**Лабораторный практикум по дисциплине   
«Разработка мобильных приложений»**

**Составитель: Тюньков Дмитрий Александрович**

**Лабораторная работа № 1. Intro.**

1. Установить необходимый софт.
2. Создать Git репозиторий.
3. Добавить правильный .gitignore
4. Создать проект с Empty Activity.
5. Запустить проект.
6. Создать Blank Fragment.
7. В нашей Activity добавить фрагмент в контейнер.
8. Добавить во фрагмент кнопочек, текстовых полей.
9. \*Все строковые ресурсы выносить в string.xml.
10. \*Используя ButterKnife + Zelezny Butterknife связать разметку и код.
11. \*Сделать какие-нибудь обработчики кнопок, которые будут изменять текст полей и показывать Toast’ы.

*Задания со \* дополнительные, на максимальный балл.*

Коммитов в репозиторий должно быть не меньше 4, так как коммиты должны быть атомарны. Плюс будет видна последовательность действий, и вы лучше её запомните.

**Лабораторная работа № 2. Навигация.**

Следует помнить, что если у вас многоэкранное приложение с фрагментами, то вам необходимо осуществлять коммуникацию между активити и фрагментом (или между фрагментом и хост-фрагментом). Как это сделать:

1. Прямое обращение к хосту

2. Хост должен имплементить коллбэки фрагмента

3. Шина событий (otto events, green robot EventBus3)

и тут важно понимать, что 1 – нарушение SRP и DI (SOLID модель, первый и последний принципы), 2 – иногда бывает трудно реализовать, 3 – нужно понять

В демо приложении:

<https://github.com/Pasenchuk/TT-Android-School-2016/>

Есть примеры реализации всех трёх подходов с преобладанием первого (так не нужно делать, делалось по причине скорости выполнения).

Теперь задание. Используя ваши навыки исследователей, решите следующие задачи:

1. Коллбэки от фрагмента к активити (google way) – сделайте внутри фрагмента inner interface с методами (число методов = числу кнопок при нажатии на которые должно открыться новое активити/фрагмент). Активити-хост должно имплементить этот интерфейс. На аттаче у фрагмента нужно проверить, имплементит ли активити этот интерфейс и если да – использовать активити, как коллбэк. Пример – в андроид студио при создании фрагмента поставьте галочку на add interface callbacks. Или демо приложение смотрите.

2. Активити стэк и интент флаги – добиться желаемого поведения в открытии активити комбинацией флагов. Создайте несколько фрагментов и открывайте их и возвращайтесь назад аппаратной кнопкой согласно придуманной бизнес-задаче. Бизнес-задачу опишите. Новые активити или фрагмент транзакции запускайте в активити. Коммуникация через шину событий или через коллбэки.

3. Фрагменты и бэкстек. Создайте несколько фрагментов и открывайте их и возвращайтесь назад аппаратной кнопкой согласно придуманной бизнес-задаче. Бизнес-задачу опишите. Новые активити или фрагмент транзакции запускайте в активити. Коммуникация через шину событий или через коллбэки.

