**Введение**

Кнопка - один из самых распространенных элементов управления в программировании. Наследуется от **TextView** и является базовым классом для класса **СompoundButton**. От класса **CompoundButton** в свою очередь наследуются такие элементы как **CheckBox**, **ToggleButton** и **RadioButton**. В Android для кнопки используется класс **android.widget.Button**. На кнопке располагается текст и на кнопку нужно нажать, чтобы получить результат. Альтернативой ей может служить компонент [ImageButton](http://developer.alexanderklimov.ru/android/views/imagebutton.php) (**android.widget.ImageButton**), у которого вместо текста используется изображение.

В студии кнопка представлена компонентом **Button** в разделе **Widgets**. Управлять размером шрифта, цветом текста и другими свойствами можно через атрибут **textAppearance**, который задействует системные стили. Выпадающий список данного свойства содержит огромный перечень вариантов. Также вы можете вручную задать конкретные индивидуальные настройки через отдельные свойства.

Если вы растягиваете кнопку по всей ширине экрана (**android:layout\_width="match\_parent"**), то дополнительно рекомендую использовать атрибут **android:layout\_margin** (или родственные ему **layout\_marginRight** и **layout\_marginLeft**) для создания отступов от краев экрана (веб-мастера знакомы с этими терминами).

Так как кнопка является наследником **TextView**, то использует многие знакомые атрибуты: **textColor**, **textSize** и др.

Три способа обработки событий нажатий на кнопку

Если вы разместили на экране кнопку и будете нажимать на неё, то ничего не произойдёт. Необходимо написать код, который будет выполняться при нажатии. Существует несколько способов обработки нажатий на кнопку.

Первый способ - атрибут onClick

Относительно новый способ, специально разработанный для Android - использовать атрибут **onClick** (на панели свойств отображается как **On Click**):

android:onClick="onMyButtonClick"

Имя для события можно выбрать произвольное, но лучше не выпендриваться. Далее нужно прописать в классе активности придуманное вами имя метода, который будет обрабатывать нажатие. Метод должен быть открытым (public) и с одним параметром, использующим объект **View**. Вам нужно выучить пять слов для создания метода, а сам метод поместить в класс (если вы ещё путаетесь в структуре Java-кода, то вставьте метод перед последней фигурной скобкой):

public void onMyButtonClick(View view)

{

// выводим сообщение

Toast.makeText(this, "Зачем вы нажали?", Toast.LENGTH\_SHORT).show();

}

Когда пользователь нажимает на кнопку, то вызывается метод **onMyButtonClick()**, который в свою очередь генерирует всплывающее сообщение.

Обратите внимание, что при подобном подходе вам не придётся даже объявлять кнопку через конструкцию **(Button)findViewById(R.id.button1)**, так как Android сама поймёт, что к чему. Данный способ применим не только к кнопке, но и к другим элементам и позволяет сократить количество строк кода.

Данный способ не будет работать в фрагментах. Кнопка должна быть частью активности, а не фрагмента.

Второй способ - метод setOnClickListener()

Более традиционный способ в Java - через метод **setOnClickListener()**, который прослушивает нажатия на кнопку. Так как для начинающего программиста код может показаться сложным, то рекомендуется использовать подсказки студии. Вот как это будет выглядеть. Предположим, у вас на экране уже есть кнопка **button**. В коде вы объявляете её обычным способом:

Button button = (Button) findViewById(R.id.button);

Следующий шаг - написание метода для нажатия. Напечатайте имя элемента и поставьте точку *button.* - среда разработки покажет вам список доступных выражений для продолжения кода. Вы можете вручную просмотреть и выбрать нужный вариант, а можно продолжать набирать символы, чтобы ускорить процесс. Так как с нажатиями кнопок вам часто придётся работать, то запомните название его метода (хотя бы первые несколько символов) - набрав четыре символа (*seto*), вы увидите один оставшийся вариант, дальше можно сразу нажать клавишу Enter, не набирая оставшиеся символы. У вас появится строка такого вида:

button.setOnClickListener()

Курсор будет находиться внутри скобок и появится подсказка **OnClickListener l**. Начинайте набирать *new OnClickListener*. Здесь также не обязательно набирать имя полностью. Набрав слово *Oncl*, вы увидете нужный вариант и снова нажимайте Enter. В результате вы получите готовую заготовку для обработки нажатия кнопки:

button.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

}

});

Теперь у вас есть рабочая заготовка и сразу внутри фигурных скобок метода **onClick()** вы можете писать свой код. Рекомендую потренироваться и набить руку в создании заготовки. Это не так сложно, и с практикой навык закрепится автоматически.

