
CameraExample /% d.jpg", System.currentTimeMillis ()));
        os.write (paramArrayOfByte);
        os.close ();
      }
```

```
      catch (Exception e) {}

      // після того, як знімок зроблений, показ прев'ю відключається.
      // Необхідно включити його
      paramCamera.startPreview ();
    }
  }

  class MyPreviewCallback implements Camera.PreviewCallback {

    @Override
    public void onPreviewFrame (byte [] paramArrayOfByte, Camera paramCamera) {
      // тут можна обробляти зображення, що показується в preview
    }
  }
}
```

Лістинг 6.1. Клас CameraActivity для роботи з камерою

```
package com.example.lab5_4_media;

import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnPreparedListener;
import android.os.Bundle;
import android.view.Gravity;
import android.view.SurfaceView;
import android.view.View;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.app.Activity;

public class MediaActivity extends Activity implements OnPreparedListener,
OnCompletionListener {

  MediaPlayer mediaPlayer;
  CheckBox chbLoop;

    @Override
    protected void onCreate (Bundle savedInstanceState) {
        super.onCreate (savedInstanceState);
        setContentView (R.layout.activity_media);

      chbLoop = (CheckBox) findViewById (R.id.chb_Loop);
    chbLoop.setOnCheckedChangeListener (new OnCheckedChangeListener () {
      @Override
      public void onCheckedChanged (CompoundButton buttonView,
          boolean isChecked) {
        if (mediaPlayer! = null)
          mediaPlayer.setLooping (isChecked);
      }
    });
    }

    public void onClickStart (View view) {
```

```
    releaseMP ();

    String DATA = ((EditText) findViewById (R.id.et_MediaPath)). GetText (). 
ToString ();
    try {
      mediaPlayer = new MediaPlayer ();
      mediaPlayer.setDataSource (DATA);
      mediaPlayer.setDisplay (((SurfaceView)
          findViewById (R.id.surfaceView1)). getHolder ());
      //mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
      mediaPlayer.setOnPreparedListener (this);
      mediaPlayer.prepareAsync ();
    } Catch (Exception e) {
      showMessage ( "Помилка відтворення");
    }

    if (mediaPlayer == null)
      return;

    mediaPlayer.setLooping (chbLoop.isChecked ());
    mediaPlayer.setOnCompletionListener (this);
  }

    private void showMessage (String text) {
    Toast toast = Toast.makeText (getApplicationContext (), text,
        Toast.LENGTH_SHORT);
    toast.setGravity (Gravity.CENTER, 0, 0);
    toast.show ();
  }

    private void releaseMP () {
    if (mediaPlayer! = null) {
      try {
        mediaPlayer.release ();
        mediaPlayer = null;
      } Catch (Exception e) {
        e.printStackTrace ();
      }
    }
  }

    public void onClick (View view) {
    if (mediaPlayer == null)
      return;
    switch (view.getId ()) {
    case R.id.b_Pause:
      if (mediaPlayer.isPlaying ())
        mediaPlayer.pause ();
      break;
    case R.id.b_Resume:
      if (! mediaPlayer.isPlaying ())
        mediaPlayer.start ();
      break;
    case R.id.b_Stop:
      mediaPlayer.stop ();
      break;
    }
  }
```

```
      @Override
  public void onPrepared (MediaPlayer mp) {
    mp.start ();
  }

  @Override
  public void onCompletion (MediaPlayer mp) {}

  @Override
  protected void onDestroy () {
    super.onDestroy ();
    releaseMP ();
  }

}
```

Лістинг 6.2. клас MediaActivity

```
package com.example.lab5_4_media;

import java.io.File;
import java.util.ArrayList;
import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

      public class GalleryActivity extends Activity {

  int currentImage = 0;
  ArrayList <String> images;
  ImageView imageView;
  TextView nameView;

  @Override
  protected void onCreate (Bundle savedInstanceState) {
    super.onCreate (savedInstanceState);
    setContentView (R.layout.activity_gallery);
  }

  @Override
  public void onResume () {
    super.onResume ();

    currentImage = 0;

    Log.d ( "myLogs", "onResume cI =" + currentImage);

    nameView = ((TextView) findViewById (R.id.imageName));
    images = new ArrayList <String> ();

    imageView = ((ImageView) findViewById (R.id.image));
    try {
      File imagesDirectory = new File ( "/ sdcard / TrainingMedia /");
      images = searchImage (imagesDirectory);
```

```
      updatePhoto (Uri.parse (images.get (currentImage)));
    } Catch (Exception e) {
      nameView.setText ( "Помилка: Папка '/ sdcard / TrainingMedia /' не
знайдено");
      Log.d ( "myLogs", "Помилка");
    }
  }

