|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА  Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра математического обеспечения и стандартизации ИТ**

**ОТЧЁТ ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №14**

**по дисциплине**

**«Разработка мобильных приложений»**

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет представлен к  рассмотрению:  Студенты группы ИНБО-04-20 | « » февраля 2022 г. | (подпись) | Ло В.Х. |
|  |  |  |  |
| Преподаватель | « » 2022 г. | (подпись) | Фандеев И.И. |

Москва, 2022г.

**СОДЕРЖАНИЕ**

[ОТЧЕТ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОЙ РАБОТЫ. 3](#_Toc104337118)

[1. Добавление анимации. Эффект затухания (crossfading) 3](#_Toc104337119)

[1.1 Реализовать создание объектов для переключения 3](#_Toc104337120)

[1.2 Реализовать создание анимации 4](#_Toc104337121)

[2. Использование ViewPager для создания эффекта перелистывания 5](#_Toc104337122)

[2.1 Реализовать создание объектов 6](#_Toc104337123)

[2.2 Реализовать создание фрагментов 6](#_Toc104337124)

[3. Настройка анимации с использованием PageTransformer 8](#_Toc104337125)

[3.1 Реализовать трансформер Zoom-out 8](#_Toc104337126)

[3.2 Реализовать трансформер с прозрачностью 8](#_Toc104337127)

[4. Эффект переворачивания (Card Flip) 9](#_Toc104337128)

[4.1 Реализовать создание Аниматоров 9](#_Toc104337129)

[4.2 Реализовать создание контейнеров 11](#_Toc104337130)

[4.3 Реализовать создание фрагмента 11](#_Toc104337131)

[4.4 Реализовать создание анимации 12](#_Toc104337134)

[5. Эффект увеличения (zooming) 13](#_Toc104337138)

[5.1 Реализовать создание контейнеров 13](#_Toc104337139)

[5.2 Реализовать создание анимации 14](#_Toc104337140)

[5.3 Реализовать увеличение ImageView 15](#_Toc104337141)

[6. Анимация изменений разметки 18](#_Toc104337142)

[6.1 Реализовать создание разметки 18](#_Toc104337143)

[6.2 Реализовать добавление, изменение и удаление элементов разметки 18](#_Toc104337144)

[ВЫВОД 19](#_Toc104337145)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc104337146)

# ОТЧЕТ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОЙ РАБОТЫ.

**1.** **Добавление анимации. Эффект затухания (crossfading)**

Анимация может создавать небольшие визуальные подсказки, которые уведомляют пользователя о том, что в приложении что-то происходит, тем самым улучшая психологическую модель восприятия интерфейса вашего приложения.

Анимация особенно полезна, когда экран меняет свое состояние, когда загружается новое содержимое или становятся доступными новые действия.

Анимацию можно использовать, чтобы придать приложению профессиональный вид.

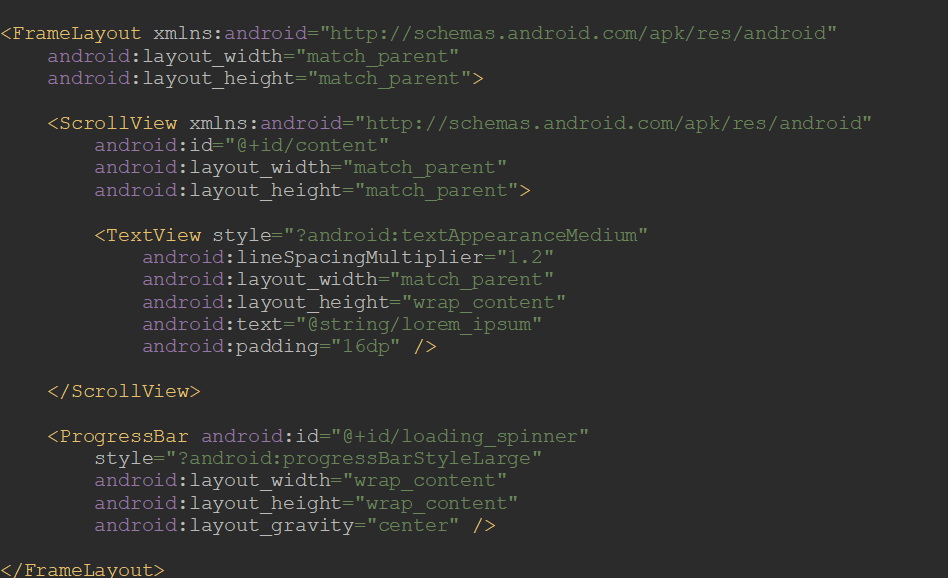
Crossfade анимация (также известная как затухание), это плавное

исчезновение одного компонента с одновременным плавным появлением другого.

Эта анимация удобна при переключении содержимого в приложении.

Crossfade анимация довольно короткая, но предлагает плавное переключение с одного экрана на другой.

