

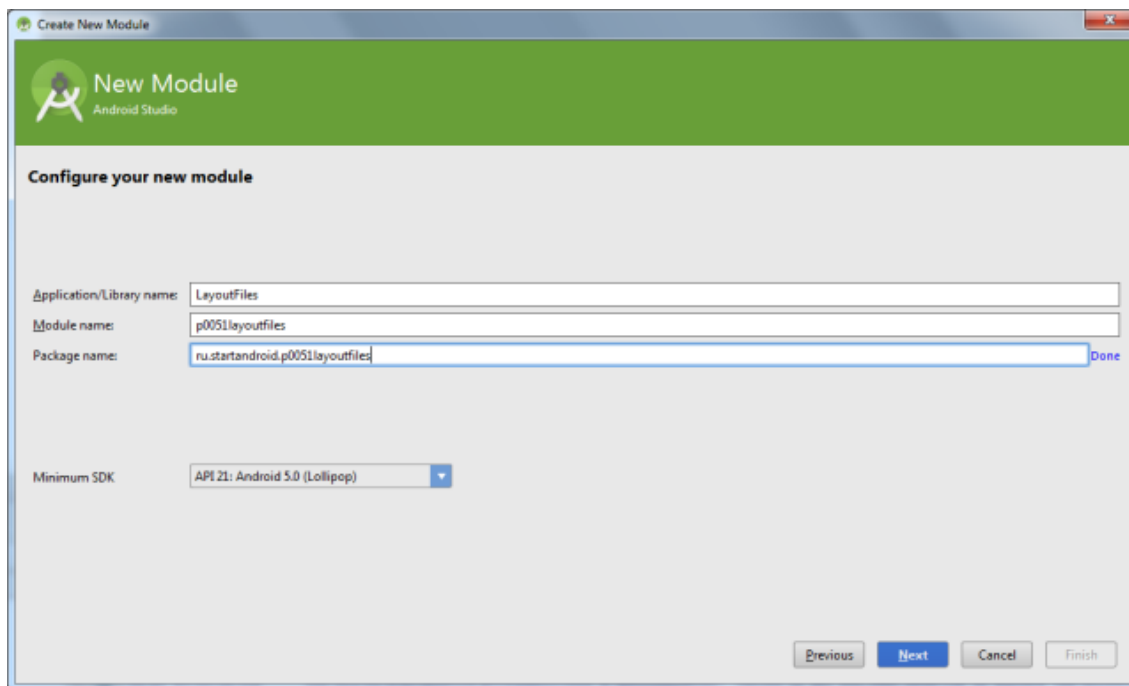
XML представление. Смена ориентации экрана.

Создадим новое приложение - модуль в проекте Android lessons со следующими параметрами

Application/Library name: LayoutFiles

Module name: p0051layoutfiles

Package name: ru.startandroid.p0051layoutfiles



Create New Module

New Module
Android Studio

Configure your new module

Application/Library name: LayoutFiles

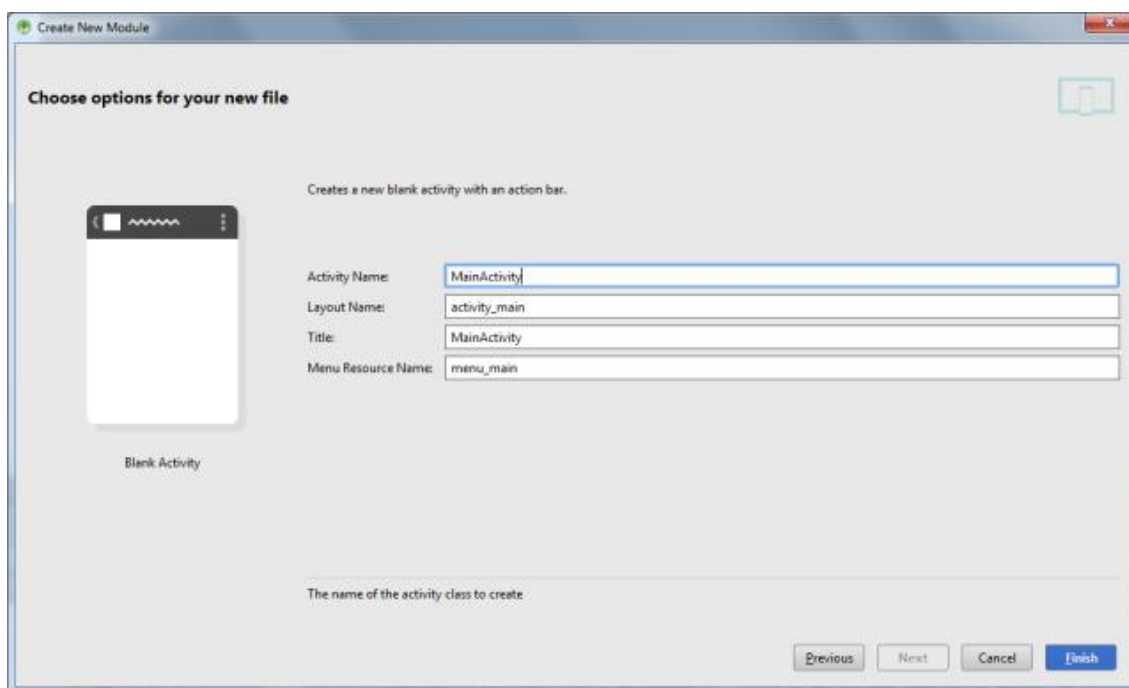
Module name: p0051layoutfiles

Package name: ru.startandroid.p0051layoutfiles Done

Minimum SDK: API 21: Android 5.0 (Lollipop)

Previous Next Cancel Finish

При создании модуля укажем, что надо создать Activity, для этого в поле **Activity Name** запишем MainActivity:



Create New Module

Choose options for your new file

Creates a new blank activity with an action bar.

