МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

**ЗВІТ**

з практичного заняття №3

з дисципліни “Програмування під Android”

Виконав: Перевірив:

ст. гр. ПЗПІ-23-4 Сокорчук І. П.

Кулаков С. М.

Харків, 2024

ЗАВДАННЯ

Ознайомитися з основними типами діалогових панелей в Android.

Навчитися створювати та використовувати діалогові панелі в Android-застосунках.

Засвоїти методи для налаштування діалогових панелей та їх взаємодії з користувачем.

Засвоїти концепцію черги повідомлень (Message Queue) в Android.

Вивчити роботу з обробником повідомлень (Handler) для взаємодії з потоками.

Навчитися створювати та використовувати Handler для асинхронної обробки даних.

Навчитися працювати з компонентом **RecyclerView** для створення списків із прогортанням в Android, а також ознайомити їх з принципами створення та використання адаптерів (Adapters) і керування даними у списках.

ХІД РОБОТИ

В Android-Studio створюємо новий проєкт. Почнемо створення наших Діалогів. Перший на черзі - AlertDialog. Це діалогове вікно з можливістю вибору між кількома варіантами або підтвердженням дії. В моєму випадку він містить дві кнопки - ОК та Cancel.

Кнопка для виклику AlertDialog:

1. <Button  
   2. android:id="@+id/showDialogButton"  
   3. android:layout\_width="wrap\_content"  
   4. android:layout\_height="wrap\_content"  
   5. android:text="ShowDialog"  
   6. android:layout\_marginRight="16dp" />

Ініціалізація в MainActivity:

1.new AlertDialog.Builder(MainActivity.this)  
2. .setTitle("Діалог")  
3. .setMessage("Це приклад AlertDialog.")  
4. .setPositiveButton("OK", new 5.DialogInterface.OnClickListener() {  
6. public void onClick(DialogInterface dialog, 7. int which) {}  
8. })  
9. .setNegativeButton("Cancel", new 10.DialogInterface.OnClickListener() {  
11. public void onClick(DialogInterface dialog, 12.int which) {}  
13. })  
14. .show();

Другий з нашого списку - CustomDialog. Цей діалог цікавий тим, що в ньому можна власноруч змінювати view, тобто зробити розмітку на свій смак. Тож, кнопка виклику даного діалогу:

1. <Button  
   2. android:id="@+id/showCustomDialog"  
   3. android:layout\_width="wrap\_content"  
   4. android:layout\_height="wrap\_content"  
   5. android:text="ShowCustomDialog" />

Тепер, як я зазначив ми маємо створити розмітку для цього діалогу у окремому xml. файлі. [ДОДАТОК А]

Далі йде DatePickerDialog. Як можна здогадатися з назви, цей діалог призначений для вибору дати з календарю. Кнопка для виклику календарю:

1. <Button  
   2. android:id="@+id/showDatePickerButton"  
   3. android:layout\_width="wrap\_content"  
   4. android:layout\_height="wrap\_content"  
   5. android:text="DatePickerButton" />

Реалізація в MainActivity:

1. showDatePickerButton.setOnClickListener(new 2.View.OnClickListener() {  
   3. @Override  
   4. public void onClick(View v) {  
   5. DatePickerDialog datePickerDialog = new 6.DatePickerDialog(MainActivity.this,  
   7. new DatePickerDialog.OnDateSetListener() 8.{  
   9. @Override  
   10. public void onDateSet(DatePicker 11.view, int year, int month, int dayOfMonth) {  
   12. }  
   13. }, 2023, 8, 1);  
   14. datePickerDialog.show();  
   15. }  
   16. });

Наступний крок цього практичного заняття - Handler.

**Handler** – це об’єкт, який дозволяє відправляти та обробляти повідомлення та виконувані об’єкти (Runnable) в потоці, пов’язаному з ним.

Спочатку у файлі розмітки створимо такі елементи: кнопку та текстове поле:

1. <Button  
   2. android:id="@+id/startHandlerButton"  
   3. android:layout\_width="wrap\_content"  
   4. android:layout\_height="wrap\_content"  
   5. android:text="Start Handler" />  
   6.<TextView  
   7. android:id="@+id/handlerMessageTextView"  
   8. android:layout\_width="wrap\_content"  
   9. android:layout\_height="wrap\_content"  
   10. android:text="Зміна повідомлення при натисненні кнопки"  
   11. android:layout\_marginTop="20dp" />

Далі, в MainActivity створимо Handler, прив’заний до головного потоку і реалізуємо зміну тексту в нашому текстовому через 2 секунди після натиснення на кнопку.

