МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт
з дисципліни **«ПРОГРАМУВАННЯ МОБІЛЬНИХ ДОДАТКІВ»**

здобувачів освітнього ступеня «бакалавр»
за спеціальністю 121 – Інженерія програмного забезпечення,
освітня програма «Інженерія програмного забезпечення»
галузі знань 12 – Інформаційні технології
усіх форм навчання

Черкаси 2022

Затверджено вченою радою ФІТІС, протокол №___ від _____2022 р., згідно з рішенням кафедри програмного забезпечення автоматизованих систем, протокол № 12 від 08.02.2022 р.

Упорядники Олексюк В.В., *канд. техн. наук, ст. викладач*
Куницька С.Ю., *канд. техн. наук, доцент*
Метелап В.В., *канд. техн. наук, доцент*

Рецензент Карапетян А.Р., *к.т.н., доцент*

Методичні рекомендації до виконання лабораторних робіт з дисципліни «ПРОГРАМУВАННЯ МОБІЛЬНИХ ДОДАТКІВ» для здобувачів освітнього ступеня «бакалавр» за спеціальністю 121 – Інженерія програмного забезпечення, освітня програма «Інженерія програмного забезпечення» галузі знань 12 – Інформаційні технології усіх форм навчання [Електронний ресурс] / [упоряд. В.В. Олексюк, С.Ю. Куницька, В.В. Метелап]; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси : ЧДТУ, 2022. – 63 с. – Назва з титульного екрана.

анотація

……

Навчальне електронне видання
мережного використовування

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
до виконання лабораторних робіт
з дисципліни «**ПРОГРАМУВАННЯ МОБІЛЬНИХ ДОДАТКІВ**»
здобувачів освітнього ступеня «бакалавр»
за спеціальністю 121 – Інженерія програмного забезпечення,
освітня програма «Інженерія програмного забезпечення»
галузі знань 12 – Інформаційні технології
усіх форм навчання
Упорядники: **Олексюк** Вадим Володимирович
Куницька Світлана Юріївна
Метелап Володимир Володимирович

В авторській редакції.

ЗМІСТ

Лабораторна робота №1.

Тема «Встановлення IDE Android Studio»

Мета: навчитися встановлювати та налаштовувати середовище розробки для програмування мобільних додатків ОС Android.

Звіт повинен містити:

1. Постановка завдання.
2. Теоретичні відомості по проекту.
3. Результати виконання завдання.
4. Висновок.

Завдання

1. Встановити середовище розробки для програмування мобільних додатків ОС Android.
2. Описати хід виконання роботи.
3. Зробити висновки.

Теоретичні відомості

Розробка Android-додатків, як і у випадку з будь-якими іншими додатками, починається з встановлення середовища розробки (SDK).

Налаштування середовища

Ми поділимо цей етап на три кроки:

Крок 1. Встановлення JDK (рис.1, якщо не встановлено);

JDK можна знайти за посиланням:

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Рисунок 1 – Встановлення Java Development Kit

Крок 2. Встановлення Android Studio (рис. 2);

Останню версію **Android Studio** можна знайти за посиланням:

https://developer.android.com/studio/index.html

Детальну інструкцію з встановлення JDK та Android Studio можна отримати за посиланням:

https://www.youtube.com/watch?v=9ucX3UlCT6E&list=PLyfVjOYzujugap6Rf3ETNKkx4v9ePllNK&index=2

Рисунок 2 – Встановлення Android Studio

Крок 3. Встановлення емулятора пристрою (рис. 3) Android Virtual Device (AVD).

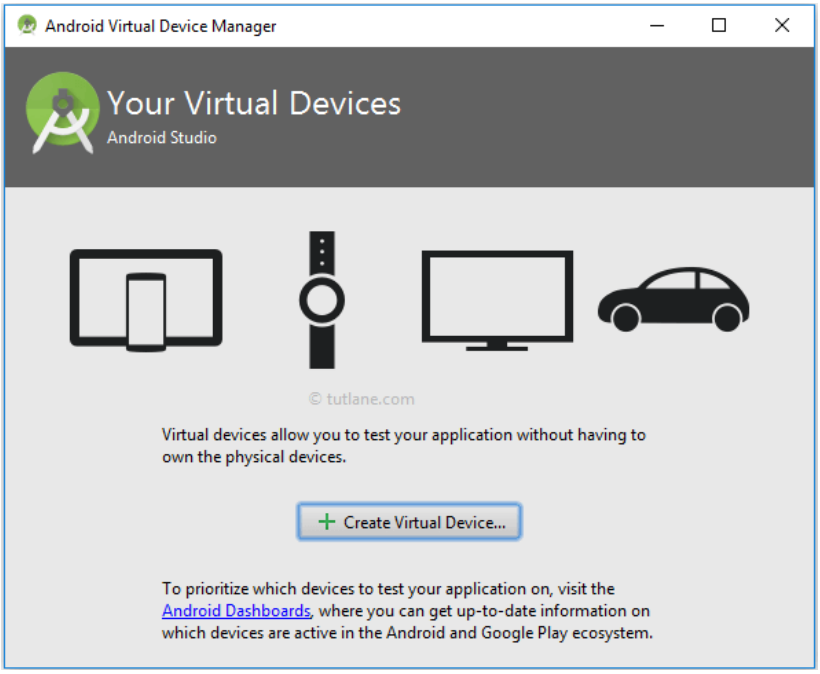


Рисунок 3 – Встановлення Android Virtual Device

Налаштування пристрою

Створення віртуального пристрою (AVD) та структуру проекту найпростішого додатку можна знайти за посиланням:

https://www.youtube.com/watch?v=SbWzaPtvzJA&list=PLyfVjOYzujugap6Rf3ETNKkx4v9ePllNK&index=3

Налаштування пристрою для розробки по USB:

На своєму пристрої Android перейдіть до «*Налаштування системи*», «*Параметри розробника*», і переконайтеся, що увімкнено «*Налагодження USB* (*USB Debugging*)».

Примітка. На Android 4.2 і новіших версіях параметри розробника за замовчуванням приховані. Щоб зробити його доступним, перейдіть у Налаштування > Про телефон і сім разів натисніть по Номеру збірки. Поверніться на попередній екран, щоб знайти параметри розробника, зокрема «Налагодження USB».

Лабораторна робота №2.

Тема: «Знайомство з проектом мобільного додатку в Android Studio»

Мета: отримати навички роботи з найпростішим проектом додатку під ОС Android.

Звіт повинен містити:

1. Постановка завдання.
2. Теоретичні відомості по проекту.
3. Результати виконання завдання.
4. Висновок.

Завдання

1. Створити мобільний додаток описаний в теоретичних відомостях.
2. Описати хід роботи.
3. Зробити висновки.

Теоретичні відомості

В якості мови програмування для Android використовується Java. Для створення призначеного для користувача інтерфейсу використовується XML. За традицією, закладеною в минулому столітті, кожен програміст повинен був написати «Hello World!» в якості першої програми. Часи змінюються, і програма «Hello World!» вже вбудована в середовище розробки під Android.

Спочатку запустимо готову програму «Hello World!» без написання коду, щоб переконатися, що весь інструментарій коректно встановився, і ми можемо створювати і налагоджувати програми. А потім вже напишемо свою першу програму.

Створення нового проекту

Запускаємо **Android Studio** і вибираємо File | New | New Project З'явиться діалогове вікно майстра, в якому ми можемо обирати типи пристроїв, під які будемо розробляти свій додаток (рис.1).

У більшості випадків ми будемо писати для смартфонів і планшетів, тому залишаємо вибраним перший пункт. Також ви можете писати програми для Wear OS, Android TV і т.д..

Для обраного пристрою слід вибрати зовнішній вигляд екрану програми. Запропоновані шаблони дозволяють заощадити час на написання стандартного коду для типових ситуацій. Досвідчений розробник може вручну написати будь-який із запропонованих варіантів, використовуючи варіант ***No Activity***, де ніяких заготовок не буде.

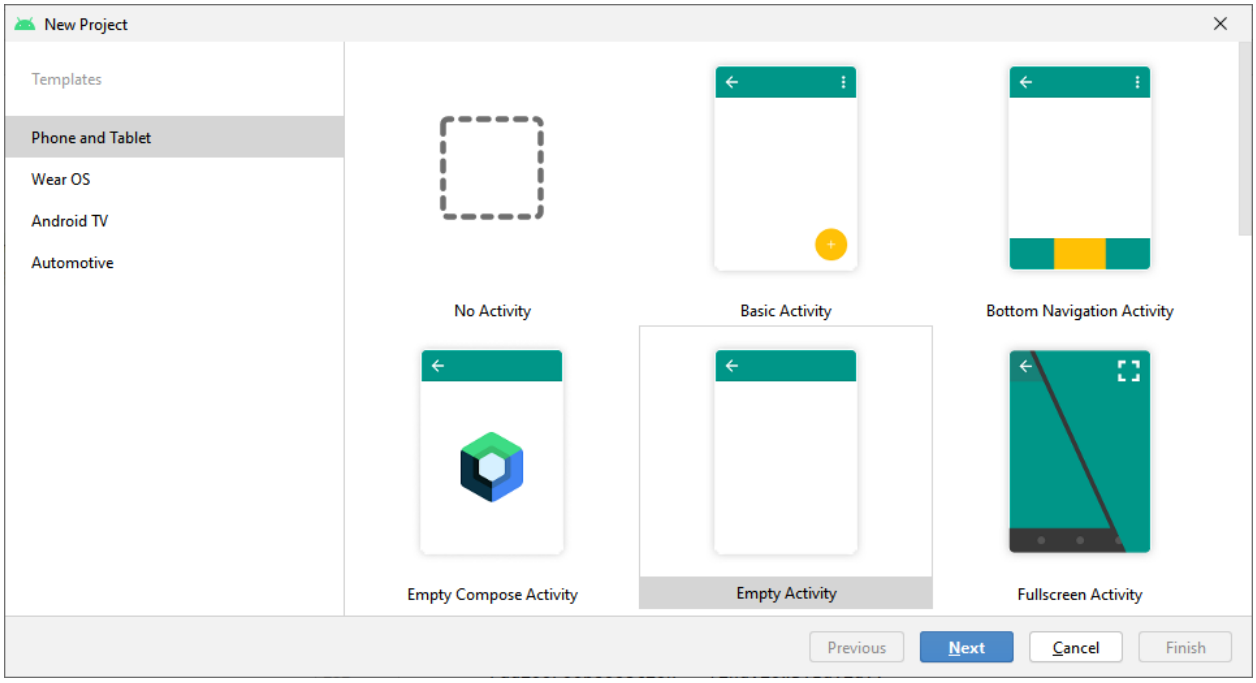


Рисунок 1 – Вікно вибору типи пристроїв

Шаблон ***Empty Activity*** призначений для звичайних телефонів. На зображенні над назвою шаблону ви бачите приблизний вигляд програми з використанням даної заготовки. Для навчальних програм в 99% підійде цей варіант.

Шаблон ***Primary/Detail Flow*** призначений для планшетів з реалізацією двох-панельного режиму. Шаблон ***Fullscreen Activity*** можна використовувати коли потрібно додатковий простір без зайвих деталей. Інші шаблони потрібні для створення додатків з Google Maps або сервісами Google Play. Слід зауважити, що список шаблонів постійно поповнюється.

Отже, ми вибрали варіант Empty Activity і переходимо до наступного вікна натиснувши кнопку *Next*. В даному вікні (рис. 2) ми бачимо наступні поля:

Поле Application ***Name***: – зрозуміле ім'я для програми, яка буде відображатися в заголовку програми. За замовчуванням у вас вже може бути My Application. Замінимо на «Hello World». В принципі ви могли написати тут і «Привіт, світ!», але у Android є чудова можливість виводити потрібні рядки на телефонах з різними мовами. Скажімо, у американця на телефоні з'явиться напис англійською, а в українця – українською. Тому в початкових настройках завжди використовуються англійські варіанти, а локалізовані рядки підготуєте пізніше. Необхідно відразу виробляти звичку до правильного коду.