Blank Activity

Activity Name: MainActivity

Layout Name: activity_main

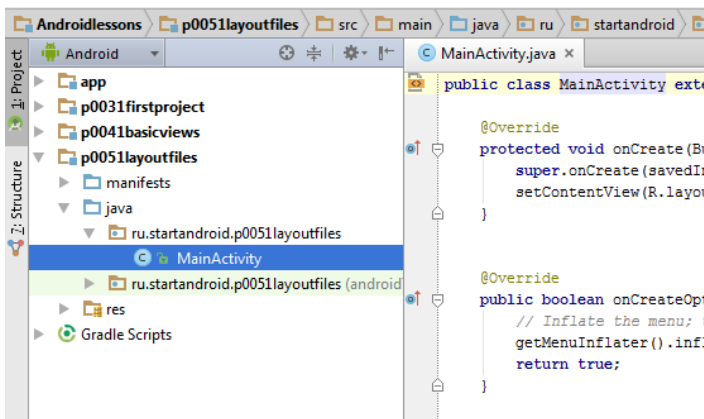
Title: MainActivity

Menu Resource Name: menu_main

The name of the activity class to create

Previous Next Cancel Finish

Тогда среда разработки создаст соответствующий класс. Чтобы посмотреть созданный класс, откроем двойным кликом файл: MainActivity.java



Смотрим java-код. Нас интересует метод **onCreate** — он вызывается, когда приложение создает и отображает Activity (на остальные методы пока не обращаем внимания). Посмотрим код реализации onCreate:

Первая строка:

```
super.onCreate(savedInstanceState);
```

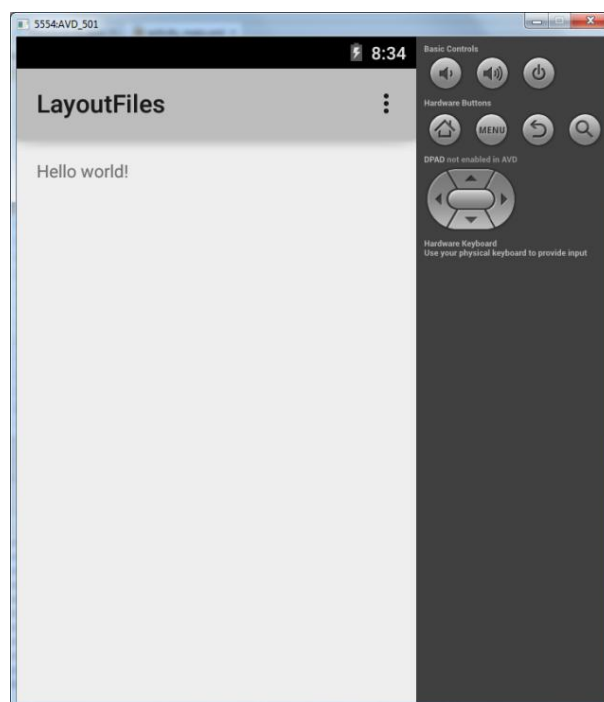
- это вызов метода родительского класса, выполняющего необходимые процедуры, его не трогаем.

В следующей строке используется метод [setContentView\(int\)](#), который устанавливает содержимое Activity из layout-файла. Но в качестве аргумента указывается не путь к layout-файлу (res/layout/activity_main.xml), а константа, которая является ID файла. Эта константа генерируется автоматически в файле R.java.

Файл res/layout/activity_main.xml был создан средой разработки вместе с Activity. Его название запрашивалось на том же экране, где и название Activity (рис. выше).

Откроем двойным кликом res/layout/activity_main.xml, посмотрим, что там...

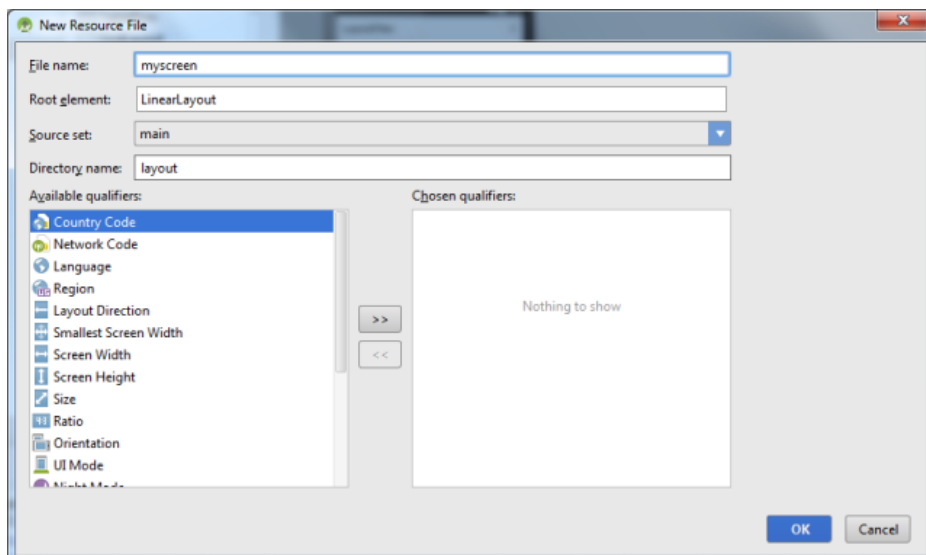
Запустим приложение и посмотрим, что оно нам покажет:



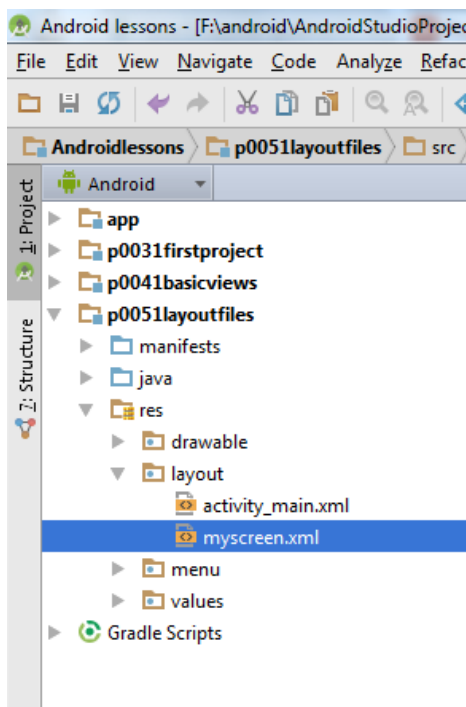
Все верно - Activity отобразил то, что прописано в activity_main.xml.

Попробуем отобразить содержимое другого файла. Создадим еще один layout-файл, например myscreen.xml. Для этого выделим папку res/layout в нашем модуле и нажмем на ней правую кнопку мыши. В появившемся меню выбираем New > Layout resource file. Для любителей горячих клавиш есть более удобный путь: при выделенной папке res/layout нажать ALT+Insert, и там уже Enter на пункте Layout resource file.

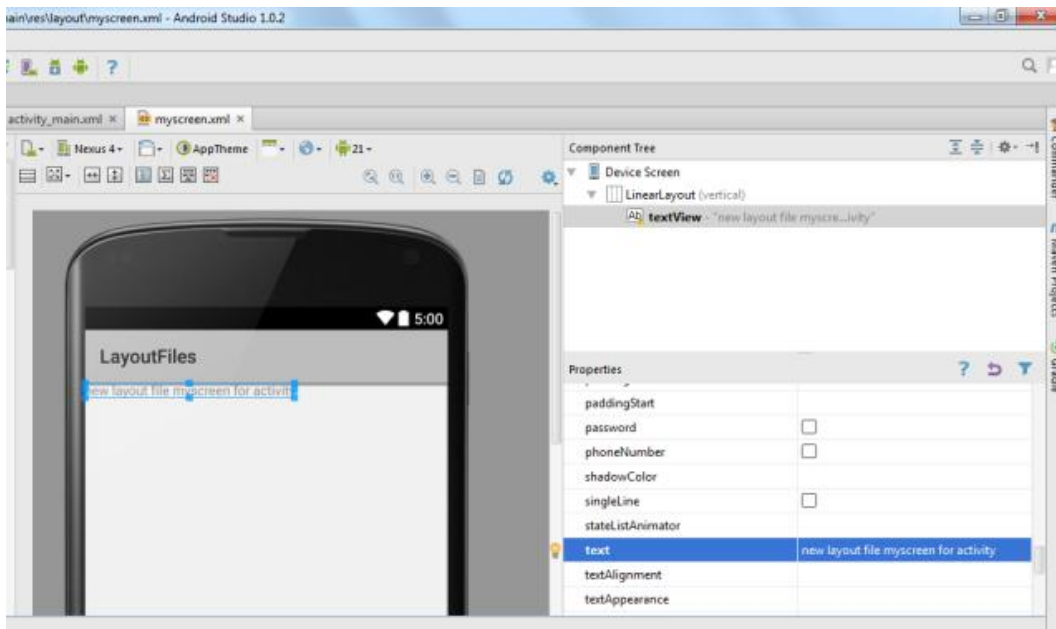
Откроется окно:



Вводим имя файла myscreen, остальное пока не меняем, жмем ОК. В папке layout должен появиться новый файл myscreen.xml:



Этот новый layout-файл должен сразу открыться на редактирование. Добавим на экран элемент Plain TextView из списка слева и через Properties изменим его текст на: «new layout file myscreen for activity».

