Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЗВІТ

з лабораторної роботи №2
з дисципліни «Сучасні мобільні операційні системи»

Тема: «Основи верстки»

Варіант №4

Виконав:
студент 1 курсу, групи ІМ-51мн
Ковальов Олександр

Перевірив:
асистент, Нестерук Андрій Олександрович

Дата здачі: 13.02.2026

КИЇВ – 2026

Мета роботи. Вивчити основи верстки. Навчитися керувати інтерфейсом мобільного пристрою при розробці програми.

Завдання. Розробити мобільний додаток, що складається з чотирьох Activity. Після запуску програми користувач повинен потрапляти на екран з `Activity1`. На цьому екрані має бути представлено меню, що складається з чотирьох кнопок. Висота кнопок повинна складати 20% від висоти екрана. Відстань між кнопками – 2%. Перша і остання кнопка повинні бути на рівній відстані від країв екрану. Ширина кнопок – 75%, вирівнювання посередині. Після натискання на першу кнопку користувач повинен переходити до `Activity2`, його зовнішній вигляд представлений на *рис. 1*. Верстка повинна здійснюватися з використанням `LinearLayout`, ширина кнопок повинна задаватися в відсотках від ширини екрану.

Рис. 1 – Зовнішній вигляд екрану для першого завдання.

Після натискання на другу кнопку в `Activity1` користувач повинен переходити до `Activity3`, його зовнішній вигляд представлений на *рис. 2*. Верстка повинна здійснюватися з використанням `RelativeLayout` (не використовувати `LinearLayout`).

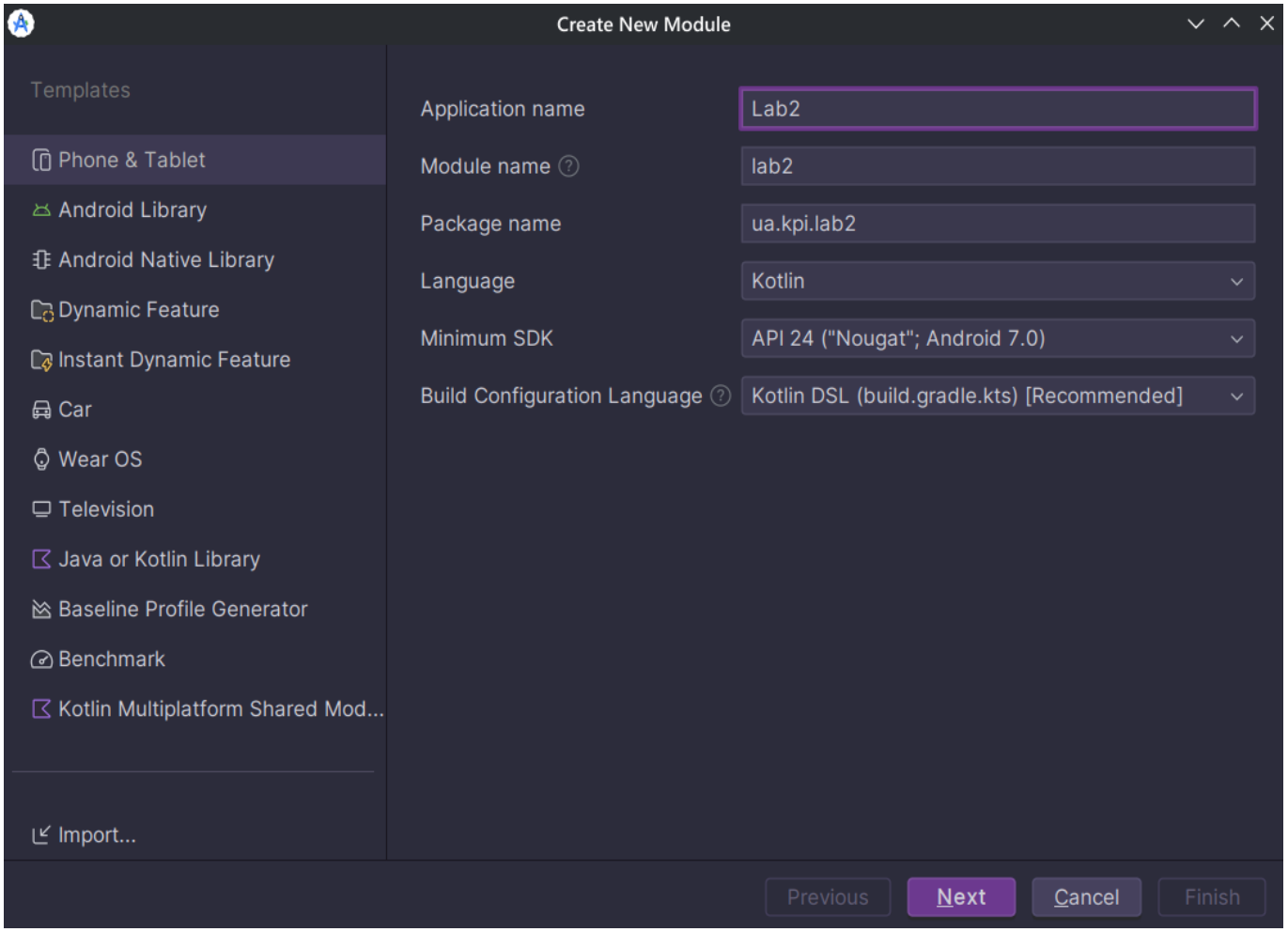
Рис. 2 – Результат виконання першого етапу завдання.

