МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Звіт

з практичної роботи №2

Виконав: Перевірив:

ст. гр. ПЗПІ-23-2 ст. викладач кафедри ПІ

Семьонов Олег Сокорчук Ігор Петрович

Харків 2024

ПУНКТИ ЗАВДАНЬ

1.Розмітка екрана

1.1 **Редагування XML-файлу розмітки**

**1.2 Практика з LinearLayout**

**1.3 Практика з ConstraintLayout**

**1.4 Практика з атрибутами елементів**

**1.5 Обробка подій кнопки**

**1.6 Завдання для самостійної роботи**

**2.Файли розмітки Layout**

**2.1 Основи роботи з XML-розміткою**

**2.2 Створення різних Layout**

**2.3 Практика роботи за атрибутами**

**2.4 Додаткові завдання**

**2.5 Завдання для самостійної роботи**

**3.Життєвий цикл та методи Activity**

**3.1 Дослідження методів життєвого циклу**

**3.2 Вивчення поведінки при зміні орієнтації екрана**

**3.3 Практичне завдання з ручного збереження стану**

**3.4 Дослідження переходу між активностями**

**3.5 Обробка завершення активності**

**3.6 Реалізація застосунку зі збереженням стану користувача**

**3.7 Додавання таймеру до застосунку**

ХІД РОБОТИ

1. РОЗМІТКА ЕКРАНА

1.1 **Редагування XML-файлу розмітки**

Відкриємо файл res/layout/activity\_main.xml та створимо простий інтерфейс з використанням елементів, таких як TextView, Button, та ImageView:

1.2 Практика з LinearLayout

Створимо нову розмітку з використанням LinearLayout. В ній додамо декілька кнопок розташованих одне під одною.

Для розмітки задамо атрибут android:orientation="vertical", для того щоб кнопки йшли у вертикальному порядку. Для кнопок задамо властивості layout\_width та layout\_height з значеннями wrap\_content та match\_parent.

1.3 Практика з ConstraintLayout

Створимо макет з використанням різних елементів, використовуючи обмеження для позиціюванняю

1.4 **Практика з атрибутами елементів**

**Дослідимо атрибути, як** padding**,**margin**,**gravity**та**layout\_gravity**для налаштування відступів та вирівнювання елементів, а також налаштуємо текстові поля( розмір шрифту, колір тощо)**

**1.5 Обробка подій кнопки**

**Додамо до кнопки обробник подій.**

**Для самої кнопки додамо атрибут** android:onClick="onButtonClick"**, а у коді реалізуємо відповідний метод, а саме:**

**private int count = 0;**

**public void onButtonClick(View v){**

**count++;**

**Button button = (Button) v;**

**button.setText("clicked" + count);**

**}**

**Данний метод буде підраховувати кількість натиснень на кнопку.**

**1.6 Завдання для самостійної роботи**

**Створимо розмітку GridLayout та розмістимо у ній текстові поля, кнопки та зображення. Додамо кілька кнопок з різним функціоналом. Перша кнопка буде такою самою, як у минулому пункті. Друга кнопка буде змінювати колір на червоний при натиснені:**

**public void onColorClick(View v){**

**Button button = findViewById(R.id.button6);**

**int color = Color.rgb(255,0,0);**

**button.setBackgroundColor(color);**

**}**

**А третя кнопка буде змінювати картинку на невидиму:**

**public void onClearClick(View v){**

**ImageView image = findViewById(R.id.imageView3);**

**image.setVisibility(INVISIBLE);**

**}**

**2.ФАЙЛИ РОЗМІТКИ LAYOUT**

**2.1 Основи роботи з XML-розміткою**

**Використаємо LinearLayout як основний контейнер. Також змінемо його орієнтацію на вертикальну й додамо такі елементи як TextView, Button, ImageView і змінимо їх ширину та висоту.**

**2.2 Створення різних Layout**

**Створимо макет на основі LinearLayout з використанням кнопок одної під одною та використанням атрибутів gravity i layout\_gravity:**

**android:gravity="center"**

**<Button**

**android:id="@+id/button7"**

**android:layout\_width="186dp"**

**android:layout\_gravity="left"**

**android:layout\_height="wrap\_content"**

**android:text="Button" />**

**Далі створимо макет на основі RelativeLayout та розмістимо декілька елементів використовуючи атрибути** layout\_below**.**

