МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

Кафедра комп'ютерних наук

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт

з дисципліни

«Програмування для мобільних пристроїв»

для студентів денної форми навчання
спеціальності
126 «Інформаційні системи та технології»

Тернопіль
2020

УДК 681.3(07)
М54

Укладачі:
В. А. Готович, канд. техн. наук, асистент;
Т. В. Михайлович, асистент.

Рецензент:
Михалик Д. М., канд. тех. наук, доцент.

Методичні вказівки розглянуто і затверджено на
засіданні кафедри комп'ютерних наук
Тернопільський національний технічний університет іиені Івана Пулюя.
Протокол № 11 від 10 червня 2020 року.

Методичні вказівки схвалено Науково-методичною комісією
факультету комп'ютерно-інформаційних систем і програмної інженерії.
Протокол № 1 від 16 вересня 2020 року.

М54 Методичні вказівки до виконання лабораторних робіт з дисципліни «Програмування для мобільних пристроїв» для студентів денної форми навчання спеціальності 126 «Інформаційні системи та технології» / Укладачі: Готович В. А., Михайлович Т. В. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2020. – 100 с.

УДК 681.3(07)

ЗМІСТ

ВСТУП

Методичні вказівки розроблено відповідно до робочої програми дисципліни «Програмування для мобільних пристроїв» для студентів денної форми навчання ступеня Бакалавр за спеціальністю 126 «Інформаційні системи та технології».

Матеріал лабораторних робіт присвячено вивченню практичних основ розробки додатків, призначених для функціонування на мобільних пристроях під управлінням ОС Android. Виконання лабораторних робіт повинне закріпити теоретичні знання, отримані студентами при вивченні лекційних матеріалів дисципліни.

Для успішного засвоєння матеріалу та виконання завдань запропонованих лабораторних робіт студенти повинні володіти базовими знаннями з теорії алгоритмів, програмування та методології об'єктно-орієнтованого програмування. Необхідним є володіння на базовому рівні хоча б однією із сучасних об'єктно-орієнтованих мов програмування (C++, C#, Java).

Освоєння дисципліни «Програмування для мобільних пристроїв» дозволяє розширити професійні компетенції майбутніх фахівців в галузі інформаційних технологій щодо проектування та розробки програмного забезпечення в цілому а також є однією із передумов для підготовки та захисту студентами кваліфікаційної роботи.

ЛАБОРАТОРНА РОБОТА № 1.
СТВОРЕННЯ НАЙПРОСТІШОЇ ПРОГРАМИ ТА ЗАПУСК ЇЇ НА ЕМУЛЯТОРІ

Мета: метою виконання лабораторної роботи є набуття студентами навичок:

- створення нового проекту з використанням шаблонів;
- використання засобів середовища Android Studio для розробки додатків;
- використання емулятора для відлагодження додатків;
- використання логування для відлагодження додатків.

Короткі теоретичні відомості

Середовище Android Studio являє собою потужне інтегроване середовище розробки (IDE), включаючи вдосконалений редактор коду та набір шаблонів для створення додатків. Воно включає в себе інструменти для розробки, налагодження, тестування та перевірки продуктивності функціонування додатків, які пришвидшують і спрощують розробку мобільних додатків. Наявна можливість тестування розроблюваних додатків на основі широкого діапазону заздалегідь налаштованих емуляторів або ж на реальному мобільному пристрої, публікації розроблених додатків у магазині Google Play.

Android Studio доступне для комп'ютерів під управлінням Windows або Linux, а також для Mac під управлінням macOS. Найновіший OpenJDK (Java Development Kit) постачається в комплекті з Android Studio. Android Studio доступне безкоштовно. Більшість його компонентів доступна на умовах ліцензії Apache 2.0.

Перед встановленням Android Studio необхідно переконатися, що використовувана вами система відповідає системним вимогам, які можна знайти за посиланням https://developer.android.com/studio/index.html#Requirements. Процес інсталяції є однаковим для всіх платформ та описаний за посиланням https://developer.android.com/studio/install.html. Завантажити інсталяційний пакет можна за посиланням https://developer.android.com/studio. Після завершення інсталяції майстер установки завантажить та встановить деякі

додаткові компоненти, включаючи Android SDK. Будьте терплячі, це може зайняти деякий час, залежно від швидкості вашого Інтернету та потужності комп'ютера. Деякі кроки під час інсталяції можуть здатися вам зайвими.

Хід роботи

1. Встановлення середовища Android Studio

Завантажте та встановіть середовище Android Studio. Під час інсталяції дотримуйтесь значень параметрів, встановлених по замовчуванню. Також переконайтеся, що всі компоненти середовища є вибраними для інсталяції.

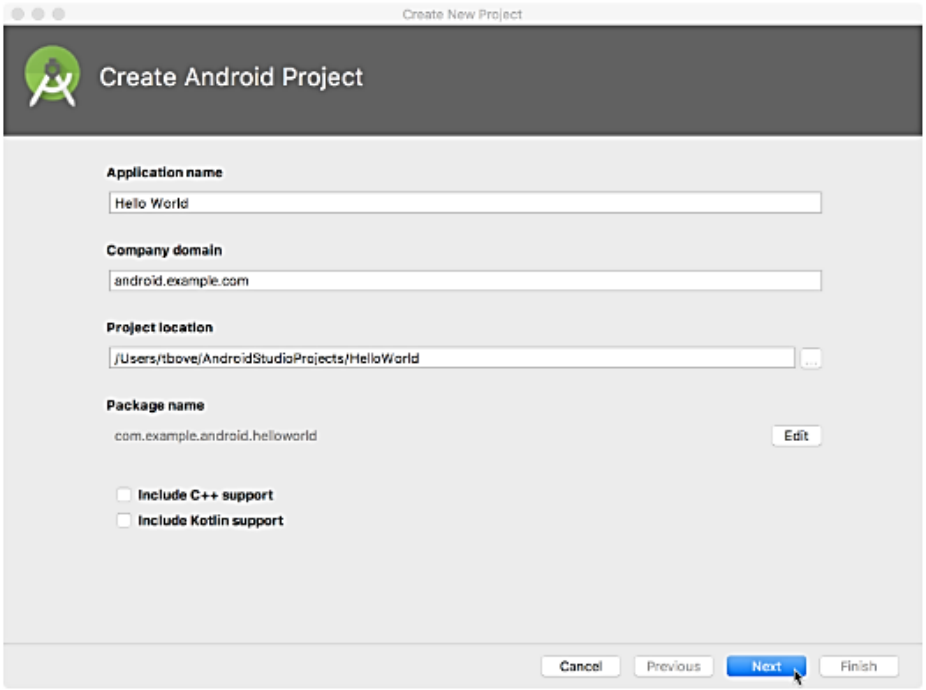
2. Створення нового проекту «Hello World»

2.1 Створення проекту

2.1.1 Запустіть середовище Android Studio.

2.1.2 У вікні **Welcome to Android Studio** натисніть **Start a new Android Studio project.**

2.1.3 У вікні **Create Android Project** введіть назву додатку **Hello World:**

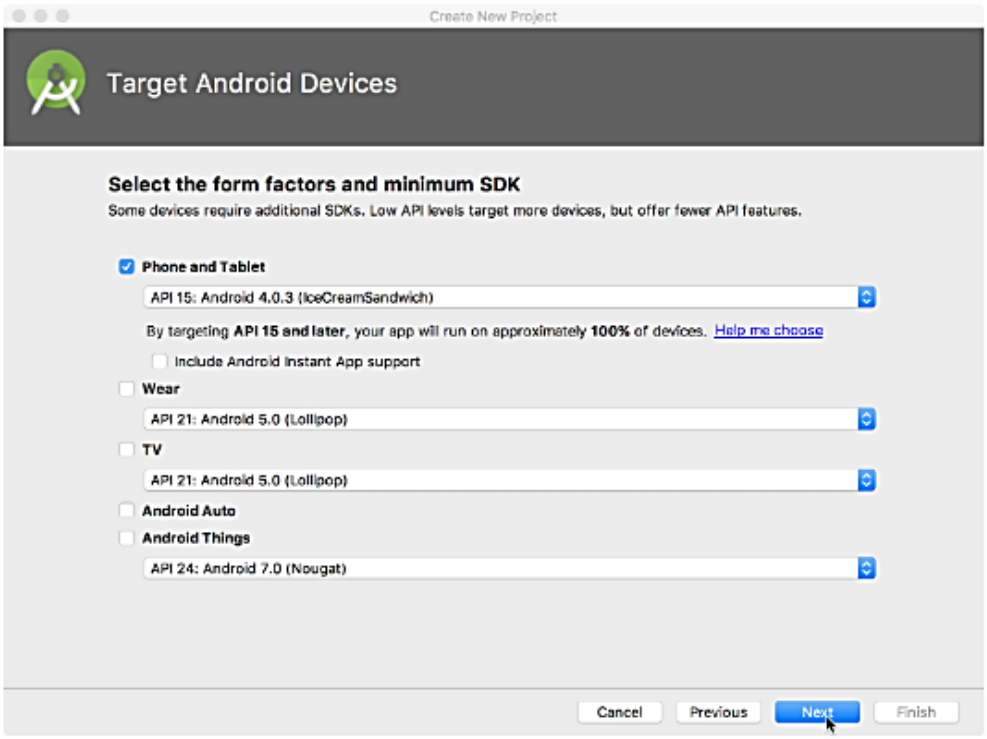


2.1.4 У полі **Project location** задайте папку, в якій зберігатимуться файли проекту.

2.1.5 Задайте значення **Company Domain**, або залиште його по замовчуванню. Задавати значення домену компанії обов'язково, якщо ви маєте намір опублікувати розроблений додаток.

2.1.6 Залиште невідміченими опції **Include C++ support** та **Include Kotlin support**, після чого натисніть кнопку **Next**.

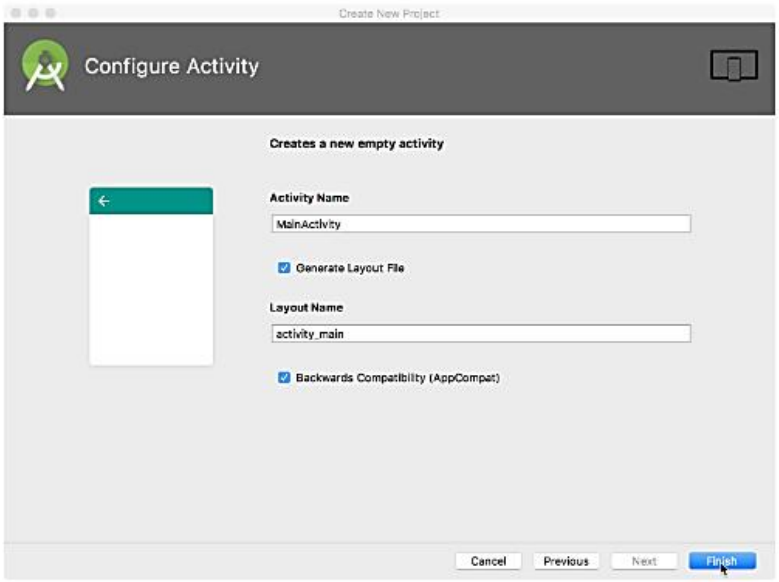
2.1.7 У вікні **Target Android Devices** опція **Phone and Tablet** повинна бути відміченою. Рекомендується встановити значення **Minimum SDK** у **API 15: Android 4.0.3 IceCreamSandwich**:



2.1.8 Залиште невідміченою опцію **Include Instant App support.** Якщо ваш проект потребує додаткових компонент, відповідно до обраного **Minimum SDK**, середовище Android Studio завантажить та інсталює їх автоматично.

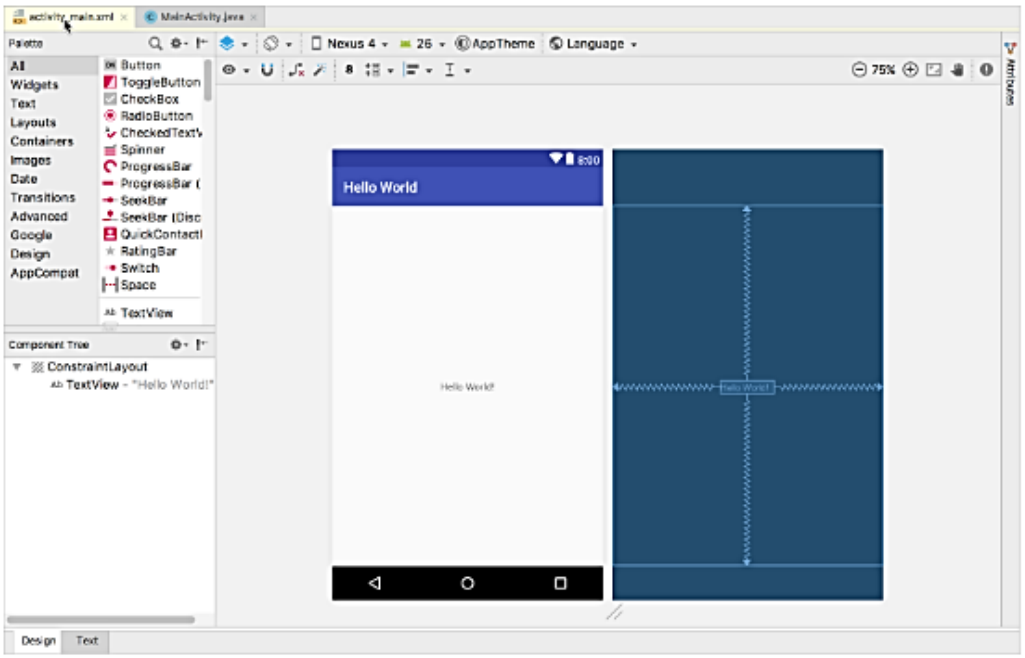
2.1.9 У вікні **Add an Activity** виберіть шаблон **Empty Activity.**

2.1.10 У вікні **Configure Activity** залиште значення назви головного activity проекту встановленим по замовчуванню *MainActivity***:**



2.1.11 Відмітьте опцію **Generate Layout file**. Залиште значення **Layout Name** по замовчуванню *activity_main*.

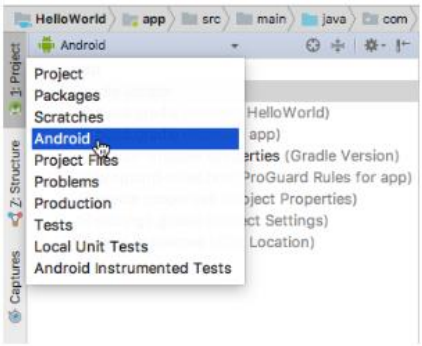
2.1.12 Переконайтеся, що опція **Backwards Compatibility (App Compat)** є відміченою. Вона гарантує, що ваш додаток буде сумісним із попередніми версіями Android. Натисніть кнопку **Finish**. Середовище Android Studio створить проект. На це йому знадобиться трохи часу. У вікні середовища перейти в редактор розмітки можна клацнувши на закладку *activity_main.xml*. Закладка *Design* дозволить вам працювати із графічним представленням розмітки:



Щоб перейти в редактор коду проекту необхідно клацнути на закладку *MainActivity.java*.

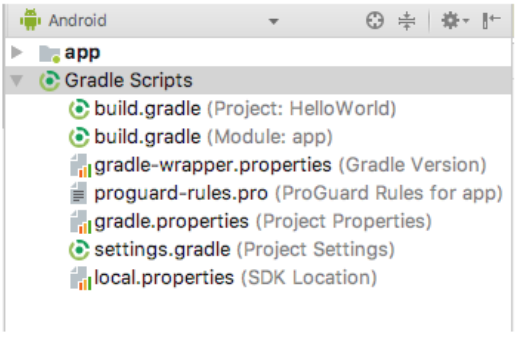
2.2 Ознайомлення із структурою проекту

Натисніть вкладку *Project* у вертикальному списку вкладок у верхньому лівому куті вікна Android Studio. З'явиться панель *Project*. Щоб побачить файли проекту необхідно вибрати пункт *Android* з випадного меню:



2.3 Ознайомлення із вмістом папки Gradle Scripts

Система Gradle build в Android Studio дозволяє легко включати зовнішні бінарні файли або інші модулі бібліотеки до вашого проекту у вигляді залежностей. Коли ви створюєте новий проект програми, на панелі *Project>Android* з'являється папка Gradle Scripts наступного вмісту:

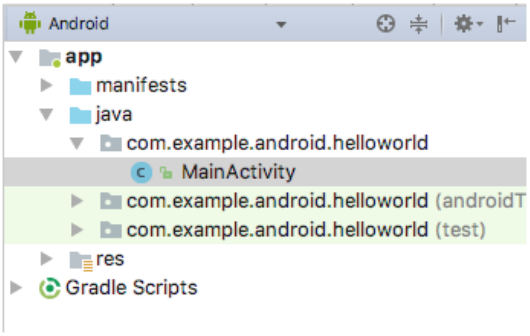


У файлі *build.gradle(Project: HelloWorld)* ви знайдете параметри конфігурації, спільні для всіх модулів проекту. Кожен проект Android Studio містить єдиний файл *Gradle build* верхнього рівня. Здебільшого вам не потрібно вносити жодних змін у цей файл. По замовчуванню він

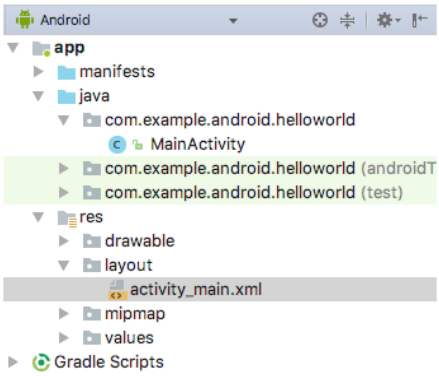
використовує блок *buildscript* для визначення репозиторіїв та залежностей, спільних для всіх модулів проекту.

2.4 Ознайомлення із вмістом папок app та res

В даних папках знаходяться всі файли коду та ресурсів проекту:



У папці *java\com.example.hello.helloworld* знаходяться файли класів проекту. Дві інші підпапки папки *java* містять файли, які використовуються для модульного тестування проекту. Файли ресурсів проекту знаходяться в папці *res*:

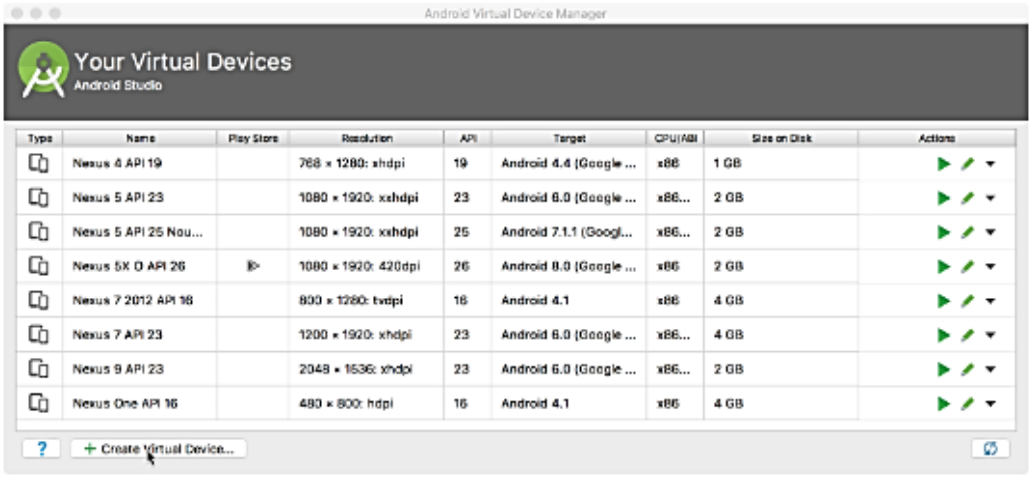


3. Запуск проекту на виконання з допомогою емулятора

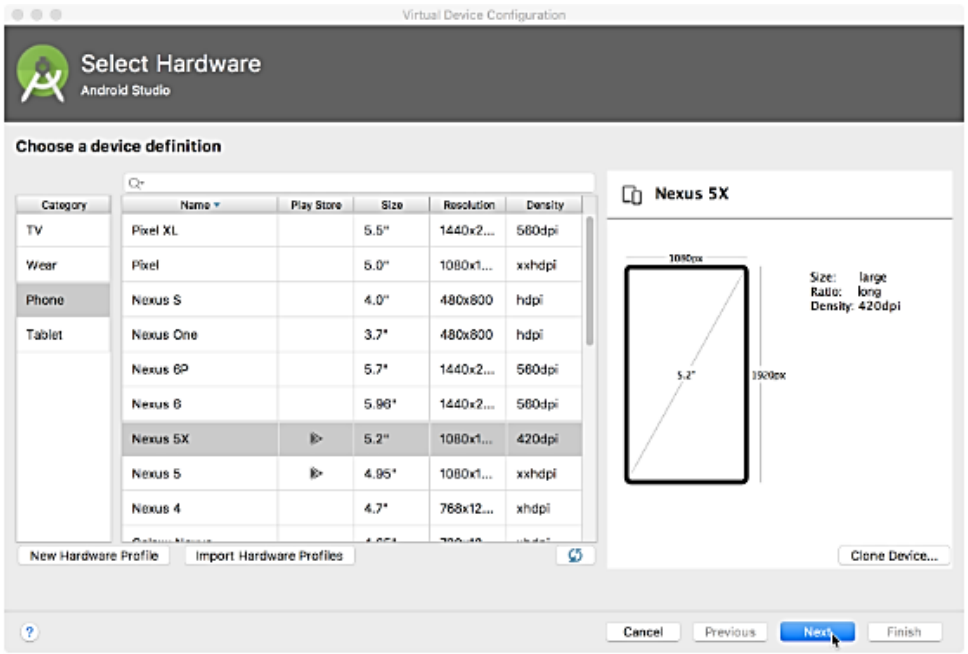
Використовуючи AVD Manager ви визначаєте апаратні характеристики віртуального пристрою, його рівень API, сховище та інші властивості. За допомогою таких віртуальних пристроїв ви можете тестувати програми на різних конфігураціях пристроїв (наприклад, планшети та телефони) з різними рівнями API.