Третя кнопка в `Activity1` повинна створювати `Activity4`. Зовнішній вигляд `Activity4` наведений на *рис. 3*. Кнопка повинна бути вирівняна по центру екрана. Колір обведення кнопки `#505050`. Товщина обведення відповідно до місяця вашого народження (від 1 до 12). Радіус заокруглення – `24dp`. Колір фону екрану – `#FFFFFF`. При натисканні на кнопку її колір повинен змінюватися на світло-зелений. Натискання на четверту кнопку в `Activity1` повинно призводити до закриття програми.

Рис. 3 – Інтерфейс програми на етапі `Activity3`.

Хід роботи.

Створено модуль лабораторної роботи:



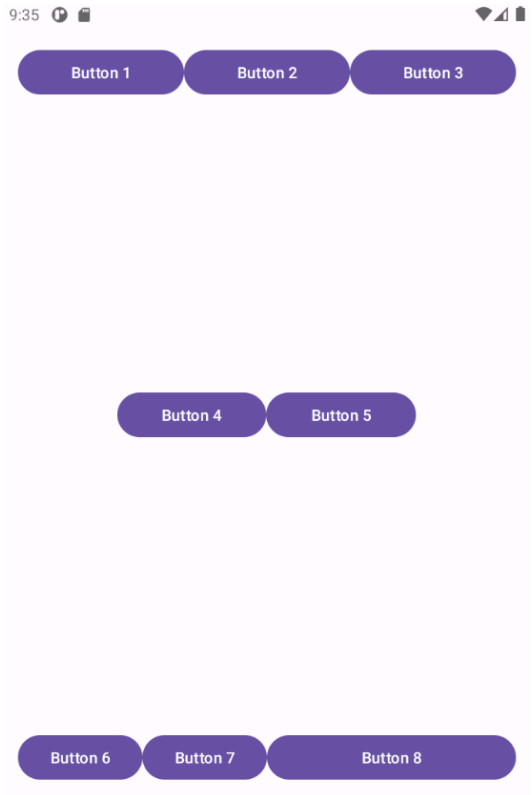
Для виконання першого завдання, треба використовувати елемент `LinearLayout`. Головний – має вагу 100 та вертикальну орієнтацію. Всередині нього – вкладені `LinearLayout` та елементи `View`, які слугують заглушками. Так як висота кнопок повинна бути 20% від висоти екрану, а відстань між кнопками – 2%, треба заздалегідь розрахувати вагу елементів.

- **Кнопки:** 4 по 20% = 80%.
- **Відступи між ними:** 3 по 2% = 6%.
- **Разом зайнято:** 80% + 6% = 86%.
- **Залишилося вільного місця:** 100% - 86% = 14%.
- **Відступи зверху і знизу:** Щоб було симетрично (перша і остання на рівній відстані), треба поділити залишок навпіл: 14% / 2 = 7%.

Отже, схема висоти: 7% (View) - 20% (`btn1`) - 2% (View) - 20% (`btn2`) - 2% (View) - 20% (`btn3`) - 2% (View) - 20% (`btn4`) - 7% (View).

Вкладені `LinearLayout` потрібні, щоб обмежити ширину кнопок на 75%.

Щодо другої активності використовується той самий принцип, що і в головному меню.

