Лабораторная работа № 6. Разработка приложения с использованием Android Studio

Цель лабораторной работы:

Разработка простого приложения, помогающего понять структуру приложения, применить разные разметки, сменять ориентацию приложения, освоить основные операторы.

Задачи лабораторной работы:

- создать новое приложение «Угадай число» и изучить его структуру;
- настроить интерфейс приложения;
- реализовать логику приложения;
- изменить цветовую схему;
- добавить локализацию приложения;
- реализовать макет для альбомной ориентации каждого приложения;
- реализовать приложение «Угадай число» на языке Kotlin;
- разработать приложение калькулятор на языке Java или Kotlin.

Содержание

Для достижения поставленной цели в лабораторной работе создадим приложение в среде разработки Android Studio, подробно рассмотрим структуру полученного проекта и разберём назначение основных его элементов.

Разработать приложение "Угадай число". Суть приложения в том, что программа случайным образом "загадывает" число от 0 до 200, а пользователь должен угадать это число. При каждом вводе числа, программа сообщает пользователю результат: введённое число больше загаданного, меньше или же "Ура, победа!" число угадано.

Разрабатываемое приложение выполняет свои функции только когда видимо на экране, когда оно не видимо его работа приостанавливается, т. е. имеем дело с приложением переднего плана. Для выполнения всей работы достаточно определить одну активность в приложении, фоновые процессы не предусмотрены.

Далее в работе рассмотрим простейшие элементы интерфейса пользователя и добавим их в приложение, а также рассмотрим вопросы, связанные непосредственно с программированием: научимся обрабатывать события, возникающие при взаимодействии приложения с пользователем; реализуем логику проверки числа на

совпадение с загаданным.

Критерии оценивания

4 — приложение на основе примера из лабораторной работы и ответы на вопросы по структуре проекта;

5-6 — приложение на основе примера «Угадай число» из лабораторной работы и ответы на контрольные вопросы;

6-7 — приложение «Угадай число» на основе примера из лабораторной работы, задание 3.1 для самостоятельной работы, задание 3.2 — реализовать пример на языке Kotlin и ответы на контрольные вопросы. **Генерация кода на языке Kotlin из java-проекта не разрешается**;

8 — приложение «Угадай число» на основе примера из лабораторной работы, задания 3.1 и 3.3 для самостоятельной работы на языке Java и 3.2 — на языке Kotlin с дополнительным функционалом, и ответы на контрольные вопросы. **Генерация кода на языке Kotlin из java-проекта не разрешается**;

9 — приложение «Угадай число» на основе примера из лабораторной работы, задание 3.1 для самостоятельной работы на языке Java, задания 3.2 и 3.3 — на языке Kotlin, дополнительный функционал, предложенный студентом самостоятельно, и ответы на контрольные вопросы

1. Теоретические сведения

Компоненты

Всего в Android-приложениях существует четыре типа компонентов из уровня каркаса приложений: деятельность (Activity), служба (Service), приемник широковещательных намерений (Broadcast Receiver), контент-провайдер (Content Provider). Взаимодействие компонентов осуществляется с помощью объектов Intent.

Activity представляет собой визуальный пользовательский интерфейс для приложения – окно. Activity может также использовать дополнительные окна, например всплывающее диалоговое окно. Все деятельности реализуются как подкласс базового класса Activity.

Компонент **Service** не имеет визуального интерфейса пользователя и выполняется в фоновом режиме в течение неопределенного периода времени, пока не завершит свою работу. Service, как правило, требуется для длительных операций или для обеспечения работы удаленных процессов, но в общем случае это просто режим, который функционирует, когда приложение не в фокусе. Примером такого процесса может стать прослушивание музыки в то время, когда пользователь делает что-то другое, или получение данных по сети без блокирования текущей активности. Приложения могут подключаться к компоненту Service или запускать его, если он не запущен, а также останавливать уже запущенные компоненты.

Broadcast Receiver используется для отслеживания внешних событий и

реакции на них. Приложение может иметь несколько компонентов Broadcast Receiver, чтобы ответить на любые объявления, которые оно считает важными. При этом каждый компонент Broadcast Receiver будет определять код, который выполнится после возникновения конкретного внешнего события. С помощью класса Notification Manager можно сообщить пользователю информацию, требующую его внимания. Примером оповещений может быть сигнал о том, что информация загружена на устройство и доступна к использованию.

Content Provider делает определенный набор данных, используемых приложением, доступным для других приложений. Данные могут быть сохранены в файловой системе, базе данных SQLite, сети или любом другом месте, к которому приложение может иметь доступ. Контент-провайдеры для безопасного доступа к данным используют механизм разрешений. Посредством Content Provider другое приложение может запрашивать данные и, если выставлены соответствующие разрешения, изменять их. Например, система Android содержит Content Provider, который управляет пользовательской информацией о контактах.

Intent – специальные классы в коде приложения, которые определяют и описывают запросы приложения на выполнение каких-либо операций. Намерения добавляют слой, позволяющий оперировать компонентами с целью их повторного использования и замещения. В некоторых случаях это может быть очень мощным средством интеграции приложений. Главная особенность платформы Android состоит в том, что одно приложение может использовать элементы других приложений при условии, что эти приложения разрешают использовать свои компоненты. При этом ваше приложение не включает код другого приложения или ссылки на него, а просто запускает нужный элемент другого приложения [1-5].

Графическое представление Activity

Основу Android-приложения составляет Activity – рабочее окно. В конкретный момент времени обычно отображается одно Activity и занимает весь экран. Работа с набором окон осуществляется путем переключения различных Activity. В качестве примера можно рассмотреть почтовое приложение: в нем одно Activity – список писем, другое – просмотр письма, третье – настройки ящика.

С программной точки зрения Activity представлено файлом графической разметки (activity_main.xml на рисунке 1) и соответствующим java-классом (MainActivity на рисунке 1), реализующем запуск данного Activity с требуемым графическим представлением и последующей обработкой

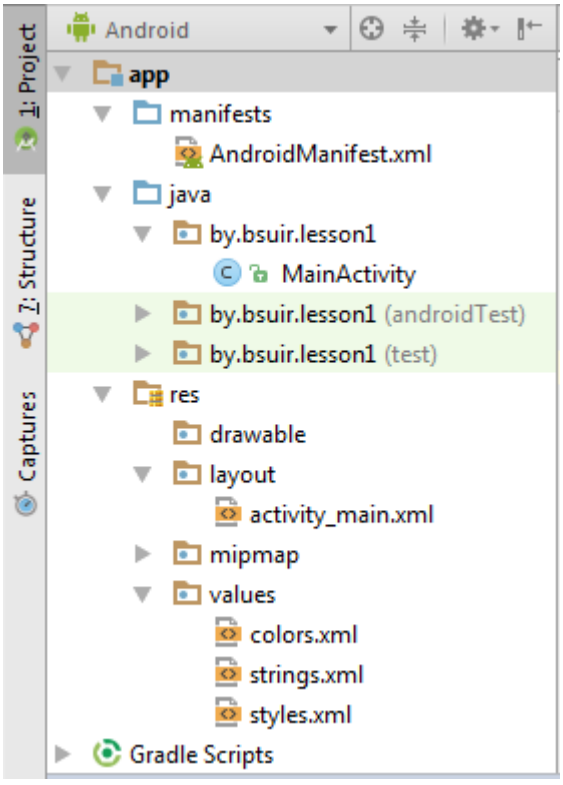


Рис. 1 Структура проекта в Android Studio

событий.

Графическое представление Activity формируется из различных компонентов (кнопка, поле ввода, чекбокс и т. д.), называемых View (рисунок 2).

Графическое представление каждого Activity в виде требуемого набора и взаимного расположения View-элементов хранится в xml-файле папки layout. Данный файл графического представления прописывается в соответствующем java-классе. При запуске приложения Activity читает этот файл и отображает его содержимое.

После создания проекта файл activity_main.xml открывается по умолчанию (рисунок 3) в режиме конструктора (вкладка Design), помимо которого существует соответсвующее xml-описание графического представления (вкладка Text).

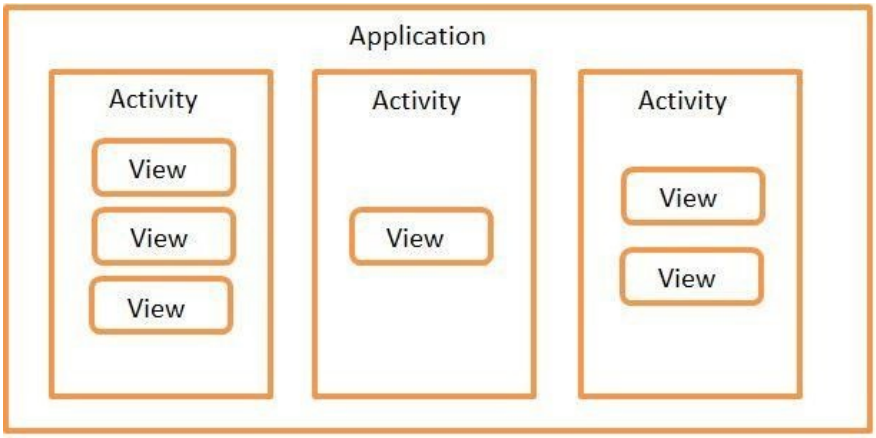


Рисунок 2. Представление приложения Android как набора окон (Activity) с View-компонентами

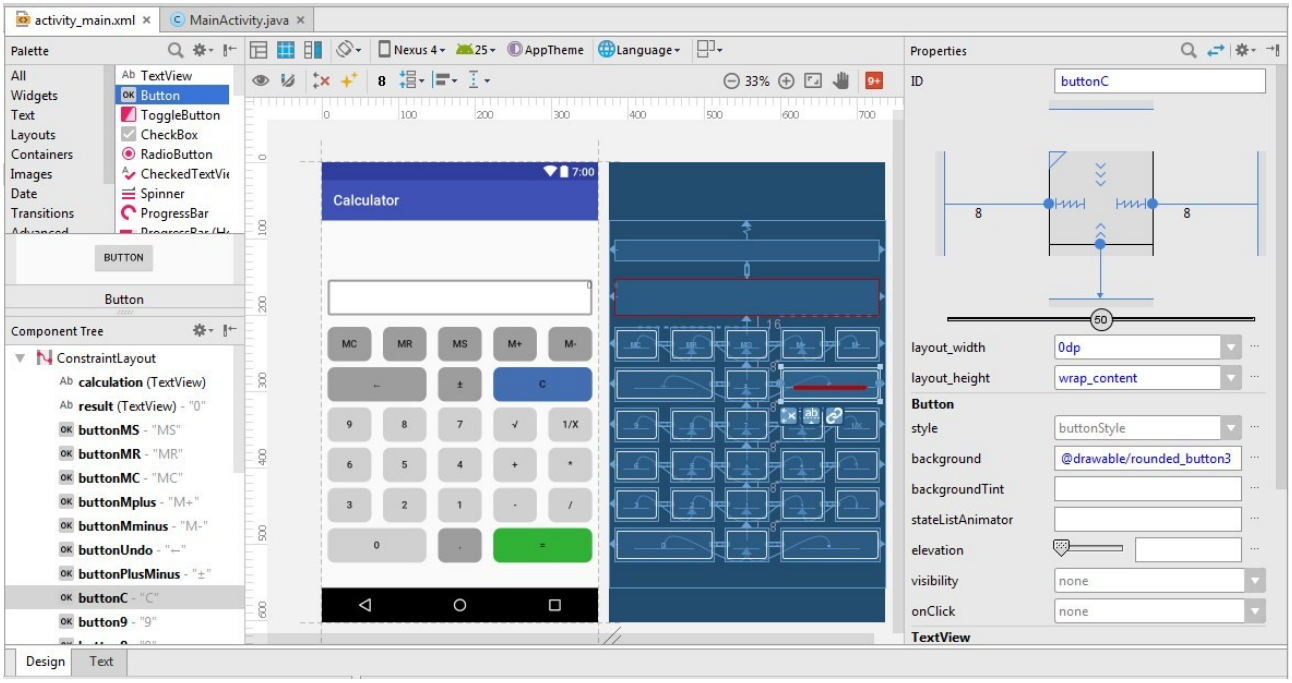


Рисунок 3. Графическое представление Activity. Вкладка Design

В центре вкладки Design по умолчанию расположены два графических редактора, представляющие собой экран мобильного устройства в режиме Design и режиме Blueprint. Режим Blueprint специально разработан для удобства размещения элементов и настройки их взаимосвязей.

В левой части вкладки Design отображаются окно Palette со списком всех возможных виджетов и окно Component Tree с деревом компонентов, используемых в данном графическом представлении.