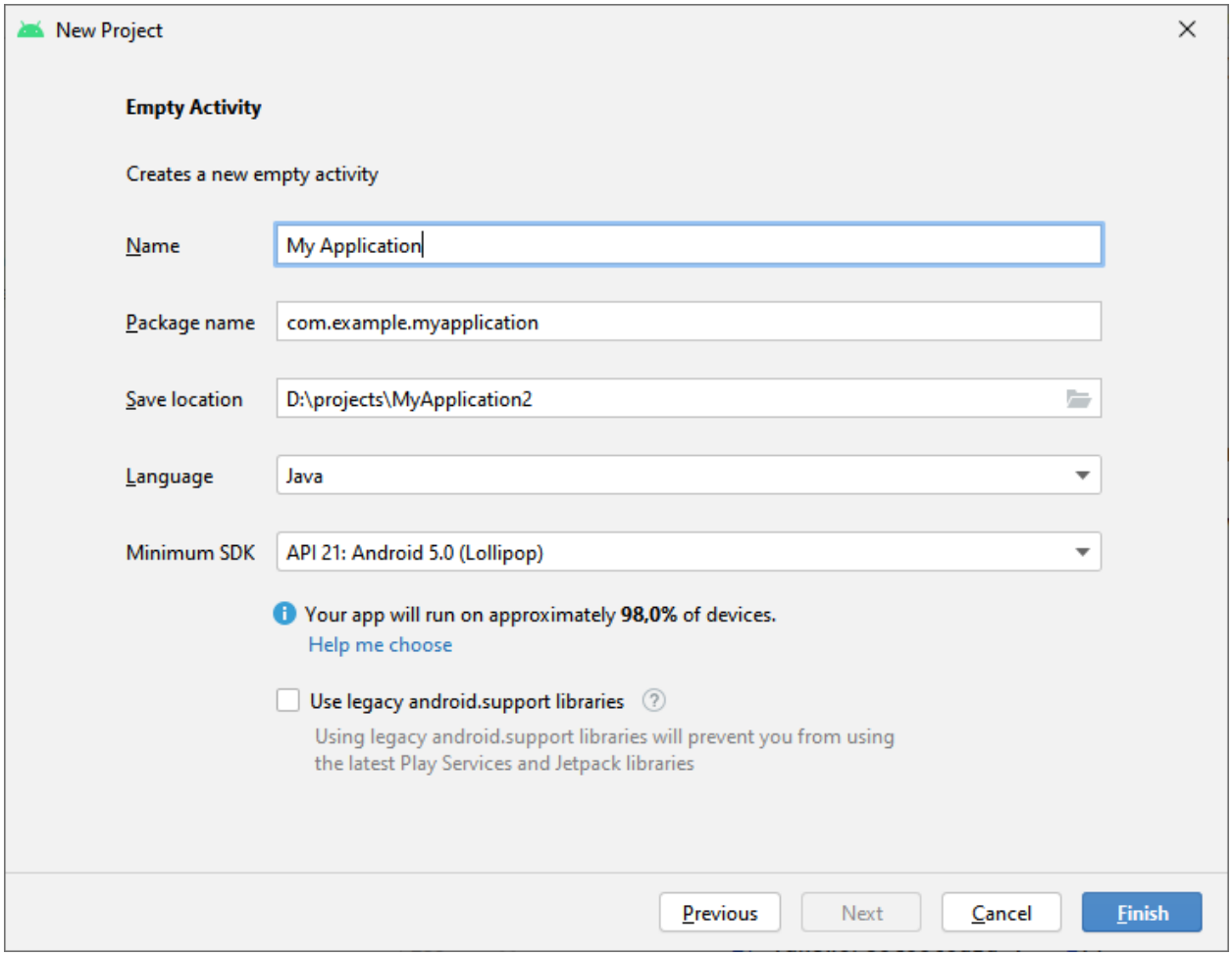


Рисунок 2 – Налаштування проекту

Поле ***Package name***: формує спеціальний Java-пакет на основі вашого імені з попереднього поля. В Java використовується перевернутий варіант

для найменування пакетів, тому спочатку йде *ua*, *org*, а потім вже назва сайту. Пакет служить для унікальної ідентифікації вашої програми, коли ви будете його поширювати в Google Play. Зверніть увагу, що Google в своїй документації використовує пакет *com.example* в демонстраційних цілях. Якщо ви будете просто копіювати приклади з документації і в такому вигляді спробуєте викласти в Google Play, то у вас нічого не вийде – це назва зарезервовано і заборонено до використання в магазині додатків. Нам дається змога відредагувати запропонований варіант.

Поле ***Save location***: дозволяє вибрати місце на диску для створюваного проекту.

Поле ***Language***: дає можливість вибору мови програмування Java або Kotlin.

Поле ***Minimum SDK***: мінімальна версія андроїд системи, яку буде підтримувати наш додаток. Виберіть свій варіант. На даний момент Google підтримує версії, починаючи з API 15, випускаючи спеціальні бібліотеки сумісності (android.support libraries) для старих пристроїв. Але ви можете вибрати більш сучасний варіант.

Ми закінчили з первинною настройкою. Натискаємо кнопку Finish.

Далі студія формує проект і створює необхідну структуру з різних файлів і папок.

У лівій частині середовища розробки на вкладці Android з'явиться ієрархічний список з папок, які відносяться до проекту. У деяких випадках бажано перемкнутися на режим Project, який показує справжнє розташування файлів. Але на початку зручніше використовувати саме вид Android, який ховає службові файли, щоб не плутати новачків.

Вміст проекту

Вкладка Android містить дві основні папки: app і Gradle Scripts. Перша папка app є окремим модулем для програми та містить всі необхідні файли програми – код, ресурси картинок і т.п. Друга папка служить для різних налаштувань, управління проектом та багатьох інших речей.

Зараз нас повинна цікавити папка app. Розкрийте її. У ній знаходяться три папки: manifest, java, res.

Директорія manifest

Директорія manifest містить єдиний файл маніфесту AndroidManifest.xml. У цьому файлі повинні бути оголошені всі активності (Activity), служби (Service), приймачі (Broadcast Receiver) і контент-провайдери (Content Provider) додатку. Також він повинен містити необхідні додатку дозволи. Наприклад, коли програмі слід надати доступ до мережі, це повинно бути визначено тут. «AndroidManifest.xml» можна розглядати, як опис (для цільової ОС) по розгортанню Android-додатку.

Директорія java

Директорія java містить три папки – робочу і для тестів. Робоча папка має назву вашого пакета і містить файли класів. Зараз там один клас MainActivity. Папки для тестів можете не чіпати.

Директорія res

Директорія res містить файли ресурсів, розбитих на окремі папки:

drawable – в цих папках зберігають графічні ресурси – картинки і xml-файли, що описують колір і фігури.

layout – в цій папці містяться xml-файли, що описують зовнішній вигляд

форм і різних елементів форм. Після створення проекту там вже є файл activity_main.xml, який відповідає за зовнішній вигляд головного вікна програми.

mipmap – тут зберігають значки додатку під різні роздільні здатності екрану.

values – тут розміщуються рядкові ресурси, ресурси кольорів, тем, стилів і вимірювань, які ми можемо використовувати в нашому проекті. Тут ви можете бачити файли *colors.xml, dimens.xml, strings.xml, styles.xml.*

Для вивчення вам потрібно відкрити два файли – MainActivity і activity_main.xml (res / layout) в центральній частині Студії.

Якщо файли не відкриті, то відкрийте їх самостійно подвійним клацанням для редагування (або перегляду). Таким способом ви можете відкрити будь-який потрібний вам файл.

Не будемо поки вивчати код, а просто натиснемо на зелений трикутник Run (Shift + F10) на панелі інструментів у верхній частині студії для запуску програми.

Якщо все зробили правильно, то в емуляторі або на пристрої завантажиться ваша програма.

Лабораторна робота №3.

Тема «Основи створення інтерфейсу»

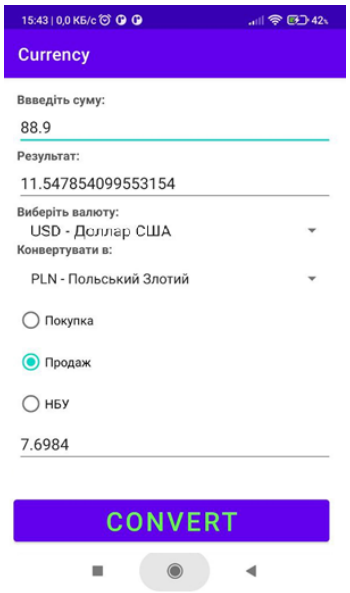
Мета: отримати навички роботи зі створення інтерфейсу.

Звіт повинен містити:

1. Постановка завдання.
2. Теоретичні відомості по проекту.
3. Результати виконання завдання.
4. Висновок.

Завдання

1. Створити інтерфейс мобільного додатку зображеного на рисунку:



2. Описати хід роботи.
3. Зробити висновки.

Звіт повинен містити:

1. Постановка завдання.
2. Теоретичні відомості про призначення проекту.
3. Кілька екранних форм призначень ресурсів проекту з завданням.
4. Висновок.

Теоретичні відомості

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

Прості об'єкти View є елементами управління та інші віджети, наприклад, кнопки, текстові поля і т.д., через які користувач взаємодіє з програмою:

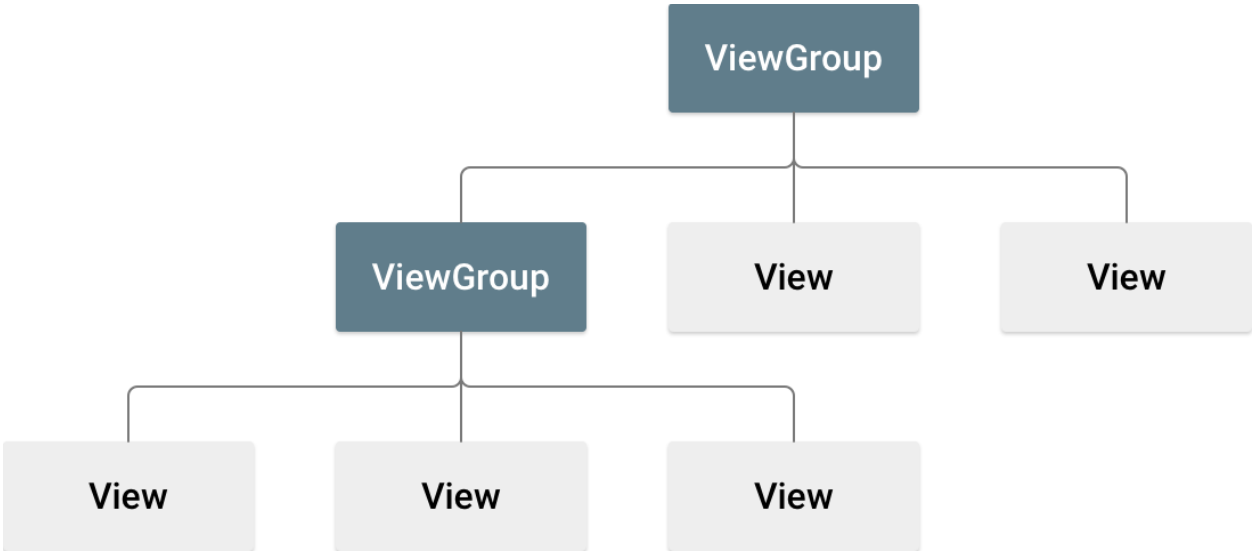


Рисунок 1 – Ієрархія View компонентів

Більшість візуальних елементів, які наслідуються від класу View, такі як кнопки, текстові поля і інші, розташовуються в пакеті android.widget.

Стратегії визначення інтерфейсу

Розмітка визначає візуальну структуру користувальницького інтерфейсу.

Встановити розмітку можна двома способами:

• Створити елементи управління програмно в коді java

• Оголосити елементи інтерфейсу в XML

• Поєднання обох способів - базові елементи розмітки визначити в XML, а решту додавати під час виконання програми.

Створення інтерфейсу в коді java

Визначимо в класі MainActivity найпростіший інтерфейс:

```
package com.example.myapplication;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  // створення TextView
  TextView textView = new TextView(this);
  // встановлення тексту в TextView
  textView.setText("Hello Android!");
  // встановлення висоти тексту
  textView.setTextSize(22);
  // встановлення візуального інтерфейсу для activity
  setContentView(textView);
 }
}
```

Тут весь інтерфейс представлений елементом TextView, який призначений для вивода тексту. За допомогою методів, які, як правило, починаються на set, можна встановити різні властивості TextView. Наприклад, в даному випадку метод setText() встановлює текст в поле, а setTextSize() задає висоту шрифту.

Для установки елемента в якості інтерфейсу додатку в коді Activity викликається метод setContentView(), в який передається візуальний елемент. Хоча ми можемо використовувати подібний підхід, в той же час більш оптимально визначати візуальний інтерейса в файлах xml, а всю пов'язану логіку визначати в класі activity. Тим самим ми досягнемо розмежування інтерфейсу і логіки додатка, їх легше буде розробляти і надалі модифікувати.

Визначення інтерфейсу у файлі XML. Файли layout у додатках Android візуальний інтерфейс нерідко завантажується зі спеціальних файлів xml, які зберігають розмітку. Ці файли є ресурсами розмітки. Подібний підхід нагадує створення веб-сайтів, коли інтерфейс визначається в файлах html, а логіка програми - в коді javascript.

