**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ЯРОСЛАВСКОЙ ОБЛАСТИ**

государственное профессиональное образовательное учреждение   
Ярославской области

Рыбинский полиграфический колледж

**КУРСОВОЙ ПРОЕКТ**

|  |  |
| --- | --- |
| Разработка класса визуальных компонентов для управления элементами списка при | |
| помощи жестов | |
| по дисциплине | Объектно-ориентрованное программирование |
|  | |

Пояснительная записка

|  |
| --- |
| КП.0902.14.000000.00 ПЗ |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы | 3-ИС-2 |  |  |  | Тимофеев Р В |
|  | *(Код учебной группы)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Руководитель | преподаватель |  |  |  | Смирнов В Б |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Нормоконтроль | преподаватель |  |  |  | Смирнов В Б |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |

г. Рыбинск

2022

**Оглавление**

[Введение 3](#_Toc107435794)

[1 Исследовательский раздел 6](#_Toc107435795)

[2 Конструкторский раздел 9](#_Toc107435796)

[2.1 Математическая модель 9](#_Toc107435797)

[2.2 Разработка структуры компонента 10](#_Toc107435798)

[2.3 Разработка графического интерфейса компонента 13](#_Toc107435799)

[2.4 Описание входных и выходных данных компонента 14](#_Toc107435800)

[2.5 Разработка алгоритмов компонента 15](#_Toc107435801)

[3 Технологический раздел 25](#_Toc107435802)

[3.1 Отладка программного кода класса 25](#_Toc107435803)

[3.2 Инструкция по установке компонента в среду разработки 27](#_Toc107435804)

[3.3 Инструкция по эксплуатации компонента 28](#_Toc107435805)

[4 Раздел охраны труда 30](#_Toc107435806)

[Заключение 32](#_Toc107435807)

[Список используемых источников 33](#_Toc107435808)

[Приложение А 34](#_Toc107435809)

[Приложение Б 40](#_Toc107435811)

[Приложение В 41](#_Toc107435813)

# Введение

В настоящее время, почти каждый человек в мире так или иначе связан с вычислительными машинами. Используя их, человек может получить большинство знаний мира в любой момент и в любое время суток, если у него есть возможность. С каждым годом, люди усовершенствуют технологии, чтобы достигать более новых возможностей. Но так было не всегда, так с чего же всё началось?

Любая вычислительная техника работает на основе программного обеспечения, которое, в свою очередь, написано благодаря языкам программирования. Язык программирования – это набор алгоритмов, представленных в виде машинного кода, который упрощает разработку программного обеспечения для гаджетов. Одним из первых популярных языков программирования стал ассемблер, использующий принцип трансляции программы в машинный код, как указано в работе [1].

Ассемблер является низкоуровневым языком программирования, что обозначает то, что он близок к программированию на машинном коде. Приход ассемблера сильно упростил работу программистов. Связано это с тем, что программисту больше не требовалось помнить машинные коды, вместо этого использовались зарезервированные слова, команды, которые включали в себя выполнение тех самых машинных кодов.

Тем не менее, со временем сложность программ увеличивалась, и ассемблера стало не хватать для продуктивной разработки программного обеспечения. Программисту приходилось писать один и тот же процедурный код по несколько раз в программе, что сильно увеличивало время разработки программного обеспечения, а также утруждало разработчика. Программисты занялись решением вышеописанных проблем и в скором времени был предоставлен новый тип языков программирования, одним из наиболее известных представителей стал Фортран. Фортран имел большой набор математических функций, поддерживал работу с целыми, вещественными и комплексными числами двойной и повышенной точности, включал в себя богатый инструментарий работы с массивами и внешними файлами. Программное обеспечение, созданное с помощью высокоуровневых языков программирования, было более простым в написании и чтении другими программистами [2].

Со временем, даже таких языков программирования стало не хватать программистам, возрастал риск возникновения множества ошибок при работе над большим проектом в связи с тем, что приходилось писать много процедур, данные которых доступны только внутри них самих же, что обязывало программистов писать аналогичный код. Программистам стало не хватать стандартных типов данных, так как объект следовало описывать несколькими переменными, что также являлось неудобным. Принцип объектно-ориентрованного программирования решил эти проблемы. В основе объектно-ориентированных языков программирования стало то, что все, что поддавалось разработке, являлось объектом или классом. Также, он позволяет создавать составные типы данных, основанные на стандартных. Под объектом в идеологии объектно-ориентированного языка программирования понималась сущность, обладающая некоторым состоянием и поведением. Классом же является шаблон для создания объектов программирования, обеспечивающий начальные значения состояний и реализацию его поведения.

После появления множества языков программирования, принцип которых построен на объектно-ориентированном программировании, разработка мобильных приложений стала намного легче, в связи с возможностями данного подхода. В основе мобильных приложений лежат классы компонентов, в том числе и визуальных, вследствие этого темой данного курсового проекта и была выбрана разработка класса визуальных компонентов для управления элементами списка при помощи жестов.

Разработка списка на мобильном приложении наиболее эффективна при условии использования объектно-ориентированных языков программирования, потому что данный подход точно определяет взаимодействие нескольких объектов между друг другом, что требуется в данном курсовом проекте.

# 1 Исследовательский раздел

Чтобы создать визуальный компонент требуется среда разработки для написания программных продуктов для мобильных устройств. Перед началом создания визуального компонента были рассмотрены три среды разработки, среди которых были Visual Studio 2019, Android Studio Bumblebee и XCode 13.

Visual Studio – один из старейших программных продуктов для создания программ с графическим интерфейсом. Это - удобная среда разработки от компании Microsoft, позволяющая быстро и эффективно разрабатывать проект, выбрав для этого все необходимое. Основными инструментами для создания мобильного приложения в Visual Studio является ядро разработки .NET, Xamarin.Forms и XAML. Плюсом Visual Studio в категории мобильной разработки является то, что в ней возможно создание как продукта под операционную систему Android, так и под iOS. Одним из главных недостатков Visual Studio является то, что начинающим программистам довольно трудно разобраться с интерфейсом. В начале пути программиста это сильно его отпугивает.

Android Studio Bumblebee – программный продукт от компании Google, являющийся самым новым среди рассматриваемых и одной из самых быстроразвивающихся сред разработки. Плюсами разработки в Android Studio можно назвать возможность создания макетов не только для мобильной разработки, но и под другие гаджеты, такие как умные часы, планшеты и умные телевизоры. Минусом Android Studio является её сложная первоначальная настройка для работы.

