Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №3**

**Дисциплина**: Проектирование мобильных приложений

**Тема**: Lifecycle компоненты. Навигация в приложении

Выполнил студент гр. 3530901/70203 И.Д. Иванов

(подпись)

Преподаватель И.С. Егорова

(подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург

2019

1. Цели

1.1 Ознакомиться с методом обработки жизненного цикла activity/fragment при помощи Lifecycle-Aware компонентов

1.2 Изучить основные возможности навигации внутри приложения: созданиеновых activity, navigation graph

Вариант 6.

Задание для задачи 2: Option Menu.

Исходный код решений всех представленных задач находится по адресу: https://github.com/EndurableFawn/androidLabs

2. Программа работы

2.1. Задача 1. Обработка жизненного цикла с помощью Lifecycle-Aware компонентов.

ViewModel - класс, позволяющий Activity и фрагментам сохранять необходимые им объекты живыми в течение всего жизненного цикла.

LiveData - хранилище данных, работающее по принципу паттерна Observer (наблюдатель). В это хранилище можно поместить какой-либо объект, а также на него можно подписаться, чтобы получать помещаемые в хранилище объекты.

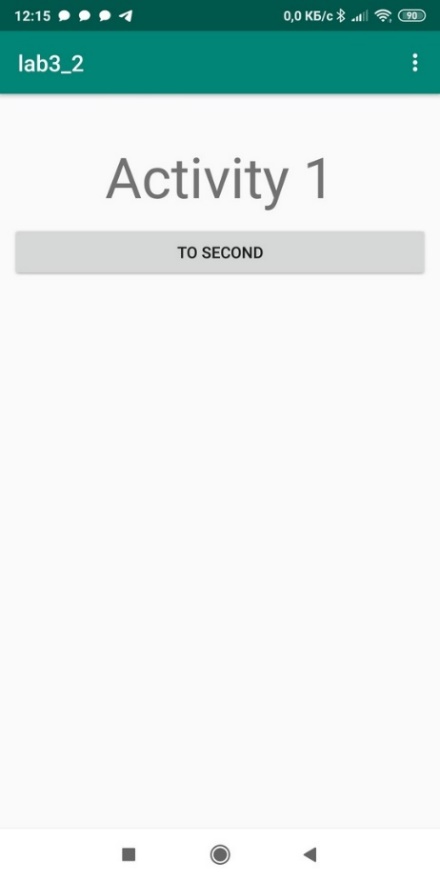
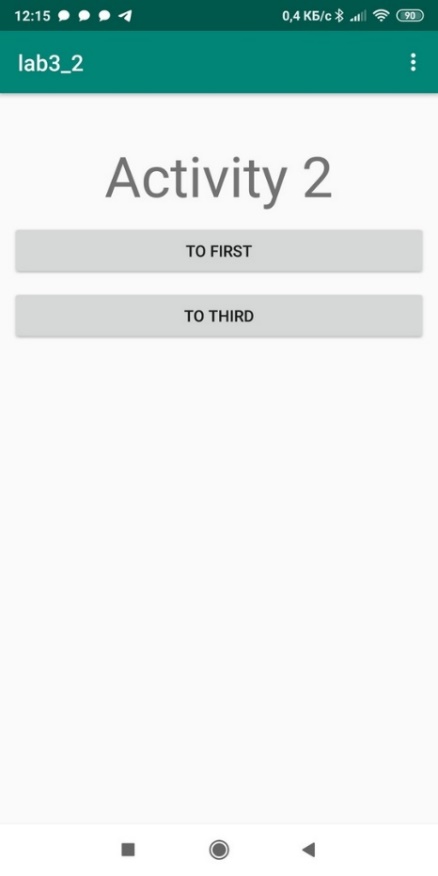
Lifecycle — класс, который хранит информацию про состояние жизненного цикла и разрешает другим объектам отслеживать его c помощью реализации LifecycleObserver. Состоит из методов: addObserver(LifecycleObserver), removeObserver(LifecycleObserver) и getCurrentState().

