Тема 8. «Обработчики событий на примере Button»[1].

В этом уроке мы научимся обрабатывать нажатие кнопки и узнаем, что такое обработчик.

Создадим проект:
Project name: P0091_OnClickButtons
Build Target: Android 2.3.3
Application name: OnClickButtons
Package name: ru.startandroid.develop.onclickbuttons
Create Activity: MainActivity
В layout-файл main.xml напишем следующее и сохраним:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="30dp"
        android:orientation="vertical">
        <TextView
            android:id="@+id/tvOut"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginBottom="50dp"
            android:text="TextView">
        </TextView>
        <Button
            android:id="@+id/btnOk"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="OK">
        </Button>
        <Button
            android:id="@+id/btnCancel"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="Cancel">
        </Button>
    </LinearLayout>
</LinearLayout>
```

У нас есть TextView с текстом и две кнопки: OK и Cancel. Мы сделаем так, чтобы по нажатию кнопки менялось содержимое TextView. По нажатию кнопки OK – будем выводить текст: «Нажата кнопка OK», по нажатию Cancel – «Нажата кнопка Cancel».

Открываем MainActivity.java. Описание объектов вынесем за пределы метода onCreate. Это сделано для того, чтобы мы могли из любого метода обращаться к ним. В onCreate мы эти объекты заполним с помощью уже пройденного нами метода findViewById. В итоге должен получиться такой код:

```
public class MainActivity extends Activity {

    TextView tvOut;
    Button btnOk;
    Button btnCancel;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // найдем View-элементы
        tvOut = (TextView) findViewById(R.id.tvOut);
        btnOk = (Button) findViewById(R.id.btnOk);
        btnCancel = (Button) findViewById(R.id.btnCancel);

    }
}
```

Обновляем секцию import (CTRL+SHIFT+O). Объекты tvOut, btnOk и btnCancel соответствуют View-элементам экрана, и мы можем с ними работать. Нам надо научить кнопку реагировать на нажатие. Для этого у кнопки есть метод setOnClickListener (View.OnClickListener l). На вход подается объект с интерфейсом View.OnClickListener. Именно этому объекту кнопка поручит обрабатывать нажатия. Давайте создадим такой объект. Код продолжаем писать в onCreate:

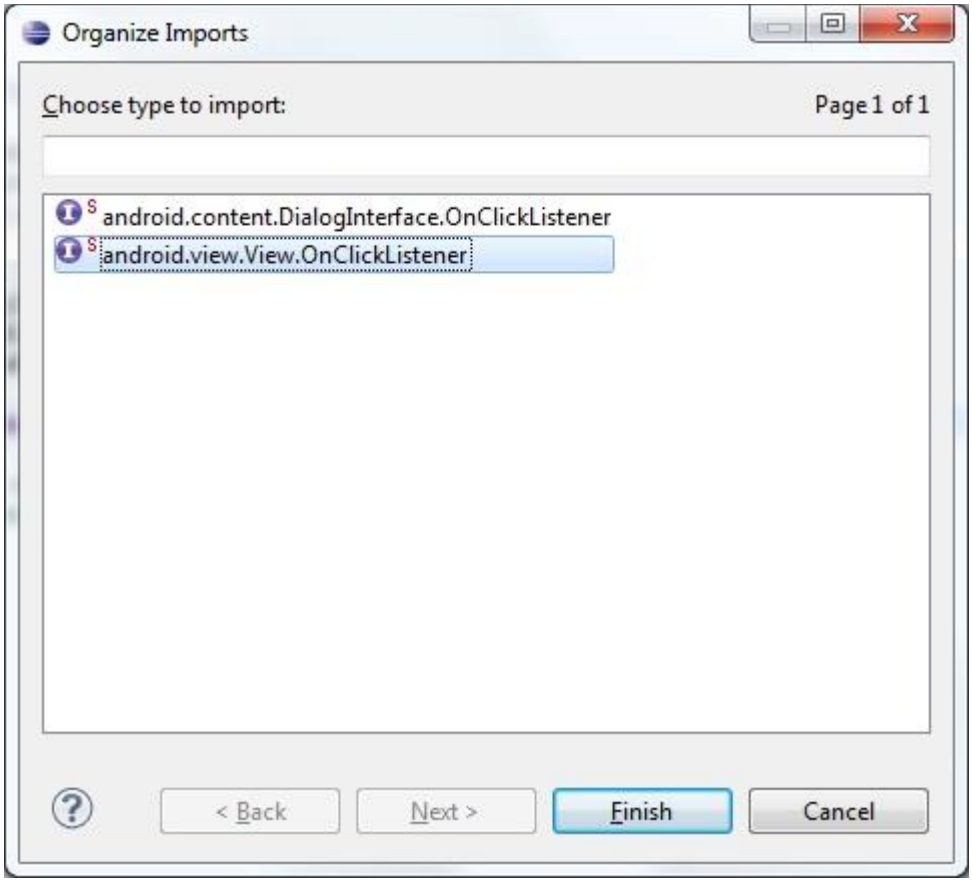
```
OnClickListener oclBtnOk = new OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub

    }
};
```

Eclipse подчеркивает OnClickListener красной линией

```
btnCancel = (Button) findViewById(R.id.btnCancel);

OnClickListener oclBtnOk = new OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub

    }
};
```

т.к. пока не знает его. Необходимо обновить секцию import.

Жмем CTRL+SHIFT+O, Eclipse показывает нам, что он знает два интерфейса с именем onClickListener и предлагает выбрать. Нам нужен View.OnClickListener, т.к. метод кнопки setOnClickListener принимает на вход именно его.



Итак, мы создали объект oclBtnOk, который реализует интерфейс View.OnClickListener. Объект содержит метод onClick – это как раз то, что нам нужно. Именно этот метод будет вызван при нажатии кнопки. Мы решили, что по нажатию будем выводить текст: «Нажата кнопка ОК» в TextView (tvOut). Реализуем это.

В методе onClick пишем:

```
1 | tvOut.setText("Нажата кнопка ОК");
```

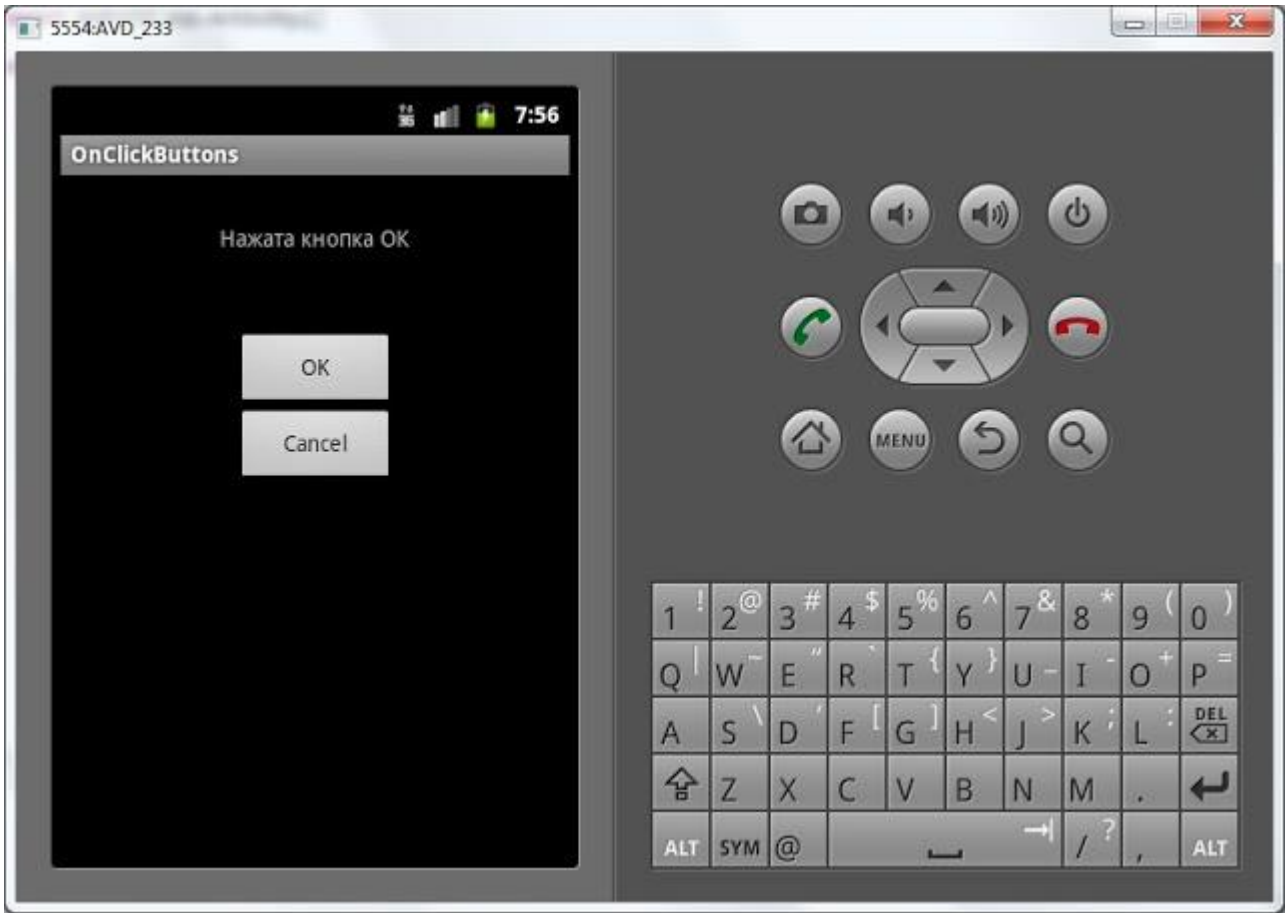
Обработчик нажатия готов. Осталось «скормить» его кнопке с помощью метода setOnClickListener.

```
1 | btnOk.setOnClickListener(oclBtnOk);
```

В итоге должен получится такой код:

```java
public class MainActivity extends Activity {

    TextView tvOut;
    Button btnOk;
    Button btnCancel;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // найдем View-элементы
        tvOut = (TextView) findViewById(R.id.tvOut);
        btnOk = (Button) findViewById(R.id.btnOk);
        btnCancel = (Button) findViewById(R.id.btnCancel);

        // создаем обработчик нажатия
        OnClickListener oclBtnOk = new OnClickListener() {
            @Override
            public void onClick(View v) {
                // Меняем текст в TextView (tvOut)
                tvOut.setText("Нажата кнопка OK");
            }
        };

