МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Звіт

з практичної роботи № 3 з дисципліни

Програмування під Android

Виконав: Перевірив:

ст. гр. ПЗПІ-23-4 ст. викладач кафедри ПІ

Цисельський Андрій Сокорчук Ігор Петрович

Харків 2024

ЗАВДАННЯ

Ознайомитися з основними видами діалогових вікон в Android.

Навчитися створювати та використовувати діалогові вікна в Android-застосунках.

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

Зрозуміти концепцію черги повідомлень (Message Queue) в Android.

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

Опанувати створення та використання Handler для асинхронної обробки даних.

Навчитися працювати з компонентом RecyclerView для створення списків з прокручуванням в Android, а також ознайомитися з принципами створення адаптерів (Adapters) і керування даними у списках.

ХІД РОБОТИ

Створюємо новий проєкт та обираємо шаблон Empty Views Activity.

Діалогові панелі (Dialog) – це невеликі вікна, які відображаються поверх поточного інтерфейсу для привернення уваги користувача або запиту додаткових дій.

Основні типи діалогових панелей:

* AlertDialog – вікно для вибору між кількома варіантами або підтвердження дії.
* DatePickerDialog – діалог для вибору дати.
* TimePickerDialog – діалог для вибору часу.
* CustomDialog – діалогове вікно з індивідуальним дизайном.
* ProgressDialog – вікно для відображення прогресу виконання завдань.

Починаємо з AlertDialog:

Додаємо кнопку для виклику діалогового вікна:

1 . <Button

2 . android:id="@+id/alertDialog"

3 . android:layout\_width="0dp"

4 . android:layout\_height="wrap\_content"

5 . android:text="Show Alert dialog"

6 . android:textColor="#FFFFFF"

7 . android:backgroundTint="#6200EE"

8 . android:padding="16dp"

9 . android:textSize="16sp"

10. app:layout\_constraintTop\_toBottomOf="@id/title"

11. app:layout\_constraintStart\_toStartOf="parent"

12. app:layout\_constraintEnd\_toEndOf="parent"

13. app:layout\_constraintBottom\_toTopOf="@id/progressDialog"/>

Ініцалізуємо та підписуємо подію за допомогою setOnClickListener у MainActivity:

1 . alertButton.setOnClickListener(v ->

2 . new AlertDialog.Builder(context)

3 . .setTitle("Test Alert")

4 . .setMessage("This is a test alert dialog")

5 . .setPositiveButton("Accept", (dialog, which) ->

6 . Toast.makeText(context, "Alert confirmed", Toast.LENGTH\_SHORT).show()

7 . )

8 . .setNegativeButton("Decline", (dialog, which) ->

9 . Toast.makeText(context, "Alert declined", Toast.LENGTH\_SHORT).show()

10. )

11. .create()

12. .show()

13. );

Переходимо до DatePickerDialog.

Створюємо кнопку з ідентифікатором date, ініціалізуємо її та додаємо обробник подій за допомогою setOnClickListener із лямбда-виразом.

Для визначення поточної дати використовуємо допоміжний клас Calendar. Діалог забезпечує зручний вибір дати через інтерфейс, схожий на календар:

1 . datePickerButton.setOnClickListener(v -> {

2 . Calendar calendar = Calendar.getInstance();

3 . new DatePickerDialog(context, (view, year, month, dayOfMonth) ->

4 . Toast.makeText(context,

5 . String.format("Selected: %d-%d-%d", dayOfMonth, month + 1, year),

6 . Toast.LENGTH\_SHORT).show(),

7 . calendar.get(Calendar.YEAR),

8 . calendar.get(Calendar.MONTH),

9 . calendar.get(Calendar.DAY\_OF\_MONTH)

10. ).show();

11. });

Переходимо до TimePickerDialog.

Створюємо кнопку з ідентифікатором time, ініціалізуємо її та додаємо обробник подій за допомогою setOnClickListener із лямбда-виразом.