Оголошення призначеного для користувача інтерфейсу в файлах XML дозволяє відокремити інтерфейс програми від коду. Що означає, що ми можемо змінювати визначення інтерфейсу без зміни коду java. Наприклад, в додатку можуть бути визначені розмітки в файлах XML для різних орієнтацій монітора, різних розмірів пристроїв, різних мов, тощо. Крім того, оголошення розмітки в XML дозволяє легше візуалізувати структуру інтерфейсу і полегшує налагодження. Файли розмітки графічного інтерфейсу розташовуються в проекті в каталозі res / layout. За замовчуванням при створенні проекту вже є один файл ресурсів розмітки activity_main.xml, який може виглядати приблизно так:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".MainActivity">

 <TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Hello World!"
  app:layout_constraintBottom_toBottomOf="parent"
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintRight_toRightOf="parent"
```

```
  app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

У файлі визначаються всі графічні елементи і їх атрибути, які складають інтерфейс. При створенні розмітки в XML слід дотримуватися деяких правил: кожен файл розмітки повинен містити один кореневий елемент, який повинен представляти об'єкт View або ViewGroup.

В даному випадку кореневим елементом є елемент ConstraintLayout, який містить елемент TextView.

При компіляції кожен XML-файл розмітки компілюється в ресурс View.

Завантаження ресурсу розмітки здійснюється в методі Activity.onCreate. Щоб встановити розмітку для поточного об'єкта activity, треба в метод setContentView як параметр передати посилання на ресурс розмітки.

Для отримання посилання на ресурс в коді java необхідно вжити вислів R.layout.[Назва_ресурса]. Назва ресурсу layout буде збігатися з ім'ям файлу, тому щоб використовувати файл activity_main.xml як джерело візуального інтерфейсу, нам треба змінити код MainActivity наступним чином:

```
package com.example.myapplication;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

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

Додавання файлу layout

Але у нас може бути і кілька різних ресурсів layout. Як правило, кожен окремий клас Activity використовує свій файл layout. Або для одного класу Activity може використовуватися відразу кілька різних файлів layout.

Наприклад, додамо в проект новий файл розмітки інтерфейсу. Для цього натиснемо на папку res / layout правою кнопкою миші і в меню виберемо пункт New -> Layout resource file. Після цього в спеціальному віконці буде запропоновано вказати ім'я і кореневий елемент для файлу layout (рис. 2):

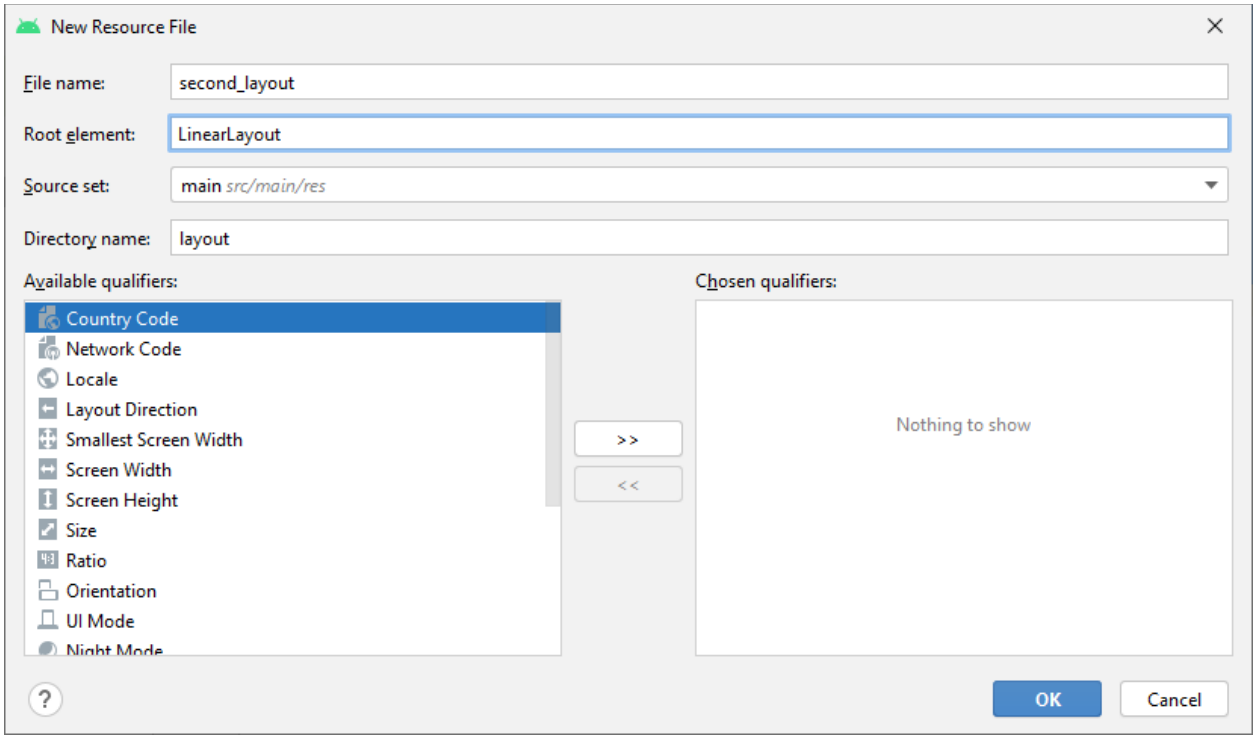


Рисунок 2 – Створення layout файлу

Після цього в папку res / layout буде додано новий файл second_layout.xml, з яким ми можемо працювати так само, як і з activity_main.xml. Зокрема, відкриємо файл second_layout.xml і змінимо його вміст наступним чином:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical"
```

```
 android:layout_width="match_parent"
 android:layout_height="match_parent">

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

</LinearLayout>
```

Тут визначено текстове поле TextView, яке виводить на екран найпростіший текст "Second Activity". Отримання елементів в коді і їх ідентифікатори Крім тексту, ширини і висоти текстове поле встановлює ще один важливий атрибут – id. Знак + в запису android: id = "@ + id / header" означає, що якщо для елемента невизначений id зі значенням header, то його слід визначити.

Щоб використовувати цей файл в якості основного інтерфейсу перейдемо до MainActivity і змінимо її код:

```
package com.example.myapplication;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  // встановлюємо в якості інтерфейсу файл
  setContentView(R.layout.second_layout);
  // отримуємо елемент textView
  TextView textView = (TextView) findViewById(R.id.header);
  // перевстановлюємо в нього текст
  textView.setText("Hello Android 7!"); }
}
```

За допомогою методу setContentView() встановлюється розмітка з файлу second_layout.xml.

Інший важливий момент, який варто відзначити – отримання візуального елемента TextView. Так як в його коді ми визначили атрибут android: id, то через цей id ми можемо його отримати. Для отримання елементів з id клас Activity має метод findViewById (). У цей метод передається ідентифікатор ресурсу у вигляді R.id.[ідентіфікатор_елемента]. Цей метод повертає об'єкт View - об'єкт базового класу для всіх елементів, тому результат методу ще необхідно привести до типу TextView.

Далі ми можемо щось зробити з цим елементом, в даному випадку змінюємо його текст. Причому що важливо, отримання елемента відбувається після того, як в методі setContentView була встановлена розмітка, в якій цей візуальний елемент був визначений.

Графічні можливості Android Studio

Android Studio має досить просунутий інструментарій, який полегшує розробку графічного інтерфейсу. Ми можемо відкрити файл activity_main.xml і внизу за допомогою кнопки Design (рис. 3) переключитися в режим дизайнера до графічного подання інтерфейсу у вигляді ескізу смартфона:

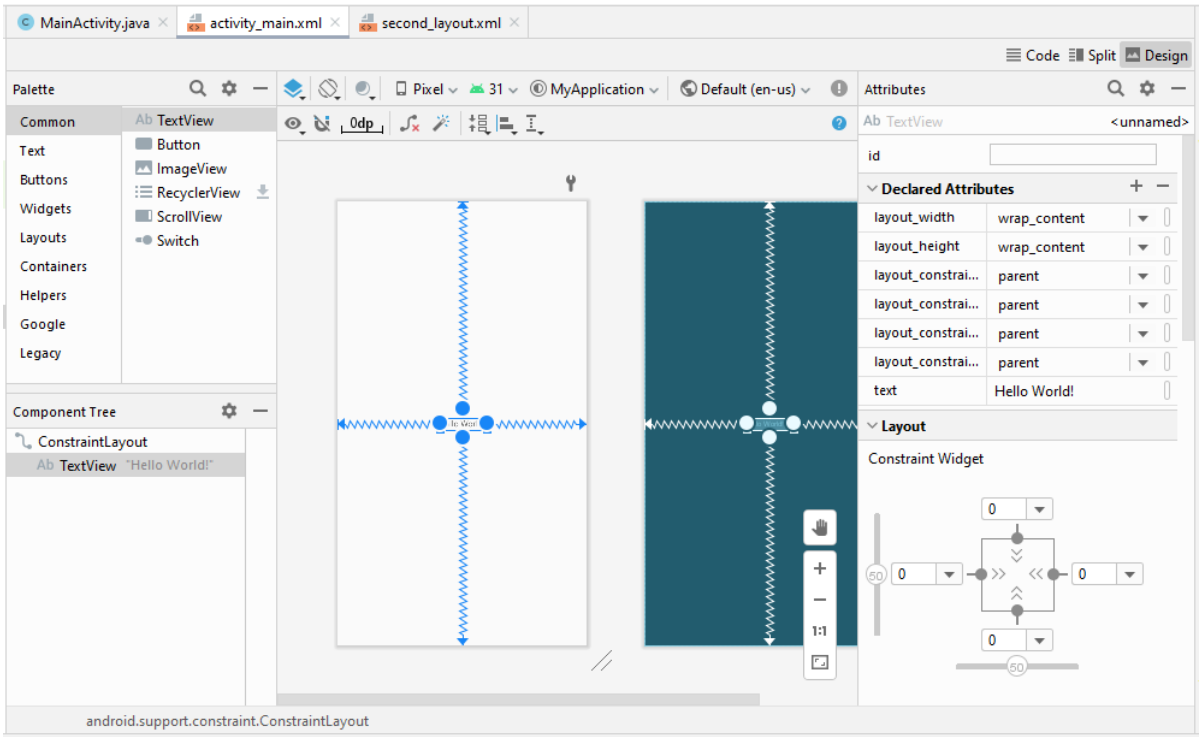


Рисунок 3 – Режим Design

Зліва буде знаходитися панель інструментів (Palette), з якою ми можемо переносити потрібний елемент мишкою на ескіз смартфона. І всі перенесені елементи будуть автоматично додаватися в файл activity_main.xml. За допомогою миші ми можемо змінювати позиціонування вже доданих елементів, переносячи їх в інше місце на смартфоні.

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

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

Але навіть якщо ми віддаємо перевагу працювати з розміткою інтерфейсу в текстовому вигляді, то навіть тут ми можемо включити попередній перегляд для файлу activity_main.xml. Для цього після перемикання в текстовий режим необхідно натиснути на кнопку Split зліва кнопки Designer:

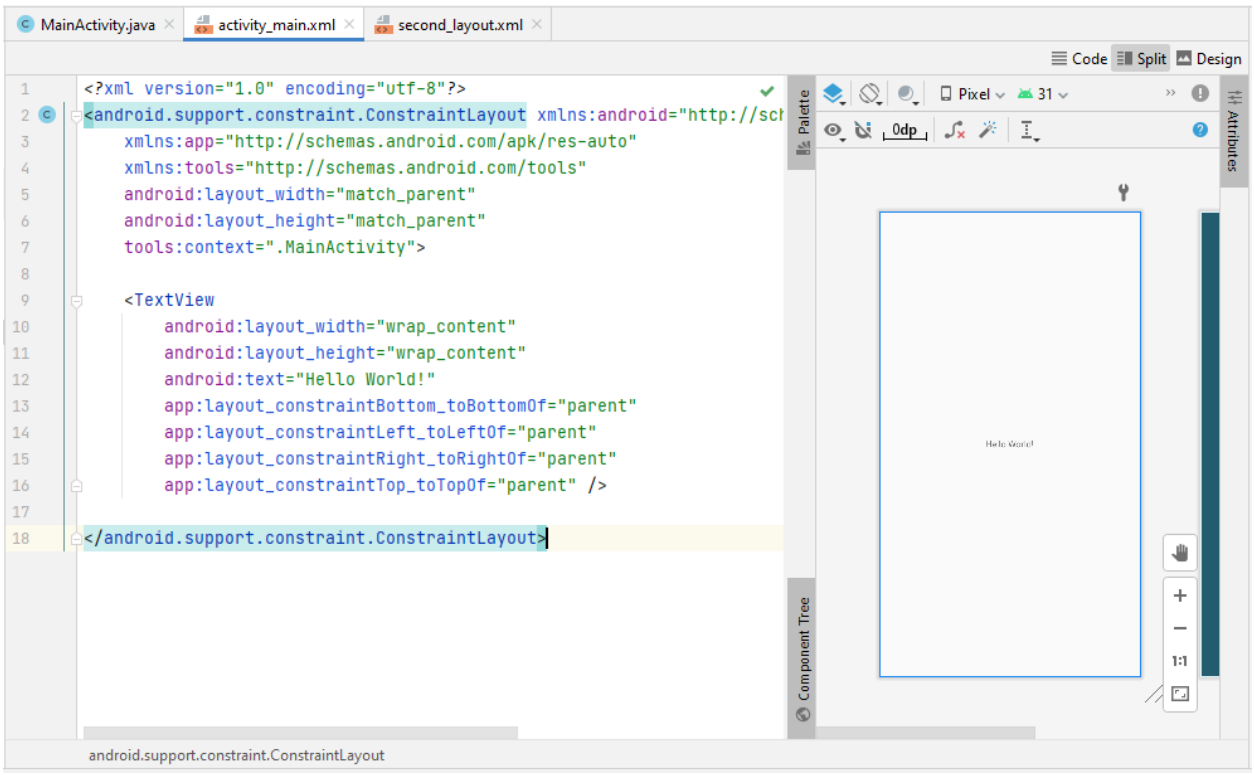


Рисунок 3 – Режим Split

Це дуже зручно, так як відразу дозволяє переглянути, як буде виглядати додаток. А при будь-яких змінах область попереднього перегляду буде автоматично синхронізуватися з вмістом файлу activity_main.xml.