Последней рассматриваемой средой разработки является XCode 13 – программное обеспечение, созданное компанией Apple. Данная среда разработки используется для создания программного обеспечения только под операционную систему iOS, что является одним из её минусов. Также, использование XCode возможно лишь на продукции Apple, что делает её недоступной для программистов, имеющих отличную от macOS операционную систему. Плюсами данной среды разработки является её лёгкая и быстрая первоначальная настройка, поддержка нескольких языков программирования, таких как Objective-C и swift, а в связи с тем, что доступ к программе имеется лишь на macOS, быстродействие программы на более высоком уровне, чем у других сред разработки. Из работ [3, 4, 5].

Среди рассмотренных сред разработки, был сделан выбор в сторону Android Studio Bumblebee в связи с тем, что она сильнее подходит под тему курсового, чем другие рассмотренные варианты.

Родительским классом компонента был выбран класс ConstraintLayout, данный выбор был сделан по причине того, что он является контейнером, внутри которого можно располагать множество дочерних элементов, что потребуется в рамках данного проекта. Также одной из причин выбора ConstraintLayout в качестве родительского класса стало то, что у него широкий выбор возможностей компоновки внутренних объектов, что подходит для разрабатываемого компонента.

Список, реализуемый в данном курсовом проекте, представляет собой перечень элементов – фотографий, которые можно листать по горизонтали, с одновременным выполнением действий, описанных в коде при перелистывании в каждую из сторон. Подобного вида списки наиболее часто используются в мобильной разработке. Реализация таких списков возможна и в web-программировании, одним из таких примеров является список web-приложения Tinder. Список в данном приложении имеет возможности перелистывания по горизонтали, уходя с экрана с определенной анимацией. При перелистывании элемента списка в одну из стороны происходит конкретное действие, описываемое программистом. В отличии от списка, который будет реализован в проекте, пример имеет лишь один объект отображения одновременно.

В разрабатываемом проекте будет реализована стандартная логика списка – отображение главного объекта, с которым будут происходить манипуляции пользователем, с возможностью настройки анимации исчезновения объекта. Будет реализована возможность отображения заданного количества фотографий на экране. Загрузка фотографий в список будет реализована из удаленных источников, что потребует подключение к глобальной сети Интернет для работы компонента.

Основным процессом в проекте будет являться реализация анимации исчезновения главного объекта с экрана с учетом настроек, заданными пользователем.

# 2 Конструкторский раздел

## 2.1 Математическая модель

Математическая модель описывает процесс сортировки пользователем элементов списка, представленных в виде фотографий, на две категории. Сортировка будет осуществляться путем перетаскивания очередного объекта, находящегося вверху стопки и отображаемого в качестве основной картинки на экране, в одну из сторон по горизонтали. Такой компонент может быть использован в оценке показываемого элемента, при его перетаскивании в одну сторону –формирование положительного отзыва, при перетаскивании в другую сторону – отрицательного.

Процесс анимации будет зависеть от того, как далеко от центра списка пользователь перетащит объект списка. Концом анимации будет являться точка невозврата, настраиваемая пользователем. Если объект списка не достигнет точки невозврата, то он вернётся в центр списка и останется в качестве основной картинки, иначе же, он исчезнет, а следующая картинка будет установлена как главная, пока список не окажется пуст, в таком случае, будет выведен текст, который также настраивается пользователем.

Анимация исчезновения объекта списка будет зависеть от пользовательских настроек, что означает свободную настройку объектов списка и анимации исчезновения под себя. Пользователь сможет выбрать радиус закругления, прозрачность и отступы объектов списка, скорость исчезновения, точку невозврата и конкретную анимацию исчезновения объекта.

Список изначально будет заполнятся ссылками на фотографии, которые подготавливает прикладной программист. Список будет отображается на экране в соответствии с местом, выбранным прикладным программистом. После загрузки фотографий, появится возможность с ними взаимодействовать.

## 2.2 Разработка структуры компонента

Для формирования структуры класса будет использоваться путь проведения серии декомпозиций, начиная с представления компонента в виде модель «Черная сфера». Модель «Черная сфера» представляет собой систему, внутреннее устройство которой неизвестно. В эту систему подаются входные данные, а на выходе из системы поступают выходные данные. Модель «Черная сфера» приведена на рисунке 2.1.



Рисунок 2.1 – Модель «Черная сфера»

Представим наш компонент в виде модели «черной сферы». В компоненте будут присутствовать такие входные данные, как размеры компонента, пользовательские настройки и воздействия пользователя, управляющие компонентом. В выходных данных будет отображение фотографий и проведение анимации. Черная сфера с перечисленными входными и выходными данными приведена на рисунке 2.2.



Рисунок 2.2 – Черная сфера с параметрами

Для выявления основных параметров и процессов компонента проведем несколько этапов декомпозиции. Декомпозиция – операция [мышления](https://ru.wikipedia.org/wiki/%D0%9C%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5), состоящая в разделении целого на части. Также декомпозицией называется общий приём, применяемый при решении [проблем](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D0%B0), состоящий в разделении проблемы на множество частных проблем, а также [задач](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0), не превосходящих суммарно по [сложности](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) исходную проблему, с помощью объединения решений которых, можно сформировать решение исходной проблемы в целом.

Проведем первый этап декомпозиции с выявлением основных внутренних структур данных, необходимых для хранения информации и процессов компонента. Размеры компонента будут влиять на отображаемые фотографии, а также возможную зону действия анимации. Так как изменение размеров компонента возможно только на момент его инициализации, обрабатывать входные данные будет метод родительского класса, который зафиксирует размеры до завершения приложения. Пользовательские настройки, также как и размеры компонента, можно задать только на этапе его инициализации, поэтому изменение настроек возможно лишь один раз за жизненный цикл приложения. В связи с этим, отдельного механизма изменений настроек не предусмотрено. Обработка воздействия пользователя на компонент будет выполнять событие, реагирующем на нажатие и последующем удержании и перемещении фотографии горизонтально в одну из стороны по горизонтали. Результат первого этапа декомпозиции приведен на рисунке 2.3.

