

Приложения, которые что-то делают

Android Studio

CatOrDog

На этом уроке мы создадим приложение, взаимодействующее с пользователем.

В приложении **CatOrDog** пользователь выбирает, кто ему нравится больше: кошки или собаки.



Структура приложения



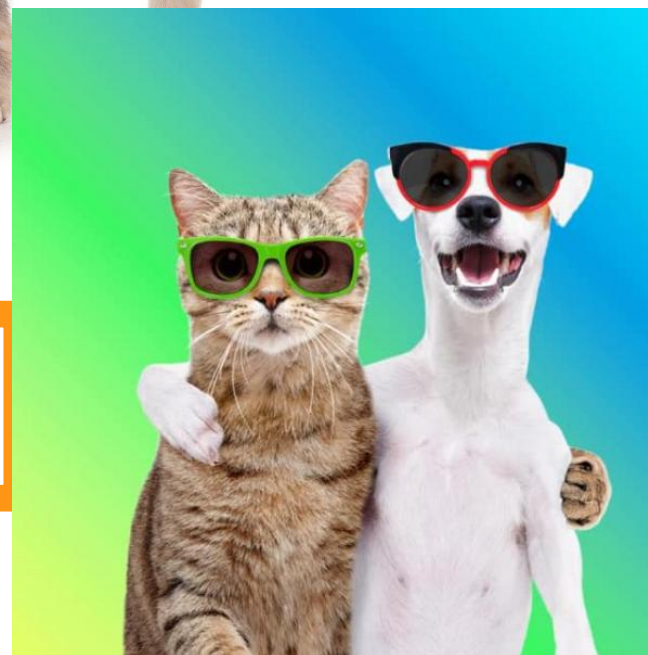
Макет определяет, как будет выглядеть приложение.

Он состоит из трех компонентов графического интерфейса:

- раскрывающегося списка значений, в котором пользователь выбирает кто нравится больше: кошки или собаки;
- кнопки, которая при нажатии возвращает подборку;
- надписи для пород.

Структура приложения

- Файл `strings.xml` включает все строковые ресурсы, необходимые макету, — например, текст надписи на кнопке.
- **Активность** определяет, как приложение должно взаимодействовать с пользователем. Она получает выбор пользователя, и использует его для вывода списка сортов, которые могут представлять интерес для пользователя. Для решения этой задачи используется вспомогательный класс Java.