  @Override
    protected void onPause ()
    {
        super.onPause ();
        images.clear ();
        Log.d ( "myLogs", "onPause cI =" + currentImage);
    }

  private ArrayList <String> searchImage (File dir) {
    ArrayList <String> imagesFinded = new ArrayList <String> ();
    for (File f: dir.listFiles ()) {
      if (! f.isDirectory ()) {
        String fileExt = getFileExt (f.getAbsolutePath ());
        if (fileExt.equals ( "png") || fileExt.equals ( "jpg") || fileExt.equals
( "jpeg")) {
          Log.d ( "myLogs", "Файл знайдений" + f.getAbsolutePath ());
          imagesFinded.add (f.getAbsolutePath ());
        }
      }
    }
    return imagesFinded;
  }

  public static String getFileExt (String filename) {
    return filename.substring (filename.lastIndexOf ( ".") + 1);
  }

  public void updatePhoto (Uri uri) {
    try {
      nameView.setText ((currentImage + 1) + "/" + images.size ());
      imageView.setImageURI (uri);
    } Catch (Exception e) {
      nameView.setText ( "Помилка завантаження файлу");
    }
  }

  public void onNext (View v) {
    if (currentImage + 1 <images.size () && images.size ()> 0) {
      currentImage ++;
      updatePhoto (Uri.parse (images.get (currentImage)));
    }
  }

  public void onPrevious (View v) {
    if (currentImage> 0 && images.size ()> 0) {
      currentImage--;
      updatePhoto (Uri.parse (images.get (currentImage)));
    }
  }
}
```

Лістинг 6.3. клас GalleryActivity

```
package com.example.lab5_4_media;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;

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

    OnClickListener btnClick = new OnClickListener () {
      @Override
      public void onClick (View v) {
        Log.d ( "myLogs", v.getId () + "");
        Click (v.getId ());
      }
    };
    ((ImageButton) findViewById (R.id.bMusic)). SetOnClickListener (btnClick);
    ((ImageButton) findViewById (R.id.bCamera)). SetOnClickListener (btnClick);
    ((ImageButton) findViewById (R.id.bGallery)). SetOnClickListener (btnClick);
  }
  protected void Click (int view) {
    Intent intent = null;
    Log.d ( "myLogs", view + "");
    switch (view) {
      case R.id.bMusic: intent = new Intent (this, MediaActivity.class); break;
      case R.id.bGallery: intent = new Intent (this, GalleryActivity.class);
break;
      case R.id.bCamera: intent = new Intent (this, CameraActivity.class);
break;
      default: break;
    }
    if (intent! = null) {
      Log.d ( "myLogs", "Інтент =" + intent.toString ());
      startActivity (intent);
    }
  }
}
```

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Bill Phillips, Chris Stewart, Kristin Marsicano Android Programming: The Big Nerd Ranch Guide(3rd Edition). –USA, Atlanta: Big Nerd Ranch Guides, 2017 – 695P.
2. Neil Smyth Android Studio 3.0 Development Essentials - Android 8 Edition 1st Edition. –USA: CreateSpace Independent Publishing Platform; 1 edition, 2017 – 726P.
3. Dawn Griffiths, David Griffiths Head First Android Development: A Brain-Friendly Guide 2nd Edition. – USA, California: O'Reilly Media; 2 edition, 2017 – 928P.
4. Ian F. Darwin Android Cookbook: Problems and Solutions for Android Developers 2nd Edition. – USA, California: O'Reilly Media; 2 edition, 2017 – 772P.
5. Josh Skeen, David Greenhalgh Kotlin Programming: The Big Nerd Ranch Guide 1st Edition –USA, Atlanta: Big Nerd Ranch Guides, 2018 – 384P.
6. Mark Wickham Practical Android: 14 Complete Projects on Advanced Techniques and Approaches. - USA, New York: Apress; 1st ed. Edition. – 2018 – 260P.
7. Adam Gerber, Clifton Craig Learn Android Studio: Build Android Apps Quickly and Effectively 2nd Edition. - USA, New York: Apress; 1st ed. Edition. – 2018 – 485P.
8. Wyken Seagrave Rapid Android App Development Using Basic. – UK, Coventry: Penny Press Ltd.- 2018. – 670P.
9. Reto Meier, Ian Lake Professional Android 4th Edition. – UK, Birmingham: Wrox; 4 edition. – 2018.-928P.
10. Ted Hagos Learn Android Studio 3: Efficient Android App Development. - USA, New York: Apress; 1st ed. Edition. – 2018 – 280P.
11. Milos Vasic (Author) Mastering Android Development with Kotlin: Deep dive into the world of Android to create robust applications with Kotlin. – UK, Birmingham: Packt Publishing.- 2017.- 378P.
12. Andrés Colubri Processing for Android: Create Mobile, Sensor-Aware, and VR Applications Using Processing. - USA, New York: Apress; 1st ed. Edition. – 2017 – 408P.