МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»



**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

по выполнению лабораторных работ

по дисциплине «Программирование мобильных устройств»

для студентов направлений 09.03.03 «Прикладная информатика» и 09.03.02 «Информационные системы и технологии».

Ставрополь

2021

# **Лабораторная работа 13. Обработка жестов.**

**Цель работы:** Научиться добавлять различные виды активностей.

**Формируемые компетенции:** ПК-7

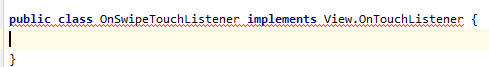
**Теоретическая часть:**

Обработка жестов является одним из важнейших компонентов при построении мобильных приложений. Однако, это правильно использовать в играх, а не в бизнес-приложениях, которые создаются с помощью Android Studio. Тем не менее, будет полезно узнать, как же можно отслеживать движения пальцем по экрану.

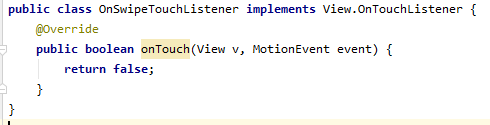
Жесты обрабатываются с помощью слушателей. Подобным образов мы делали обработку нажатия кнопки на лабораторной работе 3. Как Вы помните, мы могли использовать слушателя двумя способами: создав объект класса слушателя или использовать анонимно.

При использовании жестов лучшим вариантом будет создать класс обработчик. Это позволит один раз правильно настроить класс и затем просто использовать то, как мы обработаем движения.

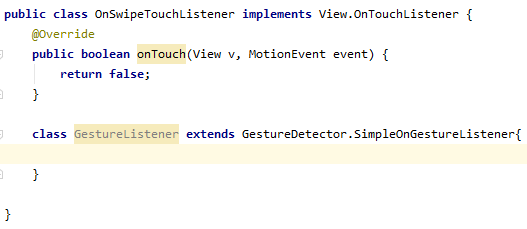
В папке с Java создадим новый класс и назовём его OnSwipeTouchListener:



Так же, как и с кликом мы должны использовать перегруженный метод. Здесь он называется onTouch:



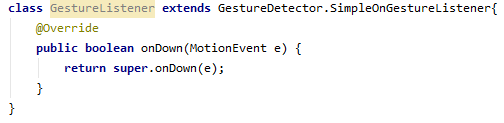
Touch – это простое прикосновение к экрану. Событие Click можно смело заменить на Touch и это будет работать. Однако, наша задача заключается в том, чтобы обработать не само касание, а некоторое движение пользователя по экрану. Для этого используется уже другой обработчик. Внутри нашего класса создадим вложенный класс (как в лабораторной работе 1), который будет называться GestureListener:



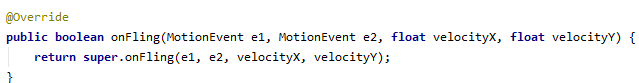
Данный класс будет обрабатывать движения. Сам процесс движения заключается в следующем:

1. Пользователь ставит палец на экран и срабатывает событие onTouch;
2. Пользователь двигает палец по экрану в какую-нибудь сторону;
3. При движении мы можем получить основную информацию, связанную с движением:
   1. Где находился до начала движения;
   2. Где стал находиться после начала движения;
   3. Скорость, с которой двинулся по оси X;
   4. Скорость, с которой двинулся по оси Y.

Добавим использование этих подходов. Во-первых, добавим перегрузку метода onDown (пункт 1):

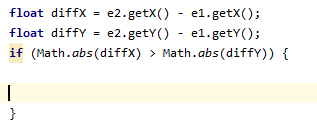


Следующим шагом определим движение. Перегружаем метод onFling:



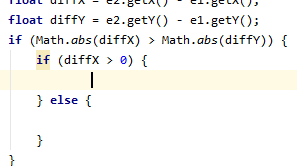
4 параметра, передаваемые в данный метод – та самая информация из пункта 3 (a, b, c, d).

Напишем простой метод, который будет смотреть, куда двинулся палец. Начнём с начала:

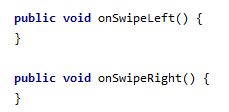


Решим для начала, по какой оси двигался палец: Если больше по X, то влево или вправо, в противном случае – вверх/вниз.

Самый простой вариант будет работать так: если хотя бы на пиксель сдвинулось прикосновение, то в ту сторону и сработало. Если diffX > 0, то по X смещение ушло в правую сторону, иначе – влево:

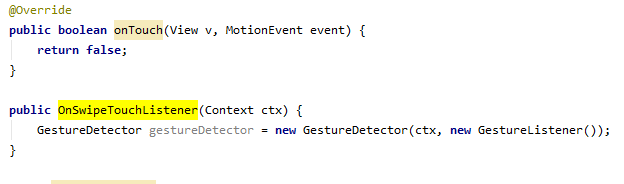


Добавим два пустых метода, которые мы позже переопределим:

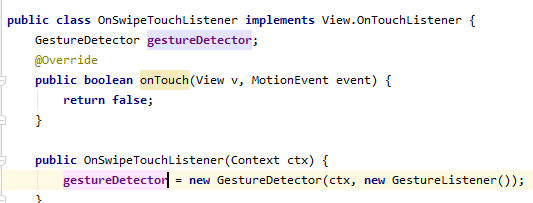


Они должны находиться в классе OnSwipeTouchListener. Поместим их вызовы внутри if/else.

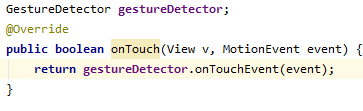
Теперь создадим простую реализацию данного класса. В первом слушателе добавим конструктор:



Вынесем переменную gestureDetector в качестве свойства:



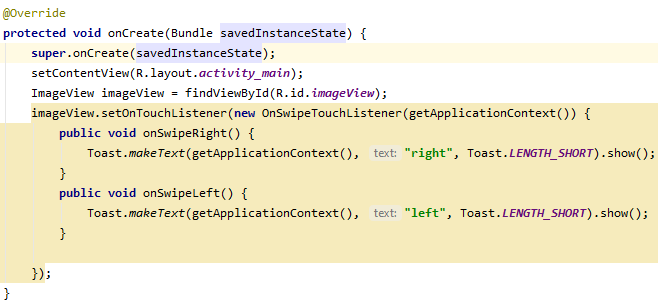
И используем его в методе onTouch:



Теперь идём к нашей активности. Создадим простой квадрат по центру экрана:



В методу onCreate навесим нашего слушателя на движения пальца по imageView:



Запустим приложение. Начнём движение с квадрата, двинем немного мышку (палец) влево и отпустим. Должно появиться сообщение, “left”. Проделав то же самое, но вправо – увидим, что появилось сообщение “right”. Подобным образом можем добавить и движения вверх и вниз.

Естественно, можно добавить ограничения. Например, чтобы свайп срабатывал только тогда, когда палец «пройдёт» некоторое количество пикселей. Это позволит избежать случайных движений. Можно отслеживать и скорость движения.

Координаты движения представлены в объектах MotionEvent.

**Задание:** Согласно заданию, выбранному на предыдущей лабораторной работе реализовать механизм работы обработки движений.

**Оборудование и материалы:** для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 8 (8.1, 10) и программным продуктом: AndroidStudio.

**Указания по технике безопасности:** к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

**Содержание отчета:** отчет по лабораторной работе должен быть выполнен в редакторе MSWord и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выровнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

**Контрольные вопросы:**

1. Окна

**Список литературы, рекомендуемый к использованию по данной теме:**

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book\_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book\_red&id=442808&sr=1