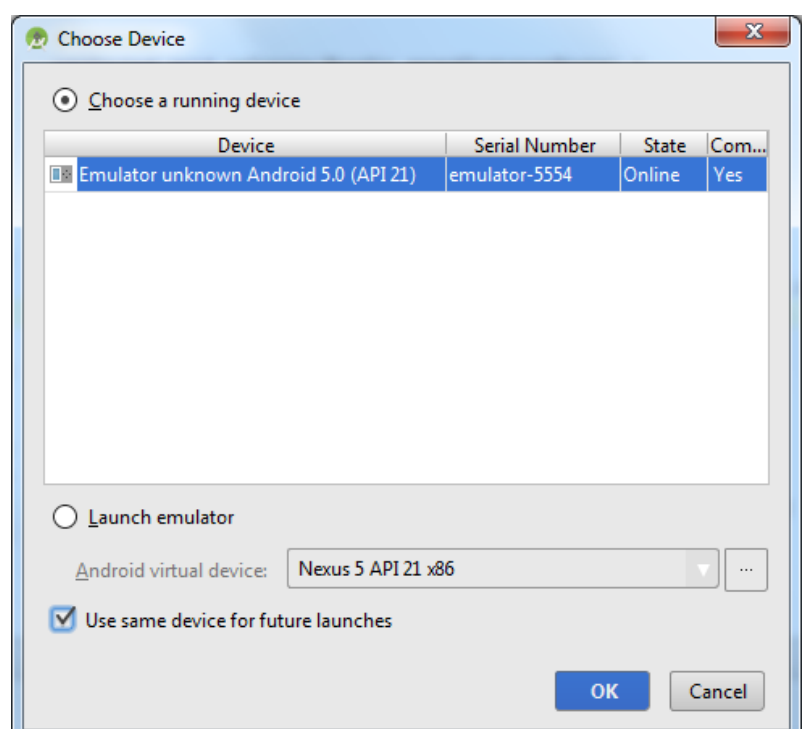


Обязательно сохраняем (CTRL+S). При создании нового layout-файла myscreen, среда добавила в R.java новую константу для этого файла - R.layout.myscreen. Настроим так, чтобы Activity использовало новый файл myscreen.xml, а не activity_main.xml, который был изначально. Откроем MainActivity.java и поменяем аргумент метода setContentView. Замените «R.layout.activity_main», на «R.layout.myscreen» (ID нового layout-файла). Должно получиться так:

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

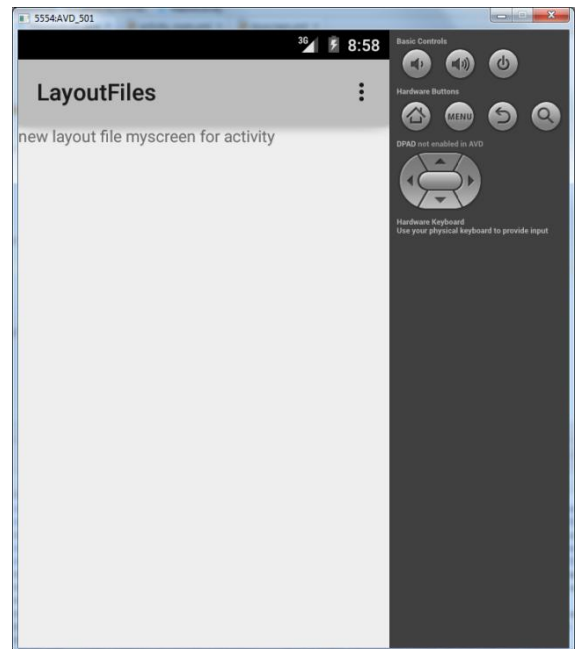
Сохраняем код (CTRL+S) и запускаем приложение (SHIFT+F10). Теперь нам предложат подтвердить, что мы хотим запустить приложение на включенном эмуляторе.

Чтобы он при каждом запуске это не спрашивал, включите галку Use same device for future launches и нажмите OK.



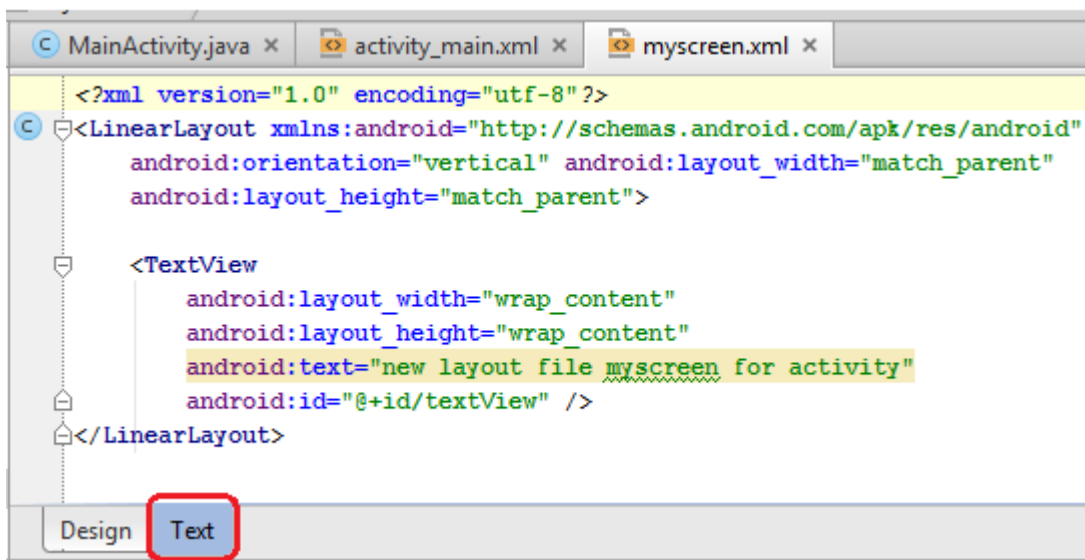
Приложение запустилось:

Видим, что теперь оно отображает содержимое из myscreen.xml, т.к. мы явно ему это указали в методе setContentView, который выполняется при создании (onCreate) Activity