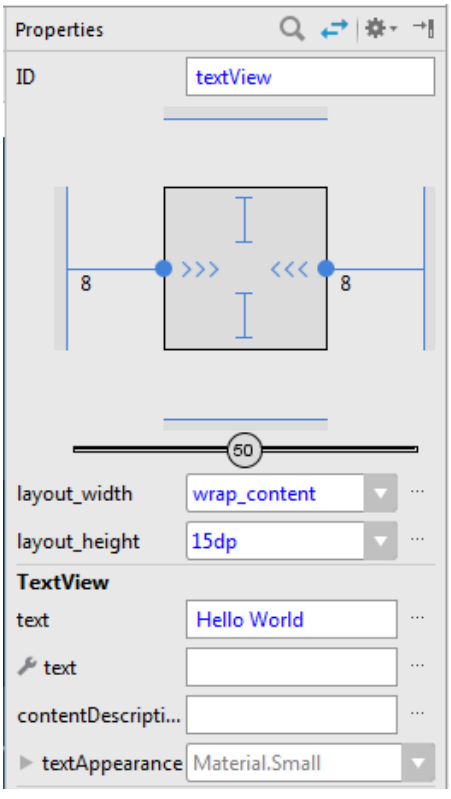
В правой части кладки вкладки Design отображается окно Properties с набором свойств для каждого выбранного элемента.

Для размещения View-компонентов используются специальные контейнеры

(ViewGroup), называемые Layout. Layout бывают различных типов (LinearLayout, RelativeLayout, FrameLayout, TableLayout, ConstraintLayout и т. д.) и отвечают за то, как будут расположены их дочерние View-компоненты на экране (таблицей, строкой, столбцом). Android Studio по умолчанию использует ConstraintLayout в качестве корневого контейнера для создания разметки экрана и размещения компонентов. Данный контейнер обладает широким спектром возможностей, что позволяет реализовывать сложные взаиморасположения элементов на экране.

Для добавления на экран требуемого компонента необходимо найти его в списке Palette и переместить мышкой на экран в режиме Design или режиме Blueprint. После этого компонент появится на обоих экранах, а также в окне Component Tree.

Квадратные опорные точки в углах компонента (рисунок 4, а) позволяют изменять его размеры.



а *б*

а – изображение компонента TextView в режиме BluePrint; *б* – свойства компонента TextView в окне Properties

Рисунок 4 – Пример изображения компонента TextView и его свойств

Круглые опорные точки (рисунок 4, а), расположенные по сторонам виджета, позволяют создавать привязки (constraints) к сторонам контейнера или другим компонентам и управлять отступами от краев экрана и других компонентов. Для создания своей собственной привязки необходимо нажать круглую опорную точку на одной из сторон виджета, которую требуется привязать. Далее, не отпуская кнопку мыши, прочертить линию к виджету или краю контейнера. В результате будет создана связь constraint.

В окне Properties (рисунок 4, б) находится схематичное изображение компонента, а также указаны его свойства.

Базовым параметром каждого компонента является id – идентификационный номер. Необходимо давать компонентам уникальные и осмысленные имена, чтобы с ними в последующем было удобно работать из java-кода.

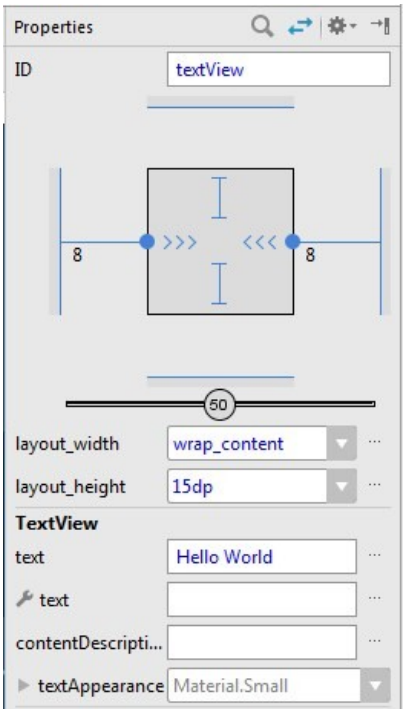
Каждый компонент характеризуется шириной (layout_width) и высотой (layout_heigth). Для установки размера компонента внутри его схематичного изображения в окне Properties необходмо выбрать один из трех возможных типов линий для ширины и высоты:

- линии  означают атрибут wrap content и устанавливают размер элемента по его текущему содержимому;
- линии означают атрибут fixed и устанавливают фиксированный размер компонента в dp (density-independent pixels, абстрактная единица измерения, позволяющая приложению выглядеть одинаково на различных экранах и разрешениях). Размер компонента можно отредактировать вручную с помощью параметра layout_width или layout_heigth;
- линии  означают атрибут match constraints и устанавливают максимально допустимый размер элемента.

Переключение между типами линий осуществляется путем нажатия значка, их изображающего.

По сторонам схематичного изображения компонента в окне Properties имеются числа (рисунок 5). Они отвечают за атрибут margin (отступы). Если подвести к ним мышку, то появится выпадающий список со значениями для изменения отступа.

Несколько компонентов можно объединять в одну



цепь (вертикальную или горизонтальную). Для этого необходимо выделить компоненты и через контекстное меню выбрать оптию Center Horizontally или Center Vertically (рисунок 6, а). Рядом с выбранными компонентами появится символ цепи, а между ними будет нарисована связь (рисунок 6, б). Если последовательно нажимать значок цепи, то кнопки будут центрироваться с разными стилями.

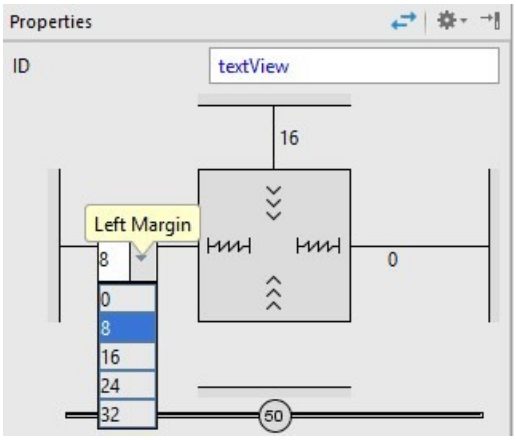
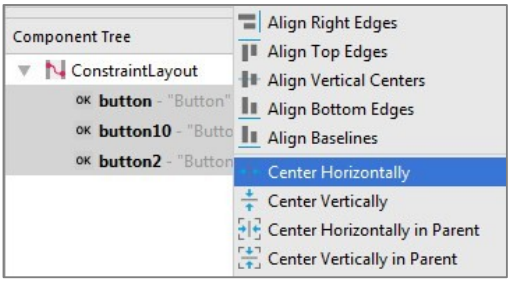
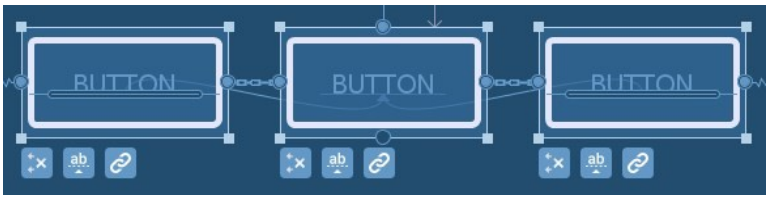


Рисунок 5 – Пример установки отступов компонента





а б

а – объединение компонентов в горизонтальную цепь с помощью опции

Center Horizontally; б – визуальное отображение компонентов в цепи

Рисунок 6 – Пример создания цепи компонентов

Можно присвоить компонентам цепи весовые коэффициенты для пропорционального масштабирования их размеров (при этом соответствующий параметр – высота или ширина – должны иметь атрибут match constraints). Для этих целей используются атрибуты layout_constraintHorizontal_weight и layout_constraintVertical_weight (их можно найти в расширенном списке Properties).

Закругленный прямоугольник на изображении компонента в режиме Blueprint (рисунок 6, б) указывает на базовую линию текста и используется при выравнивании по базовой линии другого компонента.

Layout-файл при смене ориентации экрана

По умолчанию layout-файл настроен под вертикальную ориентацию экрана. Однако при повороте смартфона включится горизонтальная ориентация, что может

привести к некорректному отображению View-элементов. Для устранения данной проблемы необходимо создать еще один layout-файл для горизонтальной ориентации экрана. Эта задача решается с помощью опции Create Landscape Variation (рисунок 7).

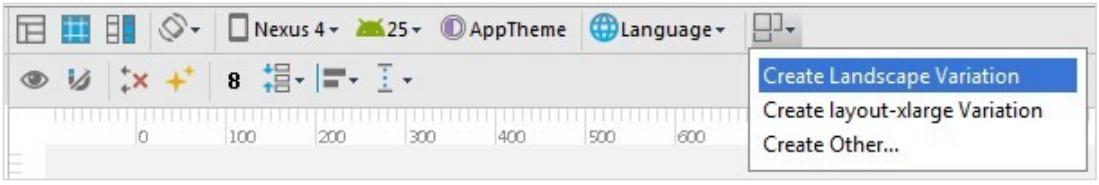


Рисунок 7 – Создание графического представления для горизонтальной ориентации экрана

В структуре проекта появится новый xml-файл activity_main.xml(land). При этом текущее содержимое файла activity_main.xml будет скопировано в горизонтальное представление activity_main.xml(land). Далее меняем параметры viewэлементов таким образом, чтобы в горизонтальной ориентации экрана они отображались корректно. При этом id элементов не меняем!

Все последующие изменения activity_main.xml никак не будут влиять на содержимое activity_main.xml(land), в связи с чем создавать горизонтальное представление целесообразно после размещения всех необходимых компонентов в вертикальном представлении и корректировке id-параметров этих компонетов.

При запуске приложения Activity прочитает подключенный в коде layoutфайл и отобразит его содержимое. При этом будет учтена ориентация экрана, и в случае горизонтального расположения автоматически отобразится файл land.

Файл MainActivity.java. Подключение графического представления к Activity

При запуске деятельности система должна получить ссылку на корневой узел дерева разметки, который будет использоваться для прорисовки графического интерфейса на экране мобильного устройства. Для этого в методе onCreate( ) необходимо вызвать метод setContentView( ).

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

Метод setContentView(int) 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 и хранится в файле R.java. Имена этих id-констант совпадают с именами файлов ресурсов (без расширений).

Можно создать новый xml-файл в папке res > layout и прописать его вместо

activity_main в методе setContentView(int). После запуска приложения отобразится новый файл разметки.

Доступ к элементам экрана из кода

Чтобы обратиться к элементу экрана из кода, необходим его id. Он прописывается в окне Properties либо в xml-коде:

```
<Button

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="OK"
            android:id="@+id/btnOK" />
```

Зная id View-элемента, обратиться к нему из кода можно по константе R.id.btnOK. Для этого понадобится метод findViewById:

```
public class MainActivity extends ActionBarActivity{

    private Button btnOK;

    @Override
    protected void onCreate(Bundle
savedInstanceState)
{         super.onCreate(savedInstan
ceState);
setContentView(R.layout.activity_mai
n);         initViews();
    }

    private void
initViews() {
// находим View-
элементы
        btnOK = (Button) findViewById(R.id.btnOK);
    }
```

Следует отметить, что в приведенном фрагменте кода обнаружение Viewэлементов вынесено в отдельный пользовательский (не являющийся библиотечным) метод initViews() для реализации принципа модульного программирования.

Обработка событий на примере нажатия кнопки

Механизм обработки нажатия кнопки основан на использовании интерфейса View.OnClickListener и реализации метода onClick, в котором и прописывается логика действий в ответ на нажатие (рисунок 8).

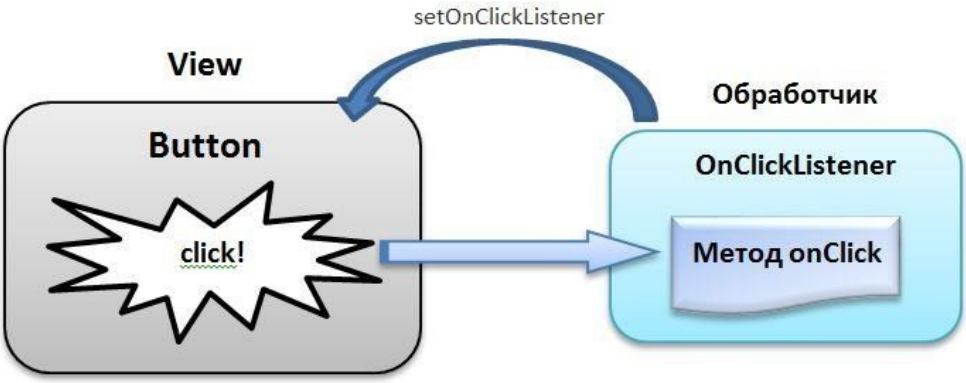


Рисунок 8 – Механизм обрабоки нажатия кнопки на основе интерфейса
View.OnClickListener

При этом обработчик (его также называют слушателем – listener) присваивается кнопке с помощью метода setOnClickListener. Когда нажимают кнопку, обработчик реагирует и выполняет код из метода onClick. Для реализации данного механизма необходимо выполнить следующие шаги:

- создать обработчик (объект от интерфейса View.OnClickListener);
- заполнить метод onClick (т. к. в интерфейсе он не реализован, а только заявлен);
- присвоить обработчик кнопке (используем метод setOnClickListener).

