МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Звіт

з практичної роботи № 2 з дисципліни

Програмування під Android

Виконала: Перевірив:

ст. гр. ПЗПІ-23-5 ст. викладач кафедри ПІ

Євсєєва Олександра Сокорчук Ігор Петрович

Харків 2024

ЗАВДАННЯ

1. Розмітка екрана
   1. Підготовка робочого середовища
   2. Практика з Linear/Constraint-Layout
   3. Практика з атрибутами елементів
   4. Обробка подій
2. Файли розмітки Layout
   1. Створення різних Layout
   2. Практика роботи з атрибутами
3. Життєвий цикл та методи Activity
   1. Ознайомлення з життєвим циклом Activity
   2. Дослідження методів життєвого циклу
   3. Вивчення поведінки Activity при зміні орієнтації екрана
   4. Практичне завдання з ручного збереження стану
   5. Дослідження переходу між Activity
   6. Обробка завершення Activity

ХІД РОБОТИ

1. Розмітка екрана

Відкриваємо Android Studio та створюємо новий проєкт, вибравши шаблон Empty Views Activity. Відкрийте файл res/layout/activity\_main.xml. [додаток А]

Практика з LinearLayout: cтворюємо розмітку, що містить кілька кнопок, розташованих одна під одною. Використовуємо властивості orientation для вертикальної орієнтації. Задаємо для кожної кнопки властивості layout\_width та layout\_height, поекспериментуймо з різними значеннями, як wrap\_content та match\_parent та властивістю weight.

Практика з ConstraintLayout: розміщуємо елементи на екрані, використовуючи обмеження для позиціонування (constraints), щоб задати їх взаємне розташування. У моєму випадку я розташовувала інші розмітки усередині цієї розмітки.

Практика з RelativeLayout: розміщуємо кілька елементів відносно один одного, використовуючи атрибути, такі як layout\_below, layout\_toRightOf.

Практика з FrameLayout: додаємо кілька кнопок, щоб побачити, як вони накладаються один на одного.

Практика з GridLayout: розмістимо 6 кнопок у вигляді таблиці, задаючи кількість стовпців і рядків.

Використовуємо такі атрибути, як padding, margin, gravity та layout\_gravity для налаштування відступів та вирівнювання елементів. Налаштовуємо текстові поля.

1. Обробка подій.

Додаємо до кнопки обробник події натискання у файлі MainActivity.java.

1. public void onButtonClick(View v) {  
2. Button button = findViewById(R.id.*button1*);  
3. button.setText(getString(R.string.*newButtonName*));  
4.}

У файлі activity\_main.xml додайте атрибут android:onClick="onButtonClick" до кнопки, а в класі MainActivity створіть відповідний метод onButtonClick.

1. android:onClick="onButtonClick"

1. Життєвий цикл та методи Activity

Додаємо до кожного методу життєвого циклу (onCreate, onStart, onResume тощо) функцію виведення повідомлень у лог (Log.d) для відстеження послідовності викликів. Всі вони однакові, тож приведу лише один як приклад:

1. Log.d(TAG, "onStart: ");

Змінюємо орієнтацію пристрою (з портретної на ландшафтну) та спостерігайте за викликами методів життєвого циклу у логах.

Досліджуємо, як зміна орієнтації перезапускає Activity (викликаються методи onPause(), onStop(), onDestroy(), а потім onCreate(), onStart(), onResume()).

Додаємо в Activity поле для введення тексту (наприклад, EditText).

При зміні орієнтації екрана текст у полі буде очищено, оскільки Activity перезавантажується.

Але воно не буде видалено та створено знову, якщо дати елементу унікальний id. Отже, методичні вказівки трохи помилюяться.

Було реалізовано збереження та відновлення вручну у додатковому завданні. Було створено клікер, що зберігаю кількість кліків.

