**ЛЕКЦІЯ 13. ДІАЛОГОВІ ВІКНА**

**13.1 DatePickerDialog і TimePickerDialog**

В Android існують два діалогових вікна – *DatePickerDialog* і *TimePickerDialog* , які дозволяють вибрати дату і час. Крім установки дати *DatePickerDialog* дозволяє обробити вибір дати за допомогою слухачів *OnDateChangedListener* і *OnDateSetListener*, що дозволяє використовувати обрану дату далі в застосунку. Подібним чином *TimePickerDialog* дозволяє обробити вибір часу за допомогою слухачів *OnTimeChangedListener* і *OnTimeSetListener*.

При роботі з даними компонентами треба враховувати, що відлік місяців в *DatePickerDialog* починається з нуля, тобто січень буде мати номер 0, а грудень – 11. І аналогічно в *TimePickerDialog* відлік секунд і хвилин буде йти з 0 до 59, а годин – з 0 до 23.

Використаємо *DatePickerDialog* і *TimePickerDialog* в додатку. Визначимо наступну розмітку інтерфейсу в *activity\_main.xml*:

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="vertical">

<TextView android:id=["@+id](mailto:@+id)/currentDateTime" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:textSize="18sp" />

<Button android:id=["@+id/timeButton" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Изменить время" android:onClick="setTime" />](mailto:@+id)

<Button android:id=["@+id/dateButton" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Изменить дату" android:onClick="setDate" />](mailto:@+id)

</LinearLayout>

Тут описані дві кнопки для вибору дати і часу та текстове поле, яке відображає вибрані дату і час. І змінимо код *MainActivity*:

public class MainActivity extends AppCompatActivity { TextView currentDateTime;

Calendar dateAndTime=Calendar.getInstance();

[@Override](mailto:@Override)

public void onCreate(Bundle savedInstance) { super.onCreate(savedInstance); setContentView(R.layout.activity\_main); currentDateTime=(TextView)findViewById(R.id.currentDateTime);

setInitialDateTime();

}

// показуємо діалоговое вікно для вибору дати

public void setDate(View v) {

new DatePickerDialog(MainActivity.this, d, dateAndTime.get(Calendar.YEAR), dateAndTime.get(Calendar.MONTH), dateAndTime.get(Calendar.DAY\_OF\_MONTH))

.show();

}

// показуємо діалоговое вікно для вибору часу

public void setTime(View v) {

new TimePickerDialog(MainActivity.this, t, dateAndTime.get(Calendar.HOUR\_OF\_DAY), dateAndTime.get(Calendar.MINUTE), true)

.show();

}

// встановлення початкових дати і часу

private void setInitialDateTime() {

currentDateTime.setText(DateUtils.formatDateTime(this, dateAndTime.getTimeInMillis(),

DateUtils.FORMAT\_SHOW\_DATE | DateUtils.FORMAT\_SHOW\_YEAR

| DateUtils.FORMAT\_SHOW\_TIME));

}

// обробник вибору часу

TimePickerDialog.OnTimeSetListener t=new TimePickerDialog.OnTimeSetListener() {

public void onTimeSet(TimePicker view, int hourOfDay, int minute) { dateAndTime.set(Calendar.HOUR\_OF\_DAY, hourOfDay); dateAndTime.set(Calendar.MINUTE, minute);

setInitialDateTime();

}

};

// обробник вибору дати

DatePickerDialog.OnDateSetListener d=new DatePickerDialog.OnDateSetListener() {

public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {

dateAndTime.set(Calendar.YEAR, year); dateAndTime.set(Calendar.MONTH, monthOfYear); dateAndTime.set(Calendar.DAY\_OF\_MONTH, dayOfMonth);

setInitialDateTime();

}

};

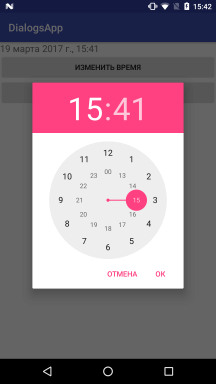
}

Ключовим класом тут є *java.util.Calendar*, який зберігається в стандартній

бібліотеці класів Java. У методі *setInitialDateTime*() ми отримуємо з примірника цього класу кількість мілісекунд *dateAndTime*.*getTimeInMillis*() і за допомогою форматування виводимо на текстове поле.