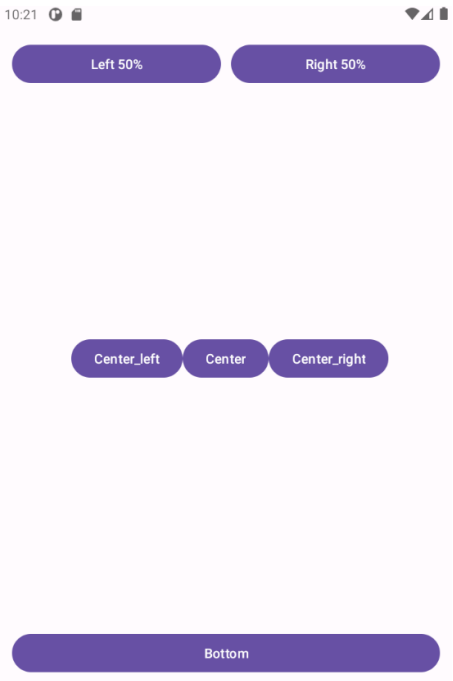


Для реалізації третього екрану було створено нову активність `RelativeActivity` та відповідний файл розмітки XML. Згідно із завданням, верстка інтерфейсу здійснювалася виключно за допомогою контейнера `RelativeLayout`, що вимагало використання відносного позиціонування елементів замість вагових коефіцієнтів.

Для розділення верхньої частини екрану на дві рівні зони (по 50% ширини для кнопок «Left» та «Right») було застосовано техніку використання опорного елемента. У центрі верхньої частини контейнера було розміщено допоміжний невидимий об’єкт `View` з нульовими розмірами, вирівняний по горизонтальному центру батьківського елемента. Ліва кнопка була позиціонована за допомогою прив’язки її лівого краю до початку екрану, а правого – до опорного центру. Права кнопка, відповідно, була прив’язана лівим краєм до центру, а правим – до кінця екрану, що забезпечило симетричний розподіл простору без використання `LinearLayout`.

Центральна група елементів була сформована відносно кнопки «Center», для якої було встановлено атрибут центрування в межах батьківського контейнера (`centerInParent`). Кнопки «`Center_left`» та «`Center_right`» були розміщені відповідно ліворуч та праворуч від центрального елемента за допомогою атрибутів відносного розміщення (`layout_toStartOf` та `layout_toEndOf`) із вирівнюванням по верхньому краю центральної кнопки.

Нижній елемент інтерфейсу – кнопку «Bottom» – було зафіксовано у нижній частині екрану за допомогою атрибута прив’язки до нижнього краю батьківського контейнера (`layout_alignParentBottom`) та розтягнуто на всю доступну ширину.



Для виконання четвертої частини завдання було реалізовано `FourthActivity`, перехід до якої здійснюється з головного меню через обробник подій кнопки за допомогою класу `Intent`. Верстка екрана базується на `ConstraintLayout` із встановленим білим фоном, де центральним елементом є кастомізована кнопка. Її візуальний стиль визначено у спеціальному XML-ресурсі `fourth_btn.xml` з використанням тегу `selector`, що дозволяє змінювати вигляд компонента залежно від його стану. У звичайному режимі кнопка має білий фон, заокруглені кути радіусом `24dp` та сіре обведення (`#505050`) товщиною `4dp`, що відповідає місяцю народження (квітень), а при натисканні колір заливки змінюється на світло-зелений (`#90EE90`). Для коректного відображення власного фону замість стандартного кольору теми Material Design властивість `app:backgroundTint` встановлено у значення `@null`. Окрім тексту чорного кольору, до кнопки додано векторну іконку Android, яку за допомогою атрибутів `app:iconTint` та `app:iconGravity` пофарбовано у зелений колір та розміщено ліворуч від напису з відповідним відступом.

Висновок. В ході виконання лабораторної роботи було розроблено багатоекранний мобільний додаток та закріплено навички роботи з основними типами розмітки в Android. Зокрема, було практично досліджено можливості `LinearLayout` для створення адаптивних інтерфейсів із використанням вагових коефіцієнтів, що дозволило реалізувати складне пропорційне розміщення елементів головного меню та вкладених структур. Також було опановано принципи відносного позиціонування компонентів за допомогою `RelativeLayout`, завдяки чому вдалося відтворити задану схему розташування кнопок. Окрему увагу було приділено кастомізації елементів інтерфейсу: створено спеціальні XML-ресурси (`Selector`) для керування візуальним станом кнопки, налаштовано її геометричну форму, параметри обведення згідно з варіантом та реакцію на натискання. Реалізація переходів між чотирма активностями дозволила закріпити знання про використання класу `Intent` для організації навігації всередині додатку.

Контрольні запитання.

1. **Що таке `Layout`?**

   `Layout` (макет) – це контейнер, який визначає візуальну структуру інтерфейсу користувача, наприклад, для активності або віджета додатку. Технічно він є спадкоємцем класу `ViewGroup` і відповідає за впорядкування, розміщення та відображення дочірніх елементів, якими можуть бути як прості віджети (кнопки, текстові поля), так і інші вкладені макети.