**1.1 Реализовать создание объектов для переключения**

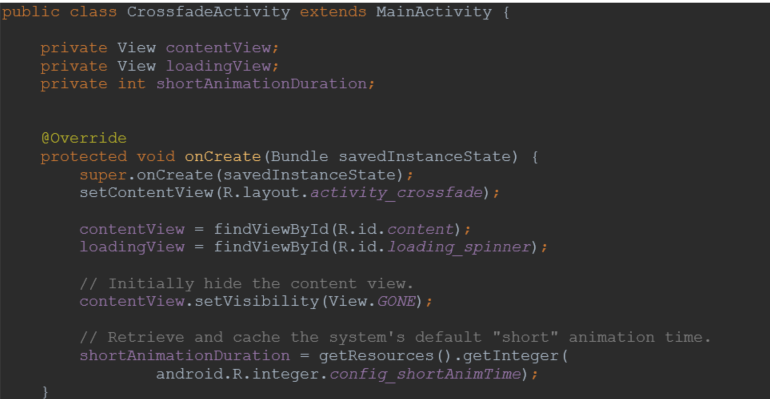


**Рисунок 1 – создание объектов для переключения**

**1.2 Реализовать создание анимации**

Для создания анимации:

* Создаем переменные для объектов, которые хотим переключать. Необходимо создать ссылки после изменения объектов во время анимации.
* Для объекта, который будет исчезать, ставим свойство visibility равное GONE. Эта опция не позволит объекту занимать пространство разметки и не включит его в расчет разметки, что ускорит процесс.
* Сохраним системное свойство config\_shortAnimTime в переменную. Это свойство описывает стандартную “короткую” продолжительность анимации. Такая продолжительность идеальна для едва заметной анимации или для анимации, которая происходит очень часто. Параметры config\_longAnimTime и config\_mediumAnimTime также доступны и вы можете их использовать.

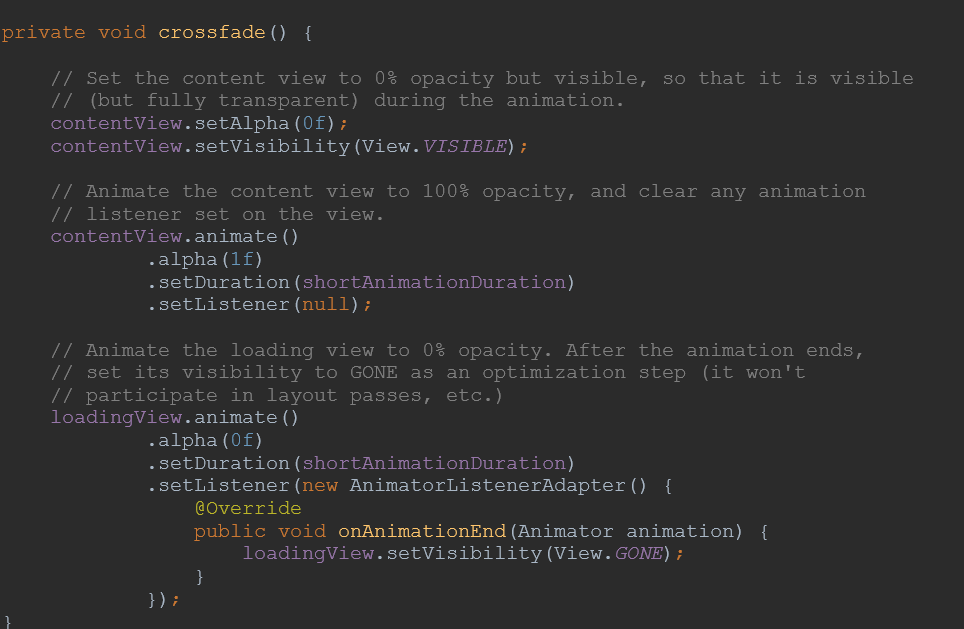
****

**Рисунок 2 – создание анимации**

**1.3 Реализовать переключение объектов**

Когда объекты подготовлены, необходимо сделать следующее для переключения между ними:

* Для объекта, который будет скрываться, устанавливаем значение alpha равное 0 и свойство visibility в значение VISIBLE. (Помните, что оно было инициализировано значением GONE). Это сделает объект видимым, но полностью прозрачным.
* Для объекта, который будет появляться, изменяем значение alpha от 0 до 1. В то же самое время, для объекта, который будет исчезать, изменяем значение alpha от 1 до 0.
* Используя метод onAnimationEnd() объекта Animator.AnimatorListener, устанавливаем свойство visibility объекта, который нужно показать, в значение GONE. Несмотря на то, что значение alpha равно 0, значение параметра visibility равное GONE не позволит объекту занимать пространство разметки и не включает его в расчет макета, что ускоряет обработку.



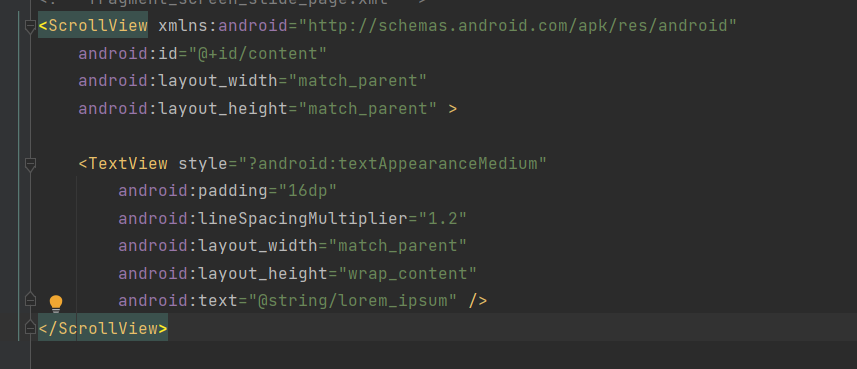
**Рисунок 3 – переключение объектов**

**2. Использование ViewPager для создания эффекта перелистывания**

Такой способ является общим для интерфейсов, вроде мастеров настройки или слайдшоу.

ViewPager позволяет создавать такой эффект автоматически.