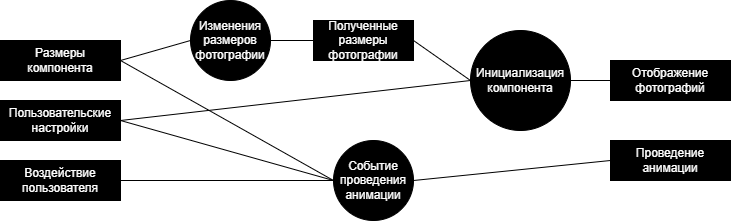


Рисунок 2.3 – Результат первого этапа декомпозиции

На первом этапе декомпозиции были получены основные структуры данных, необходимые для хранения информации и процессов компонента, которые позволят построить базовую логику. Для получения более точных структур данных и процессов компонента проведем второй этап декомпозиции.

Для применения размеров главной фотографии в зависимости от размеров компонента, используется отдельный метод. Для всех остальных фотографий, предоставлен отдельный механизм, основанный на размерах первой фотографии. Установкой пользовательских настроек, таких как, минимальная непрозрачность фотографии, процент закругления фотографий, отступы между фотографиями, нахождение точки невозврата, количество отображаемых фотографий и угол поворота фотографии во время анимации, будет использоваться метод проверяющий доступность указанных данных прикладным программистом и применяющим эти значения на момент инициализации. Событие, вызываемое воздействием пользователя на компонент, будет использовать все данные пользовательских настроек для корректного отображения анимации и последующего удаления фотографии из списка, при прохождении точки невозврата. Событие обрабатывает одновременно 3 возможных варианта воздействия пользователем, такие как нажатие на фотографию, перемещение элемента по компоненту и завершение нажатия. Проигрывание анимации происходит только во втором случае, остальные же предназначены для корректной работы самой анимации. Результат второго этапа декомпозиции приведен на рисунке 2.4 и на плакате КП.0902.14.000000.01 ПЛ.



Рисунок 2.4 – Результат второго этапа декомпозиции

Структурная схема класса описывает возможные входные и выходные данные, а также реализацию преобразования этих данных. Исходя из структурной схемы класса можно описать входные и выходные данные компонента, а также методы и события.

## 2.3 Разработка графического интерфейса компонента

Общими принципами построения графического интерфейса мобильного компонента являются компактность, удобность и минималистичность. Простой и интуитивно понятный интерфейс лежит в основе хорошо разработанных приложений. Чтобы разработать интуитивно-понятный интерфейс, требуется изучить цели и точки зрения людей, чтобы иметь возможность представить содержание, которое может понравиться обществу. Эмпатия — важный инструмент в этом исследовании, потому что он помогает понять, как люди с разными точками зрения могут реагировать на содержание и впечатления, которые создаются благодаря интерфейсу. Хотя точка зрения каждого человека представляет собой уникальное пересечение человеческих качеств, которые являются одновременно отличными и динамичными. Все точки зрения возникают из человеческих характеристик и опыта, которые все разделяют, по [8]. Придерживаясь данных принципов, был построен графический интерфейс компонента в данном курсовом проекте.

В неигровом приложении, как и в случае данного курсового проекта, чрезмерное использование цвета может сделать общение менее четким и отвлекать. Следует использовать цветовые штрихи, чтобы привлечь внимание к важной информации или показать взаимосвязь между частями интерфейса.Также следует использовать цвет последовательно во всем интерфейсе, особенно когда он используется для передачи такой информации, как статус или интерактивность. Даже когда компонент сообщает об интерактивности с помощью визуального индикатора, который не зависит от цвета, например шеврона или значка стрелки, использование другого цвета для интерактивного текста сбивает с толку, из материалов [9].

Вследствие того, что вид элементов компонента пользователь может менять самостоятельно, графический интерфейс будет детально описан с настройками по умолчанию. По умолчанию, список отображаемых элементов состоит из 3 изображений, что не делает экран слишком нагруженным, но и не оставляет ощущение пустоты. Набор отображаемых на экране элементов списка располагается в середине заданной зоны, исходя из области экрана, наиболее удобной при использовании телефона одной рукой, по [10]. Текущий элемент списка - фотография, которой будут управлять, расположен на самом верху и имеет наибольший размер, занимающий половину из доступного по горизонтали и вертикали. Остальные элементы расположены под главным, а для их видимости – каждый из них смещен на 5% вниз от высоты предыдущей фотографии.

## 2.4 Описание входных и выходных данных компонента

В результатах подраздела 2.2, была получена структурная схема компонента, из которой следует необходимость присутствия тех или иных переменных и событий для полноценной работы компонента. Для удобства все свойства сведены в таблице 2.1.

Таблица 2.1 – Свойства компонента

| Имя свойства | Тип данных | Тип доступа | Метод доступа на чтение | Метод доступа на запись |
| --- | --- | --- | --- | --- |
| minAlpha | Float | Чтение/Запись | Чтение файла разметки | Описывается в файле разметки, созданным прикладным программистом |
| swipeThreshold | Float | Чтение/Запись | Чтение файла разметки | Описывается в файле разметки, созданным прикладным программистом |
| rotationDegrees | Float | Чтение/Запись | Чтение файла разметки | Описывается в файле разметки, созданным прикладным программистом |

Окончание таблицы 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя свойства | Тип данных | Тип доступа | Метод доступа на чтение | Метод доступа на запись |
| elementsCount | Int | Чтение/Запись | Чтение файла разметки | Описывается в файле разметки, созданным прикладным программистом |
| elementRadius | Float | Чтение/Запись | Чтение файла разметки | Описывается в файле разметки, созданным прикладным программистом |

События компонента представлены в таблице 2.2

Таблица 2.2 – События компонента

| Имя события | Тип данных | Тип доступа | Метод доступа на чтение |
| --- | --- | --- | --- |
| Событие onTouchListener | OnTouchListener | Чтение/Запись | Событие перемещения фотографии по компоненту. Необходим для анимации перетаскивания фотографии. |

## 2.5 Разработка алгоритмов компонента

В результатах подраздела 2.2 получена структурная схема компонента, из которой следует необходимость присутствия тех или иных методов для полноценной работы компонента. В соответствие с этой структурной схемой в данном разделе будет рассмотрена разработка алгоритмов всех методов.

getImageSizes – метод предназначен для получения размеров главного изображения компонента в зависимости от размеров самого компонента. Входными данными метода являются заданные прикладным программистом размеры компонента в разметке. Выходными данными являются размеры для текущего элемента списка. Блок-схема алгоритма метода getImageSizes приведена на рисунке 2.5



Рисунок 2.5 – Блок-схема алгоритма метода getImageSizes

getValidValue – метод предназначен для получения действительных входных параметров от пользователя. Входными параметрами является значение, указанное пользователем, минимальное значение для данного свойства и максимальное значение для данного свойства. Блок-схема алгоритма метода getValidValue приведена на рисунке 2.6.