2. **Які існують види `Layout`?**

   Існує декілька основних видів макетів, кожен з яких реалізує різну логіку розташування елементів. Найпоширенішими є `LinearLayout`, який вибудовує компоненти в один горизонтальний або вертикальний ряд, `RelativeLayout`, що дозволяє розміщувати елементи відносно один одного або батьківського контейнера, та `ConstraintLayout`, який забезпечує створення гнучких адаптивних інтерфейсів за допомогою системи прив'язок. Також існують `FrameLayout` для накладання об'єктів, `TableLayout` для табличного розміщення та `GridLayout` для створення сіток.

3. **Які параметри мають View-елементи?**

   Всі `View`-елементи мають набір базових параметрів (атрибутів), що контролюють їхній вигляд та поведінку. Ключовими є ідентифікатор (`id`) для доступу з коду, а також ширина (`layout_width`) та висота (`layout_height`), які зазвичай приймають значення `wrap_content` або `match_parent`. До інших важливих параметрів належать зовнішні відступи (`margin`), внутрішні відступи (`padding`), вирівнювання вмісту (`gravity`), колір фону (`background`) та видимість (`visibility`).

4. **Як створити Layout-файл для роботи в горизонтальній орієнтації екрану мобільного пристрою? У яких випадках це необхідно?**

   Для підтримки горизонтальної орієнтації необхідно створити в папці ресурсів `res` нову директорію з назвою `layout-land` і скопіювати туди XML-файл розмітки з такою ж назвою, як і основний файл. Це необхідно у випадках, коли просте розтягування вертикального макету призводить до неестетичного вигляду або нераціонального використання екранного простору, наприклад, коли потрібно розмістити елементи у дві колонки замість однієї або коли вертикального простору недостатньо для відображення всього контенту.

5. **Для чого потрібні методи `setContentView`, `findViewById`?**

   Метод `setContentView` викликається у життєвому циклі активності (зазви-

чай в `onCreate`) для прив'язки XML-файлу розмітки до вікна програми, ініціюючи процес «інфлейтингу» (перетворення XML у об'єкти). Метод `findViewById` використовується для пошуку конкретного дочірнього елемента в завантаженій ієрархії `View` за його унікальним ідентифікатором (`id`), що дозволяє отримати посилання на цей об'єкт для подальшої програмної взаємодії з ним.

6. **Які існують способи обробки подій в Activity?**

   Обробку подій користувача, таких як натискання (click), можна реалізувати декількома способами. Найсучаснішим підходом у Kotlin є використання лямбда-виразів, які передаються у метод `setOnClickListener`. Також можна використовувати анонімні внутрішні класи, реалізовувати інтерфейс `View.OnClickListener` безпосередньо класом активності або визначати метод обробки в XML-атрибуті `android:onClick` (хоча останній спосіб вважається застарілим).

Лістинг.

MainActivity.kt

```
package ua.kpi.lab2

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

        // Button 1
        findViewById<Button>(R.id.btnToLinear).setOnClickListener {
            startActivity(Intent(this, SecondActivity::class.java))
        }

        // Button 2
        findViewById<Button>(R.id.btnToRelative).setOnClickListener {
            startActivity(Intent(this, ThirdActivity::class.java))
        }

        // Button 3
        findViewById<Button>(R.id.btnToStyle).setOnClickListener {
            startActivity(Intent(this, FourthActivity::class.java))
        }

        // Button 4
        findViewById<Button>(R.id.btnExit).setOnClickListener {
            finishAffinity()
        }
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:weightSum="100">

<View
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="7" />

<LinearLayout
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="20"
android:gravity="center"
android:orientation="horizontal"
android:weightSum="100">
```

```
20
21  <Button
22  android:id="@+id/btnToLinear"
23  android:layout_width="0dp"
24  android:layout_height="match_parent"
25  android:layout_weight="75"
26  android:text="To Activity 2 (Linear)" />
27  </LinearLayout>
28
29  <View
30  android:layout_width="match_parent"
31  android:layout_height="0dp"
32  android:layout_weight="2" />
33
34  <LinearLayout
35  android:layout_width="match_parent"
36  android:layout_height="0dp"
37  android:layout_weight="20"
38  android:gravity="center"
39  android:orientation="horizontal"
40  android:weightSum="100">
41
42  <Button
43  android:id="@+id/btnToRelative"
44  android:layout_width="0dp"
45  android:layout_height="match_parent"
46  android:layout_weight="75"
47  android:text="To Activity 3 (Relative)" />
48  </LinearLayout>
49
50  <View
51  android:layout_width="match_parent"
52  android:layout_height="0dp"
53  android:layout_weight="2" />
54
55  <LinearLayout
56  android:layout_width="match_parent"
57  android:layout_height="0dp"
58  android:layout_weight="20"
59  android:gravity="center"
60  android:orientation="horizontal"
61  android:weightSum="100">
62
63  <Button
64  android:id="@+id/btnToStyle"
65  android:layout_width="0dp"
66  android:layout_height="match_parent"
67  android:layout_weight="75"
68  android:text="To Styled Activity" />
69  </LinearLayout>
70
71  <View
72  android:layout_width="match_parent"
73  android:layout_height="0dp"
74  android:layout_weight="2" />
75
76  <LinearLayout
77  android:layout_width="match_parent"
78  android:layout_height="0dp"
79  android:layout_weight="20"
80  android:gravity="center"
81  android:orientation="horizontal"
82  android:weightSum="100">
```

```
<Button
android:id="@+id/btnExit"
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="75"
android:text="Exit" />
</LinearLayout>

<View
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="7" />

</LinearLayout>
```