        // присвоим обработчик кнопке OK (btnOk)
        btnOk.setOnClickListener(oclBtnOk);
    }
}
```

Все сохраняем и запускаем. Жмем на кнопку ОК и видим. Что текст изменился



Нажатие на Cancel пока ни к чему не приводит, т.к. для нее мы обработчик не создали и не присвоили. Давайте сделаем это аналогично, как для кнопки OK. Сначала мы создаем обработчик:

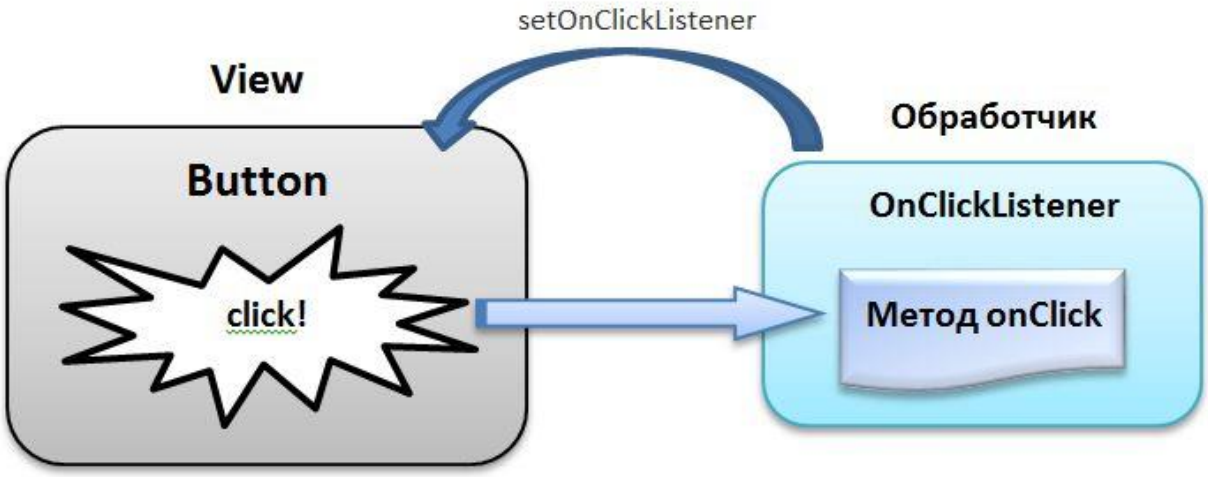
```
OnClickListener oclBtnCancel = new OnClickListener() {
    @Override
    public void onClick(View v) {
        // Меняем текст в TextView (tvOut)
        tvOut.setText("Нажата кнопка Cancel");
    }
};
```

Потом присваиваем его кнопке:

```
1 | btnCancel.setOnClickListener(oclBtnCancel);
```

Сохраняем, запускаем, проверяем. Обе кнопки теперь умеют обрабатывать нажатия.

Давайте еще раз проговорим механизм обработки событий на примере нажатия кнопки. Сама кнопка обрабатывать нажатия не умеет, ей нужен обработчик (его также называют слушателем – listener), который присваивается с помощью метода setOnClickListener. Когда на кнопку нажимают, обработчик реагирует и выполняет код из метода onClick. Это можно изобразить так:



Соответственно для реализации необходимо выполнить следующие шаги:

- создаем обработчик;
- заполняем метод onClick;
- присваиваем обработчик кнопке

и система обработки событий готова.

А теперь рассмотрим работу с обработкой щелчков.

Можете использовать старый проект или создать новый проект со стандартными настройками.

В режиме Design подведите курсор к элементу Button и перетащите его на форму. В результате ваших действий на форме появится стандартная кнопка с настройками по умолчанию.

В правой части экрана у вас имеется окно Properties, в котором вы можете настроить необходимые свойства для кнопки. Присваиваем свойству id новое значение buttonHello вместо стандартного button. Если временно переключиться в режим Text, то увидите, что на самом деле атрибут имеет значение @+id/buttonHello. Естественно, если вы редактируете свойства в текстовом виде, то вам тоже надо придерживаться этого стандарта. Свойству Text присвойте текст **Поздороваться.**

Если вы открыли предыдущий проект, то там уже был компонент TextView с текстом Hello, World.

Компонент TextView является текстовой меткой для вывода текста, который нельзя редактировать. В метке будем выводить приветствие после щелчка кнопки. В окне свойств удаляем текст из свойства Text, чтобы в текстовой метке ничего не было. Проследите, чтобы у него был идентификатор. Если в окне свойств ничего нет, то добавьте свой идентификатор, например, textView (@+id/textView).

Если вам не нравится взаимное расположение элементов, то можете на форме перетаскивать элементы, меняя их местами.

Будем считать, что интерфейс программы готов – у нас есть кнопка для нажатия и текстовая метка для вывода сообщений.

Теперь нужно научиться писать код для щелчка кнопки.

Переключитесь с режима Design на режим Text и найдите тег <Button>. Добавьте к нему еще одну строчку:

```
<Button
android:id="@+id/buttonHello"     android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignParentTop="true"     android:onClick="onClick"
android:text="Поздороваться" />
```

На самом деле это можно было сделать и через графическую модель. Переключитесь обратно в режим Design и найдите в свойствах кнопки пункт OnClick. Там будет прописано имя метода, которое мы задали через XML. Пользуйтесь любым удобным вам способом.

Таким образом мы определили событие onClick для кнопки (щелчок) и теперь осталось написать обработчик события. Вы уже знаете, как в Android Studio быстро создать заготовку. В текстовом режиме нажимаем комбинацию Alt+Enter и шаблон готов.

```
public void onClick(View view) {     }
```

Вставим код.

```
public void onClick(View view){TextView helloTextView = (TextView)findViewById(R.id.textView); helloTextView.setText("Hello Kitty!");}
```

Впрочем, это код нам уже знаком, но есть небольшое отличие. Текстовую метку мы объявляем и присваиваем ссылку на нужный компонент сразу в методе щелчка. Приложение у нас простое и доступ к текстовой метке больше нигде не осуществляется. Поэтому нет нужды объявлять переменную на уровне класса и инициализировать её в методе onCreate(). Если вы набирали текст самостоятельно, то у вас всё получится.

Запускаем проект и нажимаем на кнопку, чтобы увидеть результат. Можете изменить текст по своему желанию. Например, «Ты кто такой? Давай, до свидания!»

Полный текст кода будет выглядеть следующим образом.

```
package ru.schoolbreaker.hellokitty;
import android.app.Activity; import
android.os.Bundle; import
android.view.Menu; import
android.view.MenuItem; import
android.view.View; import
android.widget.TextView;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);        setContentView(R.layout.activity_main);
   }

   public void onClick(View view){
      TextView helloTextView = (TextView)findViewById(R.id.text
View);      helloTextView.setText("Hello Kitty!");
   }
}
```

Ещё раз обратите внимание на строчку

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

Метод findViewById() дословно переводится с английского как НайдиКомпонентПоИдентификатору и применяет тот же подход – вы указываете в параметре идентификатор ресурса и программа ищет подходящий компонент.

В дальнейшем вы постоянно будете использовать данный код в своих программах. Интерфейс будет выглядеть следующим образом, рис. 1.

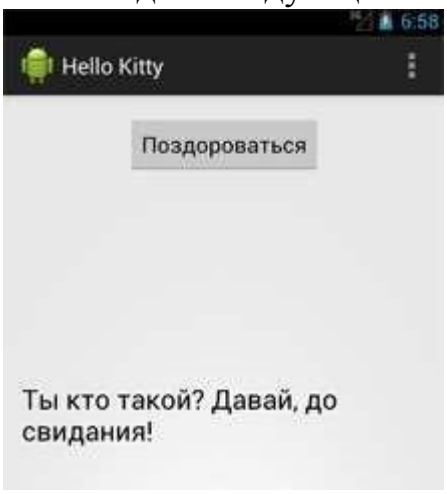


Рис. 1. Работа с кнопками

Новички на первых порах часто делают элементарную ошибку - помещают строчку инициализации объекта до метода setContentView().

Запомните, сначала нужно вывести шаблон (R.layout.activity_main), а только потом кнопки, текстовые поля, переключатели и т.д. В нашем случае используется отдельный метод для кнопки, который формируется после метода setContentView(). Позже, в других проектах вы увидите стандартные приёмы инициализации объектов.

На самом деле, способ обработки щелчка кнопки, который мы применили, является относительно новым для разработчиков. Google рекомендует использовать данный способ как удобный, требующий меньше кода и понятный для чтения. Но тем не менее вам придётся сталкиваться со старым способом. Поэтому необходимо изучить и второй способ, чтобы понимать другие примеры.

Добавьте в проект новую кнопку. Если первую кнопку мы добавляли перетаскиванием элемента Button с панели инструментов, то на этот раз мы попробуем создать кнопку вручную. Откройте файл activity_main.xml в режиме редактора кода и скопируйте код для первой кнопки. Потом с новой строки вставьте скопированный текст и подправьте несколько атрибутов, например, так.