Layout-файл в виде XML

Открыв в Android Studio layout файл activity_main или myscreen, вы видите его визуальное представление. Т.е. некий предпросмотр, как это будет выглядеть на экране. Снизу вы можете видеть две вкладки – Design и Text. Откройте вкладку Text:



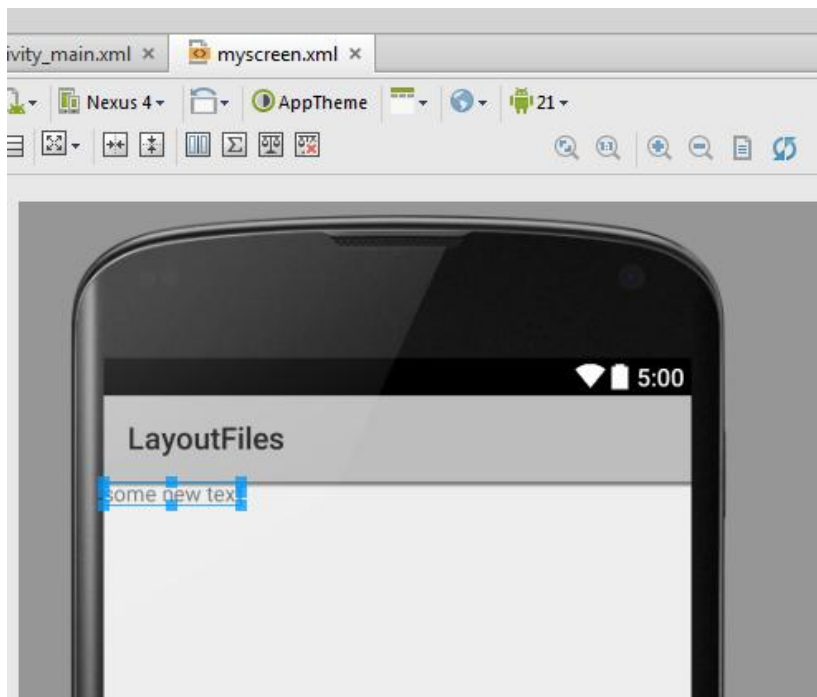
В ней представлено xml-описание всех View layout-файла. Названия xml-элементов - это классы View-элементов, xml-атрибуты - это параметры View-элементов, т.е. все те параметры, что меняются через вкладку Properties. Также можно вносить изменения прямо редакторе xml-кода и изменения будут отображаться во вкладке Design. Например, изменим текст у TextView. Вместо «new layout file myscreen for activity», напомним текст «some new text»

```
MainActivity.java x activity_main.xml x myscreen.xml x
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="some new text"
        android:id="@+id/textView" />

</LinearLayout>
```

Сохраняем. Открываем Design и наблюдаем изменения:



Layout-файл при смене ориентации экрана

По умолчанию layout-файл настраивается под вертикальную ориентацию экрана. Но что будет, если мы повернем смартфон и включится горизонтальная ориентация? Для эксперимента изменим myscreen.xml: добавим вертикальный ряд кнопок и изменим надпись. Соответствующий xml-код:

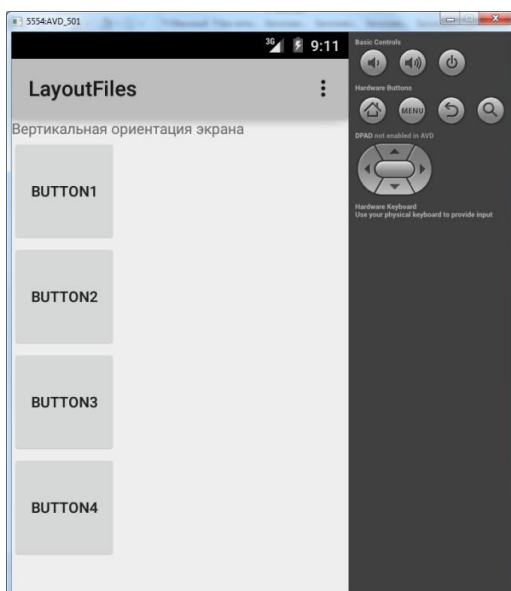
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Вертикальная ориентация экрана">
```

