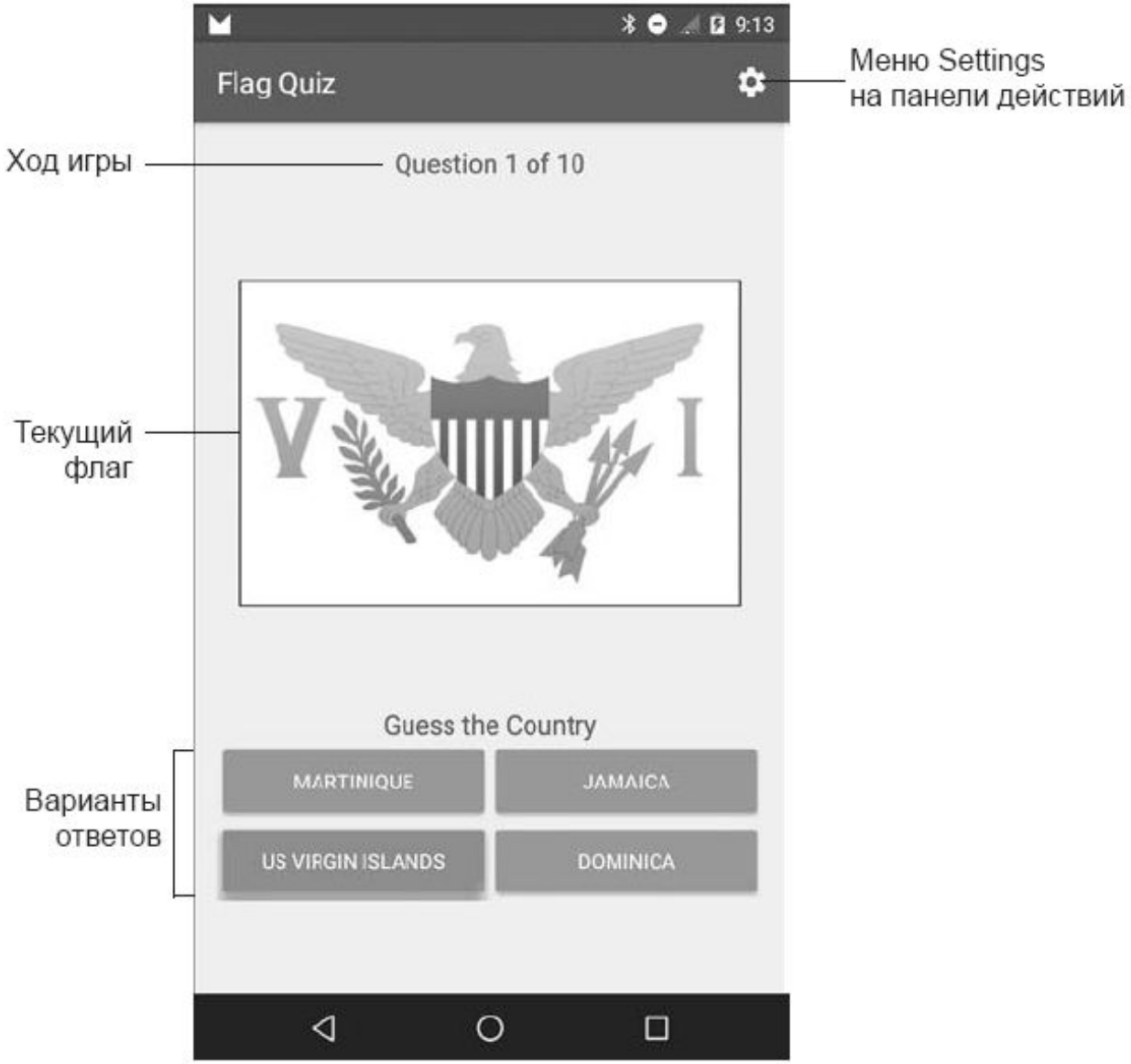
Android 6
Расширенная интерактивность

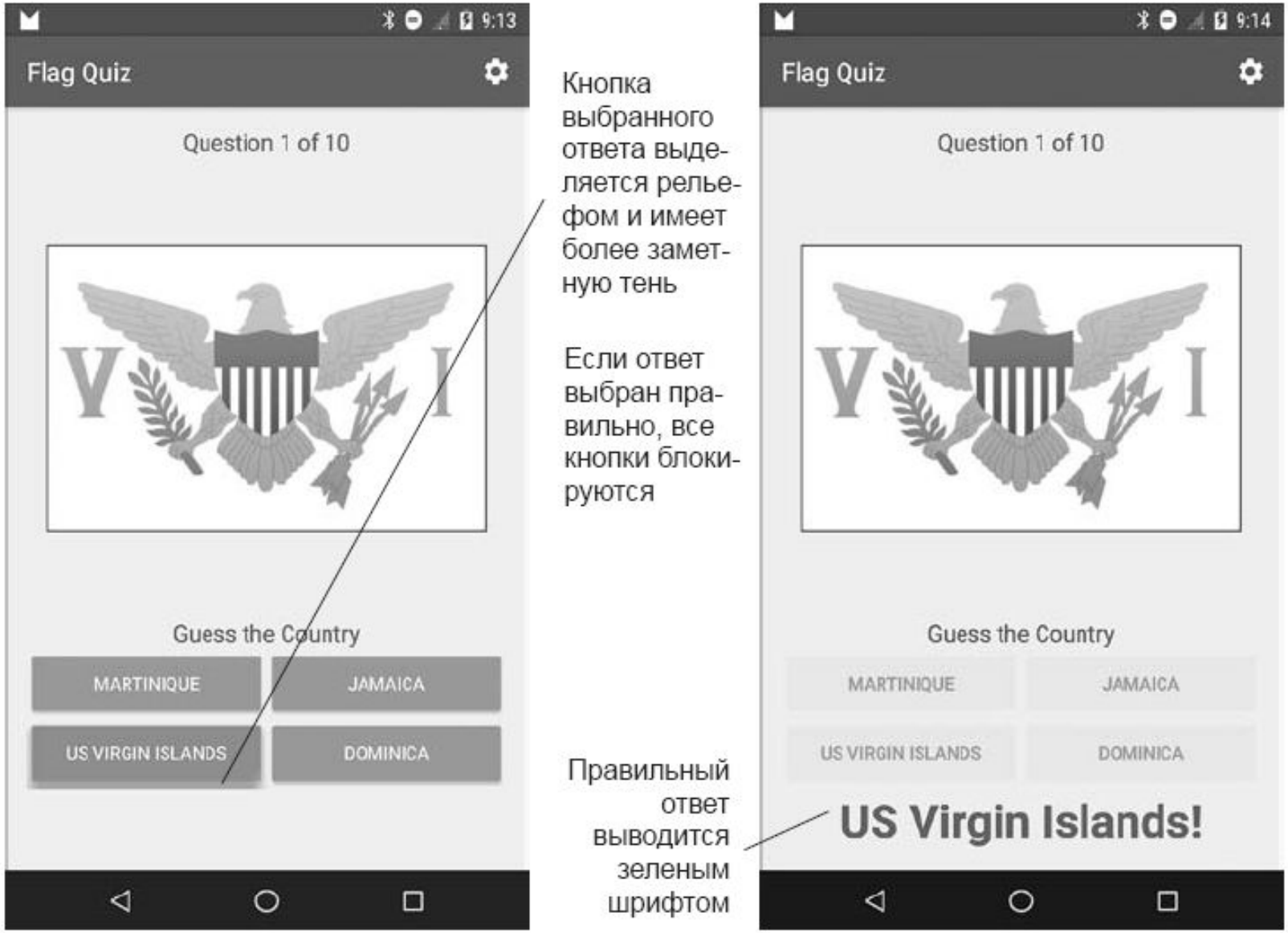
Рассматриваемые вопросы

- Использование фрагментов для эффективной организации пространства в графическом интерфейсе Activity на телефонах и планшетах
- Отображение меню на панели действий для настройки параметров приложения