1.@Override  
2.public void onClick(View v) {  
3. handler.postDelayed(new Runnable() {  
4. @Override  
5. public void run() {  
6. TextView textView = 7.findViewById(R.id.*handlerMessageTextView*);  
8. textView.setText("Обробник виконано після затримки");  
9. }  
10. }, 2000);  
11.}

Переходимо до списків з прогортанням (RecycleView Adapter)

Спочатку створимо макет для основної активності, для цього була використана LinearLayout в якій знаходиться RecycleView, що дозволяє відображати всі елементи списку.

1. <LinearLayout 2.xmlns:android="http://schemas.android.com/apk/res/android"  
   3. android:layout\_width="match\_parent"  
   4. android:layout\_height="match\_parent"  
   5. android:orientation="vertical"  
   6. android:id="@+id/main"  
   7. android:gravity="center"  
   8. android:padding="16dp">  
   9. <androidx.recyclerview.widget.RecyclerView  
   10. android:id="@+id/recyclerView"  
   11. android:layout\_width="match\_parent"  
   12. android:layout\_height="match\_parent"  
   13. android:layout\_marginTop="20dp" />  
   14. </LinearLayout>

Далі створюємо розмітку нашого окремого елементу:

1. <LinearLayout  
    2.xmlns:android="http://schemas.android.com/apk/res/android"  
   3. android:layout\_width="match\_parent"  
   4. android:layout\_height="wrap\_content"  
   5. android:orientation="vertical"  
   6. android:padding="16dp">  
   7. <TextView  
   8. android:id="@+id/textView"  
   9. android:layout\_width="match\_parent"  
   10. android:layout\_height="wrap\_content"  
   11. android:textSize="18sp"  
   12. android:textColor="#000000" />  
   13. </LinearLayout>

Далі створимо клас адаптрера MyAdapter. Цей адаптер використовує об’єкт MyViewHolder для представлення кожного елементу списку. В конструкторі адаптера передається список даних, що буде використовуватися для відображення елементів у RecycleView.

1. public MyAdapter(List<String> myDataset) {  
   2. mDataset = myDataset;  
   3. }

У методі onCreateViewHolder виконується інфляція макету для кожного елементу списку, а в методі onBindViewHolder окремий елемент отримує свій текст.

Повернемося до MainActivity. Тут ми спершу створюємо масив елементів для відображення.

1. List<String> myDataset = new ArrayList<>();  
   2. for (int i = 0; i <= 30; i ++){  
   3. myDataset.add("Елемент" + i);  
   4. }

Далі для нашего RecycleView встановлюємо менеджер макета LinearLayoutManager, що дозволяє відображати елементи у вертикальному списку.

1. RecyclerView recyclerView = findViewById(R.id.*recyclerView*);  
   2. recyclerView.setLayoutManager(new LinearLayoutManager(this));

І створюємо об’єкт класу MyAdapter та передаємо йому наш створений список myDataSet як джерело даних.

1. MyAdapter adapter = new MyAdapter(myDataset);  
   2. recyclerView.setAdapter(adapter);

ВИСНОВКИ

під час виконання лабораторної роботи я ознайомився та створив різні діалогові панелі, обробники подій Handler. Також я навчився працювати з компонентом RecycleView для створення списків із прогортанням.

3.4 Посилання: <https://youtu.be/SPeX-vBFGI0>

3.5 Використані джерела:

1- <https://dl.nure.ua/mod/page/view.php?id=611590>

2 - <https://dl.nure.ua/mod/page/view.php?id=611591>

3 - <https://dl.nure.ua/mod/page/view.php?id=611592>

3.6 Додатки

ДОДАТОК А

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/customDialogText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Це індивідуальна діалогова панель" />  
  
 <EditText  
 android:id="@+id/customDialogInput"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Введіть текст" />  
  
</LinearLayout>