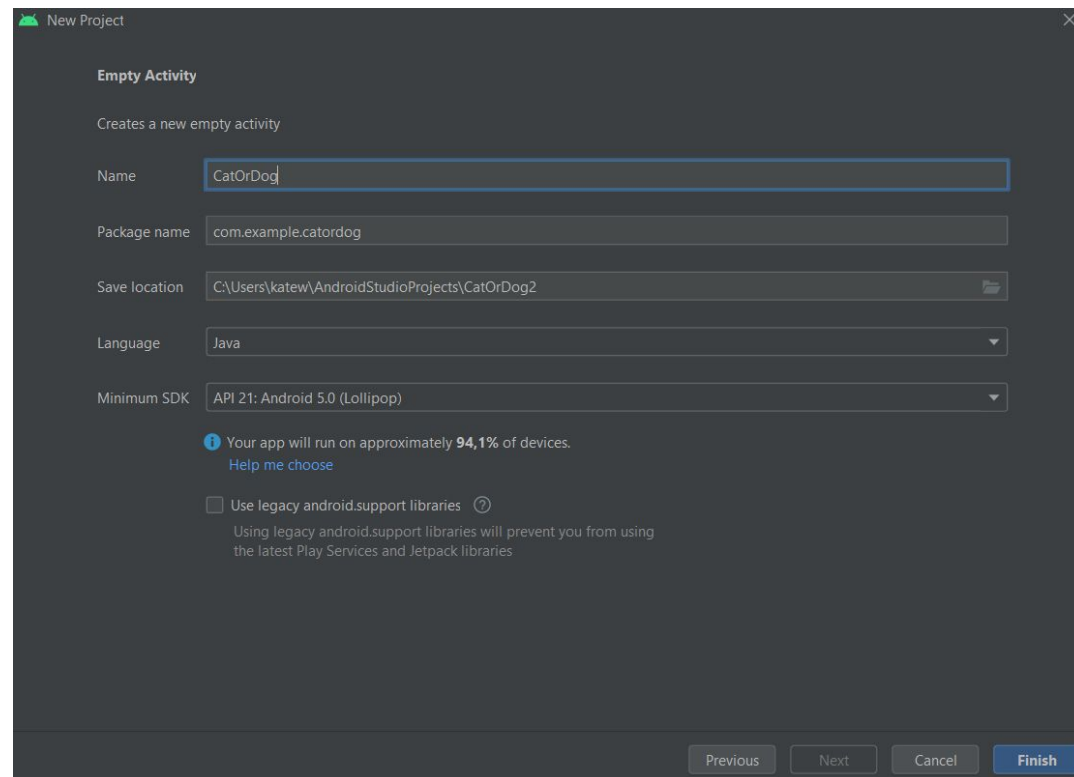


Структура приложения

- **Класс Java**, содержащий логику приложения.
- **Класс** включает метод, который получает выбор пользователя в параметре и возвращает список пород указанного типа.
- **Активность** вызывает метод, передает выбор пользователя и использует полученный ответ.