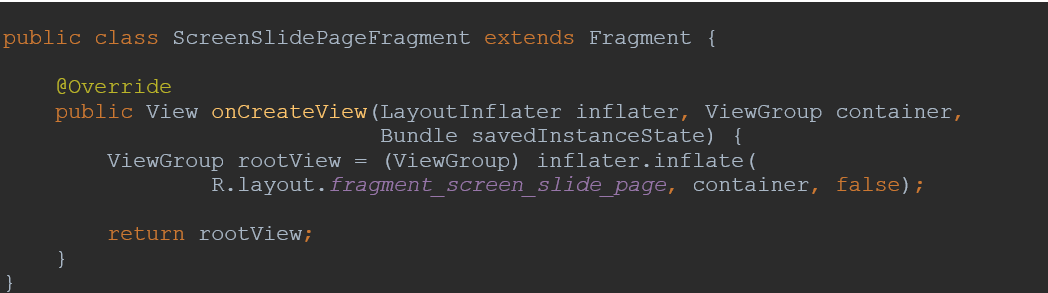
2.1 **Реализовать создание объектов**



**Рисунок 4 – файл макета**

2.2 **Реализовать создание фрагментов**

Создаем расширенный класс Fragment, который будет возвращать разметку. Разметку мы создадим в методе onCreateView().

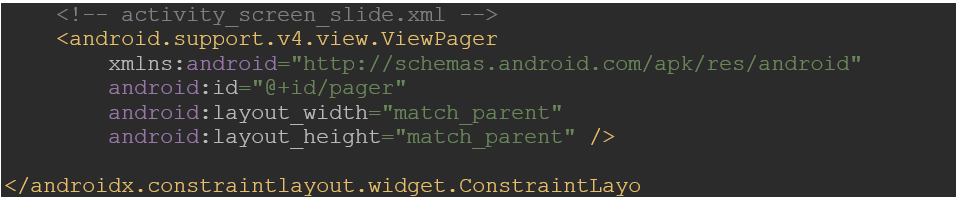
****

**Рисунок 5 – создание фрагментов**

**2.3 Реализовать добавление компонент ViewPager**

Компонент ViewPager содержит встроенные жесты для переключениястраниц, которые показывают анимацию перелистывания по умолчанию.

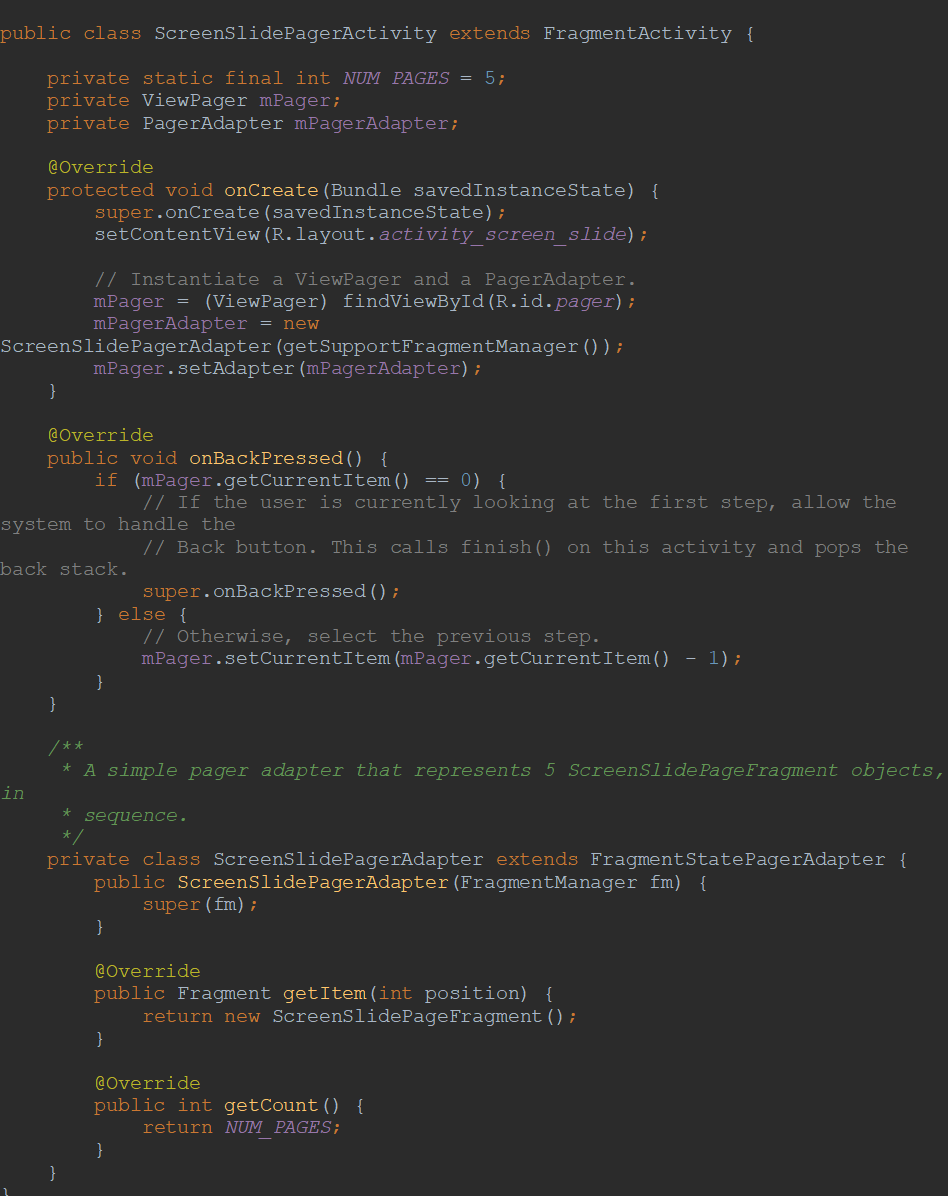
В свою очередь PageAdapter использует класс фрагмента, который мы создали ранее.



