МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Звіт

з практичної роботи № 3 з дисципліни

Програмування під Android

Виконала: Перевірив:

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

Висоцький Ігор Сокорчук Ігор Петрович

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

ХІД РОБОТИ

Cтворюємо новий проєкт та обираємо Empty Views Activity.

Діалогові панелі (Dialog) – це невеликі вікна, які з'являються на екрані поверх поточного інтерфейсу, щоб привернути увагу користувача або запросити в нього додаткові дії.

Типи діалогових панелей:

* AlertDialog: діалогове вікно з можливістю вибору між кількома варіантами або підтвердження певної дії.
* DatePickerDialog: дозволяє користувачу вибрати дату.
* CustomDialog: діалог з індивідуальним дизайном.

Почнемо із AlertDialog: додаємо кнопку для виклику діалогу:

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

Ініцалізуємо та підписуємо подію за допомогою setOnClickListener у MainActivity:

Далі по списку DatePickerDialog. Аналогічно свторюємо кнопку із id date, ініціалізуємо та підписуємось на події із setOnClickListener із лямбдою. У виклику використовуємо допоміжний клас Calendar для визначення актуальної дати. Цей діалог дозволяє обрати дати у зручному вікні вибору, схожий на звичайний календар:

1. adapter = new MyAdapter(dataset);  
2.recyclerView.setAdapter(adapter);  
3. // AlertDialog  
4. Button showDialogButton = findViewById(R.id.*showDialogButton*);  
5. showDialogButton.setOnClickListener(v -> new

6. AlertDialog.Builder(MainActivity.this)  
7. .setTitle("Діалог")  
8. .setMessage("Це приклад AlertDialog.")  
9. .setPositiveButton("OK", (dialog, which) -> {  
10. // Дія при натисканні OK  
11. })  
12. .setNegativeButton("Cancel", (dialog, which) -> {  
13. // Дія при натисканні Cancel  
14. })  
15. .show());

Після підтвердження вибору на викликаному діалогом view спливає повідомлення із обраною датою. Це створено за допомогою елемента Toast.

Далі CustomDialog. Він трохи відрізняється від інших тим, що має особливе view, що можна модифікувати самому. Якщо більш конкретніше, то можна зробити власну розмітку для конкретного діалогу. Отже, спочатку створимо розмітку custom\_dialog.xml, в який помістимо напис «Це індивідуальна діалогова панель» та поле для редагування тексту з підтекстом «Введіть текст

» дві кнопки для виходу із діалоги з позитивним та негативним результатами [ДОДАТОК А].

Обов’язково вказуємо, яку розмітку використовувати:

1. LayoutInflater inflater = getLayoutInflater();  
2. View dialogView = inflater.inflate(R.layout.*custom\_dialog*, null);

У коді описано створення та налаштування діалогу: його стиль та текстове наповнення.

Із діалогами закінчили. Перейдемо до обробки Handlers.

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

MessageQueue – черга, яка зберігає повідомлення та Runnable, що будуть оброблені об'єктом Handler.

Looper – це механізм, що циклічно обробляє чергу повідомлень для потоку.

Подивимось на реалізацію виконання деякої дії із затримкою на 5 секунди, що виконується після натискання кнопки із id «*startHandlerButton*»:

Використання поста з затримкою:  
1. MainHandler.postDelayed(() -> {  
2. TextView textView = findViewById(R.id.*handlerMessageTextView*);  
3. textView.setText("Handler executed after delay");  
4. }, 5000); // 5 секунди

Вся інша реалізація Handler наведена у [ДОДАТОК А].

Переходимо до роботи із RecyclerView.

* RecyclerView – основний контейнер для відображення списків або сіток елементів.
* LayoutManager – керує розташуванням елементів у списку (лінійний, сітковий або кастомний макети).
* Adapter – клас, який адаптує дані для відображення в RecyclerView.
* ViewHolder – зберігає посилання на елементи інтерфейсу для повторного використання.

Почнемо із додавання RecyclerView до файлу розмітки:

1. <androidx.recyclerview.widget.RecyclerView  
2. android:id="@+id/recyclerView"  
3. android:layout\_width="match\_parent"  
4. android:layout\_height="match\_parent"  
5. android:layout\_marginTop="20dp" />

Далі для відображення елементів у RecyclerView використовуємо GridLayoutManager

для відображення компонентів у колонках. Самі компоненти, що відображаються називаються item.

В основному файлі коду генеруємо наповнення списку та викликаємо сам RecyclerView:

1. // Ініціалізуємо RecyclerView  
2. recyclerView = findViewById(R.id.*recyclerView*);  
3. recyclerView.setLayoutManager(new GridLayoutManager(this, 2));  
4. // Заповнюємо дані  
5. dataset = new ArrayList<>();  
6. for (int i = 1; i <= 30; i++) {  
7. dataset.add("Item " + i);  
8. }  
9. // Створюємо адаптер і встановлюємо його для RecyclerView  
10. adapter = new MyAdapter(dataset);  
11. recyclerView.setAdapter(adapter);

Створюємо ItemAdapter, окремий клас для того, щоб пояснити списку, як показувати елементи у ньому [ДОДАТОК Б].