Создадим новое приложение

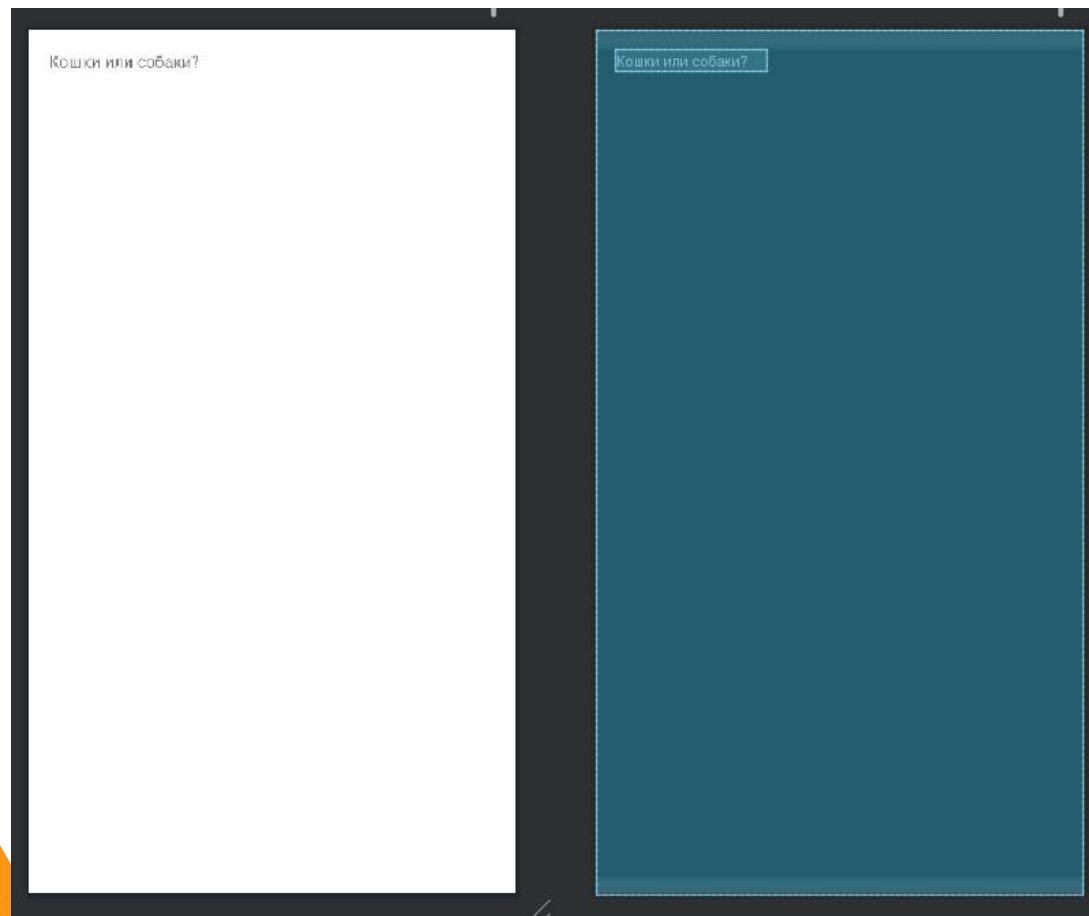


Начнем с редактирования файла макета activity_main.xml

Переключитесь на текстовую версию разметки, чтобы открыть редактор кода, и замените код из activity_main.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Вечный вопрос: Кошки или собаки?" />
</LinearLayout>
```

<LinearLayout>



Мы изменили код, сгенерированный Android Studio, чтобы в нем использовался элемент **<LinearLayout>**.

Он используется для вывода компонентов графического интерфейса рядом друг с другом, по вертикали или по горизонтали.

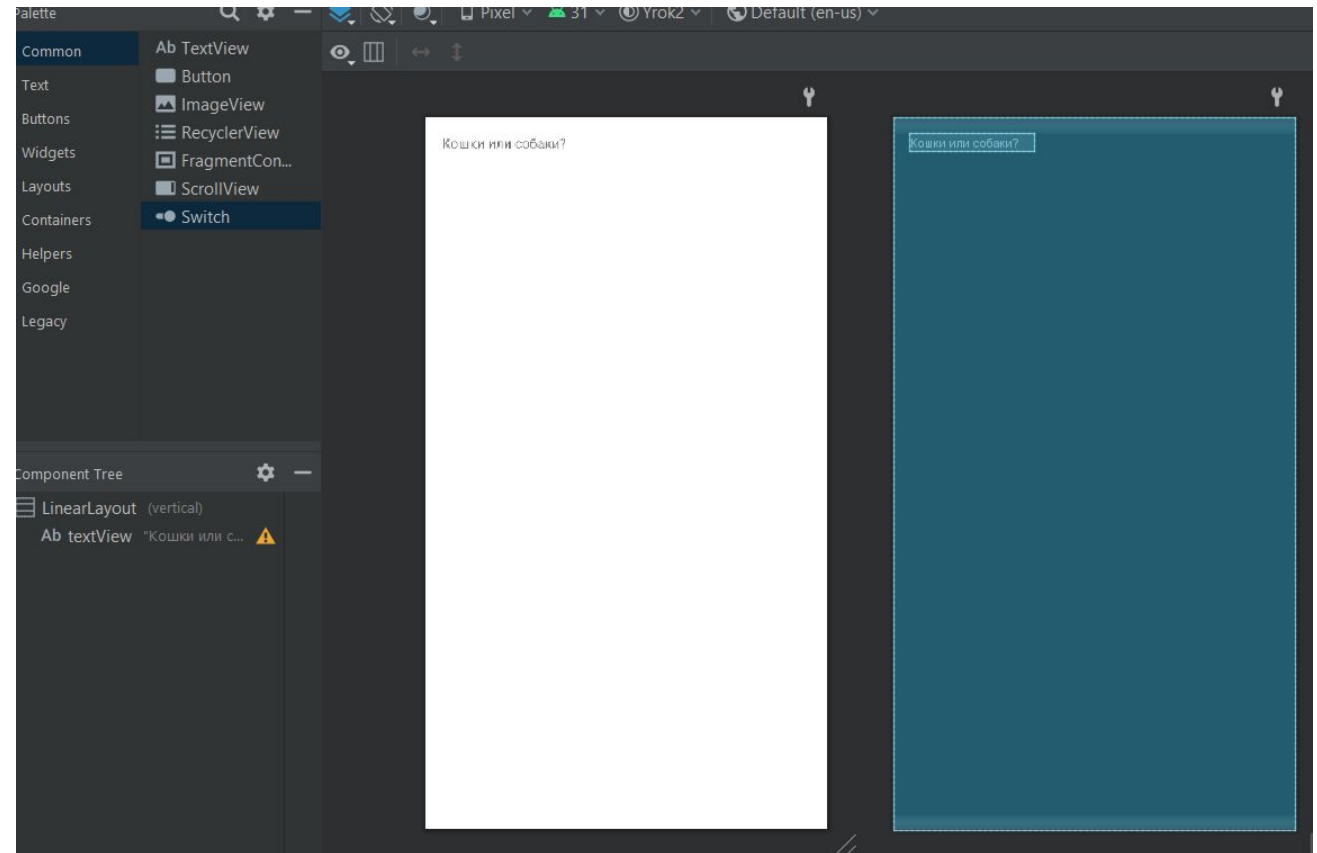
Если компоненты выстраиваются по вертикали, они образуют столбец, а если по горизонтали — строку.

Любые изменения, вносимые в XML-разметку макета, отражаются в визуальном редакторе Android Studio; чтобы убедиться в этом, достаточно щелкнуть на вкладке Design.

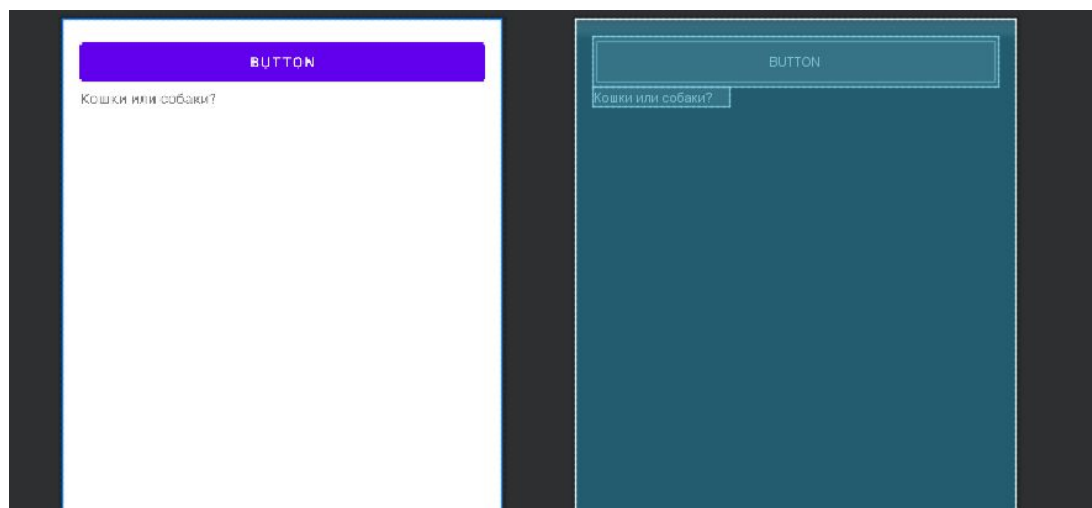
Знакомство с визуальным редактором

Визуальный редактор предоставляет более наглядные средства для редактирования кода макета, нежели при редактировании разметки XML. В нем предусмотрены два разных представления макета. Одно показывает, как макет будет выглядеть на устройствах, а в другом представлении выводится эскиз структуры макета.

Слева от визуального редактора располагается палитра с компонентами графического интерфейса, которые можно перетаскивать мышью на макет.



Добавление кнопки в визуальном редакторе



Добавим кнопку в макет в визуальном редакторе. Найдите в палитре компонент **Button**, щелкните на нем и перетащите в визуальный редактор, чтобы кнопка располагалась над надписью.

Кнопка появляется в макете.

Изменения, внесенные в визуальном редакторе, отражаются **XML**.

<Button>

В мире Android кнопка нажимается пользователем, чтобы инициировать какое-либо действие.

Элемент <Button> обладает свойствами, управляющими размером и внешним видом кнопки. Эти свойства существуют не только у кнопок — они есть и у других компонентов графического интерфейса.

Тот факт, что кнопки и надписи имеют так много общих свойств, вполне логичен — оба компонента наследуют от одного класса **Android View**.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Кошки или собаки?" />

</LinearLayout>
```