**Рисунок 6 – макет, который содержит**[**ViewPager**](http://developer.android.com/reference/android/support/v4/view/ViewPager.html)

Явление, которое делает следующее:

* Устанавливает компонент, который будет размещаться в ViewPager.
* Создает наследник абстрактного класса FragmentStatePagerAdapter и реализует метод getItem() для поставки экземпляров класса ScreenSlidePageFragment как новых страниц. Адаптер страниц также требует реализации метода getCount(), который возвращает количество страниц адаптера, который будет
* Подключает PagerAdapter в ViewPager.
* Обрабатывает аппаратную кнопку “назад”, перемещаясь обратно по виртуальному стеку фрагментов.



**Рисунок 7 – Явление**

3. **Настройка анимации с использованием PageTransformer**

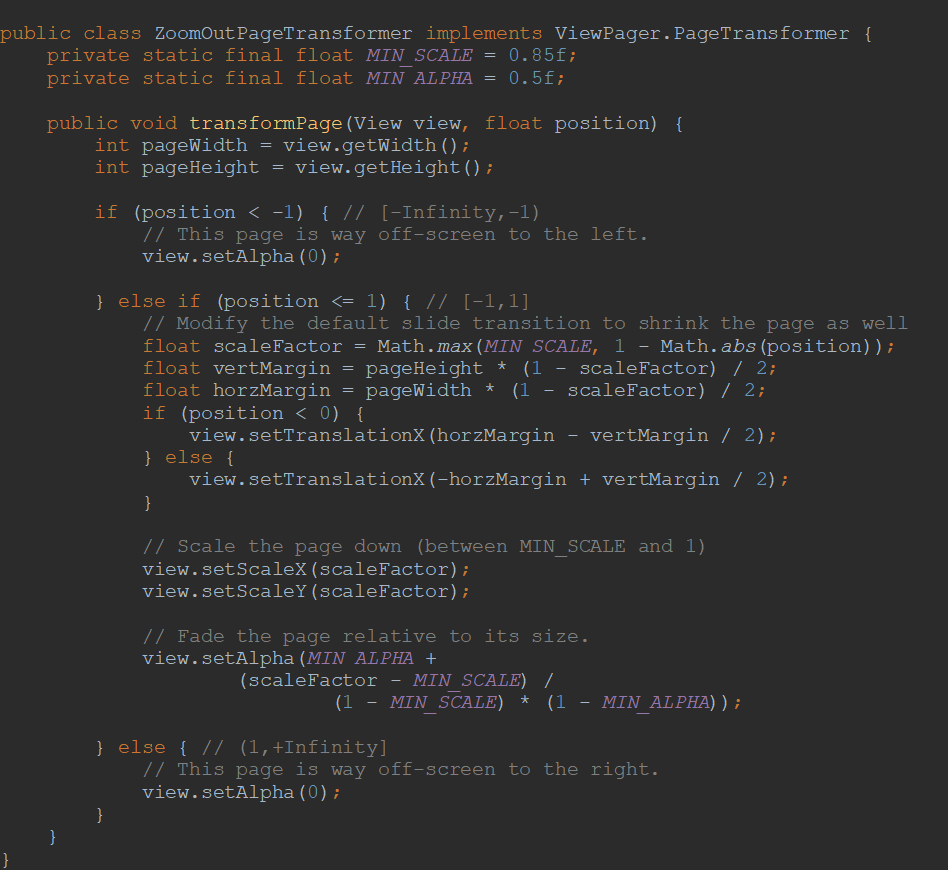
Для использования другой анимации, реализуйте интерфейс ViewPager.PageTransformer и поставьте его для ViewPager.

Интерфейс предоставляет единственный метод transformPage().

В каждой точке перехода, этот метод вызывается однократно для каждой видимой страницы (как правило есть только одна видимая страница) и для соседних страниц.

3.1 **Реализовать трансформер Zoom-out**

Этот трансформер сначала сжимает, а затем прячет страницу при прокрутке. Когда страница становится ближе к центру, она растет обратно до нормальных размеров.



