Лабораторная работа 3. Разработка приложений с несколькими Activity. Передача данных между Activity

Цель – формирование знаний о жизненном цикле Activity, а также формирование навыков создания и вызова нового Activity и передачи данных между ними.

Формируемые навыки:

— Создание и вызов нового Activity.
— Передача данных между Activity.
— Хранение данных с помощью SharedPreferences.
— Создание всплывающих сообщений и логирование.

Задание. Разработать приложение с двумя Activity для хранения данных о составе группы 131-ПИо. Требования к ПП:

— Основной Activity должен содержать вводную информацию, кнопку для добавления обучающегося и итоговый список группы.
— Вспомогательный Activity должен содержать форму для ввода информации об обучающемся, а именно:
  - ФИО студента.
  - Пол.
  - Предпочитаемые ЯП.
  - Предпочитаемые IDE.
— Режим вызова вспомогательного Activity должен быть двусторонним с получением результатов.
— Передачу данных между Activity осуществлять с помощью SharedPreferences API.
— Предусмотреть смену ориентации экрана.
— Приемлемые контейнеры: LinearLayout, TableLayout.

Теоретическая справка

Всплывающие окна

Всплывающие сообщения организуются с помощью класса Toast:

```
Toast.makeText(context, text, duration).show();
```

– context – объект, который предоставляет доступ к базовым функциям приложения, т.к. Activity является подклассом Context, то в качестве объекта от Context используется текущее Activity, т.е. this.

– text – текст, который надо вывести.

– duration – продолжительность показа (Toast.LENGTH_LONG – длинная (3,5 сек), Toast.LENGTH_SHORT – короткая (2 сек)).

По умолчанию всплывающее сообщение отображается в нижней части рабочего окна по центру. Для изменения места расположения всплывающего сообщения используется метод setGravity:

```
toast.setGravity(Gravity.CENTER, 0, 0);
```

Жизненный цикл Activity

Созданное при работе приложения Activity может быть в одном из трех состояний:

1. Resumed: Activity видно на экране, оно находится в фокусе, пользователь может с ним взаимодействовать. Это состояние также иногда называют Running.

2. Paused: Activity не в фокусе, пользователь не может с ним взаимодействовать, но его видно (оно перекрыто другим Activity, которое занимает не весь экран или полупрозрачно).

3. Stopped: Activity не видно (полностью перекрывается другим Activity), соответственно оно не в фокусе и пользователь не может с ним взаимодействовать.

Когда Activity переходит из одного состояния в другое, система автоматически вызывает соответствующие методы, в которые можно добавлять свой код. Методы Activity, которые вызывает система:

– onCreate() – вызывается при первом создании Activity.
– onStart() – вызывается перед тем, как Activity будет видно пользователю.
– onResume() – вызывается перед тем, как будет доступно пользователю.
– onPause() – вызывается перед тем, как будет показано другое Activity.
– onStop() – вызывается когда Activity становится не видно пользователю.
– onDestroy() – вызывается перед тем, как Activity будет уничтожено.

Важно! Сами методы НЕ вызывают смену состояния. Наоборот, смена состояния Activity является триггером, который вызывает эти методы.

Для того, чтобы в ответ на происходящие изменения состояния Activity отработал требуемый код, необходимо переопределить вышеописанные методы жизненного цикла Activity (вызов и вставка в код переопределяемого метода производится с помощью комбинации клавиш Ctrl + O).

Создание нового Activity

Создать новый Activity можно либо вручную, либо через встроенную функцию добавления нового компонента.

Создание нового Activity вручную:

1. Создать новый xml-файл разметки.
2. Создать новый java-класс для Activity.
3. Прописать логику Activity (наследование от суперкласса с подключением графической библиотеки):

```
public class SecondActivity extends ActionBarActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_second);
} }
```

4. Зарегистрировать новое Activity в манифест-файле:

```
<activity
android:name=".MainActivity"
android:label="@string/app_name">
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>

<activity
android:name=".SecondActivity"
android:label="@string/app_name">
</activity>
```

Создание нового Activity через встроенную функцию добавления нового компонента реализуется с помощью меню File/New или нажатием Alt+Insert. Далее необходимо заполнить форму кастомизации нового Activity с указанием имен xml-файла и java-файла. Запись в манифест-файле создастся автоматически.