4. BaseActivity/BaseFragment паттерн. Пишите туда общий код, а также регистрацию на шину

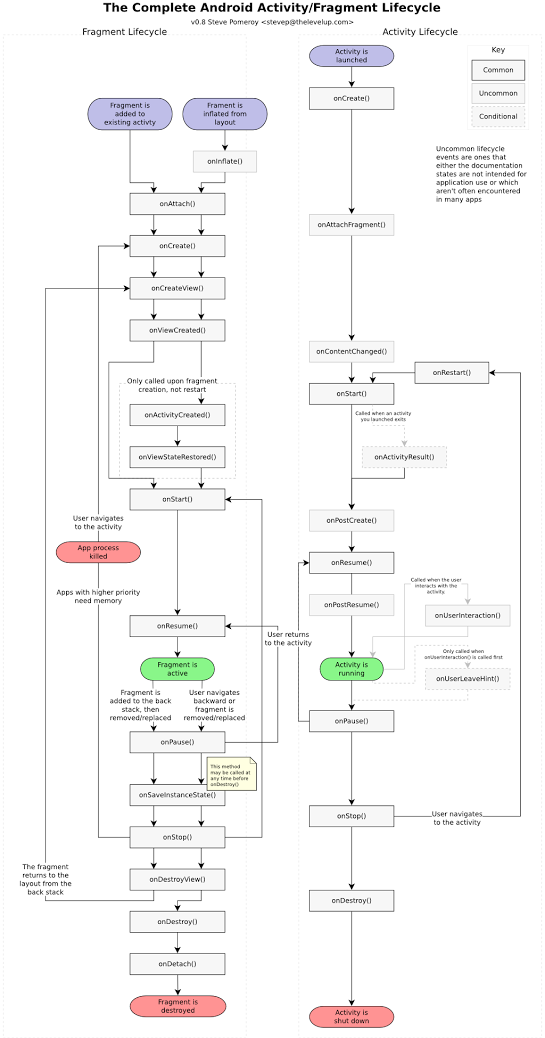
5\*. Шины событий. Рекомендую otto. В примере тоже otto. EventBus3 (green robot) по желанию на бонус балл. Шину хранить в классе приложения

6\*. ChildFragmentManager для самостоятельного изучения. Работает как обычный, только во фрагменте. Можно запихивать фрагмент во фрагменты. Сделайте несколько фрагментов, которые будут хоститься внутри другого фрагмента.

*Задания со \* дополнительные, на максимальный балл.*

Для зачета л/р вам нужно что-то сделать через коллбэки, что-то через шину событий. Прямые вызовы можно и нужно не делать.

***PS. Ниже приведены схемы жизненного цикла:***



**Лабораторная работа№ 3. Layouts.**

Итак, займёмся XML разметкой. Если подзабыли, что и куда, сайт Александра Климова вам поможет:

<http://developer.alexanderklimov.ru/android/theory/layout.php>

**----------------------------------------**

**| |**

**|** scroll **|**

**|** view **|**

**| |**

**----------------------------------------**

**| edit text | btn |**

**----------------------------------------**

Задание – сделайте разными способами что-то вроде окна чата:

В левом нижнем углу есть кнопка с height/width=wrap\_content (или лучше фиксированные величины, измеряемые в dp и лежащие в dimens.xml).

Внизу экрана слева от кнопки есть поле ввода текста, height аналогично кнопке (можно, чтобы совпадали), но ширина должна занимать всё свободное место на экране.

Всё свободное место над этой конструкцией должно занимать scroll view.

В scroll view можно положить вертикальный linear layout и внутрь него положить много text view с каким-нибудь текстом, чтобы была видимость скроллируемого контента.

1. LinearLayout. Используя только LinearLayout’ы можно сделать просто, понятно и с небольшими затратами по силам и времени почти любую компоновку. Читаем статью у Климова:

<http://developer.alexanderklimov.ru/android/layout/linearlayout.php>

Хотя кое-что воспринимаем критически: weight\_sum в большинстве случаев выставлять не требуется, если вам нужно разделить экран на 7 равных полосок (как в его примере), просто устанавливайте для каждой полоски layout\_weight=1.

Теперь делаем предложенную в задании разметку:

В корне вертикальный linear layout со scroll view с weight=1, внизу горизонтальный linear layout. В нижнем лэйауте у текстового поля вес равен 1.

2. ConstraintLayout. Требует для многих элементов задавать id, кроме того, если один элемент убрать – остальные съезжают и перемешиваются. Но он быстрее прорисовывается, чем linear layout.

Читаем опять у Климова:

[http://developer.alexanderklimov.ru/android/layout/constraintlayout.php](http://developer.alexanderklimov.ru/android/layout/relativelayout.php)

Сделать предложенную разметку с помощью данного контейнера также можно, правильно выставить только якоря.

3. FrameLayout. Самый простой вид разметки, но иногда позволяет делать интересные вещи. На нём конечно не будем делать окно чата, есть другое интересное задание.