Как вариант, можно вынести код для **OnClickListener** в отдельное место, это удобно, когда кнопок на экране несколько и такой подход позволит упорядочить код. Удалите предыдущий пример и начните писать код заново. Принцип такой же, немного меняется порядок. В предыдущем примере мы сразу прописали в методе **setOnClickListener** слушателя **new OnClickListener...** с методом **onClick()**. Можно сначала отдельно объявить отдельную переменную **myButtonClickListener**:

OnClickListener myButtonClickListener = new OnClickListener() {

@Override

public void onClick(View v) {

}

};

Во время набора активно используйте подсказки через Ctrl+Space. Набрали несколько символов у первого слова и нажимайте эту комбинацию, набрали после слова new несколько символов и снова нажимайте указанную комбинацию - заготовка будет создана за несколько секунд, а вы избежите возможных опечаток.

У нас есть готовая переменная, и теперь, когда вы будете набирать код **button.setOnClickListener**, то вместо **new OnClickListener** впишите готовую переменную.

Для новичка описание может показаться сумбурным и не понятным, но лучше самостоятельно проделать эти операции и понять механизм.

Третий способ - интерфейс OnClickListener

Третий способ является родственным второму способу и также является традиционным для Java. Кнопка присваивает себе обработчика с помощью метода **setOnClickListener (View.OnClickListener l)**, т.е. подойдет любой объект с интерфейсом **View.OnClickListener**. Мы можем указать, что наш класс **Activity** будет использовать интерфейс **View.OnClickListener**.

Опять стираем код от предыдущего примера. Далее после слов **extends Activity** дописываем слова **implements OnClickListener**. При появлении подсказки не ошибитесь. Обычно первым идёт интерфейс для диалогов, а вторым нужный нам **View.OnClickListener**.

Название вашего класса будет подчёркнуто волнистой красной чертой, щёлкните слово **public** и дождитесь появления красной лампочки, выберите вариант **Implement methods**. Появится диалоговое окно с выделенным методом **onClick**. Выбираем его и в коде появится заготовка для нажатия кнопки.

@Override

public void onClick(View v) {

}

Метод будет реализован не в отдельном объекте-обработчике, а в **Activity**, который и будет выступать обработчиком. В методе **onCreate()** присвоим обработчик кнопке. Это будет объект **this**, т.е. текущий объект нашей активности.

button.setOnClickListener(this);

На первых порах такой способ также покажется вам сложным и непонятным. Со временем и опытом понимание обязательно придёт.

Лично я рекомендую вам использовать первый способ, как самый простой и понятный. Использование второго и третьего способа дадут вам представление, как писать обработчики для других событий, так как кнопка может иметь и другие события. Например, кроме обычного нажатия существует долгое нажатие на кнопку (long click). Один из таких примеров с методом касания я привёл в конце этой статьи.

Продолжительное нажатие

Кроме обычного щелчка, в Android есть особый вид нажатия на кнопку - продолжительное нажатие. Это событие происходит, когда пользователь нажимает и удерживает кнопку в течение одной секунды. Этот тип нажатия обрабатывается независимо от обычного щелчка.

Для обработки продолжительного нажатия нужно реализовать класс **View.OnLongClickListener** и передать его в метод **setOnLongClickListener()**. Класс **OnLongClickListener** имеет один обязательный метод **OnLongClick()**. В принципе это похоже на метод **OnClick()**, только имеет возвращаемое значение.

imageButton.setOnLongClickListener(new OnLongClickListener() {

@Override

public boolean onLongClick(View v) {

Toast toast = Toast.makeText(getApplicationContext(),

"Long Click",

Toast.LENGTH\_SHORT);

toast.setGravity(Gravity.CENTER, 0, 0);

toast.show();

return false;

}

});

Запустите проект и убедитесь, что при быстром нажатии ничего не происходит, а при более продолжительном нажатии всплывает сообщение.