Явный вызов нового Ativity осуществляется через создания намерения Intent:

```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```

Неявный вызов нового Ativity работает с настройками Intent Filter искомого Activity в манифест-файле. Intent Filter включает ряд параметров (action, data, category), комбинация которых определяет желаемую цель (отправка письма, открытие гиперссылки, редактирование текста, просмотр картинки, звонок по определенному номеру и т.д.).

```
<activity
android:name=".SecondActivity"
android:label="@string/app_name">

<intent-filter>
    <action android:name="android.intent.action.SecondActivity" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>

</activity>
```

Тогда неявно вызвать SecondActivity из MainActivity можно следующим образом:

```
intent = new Intent("android.intent.action.SecondActivity");
startActivity(intent);
```

Двусторонний запуск Activity с получением результата

Используется метод startActivityForResult() вместо startActivity():

1. Вызвать из исходного Ativity новое Ativity с помощью метода startAtivityForResult (Intent intent1, int requestCode), где requestCode – это идентификатор для определения, с какого Ativity пришел результат.

2. Реализовать в новом Ativity метод setResult(int resultCode, Intent intent1), где resultCode – код возврата.

3. В исходном Activity в методе onActivityResult прописать логику обработки результата, который придет из нового Activity. Синтаксис метода onActivityResult:

```
protected void onActivityResult(int requestCode, int resultCode, Intent data);
```

где:

– requestCode – тот же идентификатор, что и в startActivityForResult. По нему определяется, с какого Activity пришел результат.

– resultCode – код возврата.

– data – Intent, в котором возвращаются данные.

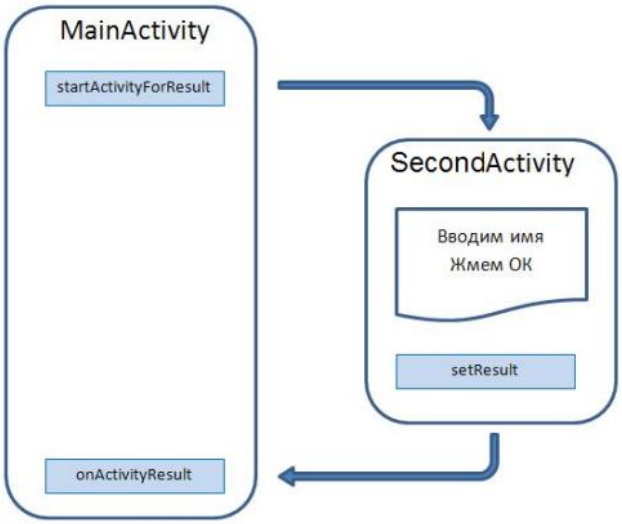

Рисунок 1. Механизм вызова нового Activity с возвратом результатов

Передача данных между Activity

Передача данных между Activity осуществляется с помощью класса Intent в виде пар ключ-значение. Сохранение данных в Intent реализуется с помощью метода putExtra(), а извлечение значения по ключу – getExtra()[1]. Чтобы сохранить пользовательский тип данных используется метод putSerializable(), а извлечение – getSerializable().

Для хранения относительно небольшой коллекции данных, как правило, используется SharedPreferences API – он указывает на файл, содержащий пары ключ-значение, и содержит в себе методы для их чтения и записи.

Сохранение данных:

```
pref = getPreferences(MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();
editor.putString(fio, EditText1.getText().toString());
editor.commit();
```

Чтение данных:

```
pref = getPreferences(MODE_PRIVATE);
myEditText.setText(pref.getString(fio, ""));
```

---

[1] Перечень всех типов данных помимо String доступен по ссылке https://developer.android.com/reference/android/content/Intent#pubmethods