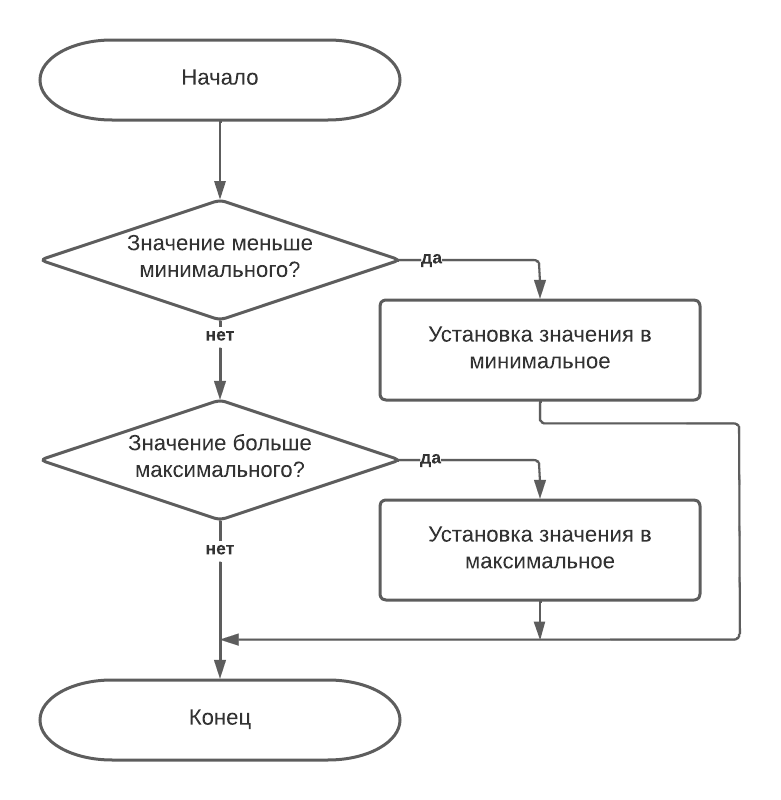


Рисунок 2.6 – Блок-схема алгоритма метода getValidValue

createImageView – метод предназначен для создания объекта, в который будет загружена фотография по ссылке из списка всех фотографий. Входными данными является порядковый номер фотографии в списке. Выходными данными является созданный объект. Блок-схема алгоритма метода createImageView приведена на рисунке 2.7.

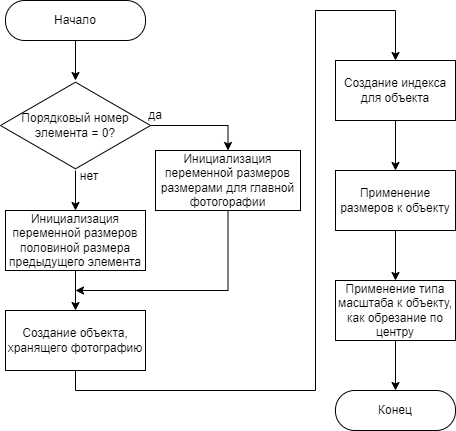


Рисунок 2.7 – Блок-схема алгоритма метода createImageView

createCardView – метод предназначен для создания подложки для хранения объекта, в которой хранится фотография. Входными данными является порядковый номер фотографии в списке. Выходными данными является созданная подложка. Блок-схема алгоритма метода createCardView приведена на рисунке 2.8.



Рисунок 2.8 – Блок-схема алгоритма метода createCardView

createElement – метод предназначен для объединения методов createImageView и createCardView. Входными данными является порядковый номер фотографии в списке. Выходных данных не имеется. Блок-схема алгоритма приведена на рисунке 2.9.

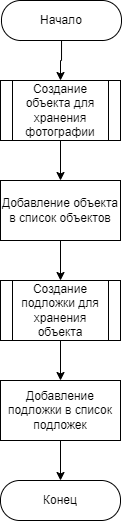


Рисунок 2.9 – Блок-схема алгоритма метода createElement

removeMainImage – метод предназначен для удаления текущего элемента списка из него и обновления отображаемых элементов в компоненте. Входных и выходных данных не имеется. Блок-схема алгоритма приведена на рисунке 2.10.



Рисунок 2.10 – Блок-схема алгоритма метода removeMainImage

loadImage – метод предназначен для загрузки фотографий, по ссылкам из списка, в отображаемые компонентом объекты. Входными данными является список строковых ссылок на фотографии. Выходных параметров не имеется. Блок-схема алгоритма приведена на рисунке 2.11 и на КП.0902.14.000000.02 ПЛ.

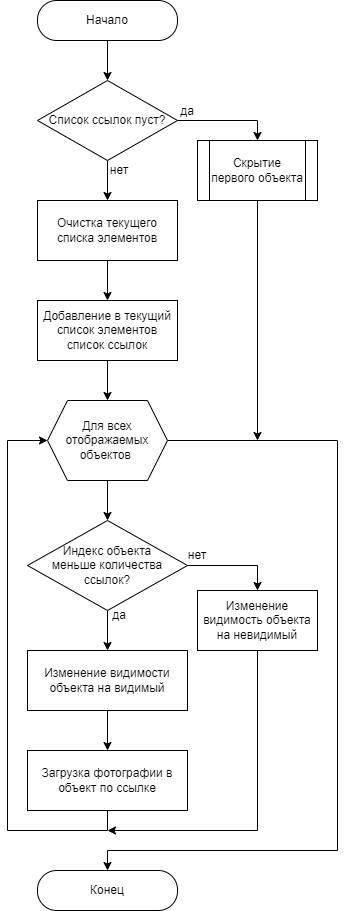


Рисунок 2.11 – Блок-схема алгоритма метода loadImage

hideImageCard – метод предназначен для установки невидимости подложки, в которой хранится объект, хранящий фотографию. Входными данными является порядковый номер элемента фотографии в списке. Выходных данных не имеется. Блок-схема алгоритма приведена на рисунке 2.12.

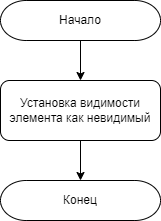


Рисунок 2.12 – Блок-схема алгоритма метода hideImageCard

getComponentSizes – метод предназначен для получения размеров компонента. Входные данные отсутствуют. Выходные данные – ширина и высота компонента. Блок-схема алгоритма приведена на рисунке 2.13.