3.1 Створення AVD

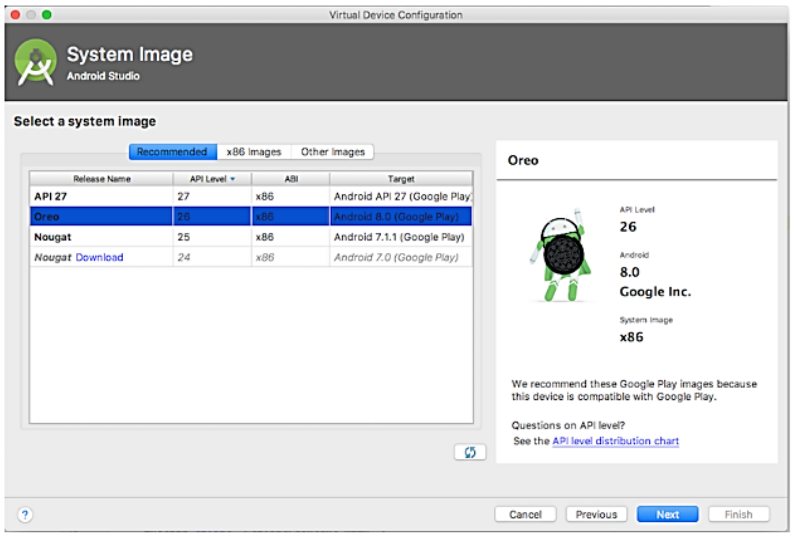
Застосуйте команду *Tools>Android>AVD Manager* або натисніть на іконку  на панелі. З'явиться вікно *Your Virtual Devices*:



Натисніть кнопку *+Create Virtual Device*. Зявиться вікно *Select Hardware* із переліком попередньо підготовлених конфігурацій пристроїв:



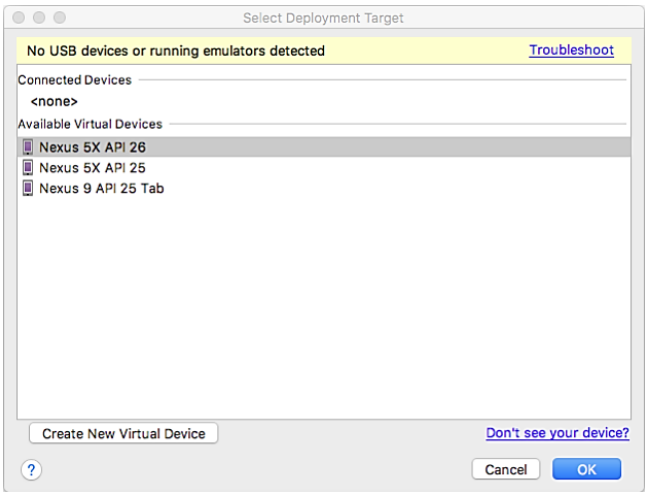
Після вибору конфігурації з'явиться вікно *System Image*. На вкладці *Recommended* необхідно вибрати версію Android:



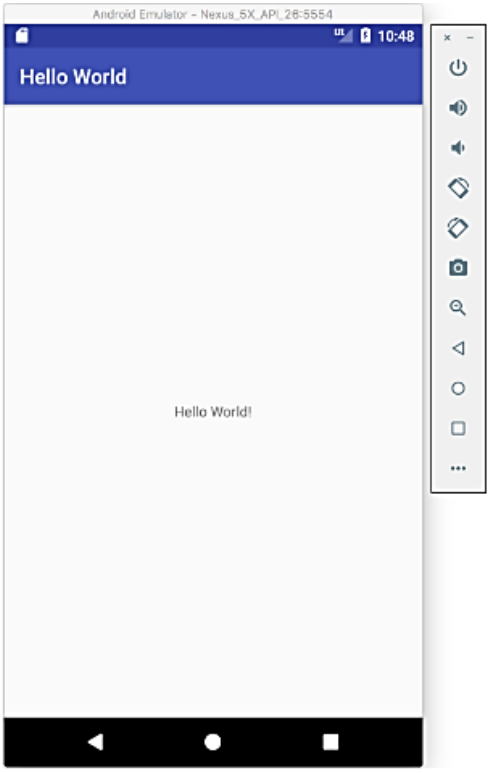
Якщо кнопка *Download* навпроти відповідної версії ОС є видимою, то її необхідно натиснути, щоб середовище завантажило образ системи. Після вибору образу системи натисніть *Next*.

3.2 Запуск додатку на віртуальному пристрої

В середовищі Android Studio застосуйте команду *Run>Run app* або натисніть на кнопку  на панелі. Тоді у вікні *Select Deployment Target* виберіть віртуальний пристрій:



Емулятор запуститься і завантажиться як фізичний пристрій. Як тільки ваш додаток успішно скомпілюється, Android Studio завантажить його в емулятор і запустить на виконання. Ви повинні побачити вікно додатку *Hello World*, як показано на наступному малюнку:

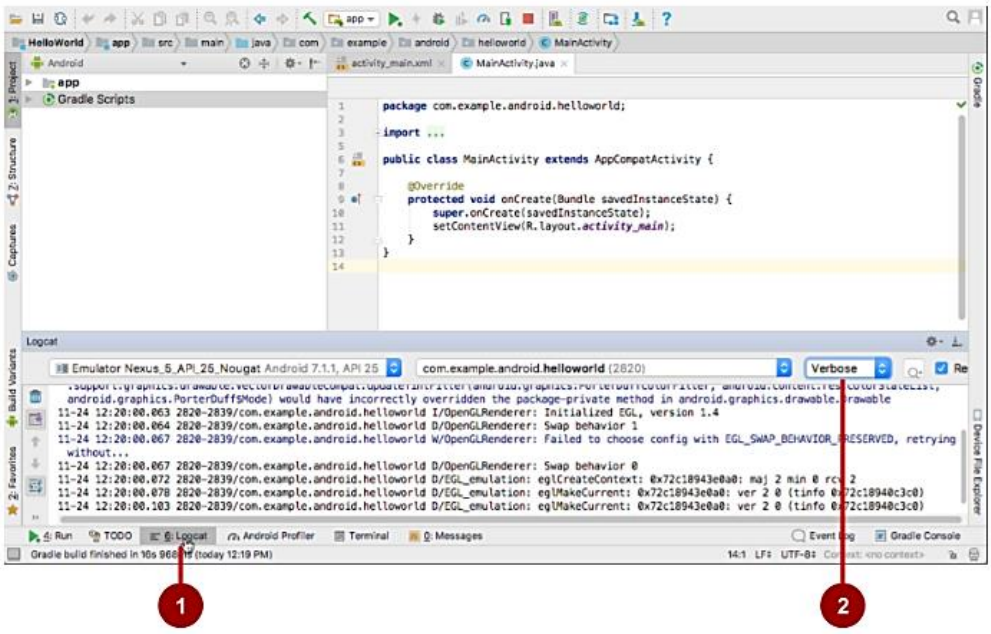


Під час тестування на віртуальному пристрої хорошим підходом є запуск його один раз, на самому початку сеансу. Не слід закривати його, поки ви не закінчите тестування свого додатку, щоб додатку не довелося повторно проходити процес запуску пристрою.

4. Застосування логування

4.1 Вікно з повідомленнями журналу

Повідомлення журналу (логування) є потужним інструментом, який допомагає контролювати стан змінних, шляхів виконання та помилки під час відлагодження додатку. Вони з’являються в панелі *Logcat pane*, яку можна відкрити натиснувши на закладку *Logcat* внизу середовища Android Studio:



На рисунку: 1 – закладка *Logcat*; 2 – меню вибору рівня повідомлень журналу, які будуть відображені в *Logcat pane; Verbose* – значення по замовчуванню (показувати всі повідомлення). Інші доступні значення: *Debug*, *Error*, *Info* та *Warn***.**

4.2 Додавання логування в проект

Повідомлення логування додаються в журнал командою

```
Log.d("MainActivity", "Hello World");
```

Тут:

- `Log` – спеціальний клас;
- `d()` – метод класу, який додає в журнал повідомлення рівня *Debug*. Інші методи: `e()`, `w()` та `i()`;
- `"MainActivity"` – перший параметр, т.зв. «тег», який можна використовувати для фільтрації повідомлень журналу. Рекомендується для цього використовувати найменування activity, з якого поступають повідомлення журналу. Найменування слід записувати в приватну змінну класу activity, наприклад, так:

```
private static final String LOG_TAG =
MainActivity.class.getSimpleName();
```

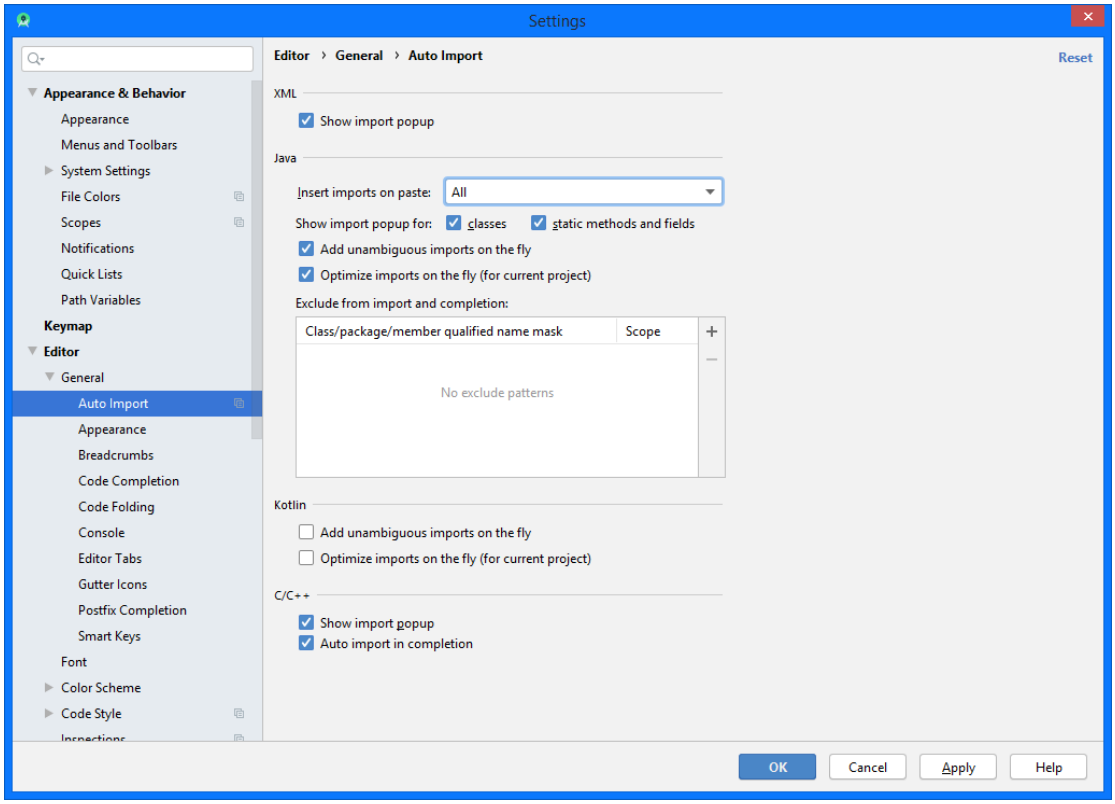
- `"Hello world"` – другий параметр, власне, повідомлення.

Тепер виконайте наступні кроки:

4.2.1 Перейдіть в клас *MainActivity* проекту.

4.2.2 Виберіть пункт меню *File>Settings* (Windows) або *Android Studio>Preferences* (macOS).

4.2.3 Виберіть пункт *Editor>General>Auto Import*. Відмітьте усі прапорці та встановіть *Insert imports on paste* у значення *All*:



Ці дії необхідні для використання класу *Log*.

4.2.4 Натисніть *Apply*, тоді *OK*.

4.2.5 Метод *onCreate*() класу *MainActivity* повинен мати вигляд:

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

4.2.6 Відкрийте *Logcat pane*.

4.2.7 Запустіть проект на виконання. У *Logcat pane* ви повинні побачити повідомлення, схоже на таке: `11-24 14:06:59.001 4696-4696/? D/MainActivity: Hello World`

5. Додаткове завдання

Запустіть додаток *Hello World* на фізичному пристрої.

Контрольні запитання

1. Опишіть основні елементи середовища Android Studio.
2. Опишіть структуру файлів найпростішого проекту.
3. Що таке Емулятор? Переваги та недоліки застосування емуляторів при відлагодженні додатків.
4. Опишіть процес створення віртуального пристрою. Як запустити додаток на віртуальному пристрої?
5. Як запустити додаток на фізичному пристрої?
6. Що таке повідомлення журналу (логування) та як вони можуть допомогти в процесі відлагодження додатку?

Перелік посилань

1. Конспект лекцій з курсу «Програмування для мобільних пристроїв» для здобувачів освітнього ступеня «бакалавр», спеціальності 126 «Інформаційні системи та технології», денної форми навчання. Укладачі Готович В. А., Михайлович Т. В. ТНТУ, 2020 р.
2. https://codelabs.developers.google.com/codelabs/android-training-hello-world/index.html?index=..%2F..%2Fandroid-training#0
3. https://developer.android.com/studio/projects
4. https://developer.android.com/studio/run
5. https://developer.android.com/studio/run/managing-avds
6. https://gradle.org/
7. https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-1-get-started/lesson-1-build-your-first-app/1-0-c-introduction-to-android/1-0-c-introduction-to-android.html

ЛАБОРАТОРНА РОБОТА № 2.
РОБОТА З ОБ'ЄКТАМИ ACTIVITY ТА INTENT

Мета: метою виконання лабораторної роботи є набуття студентами навичок:

- створення нової Activity в середовищі Android Studio;
- оголошення батьківської та дочірньої Activity для навігації;
- запуску на виконання Activity з допомогою явного Intent;
- передачі даних між Activity.

Короткі теоретичні відомості

Activity представляє собою окремий екран, за допомогою користувач додатку може виконати єдине цілеспрямоване завдання, наприклад, зробити фотографію, надіслати електронний лист або переглянути карту. Activity зазвичай представляється користувачеві як повноекранне вікно.

Додаток зазвичай складається з декількох екранів, які слабко пов'язані один з одним. Кожен екран додатку – це окрема Activity. Зазвичай одна Activity у додатку визначається як основна (*MainActivity.java*), яка подається користувачеві під час запуску програми. Потім основна Activity може запускати інші для виконання різних дій.

Кожного разу, коли запускається нова Activity, попередня Activity зупиняється, але система зберігає Activity у стеку (*back stack*). Коли нова Activity запускається, вона поміщується на вершину стеку і отримує фокус користувача. Коли користувач закінчує роботу поточної Activity натискаючи на кнопку Back, ця Activity вилучається зі стеку та знищується, а робота попередньої Activity відновлюється.

Activity запускається або активується з допомогою Intent. Об'єкт Intent – це асинхронне повідомлення, яке ви можете використовувати у своїй Activity, щоб надіслати запит на дію з іншої Activity чи з іншого компонента програми. Ви використовуєте Intent для запуску однієї Activity з іншої Activity та для передачі даних між ними. Intent може бути явним або неявним:

- явний (*explicit*) Intent – це такий Intent, мета якого є відомою. Тобто ви знаєте повністю назву класу конкретної Activity. Ви визначаєте конкретний цільовий компонент для передачі даних;

- неявний (*implicit*) Intent – це такий Intent, для якого ім’я цільового компонента є невідомим, але у вас є певна загальна для виконання дія. Для такого Intent визначається функціональність, але не цільовий компонент.

В даній лабораторній роботі ви працюватимете із явними об’єктами типу Intent.

Кожна нова Activity, яку ви додаєте до свого проекту, має власний макет (*layout*) та Java-файл, відокремлені від інших в проекті. У файлі *AndroidManifest.xml* вона має свій відповідний елемент <*activity*>. Як і головна Activity проекту, всі інші Activity наслідуються від класу AppCompatActivity.

Хоча кожна Activity в проекті слабо пов’язана з іншими, ви можете визначити Activity як частину іншої Activity в файлі *AndroidManifest.xml*. Це відношення батько-нащадок змушує Android додавати навігаційні підказки, такі як стрілки «ліворуч» у рядку заголовка для кожного Activity.

Об’єкт Intent може передавати дані цільовому Activity двома способами: використовуючи поле даних (*intent data*) або додатки (*intent extras*). Поле даних являє собою URI, який вказує на конкретні дані для дії. *Intent extras* є об’єктом типу *Bundle*, що представляє собою колекцію ключ/значення.

Хід роботи

1. Створення проекту, який складається із двох Activity

1.1 Створення нового проекту

Запустіть Android Studio та створіть новий проект під назвою *Two Activities*. Серед шаблонів Activity виберіть *Empty Activity*. Залишіть ім’я головного Activity по замовчуванню (*MainActivity*) без змін. Переконайтесь, що опції *Generate Layout file* та *Backwards Compatibility (AppCompat)* залишились увімкненими.

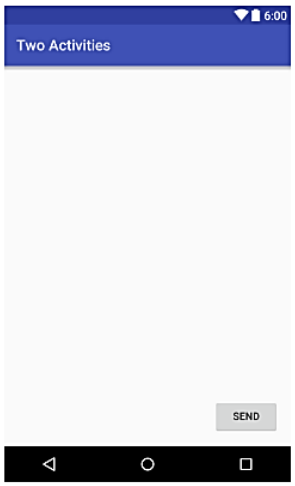
1.2 Визначення Layout для головного Activity

1.2.1 Відкрийте в редакторі розмітки *res>layout>activity_main.xml*.

1.2.2 Перейдіть в закладку *Design* і видаліть *TextView*, який містить напис «Hello World».

1.2.3 При увімкненій по замовчуванню опції *Autoconnect* перетягніть кнопку *Button* із панелі компонентів у правий нижній кут розмітки інтерфейсу користувача.

1.2.4 В панелі атрибутів *Attributes* встановіть значення її властивостей: *ID* на *button_main*, *layout_width* та *layout_height* на *wrap_content*, напис *Send*. В результаті ви повинні отримати:



1.2.5 Перейдіть на закладу *Text* для редагування xml-коду. Додайте для кнопки атрибут: `android:onClick="launchSecondActivity"`

1.2.6 Застосуйте ресурси, щоб присвоїти кнопці напис *Send*. Відповідному ресурсу присвойте імя *button_main*. Розмітка кнопки повинна мати вигляд:

```
<Button
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/button_main"
        android:onClick="launchSecondActivity"

app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent"
/>
```

1.3 Прив'язування обробника подій натискання мишкою до кнопки

Визначимо метод *launchSecondActivity*() та прив'яжемо його до атрибуту *android:onClick* нашої кнопки.