Визначення розмірів

В ОС Android ми можемо використовувати різні типи вимірювань:

• px: пікселі поточного екрану. Однак ця одиниця виміру не рекомендується, так як реальне уявлення зовнішнього вигляду може змінюватися в залежності від пристрою; кожен пристрій має певний набір пікселів на дюйм, тому кількість пікселів на екрані може також змінюватися

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

Абстрактна одиниця виміру, заснована на фізичній щільності екрану з роздільною здатністю 160 dpi (точок на дюйм). В цьому випадку 1dp = 1px. Якщо розмір екрану більше або менше, ніж 160dpi, кількість пікселів, які застосовуються для відтворення 1dp відповідно збільшується або зменшується. Наприклад, на екрані з 240 dpi 1dp = 1,5px, а на екрані з 320dpi 1dp = 2px. Загальна формула для отримання кількості фізичних пікселів з dp: px = dp * (dpi / 160)

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

Припускають настройку розмірів, вироблену користувачем. Рекомендуються для роботи зі шрифтами.

• pt: 1/72 дюйма, базуються на фізичних розмірах екрану

• mm: міліметри

• in: дюйми

Кращими одиницями для використання є dp. Це пов'язано з тим, що світ мобільних пристроїв на Android сильно фрагментований в плані роздільної здатності і розмірів екрану. І чим більше щільність пікселів на дюйм, тим відповідно більше пікселів нам буде доступно (рис. 4):

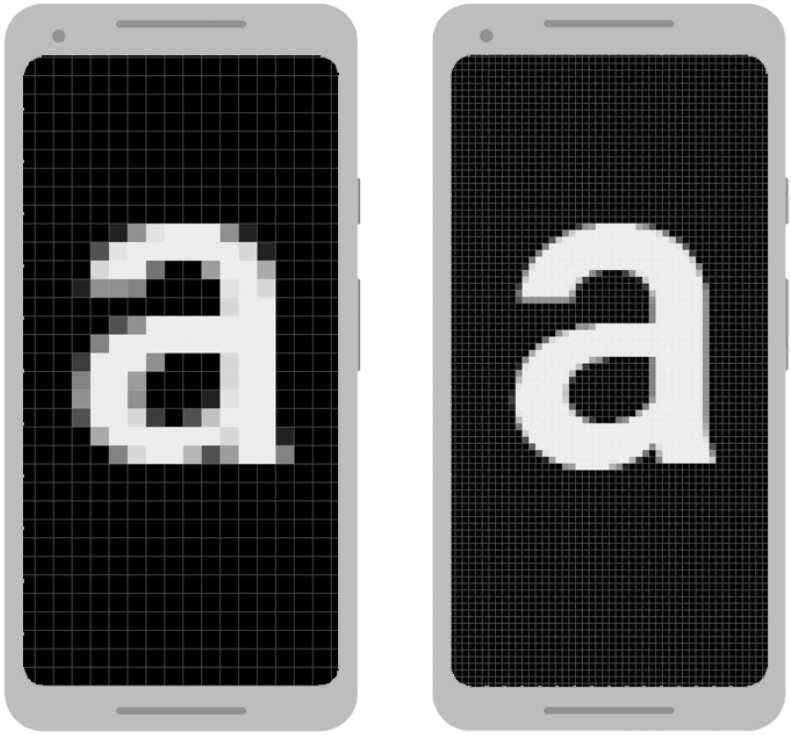


Рисунок 4 – Екрани з різною роздільною здатністю

Для спрощення роботи з розмірами всі розміри розбиті на кілька груп:

• ldpi (low): ~ 120dpi

• mdpi (medium): ~ 160dpi

• hdpi (high): ~ 240dpi (до цієї групи відносять Nexus One)

• xhdpi (extra-high): ~ 320dpi (Nexus 4)

• xxhdpi (extra-extra-high): ~ 480dpi (Nexus 5 / 5X, Samsung Galaxy S5)

• xxxhdpi (extra-extra-extra-high): ~ 640dpi (Nexus 6/6P, Samsung Galaxy S6)

Ширина і висота елементів

Всі візуальні елементи, які ми використовуємо в додатку, як правило, упорядковуються на екрані за допомогою контейнерів. В Android подібними контейнерами служать такі класи як RelativeLayout, LinearLayout, GridLayout, TableLayout, ConstraintLayout, FrameLayout. Всі вони по різному розташовують елементи і керують ними, але є деякі загальні моменти при компонуванні візуальних компонентів, які ми зараз розглянемо.

Для організації елементів всередині контейнера використовуються параметри розмітки. Для їх завдання в файлі xml використовуються атрибути, які починаються з префікса layout_. Зокрема, до таких параметрів

належать атрибути layout_height і layout_width, які використовуються для установки розмірів і можуть приймати одне з наступних значень:

• точні розміри елемента, наприклад ***96 dp***;

• значення ***wrap_content***: елемент розтягується до тих меж, які є достатніми, щоб вмістити весь його вміст;

• значення ***match_parent***: елемент заповнює всю область батьківського контейнера.

Програмна установка ширини і висоти

Якщо елемент, наприклад, той же TextView створюється в коді java, то для установки висоти і ширини можна використовувати метод setLayoutParams(). Так, змінимо код MainActivity:

```
package com.example.myapplication;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.RelativeLayout;
import android.widget.TextView;

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

  RelativeLayout relativeLayout = new RelativeLayout(this);
  TextView textView1 = new TextView(this);
  textView1.setText("Hello Android 7");
  textView1.setTextSize(26);

  // встановлюємо розміри
  textView1.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
  // додаємо TextView в RelativeLayout
```

```
  relativeLayout.addView(textView1);
  setContentView(relativeLayout);
 }
}
```

У метод setLayoutParams() передається об'єкт ViewGroup.LayoutParams. Цей об'єкт ініціалізується двома параметрами: шириною і висотою. Для вказівки ширини і висоти можна використовувати одну з констант ViewGroup.LayoutParams.WRAP_CONTENT або ViewGroup.LayoutParams.MATCH_PARENT.

Також ми можемо передати точні значення або комбінувати типи значень:

```
textView1.setLayoutParams(new ViewGroup.LayoutParams(
ViewGroup.LayoutParams.MATCH_PARENT, 200));
```

Лабораторна робота №4.

Тема «Створення інтерфейсу засобами LinearLayout, RelativeLayout, TableLayout, FrameLayout, GridLayout, ConstraintLayout та ScrollView»

Мета: отримати навички роботи з контейнерами LinearLayout, RelativeLayout, TableLayout, FrameLayout, GridLayout, ConstraintLayout, ScrollView.

Звіт повинен містити:

1. Постановка завдання.
2. Теоретичні відомості по проекту.
3. Результати виконання завдання.
4. Висновок.

Постановка завдання:

1. Створити 3 проекти мобільних додатків для реалізації дизайну (рис. 1) трьома обраними контейнерами (наприклад: LinearLayout, RelativeLayout, GridLayout):

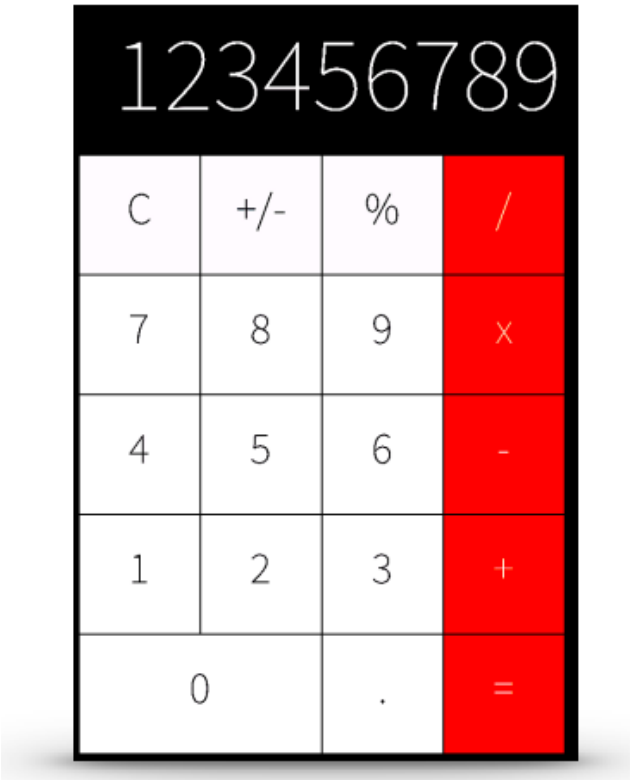


Рисунок 1 – Зовнішній вигляд додатку

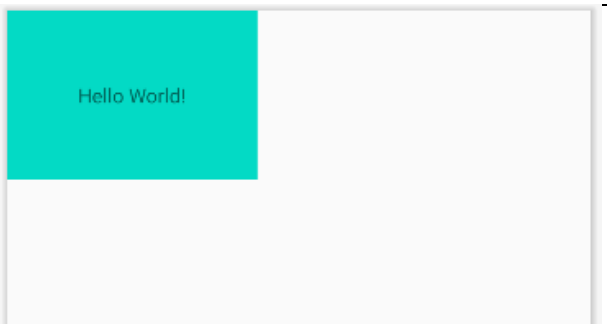
Теоретичні відомості

Параметри розмітки дозволяють задати відступи як від зовнішніх кордонів елемента до кордонів контейнера, так і всередині самого елемента між його межами і вмістом.

Padding

Для установки внутрішніх відступів застосовується атрибут android: padding. Він встановлює відступи контенту від усіх чотирьох сторін контейнера. Можна встановлювати відступи тільки від однієї сторони контейнера, застосовуючи такі атрибути: android:paddingLeft, android: paddingRight, android:paddingTop і android:paddingBottom.