```
<Button
   android:id="@+id/buttonCrowsCounter"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_below="@+id/buttonHello"     android:text="Считаем ворон"
/>
```

Обратите внимание, что поменялся текст и идентификатор для кнопки, а также некоторые параметры, отвечающие за положение кнопки на экране.

Переключитесь в графический режим и посмотрите, как выглядит кнопка. Если вас не устраивает положение кнопки, то перетащите её в другое место.

Снова открываем файл MainActivity.java и пишем следующий код сразу после объявления класса и до метода onCreate():

```
private Button mCrowsCounterButton;
```

Во время набора студия будет пытаться угадать, что вы хотите ввести. Активно используйте подсказки. Например, уже при первом вводе символа **B** студия предложит несколько вариантов на эту букву. Если нужное слово находится первым в списке, то нажимайте клавишу Enter, иначе выберите нужное слово из списка и потом уже нажимайте Enter. Это удобно при наборе длинных имён классов и переменных.

Google разработал целое руководство по наименованию переменных. Например, закрытая переменная на уровне класса должна начинаться с символа m (member), а далее идёт понятное название с заглавной буквы. Давайте попробуем придерживаться этого стиля.

После строчки setContentView(R.layout.activity_main); пишем:

```
mCrowsCounterButton = (Button)findViewById(R.id.buttonCrowsCounte r);
```

Переходим к самому важному - обработчику щелчка кнопки. Нам понадобится дополнительная переменная-счётчик mCount, которая будет содержать число подсчитанных ворон (её необходимо разместить выше метода onCreate() рядом с переменной mButtonCrowsCounter).

```
private int mCount = 0;
```

Теперь мы создаём обработчик, активно используя всплывающие подсказки для быстрого набора. Код добавляется в методе onCreate() после других строк, написанных ранее в этом методе.

Как использовать автодополнение.

Сначала вводим первые символы слова mCrowsCounterButton (можно маленькими буквами) и нажимаем Enter, если видим, что нужная подсказка появилась. После этого слова ставим точку и должны появиться опять подсказки, которые относятся к данной переменной. Начинаем вводим первые буквы слова setOnClickListener. Здесь тоже проблем обычно не возникает.

На данный момент у нас получилась строка

mCrowsCounterButton.setOnClickListener();

Ставим курсор внутри круглых скобок и набираем new OnClickListener. Здесь важно набрать символ O в верхнем регистре. Тогда у вас появится нужная подсказка типа OnClickListener{...} (android.view.View). Нажимаем Enter и получаем нужную заготовку, внутри которой вставляем код:

```
        mCrowsCounterButton.setOnClickListener(new  View.OnClickListener()
{
            @Override
            public  void onClick(View  v) {
               mInfoTextView.setText("Я  насчитал " + ++mCount  + " ворон"
);
            }
         });
```

Ранее мы объявили переменную helloTextView внутри метода onClick(), из которого она недоступна в других методах. Поэтому поступим с ней так же, как с кнопкой – объявим текстовую метку на уровне класса и инициализируем её в методе onCreate(). Сделайте это самостоятельно, создав переменную с именем mInfoTextView.

Запускаем приложение и начинаем щёлкать по кнопке. При каждом щелчке счётчик mCount будет увеличиваться на единицу, и эта информация будет отображаться на экране.

Нет смысла дублировать код для одной текстовой метки. Поэтому для первой кнопки, которая здоровалась, код можно сократить.