1. public void onIncreaseCounter(View v) {

2. TextView count = findViewById(R.id.count);

3. Integer previousCounter = Integer.parseInt(count.getText().toString());

4. Integer newValue = previousCounter + 1;

5. count.setText(newValue.toString());

6. }

Зберігається за допомгою:

1.protected void onSaveInstanceState(@NonNull Bundle outState) {

2. TextView count = findViewById(R.id.count);

3. outState.putString("counter", count.getText().toString());

4. super.onSaveInstanceState(outState);

Відновлюється за допогою:

1. protected void onRestoreInstanceState(@NonNull Bundle savedInstanceState){

2. super.onRestoreInstanceState(savedInstanceState);

3. TextView count = findViewById(R.id.count);

4. count.setText(savedInstanceState.getString("counter"));

5. }

Створюємо друге Activity у нашому застосунку.

Додаємо кнопку для переходу до другого Activity.

Відстежуємо виклики методів життєвого циклу при переході між Activity.

Додаємо кнопку для завершення Activity (виклик методу finish()).

1. public void onFinishClick(View v) {

2. finish();

3. }

ВИСНОВКИ

Під час практичної роботи було досліджено: створення та налаштування різних розміток, використовуючи XML-файли. Виявлено, як компоненти та атрибути взаємодіють між собою.

Ознайомилися із життєвим циклом компонентів Activity у застосунках Android. Навчилися керувати станами Activity через використання ключових методів життєвого циклу.

Відеозвіт на YouTube. URL: <https://youtu.be/jZjEamo40Ic>

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Методичні вказівки до практичних занять: Розмітка екрана в Android. URL: <https://dl.nure.ua/mod/page/view.php?id=599542> (дата звернення: 07.10.2024)
2. Методичні вказівки до практичних занять: Файли розмітки Layout. URL: <https://dl.nure.ua/mod/page/view.php?id=599543> (дата звернення: 07.10.2024)
3. Методичні вказівки до практичного заняття: Життєвий цикл та методи Activity. URL: <https://dl.nure.ua/mod/page/view.php?id=599544> (дата звернення: 07.10.2024)

ДОДАТОК А

Файл розмітки.

<?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"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<TextView

android:id="@+id/text1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!"

android:textSize="32sp"

android:textStyle="bold"

android:textColor="@color/textcolor"

app:layout\_constraintBottom\_toTopOf="@+id/button1"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_chainStyle="spread"/>

<Button

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/button1Name"

android:onClick="onButtonClick"

app:layout\_constraintTop\_toBottomOf="@id/text1"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintBottom\_toTopOf="@id/image1"

app:layout\_constraintVertical\_chainStyle="spread"

/>

<ImageView

android:id="@+id/image1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@+id/button1"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintBottom\_toTopOf="@id/ll"

app:layout\_constraintVertical\_chainStyle="spread"

android:src="@drawable/cat"

/>

<LinearLayout

android:id="@+id/ll"

android:layout\_marginStart="30dp"

android:layout\_marginEnd="30dp"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

app:layout\_constraintBottom\_toTopOf="@id/gl"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@id/image1">

<Button

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/button1Name"

/>

<Button

android:id="@+id/button3"

android:padding="5dp"

android:layout\_gravity="center"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/button1Name"

/>

<Button

android:id="@+id/button4"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/button1Name"

/>

</LinearLayout>

<GridLayout

android:id="@+id/gl"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:columnCount="3"

android:layout\_marginStart="16dp"

android:layout\_marginEnd="16dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@id/ll">

<Button android:text="1" android:onClick="onButtonClickOther"/>

<Button android:text="2" android:onClick="onButtonClickOther"/>

<Button android:text="3" android:onClick="onButtonClickOther"/>

<Button android:text="4" android:onClick="onButtonClickOther"/>

<Button android:text="5" android:onClick="onButtonClickOther"/>

<ImageView

android:id="@+id/image2"

android:layout\_width="50dp"

android:layout\_height="50dp"

android:src="@drawable/cat"/>

<TextView

android:id="@+id/text2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!"

android:textColor="@color/textcolor"/>

</GridLayout>

</androidx.constraintlayout.widget.ConstraintLayout>