Система обработки событий готова, а именно, когда нажимают кнопку, обработчик реагирует и выполняет код из метода onClick. Пример:

```
OnClickListener listener = new OnClickListener() {
   @Override
   public void onClick(View v) {
        // метод, который будет вызван по нажатию
       }
   };

button.setOnClickListener(listener);
```

Существуют различные способы программной реализации обработки нажатий в зависимости от требуемого набора кнопок одного Activity (с использованием в xml-представлении атрибута onClick, своего обработчика для каждого Viewэлемента, одного обработчика для нескольких View-элементов). Однако наиболее оптимальным является использование Activity в качестве единого обработчика. В данном случае сам класс Activity реализует интерфейс View.OnClickListener:

```
public class MainActivity extends
ActionBarActivity implements View.OnClickListener{
```

```
    private Button btnOK, btnCancel;

    @Override
    protected void onCreate(Bundle
savedInstanceState)
{         super.onCreate(savedInstan
ceState);
setContentView(R.layout.activity_mai
n);         initViews();
    }

    private void initViews() {
// находим View-элементы
btnOK = (Button)
findViewById(R.id.btnOK);
btnCancel = (Button)
findViewById(R.id.btnCancel);
        // подключаем обработчик к кнопкам
btnOK.setOnClickListener(this);
        btnCancel.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {

switch
(v.getId()) {
case
R.id.btnOK:

btnOK.setText("Hello
");
btnOK.setEnabled(fal
se);
btnCancel.setEnabled
(true);
break;
case R.id.btnCancel:
btnCancel.setText("G
oodbye");
btnCancel.setEnabled
(false);

  btnOK.setEnabled(true);
  break;
         }
     }
  }
```

Для считывания данных из текстовых полей (например, из поля et_message) используется следующий код:

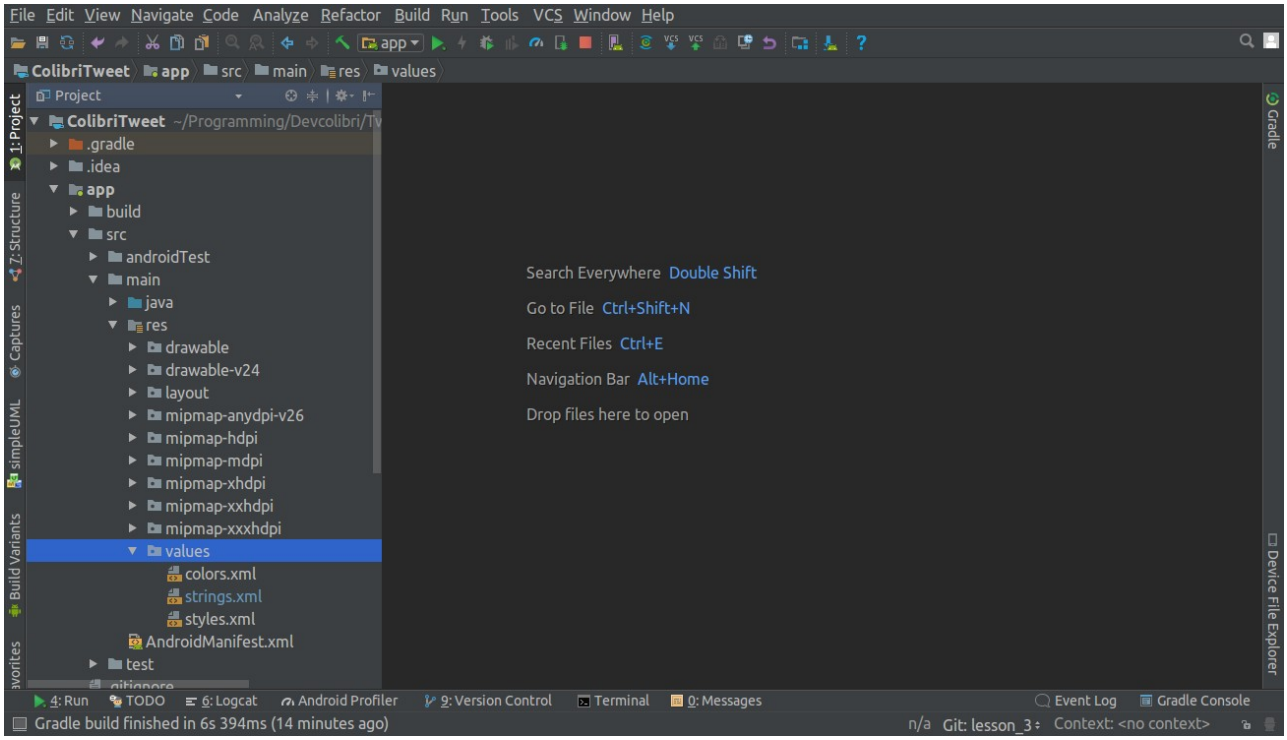
```
String message = et_message.getText().toString();
```

Ресурсы

Отделение ресурсов от кода программы

Независимо от используемой среды разработки, весьма разумно отделять используемые приложением ресурсы от кода. Внешние ресурсы легче поддерживать, обновлять и контролировать. Такая практика также позволяет описывать альтернативные ресурсы для поддержки вашим приложением различного оборудования и реализовывать локализацию приложения.

Приложения Android используют разнообразные ресурсы из внешних (по отношению к коду) файлов, от простых (строки и цвета) до более сложных (изображения, анимации, визуальные стили). Чрезвычайно полезно также отделять от кода такие важные ресурсы, как разметки экранов (Layout), используемые в Активностях. Android автоматически выбирает наиболее подходящие варианты из дерева ресурсов приложения, содержащие разные значения для разных аппаратных конфигураций, языков и регионов, не требуя при этом ни единой строчки кода.



Видим три файлы:

colors.xml — содержит в себе все цвета.

strings.xml — хранит все строковые ресурсы.

styles.xml — и все стили приложения.

Создание ресурсов

Ресурсы приложения хранятся в каталоге res в дереве каталогов проекта. При создании проекта автоматически создается каталог res с подкаталогами values, layout и drawable-*, в которых хранятся, соответственно: строковые константы,

разметка по умолчанию и иконка приложения для разных плотностей пикселей на экране.

Для девяти главных типов ресурсов используются разные подкаталоги каталога res, это:

• простые значения

• изображения

• разметка

• анимация

• стили

• меню

• настройки поиска

• XML

• «сырые» данные

При сборке пакета .apk эти ресурсы максимально эффективно компилируются и включаются в пакет.

Для работы с ресурсами внутри кода автоматически генерируется файл класс R, содержащий ссылки на все ресурсы. Имена файлов ресурсов могут содержать только латинские буквы в нижнем регистре, подчеркивания и точки.

Простые значения (values)

Android поддерживает следующие типы значений: строки, цвета, размеры и массивы (строковые и целочисленные). Эти данные хранятся в виде XML-файла в каталоге res/values. Ниже показан пример подобного файла:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">To Do List</string>
<color name="app_background">#FF0000FF</color>
<dimen name="default_border">5px</dimen>
<array name="string_array">
<item>Item 1</item>
<item>Item 2</item>
<item>Item 3</item>
</array>
<array name="integer_array">
```

```
<item>3</item>
<item>2</item>
<item>1</item>
</array>
</resources>
```

В примере содержатся все доступные типы простых значений, но на самом деле каждый тип ресурсов хранится в отдельном файле, например, res/values/arrays.xml содержит массивы, а res/values/strings.xml – строковые константы.

Строки

Строковые константы определяются с помощью тэга <string>, как показано на примере:

```
<string name="greeting_msg">Привет!</string>
```

Для выделения текста в строках можно использовать HTML-тэги <b>, <i> и <u>.

Пример:

```
<string name="greeting_msg"><b>Привет</b>, Петр!</string>
```

При необходимости использования данной строки в коде программы используется вышеупомянутый класс R:

```
String greeting = getString(R.string.greeting_msg);
```

Цвета

Для описания цветов используется тэг <color>. Значения цвета указываются в шестнадцатеричном виде в одном из следующих форматов:

- #RGB
- #ARGB
- #RRGGBB
- #AARRGGBB

В примере показаны описания полупрозрачного красного цвета и непрозрачного зеленого:

```
<color name="transparent_red">#77FF0000</color>
<color name="opaque_green">#0F0</color>
```

Локализация приложения с помощью внешних ресурсов

Одно из основных преимуществ использования внешних по отношению к коду приложения ресурсов – возможность использования механизма автоматического выбора ресурсов. Пользуясь описанным ниже механизмом, можно создавать

индивидуальные ресурсы для различных аппаратных конфигураций, языков, регионов и т. д. Android во время выполнения приложения сам выберет наиболее подходящие ресурсы.

Для индивидуальной настройки приложения доступны следующие возможности:

• MCC (Mobile Country Code) и MNC (Mobile Network Code)

• Язык и регион. Например, en-rUS для английского языка в американском регионе (маленькая r – от «region»), ru для русского

• Размер экрана (small, medium или large)

• «Широкоформатность» экрана (long или notlong)

• Ориентация экрана (port, land или square)

• Плотность пикселей на экране (ldpi, mdpi, hdpi или nodpi)

• Тип сенсорного экрана (notouch, stylus или finger)

• Доступность клавиатуры (keysexposed, keyshidden или keyssoft)

• Тип ввода (nokeys, qwerty или 12key)

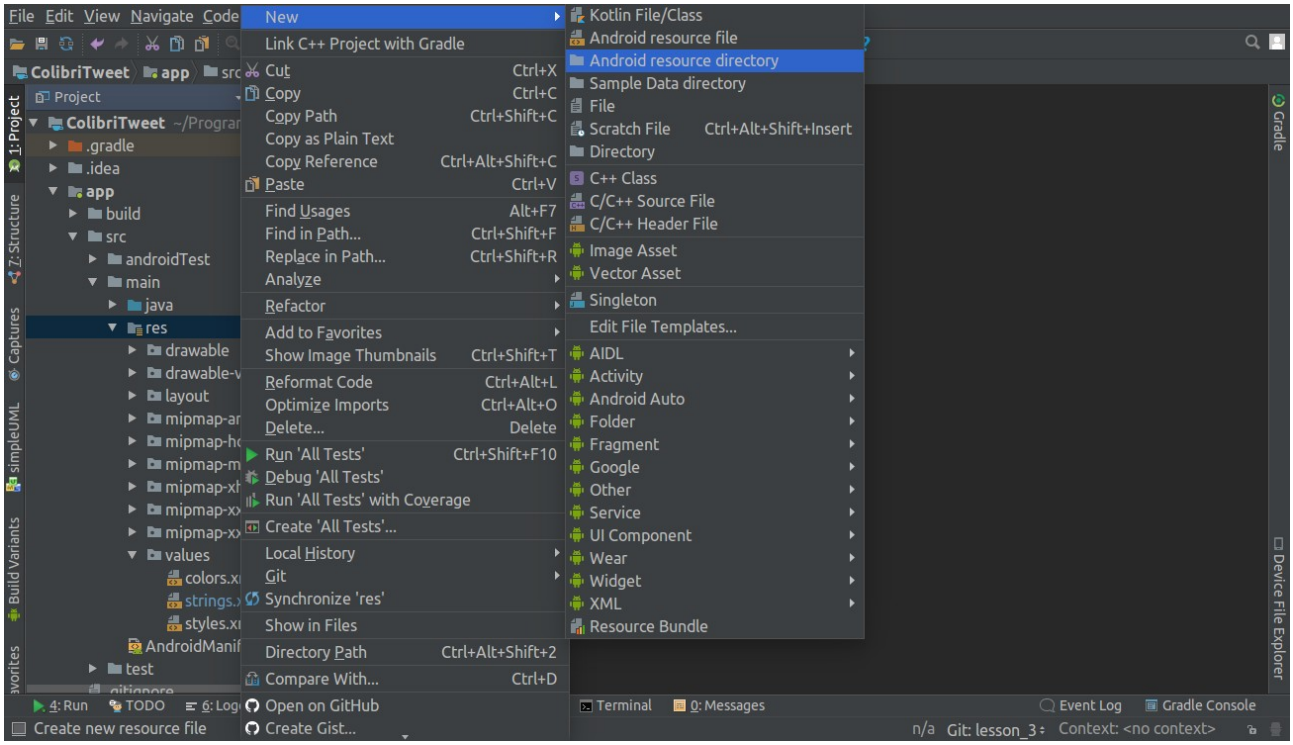
• Способ навигации (nonav, dpad, trackball или wheel)

Альтернативные ресурсы располагаются в подкаталогах каталога res, при этом используются модификаторы стандартных имен подкаталогов с ресурсами. Например, файл, содержащий строковые константы для русского языка, будет располагаться по следующему пути: res/values-ru/strings.xml. Модификатором в данном случае является суффикс -ru, добавленный к имени каталога values.

Как локализовать приложение?

Мы будем поддерживать два языка в нашем приложении: английский и русский. Для локализации приложения используют файл `strings.xml`. Туда помещается весь текст вашего приложения, который должен быть подвержен локализации. Чтобы добавить поддержку ещё одного языка, надо создать папку с именем `values-language`. Т.к. мы поддерживаем русский язык, то создадим папку `values-ru`.

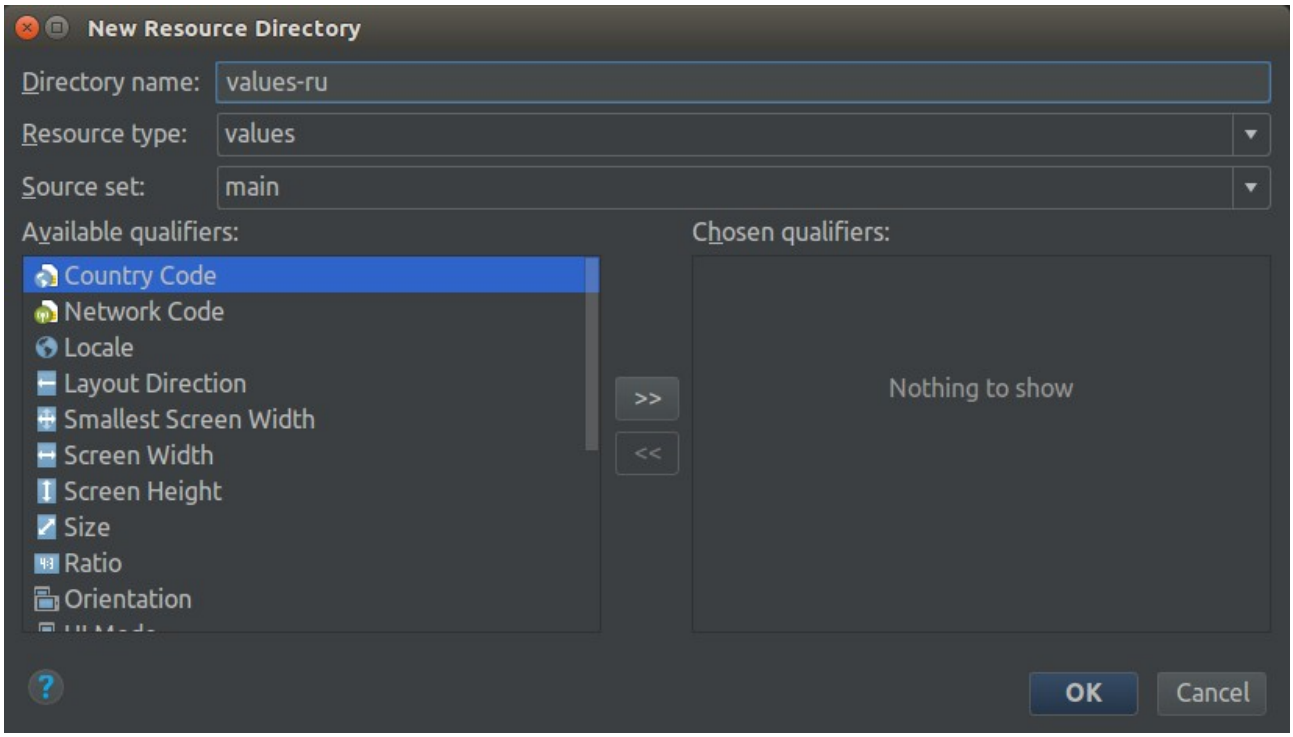
Для этого нажмём правой кнопкой на папку `res` и выберем пункт `New -> Android resource directory:`



При создании директории у нас появится окно, в котором надо заполнить поля следующими значениями:

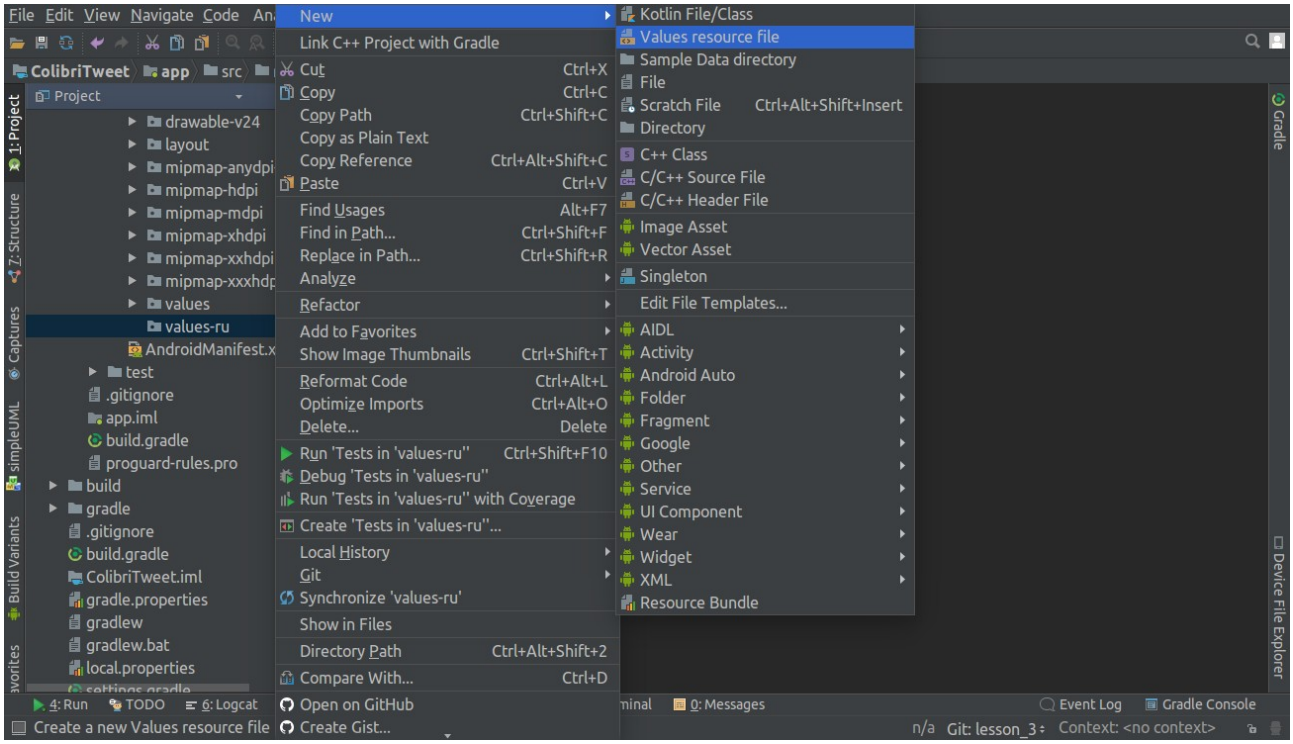
- `Directory name – values-ru`
- `Resource type – values`
- `Source set – main`

Вот как это должно выглядеть:

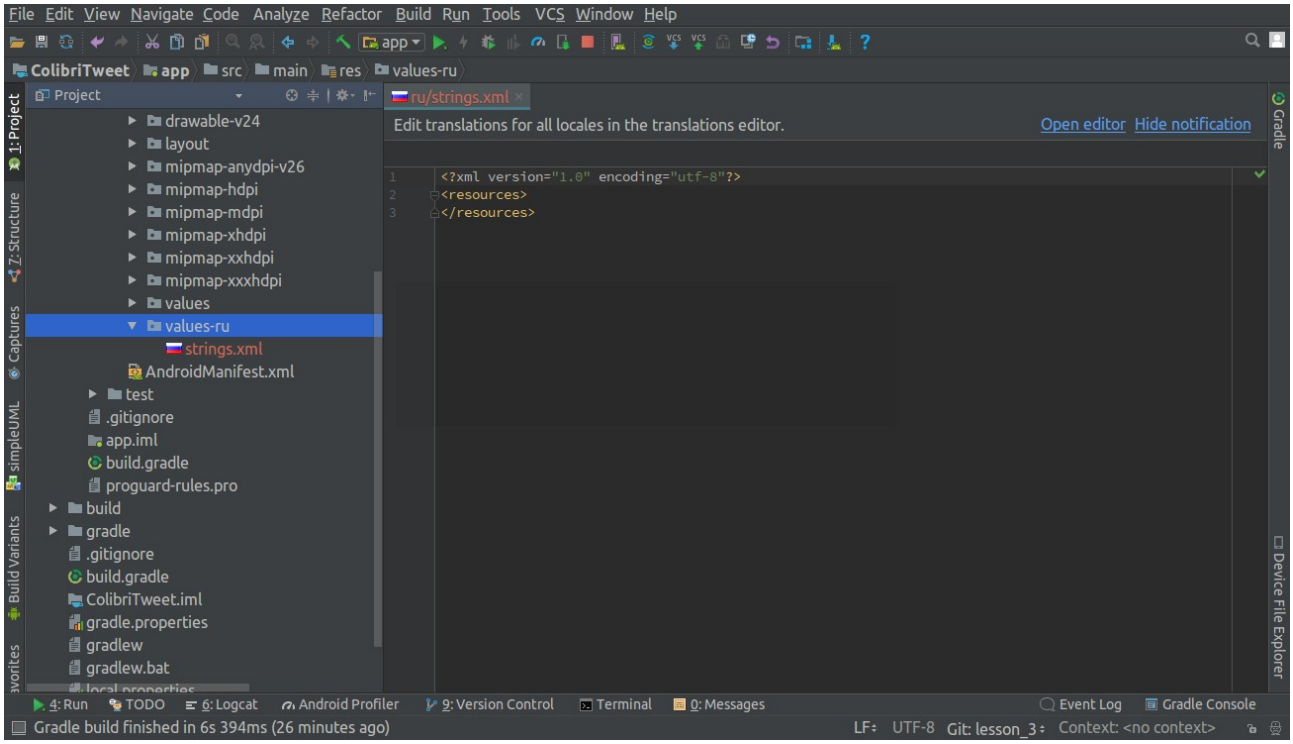


После этого видим, что у нас создалась папка `values-ru`. Отлично, создадим

новый файл `strings.xml`. Для этого нажимаем правой кнопкой по папке `values-ru`, выбираем `New -> Values resource file`:

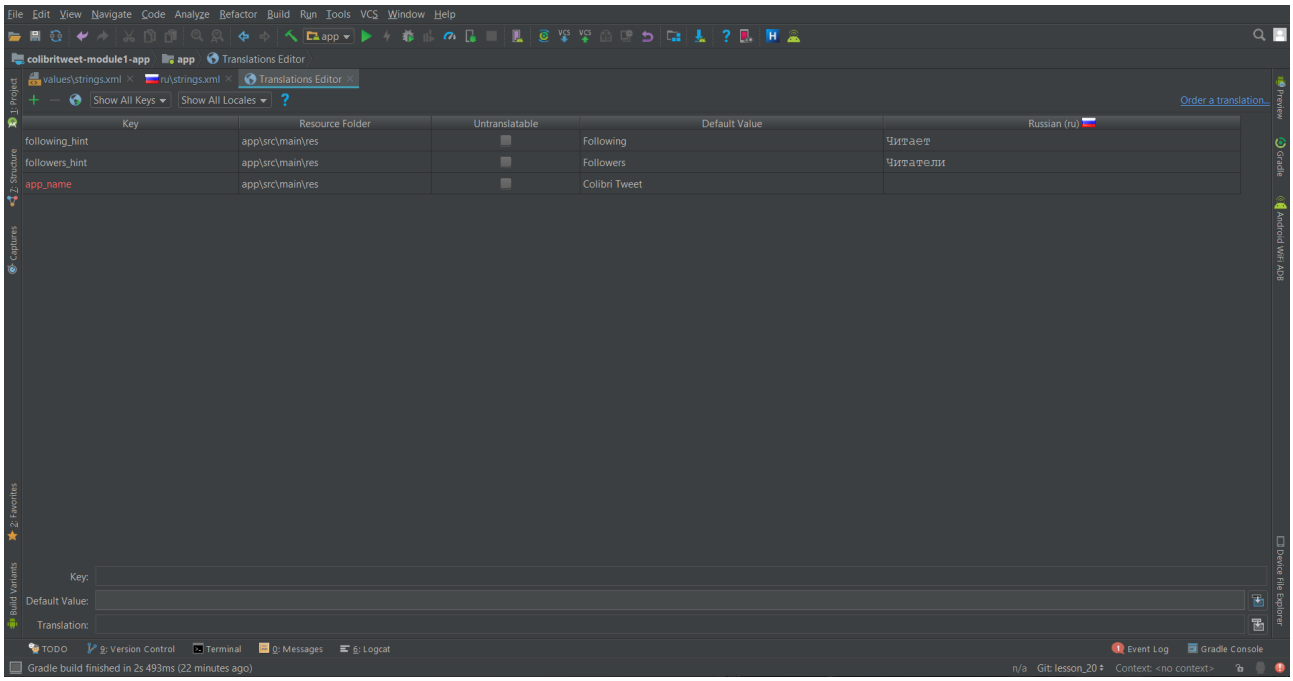


Затем появится окно, в котором необходимо ввести имя файла. Вводим `strings`. И у нас создался файл, иконка которого выглядит в виде флага России.



Отлично. Теперь всё, что нам надо сделать – это добавлять ресурсы в эту папку с таким же именем, но значения мы будем писать на русском языке. Система автоматически будет брать значения из этого файла, если у пользователя на устройстве установлен русский язык.