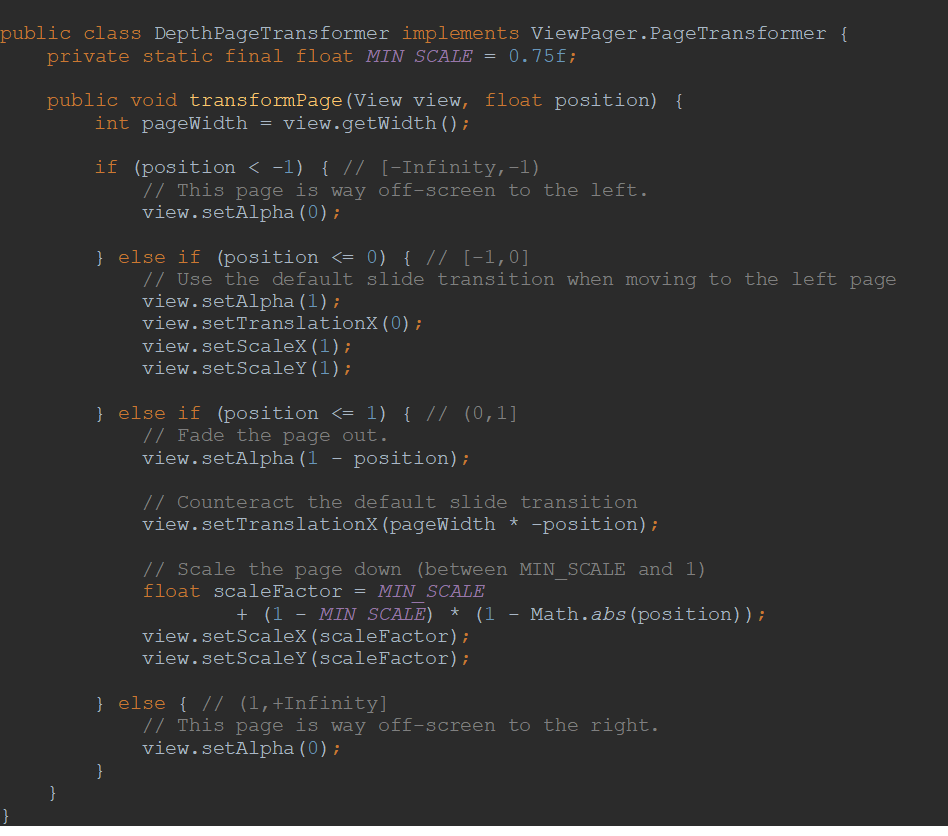
**Рисунок 8 – трансформер Zoom-out**

3.2 **Реализовать трансформер с прозрачностью**

Этот трансформер использует стандартную анимацию при прокрутке влево и анимацию с прозрачностью при прокрутке вправо. Применяем DepthPageTransformer



**Рисунок 9 – отключить анимацию перелистывания для отрицательного направления X**



**Рисунок 10 – противодействовать стандартной анимации перелистывания в трансформере**

**4. Эффект переворачивания (Card Flip)**

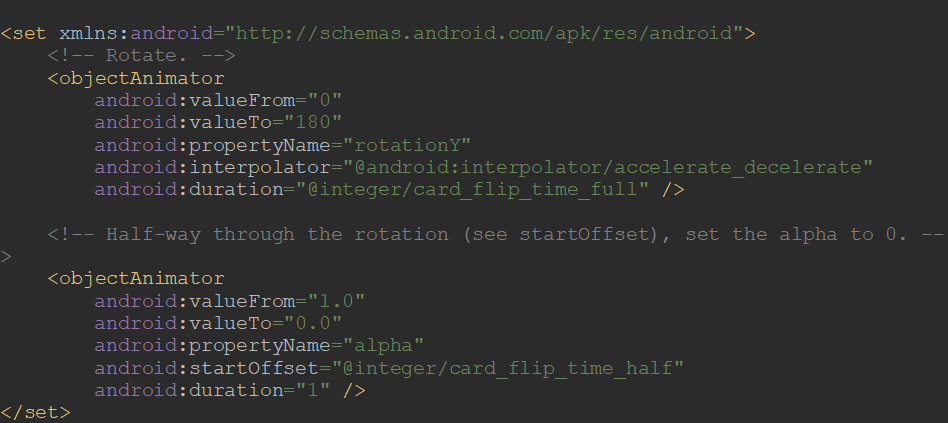
4.1 **Реализовать создание Аниматоров**

Два объекта аниматора: один для эффекта, когда передняя часть прячется через левый край, и второй, когда передняя часть появляется из левого края.

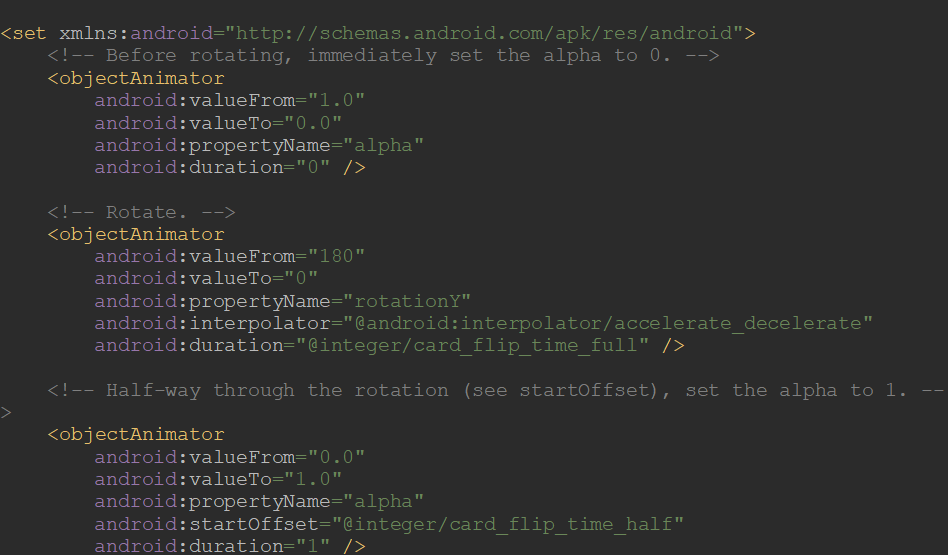
Также нужно создать два аниматора для задней части, которая будет появляться и исчезать из правого края.