```
public  void onClick(View  view)  {
   mInfoTextView.setText("Hello  Kitty!");  }
```

Мы познакомились с новым способом обработки щелчка кнопки.

Какой способ вы предпочтёте – зависит от вас. Интерфейс будет выглядеть следующим образом, рис. 2.

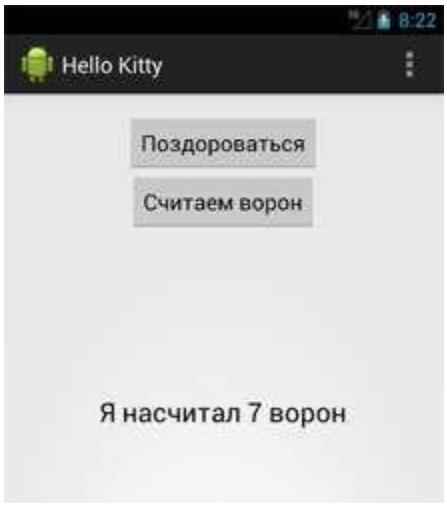


Рис. 2. Работа с кнопками

Теперь у вас есть чрезвычайно полезное приложение Счётчик ворон. Если преподаватель вас неожиданно спросит на занятии, почему вы смотрите в окно с рассеянным видом, вы можете смело достать свой телефон и сказать, что заняты очень важным делом – считаете ворон.

Исправляем ошибки.

На первых порах новички делают элементарные ошибки - опечатки, копирование куска кода без его понимания, пишут не в том месте и т.д. Студия пытается по мере возможностей подсказать вам, но не все обращают внимание на предупреждения. Небольшой ликбез по нахождению ошибок. В редакторе кода в верхнем правом углу есть прямоугольник. Он может быть зелёным (идеальный код), жёлтым (не смертельно, но лучше исправить) и красным (ошибка в коде, программа не запустится).

Ваша задача – стремиться к зелёному цвету. Жёлтый цвет желательно просматривать и решать самостоятельно, нужно ли исправлять код. Если вы понимаете в чём проблема, то исправьте. Если не понятно, то оставьте. Предупреждения не всегда бывают по делу, иногда их можно игнорировать.

Понимание придёт с опытом и практикой.

Ниже показан случай, рис. 3, когда мы сделали опечатку в названии класса String.

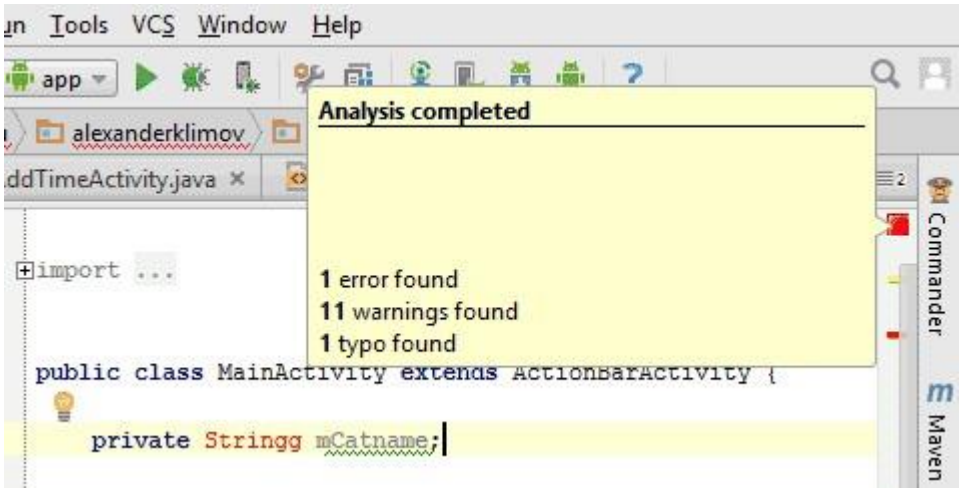


Рис. 3. Исправление ошибок

Помимо прямоугольника, там же ниже есть зарубки с такими же цветами. Подведите курсор мыши к любой из зарубок и увидите подсказку о характере ошибки или предупреждения. Щелчок по зарубке перенесёт вас в нужное место в документе.