Стоит отметить, что помимо ручного заполнения файлов с ресурсами, существует возможность делать это через специальный редактор. Чтобы в него попасть необходимо, открыв файл со строковыми ресурсами (любого языка), в правом верхнем углу нажать кнопку «Open editor» (рус. *открыть редактор*):



2. Упражнение. Создать приложение «Угадай число»

Выполнить все этапы из упражнения 2.1 и создать приложение «Угадай число».

2.1. Создать приложение и его структуру

Создайте новый проект в среде Android Studio. Процесс создания нового проекта и описание основных настроек подробно рассмотрен в лабораторной работе 5.

2.2. Настроить интерфейс приложения

До того, как начнем формировать интерфейс, имеет смысл подготовить возможность проверки разрабатываемого приложения на ошибки. Чтобы не загружать каждый раз приложение на реальное устройство, в Android SDK предусмотрена возможность использования виртуального устройства (AVD или Android virtual device), эмулирующего работу реального смартфона.

Пришло время задуматься о внешнем виде приложения. Для начала необходимо определить какие элементы графического интерфейса нам нужны, как эти элементы будут располагаться на форме и каким образом будет реализовано взаимодействие с пользователем.

Так как приложение очень простое, то и интерфейс особой сложностью отличаться не будет. Нам потребуется поле для ввода чисел (**TextEdit**), текстовая

метка для вывода информации (**TextView**) и кнопка для подтверждения введенного числа (**Button**). Располагать элементы интерфейса будем друг под другом, сверху информационная часть, ниже поле ввода, кнопку разместим в самом низу приложения. Взаимодействие приложения с пользователем организуется очень просто: пользователь вводит число в поле для ввода и нажимает кнопку, читает результат в информационном поле и, либо радуется победе, либо вводит новое число.

Если пользователь вводит правильное число, то приложение предлагает ему сыграть снова при этом кнопка будет играть роль подтверждения, а в информационное поле будет выведено приглашение к повторной игре.

Схематично интерфейс приложения изображен на рис. 2.1.

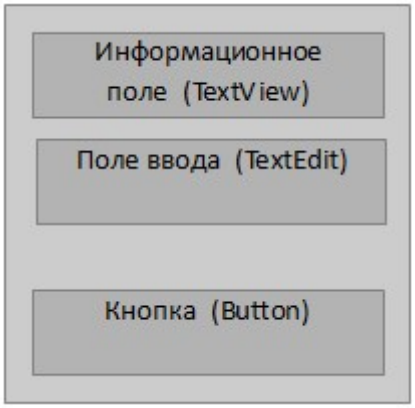


Рис. 2.1. Схема интерфейса приложения "Угадай число"

Нам необходимо добавить на форму три элемента: информационное поле (**TextView**), поле ввода (**TextEdit**) и кнопку (**Button**).

Android Studio поддерживает два способа для выполнения действий по формированию интерфейса приложения: первый основан на XML-разметке, второй относится к визуальному программированию и позволяет перетаскивать объекты интерфейса и размещать их на форме с помощью мыши. Считается, что визуальный способ подходит для новичков, а более продвинутые разработчики могут писать код вручную, однако чаще всего используется комбинированный подход.

Для формирования интерфейса будем работать с файлом **res/layout/activity_main.xml**. На рис. 2.2 можно увидеть редактор, соответствующий визуальному способу формирования интерфейса, этому режиму соответствует вкладка **Design**.