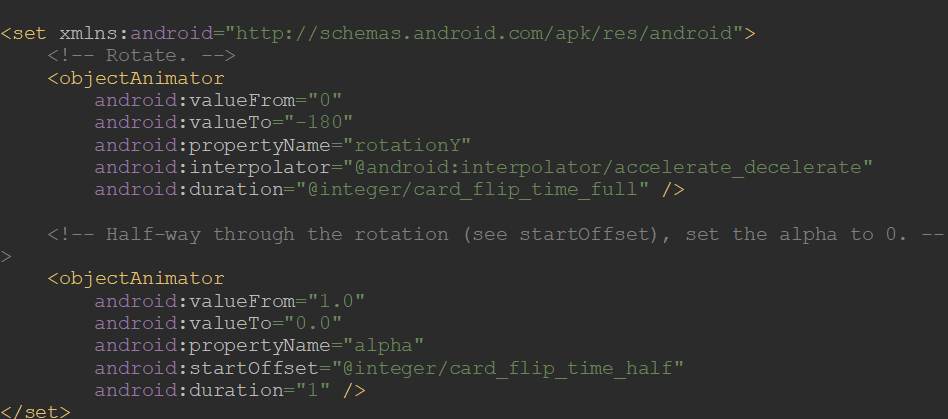
**Рисунок 11 – card\_flip\_left\_in.xml**



**Рисунок 12 – card\_flip\_left\_out.xml**



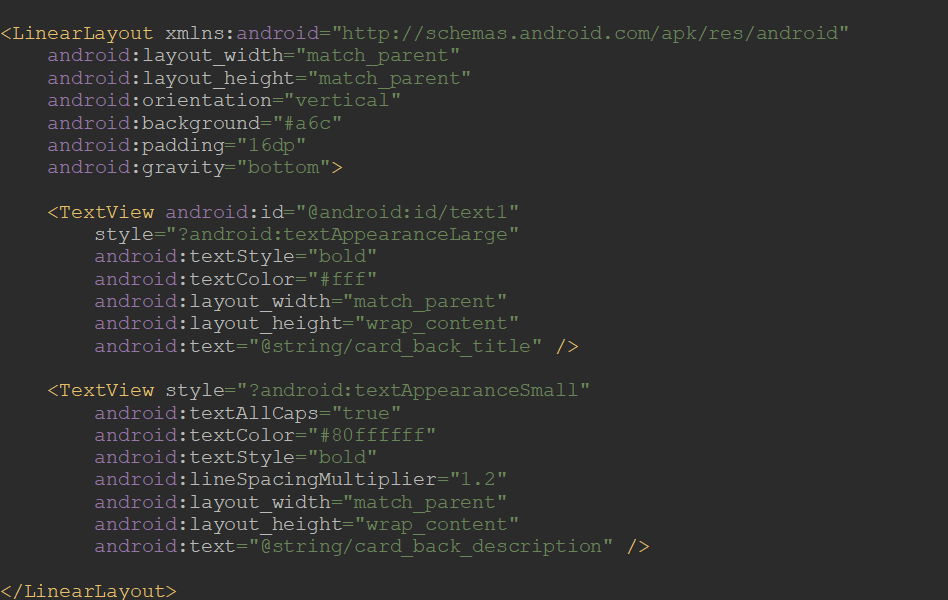
**Рисунок 13 – card\_flip\_right\_in.xml**



**Рисунок 14 – card\_flip\_right\_out.xml**

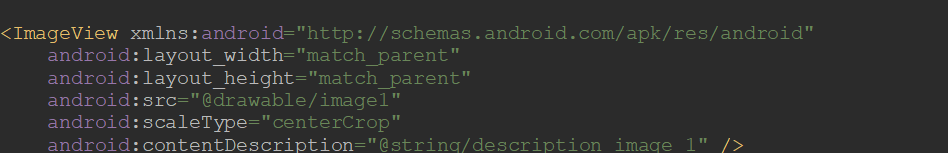
4.2 **Реализовать создание контейнеров**

На каждой стороне “карточки” есть отдельная разметка, которая может включать любое содержимое по вашему желанию, например два компонента текста, два изображения, или комбинация любых компонентов.



**Рисунок 15 – создание контейнеров(1)**

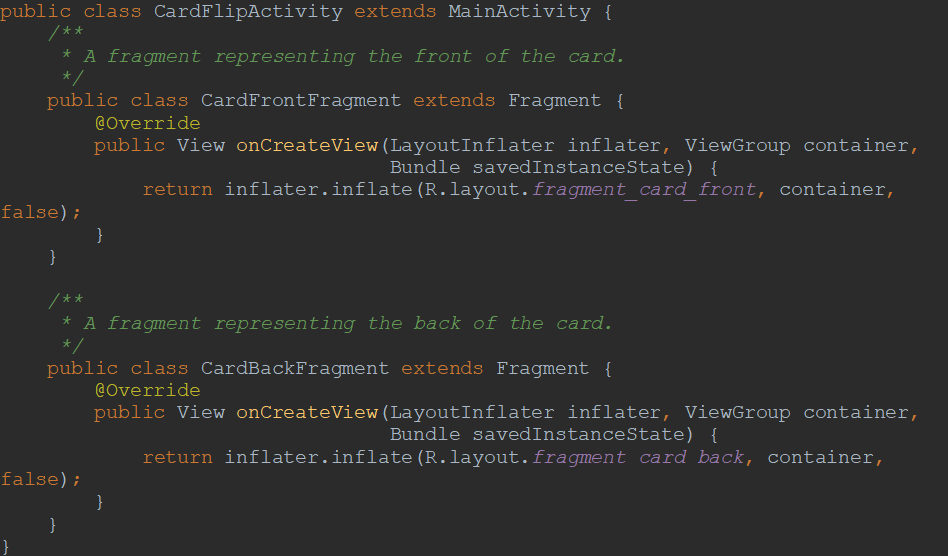
И другая сторона, содержащая ImageView:



**Рисунок 16 – создание контейнеров(2)**

**4.3 Реализовать создание фрагмента**

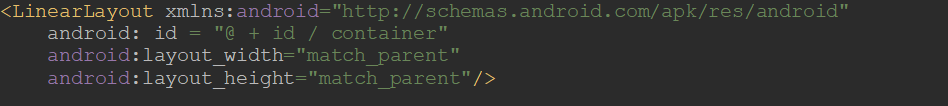
Создадим классы фрагментов для передней и задней стороны “карточки”. Эти классы будут возвращать разметку, которую мы создали в предыдущем материале, в методе onCreateView().