Был выполнен codelabs со страницы https://developer.android.com/topic/libraries/architecture/lifecycle

2.2. Задача 2. Навигация (startActivityForResult).

Реализуйте навигацию между экранами одного приложения согласно изображению ниже с помощью Activity, Intent и метода startActivityForResult.

Для выполнения задания были созданы 4 activity, внешний вид которых представлен на рис. 2.2.1-2.2.4

   
Рис. 2.2.1 Рис. 2.2.2

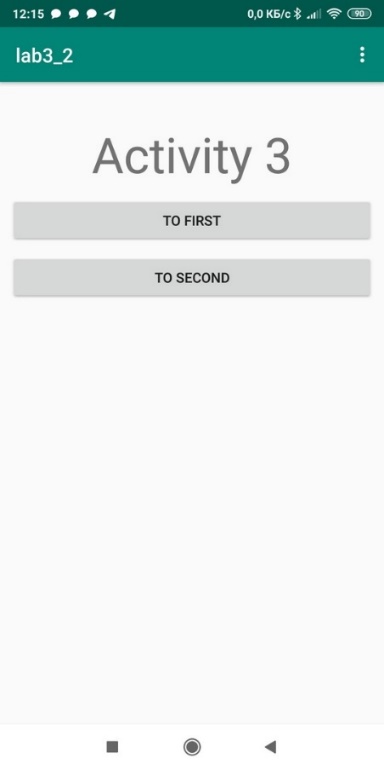
 

Рис. 2.2.3 Рис. 2.2.4

* Приложение начинает работу с отображения Activity 1;
* Кнопка 'to first' отображает на экране Activity 1;
* Кнопка 'to second' отображает на экране Activity 2;
* Кнопка 'to third' отображает на экране Activity 3;
* В любой момент в backstack любого task приложения не содержится более 4 activity;
* Во всех вариантах Activity 'About' доступна из любой другой Activity с помощью Option Menu.

Метод startActivityForResult (в качестве параметров передается intent, code) используется, когда нужно запустить другую activity и получить результат, который необходимо обработать в методе onActivityResult. В новой activity передаваемый результат определяется методом setResult.

В данном задании обработка результатов помогает завершать activity, чтобы в любой момент в backstack приложения было не более 4 activity. Например, если из activity 2 перейти в activity 1, то activity 2 завершается; при переходе из activity 3 в activity 1, завершаются activity 2 и activity 3 (activity 2 не пересоздаётся при переходе из activity 3 в activity 2, в этом случае просто завершается activity 3, но должна завершиться при переходе из activity 3 в activity 1. Узнаёт activity 2 о том, должна ли она завершиться, как раз благодаря результату, который предоставляет activity 3).

2.3. Задача 3. Навигация (флаги Intent/атрибуты Activity).

Решите предыдущую задачу с помощью Activity, Intent и флагов Intent либо атрибутов Activity.

Изменения: вместо метода startActivityForResult используется метод startActivity, которому в качестве параметра передается только intent.

К передаваемому intent-у можно добавлять различные флаги, например, REORDER\_TO\_FRONT. Этот флаг работает следующим образом: если activity уже создано и находится в backstack приложения, то при новом запуске этого же activity уже существующий экземпляр просто помещается на вершину стека, новый экземпляр при этом не создаётся.

2.4. Задача 4. Навигация (флаги Intent/атрибуты Activity).

Дополните граф навигации новым(-и) переходом(-ами) с целью демонстрации какого-нибудь (на свое усмотрение) атрибута Activity или флага Intent, который еще не использовался для решения задачи. Поясните пример и работу флага/атрибута.

Дополнение графа: в activity 1 была добавлена кнопка для перехода в activity 2 с флагом NO\_HISTORY (рис.2.4.1).