**android:layout\_below="@id/button4"**

**Також створимо складну розмітку ConstraintLayout та розташуємо елементи відносно контейнера та один одного. Приклад:**

**app:layout\_constraintBottom\_toBottomOf="parent"**

**app:layout\_constraintEnd\_toEndOf="parent"**

**app:layout\_constraintStart\_toStartOf="parent"**

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

**Тепер перейдемо до FrameLayout. Створимо картинку та кнопку, щоб побачити накладання кнопки на цю саму картинку. Для цього треба лише створити кнопку після картинки і тоді вона автоматично буде «вище», ніж картинка.**

**Останнім пунктом було створення макету на основі GridLayout, тобто таблиці. Заповнимо її текстом та кнопками, попередньо визнавич кількість рядків і стовпців:**

**android:columnCount="2"**

**android:rowCount="2"**

**2.3 Практика роботи за атрибутами**

**Дослідимо атрибути, такі як** padding**,**margin**,**textSize**,**textColor**, а також властивість weight у LinearLayout.**

**2.4 Додаткові завдання**

**Створимо інтерфейс для екрану реєстрації користувача з зображенням, двума полями введення та кнопкою. Також створимо інтерфейс для ландшафтного режиму. Андроід автоматично буде змінювати розмітку в залежності від орієнтації екрану завдяки правильно налаштованим розміткам, а саме створенню папки layout-land.**

**2.5 Завдання для самостійної роботи**

**Створимо інтерфейс користувача для застосунку калькулятора за допомогою GridLayout та додамо кілька кнопок, щоб вони виводили текст до текстового поля:**

**public void onButtonClick(View v){**

**Button button = (Button) v;**

**String btnText = button.getText().toString();**

**if (btnText.equals("Clear")){**

**output.setText("");**

**} else{**

**output.append(btnText);**

**}**

**}**

**output це змінна нашого текстого поля. Данний метод є атрибутом кожної кнопки і виводить текст самої кнопки (1,2,3...), а також доданий if-statement ( коли натискається кнопка Clear, то поле очищається, а інакше пишеться текст)**

**3.ЖИТТЄВИЙ ЦИКЛ ТА МЕТОДИ ACTIVITY**

**3.1 Дослідження методів життєвого циклу**

**Додамо до кожного методу життєвого циклу функцію логування для відстеження послідовності викликів. Наприклад для методу onCreate це буде виглядати так:**

**Log.d(TAG, "onCreate: ");**

**Та прослідкуємо за послідовністю виклику (onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy())**

**3.2 Вивчення поведінки при зміні орієнтації екрана**

**Також дослідимо як зміна орієнтації перезапускає нашу активність( також за допомогою логування).**

**Фактично маємо таку послідовність:**

**(onPause(), onStop(), onDestroy(),onCreate(), onStart(), onResume()),**

**тобто наша активність перезапускається.**

**3.3 Практичне завдання з ручного збереження стану**

**Додамо в активність поле для введення тексту (EditText) та зробимо так, щоб при зміні орієнтації екрана текст у полі залишався. Зробимо це за домогою методів onSaveInstanceState() та onRestoreInstanceState().**

**protected void onSaveInstanceState(@NonNull Bundle outState){**

**super.onSaveInstanceState(outState);**

**outState.putString("text", editText.getText().toString());**

**}**

**protected void onRestoreInstanceState(@NonNull Bundle savedInstanceState){**

**super.onRestoreInstanceState(savedInstanceState);**

**String text = savedInstanceState.getString("text");**

**editText.setText(text);**

**}**

**Перший метод зберігає нашу строку, а другий відновлює при зміні орієнтації.**

**3.4 Дослідження переходу між активностями**

**Створимо другу активність у нашому застосунку та додамо кнопку для переходу на цю другу активність.**

**public void onButtonClick(View v){**

**startActivity(new Intent(this, MainActivity2.class));**

**}**

**та додамо логування для стандартних методів. Після цього передивимось за допомогою логів, як себе поводять активності при переході**

**3.5 Обробка завершення активності**

**Додамо другій активності кнопку для завершення активності і переглянемо на послідовність викликів методів життєвого циклу.**