Для визначення поточного часу використовуємо допоміжний клас Calendar. Діалог надає зручний інтерфейс для вибору часу, подібний до звичайного годинника.

1 . timePickerButton.setOnClickListener(v -> {

2 . Calendar calendar = Calendar.getInstance();

3 . new TimePickerDialog(context, (view, hourOfDay, minute) ->

4 . Toast.makeText(context,

5 . String.format("Time: %02d:%02d", hourOfDay, minute),

6 . Toast.LENGTH\_SHORT).show(),

7 . calendar.get(Calendar.HOUR\_OF\_DAY),

8 . calendar.get(Calendar.MINUTE),

9 . true

10. ).show();

11. });

Після підтвердження вибору у діалозі на екрані з'являється спливаюче повідомлення (Toast) з обраними часом або датою.

Переходимо до CustomDialog. На відміну від інших діалогів, він має індивідуальний вигляд, який можна налаштовувати самостійно. Зокрема, для такого діалогу можна створити власну розмітку.

Спочатку створюємо макет dialog\_custom.xml, де додаємо текст «This is a custom dialog» і дві кнопки для закриття діалогу з позитивним або негативним результатом [ДОДАТОК А].

Також необхідно вказати, яку саме розмітку використовуватиме діалог:

1. dialog.setContentView(R.layout.*my\_custom\_dialog*);

Наступним йде ProgressDialog.

1. progressButton.setOnClickListener(v -> {

2. ProgressDialog progressDialog = new ProgressDialog(context);

3. progressDialog.setProgressStyle(ProgressDialog.STYLE\_SPINNER);

4. progressDialog.setMessage("Loading, please wait...");

5. progressDialog.show();

6. });

У коді програми описано створення та налаштування діалогів: їх стилі та текстове наповнення.[ДОДАТОК Б]

Завершивши роботу з діалогами, переходимо до розгляду Handlers.

Handler – це об'єкт, що дозволяє надсилати та обробляти повідомлення або виконувані об'єкти (Runnable) у потоці, з яким він пов’язаний.

MessageQueue – черга, яка містить повідомлення та Runnable, що оброблятимуться об'єктом Handler.

Looper – механізм, який циклічно обробляє чергу повідомлень у потоці.

У практичній роботі було виконано імітацію роботи з потоками, оскільки їхня повна реалізація могла б бути значно складнішою.

Розглянемо приклад виконання дії із затримкою на 3 секунди після натискання кнопки з ідентифікатором handler.

1. findViewById(R.id.handlerDialog).setOnClickListener(v ->

2. new Handler(Looper.getMainLooper()).postDelayed(() ->

3. Toast.makeText(context, "3 seconds passed", Toast.LENGTH\_SHORT).show(),

4. 3000

5. )

6. );

Переходимо до роботи з RecyclerView.

* RecyclerView – основний контейнер для відображення списків або сіток елементів.
* LayoutManager – відповідає за розташування елементів у списку (лінійний, сітковий або власний макет).
* Adapter – адаптер, який перетворює дані для відображення в RecyclerView.
* ViewHolder – зберігає посилання на елементи інтерфейсу для їх повторного використання, оптимізуючи продуктивність.

Почнемо з додавання RecyclerView у файл розмітки:

1 . <androidx.recyclerview.widget.RecyclerView

2 . android:id="@+id/list"

3 . android:layout\_width="0dp"

4 . android:layout\_height="0dp"

5 . tools:listitem="@layout/list\_item"

6 . tools:itemCount="10"

7 . android:layout\_marginStart="4dp"

8 . android:layout\_marginEnd="4dp"

9 . app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

10. app:layout\_constraintTop\_toBottomOf="@id/handlerDialog"

11. app:layout\_constraintStart\_toStartOf="parent"

12. app:layout\_constraintEnd\_toEndOf="parent"

13. app:layout\_constraintBottom\_toBottomOf="parent"/>

Одразу видно, що для відображення компонентів використовується LinearLayoutManager, який розташовує елементи лінійно. Звісно, можна використовувати GridLayoutManager для сіткового відображення або будь-який інший менеджер макетів, створений вручну. Елементи, які відображаються в списку, називаються item.