Плодитесь и размножайтесь

Когда у вас одна кнопка в окне, то у вас будет один метод, две кнопки - два метода и так далее. Если у вас несколько кнопок, то не обязательно для каждой прописывать свой метод, можно обойтись и одним, а уже в самом методе разделять код по идентификатору кнопки. Если вы посмотрите на код в предыдущих примерах, то увидите, что в методе присутствует параметр **View**, который и позволяет определить, для какой кнопки предназначен кусок кода:

public void onMyButtonClick(View view)

{

switch(view.getId()) {

case R.id.button1: // идентификатор "@+id/button1"

showAlertDialog();

break;

...

}

Предположим, у вас есть три кнопки:

import android.view.View.OnClickListener;

public class ButtonDemoActivity extends Activity implements OnClickListener...

final Button button1 = (Button)findViewById(R.id.button1);

final Button button2 = (Button)findViewById(R.id.button2);

final Button button3 = (Button)findViewById(R.id.button3);

// устанавливаем один обработчик для всех кнопок

button1.setOnClickListener(this);

button2.setOnClickListener(this);

button3.setOnClickListener(this);

// анализируем, какая кнопка была нажата. Всего один метод для всех кнопок

@Override

public void onClick(View v){

switch (v.getId()) {

case R.id.button1: editText.setText("Нажата кнопка Button1"); break;

case R.id.button2: editText.setText("Нажата кнопка Button2"); break;

case R.id.button3: editText.setText("Нажата кнопка Button3"); break;

}

}

Как видите, мы сократили количество кода. Теперь у нас один обработчик **onClick()**, в котором прописаны действия для трёх кнопок.

Сделать кнопку недоступной

Иногда нужно сделать кнопку недоступной и активировать её при определённых условиях. Через XML нельзя сделать кнопку недоступной (нет подходящего атрибута). Это можно сделать программно через метод **setEnabled()**:

button.setEnabled(false);

Как альтернативу можете рассмотреть атрибут **android:clickable**, который позволит кнопке не реагировать на касания, но при этом вид кнопки останется обычным.

Сделать кнопку плоской

Стандартная кнопка на экране выглядит выпуклой. Но в некоторых случаях желательно использовать плоский интерфейс. Раньше для этих целей можно было использовать **TextView** с обработкой щелчка. Но теперь рекомендуют использовать специальный стиль **borderlessButtonStyle**:

<Button

android:id="@+id/button1"

style="?android:attr/borderlessButtonStyle"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:onClick="onClick"

android:text="Button" />

Кнопка сохранит своё привычное поведение, будет менять свой цвет при нажатии и т.д.

С появлением Material Design добавились другие стили, например, **style="@style/Widget.AppCompat.Button.Borderless"**, который является предпочтительным вариантом. Попробуйте также **style="@style/Widget.AppCompat.Button.Borderless.Colored"**

Коснись меня нежно

Если вы внимательно понаблюдаете за поведением кнопки, то увидите, что код срабатывает в тот момент, когда вы отпускаете палец с кнопки. Для обычных приложений это вполне нормально, а для игр на скорость такой подход может оказаться слишком медленным. В подобных случаях лучше обрабатывать ситуацию не с нажатием кнопки, а с его касанием. В Android есть соответствующий слушатель **OnTouchListener()**:

Button button = (Button) findViewById(R.id.button);

button.setOnTouchListener(new OnTouchListener() {

@Override

public boolean onTouch(View v, MotionEvent event) {

if (event.getAction() == MotionEvent.ACTION\_DOWN) {

Toast.makeText(getApplicationContext(),

"Молодой человек, не прикасайтесь ко мне!",

Toast.LENGTH\_SHORT).show();

}

return false;

}

});

У метода **onTouch()** есть параметр **MotionEvent**, позволяющий более тонко определять касания экрана. Если произойдет событие, когда пользователь коснулся экрана, то ему будет соответствовать константа **ACTION\_DOWN**. Соответственно, если пользователь уберёт палец, то нужно использовать константу **ACTION\_UP**. Таким образом, можете расценивать щелчок кнопки как комбинацию двух событий - касания и отпускания.

Получить текст с кнопки

Задача - получить текст кнопки в методе **onClick()**. У метода есть параметр типа **View**, у которого нет метода **getText()**. Для этого нужно привести тип к типу **Button**.

public void onClick(View view) {

TextView infoTextView = (TextView) findViewById(R.id.textView);

Button button = (Button)view;

String buttonText = button.getText().toString();

infoTextView.setText(buttonText);

}

Если у вас несколько кнопок привязаны к методу **onClick()**, то щелчок покажет текст нажатой кнопки.