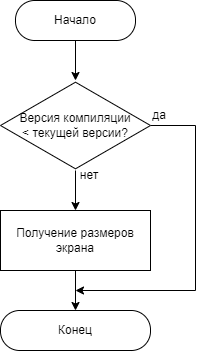


Рисунок 2.13 – Реализация метода getComponentSizes

В результате по разработанным алгоритмам, описанным в данном подразделе, был написан программный код класса Tinder, файла разметки tinder\_layout и attrs приведенные в Приложениях А, Б и В к пояснительной записке.

# 3 Технологический раздел

## 3.1 Отладка программного кода класса

Отладка – этап разработки [компьютерной программы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), на котором обнаруживают, локализуют и устраняют ошибки, по [11]. Отладка компонента будет происходить по принципу тест-кейса. Тест-кейс – проверка работоспособности программы при помощи создания текстового описания процесса тестирования какой-то части или функции проекта, по [12]. Выбор принципа отладки был сделан в пользу тест-кейсов в связи с тем, что они являются одним из наиболее популярных способов отладки проекта и содержат в себе письменное описание принципа работы функции. Тест-кейс для свойства minAlpha приведен в таблице 3.1.

Таблица 3.1 – Тест-кейс свойства minAlpha

|  |  |
| --- | --- |
| Поле | Значение |
| Номер | 1 |
| Название | Открыт проект с встроенным компонентом, должен быть использован метод loadImages хотя бы с 1 ссылкой на фотографию |
| Предусловия тест-кейса | Открыт проект с встроенным компонентом |
| Порядок действий | 1. Перейти в файл разметки 2. Поменять значение minAlpha на 0.5 3. Скомпилировать проект 4. После загрузки элементов, перенести верхнюю фотографию по горизонтали направо до конца экрана |
| Ожидаемый результат | Непрозрачность фотографии установлена в 50% |

Тест-кейс для свойства swipeThreshold приведен в таблице 3.2.

Таблица 3.2 – Тест-кейс для свойства swipeThreshold

| Поле | Значение |
| --- | --- |
| Номер | 2 |
| Название | Открыт проект с встроенным компонентом, должен быть использован метод loadImages хотя бы с 1 ссылкой на фотографию |

Окончание таблицы 3.2

|  |  |
| --- | --- |
| Поле | Значение |
| Предусловия тест-кейса | Открыт проект с встроенным компонентом, установлена любая настройка анимации |
| Порядок действий | 1. Перейти в файл разметки 2. Поменять значение swipeThreshold на -100 3. Скомпилировать проект 4. После загрузки элементов, немного перенести верхнюю фотографию по горизонтали в любую сторону |
| Ожидаемый результат | Анимация проиграна полностью |

Тест-кейс для свойства rotationDegrees приведен в таблице 3.3.

Таблица 3.3 – Тест-кейс для свойства rotationDegrees

|  |  |
| --- | --- |
| Поле | Значение |
| Номер | 3 |
| Название | Проверка работоспособности свойства rotationDegrees |
| Предусловия тест-кейса | Открыт проект с встроенным компонентом, должен быть использован метод loadImages хотя бы с 1 ссылкой на фотографию |
| Порядок действий | 1. Скомпилировать проект 2. После загрузки элементов, перенести верхнюю фотографию по горизонтали направо до конца экрана |
| Ожидаемый результат | Фотография повернута на 30 градусов в сторону перемещения |

Тест-кейс для свойства elementsCount приведен в таблице 3.4.

Таблица 3.4 – Тест-кейс для свойства elementsCount

|  |  |
| --- | --- |
| Поле | Значение |
| Номер | 4 |
| Название | Проверка работоспособности свойства elementsCount |
| Предусловия тест-кейса | Открыт проект с встроенным компонентом, должен быть использован метод loadImages хотя бы с 1 ссылкой на фотографию |

Окончание таблицы 3.4

|  |  |
| --- | --- |
| Поле | Значение |
| Порядок действий | 1. Открыть файл разметки 2. Поменять значение свойства elementsCount на 1 3. Скомпилировать проект 4. Дождаться загрузки элементов |
| Ожидаемый результат | Отображается 1 фотография в компоненте |

Тест-кейс для свойства elementRadius приведен в таблице 3.5.

Таблица 3.5 – Тест-кейс для свойства elementRadius

|  |  |
| --- | --- |
| Поле | Значение |
| Номер | 5 |
| Название | Проверка работоспособности свойства elementRadius |
| Предусловия тест-кейса | Открыт проект с встроенным компонентом, должен быть использован метод loadImages хотя бы с 1 ссылкой на фотографию |
| Порядок действий | 1. Открыть файл разметки 2. Поменять значение свойства elementRadius на 1 3. Скомпилировать проект 4. Дождаться загрузки элементов |
| Ожидаемый результат | Отображение фотографии предоставлено в виде круга |

Тест-кейс для метода loadImages приведен в таблице 3.6.

Таблица 3.6 – Тест-кейс для метода loadImages

|  |  |
| --- | --- |
| Поле | Значение |
| Номер | 5 |
| Название | Проверка работоспособности метода loadImages |
| Предусловия тест-кейса | Открыт проект с встроенным компонентом, свойство elementCount установлен по умолчанию |
| Порядок действий | 1. В классе, использующем компонент, добавить метод loadImages для экземпляра класса Tinder 2. Во входные данные компонента передать список с 3 ссылками на фотографии 3. Скомпилировать проект 4. Дождаться загрузки элементов |

Окончание таблицы 3.6

|  |  |
| --- | --- |
| Поле | Значение |
| Ожидаемый результат | Компонент отображает 3 фотографии |

## 3.2 Инструкция по установке компонента в среду разработки

Для установки компонента в проект необходимо скопировать класс компонента в директорию проекта. В открывшемся окне следует нажать кнопку «OK», пример приведен на рисунке 3.1.