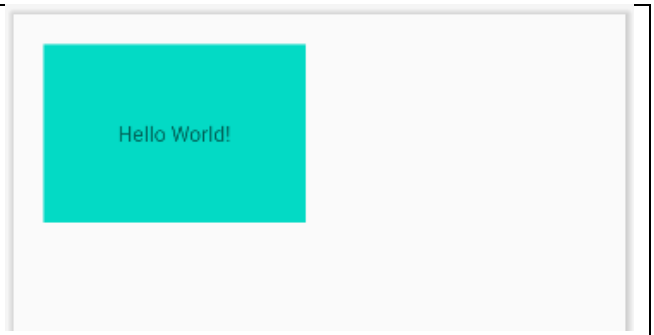
```
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="#FF03DAC5"
android:padding="50dp"
android:text="Hello World!" />
```



Margin

Для установки зовнішніх відступів використовується атрибут layout_margin. Даний атрибут має модифікації, які дозволяють задати відступ тільки від однієї сторони: android: layout_marginBottom, android: layout_marginTop, android: layout_marginLeft і android: layout_marginRight (відступи відповідно від нижньої, верхньої, лівої і правої меж):

```
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="#FF03DAC5"
android:layout_marginTop="20dp"
android:layout_marginLeft="20dp"
android:padding="50dp"
android:text="Hello World!" />
```



Програмне встановлення Padding та Margin

Для програмної установки внутрішніх відступів у елементи визивається метод setPadding (left, top, right, bottom), в який передаються чотири значення для кожної зі сторін. Для установки зовнішніх відступів необхідно реалізувати об'єкт LayoutParams для того контейнера, який застосовується. І потім викликати у цього об'єкта LayoutParams метод setMargins (left, top, right, bottom):

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.RelativeLayout;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

  RelativeLayout relativeLayout = new RelativeLayout(this);
  TextView textView1 = new TextView(this);
  textView1.setText("Hello Android 7");
  textView1.setTextSize(26);

  RelativeLayout.LayoutParams layoutParams = new
RelativeLayout.LayoutParams(
              ViewGroup.LayoutParams.MATCH_PARENT, 200);
  // встановлення зовнішніх відступів
  layoutParams.setMargins(30,40,50,60);
  textView1.setLayoutParams(layoutParams);
  // встановлення внутрішніх відступів
  textView1.setPadding(30,30,30,30);
  // додаємо TextView в RelativeLayout
  relativeLayout.addView(textView1);
  // додаємо TextView в RelativeLayout
```

```
  setContentView(relativeLayout);
 }
}
```

LinearLayout

Контейнер LinearLayout представляє об'єкт ViewGroup, який впорядковує всі дочірні елементи в одному напрямку: по горизонталі або по вертикалі. Всі елементи розташовані один за іншим. Напрямок розмітки вказується за допомогою атрибута *android:orientation*.

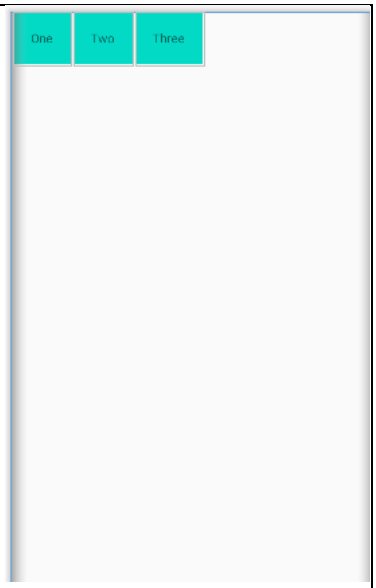
Якщо, наприклад, орієнтація розмітки вертикальна (android: orientation = "vertical"), то всі елементи розташовуються в стовпчик – по одному елементу на кожному рядку. Якщо орієнтація горизонтальна (android: orientation = "horizontal"), то елементи розташовуються в один рядок.

Наприклад, розташуємо елементи в горизонтальний ряд:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/andr
oid"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal"
 tools:context=".MainActivity">

 <TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:background="#FF03DAC5"
  android:layout_marginLeft="3dp"
  android:padding="20dp"
  android:text="One" />

 <TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:background="#FF03DAC5"
```



```
  android:layout_marginLeft="3dp"
  android:padding="20dp"
  android:text="Two" />

 <TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:background="#FF03DAC5"
  android:layout_marginLeft="3dp"
  android:padding="20dp"
  android:text="Three" />

</LinearLayout>
```

LinearLayout підтримує таку властивість, як вага (weight) елемента, яка передається атрибутом android: layout_weight. Це властивість приймає значення, яке вказує, яку частину залишку вільного місця контейнера по відношенню до інших об'єктів займе даний елемент. Наприклад, якщо один елемент у нас буде мати для властивості android: layout_weight значення 2, а інший - значення 1, то в сумі вони дадуть 3, тому перший елемент буде займати 2/3 простору, що залишилося, а другий - 1/3.

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

Продемонструємо на наступному прикладі з вертикальною орієнтацією елементів контейнера LinearLayout:

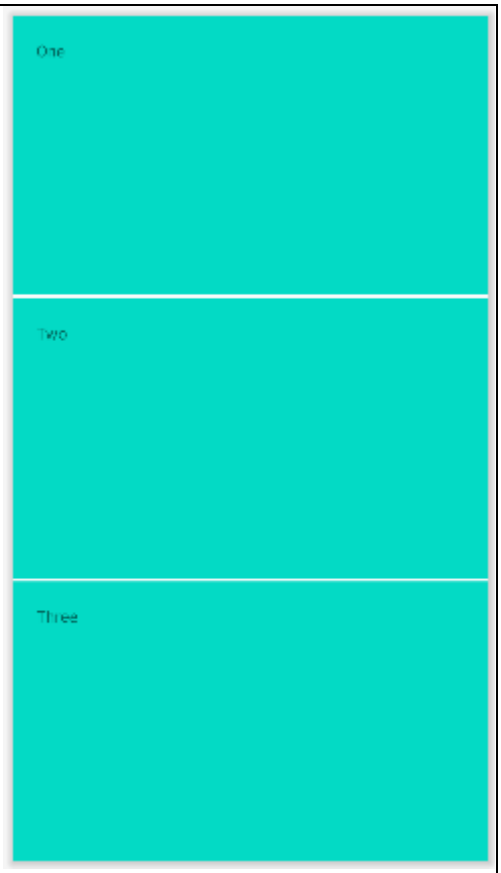
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/
res/android"
xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context=".MainActivity">

 <TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="#FF03DAC5"
  android:layout_weight="1"
  android:padding="20dp"
  android:text="One" />

 <TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="#FF03DAC5"
  android:layout_weight="1"
  android:layout_marginTop="3dp"
  android:padding="20dp"
  android:text="Two" />

 <TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="#FF03DAC5"
  android:layout_weight="1"
  android:layout_marginTop="3dp"
  android:padding="20dp"
  android:text="Three" />

</LinearLayout>
```



При цьому так як у нас вертикальний стек, то ми можемо також встановити для властивості layout_height значення 0dp. Якби LinearLayout

мав горизонтальну орієнтацію, то для властивості layout_width можна було б встановити значення 0dp.

Ще один атрибут **android:weightSum** дозволяє вказати суму ваг усіх елементів. LinearLayout тут задає суму ваг рівну 7. Тобто весь простір по вертикалі (так як вертикальна орієнтація) умовно ділиться на сім рівних частин. Перший TextView має вагу 1, тобто з цих семи частин займає тільки одну. Другий TextView має вагу 2, тобто займає дві частини із семи. І третій має вагу 3. Підсумкова сума становить 6. Але так як LinearLayout задає вагу 7, то одна частина буде вільна від усіх елементів.

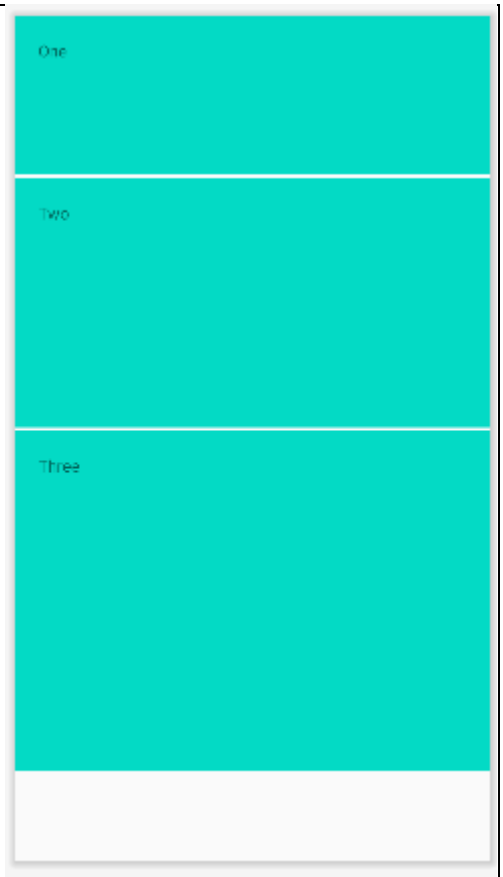
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/
res/android"

xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:weightSum="7"
 tools:context=".MainActivity">

 <TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="#FF03DAC5"
  android:layout_weight="1"
  android:padding="20dp"
  android:text="One" />

 <TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="#FF03DAC5"
  android:layout_weight="2"
  android:layout_marginTop="3dp"
  android:padding="20dp"
  android:text="Two" />

 <TextView
```



```
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="#FF03DAC5"
  android:layout_weight="3"
  android:layout_marginTop="3dp"
  android:padding="20dp"
  android:text="Three" />

</LinearLayout>
```

TableLayout

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

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TableRow>
    <TextView
      android:layout_weight="0.5"
      android:text="Логин"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

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

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

    <EditText
      android:layout_weight="1"
```

```
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
  </TableRow>
</TableLayout>
```

Використовуючи елемент TableRow, ми створюємо окремий рядок. Як розмітка дізнається скільки стовпців треба створити? Android знаходить рядок з максимальною кількістю віджетів одного рівня, і ця кількість буде означати кількість стовпців. Наприклад, в даному випадку у нас визначені два рядки і в кожній по два елементи. Якби в якій-небудь з них було б три віджета, то відповідно стовпців було б також три, навіть якщо в іншому рядку залишилося б два віджети.

Причому елемент TableRow успадковується від класу LinearLayout, тому ми можемо до нього застосовувати той же функціонал, що і до LinearLayout. Зокрема, для визначення простору для елементів в рядку використовується атрибут android: layout_weight.

Якщо якийсь елемент повинен бути розтягнутий на ряд стовпців, то ми можемо розтягнути його за допомогою атрибута layout_column, який вказує на скільки стовпців треба розтягнути елемент:

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TableRow>
    <TextView
      android:textSize="22sp"
      android:text="Логін"
      android:layout_width="100dp"
      android:layout_height="wrap_content" />

    <EditText
      android:textSize="22sp"
      android:layout_width="200dp"
      android:layout_height="wrap_content" />
  </TableRow>

  <TableRow>
    <TextView
```

```
      android:textSize="22sp"
      android:text="Email"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

    <EditText
      android:textSize="22sp"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
  </TableRow>
  <TableRow>
    <Button
      android:text="Надіслати"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
    android:layout_span="2"/>
  </TableRow>
</TableLayout>
```

RelativeLayout

RelativeLayout представляє об'єкт ViewGroup, який має в своєму розпорядженні дочірні елементи щодо позиції інших дочірніх елементів розмітки або щодо області самої розмітки RelativeLayout. Використовуючи відносне позиціонування, ми можемо встановити елемент по правому краю або в центрі або іншим способом, який надає даний контейнер. Для установки елемента в файлі xml ми можемо застосовувати такі атрибути:

• *android: layout_above:* має елемент над елементом із зазначеним Id

• *android: layout_below:* має елемент під елементом із зазначеним Id

• *android: layout_toLeftOf:* розташовується зліва від елемента з вказаним Id

• *android: layout_toRightOf:* розташовується праворуч від елемента з вказаним Id

• *android: layout_alignBottom*: вирівнює елемент по нижній межі іншого елемента із зазначеним Id

• *android: layout_alignLeft*: вирівнює елемент по лівій межі іншого елемента із

зазначеним Id

• *android: layout_alignRight*: вирівнює елемент по правій межі іншого елемента із зазначеним Id

• *android: layout_alignTop*: вирівнює елемент по верхній межі іншого елемента із зазначеним Id

• *android: layout_alignBaseline*: вирівнює базову лінію елемента за базовою лінії іншого елемента із зазначеним Id

• *android: layout_alignParentBottom*: якщо атрибут має значення true, то елемент притискається до нижньої межі контейнера

• *android: layout_alignParentRight*: якщо атрибут має значення true, то елемент притискається до правого краю контейнера

• *android: layout_alignParentLeft*: якщо атрибут має значення true, то елемент притискається до лівого краю контейнера

• *android: layout_alignParentTop*: якщо атрибут має значення true, то елемент притискається до верхньої межі контейнера

• *android: layout_centerInParent*: якщо атрибут має значення true, то елемент розташовується по центру батьківського контейнера

• *android: layout_centerHorizontal*: при значенні true вирівнює елемент по центру по горизонталі

• *android: layout_centerVertical*: при значенні true вирівнює елемент по центру по вертикалі Наприклад, позиціонування відносно контейнера RelativeLayout.

Лабораторна робота №5.

Тема: «**Адаптери та списки»**

Мета: отримати навички роботи з адаптерами та списками.

Звіт повинен містити:

1. Постановка завдання.
2. Теоретичні відомості по проекту.
3. Результати виконання завдання.
4. Висновок.

Постановка завдання:

1. Створити мобільний додаток журналу обліку успішності з предмету, що дає можливість перегляду, додавання, редагування та видалення інформації про студента та його оцінок.
2. Описати хід роботи.
3. Зробити висновки.

Теоретичні відомості

Android представляє широку палітру елементів, які представляють списки. Всі вони є спадкоємцями класу android.widget.AdapterView. Це такі віджети як ListView, GridView, Spinner. Вони можуть виступати контейнерами для інших елементів управління. Ієрархія класів зображена на рисунку 1.