1.3.1 Натисніть на *launchSecondActivity()* в коді файлу *activity_main.xml*.

1.3.2 Натисність Alt+Enter та виберіть команду *Create 'launchSecondActivity(View)' in 'MainActivity* в меню.

1.3.3 Всередині коду методу *launchSecondActivity*() напишіть код:

```
Log.d(LOG_TAG, "Button clicked!");
```

1.3.4 У верхній частині класу MainActivity додайте константу для змінної LOG-TAG:

```
private static final String LOG_TAG =

MainActivity.class.getSimpleName();
```

1.3.5 Запустіть проект на виконання. При натисканні на кнопку *Send* ви повинні побачити повідомлення "*Button Clicked*!" у вікні логів *Logcat*. Код класу MainActivity тепер повинен виглядати так:

```
package com.example.android.twoactivities;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;

public class MainActivity extends AppCompatActivity {
   private static final String LOG_TAG =

MainActivity.class.getSimpleName();
   @Override
```

```
    protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void launchSecondActivity(View view) {
        Log.d(LOG_TAG, "Button clicked!");
    }
}
```

2. Створення та запуск другого Activity

2.1 Створення другого Activity

2.1.1 Натисність на папці app у проеткі та виберіть команду *File > New>Activity>Empty Activity*.

2.1.2 Назвіть другу Activity ім'ям SecondActivity. Переконайтесь, що опції *Generate Layout File* та *Backwards Compatibility (AppCompat)* залишились відміченими. Ім'я файлу розмітки повинне бути *activity_second*. Не відмічайте опцію *Launcher Activity*.

2.1.3 Натисніть *Finish*. Android Studio згенерує та додасть в проект файли, які відповідають другому Activity, а також внесе необхідні зміни у файл маніфесту проекту.

2.2 Модифікація файла маніфесту

2.2.1 Відкрийте файл *manifests > AndroidManifest.xml*.

2.2.2 Знайдіть елемент <*activity*>, який відповідає другому Activity

```
<activity android:name=".SecondActivity"></activity>
```

2.2.3 Перепишіть цей елемент слідуючим вмістом:

```
<activity android:name=".SecondActivity"
    android:label = "Second Activity"
    android:parentActivityName=".MainActivity">
    <meta-data

android:name="android.support.PARENT_ACTIVITY"
        android:value=
```

```
"com.example.android.twoactivities.MainActivity" />
</activity>
```

Атрибут *label* тут задає ім'я Activity, яке буде виведено на панель. За допомогою атрибуту *parentActivityName* ми вказуємо, що батьківською по відношенню до Second Activity буде MainActivity.

2.2.4 Зробіть так, щоб ім'ям другої Activity було *activity2_name*, яке вибирається з ресурсів.

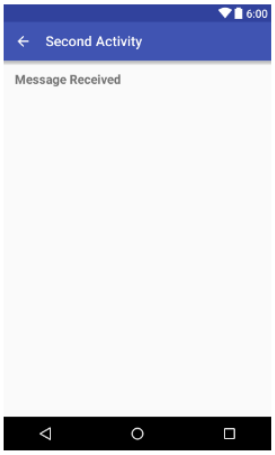
2.3 Визначення розмітки для другої Activity

2.3.1 Відкрийте файл *activity_second.xml* та перейдіть в режим дизайнера (закладка *Design*)

2.3.2 Перетягніть *TextView* з палітри компонентів у верхній лівий кут розмітки та задайте прив'язку даного елемента до верхнього та лівого краю розмітки. Встановіть атрибути *TextView* в такі значення:

Атрибут	**Значення**
id	*text_header*
Top margin	*16*
Left margin	*8*
layout_width	*wrap_content*
layout_height	*wrap_content*
text	*Message Received*
textAppearance	*AppCompat.Medium*
textStyle	*B (bold)*

Значенням атрибуту *textAppearance* є найменування стандартної теми Android, яка визначає основні стилі шрифту. Розмітка Activity повинна зараз виглядати так:



2.3.3 Перейдіть в режим xml-розмітки (закладка *Text*) та зробіть так, щоб стрічка *Message Received* вибиралася із ресурсу під назвою *text_header*.

2.3.4 Додайте до *TextView* атрибут *android:layout_marginLeft="8dp"* щоб доповнити атрибут *layout_marginStart* для старіших версій Android. Тепер файл *activity_second.xml* повинен мати такий вигляд:

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    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="com.example.android.twoactivities.Seco
ndActivity">
    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header"
        android:textAppearance=
```

```
"@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

2.4 Додавання Intent до головної Activity проекту

2.4.1 Перейдіть у файл Java-коду головної Activity

2.4.2 Створіть Intent в методі *launchSecondActivity()*

Конструктор explicit Intent приймає два аргументи: контекст програми та конкретний компонент, для якого цей Intent призначений: `Intent intent=new Intent(this, SecondActivity.class);`

2.4.3 Викликаємо метод *startActivity*(), якому в якості параметра передаємо створений intent: `startActivity(intent);`

2.4.4 Запустіть проект на виконання. Тепер при натисканні на кнопку *Send*, MainActivity посилає Intent та операційна система запускає SecondActivity, яка з'являється на екрані. Щоб повернутися до головної Activity слід натиснути кнопку *Up* (кнопка ліворуч на панелі додатку вгорі) або кнопку *Back* внизу екрану.

3. Передача даних від MainActivity до SecondActivity

3.1 Додавання EditText до розмітки головної Activity

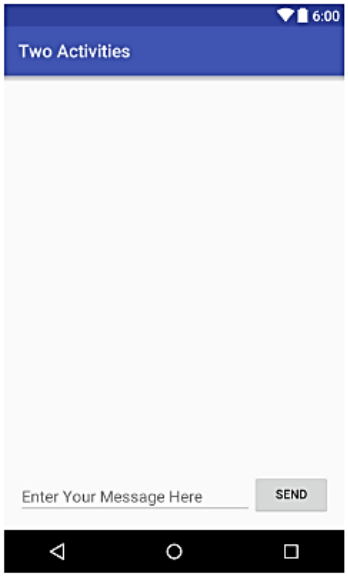
3.1.1 Перейдіть у файл *activity_main.xml*.

3.1.2 Перетягніть елемент *Plain Text (EditText)* з палітри у низ розмітки та задайте прив'язку до лівої сторони та до низу розмітки, а також до лівої сторони кнопки *Send*. Встановіть значення атрибутів:

Атрибут	Значення
id	*editText_main*
Right margin	*8*
Left margin	*8*
Bottom margin	*16*
layout_width	*match_constraint*
layout_height	*wrap_content*

inputType	*textLongMessage*
hint	*Enter Your Message Here*
text	*(Delete any text in this field)*

Розмітка головної Activity повинна зараз виглядати так:



Перейдіть в режим xml-розмітки та зробіть так, щоб стрічка «*Enter Your Message Here*» вибиралася з ресурсу під назвою *editText_main*. Тепер файл *activity_second.xml* повинен мати такий вигляд:

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    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="com.example.android.twoactivities.Main
Activity">
    <Button
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/button_main"
        android:onClick="launchSecondActivity"

app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"
/>
    <EditText
        android:id="@+id/editText_main"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:ems="10"
        android:hint="@string/editText_main"
        android:inputType="textLongMessage"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toStartOf="@+id/button_main"
        app:layout_constraintStart_toStartOf="parent"
/>
</android.support.constraint.ConstraintLayout>
```

3.2 Внесення стрічки в Intent extras

3.2.1 Відкрийте MainActivity.

3.2.2 Додайте public константу:

```
public static final String EXTRA_MESSAGE =

"com.example.android.twoactivities.extra.MESSAGE";
```

3.2.3 Додайте приватну змінну для роботи із *EditText:* `private EditText mMessageEditText;`

3.2.4 В методі *onCreate*() використайте метод *findViewById*() щоб отримати посилання на *EditText*:

```
mMessageEditText=findViewById(R.id.editText_main);
```

3.2.5 В методі *launchSecondActivity*() після оголошення Intent отримуємо текст *EditText* як стрічку:

```
String message=mMessageEditText.getText().toString();
```

3.2.6 Додаємо отриману стрічку в Inetnt, використовуючи константу *EXTRA_MESSAGE* в якості ключа:

```
intent.putExtra(EXTRA_MESSAGE, message);
```

Тепер методи повинні мати вигляд:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mMessageEditText =
findViewById(R.id.editText_main);
}
…
public void launchSecondActivity(View view) {
        Log.d(LOG_TAG, "Button clicked!");
        Intent intent = new Intent(this,
SecondActivity.class);
        String message =
mMessageEditText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
}
```

3.3 Додавання TextView до SecondActivity

3.3.1 Перейдіть у файл activity_second.xml.

3.3.2 Додайте ще один *TextView* до розмітки після *text_header TextView* та прив’яжіть його до лівої сторони розмітки та низу *text_header*.

3.3.3 Встановіть значення атрибутів:

Атрибут	Значення
id	*text_message*
Top margin	*8*
Left margin	*8*
layout_width	*wrap_content*
layout_height	*wrap_content*
text	*(Delete any text in this field)*
textAppearance	*AppCompat.Medium*

Тепер файл *activity_second.xml* повинен виглядати так:

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    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="com.example.android.twoactivities.Seco
ndActivity">
    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header"
        android:textAppearance=

"@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/text_message"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/text_header
" />
</android.support.constraint.ConstraintLayout>
```

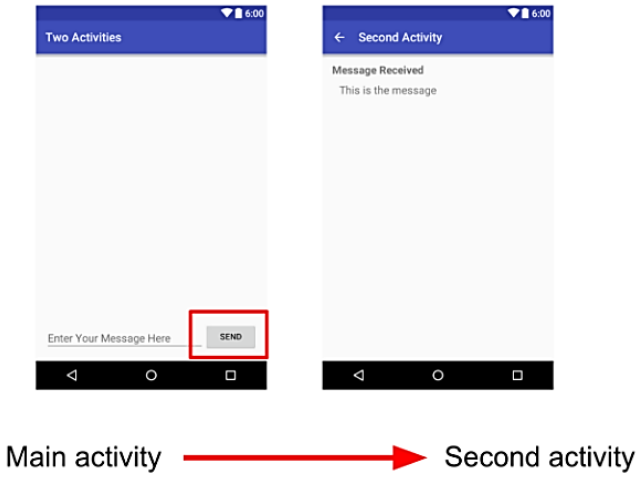
3.4 Модифікація SecondActivity для того, щоб він міг отримати дані та відобразити їх на екрані

3.4.1 Відкрийте SecondActivity

3.4.2 В код методу *onCreate*() додайте наступний код:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);
    // отримати Intent, який активує дане Activity
    Intent intent = getIntent();
    // отримати стрічку з intent по ключу
EXTRA_MESSAGE
    String message =
intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
    // використати findViewById() щоб отримати
посилання на textView з розмітки
    TextView textView =
findViewById(R.id.text_message);
    // встановити текст для textView
    textView.setText(message);
}
```

3.4.3 Запустіть додаток на виконання. Тепер коли ми вводимо в MainActivity повідомлення та натискаємо *Send*, запускається друга Activity, яка це повідомлення приймає і відображає на екрані:



Контрольні запитання

1. Де знаходяться в проекті файли класів activity та xml-файли розмітки інтерфейсу?
2. Що таке Activity?
3. Опишіть життєвий цикл Activity.
4. Як можна запустити Activity?
5. Яким чином можна передавати/отримувати дані між різними Activity?
6. Що таке Intent та для чого він використовується?
7. Які бувають види Intent?

Перелік посилань

1. Конспект лекцій з курсу «Програмування для мобільних пристроїв» для здобувачів освітнього ступеня «бакалавр», спеціальності 126 «Інформаційні системи та технології», денної форми навчання. Укладачі В.А. Готович, Т.В. Михайлович : ТНТУ, 2020 р.
2. https://codelabs.developers.google.com/codelabs/android-training-create-an-activity/index.html?index=..%2F..%2Fandroid-training#0
3. https://developer.android.com/guide/components/fundamentals.html
4. https://developer.android.com/guide/components/activities.html
5. https://developer.android.com/guide/components/intents-filters.html
6. https://developer.android.com/guide/navigation/navigation-design-graph
7. https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-1-get-started/lesson-2-activities-and-intents/2-1-c-activities-and-intents/2-1-c-activities-and-intents.html

ЛАБОРАТОРНА РОБОТА № 3.
РОЗРОБКА ДОДАТКУ З ІНТЕРАКТИВНИМ ІНТЕРФЕЙСОМ

Мета: метою виконання лабораторної роботи є набуття студентами навичок:

- створення View, ViewGroup і макетів;
- використання редактора макету;
- встановлення ширини та висоти макету;
- використання стрічкових ресурсів;
- обробка подій натискань по елементах управління.

Короткі теоретичні відомості

Компоненти екрану. Графічний інтерфейс користувача є ієрархією об'єктів *android.view.View* і *android.view.ViewGroup*. Кожен об'єкт *ViewGroup* представляє собою 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 через які, та само як і через віджети (кнопки, текстові поля і т. д.), користувач взаємодіє з програмою. Більшість візуальних елементів успадковуються від класу *View*, такі як кнопки, текстові поля та інші, і розташовуються в пакеті *android.widget*.

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

- створити елементи управління програмно в коді Java;
- оголосити елементи інтерфейсу в xml.

Можливим є поєднання обох способів – базові елементи розмітки визначити в xml, а решта додавати під час виконання додатку програмним способом.

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

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

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

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

Користувацький інтерфейс (UI), який з'являється на екрані пристрою Android, складається з ієрархії об'єктів, що називається *View* (кожен елемент екрану). Клас *View* представляє собою базовий будівельний блок для всіх компонентів інтерфейсу, а також базовий клас для класів, які надають інтерактивні компоненти інтерфейсу користувача, такі як кнопки, прапорці та поля для введення тексту:

- *TextView* – для відображення тексту;
- *EditText* – щоб дозволити користувачеві вводити та редагувати текст;
- кнопки та інші елементи (*RadioButton*, *CheckBox, Spinner*), які можна натискати для забезпечення інтерактивної поведінки;
- *ScrollView* та *RecyclerView* – для відображення елементів, що прокручуються;
- *ImageView* – для відображення зображень;
- *ConstraintLayout* та *LinearLayout*, що містять інші елементи *View* та визначають їх розміщення.

Хід роботи

1. Створення проекту

1.1 Створення нового проекту

1.1.1 Запустіть Android Studio та створіть новий проект під назвою *Hello Toast* з такими параметрами:

Атрибут	**Значення**
1	**2**
Application Name	*Hello Toast*
Company Name	*com.example.android (або ваш домен)*
Phone and Tablet Minimum SDK	*API15: Android 4.0.3 IceCreamSandwich*
1	**2**
Template	*Empty Activity*
Generate Layout file box	*Selected*
Backwards Compatibility box	*Selected*

1.1.2 Виберіть *Run>Run app* або клацніть піктограму *Run icon* на панелі інструментів, щоб створити та запустити програму на емуляторі або на вашому пристрої.

1.2 Дослідження редактора макетів

1.2.1 У папці *app>res>layout* на панелі *Project>Android* двічі клацніть файл *activity_main.xml*, якщо він ще не відкритий.

1.2.2 Клацніть на вкладку *Design*. Вона використовується для маніпулювання елементами та макетом, а вкладка *Text* – для редагування коду xml для макета.

1.2.3 На панелі *Palettes* відображаються елементи інтерфейсу, які можна використовувати в макеті програми.

1.2.4 На панелі дерева компонентів відображається ієрархія подання елементів інтерфейсу. Елементи *View* організовані в ієрархію дерев батьків та нащадків, в якій нащадки успадковують атрибути батьків. *TextView* – це нащадок *ConstraintLayout*.

1.2.5 Панелі дизайну та проекту редактора макета, що відображають елементи інтерфейсу в макеті.

1.2.6 На вкладці *Attributes* відображається область атрибутів для встановлення властивостей елемента інтерфейсу.

2. Додавання елементів подання в редакторі макета

2.1 Дослідження меж елементів

2.1.1 Відкрийте *activity_main.xml* в панелі *Project* > *Android*. Виберіть вкладку *Design*.

2.1.2 Якщо немає розмітки, натисніть кнопку *Select Design Surface* на панелі інструментів та оберіть *Design+Blueprint*.

2.1.3 Інструмент *Autoconnect* також знаходиться на панелі інструментів. По замовчуванню він увімкнений. На цьому етапі переконайтеся, що цей інструмент не вимкнено.

2.1.4 Клацніть на кнопку збільшення або зменшення масштабу, щоб наблизити панелі дизайну та креслення для крупного вигляду.

2.1.5 Виберіть *TextView* на панелі дерева компонентів. *TextView* «Hello World» виділений на панелях дизайну та проекту, а також видно прив’язки елемента.

2.1.6 Клацніть на кружок з правого боку *TextView*, щоб видалити горизонтальну прив’язку правої границі елемента до правої сторони макета. *TextView* перестрибує вліво, тому що він більше не має прив'язки із правим боком. Щоб повернути прив’язку, клацніть той самий кружок та перетягніть лінію праворуч від макета.

2.1.7 Клацніть на квадратик в правому нижньому куті для зміни розміру елемента в цьому напрямку.

2.2 Додавання кнопки до макету

2.2.1 Почніть з чистого аркуша. Видаліть елемент *TextView* – натисніть кнопку *Видалити* або виберіть *Правка* > *Видалити*. Тепер у вас повністю порожній макет.

2.2.2 Перетягніть *Button* із панелі *Palette* у будь-яку позицію макета. Якщо ви розмістите *Button* у верхню середню область макета, прив’язки меж можуть з’явитися автоматично. В іншому випадку, можна перетягнути прив’язку у верхню, ліву та праву сторони макета.

2.3 Додавання ще одної кнопки

2.3.1 Перетягніть іншу *Button* з панелі *Palette* до середини макета. *Autoconnect* може забезпечити для вас горизонтальну прив’язку (якщо ні, ви можете перетягнути її самостійно).

2.3.2 Перетягніть вертикальну прив’язку внизу макета.

2.3.3 Ви можете видалити прив’язку з елемента, вибравши елемент і навівши на нього вказівник, щоб показати кнопку *Clear Constraints*. Натисніть цю кнопку, щоб видалити всі прив’язки для вибраного елемента. Щоб очистити одну прив’язку, клацніть конкретний кружок, який встановлює прив’язку.

2.3.4 Щоб очистити всі прив’язки у всьому макеті, натисніть інструмент *Clear All Constraints* на панелі інструментів.

3. Зміна атрибутів елемента інтерфейсу

3.1 Зміна розміру кнопки

3.1.1 Виберіть верхній *Button* на панелі *Component Tree*.

3.1.2 Клацніть вкладку *Attributes* праворуч від вікна редактора макета.

3.1.3 Клацніть елемент керування шириною двічі – перший клік змінює його на *Fixed* з прямими лініями, а другий клацання змінює його на *Match Constraints* з пружинами. В результаті зміни елемента керування шириною атрибут *layout_width* на панелі *Attributes* відображає значення *match_constraint*, а елемент *Button* розтягується горизонтально, щоб заповнити простір між лівою та правою сторонами макета.

3.1.4 Виберіть другий *Button* та внесіть ті самі зміни в *layout_width* так само, як вище.

Як показано вище, атрибути *layout_width* та *layout_height* на панелі *Attributes* змінюються разом із елементами керування висотою та шириною в інспекторі. Ці атрибути можуть приймати одне з трьох значень для макета, які належать *ConstraintLayout*:

- *match_constraint* – розширює елемент *View*, щоб заповнити його батьківський контейнер по ширині або висоті – аж до краю, якщо він встановлений. У цьому випадку батьком є *ConstraintLayout*;
- *wrap_content* – скорочує розміри елемента *View* так, що він стає досить великим, щоб вмістити свій контент. Якщо контенту немає, елемент *View* стає невидимим;
- щоб вказати фіксований розмір, який відповідає розміру екрана пристрою, використовуйте фіксовану кількість пікселів (одиниць *dp*), що не залежать від зернистості екрана (*density-independent*).