**public void onButtonFinish(View v){**

**finish();**

**}**

**3.6 Реалізація застосунку зі збереженням стану користувача**

**Реалізуємо невеличкий застосунок зі збереженням кількості кліків.**

**Для цього додамо Button i TextView.**

**Пропишемо метод для кнопки, при натиснені на яку буде додаватися кількість кліків:**

**public void onCounterClick(View v) {**

**counter++;**

**txt = findViewById(R.id.textView3);**

**txt.setText("Clicks: " + counter);**

**}**

**Тепер зробимо так, щоб при зміні орієнтації екрану наші кліки зберігались. Використаємо вже згадані методи: onSaveInstanceState() та onRestoreInstanceState().**

**protected void onSaveInstanceState(@NonNull Bundle outState){**

**super.onSaveInstanceState(outState);**

**outState.putInt("counter",counter);**

**}**

**protected void onRestoreInstanceState(@NonNull Bundle savedInstanceState){**

**super.onRestoreInstanceState(savedInstanceState);**

**counter = savedInstanceState.getInt("counter");**

**txt = findViewById(R.id.textView3);**

**txt.setText("Clicks: " + counter);**

**}**

**Тепер при змінні орієнтації наші кліки залишуться на місці.**

**3.7 Додавання таймеру до застосунку**

**Останнім пунктом є додавання до застосунку таймеру, який автоматично зупиняється при виклику методі onPause() та поновлюється після onResume().**

**Код:**

**private CountDownTimer countDownTimer;**

**private long timeInMillis = 120000;**

**private boolean timerRunning = false;**

**private void startTimer(){**

**countDownTimer = new CountDownTimer(timeInMillis,1000){**

**@Override**

**public void onTick(long millisUntilFinished) {**

**timeInMillis = millisUntilFinished;**

**updateTimerText();**

**}**

**@Override**

**public void onFinish(){**

**timerRunning = false;**

**}**

**}.start();**

**timerRunning = true;**

**}**

**private void pauseTimer(){**

**if (timerRunning) {**

**countDownTimer.cancel();**

**timerRunning = false;**

**}**

**}**

**private void resumeTimer(){**

**if (!timerRunning) {**

**startTimer();**

**}**

**}**

**private void updateTimerText(){**

**int seconds = (int) (timeInMillis / 1000);**

**TextView timerView = findViewById(R.id.timer);**

**timerView.setText(String.valueOf(seconds));**

**}**

**Також додамо збереження та відновлення стану.**

**protected void onSaveInstanceState(@NonNull Bundle outState){**

**super.onSaveInstanceState(outState);**

**outState.putLong("timeInMillis", timeInMillis);**

**outState.putBoolean("timerRunning", timerRunning);**

**}**

**protected void onRestoreInstanceState(@NonNull Bundle savedInstanceState){**

**super.onRestoreInstanceState(savedInstanceState);**

**timeInMillis = savedInstanceState.getLong("timeInMillis");**

**timerRunning = savedInstanceState.getBoolean("timerRunning");**

**}**

**До методів життєвого циклу додамо методи таймеру.**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

**startTimer();**

**}**

**protected void onResume(){**

**super.onResume();**

**resumeTimer();**

**}**

**protected void onPause(){**

**super.onPause();**

**pauseTimer();**

**}**

**ВИСНОВКИ**

**Під час данної практичної роботи було досліджено різні розмітки, їх налаштування та особливості. Також було вивчено нові атрибути до розміток. Далі було вивчено методи життєвого циклу активності, їх послідовність, а також керування станами активності за допомогою спеціальних методів.**

**Відеозвіт. URL:** <https://youtu.be/yhhyuPJQlt4>

**ВИКОРИСТАНІ ДЖЕРЕЛА**

**1.** [**ПЗ Розмітка екрана**](https://dl.nure.ua/mod/page/view.php?id=599542)

<https://dl.nure.ua/mod/page/view.php?id=599542>

**2. ПЗ Файли розмітки Layout**

<https://dl.nure.ua/mod/page/view.php?id=599543>

**3.****ПЗ Життєвий цикл та методи Activity**

<https://dl.nure.ua/mod/page/view.php?id=599544>

**4.** **CountDownTimer**

<https://developer.android.com/reference/android/os/CountDownTimer>