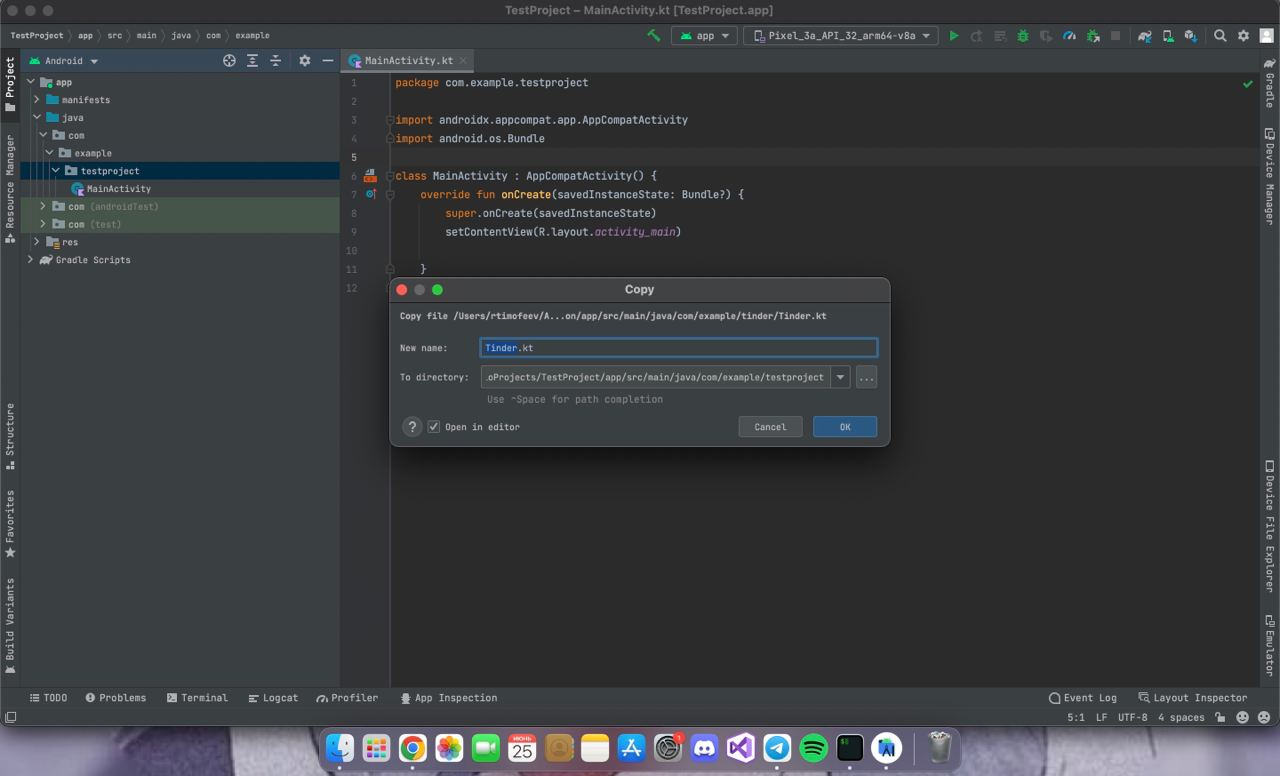


Рисунок 3.1 – Копирование класса в директорию проекта

После чего, точно такую же операцию необходимо проделать с файлом разметки компонента и файлом разметки атрибутов.

После выполнения данных действий визуальный компонент будет доступен для работы в прикладной программе через файл разметки. Для обращения к компоненту в классе следует его импортировать, написав его название и нажав комбинацию клавиш «Ctrl + Enter».

## 3.3 Инструкция по эксплуатации компонента

В результатах подраздела 2.2, была получена структурная схема компонента, из которой следует возможность использования свойств и методов класса прикладным программистом. Для удобства инструкция по эксплуатации визуального компонента приведена в табличном виде. Свойства компонента описаны в таблице 3.7.

Таблица 3.7 – Свойства компонента

|  |  |  |  |
| --- | --- | --- | --- |
| Имя свойства | Тип данных | Тип доступа | Назначение |
| minAlpha | Float | Чтение/Запись | Свойство предназначено для указания минимального значения непрозрачности фотографии во время анимации. По умолчанию установлено значение 0.2. Чтение свойство возможно только из разметки. |
| swipeThreshold | Float | Чтение/Запись | Свойство предназначено для указания точки окончания действия анимации. По умолчанию указана половина от ширины фотографии. Чтение свойство возможно только из разметки. |
| rotationDegrees | Float | Чтение/Запись | Свойство предназначено для указания максимального угла поворота фотографии во время анимации. По умолчанию установлено значение 30. Чтение свойство возможно только из разметки. |
| elementsCount | Int | Чтение/Запись | Свойство предназначено для указания максимального значения отображаемых фотографий компонентом. По умолчанию установлено значение 3. Чтение свойство возможно только из разметки. |
| elementRadius | Float | Чтение/Запись | Свойство предназначено для указания процента закругления отображаемых фотографий. По умолчанию установлено значение 0.25. Чтение свойство возможно только из разметки. |

Описание публичных методов компонента приведено в таблице 3.8.

Таблица 3.8 – Методы компонента

|  |  |  |
| --- | --- | --- |
| Имя события | Входные параметры | Назначение |
| loadImage | список ссылок на фотографии типа List<String> | Загружает фотографии по ссылкам из списка в список компонента |

Событий, доступных прикладному программисту не имеется, так как в текущей версии компонента публичных событий не предусмотрено.

# 4 Раздел охраны труда

Охрана труда – это целая система законодательных и нормативно-правовых актов, технических, гигиенических, лечебно-профилактических мероприятий и средств, которые обеспечивают безопасность, сохранение здоровья и работоспособности человека в процессе труда. В наши дни труд стал более интенсивным и требует огромных затрат умственной, эмоциональной и физической нагрузок.

На рабочем месте программист осуществляет трудовую деятельность и проводит большую часть рабочего времени. Правильная организация рабочего места программиста повышает производительность труда от 8 до 20%. Следуя рекомендациям ГОСТ 12.2.032-78, необходимо организовать рабочее место таким образом, чтобы взаимное расположение всех его элементов соответствовало физическим и психологическим требованиям. Главные элементы рабочего места программиста – это письменный стол и кресло. Рабочее место организуется в соответствии с ГОСТ 12.2.032-78.

Площадь рабочего места с компьютером с жидкокристаллическим или плазменным экраном должна быть не менее 4,5 кв. м, а расстояние между столами с мониторами (от тыла одного монитора до экрана другого) не менее 2 м. Монитор должен располагаться на расстоянии 50-70 см от глаз программиста. Параметры рабочего стола сотрудника: возможность регулировки высоты рабочего стола, или точная высота — 72,5 см, ширина — 80, 100, 120 или 140 см, глубина рабочего стола 80 или 100 см, высота и ширина пространства под столешницей (для ног) – не менее 50 см, глубина на уровне колен не менее 45 см, а на уровне вытянутых ног не менее 65 см.