Порада. *Я*кщо ви змінюєте атрибут *layout_width*, використовуючи його спливаюче меню, цей атрибут буде нульовим, оскільки відсутній встановлений розмір. Цей параметр такий самий, як *match_constraint – View* може максимально розширитися, щоб задовольнити прив'язки меж та налаштування полів.

3.2 Зміна атрибутів кнопки

3.2.1 Вибравши перший *Button*, встановіть поле *ID* у верхній частині області *Attributes* у значення *button_toast* для атрибута *android:id*.

3.2.2 Встановіть атрибут *background* у значення *@color/colorPrimary* (Коли введете *@c*, з'являться варіанти для легшого вибору).

3.2.3 Встановіть атрибут *textColor* у значення *@android:color/white*.

3.2.4 Встановіть атрибут *text* у значення *Toast*.

3.2.5 Виконайте ті самі зміни атрибутів для другого *Button*, використовуючи *button_count* як ідентифікатор *ID*; *Count* для атрибута text та ті ж кольори для фону та тексту, що і вище.

Колір *colorPrimary* – основний колір теми, один із заздалегідь визначених базових кольорів теми, визначених у файлі ресурсів *colors.xml*. Він використовується для панелі додатків. Використання основних кольорів для інших елементів інтерфейсу створює єдиний інтерфейс.

4. Додавання TextEdit та встановлення його атрибутів

4.1 Додавання TextView та меж

4.1.1 Перетягніть *TextView* з панелі *Palette* у верхню частину макета та перетягніть прив'язку з верхньої частини *TextView* до кружка внизу *Toast Button*. Це прив'яже *TextView* до нижньої межі *Button*.

4.1.2 Перетягніть обмеження знизу *TextView* до ручки у верхній частині *Count Button* та з боків *TextView* до боків макета. Це прив'язує *TextView* посередині макета між двома елементами *Button*.

4.2 Встановлення атрибутів TextView

4.2.1 Для вибраного *TextView* відкрийте область атрибутів.

4.2.2 Встановіть *ID* у значення *show_count*.

4.2.3 Встановіть *text* у значення *0*.

4.2.4 Встановіть *textSize* у значення *160sp*. *sp* означає *scale-independent pixel* – масштабно-незалежний піксель.

4.2.5 Встановіть *textStyle* у значення *B* (жирний шрифт) і *textAlignment* до *ALIGNCENTER* (абзац по центру).

4.2.6 Змініть елементи керування розміром горизонтального та вертикального перегляду (*layout_width* і *layout_height*) на *match_constraint*.

4.2.7 Встановіть *textColor* у значення *@color/colorPrimary*.

Прокрутіть область вниз і натисніть *View all attributes*, прокрутіть другу сторінку атрибутів до *background*, а потім введіть *#FFFF00* для відтінку жовтого.

4.2.8 Прокрутіть вниз *gravity*, розгорніть *gravity* і виберіть *center_ver* (для вертикального центрального вирівнювання). Атрибут *gravity* вказує, як *View* вирівнюється в межах батьківського *View* або *ViewGroup*.

Ви можете помітити, що атрибут *background* знаходиться на першій сторінці області *Attributes* для *Button*, але на другій сторінці області атрибутів для *TextView*. Панель атрибутів змінюється для кожного типу *View*: найпопулярніші атрибути типу *View* з'являються на першій сторінці, а решта – на другій. Щоб повернутися на першу сторінку області *Attributes,* клацніть на дві стрілки на панелі інструментів у верхній частині області.

5. Редагування макету в XML

5.1 Відкривання XML-коду для макета

5.1.1 Для цього завдання відкрийте файл *activity_main.xml* і натисніть вкладку *Text* внизу редактора макета.

5.1.2 З'являється редактор XML, який замінює панелі дизайну та проекту. Ви побачите, що рядки *"Toast"* та *"Count"* – жорстко прописані (*hardcoded*). Наведіть вказівник на такий рядок, щоб побачити попереджувальне повідомлення.

5.2 Застосування стрічкових ресурсів

5.2.1 Клацніть один раз на слово *"Toast"* (перше виділене попередження).

5.2.2 Натисніть клавішу *Alt+Enter* у *Windows* або *Option+Enter* у *macOS* та оберіть команду *Extract string resource* у випадаючому меню.

5.2.3 Введіть *button_label_toast* для імені ресурсу.

5.2.4 Клацніть *OK*. У файлі *values/res/string.xml* створюється стрічковий ресурс, а рядок у вашому коді замінюється посиланням на ресурс: *@string/button_label_toast*

5.2.5 Встановіть решту значень: *button_label_count* для *"Count"* і *count_initial_value* для *"0"*.

5.2.6 На панелі *Project* > *Android* розгорніть значення в розділі *res*, а потім двічі клацніть *strings.xml*, щоб переглянути свої файлові ресурси:

```
<resources>
    <string name="app_name">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
</resources>
```

Вам потрібно ще один рядок для використання в наступному завданні, яке відображає повідомлення. Додайте до файлу *strings.xml* ще один стрічковий ресурс, названий *toast_message* для фрази *"Hello Toast!"*:

```
<resources>
    <string name="app_name">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
    <string name="toast_message">Hello Toast!</string>
</resources>
```

Порада. Стрічкові ресурси містять назву програми, яка з'являється на панелі програм у верхній частині екрана, якщо ви запускаєте проект програми за допомогою порожнього шаблону. Ви можете змінити назву програми, відредагувавши ресурс *app_name*.

6. Додавання обробника *onClick* для кнопок

6.1 Додавання атрибута *onClick* і обробника події до кожної кнопки

6.1.1 Відкривши редактор XML (вкладка *Текст*), знайдіть *Button* із *android:id*, встановленим у значення *button_toast*:

```
<Button
        android:id="@+id/button_toast"
        android:layout_width="0dp"
        ...
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

6.1.2 Додайте атрибут *android:onClick* до кінця елемента *button_toast* після останнього атрибута та перед показником кінця />: `android:onClick="showToast" />`

Клацніть на червону лампочку, яка з'явиться поруч із атрибутом. Виберіть *Create click handler*, оберіть *MainActivity* та натисніть OK. Якщо піктограма червоної лампочки не відображається, клацніть назву методу (*«showToast»*). Натисніть *Alt+Enter*(*Option+Enter* на *Mac*), виберіть *Create 'showToast(view)' in MainActivity* і натисніть OK. Ця дія створює порожній метод-заповнювач для *showToast()* в *MainActivity*.

Повторіть два останні кроки за допомогою *button_count Button*: Додайте атрибут *android:onClick* до кінця та додайте обробник клацання: `android:onClick="countUp" />`

Код xml для елементів інтерфейсу користувача в *ConstraintLayout* тепер повинен виглядати так:

```
<Button
        android:id="@+id/button_toast"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@color/colorPrimary"
        android:text="@string/button_label_toast"
        android:textColor="@android:color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:onClick="showToast"/>
```

```
    <Button
        android:id="@+id/button_count"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:background="@color/colorPrimary"
        android:text="@string/button_label_count"
        android:textColor="@android:color/white"

app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        android:onClick="countUp" />
    <TextView
        android:id="@+id/show_count"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="#FFFF00"
        android:gravity="center_vertical"
        android:text="@string/count_initial_value"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary"
        android:textSize="160sp"
        android:textStyle="bold"

app:layout_constraintBottom_toTopOf="@+id/button_coun
t"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
app:layout_constraintTop_toBottomOf="@+id/button_toas
t" />
```

Якщо *MainActivity.java* ще не відкритий, розгорніть *java* у *Project>Android*, розгорніть *com.example.android.hellotoast*, а потім подвійно клацніть *MainActivity*. Редактор коду покаже код MainActivity:

```
package com.example.android.hellotoast;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void showToast(View view) {
    }
    public void countUp(View view) {
    }
}
```

6.1 Редагування обробника кнопки «Тост»

6.1.1 Знайдіть щойно створений метод *showToast()*.

```
public void showToast(View view) {
}
```

6.1.2 Щоб створити екземпляр *Toast*, викликайте фабричний метод *makeText()* класу *Toast*.

```
public void showToast(View view) {
    Toast toast = Toast.makeText(
}
```

Цей код є неповним, доки ви не закінчите всі кроки.

6.1.3 Оскільки Toast-повідомлення відображається поверх інтерфейсу користувача *Activity*, система потребує інформації про поточний *Activity*. Коли ви вже перебуваєте в контексті *Activity*, чий контекст вам потрібен, використовуйте *this* як скорочений запис. `Toast toast=Toast.makeText(this,`

6.1.4 Надайте повідомлення для відображення, наприклад рядок (*toast_message* створений вами на попередньому кроці). Стрічковий ресурс *toast_message* ідентифікується через *R.string*.

```
Toast toast=Toast.makeText(this,
R.string.toast_message,
```

Вкажіть тривалість показу. Наприклад, *Toast.LENGTH_SHORT* покаже наше Toast-повідомлення на порівняно короткий час.

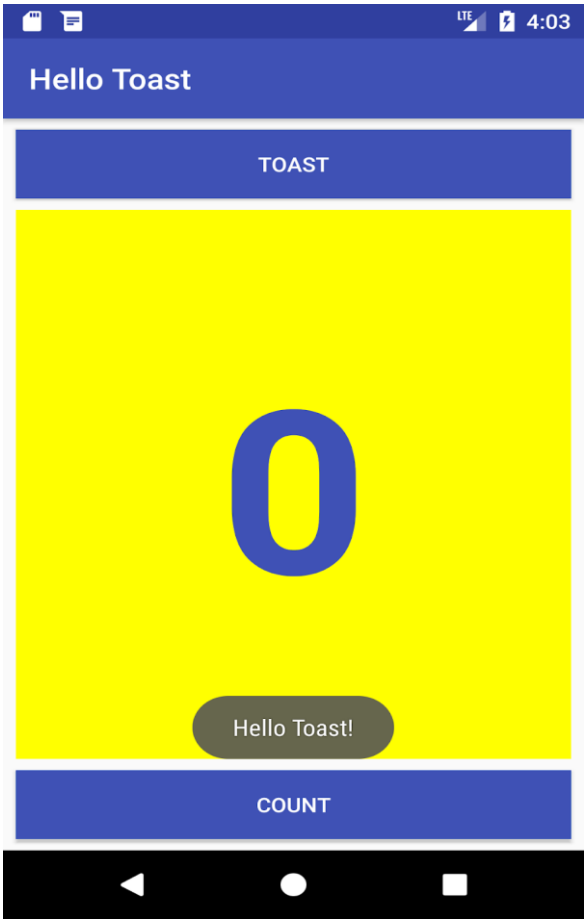
```
Toast toast=Toast.makeText(this,
R.string.toast_message,

Toast.LENGTH_SHORT);
```

Тривалість відображення *Toast* може бути *Toast.LENGTH_LONG* або *Toast.LENGTH_SHORT*. Фактична тривалість становить близько 3,5 секунди для довгого *Toast* та 2 секунди для короткого *Toast*.

Покажіть Toast, викликавши *show()*. Нижче наведено весь метод *showToast()*:

```
public void showToast(View view) {
   Toast toast=Toast.makeText(this,
R.string.toast_message,

Toast.LENGTH_SHORT);
   toast.show();
}
```

Запустіть програму та переконайтеся, що Toast-повідомлення з’являється при натисканні кнопки *Toast*.



Контрольні запитання

1. Підкласами якого класу є всі елементи графічного інтерфейсу UI?
2. Який клас описує контейнер для нащадків графічного інтерфейсу UI?
3. Що таке жорстко прописані (hardcoded) значення? Як їх змінювати?
4. Як задати код поведінки при натисненні на кнопку?
5. Як задати тривалість відображення для Toast-повідомлення?
6. Як змінити назву вашої програми?
7. Як задавати прив’язку меж елементів?

Перелік посилань

1. Конспект лекцій з курсу «Програмування для мобільних пристроїв» для здобувачів освітнього ступеня «бакалавр», спеціальності 126 «Інформаційні системи та технології», денної форми навчання. Укладачі: Готович В. А., Михайлович Т. В.: ТНТУ, 2020 р.
2. https://codelabs.developers.google.com/codelabs/android-training-layout-editor-part-a/index.html?index=..%2F..%2Fandroid-training#0
3. https://developer.android.com/studio/write/layout-editor.html
4. https://developer.android.com/training/constraint-layout/index.html
5. http://developer.android.com/guide/topics/ui/declaring-layout.html
6. http://developer.android.com/reference/android/content/Context.html
7. https://codelabs.developers.google.com/codelabs/constraint-layout/index.html

ЛАБОРАТОРНА РОБОТА № 4.
ЗАСТОСУВАННЯ ДЛЯ ОФОРМЛЕННЯ ІНТЕРФЕЙСУ ДОДАТКУ СТИЛІВ ТА ТЕМ

Мета: метою виконання лабораторної роботи є набуття студентами навичок:

- покращення вигляду інтерфейсу програми;
- зменшення кількість коду, необхідного для компонентів інтерфейсу;
- визначення та застосування стилів і тем.

Короткі теоретичні відомості

Крім застосування окремих стилів до окремих елементів інтерфейсу, ми можемо задавати стилі для всієї програми або *activity* у вигляді тем.

Ми можемо самі створити тему. Однак Android вже надає кілька попередньо встановлених тем для стилізації застосунків, наприклад, *Theme.AppCompat.Light.DarkActionBar* і ряд інших.

Для визначення тем застосунку відкриємо файл *AndroidManifest.xml*. У ньому ми можемо побачити наступне визначення елемента *application*, що представляє застосунок:

```
<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">
```

Задання теми відбувається за допомогою атрибута *android:theme*. В даному випадку використовується ресурс, визначений в стилях — в файлі *res/values/ styles.xml*:

```
<style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
```

```
    <item
name="colorPrimary">@color/colorPrimary</item>
    <item
name="colorPrimaryDark">@color/colorPrimaryDark</item
>
    <item
name="colorAccent">@color/colorAccent</item>
</style>
```

Стиль *AppTheme* використовує вбудовану тему *Theme.AppCompat.Light.DarkActionBar*, яка задає візуальні характеристики нашого застосунку. Тепер визначимо стиль, який використовує іншу тему:

```
<resources>
    <style name="NoActionBarTheme"
parent="Theme.AppCompat.DayNight.NoActionBar">
    </style>
    <style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item
name="colorPrimary">@color/colorPrimary</item>
        <item
name="colorPrimaryDark">@color/colorPrimaryDark</item
>
        <item
name="colorAccent">@color/colorAccent</item>
    </style>
    <style name="TextViewStyle">
        <item
name="android:layout_width">match_parent</item>
        <item
name="android:layout_height">50dp</item>
        <item name="android:textColor">#3f51b5</item>
        <item name="android:textSize">22sp</item>
        <item name="android:gravity">center</item>
```

```
    </style>
</resources>
```

Нехай новий стиль називається *NoActionBarTheme*, який посилається на тему *Theme.AppCompat.DayNight.NoActionBar*. Тепер встановимо його в якості теми застосунку в файлі *AndroidManifest.xml*:

```
<application
    android:theme="@styles/NoActionBarTheme"
```

Розроблюваний в даній роботі додаток *Scorekeeper* складається з двох наборів елементів кнопки та двох елементів *TextView*, які використовуються для відстеження рахунку в грі з двома гравцями.

Хід роботи

1. Створення додатку Scorekeeper

1.1 Створення нового проекту

1.1.1 Запустіть Android Studio і створіть новий проект Android Studio з назвою *Scorekeeper*.

1.1.2 Прийміть мінімальний SDK за замовчуванням і виберіть шаблон *Empty Activity*.

1.1.3 Встановіть назву за замовчуванням (*MainActivity*) та переконайтесь, що відмічено опції *Generate Layout file* та *Backwards Compatibility (AppCompat).*

1.1.4 Клацніть *Готово*.

1.2 Створення макета для MainActivity

1.2.1 Відкрийте *activity_main.xml* та натисніть вкладку *Text*, щоб побачити xml-код. Угорі або в кореневій частині ієрархії представлення є *ViewGroup ConstraintLayout*:

android.support.constraint.ConstraintLayout

1.2.2 Змініть цю *ViewGroup* на *LinearLayout*. Другий рядок коду зараз виглядає приблизно так:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/and
andr"
```

1.2.3 Видаліть наступний рядок коду XML, який пов’язаний з ConstraintLayout:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

Блок XML-коду вгорі повинен виглядати так:

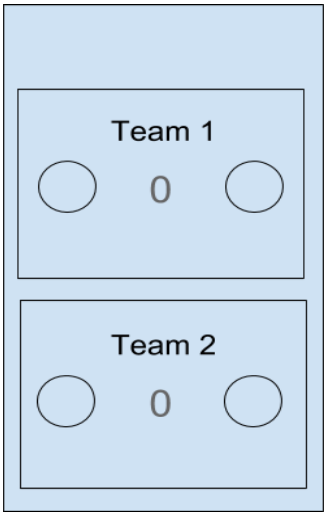
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

tools:context="com.example.android.scorekeeper.MainAc
tivity">
```

Додайте такі атрибути (не видаляючи існуючих атрибутів):

- `android:orientation="vertical"`;
- `android:padding="16dp"`.

1.3 Створення контейнерів для оцінки

Макет цього додатка включає контейнери оцінок, визначені двома елементами *RelativeLayout* групи перегляду – по одному для кожної команди. На наступній схемі показаний макет у найпростішому вигляді.



1.3.1 Всередині *LinearLayout* додайте два *RelativeLayout* елементи з такими атрибутами:

- `android:layout_width="match_parent";`
- `android:layout_height="0dp";`
- `android:layout_weight="1".`

Ви можете бути здивовані, побачивши, що для атрибута *layout_height* встановлено значення *0dp*. Це тому, що ми використовуємо атрибут *layout_weight*, щоб визначити, скільки місця займають ці елементи *RelativeLayout* в батьківському *LinearLayout*.

1.3.2 Додайте два *ImageButton* елементи до кожного *RelativeLayout*: один для зменшення оцінки та другий для збільшення оцінки. Використовуйте ці атрибути:

```
android:id="@+id/decreaseTeam1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_centerVertical="true"
```

Використайте *increaseTeam1* як *android:id* для другого *ImageButton*, що збільшує оцінку, *decreaseTeam2* та *increaseTeam2* для третього і четвертого *ImageButton*.

1.3.3 Додайте *TextView* до кожного *RelativeLayout* між елементами *ImageButton* для відображення оцінки. Використовуйте ці атрибути:

```
android:id="@+id/score_1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:text="0"
```

Використайте *score_2* як *android:id* для другого *TextView* між елементами *ImageButton*.

1.3.4 Додайте ще одну *TextView* до кожної *RelativeLayout* вище оцінки, щоб представляти назви команд. Використовуйте ці атрибути:

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:text="Team 1"
```

Використайте "*Team 2*" як значення *android:text* для другого *TextView*.

1.4 Додавання векторних ресурсів

1.4.1 Виберіть *File>New>Vector Asset*, щоб відкрити *Vector Asset Studio*.

1.4.2 Клацніть на піктограму, щоб відкрити список файлів матеріальних значків. Виберіть категорію *Content*.

1.4.3 Виберіть піктограму додавання та натисніть *OK* .

1.4.4 Перейменуйте файл ресурсу *ic_plus* і встановіть прапорець *Override* поруч із параметрами розміру.

1.4.5 Змініть розмір піктограми на *40dp x 40dp* .

1.4.6 Натисніть *Next*, а потім *Finish*.

1.4.7 Повторіть цей процес, щоб додати піктограму видалення та назвати файл *ic_minus*.

1.4.8 Додайте наступні атрибути до елементів *ImageButton* у лівій частині макета:

```
android:src="@drawable/ic_minus"
android:contentDescription="Minus Button"
```

1.4.9 Додайте наступні атрибути до елементів *ImageButton* у правій частині макета:

```
android:src="@drawable/ic_plus"
android:contentDescription="Plus Button"
```

1.4.10Витягніть усі свої рядкові ресурси. Цей процес видаляє всі ваші рядки з коду Java і поміщає їх у файл *strings.xml*. Це дозволяє легко перекладати вашу програму різними мовами.