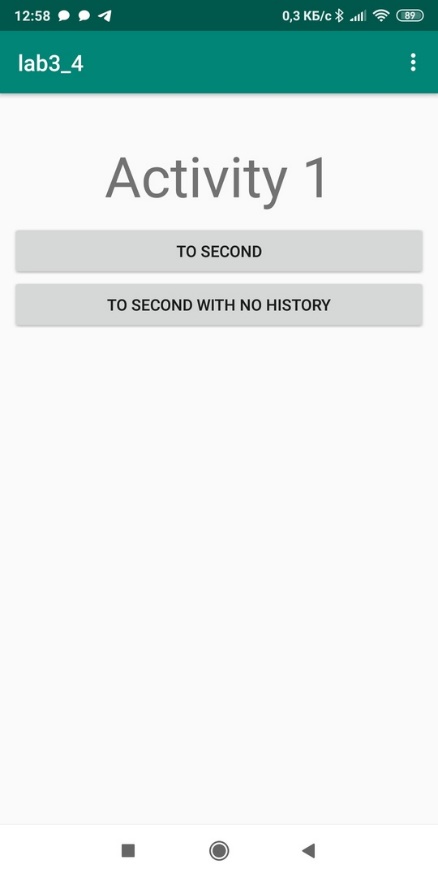


Рис. 2.4.1.

Флаг NO\_HISTORY работает следующим образом: если запустить с ним activity, то запущенное activity не сохранится в backstack приложения. Пример: нажав на кнопку to second with no history, пользователь попадет в activity 2. Перейдя из activity 2 в activity 3 и нажав кнопку «назад», пользователь окажется в activity 1.

2.5. Задача 5. Навигация (Fragments, Navigation Graph).

Решите предыдущую задачу (с расширенным графом) с использованием navigation graph. Все Activity должны быть заменены на фрагменты, кроме Activity 'About', которая должна остаться самостоятельной Activity.

Созданный navigation graph показан на Рис. 2.5.1.

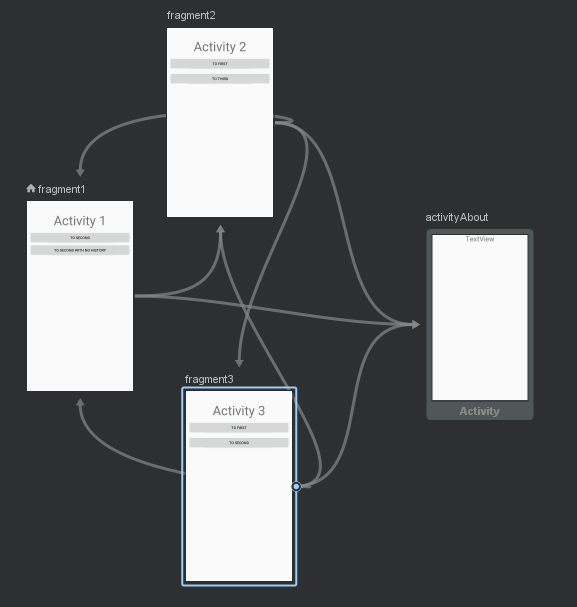


Рис.2.5.1.

NavHostFragment добавлен к mainActivity в xml-файле (Рис. 2.5.2)

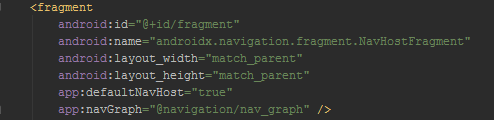


Рис.2.5.2.

android:name - имя класса NavHost реализации.

app:navGraph ассоциирует NavHostFragment с созданным navigation graph.

app:defaultNavHost="true" гарантирует, что NavHostFragment перехватывает системная кнопка «назад».

Переход между фрагментами осуществляется с помощью navСontroller-а.

**3. Выводы**

В результаты выполнения лабораторной работы рассмотрен метод обработки жизненного цикла activity/fragment при помощи Lifecycle-Aware компонентов, изучены основные возможности навигации внутри приложения: создание новых activity, navigation graph.