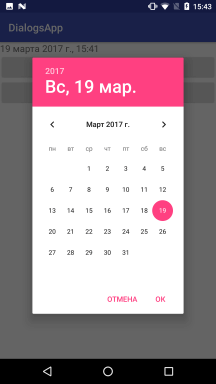
Метод *setDate*(), що викликається після натискання на кнопку, відображає вікно для вибору дати. При створенні вікна його об’єкту передається обробник вибору дати *DatePickerDialog*.*OnDateSetListener*, який змінює дату на текстовому полі.

Аналогічно метод *setTime*() відображає вікно для вибору часу. Об’єкт вікна використовує обробник вибору часу *TimePickerDialog*.*OnTimeSetListener*, який змінює час на текстовому полі. Після запуску додатку, натиснувши на кнопку зміни часу, ми зможемо встановити час:



Для встановлення часу у діалоговому вікні визначена кнопка «Встановити».

Аналогічно працює вікно встановлення дати:



**13.2 DialogFragment і створення діалогових вікон**

Для створення діалогових вікон використовується компонент *AlertDialog* в зв’язці з класом фрагмента *DialogFragment*. Розглянемо їх застосування. Спочатку додамо в проект новий клас фрагмента, який назвемо *CustomDialogFragment*:

public class CustomDialogFragment extends DialogFragment {

[@NonNull](mailto:@NonNull)

public Dialog onCreateDialog(Bundle savedInstanceState) { AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());

return builder.setTitle("Діалоговое вікно").setMessage("Для закриття вікна натисніть

ОК").create();

}

}

Клас фрагмента містить всю стандартну функціональність фрагмента з його

життєвим циклом, але при цьому успадковується від класу *DialogFragment*, який додає ряд додаткових функцій. І для його створення ми можемо використати два способи:

● перевизначення методу *onCreateDialog*(), який повертає об’єкт *Dialog*;

● використання стандартного методу *onCreateView*().

Для створення діалогового вікна в методі *onCreateDialog*() застосовується клас *AlertDialog.Builder*. За допомогою своїх методів він дозволяє налаштувати відображення діалогового вікна:

● *setTitle()*: встановлює заголовок вікна;

● *setView()*: встановлює розмітку інтерфейсу вікна;

● *setIcon()*: встановлює іконку вікна;

● *setPositiveButton()*: встановлює кнопку підтвердження дії;

● *setNeutralButton()*: встановлює «нейтральну» кнопку, дія якої може відрізнятися від дій підтвердження або скасування;

● *setNegativeButton()*: встановлює кнопку скасування;

● *setMessage()*: встановлює текст діалогового вікна, але при використанні setView даний метод необов’язковий або може розглядатися в якості альтернативи, якщо нам треба просто вивести повідомлення;

● *create()*: створює вікно.

В даному ж випадку діаговое кно просто виводить деякий повідомлення.

Для виклику цього діалогового вікна в файлі *activity\_main.xml* визначимо кнопку:

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="vertical">

<Button

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Dialog" android:onClick="showDialog"/>

</LinearLayout>

У коді *MainActivity* визначимо обробник натискання кнопки, який буде запускати діалогове вікно:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

public void onCreate(Bundle savedInstance) {

super.onCreate(savedInstance);

setContentView(R.layout.activity\_main);

}

public void showDialog(View v) {

CustomDialogFragment dialog=new CustomDialogFragment();

dialog.show(getSupportFragmentManager(), "custom");

}

}

Для виклику діалогового вікна створюється об’єкт фрагмента

*CustomDialogFragment*, потім у нього викликається метод *show*(). У цей метод передається менеджер фрагментів *FragmentManager* і рядок – довільний тег.

І поле натискання на кнопку ми зможемо ввести дані в діалогове вікно:



Тепер трохи «кастомізуємо» діалогове вікно:

public Dialog onCreateDialog(Bundle savedInstanceState) { AlertDialog.Builder builder=new AlertDialog.Builder(getActivity()); return builder

.setTitle("Діалоговое вікно")

.setIcon(android.R.drawable.ic\_dialog\_alert)

.setMessage("Для закриття вікна натисніть ОК")

.setPositiveButton("OK", null)