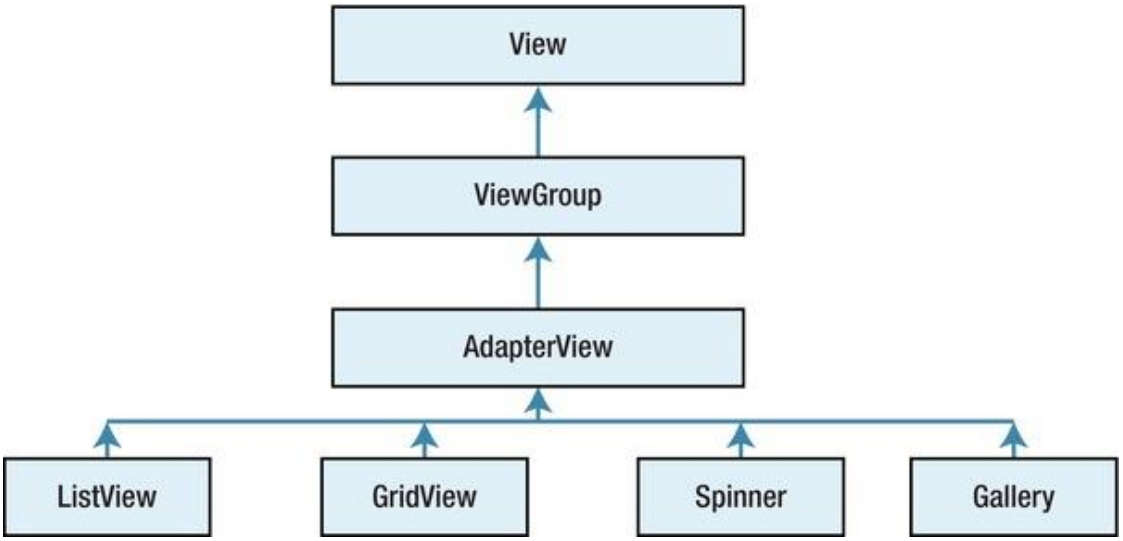


Рисунок 1 – Ієрархія класів

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

ArrayAdapter

Клас ArrayAdapter є простим адаптером, який пов'язує масив даних з набором елементів TextView, з яких, наприклад, може складатися ListView. Тобто в даному випадку джерелом даних виступає масив об'єктів. ArrayAdapter викликає у кожного об'єкта метод toString () для приведення до строкового вигляду отриманий рядок встановлює в елемент TextView. Отже, розмітка програми може виглядати так:

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

```
 <ListView android:id="@+id/countriesList"
  android:layout_height="match_parent"
  android:layout_width="match_parent" />

</LinearLayout>
```

Ми визначили елемент ListView, який буде виводити список об'єктів. Тепер перейдемо до коду activity і зв'яжемо ListView через ArrayAdapter з деякими даними:

```
package com.example.listview;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import androidx.appcompat.app.AppCompatActivity;

public class ListExample extends AppCompatActivity {
 // набір даних, який зв'яжемо зі списком
 String[] countries = { "Бразилія", "Аргентина",
   "Колумбія", "Чілі", "Уругвай"};
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  // отримуємо елемент ListView
  ListView countriesList = (ListView)
    findViewById(R.id.countriesList);
  // створюємо адаптер
  ArrayAdapter<String> adapter = new ArrayAdapter(this,
    android.R.layout.simple_list_item_1,
    countries);
  // встановлюємо для списку адаптер
  countriesList.setAdapter(adapter);
 }
}
```

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

• this: поточний об'єкт activity

• android.R.layout.simple_list_item_1: файл розмітки списку, який фреймворк представляє за замовчуванням. Він знаходиться в папці Android SDK шляхом platforms / [android-номер_версії] / data / res / layout. Якщо нас не задовольняє стандартна розмітка списку, ми можемо створити свою і потім в коді змінити id на id потрібної нам розмітки

• countries: масив даних. Тут необов'язково вказувати саме масив, це може бути список ArrayList <T>. В кінці необхідно встановити для ListView адаптер за допомогою методу setAdapter().

У підсумку ми отримаємо наступне відображення (рис. 2).

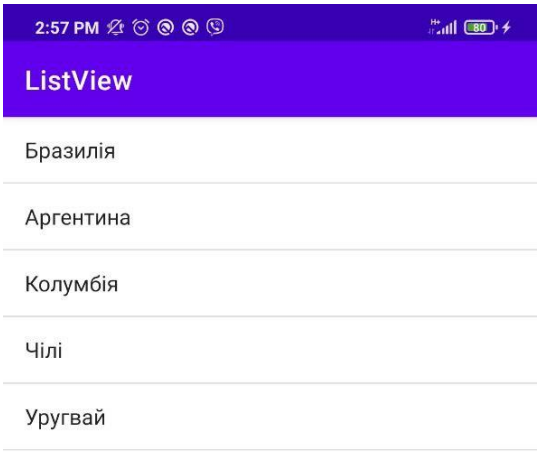


Рисунок 2 – Результат відображення списку

Ресурс string-array і ListView

У вище наведеному прикладі було розглянуто, як виводити масив рядків за допомогою ArrayAdapter в ListView. При цьому масив рядків визначався програмно в коді java. Однак подібний масив рядків набагато зручніше було б зберігати в файлі xml у вигляді ресурсу.

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

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string-array name="i'мя_масиву_рядків">
    <item>елемент</item>
  </string-array>
</resources>
```

Масив рядків задається за допомогою елемента <string-array>, атрибут name якого може мати довільне значення, за яким потім будуть посилатися на цей масив. Всі елементи масиву представляють набір значень <item> Наприклад, додамо в папку res / values новий файл. Для цього натиснемо правою кнопкою миші на даний каталог і меню виберемо пункт New -> Value resource file:

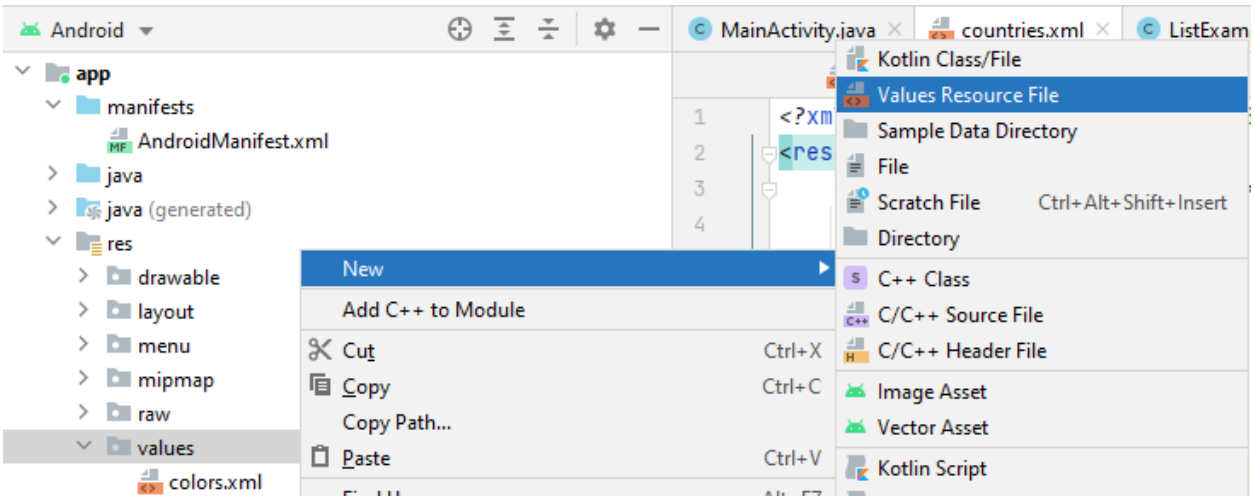


Рисунок 3 – Створення ресурсного файлу значень

У вікні, що з’явиться, дамо назву файлу як countries:

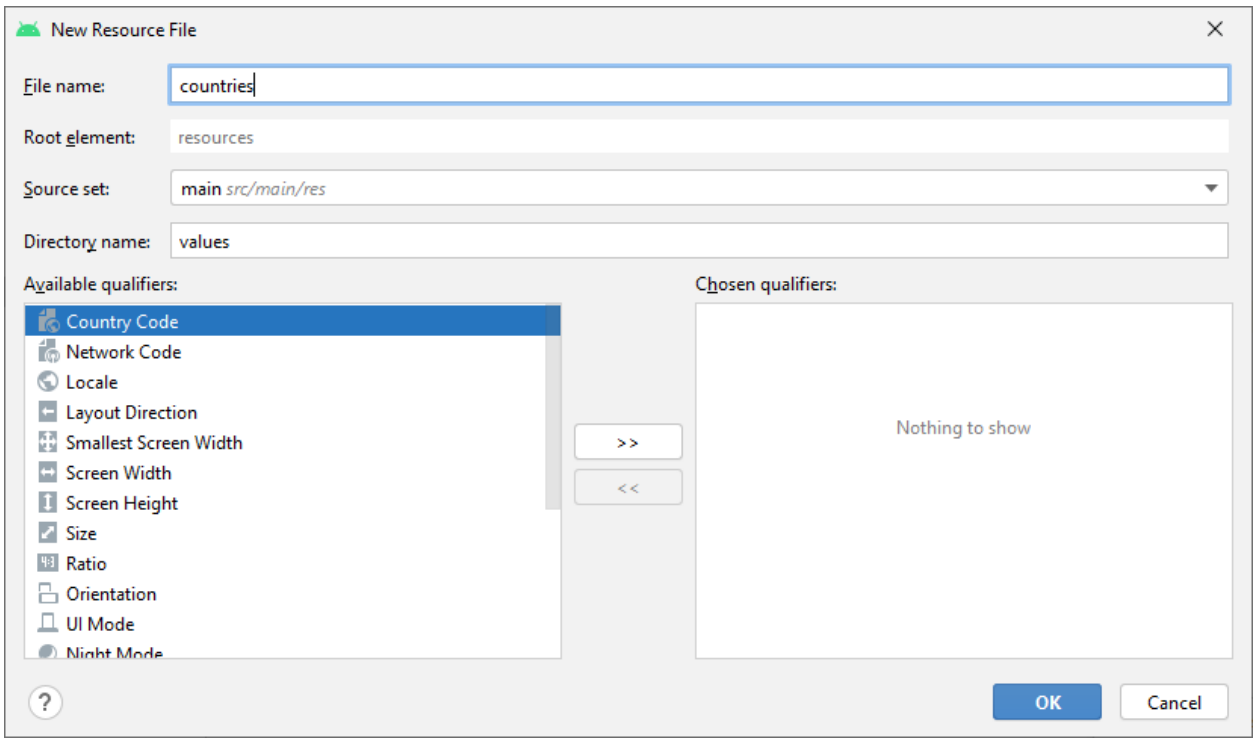


Рисунок 4 –Ресурсний файл країн

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

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string-array name="countries">
    <item>Бразилія</item>
    <item>Аргентина</item>
    <item>Колумбія</item>
    <item>Чілі</item>
    <item>Уругвай</item>
    <item>Україна</item>
  </string-array>
</resources>
```

Для отримання ресурсу в коді java застосовується вираз R.**array**.*назва_ресурсу*.

```
package com.example.listview;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import androidx.appcompat.app.AppCompatActivity;
```

```
public class ListExample extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // отримуємо елемент ListView
    ListView countriesList = (ListView)
        findViewById(R.id.countriesList);
    // отримуємо ресурс
    String[] countries =
        getResources().getStringArray(R.array.countries);

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

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

Але нам необов'язково додавати список рядків в ListView програмно. У цього елемента є атрибут entries, який в якості значення може приймати ресурс string-array:

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

  <ListView
```

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

В цьому випадку код ми можемо скоротити до стандартного:

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

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

    }
}
```

Результат не зміниться.

Вибір елемента в ListView

Крім простого виведення списку елементів ListView дозволяє вибирати елемент і обробляти його вибір. Розглянемо, як це зробити. Визначимо наступну розмітку в файлі activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
  <TextView
    android:id="@+id/selection"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/teal_200"
    android:textSize="22sp" />
  <ListView
    android:id="@+id/countriesList"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent" />
</LinearLayout>
```

Тепер зв'яжемо список ListView з джерелом даних і закріпимо за ним слухач натискання на елемент списку:

```
package com.example.listview;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class ListExample extends AppCompatActivity {

  String[] countries = {"Бразилія", "Аргентина",
      "Колумбія", "Чілі", "Уругвай"};
  private TextView selection;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // отримуємо елемент TextView
    selection = (TextView) findViewById(R.id.selection);
    // тримуємо елемент ListView
    ListView countriesList = (ListView)
        findViewById(R.id.countriesList);
    // створюємо адаптер
    ArrayAdapter<String> adapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_1,
        countries);
    // встановлюємо для списка адаптер
    countriesList.setAdapter(adapter);
    // додаємо для списку слухач
    countriesList.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
      @Override
      public void onItemClick(AdapterView<?> parent, View view, int position,
```

```
long id) {
        // по позиції отримуємо обраний елемент
        String selectedItem = countries[position];
        // встановлення тексту елемента TextView
        selection.setText(selectedItem);
      }
    });
  }
}
```

Отже, метод setAdapter пов'язує елемент ListView з певним адаптером. Далі для обробки вибору елемента списку встановлюється слухач OnItemClickListener.