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

1. private void setupRecyclerView() {

2. ArrayList<String> dataList = new ArrayList<>();

3. for (int i = 0; i < 15; i++) {

4. dataList.add("RecyclerView item #");

5. }

6. ((RecyclerView) findViewById(R.id.list)).setAdapter(new ListItemAdapter(dataList));

7. }

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

Першим кроком створюємо onCreateViewHolder:

1. @NonNull

2. @Override

3. public ListItemViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

4. View itemView = LayoutInflater.from(parent.getContext())

5. .inflate(R.layout.list\_item, parent, false);

6. return new ListItemViewHolder(itemView, items);

7. }

Та створюємо ItemViewHolder:

1 . public static class ListItemViewHolder extends RecyclerView.ViewHolder {

2 . private final TextView idTextView;

3 . private final TextView textTextView;

4 . private final List<String> items;

5 .

6 . public ListItemViewHolder(@NonNull View itemView, List<String> items) {

7 . super(itemView);

8 . this.items = items;

9 . idTextView = itemView.findViewById(R.id.list\_item\_id);

10. textTextView = itemView.findViewById(R.id.list\_item\_text);

11.

12. itemView.setOnClickListener(v -> {

13. int position = getAdapterPosition();

14. if (position != RecyclerView.NO\_POSITION) {

15. Toast.makeText(v.getContext(), "Item " + position + " has been clicked ", Toast.LENGTH\_SHORT).show();

16. }

17. });

18. }

19.

20. public void bindData(int position) {

21. idTextView.setText(String.valueOf(position));

22. textTextView.setText(this.items.get(position) + " " + (position + 1));

23. }

24. }

Цими двома кроками було створено каркас для списку. Тепер елемент знає, що і як потрібно розмістити в блоці списку [ДОДАТОК Г].

ВИСНОВКИ

Під час практичної роботи було створено додаток, який дозволяє ознайомитися з основними типами діалогових панелей та їх атрибутами для налаштування. Також було засвоєно концепцію черги повідомлень і обробки подій за допомогою Handler. Крім того, набуто навичок роботи з компонентом RecyclerView для створення списків з прокручуванням в Android, а також використання адаптерів (Adapters) і керування даними у списках.

Посилання на відеозвіт завантажений на YouTube. URL: <https://youtu.be/mp3_5Yx221w>

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Документація AndroidDevelopers Recycler View. URL: <https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView> (дата звернення: 12.11.2024)
2. Документація AndroidDevelopers handlers. URL: <https://developer.android.com/reference/android/os/Handler> (дата звернення: 12.11.2024)
3. Документація AndroidDevelopers dialogs. URL: <https://developer.android.com/develop/ui/views/components/dialogs> (дата звернення: 12.11.2024)

ДОДАТОК А

Файл розмітки my\_custom\_dialog.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
  
 <TextView  
 android:id="@+id/infotext"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:text="My custom dialog window"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <Button  
 android:id="@+id/okBtn"  
 android:text="ok"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginEnd="20dp"  
 android:layout\_marginBottom="20dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toBottomOf="@id/infotext"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@id/cancelBtn"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 />  
  
 <Button  
 android:id="@+id/cancelBtn"  
 android:text="cancel"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginEnd="20dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toBottomOf="@id/infotext"  
 app:layout\_constraintStart\_toEndOf="@id/okBtn"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