1.5 Ініціалізаація елементів TextView та змінних підрахунку балів

1.5.1 Створіть дві цілочисельні змінні, що представляють оцінку кожної команди.

```
// Member variables for holding the score
private int mScore1;
private int mScore2;
```

1.5.2 Створіть дві *TextView*-змінні для зберігання посилань на TextView елементи.

```
// Member variables for holding the score
private int mScore1;
private int mScore2;
```

1.5.3 У методі *onCreate()* для *MainActivity* знайдіть свої елементи TextView для рахунку за *id* та призначте їх змінним.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Find the TextViews by ID
        mScoreText1=
(TextView)findViewById(R.id.score_1);
        mScoreText2 =
(TextView)findViewById(R.id.score_2);
}
```

1.6 Задання обробників клацань для елементів ImageButton

1.6.1 Відкрийте *activity_main.xml* і додайте наступний атрибут *android:onClick* до першого *ImageButton* в лівій частині макета:

```
android:onClick="decreaseScore"
```

1.6.2 Назву методу *decreaseScore* підкреслено червоним кольором. Клацніть на піктограму червоної лампочки на лівому полі або клацніть на назву методу та натисніть *Alt+Enter*(*Option+Enter*) і виберіть *Create 'decreaseScore(view)' in 'MainActivity'*. Android Studio створює порожній метод *decreaseScore()* в *MainActivity*.

1.6.3 Додайте вищезазначений атрибут *android:onClick* до другого *ImageButton* зліва на макеті. Цього разу ім'я методу є дійсним, оскільки метод вже створений.

1.6.4 Додайте наступний атрибут *android:onClick* до кожного ImageButton з правого боку макета:

```
android:onClick="increaseScore"
```

1.6.5 Назва методу *increaseScore* підкреслена червоним кольором. Клацніть на піктограму червоної лампочки на лівому полі або клацніть на назву методу та натисніть *Alt+Enter* (*Option+Enter*) і виберіть *Create 'increaseScore(view)' in 'MainActivity'*. Android Studio створює порожній метод *increaseScore()* в *MainActivity*.

1.6.6 Додайте код до методів, щоб зменшити та збільшити оцінку, як показано нижче. Обидва методи використовують *view.getID()* для отримання ідентифікатора команди, на *ImageButton* яку клацнули, щоб ваш код міг оновити рахунок відповідної команди.

```
/* Method that handles the onClick of both the
decrement buttons
@param view The button view that was clicked */
public void decreaseScore(View view) {
   // Get the ID of the button that was clicked
   int viewID = view.getId();
   switch (viewID){
       //If it was on Team 1
       case R.id.decreaseTeam1:
           //Decrement the score and update the
TextView
           mScore1--;

mScoreText1.setText(String.valueOf(mScore1));
```

```
            break;
        //If it was Team 2
        case R.id.decreaseTeam2:
            //Decrement the score and update the
TextView
            mScore2--;

mScoreText2.setText(String.valueOf(mScore2));
    }
}
/* Method that handles the onClick of both the
increment buttons
@param view The button view that was clicked */
public void increaseScore(View view) {
    //Get the ID of the button that was clicked
    int viewID = view.getId();
    switch (viewID){
        //If it was on Team 1
        case R.id.increaseTeam1:
            //Increment the score and update the
TextView
            mScore1++;

mScoreText1.setText(String.valueOf(mScore1));
            break;
        //If it was Team 2
        case R.id.increaseTeam2:
            //Increment the score and update the
TextView
            mScore2++;

mScoreText2.setText(String.valueOf(mScore2));
    }
}
```

2. Створення ресурсу для малювання

2.1 Створення ShapeDrawable

ShapeDrawable – це примітивна геометрична форма, визначена у xml-файлі низкою атрибутів, включаючи колір, форму, відступи тощо. Він визначає векторну графіку, яку можна масштабувати вгору і вниз, не втрачаючи роздільної здатності.

2.1.1 На панелі *Project* > *Android* клацніть правою кнопкою миші (або лівою кнопкою при натисненому *Control*) на папці *drawable* у каталозі *res*.

2.1.2 Виберіть *New* > *Drawable resource file*.

2.1.3 Назвіть файл *button_background* і натисніть кнопку *OK*. (Не змінюйте *Source set* чи *Directory name* і не додайте кваліфікаторів). Android Studio створює файл *button_background.xml* у папці *drawable*.

2.1.4 Відкрийте *button_background.xml*, натисніть вкладку *Text*, щоб відредагувати xml-код, і видаліть увесь код, за винятком:

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

Додайте наступний код, який створює овальну форму з контуром:

```
<shape

xmlns:android="http://schemas.android.com/apk/res/and
roid"
    android:shape="oval">
        <stroke
            android:width="2dp"
            android:color="@color/colorPrimary"/>
</shape>
```

2.2 Застосування ShapeDrawable в якості фону

2.2.1 Відкрийте a*ctivity_main.xml*.

2.2.2 Додайте графічний малюнок як фон для всіх чотирьох елементів ImageButton:

```
android:background="@drawable/button_background"
```

2.2.3 Перейдіть на вкладку *Preview* в редакторі макета, щоб побачити, що фон автоматично масштабується відповідно до розміру *ImageButton*.

2.2.4 Щоб візуалізувати кожну *ImageButton* належним чином на всіх пристроях, ви змінюєте атрибути *android:layout_height* та *android:layout_width* для кожної *ImageButton* до значення *70dp*, що є великим розміром на більшості пристроїв. Змініть перший атрибут для першого *ImageButton* таким чином: `android:layout_width="70dp".`

2.2.5 Клацніть один раз на *"70dp"*, натисніть *Alt+Enter* у *Windows* або *Option+Enter* в *macOS* та виберіть *Extract dimension resource* зі спливаючого меню.

Введіть *button_size* у поле *Resource name*.

2.2.6 Натисніть кнопку *OK*. Це створює ресурсний параметр у файлі *dimens.xml* (у папці *values*), а розмірність у вашому коді замінюється посиланням на ресурс:

```
@dimen/button_size
```

2.2.7 Змініть атрибути *android:layout_height* та *android:layout_width* для кожного елемента *ImageButton*, використовуючи новий ресурсний параметр:

```
android:layout_width="@dimen/button_size"
android:layout_height="@dimen/button_size"
```

2.2.8 Клацніть на вкладці *Preview* у редакторі макета, щоб побачити макет. Тепер ShapeDrawable використовується для елементів ImageButton.

3. Стилізація елементів View

Коли ви продовжите додавати елементи перегляду та атрибути у свій макет, ваш код почне ставати великим і повторюваним, особливо коли ви застосовуєте однакові атрибути до багатьох подібних елементів. Стиль може визначати загальні властивості, такі як відступи, колір шрифту, розмір шрифту та колір тла. Атрибути, орієнтовані на компонування, такі як висота, ширина та відносне розташування, повинні залишатися у файлі ресурсу макета.

На наступному кроці ви дізнаєтесь, як створювати стилі та застосовувати їх до кількох елементів перегляду та макетів, дозволяючи одночасно оновлювати загальні атрибути з одного місця.

3.1 Створення стилів кнопок

Будь-які атрибути стилю наслідуються з батьківського елемента допоки не будуть явно задані. Також можна добавляти власні атрибути, яких немає в батьківського елемента.

Наприклад, усі чотири елементи *ImageButton* у додатку *Scorekeeper* мають спільний фон *Drawable*, але з різними піктограмами «плюс» (збільшити рахунок) та «мінус» (зменшити рахунок). Крім того, два елементи *ImageButton* мають однаковий значок, як і два «мінус» елементи *ImageButton*. Тому ви можете створити три стилі:

- стиль для всіх елементів *ImageButton*, який включає властивості *ImageButton* типово, а також фон *Drawable*;
- стиль для елементів «мінус» *ImageButton*. Цей стиль успадковує атрибути стилю *ImageButton* і включає піктограму «мінус»;
- стиль для елементів «плюс» *ImageButton*. Цей стиль успадковується від стилю ImageButton і включає піктограму “плюс”.

3.1.1 Розгорніть *res* > *value* на панелі *Project*>*Android* та відкрийте файл *styles.xml*.

Тут буде розміщений весь код вашого стилю. Стиль "*AppTheme*" завжди додається автоматично, і ви можете бачити, що він походить від *Theme.AppCompat.Light.DarkActionBar*:

```
<style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
```

Зверніть увагу на батьківський атрибут, який визначає стиль вашого батьків за допомогою xml.

Атрибут *name*, в даному випадку *AppTheme*, визначає назву стилю. Батьківський атрибут, у цьому випадку *Theme.AppCompat.Light.DarkActionBar*, оголошує атрибути, які успадковує *AppTheme*. У цьому випадку це типова тема для Android: зі світлим фоном та темною смужкою дій.

Тема – це стиль, який застосовується до всієї *Activity* або програми замість одного *View*. Використання теми створює однаковий вигляд у всій *Activity* або додатку, наприклад, однаковий вигляд і панель програм у кожній частині вашої програми.

3.1.2 Між тегами <*resources*> додайте новий стиль із наступними атрибутами, щоб створити загальний стиль для всіх кнопок:

```
<style name="ScoreButtons"
parent="Widget.AppCompat.Button">
    <item
name="android:background">@drawable/button_background
</item>
</style>
```

Вищенаведений фрагмент задає батьківський стиль *Widget.AppCompat.Button*, щоб зберегти типові атрибути кнопки. Він також додає атрибут, який змінює тло Drawable на той, який ви створили в попередньому завданні.

3.1.3 Створіть стиль для кнопок «плюс», розширивши стиль *ScoreButtons*:

```
<style name="PlusButtons" parent="ScoreButtons">
    <item name="android:src">@drawable/ic_plus</item>
    <item
name="android:contentDescription">@string/plus_button
_description</item>
</style>
```

Атрибут *contentDescription* призначений для користувачів із вадами зору. Він діє як мітка, яку деякі пристрої доступності (*accessibility*) використовують для читання вголос, щоб забезпечити деяку інформацію про значення елемента інтерфейсу.

3.1.4 Створіть стиль для «мінусових» кнопок:

```
<style name="MinusButtons" parent="ScoreButtons">
    <item name="android:src">@drawable/ic_minus</item>
    <item name=
"android:contentDescription">@string/minus_button_des
cription</item>
</style>
```

3.1.5 Тепер ви можете використовувати ці стилі для заміни конкретних атрибутів стилю елементів *ImageButton*. Відкрийте файл макета

activity_main.xml для *MainActivity* та замініть наступні атрибути для обох мінус елементів *ImageButton*:

- видаліть ці атрибути: `android:src`, `android:contentDescription`, `android:background`;
- додайте такий атрибут: `style="@style/MinusButtons"`.

Примітка: атрибут *style* не використовує простір імен *android:*, тому що *style* – це типова частина атрибутів xml.

3.1.6 Замініть наступні атрибути для обох "плюс" елементів *ImageButton*:

- видаліть ці атрибути: `android:src`, `android:contentDescription`, `android`:background;
- додайте такий атрибут: `style="@style/PlusButtons"`.

3.2 Створення стилів TextView

Назву команди та відображення результатів *TextView* також можна стилізувати, оскільки вони мають загальні кольори та шрифти. Зробіть наступне:

3.2.1 Додайте наступний атрибут до всіх елементів *TextView*:

```
android:textAppearance="@style/TextAppearance.AppComp
at.Headline"
```

3.2.2 Клацніть правою кнопкою миші (або лівою разом із *Control*) в будь-якому місці атрибутів *TextView* першої оцінки та виберіть *Refactor>Extract>Style ...*

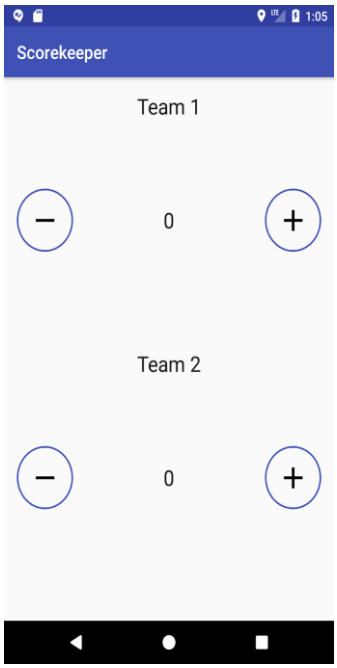
3.2.3 Назвіть стиль *ScoreText* і встановіть прапорець *textAppearance* (атрибут, який ви тільки що додали), а також *Launch 'Use Styles Where Possible' refactoring after the style is extracted*. Це перевірить файл макета на *View* з однаковими атрибутами і застосує до них стиль. Не вилучайте атрибути, пов'язані з компонуванням – натисніть на відповідні прапорці, щоб вимкнути їх.

3.2.4 Виберіть *OK*.

3.2.5 Переконайтеся, що область *scope* встановлено на файл макета *activity_main.xml* і натисніть кнопку *OK*.

3.2.6 Панель Android Studio відкриється внизу, якщо такий же стиль знайдено в інших *View*. Виберіть *Do Refactor*, щоб застосувати новий стиль до всіх *View* із тими ж атрибутами.

3.2.7 Запустіть додаток. Ви повинні побачити таке вікно:



Ніяких змін не повинно бути, за винятком того, що весь ваш код стилю тепер знаходиться у вашому файлі ресурсів, а файл макета зменшено.

3.3 Оновлення стилів

Потужність використання стилів стає очевидною, коли потрібно внести зміни до декількох елементів одного стилю. Ви можете зробити текст більшим, жирнішим і яскравішим, а також змінити піктограми на колір тла кнопок.

3.3.1 Відкрийте файл *styles.xml* та додайте або змініть такі атрибути для стилів:

- *ScoreButtons*: `<item name="android:tint">@color/colorPrimary</item>`;
- *ScoreText*: `<item name="android:textAppearance"> @style/TextAppearance.AppCompat.Display3</item>`.

Значення *colorPrimary* автоматично задається із Android Studio під час створення проекту. Ви можете знайти його на панелі *Project>Android* у файлі *color.xml* всередині папки значень. Атрибут *TextAppearance.AppCompat.Display3* – це заздалегідь визначений стиль тексту, що надається Android.

3.3.2 Створіть новий стиль під назвою *TeamText* з атрибутом *textAppearance* встановленим таким чином:

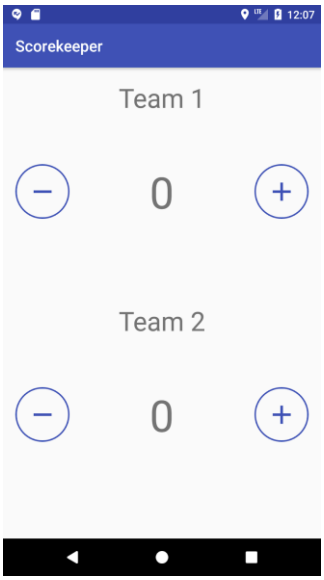
```
<style name="TeamText">
```

```
    <item name=

"android:textAppearance">@style/TextAppearance.AppCom
pat.Display1
    </item>
</style>
```

3.3.3 У *activity_main.xml* змініть атрибут стилю елементів імені команди *TextView* на новостворений стиль *TeamText*.

```
style="@style/TeamText"
```

3.3.4 Запустіть додаток. Вже лише із цими корективами у файлі *style.xml*, вигляд усіх View оновлено.



4. Теми та фінальні кроки

Ви побачили, що елементи перегляду з подібними характеристиками можна стилізувати разом у файлі *styles.xml*. Але що робити, якщо ви хочете визначити стилі для всієї *Activity* чи всього додатка? Зробити це можна за допомогою тем. Щоб встановити тему для кожної *Activity*, потрібно змінити файл *AndroidManifest.xml*.

У цьому завданні ви додасьте тему "нічного режиму" у свою програму, що дозволить користувачам використовувати низькоконтрастну

версію інтерфейсу вашого додатка, яка легша для очей у нічний час, а також дещо «відшліфуєте» UI.

4.1 Оновлення стилів

4.1.1 Відкрийте файл *AndroidManifest.xml*, знайдіть тег <*application*> та змініть атрибут *android:topic* на:

```
android:theme="@style/Theme.AppCompat.Light.NoActionBar"
```

Це заздалегідь визначена тема, яка видаляє панель дій із вашої *Activity*.

4.1.2 Запустіть додаток. Панель інструментів зникне!

4.1.3 Поверніть тему програми назад на *AppTheme*, що є дочірньою темою *Theme.Appcompat.Light.DarkActionBar*, як це можна побачити в *styles.xml*.

Щоб застосувати тему до *Activity* замість усієї програми, помістіть атрибут теми в тег <*Activity*> замість тегу <*application*>.

4.2 Додавання кнопки зміни теми у меню

Одне із достоїнств налаштування теми для вашого додатка – це надання альтернативного вигляду для перегляду вночі. У таких умовах часто краще мати низький контраст, темний фон. В рамках Android передбачена тема, яка призначена саме для цього: *DayNight*.

Ця тема має вбудовані параметри, які дозволяють програмно керувати кольорами у вашій програмі: встановивши її автоматично змінюватись за часом, або за командою користувача.

На цьому кроці ви додаєте пункт меню опцій, який переключатиме додаток між звичайною темою та темою «нічний режим».

4.2.1 Відкрийте *strings.xml* і створіть два рядкових ресурси для цього пункту меню параметрів:

```
<string name="night_mode">Night Mode</string>
<string name="day_mode">Day Mode</string>
```

4.2.2 Клацніть правою кнопкою миші (або лівою із натисненим *Control*) у каталозі res на панелі *Project>Android* та виберіть *New>Android resource file*.

4.2.3 Назвіть файл *main_menu*, змініть *Resource Type* на *Menu* та натисніть кнопку ОК. З’являється редактор макетів для файлу *main_menu.xml*.

4.2.4 Перейдіть на вкладку *Text* редактора макета, щоб показати код xml.

4.2.5 Додайте пункт меню з такими атрибутами:

```
<item
    android:id="@+id/night_mode"
    android:title="@string/night_mode"/>
```

4.2.6 Відкрийте *MainActivity*, натисніть Ctrl+O, щоб відкрити меню *Override Method*, і виберіть метод *onCreateOptionsMenu(menu:Menu):boolean*, що розташований в категорії *android.app.Activity*.

4.2.7 Натисніть кнопку *OK*. Android Studio створює порожній метод *onCreateOptionsMenu()* із єдиним оператором *return super.onCreateOptionsMenu(menu)*.

4.3 Зміна теми в меню

Тема *DayNight* використовує клас *AppCompatDelegate* для встановлення параметрів нічного режиму у вашій *Activity*.

4.3.1 У файлі *styles.xml* змініть батьківське значення *AppTheme* на *"Theme.AppCompat.DayNight.DarkActionBar"*.

4.3.2 Замініть метод *onOptionsItemSelected()* в *MainActivity* і додайте код, щоб перевірити, на який пункт меню натиснули:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   //Check if the correct item was clicked
   if(item.getItemId()==R.id.night_mode){}
       // TODO: Get the night mode state of the app.
   return true;
}
```

4.3.3 Замініть коментар *TODO:* у наведеному вище фрагменті кодом, який перевіряє, чи нічний режим увімкнено. Якщо ввімкнено, код змінює нічний режим на відключений стан; в іншому випадку код дозволяє включити нічний режим:

```
if(item.getItemId()==R.id.night_mode){
    // Get the night mode state of the app.
```

```
    int nightMode =
AppCompatDelegate.getDefaultNightMode();
    //Set the theme mode for the restarted activity
    if (nightMode ==
AppCompatDelegate.MODE_NIGHT_YES) {
        AppCompatDelegate.setDefaultNightMode
 (AppCompatDelegate.MODE_NIGHT_NO);
} else {

AppCompatDelegate.setDefaultNightMode(AppCompatDelega
te.MODE_NIGHT_YES);
}
// Recreate the activity for the theme change to take
effect.
recreate();
```

У відповідь на натискання на пункт меню, код перевіряє поточний параметр нічного режиму, викликаючи *AppCompatDelegate.getDefaultNightMode()*.

Тема може змінюватися лише під час створення *Activity*, тому викликаємо *recreate()*, щоб зміна теми набула чинності.

4.3.4 Запустіть додаток. Елемент меню *Night Mode* тепер повинен перемикати тему вашої *Activity*.