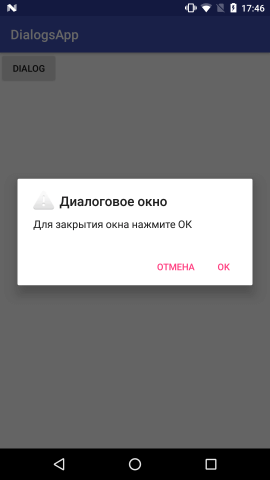
.setNegativeButton("Відмінити", null)

.create();

}

Тут добавляєтьcя іконка, яка в якості зображення використовує вбудований

ресурс *android.R.drawable.ic\_dialog\_alert* і встановлюються дві кнопки. Для кожної кнопки можна встановити текст і обробник натискання. В даному випадку для обробника натискання передається *null*, тобто обробник не встановлено. В результаті отримаємо:



Тепер додамо в папку *res/layout* новий файл *dialog.xml*:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:orientation="vertical"

android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<TextView android:gravity="center" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Hello Android 7"/>

</LinearLayout>

І змінимо код створення діалогового вікна:

public Dialog onCreateDialog(Bundle savedInstanceState) { AlertDialog.Builder builder=new AlertDialog.Builder(getActivity()); return builder

.setTitle("Диалоговое окно")

.setIcon(android.R.drawable.ic\_dialog\_alert)

.setView(R.layout.dialog)

.setPositiveButton("OK", null)

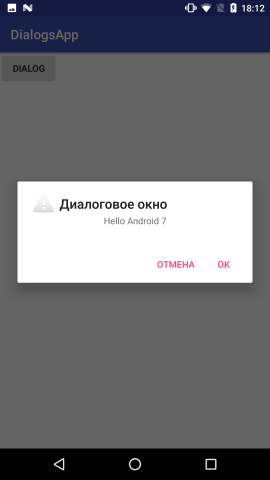
.setNegativeButton("Отмена", null)

.create();

}

Метод *setView*() встановлює в якості інтерфейсу вікна раніше доданий ресурс

*dialog.xml*. При використанні цього методу треба враховувати, що він доступний починаючи з API 21 (*Lollipop*). Тому для його застосування може знадобитися змінити мінімальну версію Android у проекті до 21.



При цьому, як можна побачити на скріншоті, кнопки і заголовок з іконкою не входять до розмітки.

**13.3 Передача даних в діалогове вікно**

Передача даних в діаговое вікно, як і в будь-який фрагмент, здійснюється за допомогою об’єкта *Bundle*. Так, визначимо в файлі *activity\_main.xml* список *ListView*:

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="vertical">

<ListView android:id=["@+id](mailto:@+id)/phonesList" android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

</LinearLayout>

У класі *MainActivity* визначимо для цього списку дані:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

public void onCreate(Bundle savedInstance) { super.onCreate(savedInstance); setContentView(R.layout.activity\_main);

ListView phonesList=(ListView) findViewById(R.id.phonesList); ArrayList<String> phones=new ArrayList<>(); phones.add("Google Pixel");

phones.add("Huawei P9"); phones.add("LG G5"); phones.add("Samsung Galaxy S8");

final ArrayAdapter<String> adapter=new ArrayAdapter<String>(this, android.R.layout.simple\_list\_item\_1, phones);

phonesList.setAdapter(adapter);

phonesList.setOnItemClickListener(new AdapterView.OnItemClickListener() {

[@Override](mailto:@Override)

public void onItemClick(AdapterView<?> parent, View view, int position, long id) { String selectedPhone=adapter.getItem(position);

CustomDialogFragment dialog=new CustomDialogFragment(); Bundle args=new Bundle();

args.putString("phone", selectedPhone);

dialog.setArguments(args);

dialog.show(getSupportFragmentManager(), "custom");

}

});

}

}

У обробнику події натиснення на елемент в списку отримуємо обраний елемент і

додаємо його в об’єкт *Bundle*. Далі через метод *dialog.setArguments()* передаємо дані з

*Bundle* під фрагмент. Тепер визначимо наступний клас фрагмента

*CustomDialogFragment*:

public class CustomDialogFragment extends DialogFragment {

[@NonNull](mailto:@NonNull)

public Dialog onCreateDialog(Bundle savedInstanceState) {

String phone=getArguments().getString("phone"); AlertDialog.Builder builder=new AlertDialog.Builder(getActivity()); return builder

.setTitle("Діалогове вікно")

.setIcon(android.R.drawable.ic\_dialog\_alert)

.setMessage("Бажаєте видалити"+phone+"?")