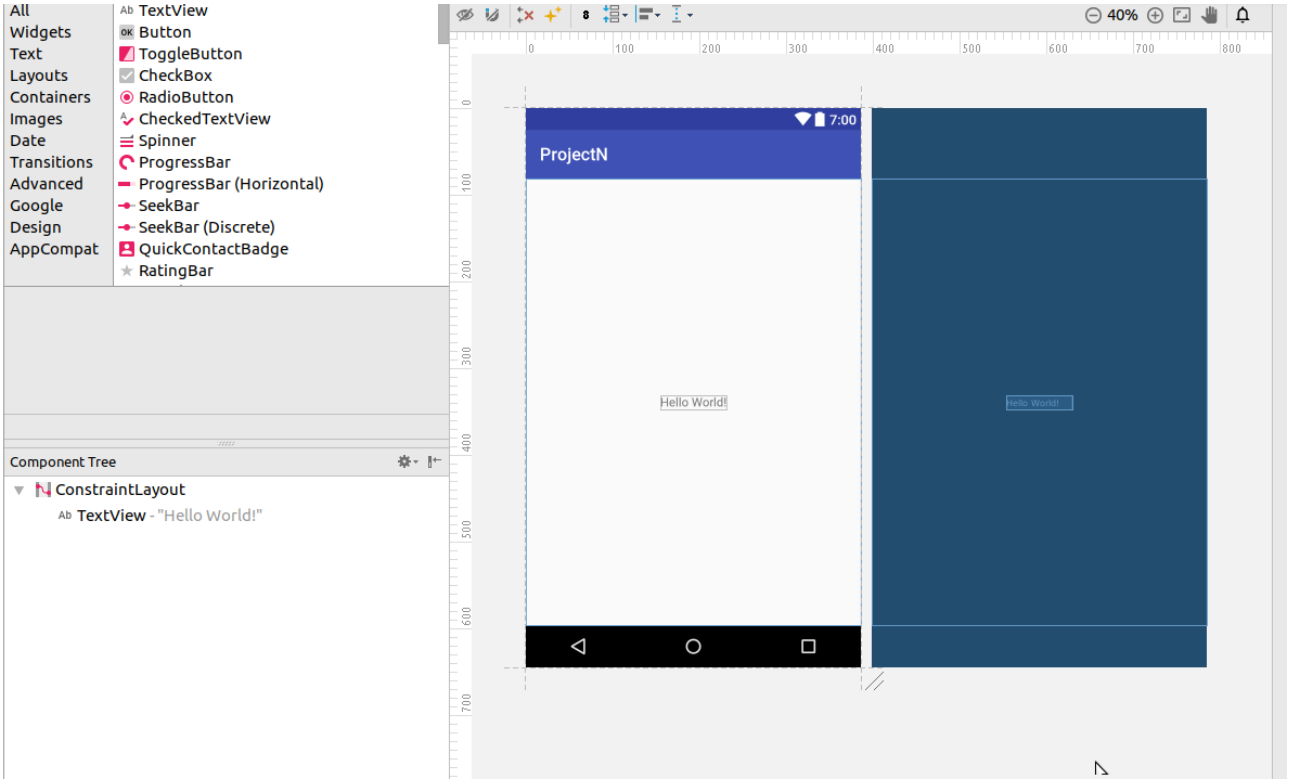


Рис. 2.2. Графическое изображение активности приложения

На рис. 2.2 рядом с вкладкой **Design** расположена вкладка **Text**. Она соответствует режиму редактирования интерфейса путем формирования XML файла. На рис. 2.3 можно увидеть редактор XML файла.

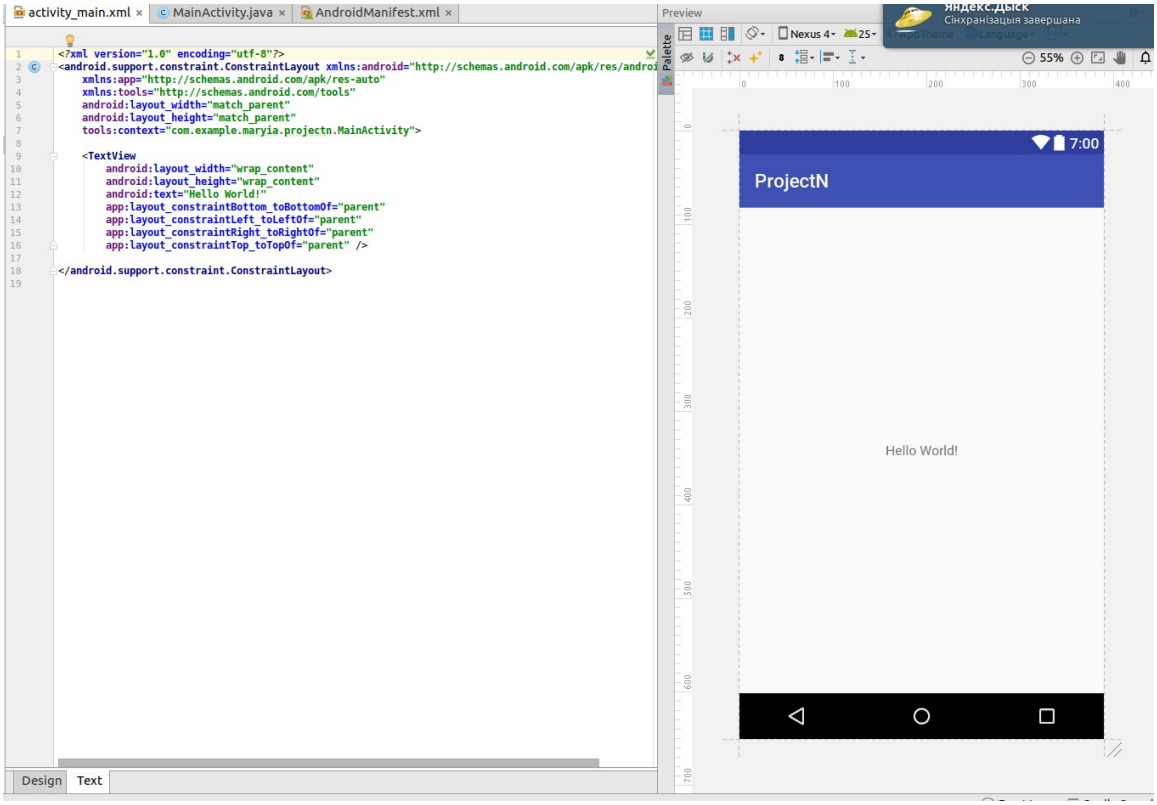


Рис. 2.3. Описание активности в XML формате

Зададим табличное расположение компонентов на форме, для этого выберем вкладку **Layouts**, найдем там **TableLayout** и добавим его на форму. На рис. 2.4 можно увидеть результат этих действий.

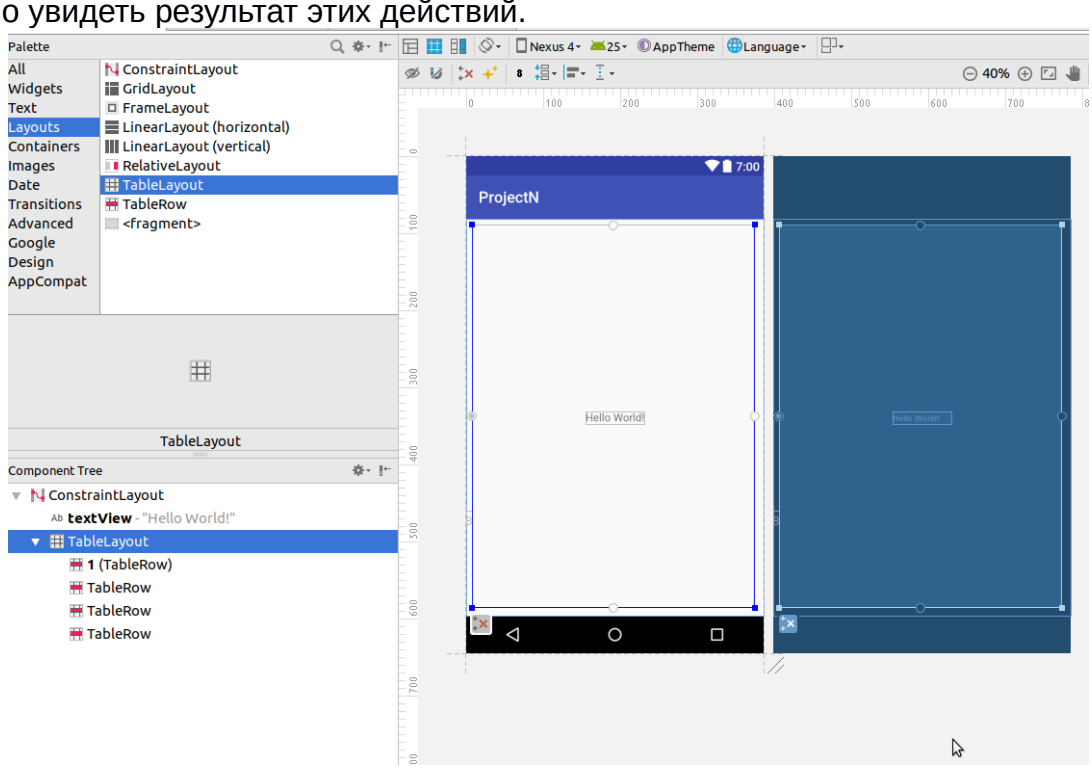


Рис. 2.4. Настройка интерфейса, добавление TableLayout

Теперь начнем добавлять элементы интерфейса, будем использовать графический режим правки.

Во-первых, нам необходимо добавить информационное поле. Для этого на панели **=Palette=** выбираем вкладку **Widgets**, на этой вкладке найдем поле **TextView**, перенесем в окно приложения, разместим в первой строке таблицы (**TableLayout**).

Во-вторых, нам потребуется поле ввода информации, на вкладке **Text Fields** найдем текстовое поле **Number** и разместим во второй строке таблицы.

В-третьих, вернемся на вкладку **Widgets**, выберем там элемент **Button** и добавим в третью строку таблицы. Не нужную четвертую строку таблицы удалим, получим следующий вид приложения, см. рис. 2.5.

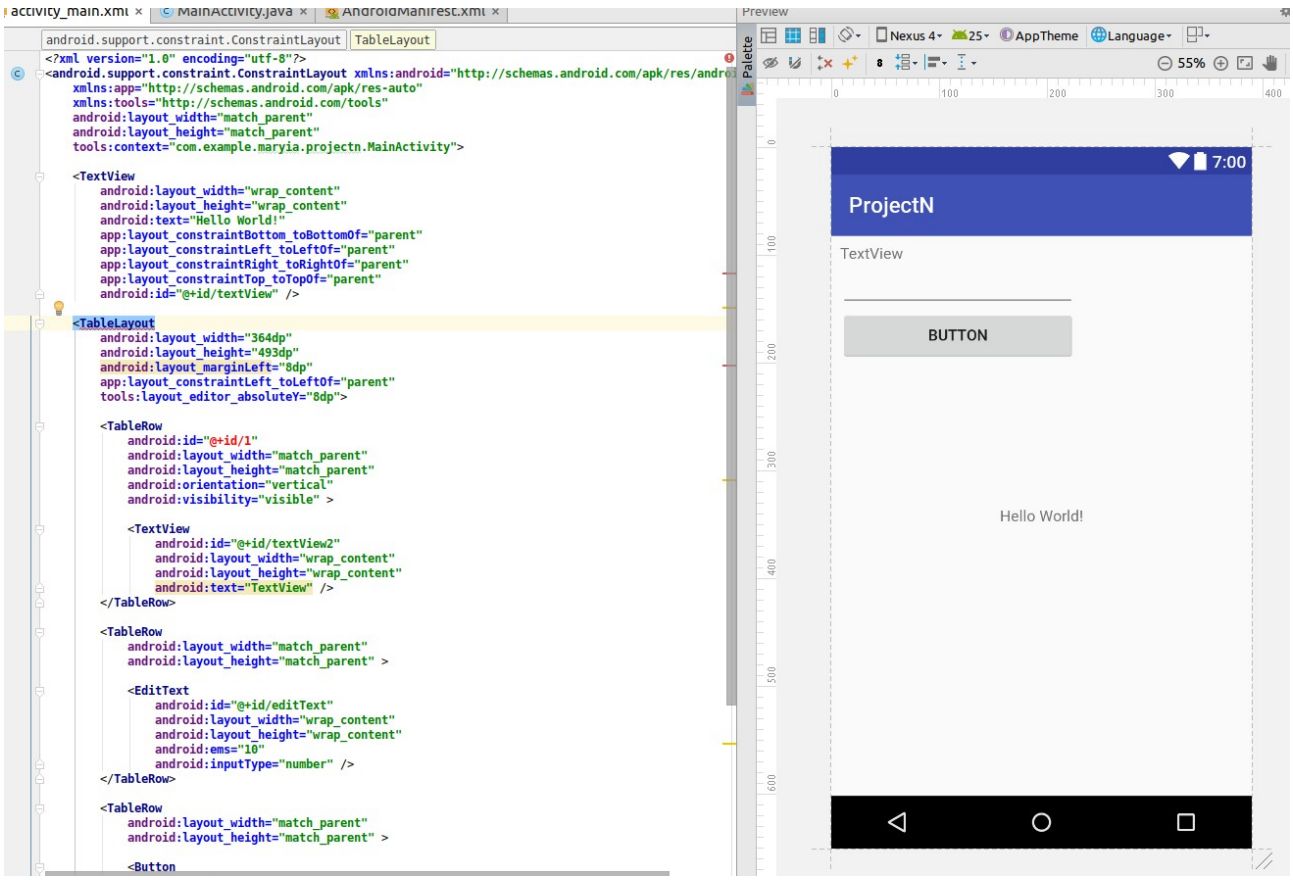


Рис. 2.5. Интерфейс приложения

После настройки интерфейса можно заглянуть в файл **activity_main.xml**, в этом файле прописано, что используется **TableLayout** и дано описание каждой из трех строк. На рис. 2.5 можно увидеть, как выглядит такое описание на примере первой строки таблицы.

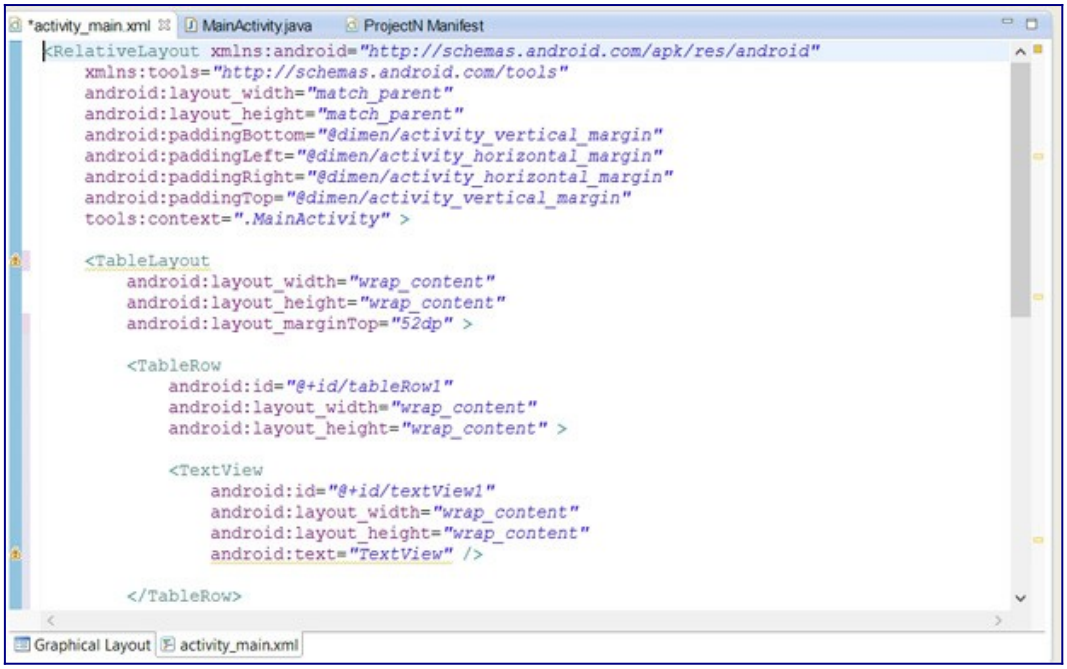


Рис. 2.6. Фрагмент файла activity_main.xml, описание строки в TableLayout

Теперь необходимо наполнить наши элементы интерфейса смыслом, нам понадобится текст для общения с пользователем, при программировании под Android существует практика разделять ресурсы и код приложения. Для хранения любых строк, которые могут понадобиться приложению, используется файл **strings.xml**. Хранение всех строковых ресурсов в этом файле серьезно облегчает локализацию приложения на другие языки. Этот файл можно найти в Package Explorer в папке **res/values**. Откроем его и посмотрим, что там есть, см. рис. 2.7.

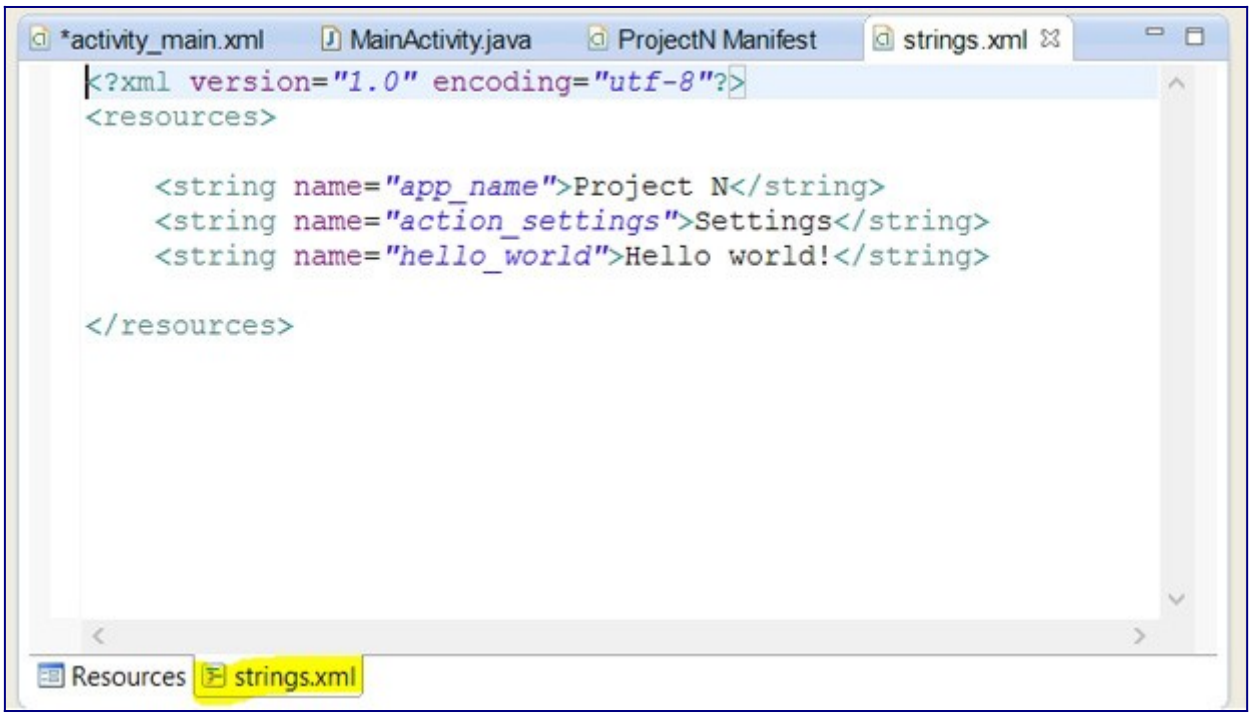


Рис. 2.7. Файл string.xml

Уберем лишние строки и добавим новые, результат можно посмотреть на рис. 2.8.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Угадай число!</string>
    <string name="behind">Недолёт!</string>
    <string name="ahead">Перелёт!</string>
    <string name="hit">В точку!</string>
    <string name="input_value">Ввести значение</string>
    <string name="play_more">Сыграть ещё</string>
    <string name="try_to_guess">Попробуйте угадать число (1 &amp;#8211; 100) </string>
    <string name="error">Неверный ввод!</string>

</resources>
```