Першим кроком створюємо onCreateViewHolder:

1. @NonNull  
2. @Override  
3. public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
4. // Створюємо представлення для кожного елемента  
5. View v = LayoutInflater.*from*(parent.getContext())  
6. .inflate(R.layout.*item\_text\_view*, parent, false);  
7. return new MyViewHolder(v);  
8. }

Та створюємо MyViewHolder:

1. public static class MyViewHolder extends RecyclerView.ViewHolder {  
2. public TextView textView;  
3.  
4. public MyViewHolder(View v) {  
5. super(v);  
6. textView = v.findViewById(R.id.*textView*);  
7. }  
8. }

Цьома двома кроками було створено каркас для списку. Тобто, тепер елемент знає що і як треба помістити у блок списку. Тепер напишемо, як саме треба наповнювати компоненти item:

1.public void onBindViewHolder(MyViewHolder holder, int position) {  
2. holder.textView.setText(mDataset.get(position));

3. }

В onCreateViewHolder зазначено тип view, за яким визнається яку саму розмітку програма буде використовувати у тому чи іншому випадку.   
  
Повний код для файлів розміток дивитись у [ДОДАТОК В].

ВИСНОВКИ

Під час практичної роботи було створено додаток для ознайомлення та опановування основних типів діалогових панелей та їхні атрибути налаштування; засвоєння концепції черги повідомлень, оброки подій Handler; навчання працювати із з компонентом RecyclerView для створення списків із прогортанням в Android та використання адаптерів (Adapters), керування даними у списках.

Відеозвіт на YouTube. URL: <https://youtu.be/eBMX7uwM3gI>

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Документація AndroidDevelopers dialogs. URL: <https://developer.android.com/develop/ui/views/components/dialogs> (дата звернення: 02.11.2024)
2. Документація AndroidDevelopers handlers. URL: <https://developer.android.com/reference/android/os/Handler> (дата звернення: 02.11.2024)
3. Питання Stack overflow «Why use handlers?» URL: <https://stackoverflow.com/questions/12312926/why-use-handler> (дата звернення: 02.11.2024)
4. Відеоурок YouTube Ukrainian android dev «Android УКРАЇНСЬКОЮ Урок 21 Recycler View Простий приклад» URL: <https://www.youtube.com/watch?v=J7NWMgi09MA> (дата звернення: 02.11.2024)
5. Документація AndroidDevelopers Recycler View. URL: <https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView> (дата звернення: 02.11.2024)

ДОДАТОК А

package ihor.vysotskyi.nure;  
  
import android.app.AlertDialog;  
import android.app.DatePickerDialog;  
import android.os.Bundle;  
import android.os.\*;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.widget.Button;  
import android.widget.DatePicker;  
import android.widget.TextView;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.graphics.Insets;  
import androidx.core.view.ViewCompat;  
import androidx.core.view.WindowInsetsCompat;  
import androidx.recyclerview.widget.GridLayoutManager;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.List;  
  
public class MainActivity extends AppCompatActivity {  
  
 private Handler mainHandler;  
 private Handler backgroundHandler;  
 private RecyclerView recyclerView;  
 private MyAdapter adapter;  
 private List<String> dataset;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (v, insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*());  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
 return insets;  
 });  
 // Ініціалізуємо RecyclerView  
 recyclerView = findViewById(R.id.*recyclerView*);  
 recyclerView.setLayoutManager(new GridLayoutManager(this, 2));  
 // Заповнюємо дані  
 dataset = new ArrayList<>();  
 for (int i = 1; i <= 30; i++) {  
 dataset.add("Item " + i);  
 }  
 // Створюємо адаптер і встановлюємо його для RecyclerView  
 adapter = new MyAdapter(dataset);  
 recyclerView.setAdapter(adapter);  
 // AlertDialog  
 Button showDialogButton = findViewById(R.id.*showDialogButton*);  
 showDialogButton.setOnClickListener(v -> new AlertDialog.Builder(MainActivity.this)  
 .setTitle("Діалог")  
 .setMessage("Це приклад AlertDialog.")  
 .setPositiveButton("OK", (dialog, which) -> {  
 // Дія при натисканні OK  
 })  
 .setNegativeButton("Cancel", (dialog, which) -> {  
 // Дія при натисканні Cancel  
 })  
 .show());  
  
 // DatePickerDialog  
 Button showDatePickerButton = findViewById(R.id.*showDatePickerButton*);  
 showDatePickerButton.setOnClickListener(v -> {  
 Calendar calendar = Calendar.*getInstance*();  
 int year = calendar.get(Calendar.*YEAR*);  
 int month = calendar.get(Calendar.*MONTH*);  
 int day = calendar.get(Calendar.*DAY\_OF\_MONTH*);  
  