SecondActivity.kt

```
package ua.kpi.lab2

import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class SecondActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_second)

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(android.R.id.content)) { v, insets
        ↪  ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
    }
}
```

activity_second.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="16dp">

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal"
android:weightSum="3">

<Button
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_weight="1"
```

```
android:text="Button 1" />

<Button
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="Button 2" />

<Button
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="Button 3" />
</LinearLayout>

<View
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1" />


<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal"
android:weightSum="10">

<View
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="2" />

<Button
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_weight="3"
android:text="Button 4" />

<Button
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_weight="3"
android:text="Button 5" />

<View
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="2" />

</LinearLayout>


<View
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1" />


<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal"
android:weightSum="4">
```

```
<Button
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:text="Button 6" />

<Button
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:text="Button 7" />

<Button
android:layout_width="0dp"
android:layout_height="match_parent"
android:layout_weight="2"
android:text="Button 8" />
</LinearLayout>

</LinearLayout>
```

ThirdActivity.kt

```
package ua.kpi.lab2

import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat

class ThirdActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_third)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(android.R.id.content)) { v, insets
        ↪ ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
    }
}
```

activity_third.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:padding="16dp">

<View
android:id="@+id/topCenterAnchor"
android:layout_width="0dp"
android:layout_height="0dp"
android:layout_centerHorizontal="true" />

```

```
<Button
android:id="@+id/btnLeft50"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentStart="true"
android:layout_alignParentTop="true"
android:layout_marginEnd="5dp"
android:layout_toStartOf="@id/topCenterAnchor"
android:text="Left 50%" />

<Button
android:id="@+id/btnRight50"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_alignParentEnd="true"
android:layout_marginStart="5dp"
android:layout_toEndOf="@id/topCenterAnchor"
android:text="Right 50%" />

<Button
android:id="@+id/btnCenter"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:text="Center" />

<Button
android:id="@+id/btnCenterLeft"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@id/btnCenter"
android:layout_toStartOf="@id/btnCenter"
android:text="Center_left" />

<Button
android:id="@+id/btnCenterRight"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@id/btnCenter"
android:layout_toEndOf="@id/btnCenter"
android:text="Center_right" />


<Button
android:id="@+id/btnBottom"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:text="Bottom" />

</RelativeLayout>
```

FourthActivity.kt

```
package ua.kpi.lab2

import android.os.Bundle
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
```

```
class FourthActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_fourth)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(android.R.id.content)) { v, insets
        ↪ ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
    }
}
```

activity_fourth.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
↪ xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#FFFFFF">

<Button
android:id="@+id/btnCustom"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="@drawable/fourth_btn"
android:paddingStart="20dp"

android:paddingTop="10dp"
android:paddingEnd="20dp"
android:paddingBottom="10dp"
android:text="New Button"

android:textColor="#000000"
app:backgroundTint="@null"

app:icon="@drawable/baseline_android_24"
app:iconGravity="textStart"
app:iconPadding="8dp"
app:iconTint="#3DDC84"

app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

fourth_btn.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
<item android:state_pressed="true">
<shape android:shape="rectangle">
<solid android:color="#90EE90" />

<corners android:radius="24dp" />
```

```
8
9   <stroke android:width="4dp" android:color="#505050" />
10  </shape>
11  </item>
12
13  <item>
14  <shape android:shape="rectangle">
15  <solid android:color="#FFFFFF" />
16
17  <corners android:radius="24dp" />
18
19  <stroke android:width="4dp" android:color="#505050" />
20  </shape>
21  </item>
22  </selector>
```