ДОДАТОК Б

Код програми MainActivity.java

package tsyselskyi.andrii.nure;  
  
import android.app.DatePickerDialog;  
import android.app.Dialog;  
import android.app.ProgressDialog;  
import android.app.TimePickerDialog;  
import android.content.Context;  
import android.os.Bundle;  
import android.os.Handler;  
import android.os.Looper;  
import android.widget.Button;  
import android.widget.Toast;  
  
import androidx.activity.EdgeToEdge;  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.graphics.Insets;  
import androidx.core.view.ViewCompat;  
import androidx.core.view.WindowInsetsCompat;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.ArrayList;  
import java.util.Calendar;  
  
public class MainActivity extends AppCompatActivity {  
 private Button alertButton;  
 private Button progressButton;  
 private Button datePickerButton;  
 private Button timePickerButton;  
 private Button customDialogButton;  
 private Context context = this;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_main*);  
  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (view, windowInsets) -> {  
 Insets insets = windowInsets.getInsets(WindowInsetsCompat.Type.*systemBars*());  
 view.setPadding(insets.left, insets.top, insets.right, insets.bottom);  
 return windowInsets;  
 });  
  
 initializeButtons();  
 setupAlertButton();  
 setupProgressDialogButton();  
 setupDatePickerButton();  
 setupTimePickerButton();  
 setupCustomDialogButton();  
 setupHandlerButton();  
 setupRecyclerView();  
 }  
  
 private void initializeButtons() {  
 alertButton = findViewById(R.id.*alertDialog*);  
 progressButton = findViewById(R.id.*progressDialog*);  
 datePickerButton = findViewById(R.id.*dateDialog*);  
 timePickerButton = findViewById(R.id.*timeDialog*);  
 customDialogButton = findViewById(R.id.*customDialog*);  
 }  
  
 private void setupAlertButton() {  
 alertButton.setOnClickListener(v ->  
 new AlertDialog.Builder(context)  
 .setTitle("Test Alert")  
 .setMessage("This is a test alert dialog")  
 .setPositiveButton("Accept", (dialog, which) ->  
 Toast.*makeText*(context, "Alert confirmed", Toast.*LENGTH\_SHORT*).show()  
 )  
 .setNegativeButton("Decline", (dialog, which) ->  
 Toast.*makeText*(context, "Alert declined", Toast.*LENGTH\_SHORT*).show()  
 )  
 .create()  
 .show()  
 );  
 }  
  
 private void setupProgressDialogButton() {  
 progressButton.setOnClickListener(v -> {  
 ProgressDialog progressDialog = new ProgressDialog(context);  
 progressDialog.setProgressStyle(ProgressDialog.*STYLE\_SPINNER*);  
 progressDialog.setMessage("Loading, please wait...");  
 progressDialog.show();  
 });  
 }  
  
 private void setupDatePickerButton() {  
 datePickerButton.setOnClickListener(v -> {  
 Calendar calendar = Calendar.*getInstance*();  
 new DatePickerDialog(context, (view, year, month, dayOfMonth) ->  
 Toast.*makeText*(context,  
 String.*format*("Selected: %d-%d-%d", dayOfMonth, month + 1, year),  
 Toast.*LENGTH\_SHORT*).show(),  
 calendar.get(Calendar.*YEAR*),  
 calendar.get(Calendar.*MONTH*),  
 calendar.get(Calendar.*DAY\_OF\_MONTH*)  
 ).show();  
 });  
 }  
  
 private void setupTimePickerButton() {  
 timePickerButton.setOnClickListener(v -> {  
 Calendar calendar = Calendar.*getInstance*();  
 new TimePickerDialog(context, (view, hourOfDay, minute) ->  
 Toast.*makeText*(context,  
 String.*format*("Time: %02d:%02d", hourOfDay, minute),  
 Toast.*LENGTH\_SHORT*).show(),  
 calendar.get(Calendar.*HOUR\_OF\_DAY*),  
 calendar.get(Calendar.*MINUTE*),  
 true  
 ).show();  
 });  
 }  
  
 private void setupCustomDialogButton() {  
 customDialogButton.setOnClickListener(v -> {  
 Dialog dialog = new Dialog(context);  
 dialog.setContentView(R.layout.*my\_custom\_dialog*);  
 dialog.setCancelable(true);  
 dialog.findViewById(R.id.*okBtn*).setOnClickListener(view -> {  
 dialog.dismiss();  
 Toast.*makeText*(context, "Custom dialog OK", Toast.*LENGTH\_SHORT*).show();  
 });  
 dialog.findViewById(R.id.*cancelBtn*).setOnClickListener(view -> {  
 dialog.dismiss();  
 Toast.*makeText*(context, "Custom dialog Cancel", Toast.*LENGTH\_SHORT*).show();  
 });  
 dialog.show();  
 });  
 }  
  