Внесем изменения в код XML

```
<Spinner  
    android:id="@+id/color"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="40dp"  
    android:layout_gravity="center"  
    android:layout_margin="16dp" />
```

Раскрывающийся список значений в системе Android. Компонент предназначен для выбора одного значения из представленного набора.

```
<Button  
    android:id="@+id/find_CatOrDog"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:layout_margin="16dp"  
    android:text="Показать породы" />
```

Свойству id кнопки задается значение «find_CatOrDog». Оно будет использовано

позд
Ширина кнопки изменяется по ширине
содержимого.

Кнопка выравнивается по центру
макета по горизонтали, и ей

```
<TextView  
    android:id="@+id/porody"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:layout_margin="16dp"  
    android:text="Кошки или собаки?" />
```

Свойству id надписи задается значение «porody»

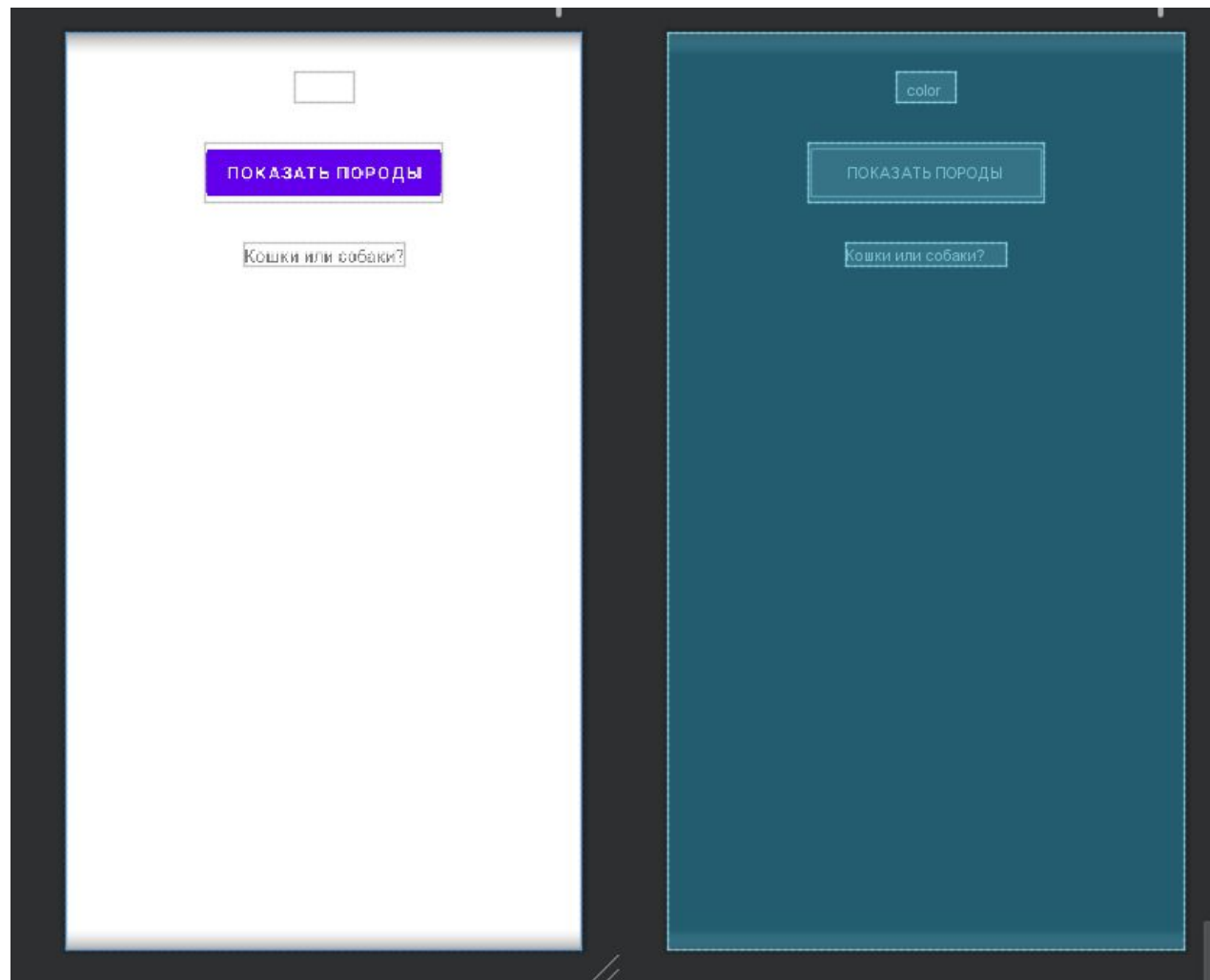
Надпись выравнивается по центру
макета по горизонтали, и ей

наз

Визуальный редактор

После внесения изменений в XML макета перейдите в визуальный редактор. Вместо макета с кнопкой и расположенной под ней надписью должен отображаться макет с раскрывающимся списком, кнопкой и надписью, выровненными по центру в один столбец.

Над кнопкой располагается раскрывающийся список **spinner**. Если коснуться его, на экране появляется набор значений, из которого пользователь выбирает одно значение.