Цей слухач має один метод onItemClick, через параметри якого ми можемо отримати виділений елемент і супутні дані. Так, він приймає такі параметри:

• parent: натиснутий елемент AdapterView (в ролі якого в даному випадку виступає наш елемент ListView)

• view: натиснутий віджет всередині AdapterView

• position: індекс натиснутого віждета всередині AdapterView

• id: ідентифікатор рядка натиснутого елемента

У підсумку, отримуючи індекс натиснутого віджета, який відповідає індексу елемента в масиві рядків, ми встановлюємо його текст як текст елемента TextView (`selection.setText(countries[position])`).

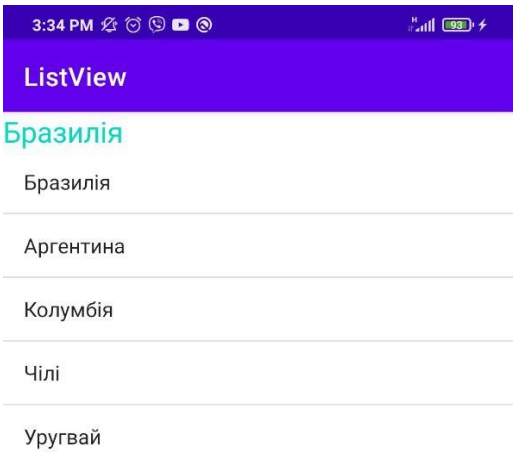


Рисунок 4 – Відображення списку країн

Додавання і видалення в ListView

Після прив'язки ListView до джерела даних через адаптер ми можемо працювати з даними – додавати, видаляти, змінювати тільки через адаптер.

ListView служить тільки для відображення даних. Для управління даними ми можемо використовувати методи адаптера або безпосередньо джерела даних. Наприклад, за допомогою методу add() класу ArrayAdapter можна додати новий елемент в кінець масиву-джерела даних.

Метод insert() дозволяє додати нове значення за певним індексом, а метод remove() дозволяє видалити об'єкт з масиву. За допомогою методу sort() можна провести сортування масиву.

Однак після застосування вищевказаних методів зміни торкнуться тільки масиву, який виступає джерелом даних. Щоб синхронізувати зміни з елементом ListView, треба викликати у адаптера метод notifyDataSetChanged().

Лабораторна робота №6.

Тема: «Фрагменти, багатопоточність та асинхронність»

Мета: отримати навички роботи з фрагментами, потоками, повідомленнями та асинхронними завданнями.

Звіт повинен містити:

1. Постановка завдання.
2. Теоретичні відомості по проекту.
3. Результати виконання.
4. Висновок.

Постановка завдання:

1. Створити додаток, який складається з двох фрагментів:

   Перший фрагмент – список, другий – детальна інформація по елементу списку. При натисненні на елемент списку у фрагменті з детальною інформацією необхідно емулювати завантаження файлу. Показувати прогрес завантаження в фрагменті з детальною інформацією.
2. Описати хід роботи.
3. Зробити висновки.

Теоретичні відомості

Організація додатку на основі декількох activity не завжди може бути оптимальною. Світ ОС Android досить сильно фрагментований і складається з багатьох пристроїв. І якщо для мобільних апаратів з невеликими екранами взаємодія між різними activity виглядає досить непогано, то на великих екранах – планшетах, телевізорах вікна activity виглядали б не дуже в силу великого розміру екрана. Власне тому і з'явилася концепція фрагментів.

Фрагмент існує в контексті activity і має свій життєвий цикл, поза activity він існувати не може. Кожна activity може мати кілька фрагментів.

Для початку роботи з фрагментами створимо новий проект з порожньою MainActivity. І спочатку створимо перший фрагмент. По-перше, фрагменти містять ті ж елементи управління, що і activity. Тому створимо в папці res / layout новий файл fragment_content.xml:

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

Тут визначені кнопка і текстове поле, які будуть складати інтерфейс фрагмента.

Тепер створимо сам клас фрагмента. Для цього додамо в одну папку з MainActivity новий клас. Для цього натиснемо на папку правою кнопкою миші і виберемо в меню New -> Java Class. Назвемо новий клас ContentFragment і визначимо у нього такий зміст:

```
package com.example.listview;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import androidx.fragment.app.Fragment;

import java.util.Date;
```

```
public class ContentFragment extends Fragment {

  @Override
  public View onCreateView(LayoutInflater inflater,
              ViewGroup container,
              Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_content, container,false);

    Button updateButton = (Button)
        view.findViewById(R.id.updateButton);
    final TextView updateBox = (TextView)
        view.findViewById(R.id.textBox);

    updateButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        String curDate = new Date().toString();
        updateBox.setText(curDate);
      }
    });

    return view;
  }
}
```

Клас фрагмента повинен успадковуватися від класу Fragment. Для створення візуального інтерфейсу фрагмент перевизначає батьківський метод onCreateView(). Він приймає три параметри:

Об'єкт LayoutInflater використовується для установки ресурсу розмітки для створення інтерфейсу

Параметр ViewGroup container встановлює контейнер інтерфейсу. Параметр Bundle savedInstanceState передає раніше збережений стан. Для створення інтерфейсу застосовується метод inflate() об'єкта LayoutInflater. Він отримує ресурс розмітки layout для даного фрагмента, контейнер, в який буде укладений інтерфейс, і третій булевий параметр вказує, чи треба прикріплювати розмітку до контейнера з другого параметра. І як і в activity, тут ми можемо встановлювати обробники дял елементів управління,

обробляти їх події. В даному випадку в текстовому полі виводиться поточна дата.

Також варто відзначити, що в Android Studio є спеціальний шаблон для додавання фрагмента. Для його використання також треба натиснути на папку пакета класів правою кнопкою миші і через меню, що випадає вибрати (рис. 1): New -> Fragment -> Fragment (Blank):

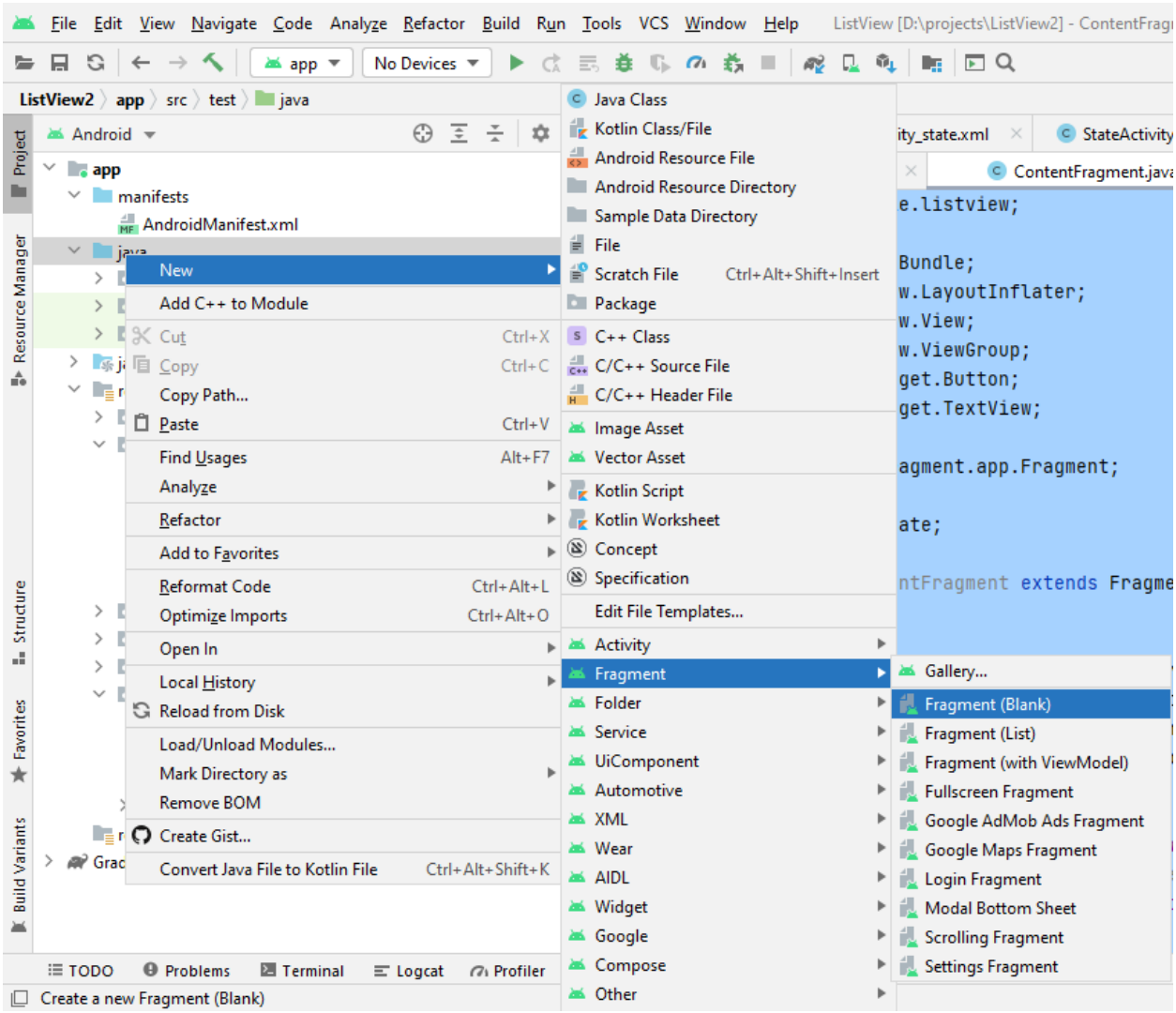


Рисунок 1 – Створення нового фрагменту

Даний шаблон відразу додасть в проект і клас фрагмента і пов'язаний з ним клас розмітки інтерфейсу. І в кінці треба додати фрагмент в activity. Для цього змінимо файл activity_main.xml, який визначає інтерфейс для MainActivity:

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

```
  android:layout_height="match_parent" >

  <fragment android:id="@+id/listFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name=" com.example.fragments.ContentFragment"/>

</RelativeLayout>
```

Кожен фрагмент задається за допомогою елемента <fragment>. Для кожного фрагмента має бути встановлено висота, ширина, id, а також ім'я. Як ім'я встановлюється повне ім'я класу з урахуванням пакета:

```
android: name = "com.example.fragments.ContentFragment"
```

Крім фрагмента ми можемо додати в розмітку activity_main.xml інші елементи або фрагменти, але в даному випадку обмежимося одним фрагментом.

Код класу MainActivity залишається тим же, що і при створенні проекту. Якщо ми запустимо додаток, то ми побачимо фактично той же самий інтерфейс, який ми могли б зробити і через activity, тільки в даному випадку інтерфейс буде визначено у фрагменті ():

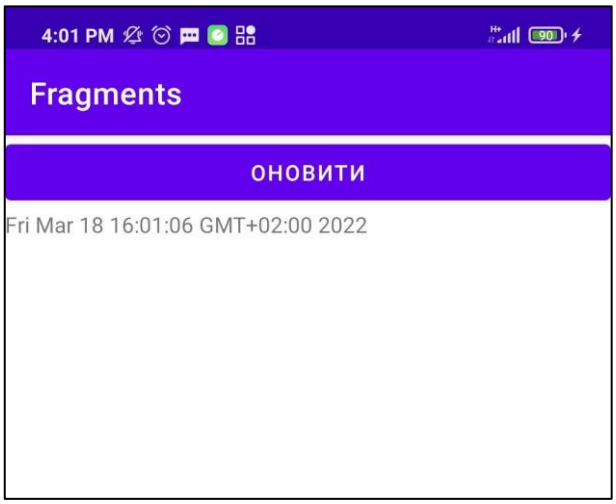


Рисунок 2 – Інтерфейс з використанням фрагментів

Додавання фрагмента в коді

Крім визначення фрагмента в xml-файлі інтерфейсу ми можемо додати його динамічно в activity.