Рис. 2.8. Отредактированный файл string.xml

Данные переменные будут выполнять следующие задачи:

- app_name установит "видимое" название приложения;
- behind, ahead, hit оповестят пользователя об его успехах в игре;
- play_more и try_to_guess установит название кнопки, которое объяснит её функции;
- input_value пригласит пользователя к вводу числа;
- error сообщит о неверном вводе.

После изменения **strings.xml**, при переходе на другую вкладку, не забудьте сохранить изменения (самый быстрый способ - нажать **Ctrl+S**).

Настроим текст в информационном поле. Для этого на вкладке **Properties** в правой части окна выберем элемент **textView1** (это и есть наше информационное поле, имеет смысл придумать ему более осмысленное имя). Найдем свойство **Text**, подставим в него значение строки с именем try_to_guess, см. рис. 2.9.

Аналогично можно настроить текст, которым нас будет приветствовать кнопка, только в этом случае надо работать с элементом **button1**.

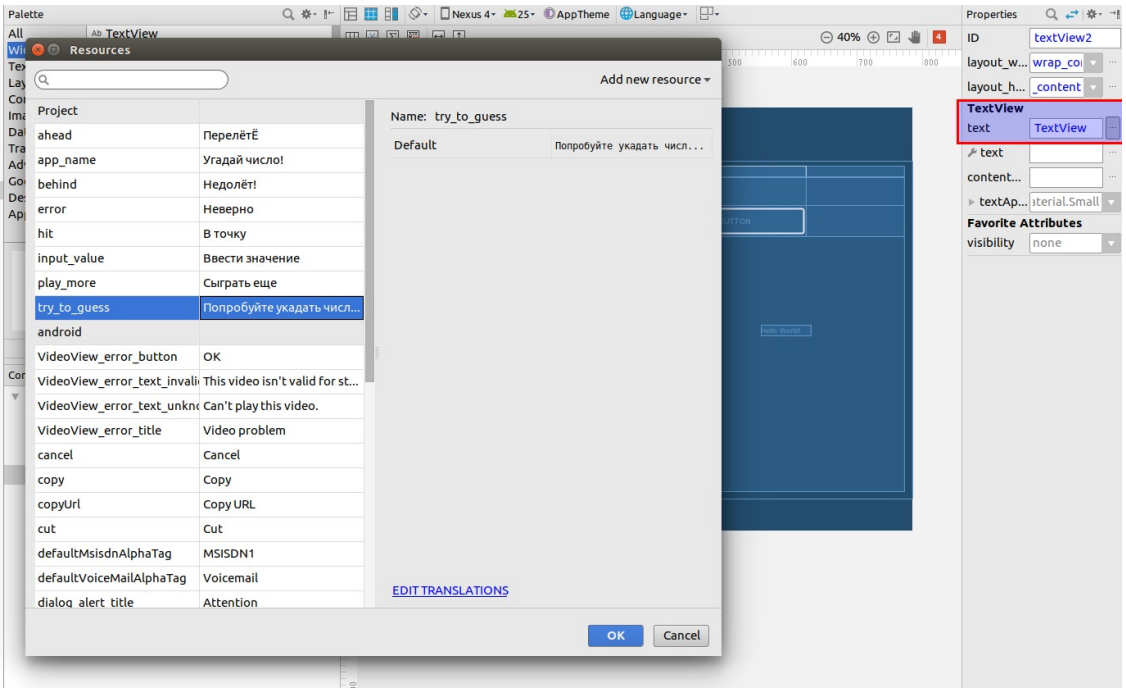


Рис. 2.9. Настройка текста для кнопки button1

Пришло время вспомнить о виртуальном устройстве, если оно работает, уже можно запустить проект и посмотреть, как приложение будет выглядеть на экране устройства, а выглядеть оно может как показано на рис. 2.10.

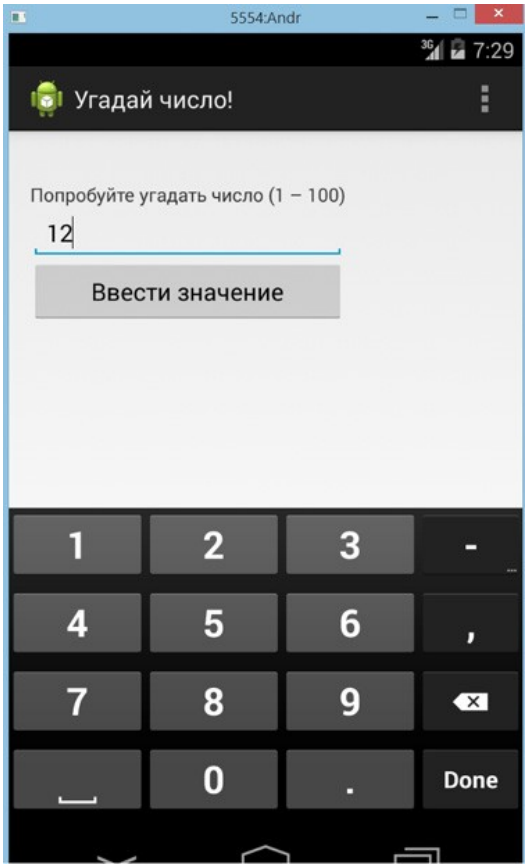


Рис. 2.10. Запуск приложения на виртуальном устройстве

Приложение выглядит довольно просто, но мы на многое и не рассчитывали. Главное, что нас интересует, это наличие всех элементов на экране, верный текст в каждом элементе, где он предусмотрен и возможность вводить числа в поле ввода. На рис. 2.10 видно, что все требования выполнены. Приложение есть, его можно запустить на виртуальном или реальном устройстве, но оно ничего не делает. Следующим шагом будет реализация логики приложения, т. е. обработка события нажатия на кнопку, как было прописано в задании.

2.3 Реализовать логику приложения

Приступим непосредственно к программированию, работать будем с файлом **src/com.example.projectn/MainActivity.java**. Найдем этот файл в Package Explorer см. рис. 2.11, откроем и начнем редактировать.

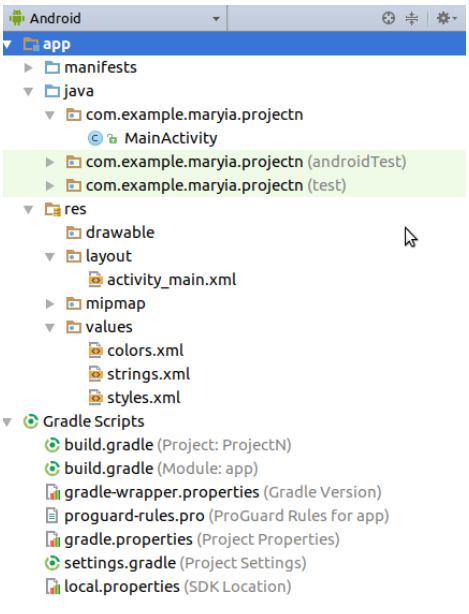


Рис. 2.11. Файл MainActivity.java в Package Explorer

Пока файл выглядит следующим образом, см. рис. 2.12.

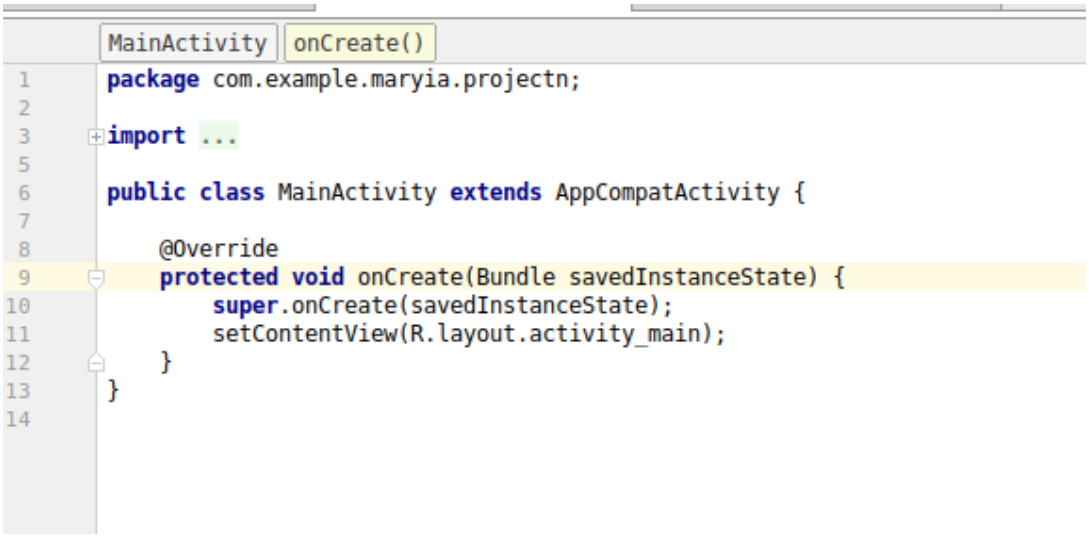


Рис. 2.12. Файл MainActivity.java после создания приложения

Можно заметить, что класс MainActivity является наследником класса Activity и в нем уже реализован метод onCreate(), который запускается при первоначальном создании активности, нам потребуется его дополнить.

Мы предполагаем программно менять информацию в поле **TextView**, получать значение из поля **EditText** и обрабатывать события нажатия на кнопку **Button**, поэтому необходимо объявить соответствующие переменные, как поля класса MainActivity:

```
TextView tvInfo;
EditText etInput;
Button bControl;
```

Чтобы не было ошибок, необходимо импортировать пакет android.widget, который содержит все элементы графического интерфейса:

```
import android.widget.*;
```

На самом деле среда разработки подскажет, что делать.

Теперь необходимо связать эти переменные с элементами интерфейса, уже добавленными нами в **activity_main.xml**, сделать это необходимо в методе onCreate(), а для получения уже созданного элемента интерфейса воспользуемся методом findViewById(). Итак в метод onCreate() добавим следующие строки:

```
tvInfo = (TextView)findViewById(R.id.textView1);
etInput = (EditText)findViewById(R.id.editText1);
bControl = (Button)findViewById(R.id.button1);
```

Проверьте соответствие имен элементов интерфейса и скорректируйте добавляемый код, если требуется.

Метод findViewById() возвращает объект класса View, который является общим предком для всех компонентов пользовательского интерфейса, для того чтобы избежать возможных ошибок в скобках перед вызовом метода указываем до какого конкретно компонента необходимо сузить возможности объекта View.

Пришло время выполнить обработку нажатия на кнопку. Вернемся к файлу **activity_main.xml** в графический режим редактирования, выберем элемент **Button** и на вкладке со свойствами элемента найдем свойство **On Click** и запишем в него **onClick** - имя метода, который будет обрабатывать нажатие на кнопку. Как это выглядит показывает рис. 2.13.

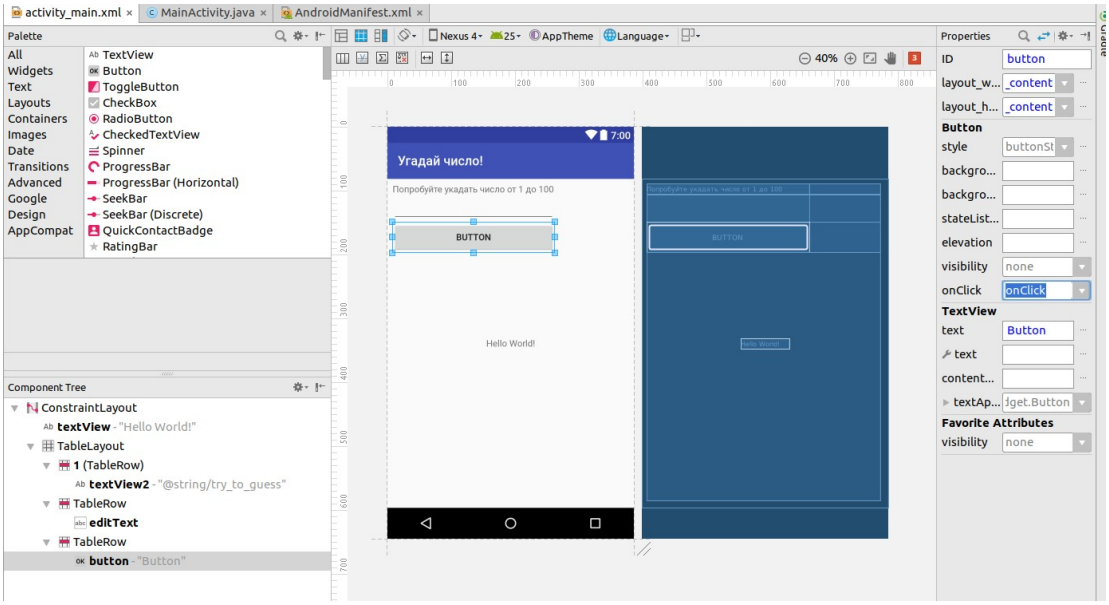


Рис. 2.13. Настройка свойства On Click для кнопки

Эти же действия можно выполнить в фале **activity_main.xml**, достаточно дописать выделенную на рис. 2.14 строку:

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="@string/input_value" />
```

Рис. 2.14. Настройка свойствами On Click для кнопки в файле XML

Для настройки свойств элементов интерфейса достаточно использовать любой способ: графический или редактирование XML файла.

Вернемся в файл **MainActivity.java**, в класс активности необходимо добавить метод:

```
public void onClick(View v){...}
```

Имя метода не обязательно должно быть onClick(), главное, чтобы оно совпадало с именем, указанным в свойстве **On Click**. В этом методе и будет происходить все наше программирование в этой лабораторной работе.

Нам потребуются две переменные:

- целочисленная для хранения загаданного числа (случайное число от 1 до 200);
- логическая для хранения состояния закончена игра или нет.

Обе эти переменные имеет смысл объявить как поля класса активности, первоначальные значения присвоить в методе onCreate.

Получить целочисленное значение из поля ввода, можно с помощью следующей конструкции:

```
Integer.parseInt(etInput.getText().toString())
```

изменить значение текста в информационном поле можно с помощью следующей конструкции:

```
tvInfo.setText(getResources().getString(R.string.ahead));
```

в данном случае в информационном поле появится значение строкового ресурса с именем ahead.

Осталось реализовать логику приложения в методе onClick().

Написать код этого метода самостоятельно, для контроля в приложении предложен листинг, который содержит один из вариантов кода описанного приложения.

3. Задания для самостоятельной работы:

Задание 3.1. Внести изменения в приложение «Угадай число»

Внести изменения в приложение, чтобы решались следующие проблемы:

- Что произойдёт, если кнопка будет нажата до ввода какого-либо числа? Скорей всего приложение будет остановлено, необходимо как-то отслеживать этот вариант развития событий и адекватно реагировать.
- Что произойдёт, если пользователь введёт число меньшее нуля или большее

200? Скорей всего приложение обработает этот ввод, но было бы лучше, если бы появилось сообщение о том, что введённое число не удовлетворяет условиям задачи.

- Что произойдет, если введено не число?
- Как завершить приложение? И надо ли это делать?
- Изменить цвет фона, текста, кнопки.
- Добавить локализацию приложения для английского и русского языков.
- Как еще можно улучшить приложение? Реализуйте эти улучшения.

Задание 3.2 Реализовать игру «Угадай число» на языке Kotlin

Изучить синтаксис языка Kotlin в главах 1-4 книги Скин Джош, Гринхол Дэвид. Kotlin. Программирование для профессионалов и реализовать приложение «Угадай число» на основе примера «A Simple Number Guessing Game» из книги Spath Peter. Learn Kotlin for Android Development (стр. 31-41) и с учетом требований, изложенных в задании 3.1.

Задание 3.3 Реализовать калькулятор по вариантам

Общие требования к вариантам

Реализовать калькулятор на языке Java и/или Kotlin с учетом критериев оценивания.

- Предусмотреть возможность ввода дробных чисел через точку (например, 0.5).
- Предусмотреть возможность ввода отрицательных чисел.
- Создать графическое представление приложения Calculator для **горизонтальной ориентации** экрана (пример приведен на рисунке 3.1, б).
- Программно реализовать обработку нажатий на кнопки с использованием Activity в качестве единого обработчика.
- В случае деления на 0 выводить вместо результата сообщение о недопустимости операции.
- Разработать перечень проверок и протестировать приложение Calculator.
- Опубликовать код приложения в репозиторий.
- Продемонстрировать работу приложения Calculator на эмуляторе или реальном устройстве.

Вариант 1.

Разработать приложение Calculator1 с одним Activity. Реализовать с кнопками цифр, математических операций (сложения, вычитания, умножения, деления, извлечение квадратного и кубического корня, возведение в степень и др.), получения результата (пример приведен на рисунке 3.1). Локализовать приложение для английского и

русского языка.

а *б*

а – вертикальная ориентация экрана; *б* – горизонтальная ориентация экрана

Рисунок 3.1. – Пример работы приложения Калькулятор 1

Вариант 2.

Разработать приложение Calculator2 с одним Activity, которая позволяет конвертировать температуру и объем. Реализовать с кнопками цифр, математических операций и блоком вывода результата (см. рис. 3.2). Локализовать приложение для английского и белорусского языка.

Вариант 3.

Разработать приложение Calculator3 с одним Activity, которая позволяет конвертировать длину и скорость. Реализовать с кнопками цифр, математических операций и блоком вывода результата. Локализовать приложение для английского и польского языка.

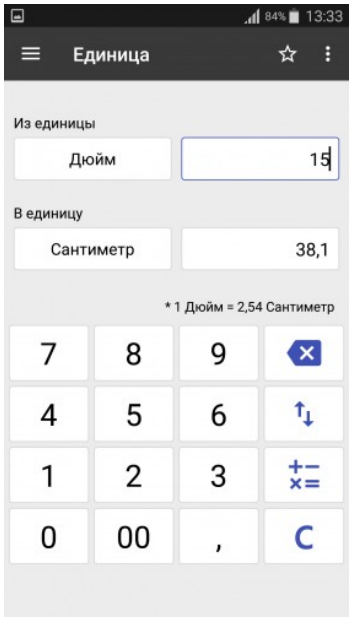


Рисунок 3.2. – Пример работы приложения Калькулятор 3

Вариант 4.

Разработать приложение Calculator 4 с одним Activity, которая позволяет конвертировать валюты (не менее 10 валют). Реализовать с кнопками цифр, математических операций и блоком вывода результата. Локализовать приложение для английского и украинского языка.

Вариант 5.

Разработать приложение Calculator5 с одним Activity, которая позволяет вычисление функции y=ax+c, y=ax2+bx+c, y=ax3+bx2+cx+d, Реализовать с кнопками цифр, математических операций и блоком вывода результата. Локализовать приложение для английского и белорусского языка.

Вариант 6.

Разработать приложение Calculator6 с одним Activity, которая позволяет конвертировать: радианы в градусы и вес (не менее 10 единиц измерения). Локализовать приложение для английского и русского языка.

Вариант 7.

Разработать приложение Calculator7 с одним Activity, которая позволяет вычислять побитовые операции. Локализовать приложение для английского и польского языка

Вариант 8.

Разработать приложение Calculator8 с одним Activity, которая позволяет преобразование из одной системы счисления в другую (2c\c, 8 c\c , 16c\c). Локализовать приложение для английского и украинского языка.

Вариант 9.

Разработать приложение Calculator9 с одним Activity, которая позволяет вычисление тригонометрических функций (арккосинус, арксинус, арктангенс, арккотангенс). Локализовать приложение для английского и русского языка.

Вариант 10.

Разработать приложение Calculator10 с одним Activity, которая позволяет вычисление гиперболических функций. Локализовать приложение для английского и польского языка.

Вариант 11.

Разработать приложение Calculator11 с одним Activity, которая позволяет отображающее размер в количестве байт и бит для обычных типов и какой тип подойдет для введенного числа. Локализовать приложение для английского и белорусского языка.

Вариант 12.

Разработать приложение Calculator12 с одним Activity, которая позволяет генерацию

случайного числа из заданного диапазона (выбор целое, дробное, четное, нечётное) и выполнять простые математические операции над сгенерированными числами. Локализовать приложение для английского и русского языка.

Вариант 13.

Разработать приложение Calculator13 с одним Activity, которая позволяет вычислить степень, корень, факториал числа. Локализовать приложение для английского и польского языка.

Вариант 14.

Разработать приложение Calculator14 с одним Activity, которая позволяет вычислить площадь сектора круга (S), через длину дуги (L), через угол (a) и длину хорды. Локализовать приложение для английского и украинского языка.

Вариант 15.

Разработать приложение Calculator15 с одним Activity, которая позволяет конвертировать объем в вес (вода, соль, масло). Можно дополнительно использовать плотность. Локализовать приложение для английского и русского языка.

Вариант 16.

Разработать приложение Calculator16 с одним Activity, которая позволяет конвертировать площадь из метрической системы в британские и американские единицы измерения площади. Примеры конвертации - https://www.convert-me.com/ru/convert/area/. Локализовать приложение для английского и русского языка.

Вариант 17.

Разработать приложение Calculator17 с одним Activity, которая позволяет конвертировать время (от секунд до столетий). Локализовать приложение для английского и украинского языка.

Вариант 18.

Разработать приложение Calculator18 с одним Activity, которая позволяет конвертировать площадь из метрической системы в единицы измерения площади Великого Княжества Литовского (https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BC%D0%B5%D1%80_%D0%B2_%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D0%BC_%D0%BA%D0%BD%D1%8F%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5_%D0%9B%D0%B8%D1%82%D0%BE%D0%B2%D1%81%D0%BA%D0%BE%D0%BC). Локализовать приложение для английского и белорусского языка.

Вариант 19.

Разработать приложение Calculator19 с одним Activity, которая позволяет конвертировать площадь из метрической системы в исторические и страновые

единицы площади (не мене 5 стран). Примеры конвертации - https://www.convert-me.com/ru/convert/area/. Локализовать приложение для английского и украинского языка.

Вариант 20.

Разработать приложение Calculator20 с одним Activity, которая позволяет конвертировать длину и массу (вес) из метрической системы в единицы измерения длины и мвссы (веса) Великого Княжества Литовского (https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BC%D0%B5%D1%80_%D0%B2_%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D0%BC_%D0%BA%D0%BD%D1%8F%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5_%D0%9B%D0%B8%D1%82%D0%BE%D0%B2%D1%81%D0%BA%D0%BE%D0%BC). Локализовать приложение для английского и белорусского языка.

Вариант 21.

Разработать приложение Calculator21 с одним Activity, которая позволяет определить мин, макс, среднее, сумму, наибольший общий делитель, наименьший общий делитель для 4 вводимых чисел. Локализовать приложение для английского и польского языка.

Вариант 22.

Разработать приложение Calculator21 с одним Activity, которая позволяет определить площадь треугольника по формуле Герона, через две стороны и угол, через сторону и два угла, а также площадь равностороннего треугольника через его сторону и высоту и только через его сторону. Локализовать приложение для английского и польского языка.

Вариант 23.

Разработать приложение Calculator6 с одним Activity, которая позволяет конвертировать: радианы в градусы и длину (не менее 10 единиц измерения). Локализовать приложение для немецкого и русского языка.

Вариант 24.

Разработать приложение Calculator21 с одним Activity, которая позволяет выполнить перевод мер длины и площади в системе СИ в старорусскую или английскую систему мер. Система мер СИ используется по умолчанию, старорусскую или английскую систему мер выбирать. Локализовать приложение для английского и русского языка.

Вариант 25.

Разработать приложение Calculator21 с одним Activity, которая позволяет выполнить перевод в прямом и обратном порядке мер веса, объёма в системе СИ в старорусскую систему мер, включающую меры веса и объёма жидких и сыпучих тел. Система мер СИ используется по умолчанию, старорусскую или английскую систему мер выбирать. Локализовать приложение для английского и русского языка.

Содержание отчета

- Скриншоты графических представлений приложения «Угадай число» в вертикальной ориентации, а так же скриншоты Calculator в вертикальной и горизонтальной ориентациях экрана в Android Studio.
- Код xml-файла графического представления приложения Calculator в вертикальной ориентации экрана.
- Код xml-файла графического представления приложения Calculator в горизонтальной ориентации экрана.
- Код java-файла Activity приложения Calculator.

4. Заключение

В лабораторной работе рассмотрен процесс разработки простого приложения переднего плана. Описано создание активности, настройка интерфейса и реализация логики приложения. Других компонентов в приложении не предусмотрено. В последующих работах будут рассматриваться приложения, содержащие несколько активностей. Смешанные приложения, работающие на переднем плане и при этом поддерживающие сервисы, работающие в фоновом режиме.

Контрольные вопросы

1. Какие основные компоненты Android-приложения Вы знаете?
2. Что такое графическое представление Activity?
3. Что такое Layout? Какие существуют виды Layout?
4. Какие параметры (атрибуты) имеют View-элементы?
5. Как создать Layout-файл для работы в горизонтальной ориентации экрана мобильного устройства? В каких случаях это необходимо?
6. Для чего нужны методы setContentView, findViewById?
7. Какие существуют способы обработки событий в Activity?
8. Какой файл содержит ссылки на все ресурсы, используемые в приложении?
9. Приведите правило именования ресурсов в проекте android.
10. Приведите примеры записи полупрозрачного цвета в полной и краткой форме.
11. Какие функции выполняет класс TableLayout? Какие дочерние элементы класса? Привести примеры позиционирования и фрагменты кода.
12. Привести примеры организации файлов локализации приложения для трех языков в структуре проекта (Android View).

13. Привести примеры организации файловой структуры для локализации приложения в случае трех языков (Project View).
14. Перечислить все возможности для индивидуальной настройки приложения в Android.

Приложение 1.

```
package com.example.projectn;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.*;

public class MainActivity extends Activity {
  TextView tvInfo;
  EditText etInput;
  Button bControl;

  int guess;
  boolean gameFinished;

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

    tvInfo = (TextView)findViewById(R.id.textView1);
    etInput = (EditText)findViewById(R.id.editText1);
    bControl = (Button)findViewById(R.id.button1);

    guess = (int)(Math.random()*100);
    gameFinished = false;
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
  }
  public void onClick(View v){
    if (!gameFinished){
      int inp=Integer.parseInt(etInput.getText().toString());
      if (inp > guess)
        tvInfo.setText(getResources().getString(R.string.ahead));
      if (inp < guess)
        tvInfo.setText(getResources().getString(R.string.behind));
      if (inp == guess)
      {
        tvInfo.setText(getResources().getString(R.string.hit));
        bControl.setText(getResources().getString(R.string.play_more));
        gameFinished = true;
      }
    }
    else
    {
      guess = (int)(Math.random()*100);
      bControl.setText(getResources().getString(R.string.input_value));
      tvInfo.setText(getResources().getString(R.string.try_to_guess));
      gameFinished = false;
    }
    etInput.setText("");
  }
}
```