- Использование объектов PreferenceFragment для автоматического управления настройками пользователя
- Использование SharedPreferences.Editor для изменения пар «ключ—значение», связанных с приложением
- Использование папок assets для организации графических ресурсов и работы с ними средствами AssetManager
- Определение анимаций и их применение к View
- Планирование выполняемых в будущем действий с помощью Handler
- Использование объектов Toast для кратковременного отображения сообщений
- Запуск конкретной активности с помощью явного интента
- Коллекции из пакета java.util
- Определение макетов для разных ориентаций устройства

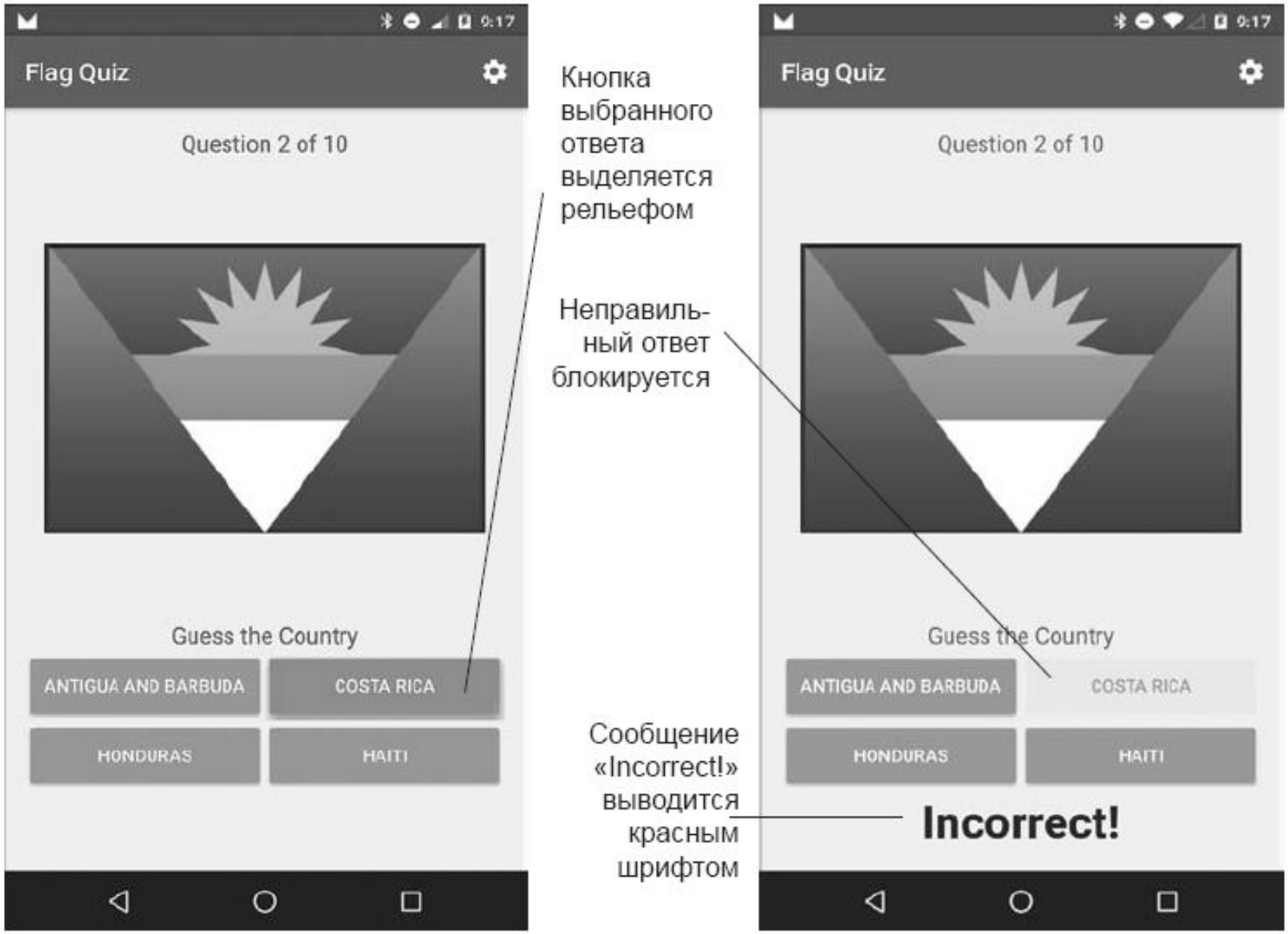
Целевое приложение (портретная ориентация)