Правильное освещение рабочего места – это очень важный момент в трудовой деятельности человека, влияющий на эффективность труда, при этом такой момент предупреждает травматизм и профессиональные заболевания. При недостаточном освещении приходится напрягать зрение, при этом ослабляется внимание и это приводит к наступлению преждевременной утомленности. Слишком яркое освещение тоже плохо, так как оно вызывает ослепление, раздражение и резь в глазах. При искусственном освещении, источниками света служат два вида ламп: лампы накаливания и люминесцентные.

Известно, что шум ухудшает условия труда и оказывает вредное воздействие на организм человека. Согласно ГОСТ 12.1.003-88 «Шум для помещений расчетчиков и программистов, уровни шума не должны превышать соответственно: 71, 61, 54, 49, 45, 42, 40, 38 дБ», по [6, 7].

При работе компьютерной техники выделяется много тепла, что может привести к пожароопасной ситуации. Источниками зажигания так же могут служить приборы, применяемые для технического обслуживания, устройства электропитания, кондиционеры воздуха. Серьёзную опасность представляют различные электроизоляционные материалы, используемые для защиты от механических воздействий отдельных радиодеталей. В связи с этим, участки, на которых используется компьютерная техника, по пожарной опасности относятся к категории пожароопасных “В”. При пожаре люди должны покинуть помещение в течение минимального времени. В помещениях с компьютерной техникой, недопустимо применение воды и пены ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования. Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях, по [8].

# Заключение

В результате работы над курсовым проектом был разработан визуальный компонент для управления элементами списка при помощи жестов. Компонент представляет из себя список изображений, количество элементов которых настраивает пользователь, с возможностью пользовательских настроек как анимации, так и самих элементов, и текстовое поле, которое показывается после пролистывания всех элементов списка. Для элементов списка можно изменить смещение между ними, процент их закругления, а также их размеры, для анимации же можно изменить размещение точки невозврата и непрозрачность элемента списка при проигрывании анимации, а для текстового поля можно изменить начертание шрифта, размеры и содержание. Система анимации построена на принципе назначенных настроек, под этим подразумевается то, что при назначении определенной настройки анимации – она будет проигрываться, если же определенная настройка не была назначена, то она не будет проиграна при анимации.

Главным достоинством разработанного компонента можно считать возможность задания настроек компонента пользователем.

У компонента есть перспективы развития в плане добавления новых типов настроек анимации, а также добавления событий при перелистывании элемента в одну из сторон по горизонтали. Также, перспективой развития компонента является изменение типа списка, а именно – использование не только фотографий, но и обычных текстовых полей, видео и других элементов.

Данный компонент можно использовать на страницах, требующих положительный, либо отрицательный ответ пользователя по предложенному элементу. Также его можно использовать в учебных целях, так как компонент может показать объектно-ориентированный подход, работу с жестами, изображениями и глобальной сети Интернет.

# Список используемых источников

1. <https://ru.wikipedia.org/wiki/Транслятор>
2. <https://ru.wikipedia.org/wiki/Фортран#Возможности_и_характерные_особенности_языка>
3. <https://visualstudio.microsoft.com/ru/vs/features/mobile-app-development/>
4. <https://wnfx.ru/android-studio-ide-ot-google/>
5. <https://developer.apple.com/xcode/>
6. <https://www.retail.ru/rbc/pressreleases/tsentr-povysheniya-kvalifikatsii-lider-organizatsiya-rabochego-mesta-ofisnogo-rabotnika/>
7. <https://xn--d1aux.xn--p1ai/opisanie-rabochego-mesta-programmista-na-predpriyatii/>
8. <https://developer.apple.com/design/human-interface-guidelines/foundations/inclusion>
9. <https://developer.apple.com/design/human-interface-guidelines/foundations/color>
10. <https://habr.com/ru/company/ifree/blog/247871/>
11. <https://ru.wikipedia.org/wiki/Отладка_программы>
12. <https://solvery.io/blog/ru/interesting/chto-takoe-test-case-primer-i-check-list-dlya-nachinayushhih-testirovshhikov/>

# Приложение А

## Программный код класса Tinder

package com.example.tinder

import android.app.Activity

import android.content.Context

import android.graphics.PointF

import android.os.Build

import android.util.AttributeSet

import android.util.Log

import android.view.\*

import android.widget.ImageView

import androidx.cardview.widget.CardView

import androidx.constraintlayout.widget.ConstraintLayout

import androidx.constraintlayout.widget.ConstraintSet

import androidx.constraintlayout.widget.ConstraintSet.\*

import com.bumptech.glide.Glide

import kotlin.math.abs

import kotlin.math.sign

class Tinder(

context: Context,

attrs: AttributeSet

) : ConstraintLayout(context, attrs) {

// Список элементов, хранящих фотографии.

private val imageItems = mutableListOf<ImageView>()

// Список элементов-подложек для элементов фотографий.

private val cardItems = mutableListOf<CardView>()

private val swipeThreshold: Float

private val imageAlpha: Float

private val imageRotation: Float

private val imagesCount: Int

private val imageRadiusPercent: Float

private val touchDown = PointF(0f, 0f)

private val startCardViewPosition = PointF(0f, 0f)

private val imagesList = mutableListOf<String>()

init {

fun getComponentSizes(): Pair<Int, Int> {

if (Build.VERSION.SDK\_INT < Build.VERSION\_CODES.R) return Pair(-1, -1)

val sizes = (context as Activity).windowManager.currentWindowMetrics.bounds

return Pair(sizes.right, sizes.bottom)

}

fun getImageSizes(lp: ViewGroup.LayoutParams): Pair<Int, Int> {

val componentWidth: Int

val componentHeight: Int

with(getComponentSizes()) {

componentWidth = this.first

componentHeight = this.second

}

return Pair(

when (lp.width) {

IMAGE\_WRAP\_CONTENT -> 200

IMAGE\_MATCH\_PARENT -> (componentWidth \* IMAGE\_SIZE\_PERCENT).toInt()

0 -> 0

else -> (lp.width \* IMAGE\_SIZE\_PERCENT).toInt()

},

when (lp.height) {

IMAGE\_WRAP\_CONTENT -> 200

IMAGE\_MATCH\_PARENT -> (componentHeight \* IMAGE\_SIZE\_PERCENT).toInt()

0 -> 0

else -> (lp.height \* IMAGE\_SIZE\_PERCENT).toInt()

}

)