****

**Рисунок 17 – создание фрагмента**

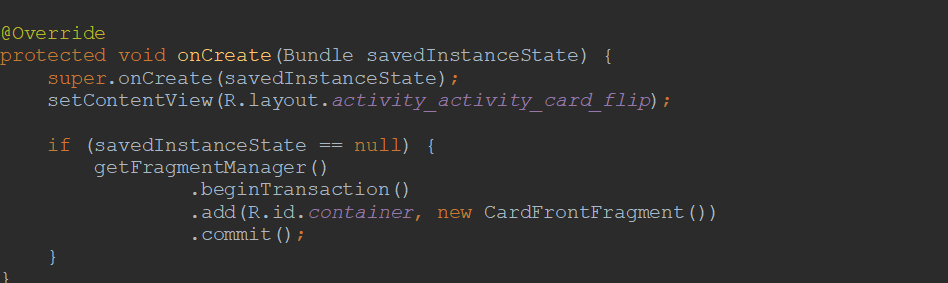
**4.4 Реализовать создание анимации**

Сначала создадим разметку явления.



**Рисунок 18 – разметка явления**

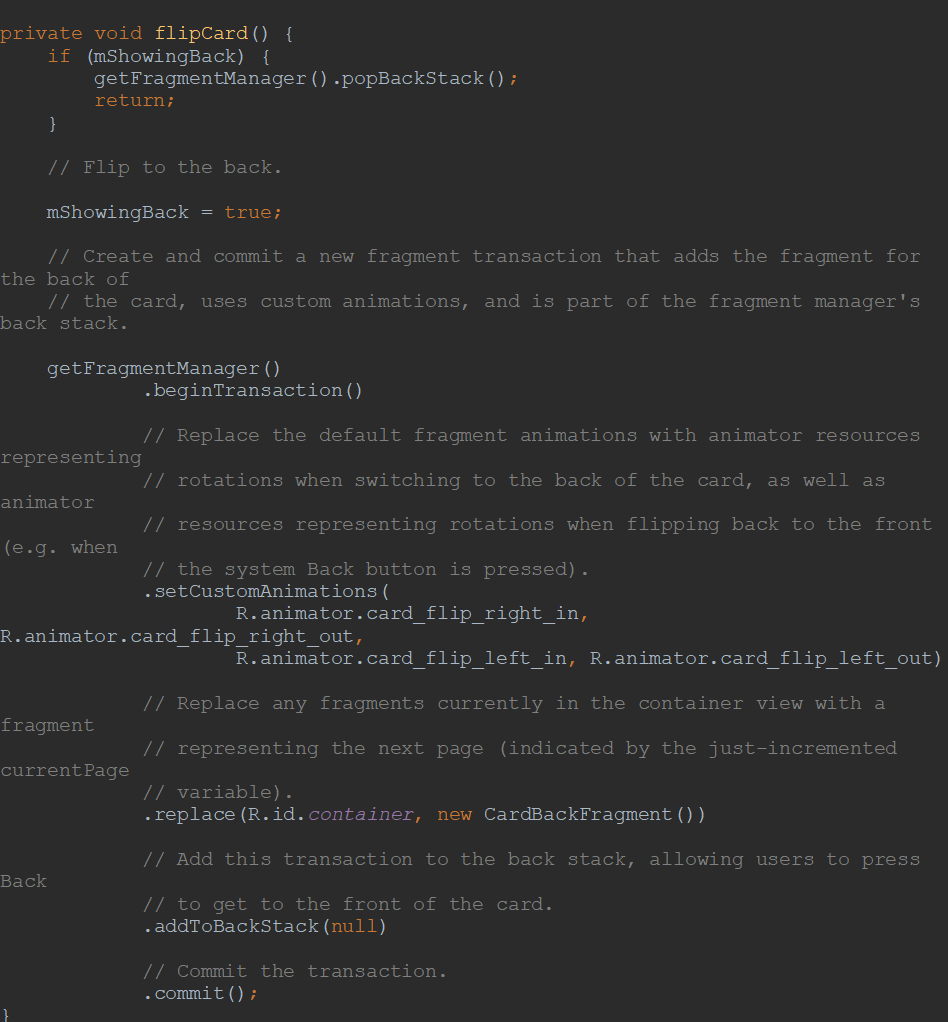
В коде установите для явления разметку, которую только что создали. Также разумно показать фрагмент по умолчанию при создании явления.



**Рисунок 19 – показать фрагмент по умолчанию при создании явления**

Метод, который будет делать следующее:

* Выбирать аниматоров, которых мы создали ранее.
* Заменять текущий фрагмент следующим, используя нашу анимацию<./li>
* Добавлять предыдущий фрагмент в стек, чтобы при нажатии кнопки “назад” он вернулся на экран.