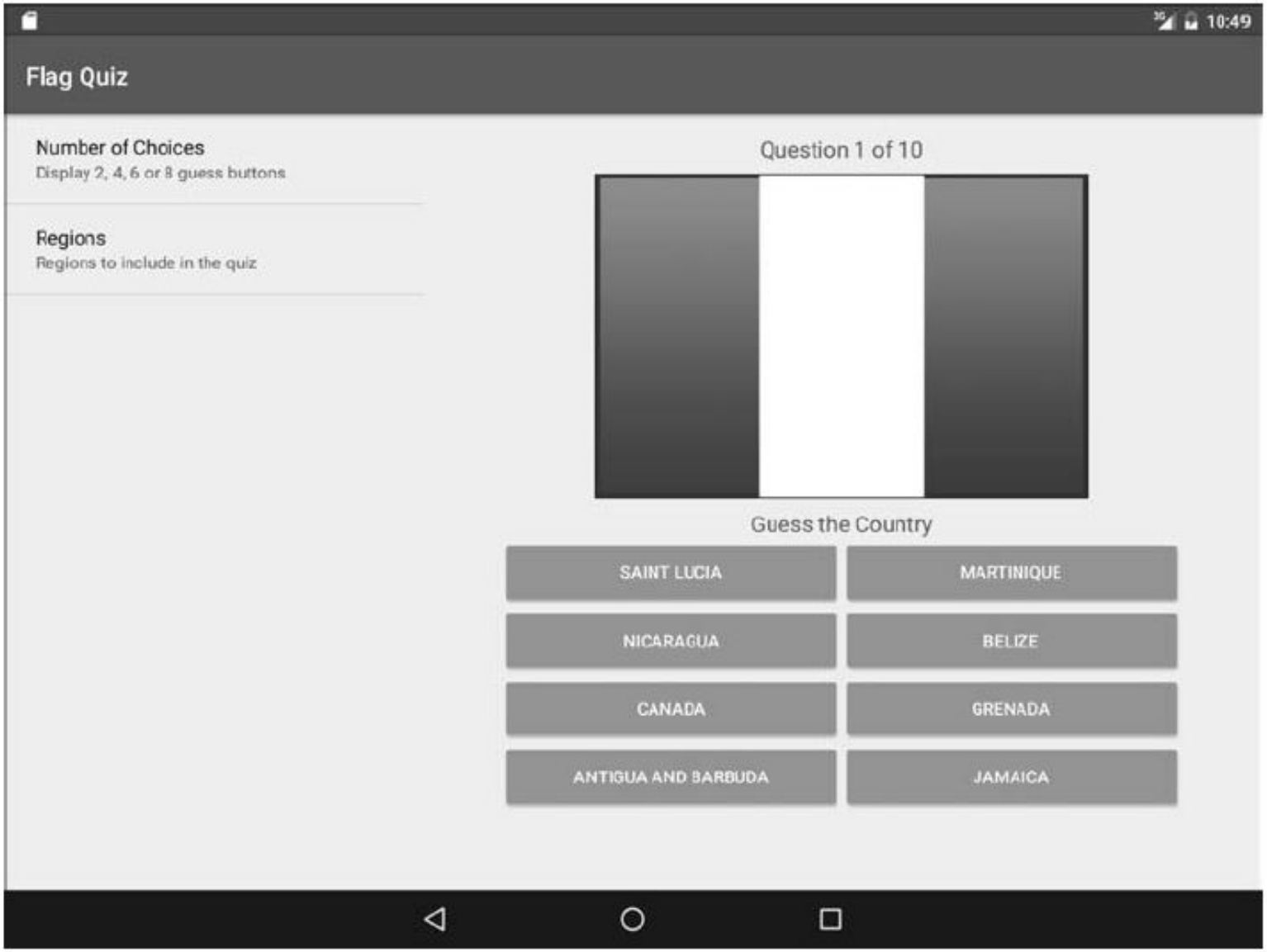
Целевое приложение (портретная ориентация)