4.3.5 Ви можете помітити, що пункт меню завжди називається *Night Mode* (нічний режим), що може бентежити вашого користувача, якщо програма вже у «темній» темі.

4.3.6 Замініть оператор *return super.onCreateOptionsMenu(menu)* у методі *onCreateOptionsMenu()* на наступний код:

```
// Change the label of the menu based on the state of
the app.
int nightMode =
AppCompatDelegate.getDefaultNightMode();
if(nightMode == AppCompatDelegate.MODE_NIGHT_YES){
menu.findItem(R.id.night_mode).setTitle(R.string.day_
mode);
```

```
} else{
menu.findItem(R.id.night_mode).setTitle(R.string.nigh
t_mode);
}
return true;
```

4.3.7 Запустіть додаток. Пункту меню *Night Mode* після того, як користувач натисне його, тепер змінює назву на *Day Mode* та перемикає тему.

4.4 SaveInstanceState

На попередніх заняттях ви дізналися, що ваша програма повинна бути готовою до того, що ваша *Activity* буде несподівано знищена та відтворена, наприклад, коли змінюється орієнтація екрану. У цьому додатку елементи *TextView*, що містять рахунок, скидаються до початкового значення 0 при повертанні пристрою. Щоб виправити це, виконайте наступне:

4.4.1 Відкрийте *MainActivity* і додайте теги під змінними, які будуть використовуватися як ключі в *onSaveInstanceState()*:

```
static final String STATE_SCORE_1 = "Team 1 Score";
static final String STATE_SCORE_2 = "Team 2 Score";
```

4.4.2 В кінці *MainActivity* замініть метод *onSaveInstanceState()*, щоб зберегти значення двох елементів TextView:

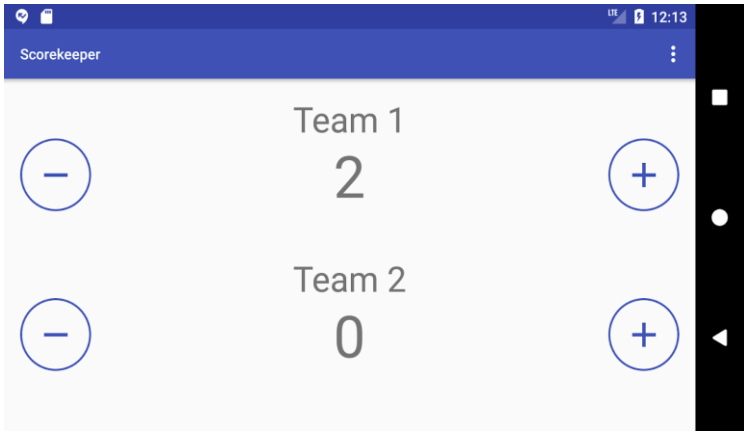
```
@Override
protected void onSaveInstanceState(Bundle outState) {
   // Save the scores.
   outState.putInt(STATE_SCORE_1, mScore1);
   outState.putInt(STATE_SCORE_2, mScore2);
   super.onSaveInstanceState(outState);
}
```

4.4.3 В кінці методу *onCreate()* додайте код, щоб перевірити наявність збереженого *InstanceState*. Якщо є, відновіть рахунок до елементів *TextView*:

```
if (savedInstanceState != null) {
```

```
    mScore1 =
savedInstanceState.getInt(STATE_SCORE_1);
    mScore2 =
savedInstanceState.getInt(STATE_SCORE_2);
    //Set the score text views
    mScoreText1.setText(String.valueOf(mScore1));
    mScoreText2.setText(String.valueOf(mScore2));
}
```

4.4.4 Запустіть додаток і натисніть *ImageButton* «плюс», щоб збільшити рахунок. Поверніть пристрій чи перемкніть емулятор на горизонтальну орієнтацію, щоб переконатися, що рахунок зберігається.



Контрольні запитання

1. Які елементи покращують вигляд інтерфейсу програми?
2. Що може зменшити кількість коду, необхідного для компонентів вашого інтерфейсу?
3. Чи може стиль включати інформацію, пов’язану з макетом?
4. Який атрибут необхідно задати, щоб успадкувати стиль?
5. Що називають темою в Android?
6. Який атрибут використовується для задання теми?

Перелік посилань

1. Конспект лекцій з курсу «Програмування для мобільних пристроїв» для здобувачів освітнього ступеня «бакалавр» спеціальності 126 «Інформаційні системи та технології» денної форми навчання. Укладачі : Готович В. А, Михайлович Т. В. ТНТУ, 2020 р.
2. https://codelabs.developers.google.com/codelabs/android-training-drawables-styles-and-themes/index.html?index=..%2F..%2Fandroid-training#0
3. https://developer.android.com/studio/write/vector-asset-studio.html
4. http://developer.android.com/tools/help/image-asset-studio.html
5. http://developer.android.com/training/best-ui.html
6. http://developer.android.com/guide/topics/resources/drawable-resource.html
7. http://developer.android.com/guide/topics/ui/themes.html
8. https://codelabs.developers.google.com/codelabs/android-training-drawables-styles-and-themes/index.html

ЛАБОРАТОРНА РОБОТА № 5.
РОЗРОБКА ДОДАТКУ ІЗ ЗАСТОСУВАННЯМ SHARED PREFERENCES

Мета: метою виконання лабораторної роботи є набуття студентами навичок:

- ідентифікації того, що можна віднести до Shared preferences;
- створення файлу Shared preferences для використання його додатком;
- збереження/читання даних з файлу Shared preferences;
- очищення даних Shared preferences.

Короткі теоретичні відомості

Shared preferences дозволяють зберігати невелику кількість примітивних даних у вигляді пар ключ/значення у файлі на пристрої. Для цього використовується відповідний об’єкт класу Shared preferences. Android framework управляє відповідним файлом із збереженими в ньому парами даних. Цей файл доступний для всіх компонентів вашої програми, але не доступний для інших програм. Дані, які зберігаються у Shared preferences, відрізняються від даних, які зберігаються у Activity State:

- дані в збереженому Activity State зберігаються для всіх екземплярів Acivity в межах одної і тої ж самої сесії користувача;
- Shared preferences зберігаються для всіх сесій користувача. Shared preferences зберігається, навіть якщо ваша програма зупиняється та перезапускається, або якщо пристрій перезавантажується.

Shared preferences слід застосовувати лише тоді, коли необхідно зберегти невелику кількість простих даних. Для управління більшою кількістю постійних даних програми слід застосовувати такі методи, як Room library або база даних SQL.

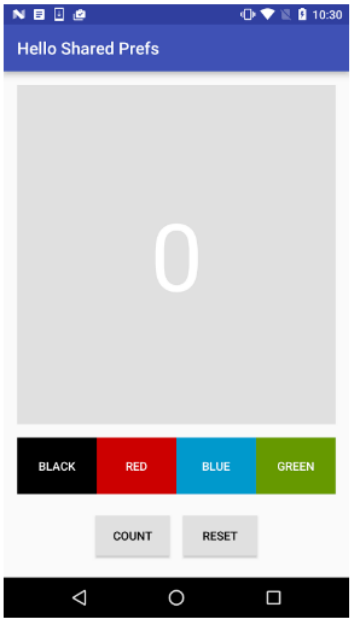
Хід роботи

1. Завантаження та аналіз проекту

1.1 Завантаження та запуск на виконання проекту

В лабораторній роботі використовується додаток *HelloSharedPrefs*, який необхідно завантажити за посиланням https://github.com/google-developer-training/android-fundamentals-starter-apps-v2/tree/master/HelloSharedPrefs.

Після завантаження проект необхідно відкрити в середовищі Android Studio, скомпілювати та запустити на виконання. Ви повинні побачити таке вікно:



Поекспериментуйте з роботою даного додатку, зокрема:

1. Натисніть кнопку *Count* для інкременту числа в головному text view (відлік);
2. Натисніть будь-яку із кольорових кнопок щоб змінити колір тла головного text view;
3. Змініть орієнтацію екрану віртуального пристрою і переконайтеся, що відлік та колір тла text view при цьому зберігаються;
4. Натисніть кнопку *Reset* для того, щоб встановити відлік та колір тла в значення по замовчуванню.

Тепер перевірте що відбувається, коли ви виходите з програми або перезапускаєте її:

1. Примусово зупиніть додаток одним із цих двох способів:

- в Android Studio застосуйте команду *Run>Stop app* або натисніть на іконку  на панелі інструментів;
- на віртуальному пристрої натисніть кнопку *Recents* (квадратна кнопка в правому нижньому куті панелі управління). Тепер, після того як зменшилося вікно програми на екрані пристрою, закрийте його натиснувши на хрестик в правому верхньому куті вікна. Якщо ви закрили додаток таким чином, зачекайте кілька секунд, перш ніж запустити його знову, щоб система могла очистити тимчасові файли.

2. Перезапустіть додаток. При цьому додаток буде запущено із поведінкою по замовчуванню: число в text view дорівнюватиме нулю, а колір його тла буде сірим.

1.2 Дослідження програмного коду проекту

Відкрийте MainActivity та дослідіть програмний код. Зверніть увагу на таке:

- змінна *mCount* оголошена як цілочисельна. Метод *countUp*(), який є обробником події *onClick*, інкрементує цю змінну та, відповідно, оновлює значення тексту в *TextView*;
- колір (змінна *mColor*) теж є цілочисельним. Значення змінної по замовчуванню задає сірий колір, який отримується із ресурсу *colors.xml* з ім'ям *default_background*;
- метод-обробник події *onClick* під назвою *changeBackground*() встановлює колір фону головного text view рівним кольору фону натиснутої кнопки;
- обидві змінні *mCount* та *mColor* зберігаються в коді методу *onSaveInstanceState*() в об'єкті *instance state*, з якого вони читаються в коді методу *onCreate*(). При цьому в якості ключів для них використовуються приватні змінні *COUNT_KEY* та *COLOR_KEY*.

2. Збереження/читання налаштувань із файлу

2.1 Ініціалізація налаштувань

2.1.1 Додайте змінні до класу MainActivity, які будуть містити ім'я файлу налаштувань та посилання на об'єкт *SharedPreferences*:

```
private SharedPreferences mPreferences;
```

```
private String sharedPrefFile =
   "com.example.android.hellosharedprefs";
```

Ви можете називати файл налаштувань як завгодно та, як правило, його назва співпадає з назвою пакету вашого додатку.

2.1.2 В методі *onCreate()* ініціалізуйте об'єкт *SharedPreferences:*

```
mPreferences = getSharedPreferences(sharedPrefFile,
MODE_PRIVATE);
```

Ви повинні отримати наступний код:

```
public class MainActivity extends AppCompatActivity {
   private int mCount = 0;
   private TextView mShowCount;
   private int mColor;
   private SharedPreferences mPreferences;
   private String sharedPrefFile =
      "com.example.android.hellosharedprefs";
   @Override
   protected void onCreate(Bundle savedInstanceState)
{
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       mShowCount = (TextView)
findViewById(R.id.textview);
       mColor = ContextCompat.getColor(this,
          R.color.default_background);
       mPreferences = getSharedPreferences(
          sharedPrefFile, MODE_PRIVATE);
       // ...
    }
}
```

2.2 Збереження налаштувань в onPause()

Збереження налаштувань в *SharedPreferences* схоже на збереження їх в *Instance State*. Однак, збереження в *SharedPreferences* повинно відбуватися при настанні події *onPause*() життєвого циклу Activity. Крім того нам необхідно використати об'єкт типу *SharedPreferences.Editor*.

2.2.1 Додайте обробник події *onPause*() життєвого циклу MainActivity:

```
@Override
protected void onPause(){
   super.onPause();
   // ...
}
```

2.2.2 В коді методу *onPause*() отримуємо об'єкт-редактор для об'єкта *SharedPreferences*:

```
SharedPreferences.Editor preferencesEditor =
mPreferences.edit();
```

Цей редактор необхідний для запису в об'єкт *SharedPreferences*.

2.2.3 Використайте метод *putInt()* для запису змінних *mCount* та *mColor* в SharedPreferences з відповідними ключами:

```
preferencesEditor.putInt(COUNT_KEY, mCount);
preferencesEditor.putInt(COLOR_KEY, mColor);
```

2.2.4 Зберігаємо налаштування: `preferencesEditor.apply();`

Метод *apply*() виконується асинхронно, в іншому потоці по відношенню до потоку, в якому виконується інтерфейс користувача додатку.

2.2.5 Витріть повністю наявний метод *onSaveInstanceState()*. Програмний код методу *MainActivity.onPause()* повинен тепер мати вигляд:

```
@Override
protected void onPause(){
   super.onPause();
   SharedPreferences.Editor    preferencesEditor    =
mPreferences.edit();
   preferencesEditor.putInt(COUNT_KEY, mCount);
```

```
    preferencesEditor.putInt(COLOR_KEY, mColor);
    preferencesEditor.apply();
}
```

2.3 Читання налаштувань в onCreate()

Аналогічно до Instance State, налаштування *SharedPreferences* повинні читатися в методі *onCreate*(). Кожного разу, коли *onCreate*() викликається (при запуску додатку, при зміні конфігурації) *SharedPreferences* використовуватимуться для відновлення стану view.

2.3.1 Знайдіть частину коду методу *onCreate(),* яка читає налаштування з *Instance State*:

```
if (savedInstanceState != null) {
    mCount = savedInstanceState.getInt(COUNT_KEY);
    if (mCount != 0) {
        mShowCountTextView.setText(String.format("%s",
mCount));
    }
    mColor = savedInstanceState.getInt(COLOR_KEY);
    mShowCountTextView.setBackgroundColor(mColor);
}
```

2.3.2 Витріть знайдений блок коду.

2.3.3 На місце видаленого коду помістіть стрічку, яка читає значення кількості із *SharedPreferences*:

```
mCount=mPreferences.getInt(COUNT_KEY, 0);
```

При читанні із *SharedPreferences* не потрібно використовувати редактор *SharedPreferences.Editor*. Зверніть увагу на другий параметр методу *getInt*(), рівний нулю. Його значення буде повернуто методом *getInt*(), якщо налаштування по заданому ключу не було знайдено. В даному випадку це теж саме значення, що і значення змінної *mCount* по замовчуванню.

2.3.4 Оновіть значення text view:

```
mShowCountTextView.setText(String.format("%s",
mCount));
```

2.3.5 Прочитайте значення кольору із налаштувань по ключу *COLOR_KEY* та присвойте його змінній *mColor*:

```
mColor=mPreferences.getInt(COLOR_KEY, mColor);
```

Тут в якості другого аргумента методу *getInt*() використано саму змінну *mColor*, яка дорівнює нулю по замовчуванню.

2.3.6 Оновіть значення кольору фону для головного text view:

```
mShowCountTextView.setBackgroundColor(mColor);
```

2.3.7 Запустіть додаток на виконання. Понатискайте кнопку *Count* для зміни кольору фону, оновлення значення *Instance State* та налаштувань.

2.3.8 Змініть орієнтацію екрану віртуального пристрою щоб пересвідчитися, що значення кольору та відліку при цьому зберігаються.

2.3.9 Примусово зупиніть роботу додатку (наприклад, командою *Run>Stop 'app.'*).

2.3.10 Знову запустіть додаток. Ви повинні побачити, що стан додатку при цьому зберігся за рахунок читання його із налаштувань. Повний код методу *MainActivity.onCreate()* повинен бути таким:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   // ініціалізація
   mShowCountTextView = (TextView)
findViewById(R.id.count_textview);
   mColor = ContextCompat.getColor(this,
R.color.default_background);
   mPreferences = getSharedPreferences(
                        mSharedPrefFile,
MODE_PRIVATE);
   // відновлення значень із налаштувань
   mCount = mPreferences.getInt(COUNT_KEY, 0);
```

```
    mShowCountTextView.setText(String.format("%s",
mCount));
    mColor = mPreferences.getInt(COLOR_KEY, mColor);
    mShowCountTextView.setBackgroundColor(mColor);
}
```

2.4 Скидання налаштувань в обробнику reset()

Кнопка *Reset* скидає значення відліку та кольору фону в значення по замовчуванню. Оскільки налаштування містять стан activity, важливо при натисканні на цю кнопку їх очищати.

2.4.1 В коді методу *reset*() після скидання кольору та відліку отримуємо посилання на об'єкт-редактор:

```
SharedPreferences.Editor preferencesEditor =
mPreferences.edit();
```

2.4.2 Витираємо всі *SharedPreferences*:

```
preferencesEditor.clear();
```

2.4.3 Застосовуємо зміни:

```
preferencesEditor.apply();
```

Повний код методу *reset*() повинен виглядати так:

```
public void reset(View view) {
    // скидаємо відлік
    mCount = 0;
    mShowCountTextView.setText(String.format("%s",
mCount));
    // скидаємо колір
    mColor = ContextCompat.getColor(this,
R.color.default_background);
    mShowCountTextView.setBackgroundColor(mColor);
    // скидаємо налаштування
    SharedPreferences.Editor preferencesEditor =
mPreferences.edit();
    preferencesEditor.clear();
    preferencesEditor.apply();
}
```

3. Додаткове завдання

Модифікуйте додаток *HelloSharedPrefs:* додайте другу activity, яка повинна використовуватися для ручного редагування/збереження/скидання налаштувань замість автоматичного збереження їх у файл налаштувань. Додайте відповідну кнопку до програми, за допомогою якої можна буде запускати цю activity.

Контрольні запитання

1. Що таке Shared preferences та для чого вони використовуються?
2. Коли доцільно застосовувати Shared preferences?
3. Як відбувається збереження/читання даних із Shared preferences?
4. В чому полягає різниця при збереженні даних в Activity State та в SharedPreferences?

Перелік посилань

1. Конспект лекцій з курсу «Програмування для мобільних пристроїв» для здобувачів освітнього ступеня «бакалавр», спеціальності 126 «Інформаційні системи та технології», денної форми навчання. Укладачі : Готович В. А. , Михайлович Т. В. ТНТУ, 2020 р.
2. https://codelabs.developers.google.com/codelabs/android-training-shared-preferences/index.html?index=..%2F..%2Fandroid-training#0
3. https://developer.android.com/training/data-storage
4. https://developer.android.com/reference/android/content/SharedPreferences.html
5. https://developer.android.com/training/data-storage/shared-preferences
6. https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-4-saving-user-data/lesson-9-preferences-and-settings/9-0-c-data-storage/9-0-c-data-storage.html
7. https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-4-saving-user-data/lesson-9-preferences-and-settings/9-1-c-shared-preferences/9-1-c-shared-preferences.html

ЛАБОРАТОРНА РОБОТА № 6.
РОЗРОБКА ДОДАТКУ ІЗ ВИКОРИСТАННЯМ МЕНЮ ТА ІНФОРМАЦІЙНИХ ВІКОН

Мета: метою виконання лабораторної роботи є набуття студентами навичок:

- додавання пунктів до меню опцій;
- додавання піктограми для елементів у меню опцій;
- встановлення пунктів меню для відображення на панелі додатку;
- додавання обробників подій натискання для елементів меню;
- додавання діалогового вікна для сповіщення;
- додавання засобу вибору дати.

Короткі теоретичні відомості

Панель додатку *app bar* (також її називають панеллю дій *action bar*) – це виділений простір у верхній частині кожного екрана *Activity*. Коли ви створюєте *Activity* із шаблону *Basic Activity*, Android Studio включає панель додатку.

Меню параметрів на панелі додатку зазвичай пропонує вибір навігації, наприклад, до іншої *Activity* в додатку. У меню також можуть бути запропоновані варіанти, які впливають на використання самого додатка, наприклад, способи зміни налаштувань або інформації профілю, що зазвичай відбувається в окремій *Activity*.

У цій роботі ви дізнаєтесь про налаштування панелі додатку та меню параметрів, як показано на рисунку нижче.

На рисунку:

1) ***Панель додатку.*** Містить назву програми, меню опцій та кнопку опцій (або «переповнення» – overflow button).

2) ***Піктограми дій меню опцій.*** Перші два пункти меню відображаються як піктограми на панелі додатку.

3) ***Кнопка опцій.*** Три вертикальні точки. Відкриває меню, де відображаються інші опції.

4) ***Меню опцій.*** Після натискання кнопки опцій в меню з'являються інші опції.

Зазвичай пункти опцій поміщаються в меню, проте ви можете розмістити деякі елементи як піктограми – скільки поміститься в панелі додатку. Використання панелі додатку для меню опцій робить вашу додаток подібним на інші додатки Android, що дозволяє користувачам швидко зрозуміти, як ним керувати.

Ви створите програму, яка відображає діалогове вікно із запитом на вибір користувача, наприклад сповіщення, яке вимагає від користувачів натиснути кнопку ОК або Cancel. Діалоговим (*dialog*) називають вікно, яке з'являється на верхній частині дисплея або заповнює екран, перериваючи нормальну роботу *Activity*. Android пропонує готові до використання діалогові вікна, які називаються засобами вибору (*pickers*) для часу або дати. Ви можете використовувати їх для того, щоб ваші користувачі вибирали дійсний час або дату правильного формату місцевого часу та дати користувача. В даній роботі ви також створите додаток із засобом вибору дати.

Хід роботи

1. Додавання елементів до меню опцій

1.1 Вивчення коду

Завантажте проект додатку *DroidCafeInput* із GitHub за посиланням:

https://github.com/google-developer-training/android-fundamentals-apps-v2/tree/master/DroidCafeInput

Перегляньте наступні файли макета в папці *res* > *layout*:

- *activity_main.xml*: Основний макет для *MainActivity*, перший екран, який бачить користувач;

- *content_main.xml*: Макет вмісту екрана *MainActivity*, який (як ви скоро побачите) включений до activity_main.xml;
- *activity_order.xml*: Макет для *OrderActivity*, який був доданий розробником додатку.

Виконайте такі дії:

1.1.1 Відкрийте content_main.xml і натисніть вкладку *Text*, щоб побачити xml-код. Атрибут *app:layout_behavior* для *ConstraintLayout* встановлений у значення *@string/appbar_scrolling_view_behavior*, яке керує тим, як прокручується екран по відношенню до панелі додатку вгорі. Цей рядовий ресурс визначений у автоматично згенерованому файлі з назвою *values.xml*, який ви не повинні редагувати.

1.1.2 Відкрийте *activity_main.xml* та натисніть на вкладку *Text*, щоб побачити код xml основного макета, який використовує *CoordinatorLayout* із вбудованим макетом *AppBarLayout*. Теги *CoordinatorLayout* та *AppBarLayout* вимагають повністю кваліфікованих імен, які визначають *android.support.design*, що є бібліотекою підтримки дизайну (*Android Design Support*).

AppBarLayout – це вертикальний *LinearLayout*. Він використовує клас *Toolbar* у бібліотеці *Android Design Support* замість рідного *ActionBar* для реалізації панелі додатку. Панель інструментів у цьому макеті має *id* "*toolbar*", а також задається, як і *AppBarLayout*, повністю кваліфікованим іменем (*android.support.v7.widget*).

Нативний *ActionBar* поводиться по-різному, залежно від версії Android на пристрої користувача. З цієї причин для меню параметрів ви повинні використовувати бібліотеку підтримки *v7 appcompat Toolbar*.

В *activity_main.xml* також використовується оператор *include layout* для включення всього макета, визначеного в *content_main.xml*. Такий поділ визначень полегшує зміну вмісту макета, крім визначення панелі інструментів та координатора. Це найкраща практика відокремлення вашого вмісту (який може знадобитися для перекладу) від формату вашого макета.

1.1.3 Запустіть додаток. Зверніть увагу на рядок у верхній частині екрана, що відображає назву програми (*Droid Cafe*). Він також показує кнопку опцій (три вертикальні точки) праворуч. Торкніться цієї кнопки, щоб побачити меню опцій, яке в цей момент має лише один варіант: *Settings*.

1.1.4 Вивчіть файл *AndroidManifest.xml*. *MainActivity* налаштована на використання теми *NoActionBar*. Ця тема визначена у файлі *styles.xml* (відкрийте *app>res>values>styles.xml*, щоб її побачити). Тема *NoActionBar* встановлює атрибут *windowActionBar* значення *false* (немає панелі додатку), а атрибут *windowNoTitle* – *true* (немає заголовка). Ці значення встановлюються тому, що ви визначаєте панель додатків за допомогою *AppBarLayout*, а не використовуєте *ActionBar*. Використання однієї з тем *NoActionBar* не дозволяє додатку використовувати рідний клас *ActionBar* для забезпечення панелі додатку.

1.1.5 Подивіться на *MainActivity*, який розширює *AppCompatActivity* і починається з методу *onCreate()*, який встановлює подання вмісту до макета *activity_main.xml* і встановлює toolbar як панель інструментів в макеті. Потім він викликає *setSupportActionBar()* і передає йому *toolbar*, встановлюючи панель інструментів як панель додатку для *Activity*.

1.2 Додавання в меню опцій інших пунктів

Ви додасьте в меню опцій такі пункти:

- ***Order***: перейти до *OrderActivity*, щоб переглянути порядок десертів;
- ***Status***: перевірити статус замовлення;
- ***Favorites***: показати улюблені десерти;
- ***Contact***: зв’язатись із кафе. Оскільки вам не потрібен існуючий елемент *Settings*, ви його зміните на *Contact*.

Android пропонує стандартний формат xml для визначення елементів меню. Замість того, щоб створювати меню у своєму коді *Activity*, ви можете визначити меню та всі його пункти в ресурсі меню xml. Потім ви можете завантажити цей ресурс як об’єкт меню у своїй *Activity*:

1.2.1 Розгорніть *res>menu* на панелі *Project>Android* та відкрийте *menu_main.xml*. Єдиний пункт меню, що надається з шаблону – це *action_settings* (вибір параметрів), який визначається як:

```
<item
    android:id="@+id/action_settings"
    android:orderInCategory="100"
    android:title="@string/action_settings"
    app:showAsAction="never" />
```

1.2.2 Змініть такі атрибути елемента *action_settings*, щоб зробити його елементом *action_contact* (не змінюйте існуючий атрибут *android:orderInCategory*):

```
android:id="@+id/action_contact"
android:title="Contact"
app:showAsAction="never"
```

1.2.3 Перемістіть жорстко прописаний рядок "*Contact*" в ресурс *action_contact*.

1.2.4 Додайте новий елемент меню, використовуючи тег <*item*> у блоці <*menu*> та надайте цьому елементу такі атрибути:

```
android:id="@+id/action_order"
android:orderInCategory="10"
android:title="Order"
app:showAsAction="never"
```

Атрибут *android:orderInCategory* вказує порядок, у якому пункти відображаються в меню, при цьому найменше число буде означати найвищу позицію в меню. Для елемента *Contact* встановлено значення 100, щоб вказати, що він відображається внизу. Ви встановлюєте елемент *Order* на 10, що ставить його над *Contact*, а в меню залишається багато місця для інших елементів.

1.2.5 Перемістіть жорстко прописаний рядок "*Order*" в ресурс *action_order*.

1.2.6 Аналогічно, додайте ще два пункти меню з наступними атрибутами.

Status:

```
android:id="@+id/action_status"
android:orderInCategory="20"
android:title="Status"
app:showAsAction="never"
```

Favorites:

```
android:id="@+id/action_favorites"
```

```
android:orderInCategory="30"
android:title="Favorites"
app:showAsAction="never"
```

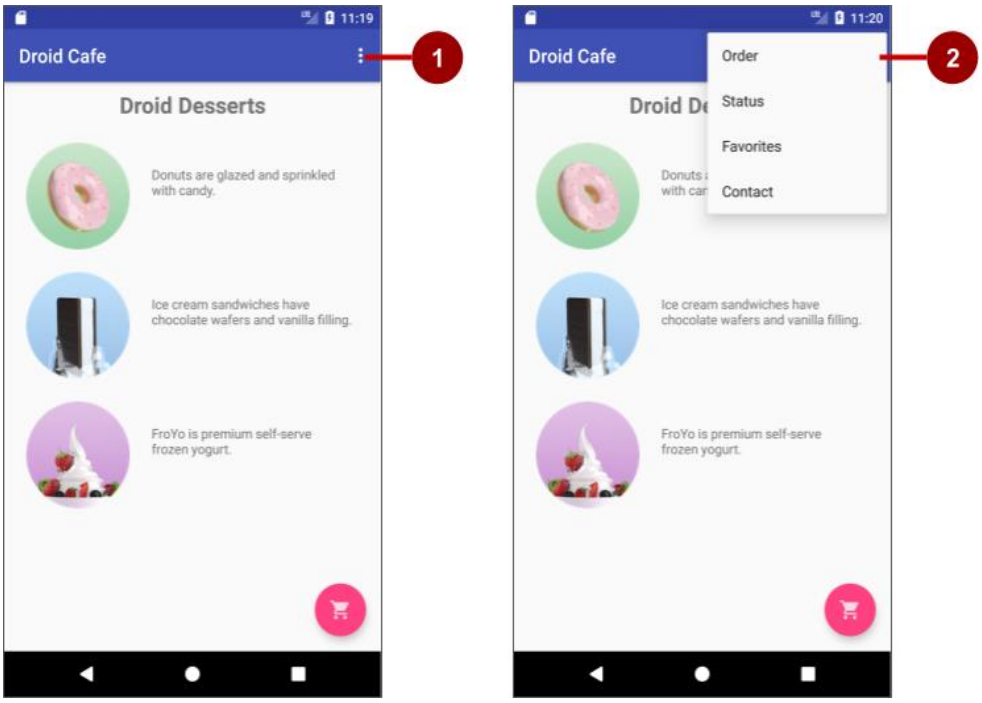
1.2.7 Перемістіть "*Status*" в ресурс *action_status*, а "*Favorites*" – в ресурс *action_favorites*.

1.2.8 Ви покажете Toast-повідомлення дії залежно від того, який пункт меню обрав користувач. Відкрийте *strings.xml* і додайте наступні назви рядків і значення для цих повідомлень:

```
<string name="action_order_message">You selected
Order.</string>
<string name="action_status_message">You selected
Status.</string>
<string name="action_favorites_message">You selected
Favorites.</string>
<string name="action_contact_message">You selected
Contact.</string>
```

1.2.9 Відкрийте *MainActivity* і змініть оператор *if* у методі *onOptionsItemSelected()*, що замінить *id* "*action_settings*" новим *id* "*action_order*": `if (id == R.id.action_order)`

1.2.10Запустіть додаток та торкніться значка опцій, вказаного в лівій частині малюнка нижче, щоб переглянути меню опцій, показаного в правій частині малюнка. Ви незабаром додасьте зворотні виклики, щоб реагувати на елементи, вибрані з цього меню.



1) Торкніться значка опцій на панелі додатків, щоб переглянути меню опцій.

2) Меню опцій випадає з панелі додатку.

Помітьте порядок елементів у меню параметрів. Ви використовували атрибут *android:orderInCategory*, щоб вказати пріоритет пунктів в меню: пункт *Order* (10), потім *Status* (20) та *Favorites* (30), і *Contact* останній (100).

2. Додавання піктограм для пунктів меню

По можливості, бажано показувати найчастіше використовувані дії за допомогою піктограм на панелі додатків, щоб користувач міг натиснути їх не заходячи в меню опцій. У цьому завданні ви додасьте піктограми для деяких пунктів меню та вкажете деякі елементи меню на панелі додатку у верхній частині екрана як піктограми.

У цьому прикладі припустимо, що дії *Order* і *Status* найчастіше використовуються. Дія *Favorites* використовується періодично, а *Contact* – найменш часто. Ви можете встановити піктограми для цих дій і вказати наступне:

- *Order* та *Status* завжди повинні відображатися на панелі додатку;
- *Favorites* має відображатися на панелі додатку, якщо він вміститься; якщо ні, він повинен відображатися в меню опцій;

- *Contact* не повинен з’являтися на панелі додатку; він повинен відображатися лише в меню опцій.

2.1 Додавання піктограм для пунктів меню

Щоб вказати піктограми для дій, потрібно спочатку додати піктограми як ресурси зображень у папку *drawable*, використовуючи ту саму процедуру, яку ви використовували у практиці щодо активних зображень. Нехай ви хочете використовувати такі піктограми (або подібні):

- *Order*;
- *Status*;
- *Favorites*;
- *Contact*: піктограма не потрібна, оскільки вона з’явиться лише в меню опцій.

Для значків Status та Favorites виконайте наступні дії:

2.1.1 Розгорніть res у панелі *Project>Android* та клацніть правою кнопкою миші (Control+лівою) папку *drawable*.

2.1.2 Виберіть *New>Image Asset*. З’явиться діалогове вікно *Configure Image Asset*.

2.1.3 У спадному меню виберіть *Action Bar and Tab Items*.

2.1.4 Переіменуйте *ic_action_name* (наприклад, на *ic_status_info* для піктограми статусу).

2.1.5 Клацніть на графічне зображення (логотип *Android* поруч із *Clipart:*). З’являється сторінка з іконками. Клацніть піктограму, яку ви хочете використати.

2.1.6 Виберіть *HOLO_DARK* зі спадного меню *Theme*. Це встановлює білий значок на темному кольорі (або чорному). Клацніть *Next*, а потім – *Finish*.

2.2 Відображення пунктів меню у вигляді піктограм на панелі додатку

Щоб відобразити елементи меню як піктограми на панелі додатку, використовуйте атрибут *app:showAsAction* у *menu_main.xml*. Наступні значення для атрибута визначають, чи має опція з’являтися на панелі додатку як значок:

- "*always*": завжди відображається на панелі додатку (якщо не вистачає місця, вона може перетинатися з іншими значками меню);
- "*ifRoom*": з’являється в панелі додатку, якщо є місце;

- "*never*": ніколи не з'являється на панелі додатку; його текст відображається в меню опцій.

Виконайте наступні дії, щоб відобразити деякі елементи меню як піктограми:

2.2.1 Відкрийте *menu_main.xml* ще раз та додайте до атрибутів *Order*, *Status* та *Favorites* такі атрибути, щоб перші два (*Order* та *Status*) завжди показувались, а пункт *Favorites* відображався лише у тому випадку, якщо для цього є місце. Якщо якийсь з атрибутів нижче, відсутній, додайте його.

Order:

```
android:icon="@drawable/ic_shopping_cart"
app:showAsAction="always"
```

Status:

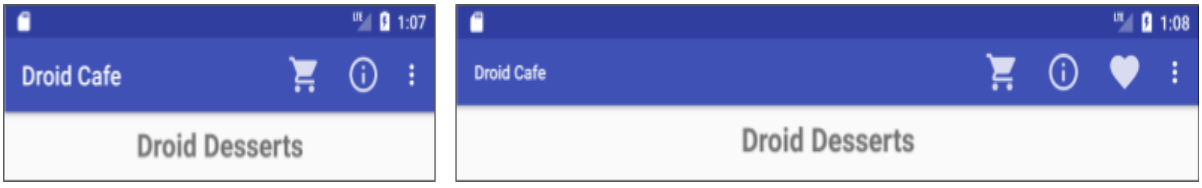
```
android:icon="@drawable/@drawable/ic_status_info"
app:showAsAction="always"
```

Favorites:

```
android:icon="@drawable/ic_favorite"
app:showAsAction="ifRoom"
```

2.2.2 Запустіть додаток. Тепер на панелі додатку ви побачите щонайменше дві піктограми: піктограму *Order* та *Status*, як показано на лівій частині малюнка нижче. Опції *Favorites* та *Contact* відображаються у меню опцій.

2.2.3 Поверніть пристрій або емулятор горизонтально. Ви побачите всі три піктограми на панелі додатку для *Order*, *Status* та *Favorites*, як показано в правій частині малюнка:



3. Реакція на вибраний пункт меню

У цьому завданні ви додасьте метод для відображення повідомлення про те, який пункт меню було натиснуто, і для цього використаєте метод *onOptionsItemSelected()*.

3.1 Створення метода відображення вибору меню

3.1.1 Відкрийте *MainActivity*.

3.1.2 Додайте наступний метод для відображення Toast-повідомлення. Ви використовуватимете його як дію для кожного вибору меню. Зазвичай ви б реалізували дію для кожного пункту меню, наприклад, запуску іншої *Activity*, як показано далі в цьому уроці.

```
public void displayToast(String message) {
   Toast.makeText(getApplicationContext(), message,
                           Toast.LENGTH_SHORT).show();
}
```

3.2 Використання обробника подій onOptionsItemSelected

Метод *onOptionsItemSelected()* обробляє виділення з меню опцій. Ви додасьте блок *switch case*, щоб визначити, який пункт меню було обрано та яку дію потрібно виконати.

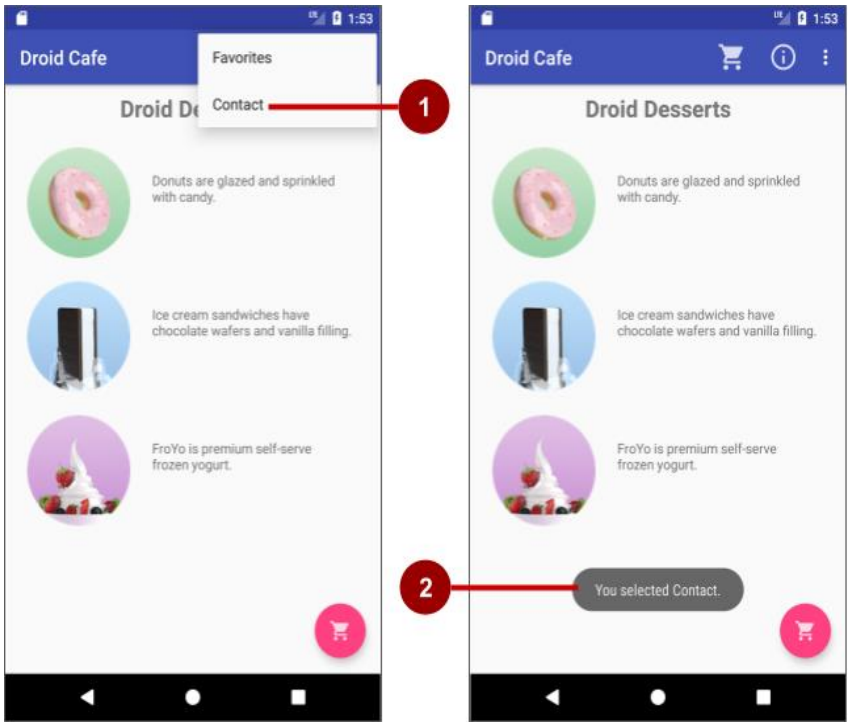
3.2.1 Знайдіть метод *onOptionsItemSelected()*, наданий шаблоном. Метод визначає, чи було натиснуто певний пункт меню, використовуючи ідентифікатор *id* елемента меню. У наведеному нижче прикладі ідентифікатором є *action_order*:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_order) {
            return true;
        }
        return super.onOptionsItemSelected(item);
}
```

3.2.2 Замініть оператор присвоєння ідентифікатора *id* та оператор *if* на наступний блок *switch case*, який встановлює відповідне значення *message* на основі *id* елемента меню:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_order:

displayToast(getString(R.string.action_order_message)
);
                return true;
            case R.id.action_status:

displayToast(getString(R.string.action_status_message
));
                return true;
            case R.id.action_favorites:

displayToast(getString(R.string.action_favorites_mess
age));
                return true;
            case R.id.action_contact:

displayToast(getString(R.string.action_contact_messag
e));
                return true;
            default:
             // Do nothing
        }
        return super.onOptionsItemSelected(item);
}
```

3.2.3 Запустіть програму. Тепер ви побачите інше Toast-повідомлення на екрані, як показано праворуч на малюнку нижче, залежно від того, який пункт меню ви вибрали.



На рисунку:

1) вибір пункту *Contact* в меню параметрів;
2) Toast-повідомлення, яке з'являється.

3.3 Запуск Activity з пункту меню

3.3.1 Зазвичай потрібно реалізувати дію для кожного пункту меню, наприклад, запустивши іншу *Activity*. Посилаючись на фрагмент з попереднього завдання, змініть код для випадку *action_order* на код нижче. Він запустить *OrderActivity*:

```
switch (item.getItemId()) {
    case R.id.action_order:
        Intent intent = new Intent(MainActivity.this,
OrderActivity.class);
        intent.putExtra(EXTRA_MESSAGE,
mOrderMessage);
        startActivity(intent);
        return true;
    // ... code for other cases
}
```

Запустіть додаток. Якщо натиснути значок кошика для покупок на панелі додатків (елемент *Order*), ви перейдете безпосередньо до екрана *OrderActivity*.

4. Використання діалогового вікна для запиту вибору користувача

Ви можете показувати діалогове вікно, щоб запитувати вибір користувача. Наприклад, діалогове вікно сповіщення може вимагати від користувача натиснути *Continue* після прочитання повідомлення, або надати користувачеві можливість погодитися з дією, натиснувши стверджуючу кнопку (наприклад, *OK*, *Agree* чи *Accept*), або не погодитися, натиснувши відхиляючу кнопку (наприклад, *Cancel*, *Decline* чи *Disagree*). Використовуйте підклас *AlertDialog* класу *Dialog*, щоб показати стандартне діалогове вікно для попередження.

У цьому завданні ви використаєте кнопку для запуску стандартного діалогового вікна сповіщення. У реальному додатку ви можете запустити діалогове вікно попередження на основі якоїсь умови або на основі того, що користувач щось натискає.

4.1 Створення нового додатку для відображення діалогового вікна попередження

У цьому завданні ви створюєте попередження за допомогою кнопок *OK* та *Cancel*. Сповіщення спрацьовує, коли користувач натискає кнопку.

4.1.1 Створіть новий проект під назвою *Dialog For Alert* на основі шаблону *Empty Activity*.

4.1.2 Відкрийте файл макета *activity_main.xml*, щоб показати редактор макета.

4.1.3 Відредагуйте елемент TextView, щоб відобразити "*Hello World! Tap to test the alert:*" замість "*Hello World!*"

4.1.4 Додайте кнопку під TextView. Можна також прив’язати кнопку внизу TextView, виставивши поля *8dp*.

4.1.5 Встановіть текст кнопки на "*Alert*".

4.1.6 Перейдіть на вкладку *Text* і перемістіть текстові стрічки для *TextView* та *Button* до стрічкових ресурсів.

4.1.7 Додайте атрибут *android:onClick* до кнопки, щоб викликати обробник кліків *onClickShowAlert()*. Після додавання, обробник кліків підкреслений червоним кольором, оскільки він ще не створений.