Для цього змінимо файл activity_main.xml:

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

</LinearLayout>
```

І також змінимо клас MainActivity:

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

public class MainActivity extends AppCompatActivity {

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

    if (savedInstanceState == null) {
      getSupportFragmentManager().beginTransaction()
          .add(R.id.container, new ContentFragment())
          .commit();
    }
  }
}
```

Метод getSupportFragmentManager () повертає об'єкт FragmentManager, який управляє фрагментами. Але в даному випадку треба зробити уточнення. В даному коді використовується фрагмент – спадкоємець класу android.support.v4.app.Fragment. Але для створення фрагментів ми також можемо використовувати клас android.app.Fragment з API 11.

У разі використання android.app.Fragment менеджер фрагментів викликається за допомогою методу getFragmentManager(). Об'єкт FragmentManager за допомогою методу beginTransaction() створює об'єкт

FragmentTransaction. FragmentTransaction виконує два методи: add() і commit(). Метод add() додає фрагмент: add (R.id.container, new ContentFragment()) - першим аргументом передається ресурс розмітки, в який треба додати фрагмент. І метод commit() підтверджуючого і завершує операцію додавання.

Підсумковий результат такого додавання фрагмента буде тим же, що і при явному визначенні фрагмента через елемент <fragment> в розмітці інтерфейсу.

Одна activity може використовувати кілька фрагментів, наприклад, з одного боку список, а з іншого - детальний опис вибраного елементу списку. У такій конфігурації activity використовує два фрагмента, які між собою повинні взаємодіяти. Розглянемо базові принципи взаємодії фрагментів в додатку.

Створимо новий проект з порожньою MainActivity. Далі створимо розмітку layout для фрагментів. Нехай у нас в додатку буде два фрагмента. Додамо в папку res / layout новий xml-файл fragment_list.xml:

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

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

І також додамо для іншого фрагмента файл розмітки fragment_detail.xml:

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

```
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <TextView
    android:id="@+id/detailsText"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:layout_marginTop="16dp"
    android:text="Привіт світ"
    android:textSize="20sp" />
</LinearLayout>
```

Обидва фрагмента будуть гранично простими: один буде містити кнопку, а другий – текстової поле. Потім додамо в проект в одну папку з MainActivity власне класи фрагментів. Додамо новий клас ListFragment наступного змісту:

```
package com.example.fragments;

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import androidx.fragment.app.Fragment;

import java.util.Date;

public class ListFragment extends Fragment {

  private OnFragmentInteractionListener mListener;

  @Override
  public View onCreateView(LayoutInflater inflater,
              ViewGroup container,
```

```
                Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_list, container, false);
    Button button = (Button)
        view.findViewById(R.id.update_button);
    // задаємо обрабник кнопки
    button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        updateDetail();
      }
    });
    return view;
  }

  interface OnFragmentInteractionListener {

    void onFragmentInteraction(String link);
  }

  @Override
  public void onAttach(Context context) {
    super.onAttach(context);
    try {
      mListener = (OnFragmentInteractionListener) context;
    } catch (ClassCastException e) {
      throw new ClassCastException(context.toString()
          + " повинен реалізовувати інтерфейс
OnFragmentInteractionListener");
    }
  }

  public void updateDetail() {
    // генеруємо деякі дані
    String curDate = new Date().toString();
    // Відсилаємо дані Activity
    mListener.onFragmentInteraction(curDate);
  }
}
```

Тут визначається обробник натискання кнопки - в даному випадку метод updateDetail, завдання якого - взаємодія з другим фрагментом.

Фрагменти не можуть безпосередньо взаємодіяти між собою. Для цього треба звертатися до контексту, в якості якого виступає клас Activity. Для звернення до activity, як правило, створюється вкладений інтерфейс. В даному випадку він називається OnFragmentInteractionListener з одним методом.

При обробці натискання кнопки в метод updateDetail () передається в цей метод деяке строкове значення:

```
private OnFragmentInteractionListener mListener;
…
public void updateDetail() {
    // генеруємо деякі дані
    String curDate = new Date().toString();
    // Відсилаємо дані Activity
    mListener.onFragmentInteraction(curDate);
  }
```

Але щоб взаємодіяти з іншим фрагментом через activity, нам треба прикріпити поточний фрагмент до activity. Для цього в класі фрагмента визначено метод onAttach (Context context). У ньому відбувається установка об'єкта OnFragmentInteractionListener:

```
mListener = (OnFragmentInteractionListener) context;
```

Тепер визначимо клас для другого фрагмента. Назвемо його DetailFragment:

```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import android.app.Fragment;

public class DetailFragment extends Fragment {

  @Override
  public View onCreateView(LayoutInflater inflater,
              ViewGroup container,
              Bundle savedInstanceState) {
```

```
    View view = inflater.inflate(R.layout.fragment_detail, container, false);
    return view;
  }

  // обновление текстового поля
  public void setText(String item) {
    TextView view = (TextView)
        getView().findViewById(R.id.detailsText);
    view.setText(item);
  }
}
```

Завдання цього фрагмента – виведення деякої інформації. Так як він не повинен передавати ніяку інформацію іншому фрагменту, тут ми можемо обмежитися тільки перевизначенням методу onCreateView (), який в якості візуального інтерфейсу встановлює розмітку з файлу fragment_detail.xml

Але щоб імітувати взаємодію між двома фрагментами, тут також визначено метод setText (), який оновлює текст на текстовому полі. Тепер змінимо файл розмітки activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity"
  android:orientation="horizontal">

  <fragment
    android:id="@+id/listFragment"
    android:layout_width="wrap_content"
    android:layout_weight="1"
    android:layout_height="match_parent"
    android:name="com.example.fragments.ListFragment"/>
  <fragment
    android:id="@+id/detailFragment"
    android:layout_width="wrap_content"
    android:layout_weight="2"
    android:layout_height="match_parent"
```

```
    android:name="com.example.fragments.DetailFragment"/>

</LinearLayout>
```

І в кінці змінимо код MainActivity:

```
package com.example.fragments;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity
              implements ListFragment.OnFragmentInteractionListener {

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

  @Override
  public void onFragmentInteraction(String link) {
    DetailFragment fragment;
    fragment = (DetailFragment)
getFragmentManager().findFragmentById(R.id.detailFragment);

    if (fragment != null && fragment.isInLayout()) {
      fragment.setText(link);
    }
  }
}
```

Для взаємодії фрагмента ListFragment з іншим фрагментом через MainActivity треба, щоб ця activity реалізовувала інтерфейс OnFragmentInteractionListener. Для цього реалізуємо метод onFragmentInteraction (), який отримує фрагмент DetailFragment і викликає у нього метод setText () У підсумку вийде, що при натисканні кнопки на фрагменті ListFragment буде спрацьовувати метод updateDetail(), який викличе метод mListener.onFragmentInteraction (newTime). mListener

встановлюється як activity, тому при цьому буде викликаний метод setText у фрагмента DetailFragment.

Таким чином, відбудеться взаємодія між двома фрагментами. Якщо ми запустимо проект, то на екран будуть виведені обидва фрагмента, які зможуть взаємодіяти між собою (рис. 3).

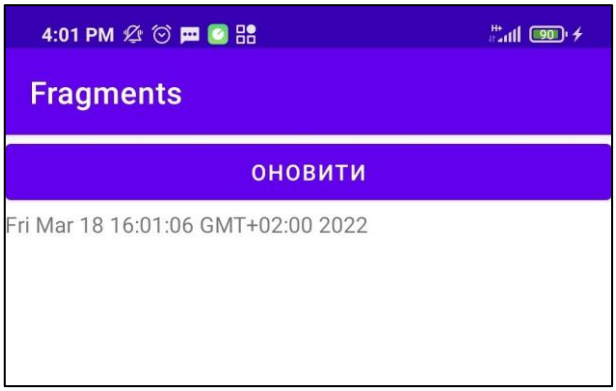


Рисунок 3 – Взаємодія фрагментів

Handlers та Loopers

Коли ми запускаємо додаток на Android, система створює потік, який називається основним потоком додатку або UI-потік. Цей потік обробляє всі зміни і події призначеного для користувача інтерфейсу. Однак для допоміжних операцій, таких як відправка або завантаження файлу, тривалі обчислення і т.д., ми можемо створювати додаткові потоки.

Для створення нових потоків нам доступний стандартний функціонал класу Thread з базової бібліотеки Java з пакета java.util.concurrent. Але використання цього класу має обмеження - ми не можемо з вторинного потоку змінювати призначений для користувача інтерфейс.

До потоку (thread) може бути прив'язано чергу повідомлень. Ми можемо поміщати туди повідомлення, а система буде за чергою стежити і відправляти повідомлення на обробку. При цьому ми можемо вказати, щоб повідомлення пішло на обробку не відразу, а через певний час.

Handler – це механізм, який дозволяє працювати з чергою повідомлень. Він прив'язаний до конкретного потоку (thread) і працює з його чергою. Handler вміє поміщати повідомлення в чергу. При цьому він ставить самого себе в якості одержувача цього повідомлення. І коли приходить час, система

дістає повідомлення з черги і відправляє його адресату (тобто в Handler) на обробку.

Handler дає нам дві цікаві і корисні можливості:

1) реалізувати відкладене за часом виконання коду

2) виконання коду не в своєму потоці

Створимо додаток, який буде підключатися до сервера, запитувати кількість файлів готових для завантаження, емулювати завантаження і відображати на екрані хід дій, використовуючи горизонтальний ProgressBar і TextView.

Критерії оцінювання дисципліни

- Оцінка «5» (*за шкалою ECTS – A,* тобто 90-100 балів) виставляється студенту, який володіє фактичним матеріалом теми, розділу, курсу. Відповідь повна, побудована послідовно, логічно і підтверджується прикладами або програмами.

- Оцінка «4» (*за шкалою ECTS – B*, тобто 85-89 балів, а також *за шкалою ECTS – C*, тобто 75-84 бали) виставляється студенту, який добре знає фактичний матеріал теми, розділу, курсу, але відповідь не достатньо повна, потребує додаткових питань викладача.

- Оцінка «3» (*за шкалою ECTS –D*, тобто 67-74 бали, а також *за шкалою ECTS – E*, тобто 60-66 бали) виставляється студенту, який володіє лише основними поняттями теми, розділу, курсу, знає основні алгоритми. Відповідає лише на деякі додаткові питання викладача.

- Оцінка «2» (*за шкалою ECTS – FX*, тобто 34-59 – це можливість повторного захисту роботи після доопрацювання, а також *за шкалою ECTS – F*, тобто 1-34 бали – з обов'язковим повторним курсом навчання) виставляється студенту, який не володіє теоретичним матеріалом теми, розділу, курсу, не знає відповідних алгоритмів, не вміє їх застосовувати.

Узагальнена система критеріїв оцінювання

Рейтинговий показник	Оцінка у національній шкалі		Оцінка ECTS
90-100	Зараховано	5 (відмінно)	A (відмінно)
82-89		4 (добре)	B(добре)
75-81			C(добре)
68-74		3 (задовільно)	D (задовільно)
60-67			E (задовільно)
35-59	Незараховано	2 (незадовільно)	FX (незадовільно) з можливістю повторного складання
1-34		-	F (незадовільно) з обов'язковим повторним вивченням

СПИСОК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Neil Smyth Android Studio 3.0 Development Essentials - Android 8 Edition 1st Edition. –USA: CreateSpace Independent Publishing Platform; 1 edition, 2017 – 726P.
2. Reto Meier. Professional Android 2. Application Development. – Wiley Publishing, Inc, 2010. – 580 p.
3. Медникс З. Программирование под Android / З. Медникс, Л. Дорнин, Б. Мик, М. Накамура – Издательство Питер, 2012. – 496 с.
4. Дэрси Л. Android за 24 часа. Программирование приложений под операционную систему Google / Дэрси Л., Кондер Ш. – М.: Рид Групп, 2011. – 464 с.
5. Bill Phillips, Chris Stewart, Kristin Marsicano Android Programming: The Big Nerd Ranch Guide (3rd Edition). –USA, Atlanta: Big Nerd Ranch Guides, 2017 – 695P.
6. Dawn Griffiths, David Griffiths Head First Android Development: A Brain-Friendly Guide 2nd Edition. – USA, California: O'Reilly Media; 2 edition, 2017 – 928P. 12. Ian F. Darwin Android Cookbook: Problems and Solutions for Android Developers 2nd Edition. – USA, California: O'Reilly Media; 2 edition, 2017 – 772P.
7. Середовище розробки Android Studio. [Електронний ресурс] Режим доступу: https://developer.android.com/studio
8. Розробка для Android [Електроний ресурс]. – Режим доступу: http://developer.android.com/.
9. The Java Tutorials. [Електронний ресурс] Режим доступу: https://docs.oracle.com/javase/tutorial/
10. Learning the Java Language. [Електронний ресурс] Режим доступу: https://docs.oracle.com/javase/tutorial/java/index.html
11. Java Development Kit. [Електронний ресурс] Режим доступу: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html