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 3. Использование механизма событий Android.

**Цель работы:** Научиться разбираться с механизмом обработки клика.

**Формируемые компетенции:** ПК-7, ПК-8

**Теоретическая часть:**

**Элементы управления** – это элементы, которые располагаются на экране телефона и позволяют вам взаимодействовать с вашей программой. Они включают менеджеры размещений, кнопки, галочки, текстовые блоки, прогресс бар и многое др. О них было упомянуто в первой лабораторной работе.

На предыдущих лабораторных работах мы научились создавать некоторые графические элементы и элементы управления, а также размещать их на экране определенным образом. Теперь нам необходимо научиться обрабатывать взаимодействие пользователя с устройством. Например, при нажатии на кнопку, возможно, вам нужно будет выполнить какие-либо вычисления или отправить какие-либо данные на сервер для обработки. Для этого существует механизм событий. Событий, на которые мы можем реагировать достаточно много, при этом каждое событие привязано к конкретному элементу. Правильнее будет даже сказать, что у каждого элемента есть список событий, которые он может генерировать. При нажатии на кнопку генерируется событие *onClick.*  Механизм событий позволяет вам подписываться на события. Эта подписка заключается в указании метода, который будет запускаться в ответ на генерацию события. Такой метод называется обработчиком события.

**Code-behind**

Как вы уже поняли, основной (статический) дизайн приложения разрабатывается с помощью *XML* в файле *activity\_main.xml*. Это так называемое **внешнее представление** вашего приложения, т.е. то, что видит пользователь в момент запуска. В тоже время MainActivity.java используется для проектирования динамики приложения: в нем описывается обработка событий, выполнение вычислений, проектирование взаимодействия между пользователем и программой. Это **внутреннее представление приложения**, скрытое от глаза пользователя. В связи с этим появилось понятие ***Code-Behind*** (с англ. «код позади», читается «**код бихайнд**»), которое обозначает именно код, который «обслуживает» внешнее представление. Этим понятием мы будем обозначать код, описывающий класс страницы с ее свойствами, методами и обработчиками событий.

Элементов управления множество, и каждый может генерировать определенный набор событий. Многие элементы имеют пересекающийся список событий в связи с тем, что они наследуют их от общих предков в иерархии наследования. Для того чтобы посмотреть какие события может генерировать элемент, можно использовать технологию IntelliSense. Для этого достаточно установить курсор на определение элемента в XML и нажать сочетание клавиш ctrl + пробел. Откроется список возможных атрибутов для использования.

**Динамическое управление подпиской на события.**

С помощью XML мы можем декларативно описать какой обработчик на какое событие реагирует. Однако, XML удобен в данном случае, лишь как точка входа в приложение, т.е. в качестве описания того, что пользователь увидит при запуске приложения. Когда же нам необходимо более гибкое динамическое поведение на помощь приходит code-behind. Динамическое поведение здесь подразумевает динамическую подписку на события и отписку от них. Например, когда при выполнении некоторых условий необходимо, чтобы обработчик не реагировал на событие (и при нажатии на кнопку ничего не происходило). К тому же code-behind позволяет подписывать на одно событие сразу несколько обработчиков, что в XML невозможно. В общем, давайте научимся работать с событиями в code-behind.

**Ход лабораторной работы:**

1. Создайте новый проект.
2. Разместите на главной странице под текстовым блоком, созданным по умолчанию, кнопку 175\*75 с текстом «Кнопка». Оберните и текст, и кнопку тегом <LinearLayout> с вертикальной ориентацией. Должно получиться следующим образом:

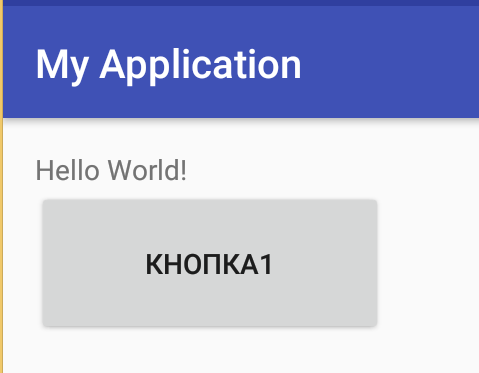


Рисунок 1 - Использование вертикальной ориентации

1. Зададим обработчик события *onClick*. Для этого добавим атрибут *onClick* к элементу *Button.* В отличие от Visualstudio, AndroidStudio не может сама генерировать обработчики событий. Поэтому, необходимо создать таковой:



Рисунок 2 - Пример создания обработчика

1. Примените созданный обработчик для кнопки. Код кнопки в итоге будет выглядеть следующим образом:

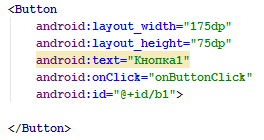


Рисунок 3 - Исходный код кнопки

Значением атрибута *onClick*теперь является строка *onButtonClick*, которая является названием метода-обработчика класса *MainActivity*.

1. Чтобы при нажатии на кнопку вывести сообщение на экран, добавьте в этот метод следующее выражение:



Рисунок 4 - Пример всплывающего события

Метод makeText – создаёт сообщение, где первым параметром указывается this, вторым – текст сообщения, далее идёт длительность сообщения: LENGTH\_SHORT – устанавливает длительность 2.5 секунды, LENGTH\_LONG – 3.5 секунды. Метод show() – заставляет показать созданное сообщение.

Запустите приложение, проверьте, как работает нажатие на кнопку.

**Идентификаторы элементов управления**

Вы можете давать имена элементам управления с помощью атрибута id:



Рисунок 5 - Присвоение идентификатора

В результате вы получите программный доступ к данному элементу в коде файла MainActivity.java под именем b1, например, можно задать текстовое значение кнопки (задается с помощью метода setText()):



Рисунок 6 - Обращение по идентификатору

Если быть более точными при указании атрибута id, анализатор XML создает свойство в классе MainActivity с соответствующим идентификатором.

1. Выполним еще пару упражнений для закрепления. Удалите значение атрибута Clickв кнопке. Теперь чтобы добавить обработчик события, вы должны указать его имя. Вы можете сделать это вручную (просто вписать имя метода), либо с помощью технологии *IntelliSense (читается «ИнтелиСэнс»)*, нажав на сочетание клавиш Ctrl+Пробел (курсор в этом случае должен быть после первой кавычки значения атрибута). При использовании *IntelliSense,* вам представится выбор из существующих обработчиков.