 DatePickerDialog datePickerDialog = new DatePickerDialog(MainActivity.this,  
 (view, selectedYear, selectedMonth, selectedDay) -> {  
 // Обробка вибраної дати  
 }, year, month, day);  
 datePickerDialog.show();  
 });  
  
 // Custom Dialog  
 Button showCustomDialogButton = findViewById(R.id.*showCustomDialogButton*);  
 showCustomDialogButton.setOnClickListener(v -> {  
 LayoutInflater inflater = getLayoutInflater();  
 View dialogView = inflater.inflate(R.layout.*custom\_dialog*, null);  
  
 new AlertDialog.Builder(MainActivity.this)  
 .setView(dialogView)  
 .setPositiveButton("OK", (dialog, id) -> {  
 // Обробка даних  
 })  
 .setNegativeButton("Cancel", (dialog, id) -> {  
 // Закриття діалогу  
 })  
 .create()  
 .show();  
 });  
  
 // Handler для головного потоку  
 mainHandler = new Handler(Looper.*getMainLooper*()) {  
 @Override  
 public void handleMessage(Message msg) {  
 TextView textView = findViewById(R.id.*handlerMessageTextView*);  
 textView.setText("Message received: " + msg.what);  
 }  
 };  
  
 // Фоновий Handler  
 HandlerThread handlerThread = new HandlerThread("BackgroundThread");  
 handlerThread.start();  
 backgroundHandler = new Handler(handlerThread.getLooper());  
  
 // Запуск Handler з затримкою  
 Button startHandlerButton = findViewById(R.id.*startHandlerButton*);  
 startHandlerButton.setOnClickListener(v -> {  
 // Використання поста з затримкою  
 mainHandler.postDelayed(() -> {  
 TextView textView = findViewById(R.id.*handlerMessageTextView*);  
 textView.setText("Handler executed after delay");  
 }, 5000); // 5 секунди  
  
 // Виконання задачі у фоновому потоці  
 new Thread(() -> {  
 try {  
 Thread.*sleep*(2000); // Імітація довгої задачі  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 // Відправка повідомлення в головний потік  
 mainHandler.post(() -> {  
 TextView textView = findViewById(R.id.*handlerMessageTextView*);  
 textView.setText("Updated from background thread");  
 });  
 }).start();  
  
 // Використання HandlerThread  
 backgroundHandler.post(() -> {  
 try {  
 Thread.*sleep*(6000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 Message msg = mainHandler.obtainMessage();  
 msg.what = 3; // Код повідомлення  
 mainHandler.sendMessage(msg);  
 });  
 });  
 }  
}

ДОДАТОК Б

package ihor.vysotskyi.nure;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.List;  
  
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {  
  
 private List<String> mDataset;  
  
 // ViewHolder зберігає посилання на елементи списку  
 public static class MyViewHolder extends RecyclerView.ViewHolder {  
 public TextView textView;  
  
 public MyViewHolder(View v) {  
 super(v);  
 textView = v.findViewById(R.id.*textView*);  
 }  
 }  
  
 // Конструктор адаптера  
 public MyAdapter(List<String> dataset) {  
 mDataset = dataset;  
 }  
  
 @NonNull  
 @Override  
 public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
 // Створюємо представлення для кожного елемента  
 View v = LayoutInflater.*from*(parent.getContext())  
 .inflate(R.layout.*item\_text\_view*, parent, false);  
 return new MyViewHolder(v);  
 }  
  
 @Override  
 public void onBindViewHolder(MyViewHolder holder, int position) {  
 // Прив’язка даних до елемента  
 holder.textView.setText(mDataset.get(position));  
 }  
  
 @Override  
 public int getItemCount() {  
 return mDataset.size();  
 }  
}

ДОДАТОК В

activity\_main.xml

<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:gravity="center"  
 android:id="@+id/main"  
 android:padding="16dp">  
  
 <Button  
 android:id="@+id/showDialogButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Show AlertDialog" />  
  
 <Button  
 android:id="@+id/showDatePickerButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Show DatePickerDialog" />  
  
 <Button  
 android:id="@+id/showCustomDialogButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Show Custom Dialog" />  
  
 <!-- Новий блок для демонстрації Handler -->  
 <Button  
 android:id="@+id/startHandlerButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="32dp"  
 android:text="Start Handler" />  
  
 <TextView  
 android:id="@+id/handlerMessageTextView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Message will appear here"  
 android:layout\_marginTop="20dp" />  
  
 <!-- RecyclerView -->  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/recyclerView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_marginTop="20dp" />  
  
</LinearLayout>

custom\_dialog.xml

<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"  
 android:paddingBottom="8dp" />  
  
 <EditText  
 android:id="@+id/customDialogInput"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:hint="Введіть текст"  
 android:textSize="18dp"  
 />  
  
</LinearLayout>

item\_text\_view.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="wrap\_content"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="18sp"  
 android:textColor="#000" />  
</LinearLayout>