**----------------------------------------**

**|img -------------------- |**

**| | | |**

**| | btn | |**

**| -------------------- |**

**----------------------------------------**

Внутрь frame layout поместите Image view (heigh/width = fill\_parent), а поверх image button (heigh/width = wrap\_content, layout\_gravity = center). Добавьте в res/drawable произвольных картинок (png, имена в snake\_case), одну из них установите на имидж вью (атрибут src). На кнопку поставьте тоже src (какую-нибудь прозрачную стрелочку). Сделайте обработчик на кнопку, чтобы он менял картинки у image view каким-нибудь образом (циклично, рандомно). Таким образом, с помощью нехитрого FrameLayout вроде как сделали вьювер с управлением прозрачными кнопками поверх картинки.

Если нужно больше информации, то вы уже знаете:

<http://developer.alexanderklimov.ru/android/layout/framelayout.php>

4\*. GridLayout. Как-то не прижился он среди нас, хотя в WPF или в JavaFx является основным. Точно такую разметку на нём сделать не получится, но если мы договоримся, что у нас есть таблица 5\*6, а кнопка имеет размер 1\*1, то будет что-то похожее (разве что пропорции будут точно сохраняться на разных устройствах, что для чата не есть правильно, но для примера сойдёт).

Информация:

<http://developer.alexanderklimov.ru/android/layout/gridlayout.php>

5\*. Альтернативные ресурсы.

Сделайте разметку с 4 кнопками одна под другой. Потом сделайте альтернативную разметку для landscape ориентации, но кнопки должны быть выстроены квадратом. В первом случае должно выглядеть:

**------------------------------**

**| |**

**------------------------------**

**|** **|**

**------------------------------**

**| |**

**------------------------------**

**| |**

**------------------------------**

Во втором – примерно так:

**----------------------------------------**

**| | |**

**----------------------------------------**

**| | |**

**----------------------------------------**

На кнопках должны быть подписи, обработчики пусть выводят toast’ы (а лучше пусть на экран выводят фрагменты с предыдущими заданиями). В портрете и ландшафте картинка должна быть разная, но фрагмент и обработчики – одни и те же.

6\*. У Климова много описано о программном создании вьюх. Давайте тоже что-то такое сделаем. В вашем “чате” есть кнопка. Помните в начале письма фразу "В scroll view можно положить вертикальный linear layout и внутрь него много text view с каким-нибудь текстом, чтобы была видимость скроллируемого контента”? Вот сделайте это и при нажатии на кнопку добавляйте в этот самый linear layout “сообщения”, то есть дополнительные text view. Размерности пусть будут wrap\_content, текст <что-то там > <число

7\*. Все задания делаем в разных фрагментах, плюс нужен ещё навигационный фрагмент с кнопками, из которых можно будет вызвать всё это.

Предлагаю в качестве навигационного фрагмента заюзать оный из пункта 5. Пусть эти 4 кнопки ведут к разным частям задания: Linear, Constraint, Frame. Если “чат” пропустили делать с помощью Grid, то 4 кнопка пусть Toast выводит, или можно 3 сделать кнопки.

*Задания со \* дополнительные, на максимальный балл.*

**Лабораторная работа № 4. Intents.**

1. Сделайте в одной Activity текстовое поле (EditText) для имени пользователя и кнопку. При нажатии на кнопку создайте интент, поместите туда текст, введённый пользователем в EditText и откройте активити2. Выведите там имя в text view.

2. В Activity 2 разместите date picker/time picker и кнопку ОК. При нажатии на ОК сделайте set result и передайте дату. Это можно сделать, например с помощью getMillis (возвращает long – кол-во мс с 1970 г).

3. В Activity 1 примите и выведите дату.

4\*. Создайте в манифесте интент фильтр, чтобы в вашем приложении открывались картинки.

5\*. Найдите способ вывести картинку в image view

6\*. Откройте плей маркет при нажатии на какую-либо кнопку.

7\*. Попробуйте получить фото от камеры и вывести в image view результат

*Задания со \* дополнительные, на максимальный балл.*