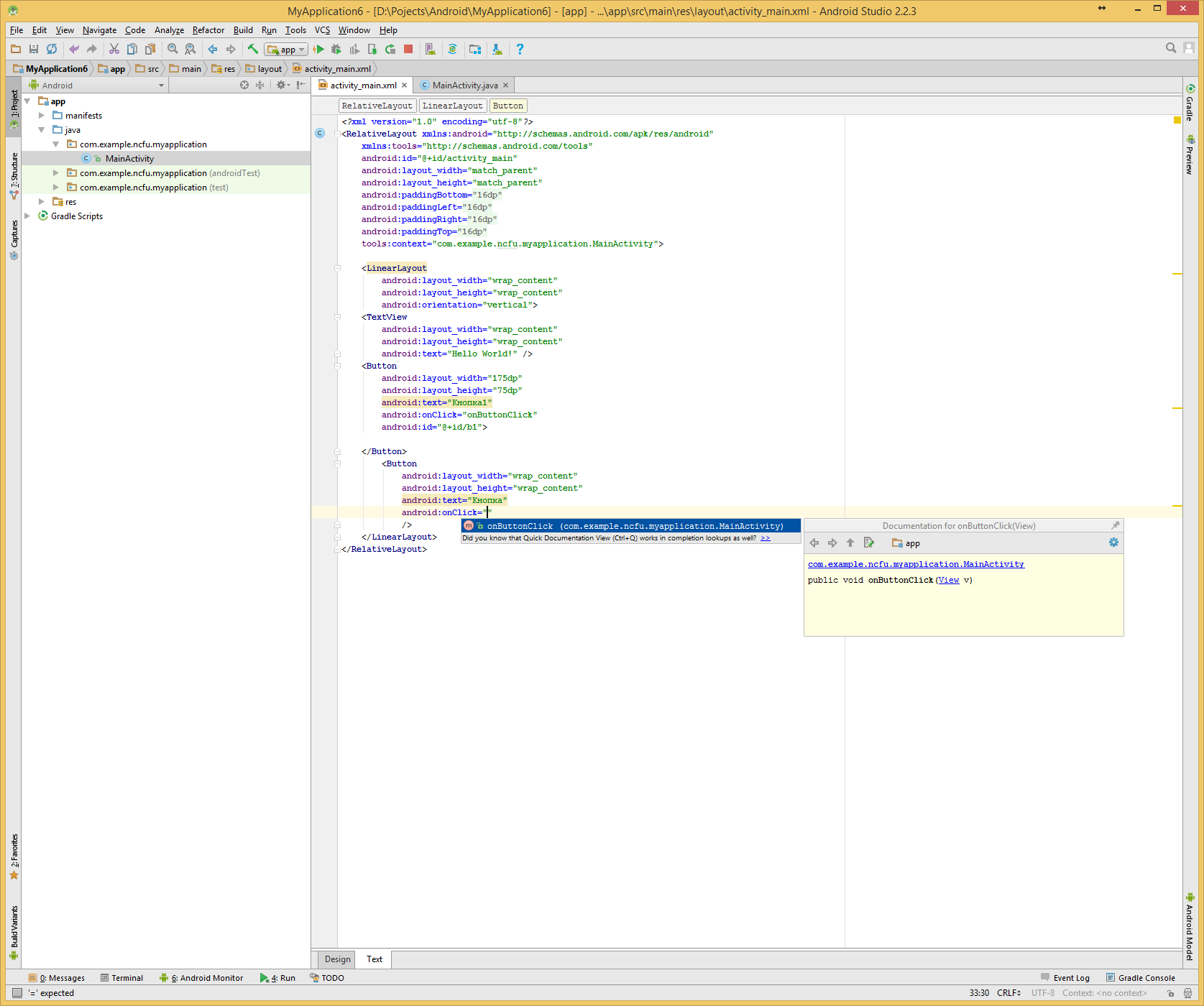


Рисунок 7 - Добавление обработчика в XML

1. Создайте новый обработчик и проделайте еще раз шаг 6, чтобы убедиться, что в качестве выбора вам представится уже 2 обработчика.
2. Создайте в классе MainActivityеще один метод с произвольным именем, но с такой же **сигнатурой**, как и метод onButtonClick.

Сигнатура метода представляет собой спецификацию метода, включающую информацию о типе возвращающего значения и типах его аргументов. Например, сигнатуру метода:

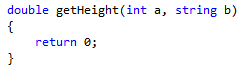


Рисунок 8 – Пример 1

можно представить как **double (int, string)**. Сигнатура не включает в себя название метода и названия его аргументов, только их типы. Если говорят, что методы имеют одинаковую сигнатуру, значит методы возвращают значения одно и того же типа и принимают аргументы тех же типов в той же последовательности. Для указанного выше метода, такую же сигнатуру будет иметь следующий метод:

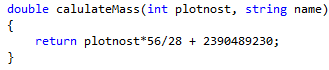


Рисунок 9 - Пример 2

Еще раз обратите внимание на то, что для сигнатуры не важны имя метода и имена его аргументов. Значение имеют только типы аргументов и тип.