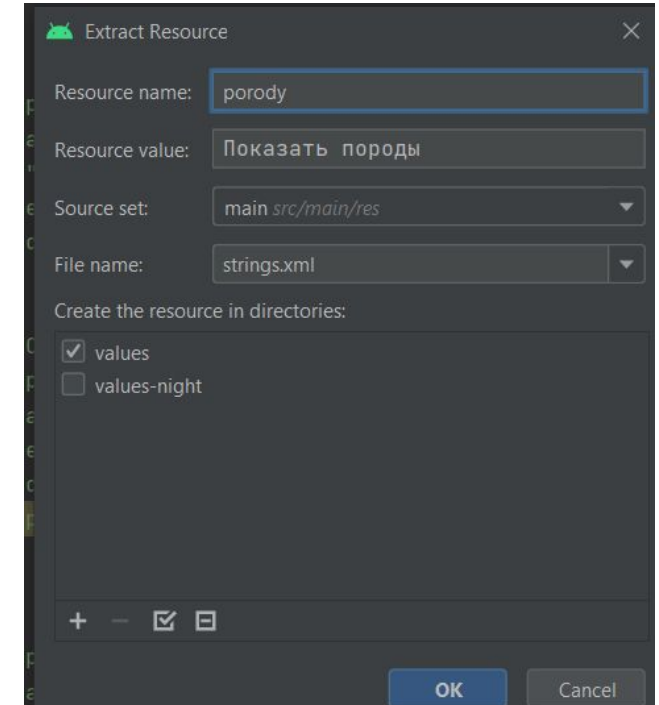
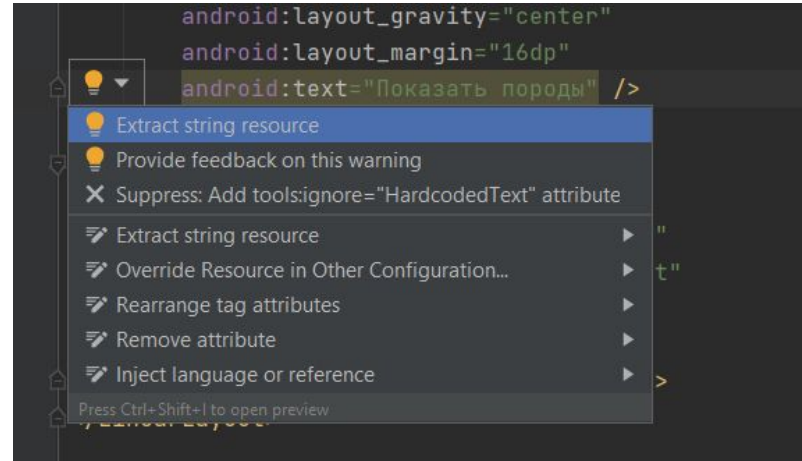
Поместите текст в файл `strings.xml` (используйте лампочку)

Лучше разместить текстовые значения в файле строковых ресурсов с именем `strings.xml`.

Выделение текстовых значений в `strings.xml` существенно упрощает интернационализацию приложений.

Вместо того чтобы изменять жестко запрограммированные текстовые значения в множестве разных файлов, достаточно заменить файл `strings.xml` его локализованной версией.

Такой подход также упрощает глобальные изменения в тексте в масштабах всего приложения, так как для этого достаточно отредактировать всего один файл. Если вам понадобится изменить текст в приложении, для этого следует изменить файл `strings.xml`.



```
<resources>
  <string name="app_name">CatOrDog</string>
  <string name="porody">Показать породы</string>
  <string name="vopros">Кошки или собаки?</string>
</resources>
```

Файл `strings.xml`

Добавление значений в список

На данный момент макет включает раскрывающийся список, но в этом списке нет никаких данных. Чтобы список приносил пользу, в нем должен отображаться список значений. Пользователь выбирает в этом списке то значение, которое ему нужно.