 private void setupHandlerButton() {  
 findViewById(R.id.*handlerDialog*).setOnClickListener(v ->  
 new Handler(Looper.*getMainLooper*()).postDelayed(() ->  
 Toast.*makeText*(context, "3 seconds passed", Toast.*LENGTH\_SHORT*).show(),  
 3000  
 )  
 );  
 }  
  
 private void setupRecyclerView() {  
 ArrayList<String> dataList = new ArrayList<>();  
 for (int i = 0; i < 100; i++) {  
 dataList.add("RecyclerView item #");  
 }  
 ((RecyclerView) findViewById(R.id.*list*)).setAdapter(new ListItemAdapter(dataList));  
 }  
}

ДОДАТОК В

Код програми ListItemAdapter.java

package tsyselskyi.andrii.nure;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.List;  
  
  
public class ListItemAdapter extends RecyclerView.Adapter<ListItemAdapter.ListItemViewHolder> {  
  
 private final List<String> items;  
  
 public ListItemAdapter(List<String> items) {  
 this.items = items;  
 }  
  
 @NonNull  
 @Override  
 public ListItemViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View itemView = LayoutInflater.*from*(parent.getContext())  
 .inflate(R.layout.*list\_item*, parent, false);  
 return new ListItemViewHolder(itemView, items);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ListItemViewHolder holder, int position) {  
 holder.bindData(position);  
 }  
  
 @Override  
 public int getItemViewType(int position) {  
 return position % 2 == 0 ? 0 : 1;  
 }  
  
 @Override  
 public int getItemCount() {  
 return items.size();  
 }  
  
  
 public static class ListItemViewHolder extends RecyclerView.ViewHolder {  
 private final TextView idTextView;  
 private final TextView textTextView;  
 private final List<String> items;  
  
 public ListItemViewHolder(@NonNull View itemView, List<String> items) {  
 super(itemView);  
 this.items = items;  
 idTextView = itemView.findViewById(R.id.*list\_item\_id*);  
 textTextView = itemView.findViewById(R.id.*list\_item\_text*);  
  
 itemView.setOnClickListener(v -> {  
 int position = getAdapterPosition();  
 if (position != RecyclerView.*NO\_POSITION*) {  
 Toast.*makeText*(v.getContext(), "Item " + position + " has been clicked ", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
  
 public void bindData(int position) {  
 idTextView.setText(String.*valueOf*(position));  
 textTextView.setText(this.items.get(position) + " " + (position + 1));  
 }  
 }  
}

ДОДАТОК Г

Файл розмітки list\_item.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="16dp"  
 android:layout\_marginTop="16dp"  
 android:background="@drawable/background\_shape"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <ImageView  
 android:id="@+id/item\_image"  
 android:layout\_width="120dp"  
 android:layout\_height="120dp"  
 android:src="@drawable/myimg"  
 android:layout\_marginBottom="8dp"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"/>  
  
 <TextView  
 android:id="@+id/list\_item\_id"  
 tools:text="10"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:textColor="#FFFFFF"  
 android:textSize="24sp"  
 android:textStyle="bold"  
 android:layout\_marginBottom="8dp"  
 android:gravity="center"  
 app:layout\_constraintTop\_toBottomOf="@id/item\_image"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintBottom\_toTopOf="@id/list\_item\_text" />  
  
 <TextView  
 android:id="@+id/list\_item\_text"  
 tools:text="Item Id: 10"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:textColor="#FFFFFF"  
 android:textSize="18sp"  
 android:gravity="center"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginBottom="16dp"  
 app:layout\_constraintTop\_toBottomOf="@id/list\_item\_id"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintBottom\_toBottomOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>