****

**Рисунок 20 – метод flipcard**

**5. Эффект увеличения (zooming)**

5.1 **Реализовать создание контейнеров**

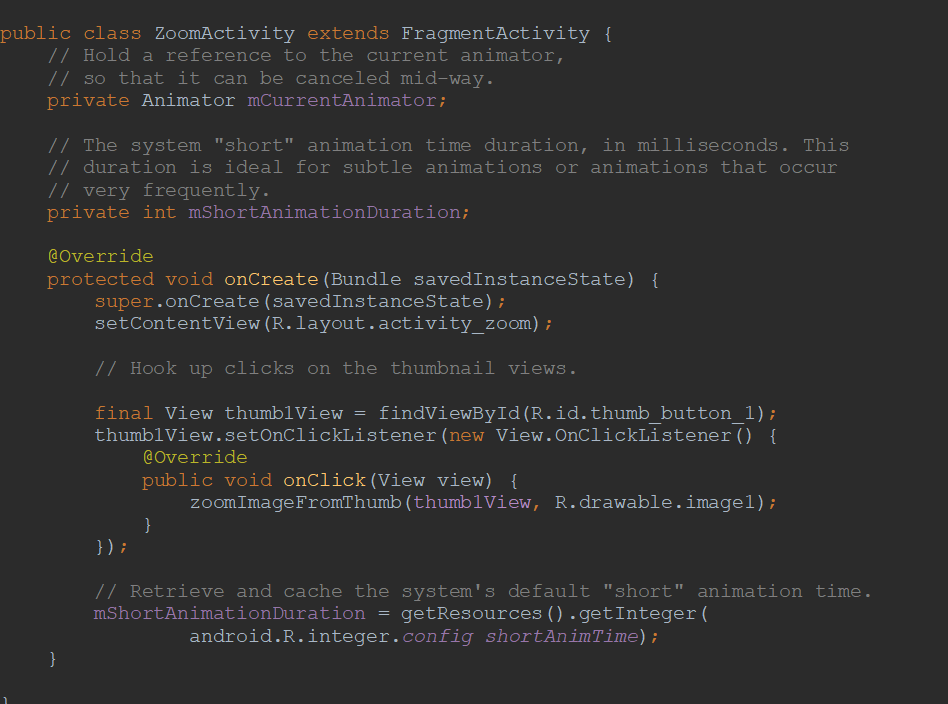
Создайте файл разметки, содержащий большую и маленькую версию содержимого, которое вы хотите увеличивать.



**Рисунок 21 – создание контейнеров**

5.2 **Реализовать создание анимации**

Теперь создадим события, которые будут запускать анимацию. Добавим к кнопке ImageButton обработчик нажатия View.OnClickListener, чтобы запустить анимацию при щелчке по этой кнопке:



**Рисунок 22 – создание анимации**

**5.3 Реализовать увеличение ImageView**

Следующие шаги показывают как реализовать анимацию увеличения:

* Вставьте изображение с высоким разрешением в скрытый компонент ImageView. В следующем примере для простоты мы будем загружать изображение в потоке интерфейса. В реальных приложениях вы должны всегда загружать и обрабатывать изображения в отдельном потоке! В идеале, размер изображения не должен быть больше размера экрана.
* Рассчитываем начальные и конечные границы ImageView
* Одновременно анимируем каждое из четырех свойств размеров и позиций X, Y (SCALE\_X и SCALE\_Y), от начальных границ до конечных. Эти четыре аниматора добавлены в AnimatorSet и могут быть запущены в любое время.
* Будем уменьшать изображение, выполнив аналогичную анимацию, когда пользователь коснется полноразмерного изображения. Вы можете добавить обработчик View.OnClickListener для объекта ImageView. При клике по ImageView уменьшите этот элемент до размера миниатюры, а затем присвойте его свойству visibility значение GONE, чтобы его скрыть.



**Рисунок 23 – увеличение ImageView(1)**



**Рисунок 24 – увеличение ImageView(2)**



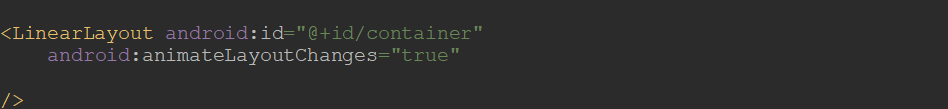
**Рисунок 25 – увеличение ImageView(3)**

**6. Анимация изменений разметки**

Анимация разметки это предустановленная анимация, которую система запускает каждый раз, когда разметка меняется.

**6.1 Реализовать создание разметки**

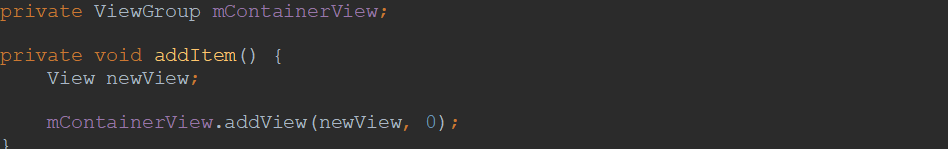
В файле разметки установите атрибут android:animateLayoutChanges в значение true.



**Рисунок 26 – создание разметки**

**6.2 Реализовать добавление, изменение и удаление элементов разметки**

Теперь при добавлении, изменении или удалении элементов разметки анимация будет добавляться автоматически:



**Рисунок 26 – добавление элементов разметки**

# ВЫВОД

В ходе выполнения практической работы были изучены:

* Методические указания Графика и анимация

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Лекции по дисциплине «Разработка мобильных приложений» / И. В. Синицын, МИРЭА — Российский технологический университет, 2022.