Список значений для раскрывающегося списка определяется практически так же, как мы определим текст на кнопке и надписи: нужно создать для него ресурс. До сих пор в файле **strings.xml** определялись одиночные строковые значения. Все, что нам нужно, — это определить массив строковых значений и передать ссылку на него раскрывающемуся списку.

```
<resources>
  <string name="app_name">CatOrDog</string>
  <string name="porody">Показать породы</string>
  <string name="vopros">Кошки или собаки?</string>
  <string-array name="dog_porody">
    <item>Дворовые</item>
    <item>Корги</item>
    <item>Лайки</item>
    <item>Таксы</item>
  </string-array>
</resources>
```

В файл strings.xml добавляется элемент stringarray. Он определяет массив строк с именем beer_colors, состоящий из четырех значений

Для обращения к массиву строк в макете используется синтаксис, сходный с синтаксисом получения строкового значения:

```
<Spinner
  android:id="@+id/color"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_marginTop="40dp"
  android:layout_gravity="center"
  android:layout_margin="16dp"
  android:entries="@array/dog_porody" />
```

Меняем список

1. Поменять значение в выпадающем списке

```
<resources>
  <string name="app_name">CatOrDog</string>
  <string name="porody">Показать породы</string>
  <string name="vopros">Кошки или собаки?</string>
  <string-array name="CatOrDog">
    <item>Кошки</item>
    <item>Собаки</item>
  </string-array>
</resources>
```

2. Сменим имя массива.

```
<Spinner
  android:id="@+id/color"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="center"
  android:layout_margin="16dp"
  android:layout_marginTop="40dp"
  android:entries="@array/CatOrDog"
  android:minHeight="48dp" />
```


Как заставить кнопку вызвать метод

Чтобы щелчок на кнопке приводил к вызову метода активности,

необходимо внести изменения в двух файлах:

1. Изменения в файле макета activity_main.xml.

Необходимо указать, какой метод активности должен вызываться при щелчке на кнопке.

2. Изменения в файле активности MainActivity.java.

Необходимо написать метод, который будет вызываться при щелчке.

onClick и метод, вызываемый при щелчке

Чтобы сообщить Android, какой метод должен вызываться при щелчке на кнопке, достаточно всего одной строки разметки XML. Все, что для того нужно — добавить атрибут **android:onClick** в элемент **<button>** и указать имя вызываемого метода.

```
<Button
  android:id="@+id/find_CatOrDog"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="center"
  android:layout_margin="16dp"
  android:text="@string/porody"
  android:onClick="onClickCatOrDog" />
```

Добавим атрибут android:onClick в элемент <button>

Воспользуемся подсказкой

- Provide feedback on this warning
- Provide feedback on this warning
- Suppress: Add tools:ignore="OnClick" attribute
- Create 'onClickCatOrDog(View)' in 'MainActivity'
- Suppress: Add tools:ignore="UsingOnClickInXml" attribute
- Create onClick event handler
- Override Resource in Other Configuration...
- Rearrange tag attributes
- Remove attribute
- Inject language or reference

MainActivity.java

1. Дописываем строки в import, нам это пригодится
2. Создаем новый class «CatOrDogExpert»
3. Воспользуемся лампочкой

```
package com.example.yrok2;

import android.os.Bundle;
import android.view.View;
import android.widget.Spinner;
import android.widget.TextView;
import java.util.List;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    public CatOrDogExpert porody_vybor = new CatOrDogExpert ();

    private savedInstanceState) {
        savedInstanceState);
        activity_main);

    /view view) {
        TextView porody = (TextView) findViewById(R.id.porody);
```

Дописываем

Создаем класс

- Create class 'CatOrDogExper'
- Create enum 'CatOrDogExper'
- Create inner class 'CatOrDogExper'
- Create interface 'CatOrDogExper'
- Create type parameter 'CatOrDogExper'
- Split into declaration and initialization
- Change access modifier
- Move initializer to constructor

Press Ctrl+Shift+I to open preview

onCreate().

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

Этот код — все, что необходимо для создания простейшей активности. Как видите, в нем создается класс, который расширяет класс `android.app.Activity` и реализует метод `onCreate()`.

Все активности (не только эта) должны расширять класс `Activity` или один из его subclasses. Класс `Activity` содержит набор методов, которые превращают обычный класс Java в полноценную активность `Android`. Все активности также должны реализовать метод `onCreate()`.

Метод `onCreate()` вызывается при создании объекта активности и используется для настройки основных параметров — например, выбора макета, с которым связывается активность. Это делается при помощи метода `setContentView()`. В приведенном примере вызов `setContentView(R.layout.activity_main)` сообщает `Android`, что эта активность использует макет `activity_main`.