}

this.layoutParams = LayoutParams(context, attrs)

val imageSizes = getImageSizes(this.layoutParams)

val view = inflate(context, R.layout.tinder\_layout, this)

fun <T : Comparable<T>> getValidValue(value: T, minValue: T, maxValue: T): T {

return when {

value is Float && value == -1f -> value

value < minValue -> minValue

value > maxValue -> maxValue

else -> value

}

}

// Получение списка всех указанных атрибутов

val tinderAttrs = context.obtainStyledAttributes(attrs, R.styleable.Tinder, 0, 0)

imageAlpha = getValidValue(tinderAttrs.getFloat(R.styleable.Tinder\_minAlpha, -1f), 0f, 1f)

imageRotation = getValidValue(tinderAttrs.getFloat(R.styleable.Tinder\_rotationDegrees, 30f), 0f, 30f)

swipeThreshold = getValidValue(

tinderAttrs.getFloat(R.styleable.Tinder\_swipeThreshold, (getComponentSizes().first - imageSizes.first) / 2f),

0f,

getComponentSizes().first / 2f

)

imagesCount = getValidValue(tinderAttrs.getInt(R.styleable.Tinder\_elementsCount, 1), 1, 5)

imageRadiusPercent = getValidValue(tinderAttrs.getFloat(R.styleable.Tinder\_elementsRadius, .25f), 0f, 1f)

fun createImageView(item: Int): ImageView {

val sizes =

if (item == 0) imageSizes

else Pair(

(imageItems[item - 1].layoutParams.width \* (1 - IMAGE\_SIZE\_LESS\_PERCENT)).toInt(),

(imageItems[item - 1].layoutParams.height \* (1 - IMAGE\_SIZE\_LESS\_PERCENT)).toInt()

)

return ImageView(context).apply {

id = View.generateViewId()

layoutParams = LayoutParams(sizes.first, sizes.second)

scaleType = ImageView.ScaleType.CENTER\_CROP

}

}

fun createCardView(item: Int): CardView {

return CardView(context).apply {

id = View.generateViewId()

addView(imageItems[item])

layoutParams = LayoutParams(LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT)

radius = imageItems[item].layoutParams.width \* imageRadiusPercent

}

}

fun createElement(item: Int) {

imageItems.add(createImageView(item))

cardItems.add(createCardView(item))

}

createElement(0)

with(cardItems[0]) {

this@Tinder.addView(this)

ConstraintSet().apply {

clone(view as ConstraintLayout)

connect(id, TOP, view.id, TOP)

connect(id, BOTTOM, view.id, BOTTOM)

connect(id, START, view.id, START)

connect(id, END, view.id, END)

applyTo(view)

}

}

for (idx in 1 until imagesCount) {

createElement(idx)

this.addView(cardItems[idx])

ConstraintSet().apply {

clone(view as ConstraintLayout)

connect(cardItems[idx].id, TOP, cardItems[idx - 1].id, TOP)

connect(cardItems[idx].id, START, cardItems[idx - 1].id, START)

connect(cardItems[idx].id, END, cardItems[idx - 1].id, END)

setMargin(

cardItems[idx].id,

TOP,

(imageItems[idx - 1].layoutParams.height \* IMAGE\_SIZE\_LESS\_PERCENT + imageItems[idx - 1].layoutParams.height \* IMAGE\_MARGIN\_PERCENT).toInt()

)

applyTo(view)

}

}

for (idx in imagesCount - 1 downTo 0) {

cardItems[idx].bringToFront()

}

fun removeMainImage() {

if (imagesList.isEmpty()) return

loadImage(imagesList.drop(1))

}

cardItems[0].setOnTouchListener { curView, event ->

val viewDiff = curView.x - startCardViewPosition.x

val absoluteDiff = abs(viewDiff)

val absoluteDiffPercent = absoluteDiff / swipeThreshold

when (event.action) {

MotionEvent.ACTION\_DOWN -> {

startCardViewPosition.x = curView.x

// Запоминаем координаты касания фотографии

with(touchDown) {

x = event.x

y = event.y

}

}

MotionEvent.ACTION\_MOVE -> {

// Перемещение фотографии

curView.x += event.x - touchDown.x

if (imageAlpha != -1f)

curView.alpha = if (absoluteDiffPercent < 1f) 1 - (1 - imageAlpha) \* absoluteDiffPercent else imageAlpha

//

if (imageRotation != -1f) {

val diffPercent = viewDiff / swipeThreshold

curView.rotation = if (absoluteDiffPercent < 1f) diffPercent \* imageRotation else imageRotation \* diffPercent.sign

}

}

MotionEvent.ACTION\_UP -> {

curView.x = startCardViewPosition.x

if (absoluteDiffPercent >= 1f)

removeMainImage()

if (imageAlpha != -1f) curView.alpha = 1f

if (imageRotation != -1f) curView.rotation = 0f

}

}

true

}

tinderAttrs.recycle()

}

/\*\*

\* Загружает фотографии по ссылкам из [images] в компонент.

\* @param images список ссылок на фотографии

\*/

fun loadImage(images: List<String>) {

if (images.isEmpty()) return hideImageCard(0)

with(imagesList) {

removeAll(this)

addAll(images)

}

for (idx in 0 until imagesCount) {

if (idx < images.size) {

cardItems[idx].visibility = View.VISIBLE

Glide.with(context).load(images[idx]).into(imageItems[idx])

} else hideImageCard(idx)

}

}

private fun hideImageCard(item: Int) {

cardItems[item].visibility = View.INVISIBLE

}

companion object {

// Процент размера отображаемой фотографии от размеров компонента

private const val IMAGE\_SIZE\_PERCENT = .5f

// Процент снижения размеров последующих фотографий

private const val IMAGE\_SIZE\_LESS\_PERCENT = .1f

// Процент смещения последующих фотографий

private const val IMAGE\_MARGIN\_PERCENT = .05f

private const val IMAGE\_WRAP\_CONTENT = -2

private const val IMAGE\_MATCH\_PARENT = -1

}

}

# Приложение Б

## Программный код файла разметки tinder\_layout

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/tinderLayout"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

# Приложение В

## Программный код файла разметки атрибутов attrs

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

<resources>

<declare-styleable name="Tinder">

<attr name="minAlpha" format="float"/>

<attr name="rotationDegrees" format="float"/>

<attr name="swipeThreshold" format="float"/>

<attr name="elementsCount" format="integer"/>

<attr name="elementsRadius" format="float"/>

</declare-styleable>

</resources>