1. Тело вашего метода должно содержать код, выводящий на экран сообщение с произвольным числом от 1 до 100.
2. Проделайте шаг 7 и убедитесь, что в предложенных вариантах обработчиков события *onClick* присутствует ваш новый метод. **Если вы создадите метод с другой сигнатурой, он в списке не отобразится.**
3. Рассмотрим, как можно сделать задание 3 с использованием аргумента *View.*

Взгляните еще раз на обработчик события, который подписан на событие *onClick* кнопки.  
При генерации события вобработчик события передается ссылка на объект, генерировавший событие. Эта ссылка хранится в аргументе *view*. Но как видно из сигнатуры обработчика, этот аргумент имеет тип View, а кнопка имеет тип Button. Нам необходимо явное приведение типов:



Рисунок 10 - Изменение содержимого кнопки

В первой строке мы приводим ссылку к типу Buttonи сохраняем ее в переменную b1. Во второй строке мы обращаем к методуsetTextуже приведенного типа. Запись можно сократить следующим образом:



Рисунок 11 - Краткая запись

1. Создайте новый проект. Добавьте на главную страницу кнопку. Задайте ей имя b.



Рисунок 12 - Использование кнопки в java

1. К сожалению, события в XMLи события в java подписываются по-разному.Если в XML необходимо передавать Viewи возвращать void, то здесь необходимо следовать правилам:

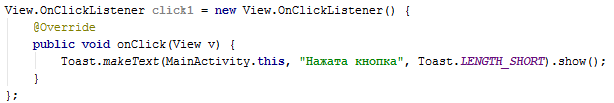


Рисунок 13 - Создание простого события

Разберёмся подробнее. Во-первых, необходимо понять, что здесь событие – такой же тип данных, как, например, int или string. Следовательно, необходимо объявить переменную типа View.OnClickListener. Во-вторых, внутри необходимо переопределить метод onСlick(). Например, если мы хотим, чтобы и в java и в XML при нажатии генерировались одинаковые события, то мы можем поместить вызов одного метода внутри другого:

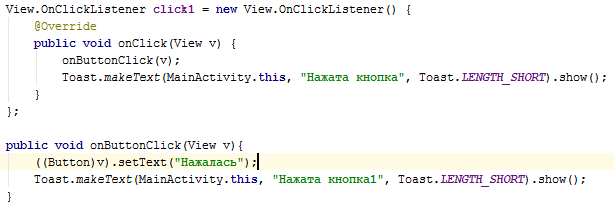


Рисунок 14 - Добавление события

1. В конструкторе страницы(code-behind) наберите:

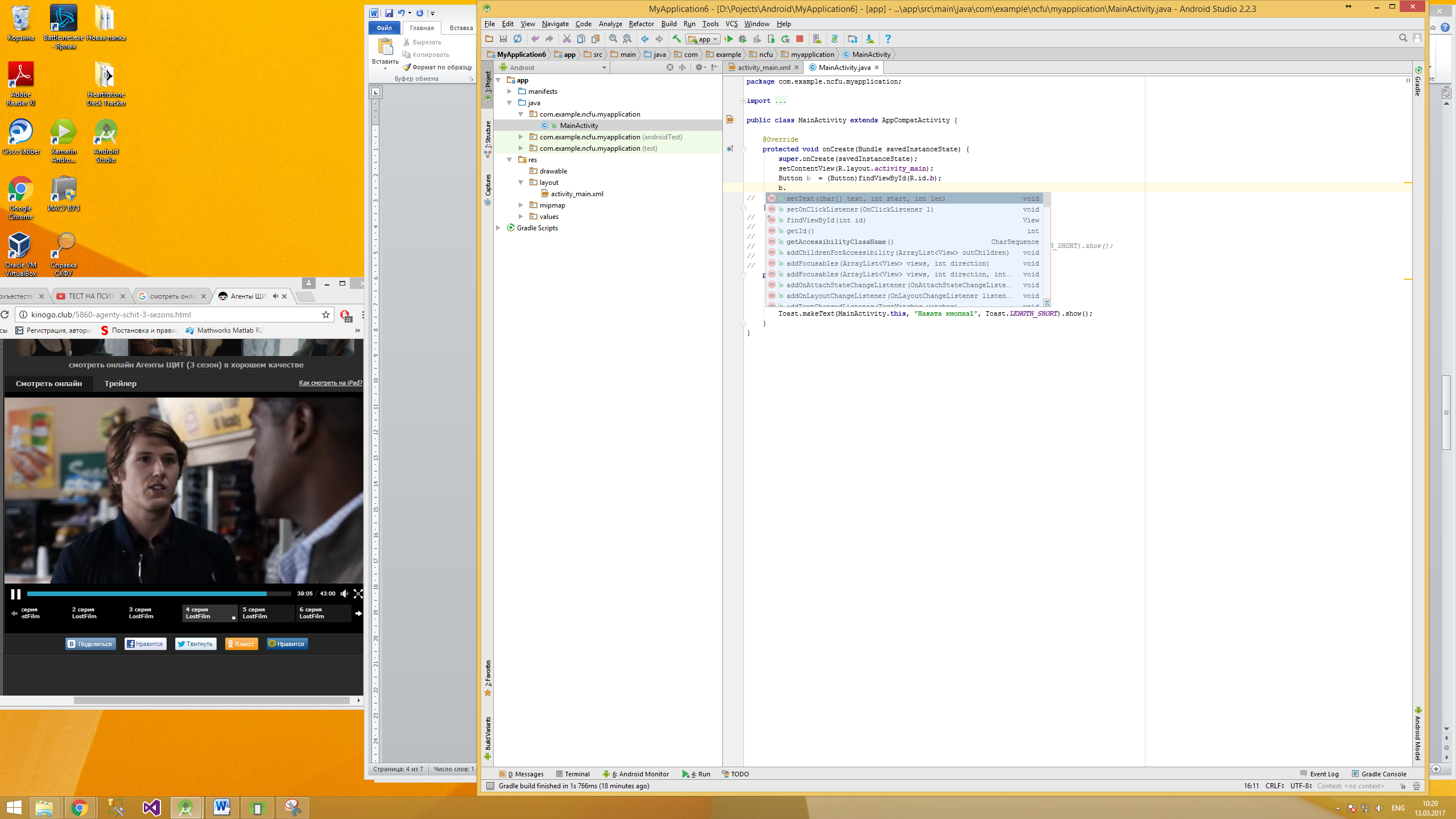


Рисунок 15 - Добавление слушателя

Выберите событие setOnClickListener. Нажмите Enter. Передайте в качестве параметра нашу объявленную переменнуюclick1.



Рисунок 16 - Применение события

1. В сгенерированном методе выведите сообщение на экран о том, что у вас все получилось.

Итак, для подписки на событие в коде необходимо использовать следующую схему:

Объект.УстановитьСлушателя(имяОбработчика);

Пример: b.setOnClickListener(click1);

Для отписки от события используем следующую запись

Объект. УстановитьСлушателя(null);

Пример: b.setOnClickListener(null);

**Задания:**для выполнения лабораторной работы необходимо выполнить следующее:

1. Сделайте так, чтобы при нажатии на кнопку на экране отображалась ваша фамилия, как это вы делали на лабораторных работах 1 и 2.
2. Добавьте еще одну кнопку, и сделайте так, чтобы при нажатии на эту кнопку текст на ней менялся бы на текст вашей фамилии.
3. Создайте 3 кнопки. Сделайте так, чтобы при нажатии на одну из кнопок, выводилась ваша фамилия именно на той кнопке, которая была нажата (Примечание: необходимо для каждой кнопки добавить своё событие).
4. Выполните задание 3, используя аргумент view.
5. Добавьте в новое приложение 2 кнопки. Сделайте так, чтобы на экран выводилось сообщение с вашей фамилией либо при нажатии одной кнопки, либо при нажатии другой попеременно.

**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 выравнен 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. CodeBehind
2. Событие
3. Слушатели

**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