Ранее мы добавили атрибут `onClick` к кнопке в макете и присвоили ему значение `ClickCatOrDog`. Теперь нужно добавить этот метод в активность, чтобы он вызывался при нажатии кнопки. Таким образом, активность будет реагировать на нажатия пользователем кнопки в интерфейсе.

Построение вспомогательного класса Java

Переписываем код со слайда в
вспомогательный класс
`CatOrDogExpert.java`

`ArrayList` - автоматически расширяемый
массив.

Работать с `ArrayList` просто: вставить
строку можно методом `add()`.

В Java сравнение объектов
производится с помощью метода
`equals()` класса `Object`. Этот метод
сравнивает содержимое объектов и
выводит значение типа `boolean`.
Значение `true` - если содержимое
эквивалентно, и `false` — если нет.

```
import java.util.ArrayList;
import java.util.List;

public class CatOrDogExpert {
    List<String> getPorody(String color) {
        List<String> porody = new ArrayList<>();
        if (color.equals("Кошки")) {
            porody.add("Британские");
            porody.add("Дворовые");
        } else {
            porody.add("Корги");
            porody.add("Таксы");
        }
        return porody;
    }
}
```

MainActivity.java

Переписываем код со слайда в MainActivity.java

```
<Button
    android:id="@+id/find_CatOrDog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_margin="16dp"
    android:text="@string/porody"
    android:onClick="onClickCatOrDog" />
```

```
public void onClickCatOrDog(View view) {
    TextView porody = (TextView) findViewById(R.id.porody); //Получить ссылку на TextView
    Spinner color = (Spinner) findViewById(R.id.color); //Получить ссылку на Spinner
    String porodaType = String.valueOf(color.getSelectedItem()); //Получить вариант, выбранный в Spinner

    //Получить рекомендации от класса CatOrDogExpert
    List<String> porodyList = porody_vybor.getPorody(porodaType); // Получить контейнер List с породами
    StringBuilder porodyFormatted = new StringBuilder(); //Построить String по данным из List
    for (String new_poroda : porodyList) {
        porodyFormatted.append(new_poroda).append('\n'); //Каждая порода выводится с новой строки
    }
    //Вывести породы
    porody.setText(porodyFormatted); // Вывести результаты в надписи
}
```

findViewById()

Необходимо сначала получить ссылки на оба компонента графического интерфейса в макете — раскрывающийся список и надпись. С помощью этих ссылок мы сможем получить значение, выбранное в списке и вывести текст в надписи.

Метод **findViewById()** получает идентификатор компонента в виде параметра и возвращает объект **View**. Далее остается привести возвращаемое значение к правильному типу компонента (например, **TextView** или **Button**).

```
public void onClickCatOrDog(View view) {
    TextView porody = (TextView) findViewById(R.id.porody); //Получ
    Spinner color = (Spinner) findViewById(R.id.color); //Получить
    String porodaType = String.valueOf(color.getSelectedItem()); //

    //Получить рекомендации от класса CatOrDogExpert
    List<String> porodyList = porody_vybor.getPorody(porodaType); /
    StringBuilder porodyFormatted = new StringBuilder(); //Построит
    for (String new_poroda : porodyList) {
        porodyFormatted.append(new_poroda).append('\n'); //Каждая п
    }
    //Вывести породы
    porody.setText(porodyFormatted); // Вывести результаты в надпис
}
```

Класс `StringBuilder`, МЕТОДЫ `append()` и `setText`

```
//Получить рекомендации от класса CatOrDogExpert
List<String> porodyList = porody_vybor.getPorody(porodaType);
StringBuilder porodyFormatted = new StringBuilder(); //Построит
for (String new_poroda : porodyList) {
    porodyFormatted.append(new_poroda).append('\n'); //Каждая
}
//Вывести породы
porody.setText(porodyFormatted); // Вывести результаты в надписи
}
```

Класс `StringBuilder` представляет расширяемые и доступные для изменений последовательности символов, позволяя вставлять символы и подстроки в существующую строку и в любом месте. Данный класс гораздо экономичнее в плане потребления памяти и настоятельно рекомендуется к использованию.

Метод `append()` — обновляет значение объекта, который вызвал метод.

В Java `append()` возвращает обновленные объекты `StringBuilder`

Метод `setText` — содержимое `String` отображается в надписи

Сохраняем и запускаем