```

</TextView>
<LinearLayout
    android:id="@+id/linearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <Button
        android:id="@+id/button1"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:text="Button1">
    </Button>
    <Button
        android:id="@+id/button2"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:text="Button2">
    </Button>
    <Button
        android:id="@+id/button3"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:text="Button3">
    </Button>
    <Button
        android:id="@+id/button4"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:text="Button4">
    </Button>
</LinearLayout>
</LinearLayout>

```

Сохраним файл, запустим приложение:

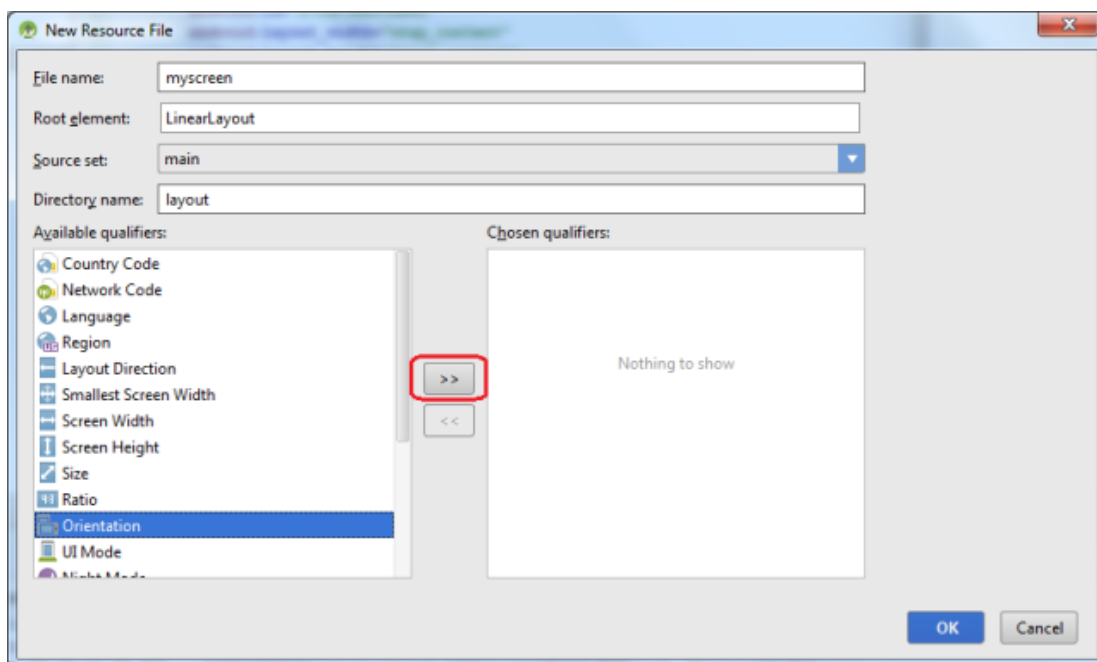


В вертикальной ориентации все ок.

Нажмем в эмуляторе CTRL+F12: ориентация сменилась на горизонтальную, и кнопки уже не влезают в экран:

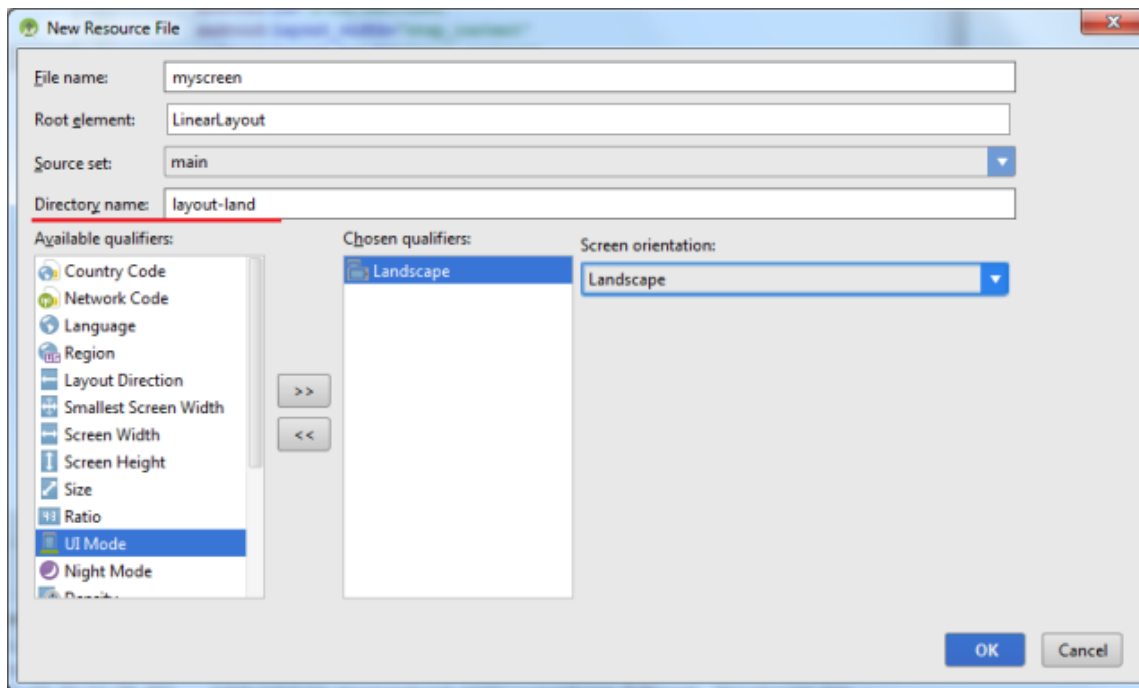


Т.е. необходим еще один layout-файл, который в этом случае выводил бы кнопки горизонтально. Но как дать знать Activity, что она в вертикальной ориентации должна использовать один layout-файл, а в горизонтальной – другой? Есть возможность создать layout-файл, который будет использоваться приложением, когда устройство находится в горизонтальной ориентации. Создание такого файла почти не отличается от