```
android:onClick="onClickShowAlert"
```

4.2 Додавання діалогу попередження до основної Activity

Шаблон дизайну побудови (*builder*) полегшує створення об'єкта з класу, який має багато необхідних та необов'язкових атрибутів і потребував би налаштування безлічі параметрів. Без цього шаблону вам довелося би створювати конструктори для комбінацій необхідних та необов'язкових атрибутів; а з цією схемою код легше читати та підтримувати.

Клас побудови (*builder*), як правило, є статичним класом для того класу, який він будує. Використовуйте *AlertDialog.Builder* для створення стандартного діалогового діалогового вікна, із методами: *setTitle()* для встановлення його назви, *setMessage()* для встановлення його повідомлення, *setPositiveButton()* і *setNegativeButton()* для встановлення його кнопки.

Щоб реалізувати попередження вам потрібно створити об'єкт *AlertDialog.Builder*. Ви додасьте обробник кліків *onClickShowAlert()* для кнопки *Alert*. Це означає, що діалогове вікно буде створено лише тоді, коли користувач натисне кнопку *Alert*. Хоча цей шаблон кодування є логічним для використання кнопки для тестування оповіщення, для інших додатків ви можете створити діалогове вікно методом *onCreate()*, щоб завжди був доступний інший код для його запуску.

4.2.1 Відкрийте *MainActivity* і додайте початок методу *onClickShowAlert()*:

```
public void onClickShowAlert(View view) {
    AlertDialog.Builder myAlertBuilder = new

AlertDialog.Builder(MainActivity.this);
    // Set the dialog title and message.
}
```

*Я*кщо *AlertDialog.Builder* не розпізнається під час його введення, натисніть значок червоної лампочки та виберіть версію бібліотеки підтримки (*android.support.v7.app.AlertDialog*) для імпорту в свою Activity.

4.2.2 Додайте код, щоб встановити заголовок та повідомлення діалогового вікна сповіщення в *onClickShowAlert()* після коментаря:

```
// Set the dialog title and message.
myAlertBuilder.setTitle("Alert");
myAlertBuilder.setMessage("Click OK to continue, or
Cancel to stop:");
// Add the dialog buttons.
```

4.2.3 Перемістіть використані вище стрічки до ресурсів *alert_title* та *alert_message*.

4.2.4 Додайте кнопки *OK* та *Cancel* до попередження методами *setPositiveButton()* та *setNegativeButton()*:

```
// Add the dialog buttons.
myAlertBuilder.setPositiveButton("OK", new

DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int
which) {
        // User clicked OK button.
        Toast.makeText(getApplicationContext(),
"Pressed OK",
                       Toast.LENGTH_SHORT).show();
    }
});
myAlertBuilder.setNegativeButton("Cancel", new

DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int
which) {
        // User cancelled the dialog.
        Toast.makeText(getApplicationContext(),
"Pressed Cancel",
                       Toast.LENGTH_SHORT).show();
    }
```

```
});
// Create and show the AlertDialog.
```

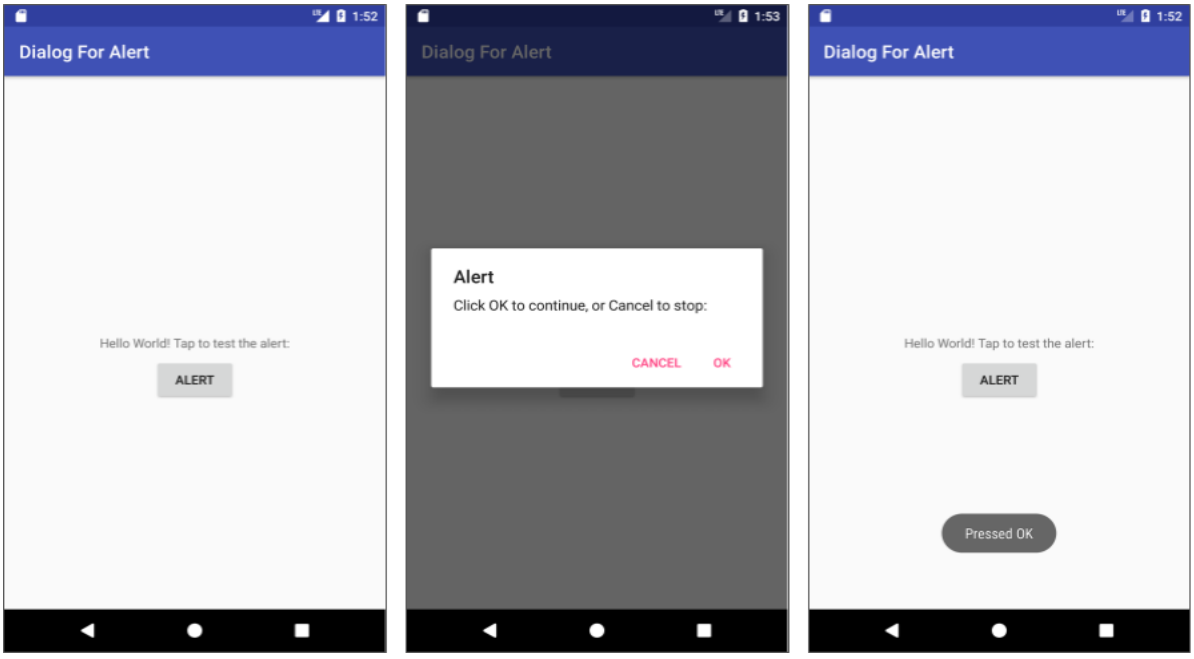
Після того, як користувач натисне кнопку *OK* або *Cancel* у сповіщенні, ви можете використати це у своєму коді. У цьому прикладі ви показуєте Toast-повідомлення.

4.2.5 Перемістіть стрічки для *OK* і *Cancel* в ресурси як *ok_button* і *cancel_button*, також перемістіть стрічки для Toast-повідомлень.

4.2.6 Наприкінці методу *onClickShowAlert()* додайте *show()*, який створює, а потім відображає діалогове вікно попередження:

```
// Create and show the AlertDialog.
myAlertBuilder.show();
```

4.2.7Запустіть додаток. Натисніть кнопку *Alert*, показану на малюнку нижче зліва, щоб побачити діалогове вікно сповіщення, показане в центрі малюнка. У діалоговому вікні відображаються кнопки *OK* та *Cancel*, і з'являється Toast-повідомлення у відповідності з вашим вибором (на малюнку справа).



5. Використання засобу вибору picker

У цьому завданні ви створите новий проект і додасьте інструмент вибору дати (*date picker*). Ви також дізнаєтесь, як використовувати *Fragment*, який є поведінкою або частиною інтерфейсу користувача в *Activity*. Це як міні-*Activity* зі своїм власним життєвим циклом, що використовується для побудови вибору. Вся робота робиться за вас.

Однією з переваг використання *Fragment* для *picker* є те, що ви можете виділити розділи коду для управління датою та часом для різних культур, які відображають дату та час різними способами. Найкращою практикою показу вибору є використання екземпляра *DialogFragment*, який є підкласом *Fragment*. *DialogFragment* відображає плаваюче діалогове вікно вгорі *Activity*. У цьому завданні ви додасьте фрагмент діалогового вікна вибору та використовуватимете *DialogFragment* для управління життєвим циклом діалогу.

5.1 Створення нової програми для показу вибору дати

5.1.1 Створіть новий проект під назвою *Picker For Date* на основі шаблону *Empty Activity*.

5.1.2 Відкрийте файл макета *activity_main.xml*, щоб показати редактор макета.

5.1.3 Відредагуйте текст "*Hello World!*" елемента *TextView* до значення "*Hello World! Choose a date:*".

5.1.4 Додайте кнопку під *TextView*. Можна також прив’язати кнопку в нижній частині *TextView*, виставивши поля *8dp*.

5.1.5 Встановіть значення тексту кнопки в "*Date*".

5.1.6 Перейдіть на вкладку *Text* і перемістіть стрічки для *TextView* та *Button* в ресурси.

5.1.7 Додайте атрибут *android:onClick* до кнопки для виклику обробника кліків *showDatePicker()*. Після введення, обробник кліків підкреслений червоним кольором, оскільки він ще не створений. `android:onClick="showDatePicker"`

5.2 Створення нового фрагменту для вибору дати

5.2.1 Розгорніть *app>java>com.example.android.pickerfordate* та виберіть *MainActivity*.

5.2.2 Виберіть *File>New>Fragment>Fragment (Blank)* та назвіть фрагмент *DatePickerFragment*. Зніміть усі три прапорці, щоб не створювати

xml-макет, фабричні методи фрагментів або зворотні виклики інтерфейсу. Вам не потрібно створювати макет для стандартного *picker*. Клацніть *Finish*.

5.2.3 Відкрийте *DatePickerFragment* та відредагуйте визначення його класу, щоб розширити *DialogFragment* та реалізувати *DatePickerDialog.OnDateSetListener* для створення стандартного *date picker* зі «слухачем» вибору.

```
public class DatePickerFragment extends DialogFragment
                        implements
DatePickerDialog.OnDateSetListener {
```

Коли ви входите в *DialogFragment* та *DatePickerDialog.OnDateSetListener*, Android Studio автоматично додає кілька операторів import вгорі, включно з:

```
import android.app.DatePickerDialog;
import android.support.v4.app.DialogFragment;
```

Окрім того, значок червоної лампочки з'являється у лівому полі через декілька секунд.

5.2.4 Клацніть на значку червоної лампочки та виберіть зі спливаючого меню пункт *Implement methods*. З'явиться діалогове вікно з уже вибраним *onDateSet()* та вибраним параметром *Insert @Override*. Клацніть *OK*, щоб створити порожній метод *onDateSet()*. Цей метод буде викликаний при виборі дати користувачем.

Після додавання порожнього методу *onDateSet()*, Android Studio автоматично додає наступну інструкцію у блок імпорту вгорі: `import android.widget.DatePicker;`

Параметрами *onDateSet()* зараз є *int i, int i1* та *int i2*. Змініть їх назви на більш зручні для читання: `public void onDateSet(DatePicker datePicker, int year, int month, int day)`

5.2.5 Видаліть порожній публічний конструктор (*public constructor*) *DatePickerFragment()*.

5.2.6 Замініть весь метод *onCreateView()* на *onCreateDialog()*, який повертає *Dialog*, та анотує *onCreateDialog()* як *@NonNull*, щоб вказати, що діалогове значення, яке повертається, не може бути нульовим. Android

Studio відображає червону лампочку поруч із методом, оскільки вона ще нічого не повертає.

```
@NonNull
@Override
public Dialog onCreateDialog(Bundle
savedInstanceState) {
}
```

5.2.7 Додайте наступний код у *onCreateDialog()*, щоб ініціалізувати *year*, *month* та *day* з *Calendar*, а також повернути діалогове вікно та ці значення до Activity. Коли ви додасьте *Calendar.getInstance()*, додайте вгоді також *import java.util.Calendar*.

```
// Use the current date as the default date in the
picker.
final Calendar c = Calendar.getInstance();
int year = c.get(Calendar.YEAR);
int month = c.get(Calendar.MONTH);
int day = c.get(Calendar.DAY_OF_MONTH);
// Create a new instance of DatePickerDialog and return
it.
return new DatePickerDialog(getActivity(), this, year,
month, day);
```

5.3 Зміна основної Activity

Незважаючи на те, що значна частина коду в *MainActivity.java* залишається однаковою, вам потрібно додати метод, який створює екземпляр *FragmentManager* для управління фрагментом і показом засобу вибору дати.

5.3.1 Відкрийте *MainActivity*.

5.3.2 Додайте обробник *showDatePickerDialog()* для кнопки *Date*. Він створює екземпляр *FragmentManager* за допомогою *getSupportFragmentManager()* для автоматичного управління фрагментом та показу *picker*.

```
public void showDatePicker(View view) {
        DialogFragment newFragment = new
DatePickerFragment();
newFragment.show(getSupportFragmentManager(),"datePic
ker");
}
```

5.3.3 Перемістіть стрічку "*datePicker*" до ресурсу *datepicker*.

5.3.4 Запустіть додаток. Після натиснення кнопки *Date* повинен з'явитись *picker*.

5.4 Використання обраної дати

На цьому кроці ви передасьте дату у *MainActivity.java* і перетворите її в рядок, який ви можете показати у Toast-повідомленні.

5.4.1 Відкрийте *MainActivity* і додайте порожній метод *processDatePickerResult()*, який приймає як аргументи *year*, *month* та *day*: `public void processDatePickerResult(int year, int month, int day) {}`

5.4.2 Додайте наступний код до методу *processDatePickerResult()*, щоб перетворити *year*, *month* та *day* в окремі рядки та об'єднати три рядки з косою рискою для формату дати США:

```
String month_string=Integer.toString(month+1);
String day_string=Integer.toString(day);
String year_string=Integer.toString(year);
String dateMessage=(month_string + "/" + day_string +
"/" + year_string);
```

Ціле число місяця, що повертає нам *date picker*, починає рахуватись з 0 (січень), тому вам потрібно додати 1, щоб відобразити місяці в звичній нумерації.

5.4.3 Додайте наступне після коду вище, щоб відобразити Toast-повідомлення: `Toast.makeText(this, "Date: "+dateMessage, Toast.LENGTH_SHORT).show();`

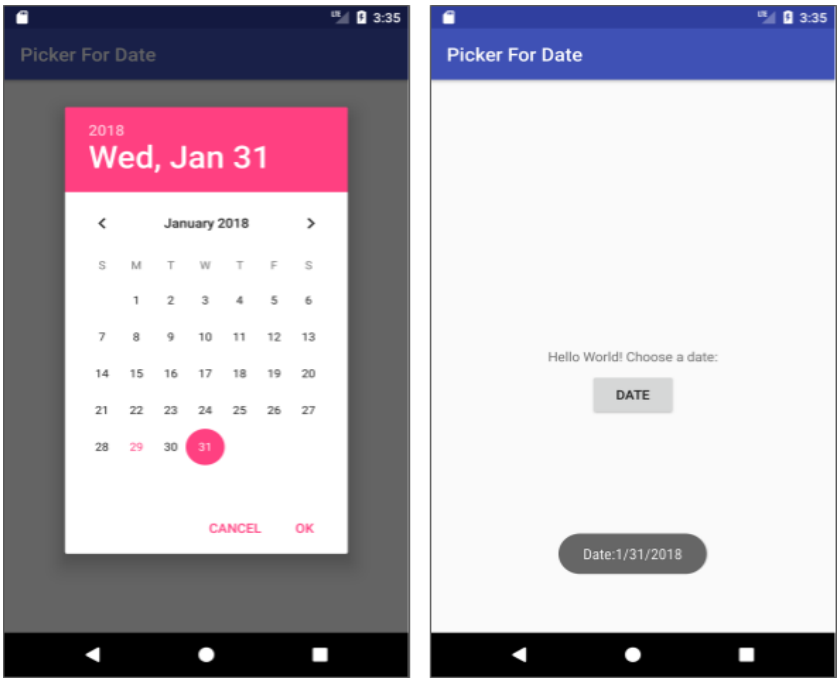
5.4.4 Перемістіть жорстко прописану стрічку "*Date:*" в ресурс з ім'ям *date*.

5.4.5 Відкрийте *DatePickerFragment* і додайте наступне до методу *onDateSet()*, щоб викликати *processDatePickerResult()* у *MainActivity* та передати йому *year*, *month* та *day*:

```
@Override
public void onDateSet(DatePicker datePicker,
                                    int year, int month,
int day) {
    MainActivity activity = (MainActivity)
getActivity();
    activity.processDatePickerResult(year, month,
day);
}
```

При використанні *getActivity()* у *Fragment* повертається *Activity*, з якою в даний час він асоціюється. Це вам потрібно, оскільки ви не можете викликати метод в *MainActivity* без його контексту (вам доведеться використовувати замість цього *intent*). *Activity* успадковує контекст, тому ви можете використовувати його для виклику методу (як у *Activity.processDatePickerResult*).

5.4.6 Запустіть додаток. Після вибору, дата з'являється у Toast-повідомленні.



Контрольні запитання

1. За допомогою якого шаблону потрібно створювати програму, щоб автоматично налаштувати панель програми, меню параметрів і плаваючу кнопку дії?
2. Який метод потрібно використати, щоб визначити, який пункт меню було вибрано?
3. Яке значення атрибута *app:showAsAction* дозволяє піктограмі опції з'являється на панелі додатку, якщо є місце?
4. Який клас слід використовувати для побудови стандартного діалогового вікна попередження?
5. Що таке *Fragment*? Які переваги його використання?
6. Що таке діалогове вікно сповіщення? Які особливості його застосування?
7. Для чого потрібна кнопка опцій (*overflow button*)?

Перелік посилань

1. Конспект лекцій з курсу «Програмування для мобільних пристроїв» для здобувачів освітнього ступеня «бакалавр», спеціальності 126 «Інформаційні системи та технології», денної форми навчання. Укладачі: Готович В.А., Михайлович Т. В. ТНТУ, 2020 р.
2. https://codelabs.developers.google.com/codelabs/android-training-menus-and-pickers/index.html?index=..%2F..%2Fandroid-training#0
3. https://developer.android.com/topic/libraries/support-library/packages#v7-appcompat
4. https://developer.android.com/guide/topics/ui/dialogs.html
5. http://developer.android.com/guide/topics/ui/controls/pickers.html
6. http://developer.android.com/guide/components/fragments.html
7. https://material.io/design/layout/responsive-layout-grid.html
8. https://uk.wikipedia.org/wiki/Будівник_(шаблон_проєктування)
9. https://codelabs.developers.google.com/codelabs/android-training-menus-and-pickers/index.html

ДЛЯ НОТАТКІВ

ДЛЯ НОТАТКІВ

Навчально-методична література

Готович В. А., Михайлович Т. В.

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт

з дисципліни

«Програмування для мобільних пристроїв»

для студентів денної форми навчання
спеціальності
126 «Інформаційні системи та технології»

Комп'ютерне макетування та верстка *А. А. Флейтути.*

Формат 60х90/16. Обл. вид. арк. 3,01. Тираж 10 прим. Зам. № 3343.

Тернопільський національний технічний університет імені Івана Пулюя.
46001, м. Тернопіль, вул. Руська, 56.
Свідоцтво суб'єкта видавничої справи ДК № 4226 від 08.12.11.