.setPositiveButton("OK", null)

.setNegativeButton("Відмінити", null)

.create();

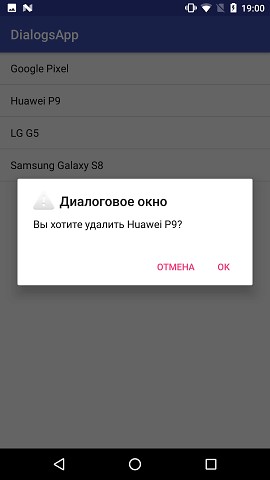
}

}

За допомогою методу *getArguments*() отримуємо переданий в *MainActivity* об’єкт

*Bundle*. І оскільки був переданий рядок, то для його отримання застосовується метод

*getString*(). При натисканні елемент списку буде передаватися в діалогове вікно:



**13.4 Взаємодія з Activity**

Взаємодія між *Activity* і фрагментом проводиться, як правило, через інтерфейс. Наприклад, в минулій темі *MainActivity* виводила список об’єктів, і тепер визначимо видалення з цього списку через діалогове вікно. Для цього додамо в проект інтерфейс *Datable*:

public interface Datable {

void remove(String name);

}

Єдиний метод інтерфейсу *remove()* отримує об’єкт, який видаляється, у вигляді

параметра *name*. Тепер реалізуємо цей інтерфейс в коді *MainActivity*:

public class MainActivity extends AppCompatActivity implements Datable{

private ArrayList<String> phones;

private ArrayAdapter<String> adapter;

[@Override](mailto:@Override)

public void onCreate(Bundle savedInstance) { super.onCreate(savedInstance); setContentView(R.layout.activity\_main);

ListView phonesList=(ListView) findViewById(R.id.phonesList);

phones=new ArrayList<>(); phones.add("Google Pixel"); phones.add("Huawei P9"); phones.add("LG G5"); phones.add("Samsung Galaxy S8");

adapter = new ArrayAdapter<String>(this, android.R.layout.simple\_list\_item\_1, phones);

phonesList.setAdapter(adapter);

phonesList.setOnItemClickListener(new AdapterView.OnItemClickListener() {

[@Override](mailto:@Override)

public void onItemClick(AdapterView<?> parent, View view, int position, long id) { String selectedPhone=adapter.getItem(position);

CustomDialogFragment dialog=new CustomDialogFragment(); Bundle args=new Bundle();

}

});

}

args.putString("phone", selectedPhone); dialog.setArguments(args); dialog.show(getSupportFragmentManager(), "custom");

[@Override](mailto:@Override)

public void remove(String name) {

adapter.remove(name);

}

}

Метод *remove* видаляє з адаптера переданий елемент. Файл *activity\_main.xml* як і

раніше визначає лише елемент *ListView*:

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="vertical">

<ListView android:id=["@+id](mailto:@+id)/phonesList" android:layout\_width="match\_parent" android:layout\_height="match\_parent" />

</LinearLayout>

І нарешті визначимо фрагмент *CustomDialogFragment*:

public class CustomDialogFragment extends DialogFragment {

private Datable datable;

[@Override](mailto:@Override)

public void onAttach(Context context){

super.onAttach(context);

datable = (Datable) context;

}

[@NonNull](mailto:@NonNull)

public Dialog onCreateDialog(Bundle savedInstanceState) { final String phone=getArguments().getString("phone"); AlertDialog.Builder builder=new AlertDialog.Builder(getActivity()); return builder

.setTitle("Діалогове вікно")

.setIcon(android.R.drawable.ic\_dialog\_alert)

.setMessage("Бажаєте видалити "+phone+"?")

.setPositiveButton("OK", new DialogInterface.OnClickListener() {

[@Override](mailto:@Override)

public void onClick(DialogInterface dialog, int which) {

datable.remove(phone);

}

})

.setNegativeButton("Отмена", null)

.create();

}

}

Метод *onAttach*() викликається на початку життєвого циклу фрагмента, і саме

тут ми можемо отримати контекст фрагмента, в якості якого виступає клас *MainActivity*. Так як *MainActivity* реалізує інтерфейс *Datable*, то ми можемо перетворити контекст до даного інтерфейсу. Потім в обробнику кнопки ОК викликається метод remove об’єкта *Datable*, який видаляє переданий у фрагмент об’єкта *phone*.