создания обычного layout-файла. Становимся на папку res/layout и создаем новый Layout resource file. Название файла указываем то же самое: myscreen. Осталось добавить квалификатор, который даст приложению понять, что этот layout-файл надо использовать в горизонтальной ориентации. Для этого в списке квалификаторов слева снизу находим Orientation:



И нажимаем кнопку со стрелкой вправо. Тем самым мы включили использование квалификатора ориентации. Далее необходимо указать горизонтальную ориентацию: Landscape (выбрать это значение из выпадающего списка).

Обратите внимание, что изменилось значение поля Directory name:

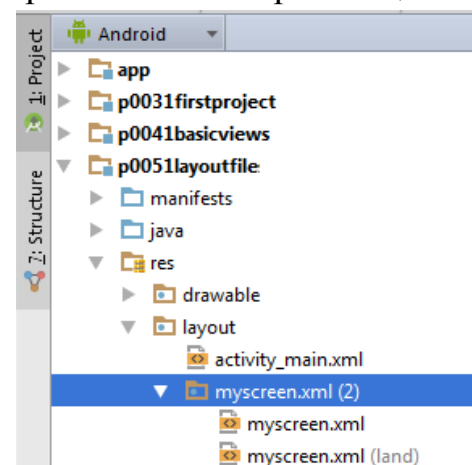
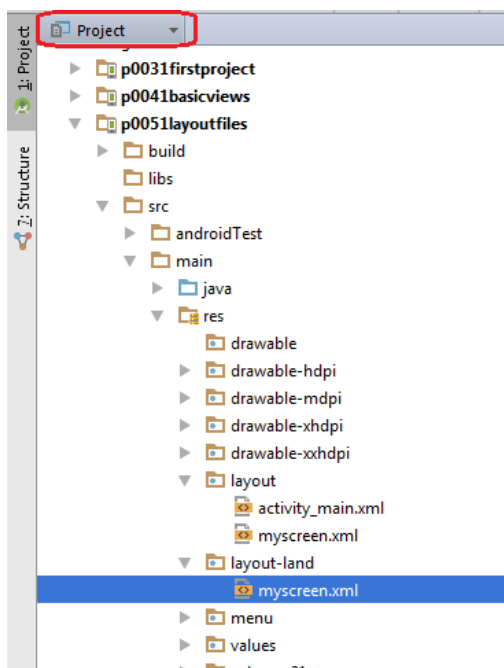


Настройкой квалификатора мы указали, что наш новый layout-файл будет создан в папке res/layout-land, а не res/layout, как обычно. Т.е. квалификатор –land указывает на то, что layout-файлы из этой папки будут использованы в горизонтальной ориентации устройства.

Нажимаем «OK».

Посмотрим на структуру модуля. Видим, что у нас теперь два файла myscreen: обычный и land. Можно это же увидеть в структуре папок.

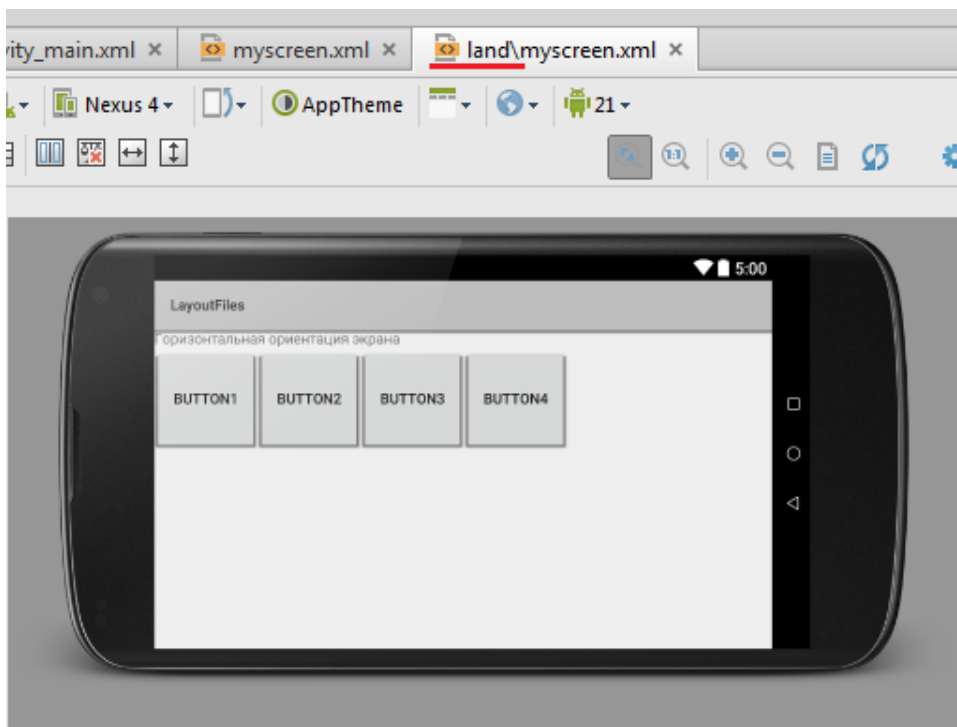
Для этого сверху поменяйте вид проекта с Android на Project:



Откроем двойным кликом файл `res/layout-land/myscreen` и поменяем его содержимое на такой xml-код:

```
<?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="vertical">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Горизонтальная ориентация экрана">
    </TextView>
    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/button1"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:text="Button1">
        </Button>
        <Button
            android:id="@+id/button2"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:text="Button2">
        </Button>
        <Button
            android:id="@+id/button3"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:text="Button3">
        </Button>
        <Button
            android:id="@+id/button4"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:text="Button4">
        </Button>
    </LinearLayout>
</LinearLayout>
```

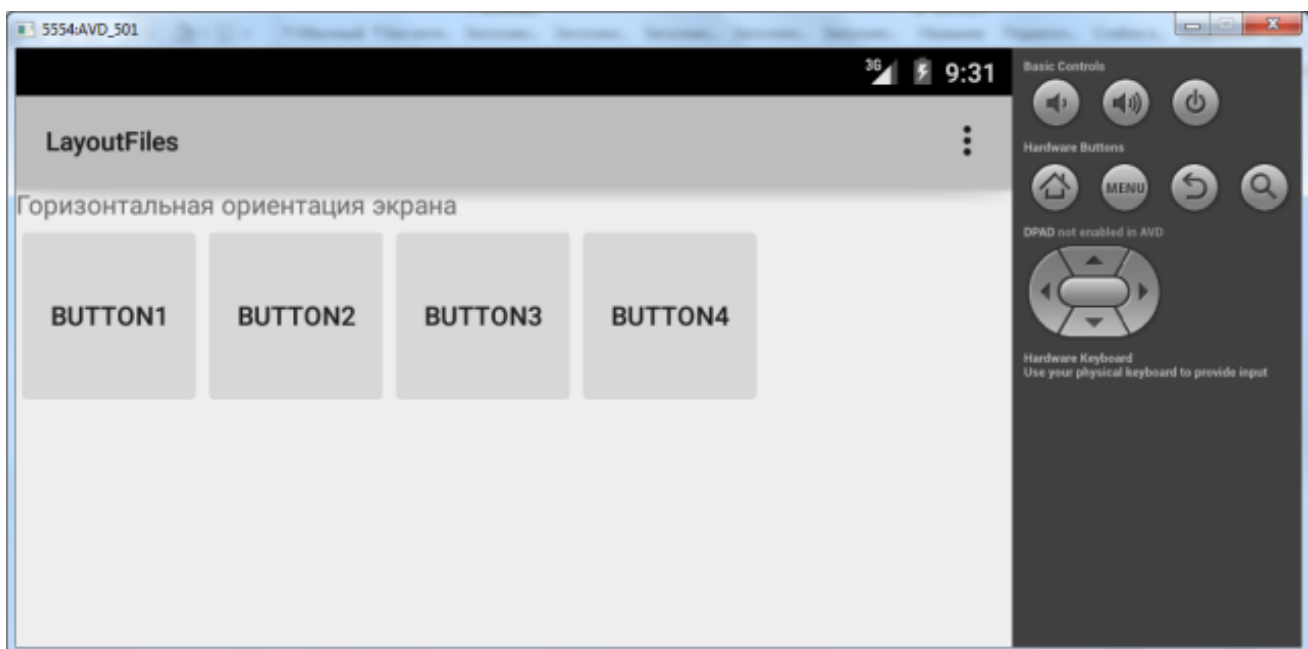
Вкладка Design покажет следующее:



В этом layout файле кнопки расположены горизонтально, чтобы они адекватно отображались в горизонтальной ориентации.

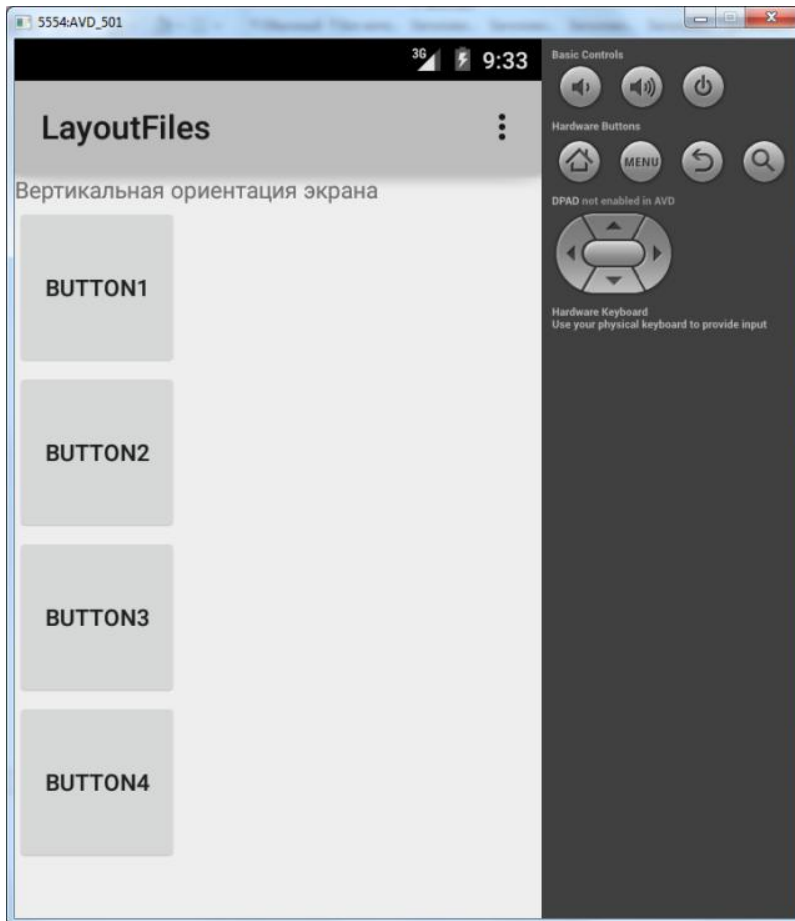
Обратите внимание на название файла сверху. Там присутствует квалификатор land.

Запустим приложение:



Activity читает layout-файл, который был указан в методе setContentView, т.е. myscreen.xml и отображает его содержимое. При этом оно учитывает ориентацию устройства, и в случае горизонтальной ориентации берет myscreen из папки res/layout-land (если он, конечно, там существует).

Переключим ориентацию CTRL+F12:



Activity понимает, что находится в вертикальной ориентации, и использует layout-файл myscreen из папки res/layout.