Целевое приложение (портретная ориентация)



Целевое приложение (альбомная ориентация)



Используемые возможности

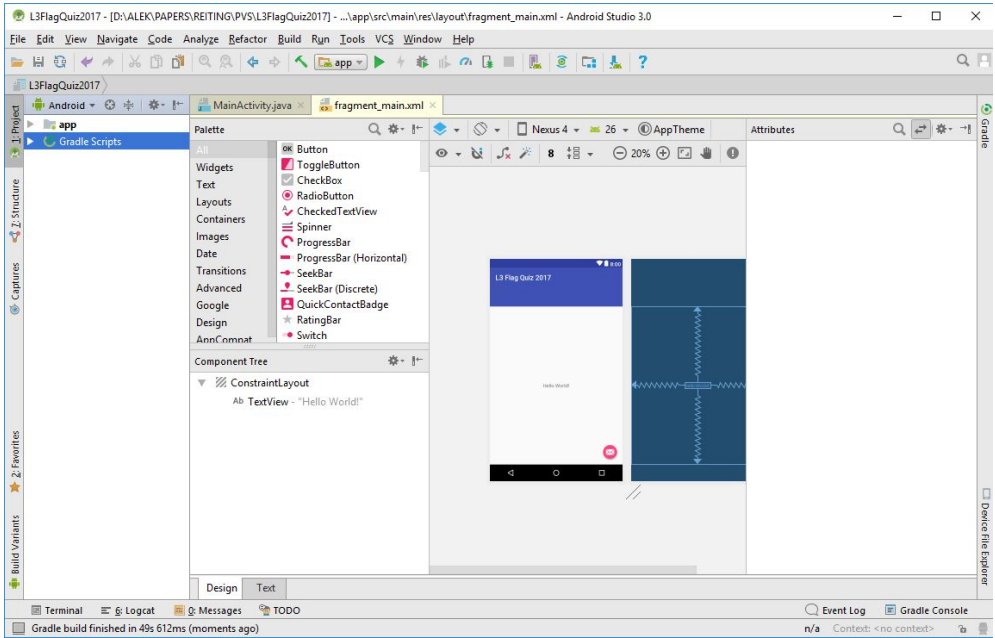
- класс **Menu** (с обработчиками)
- субклассы **Fragment** (фрагменты – повторно используемые части интерфейса или программной логики; фрагменты находятся под управлением активностей)
- методы жизненного цикла фрагментов (**onCreate**, **onCreateView**)
- объект **FragmentManager** (для управления фрагментами со стороны активностей)
- объекты **Preference** (для управления настройками приложения)
- папка **assets**, диспетчер **AssetManager** (для работы с изображениями флагов)
- папки ресурсов **menu**, **anim**, **color**, **xml**

Используемые возможности

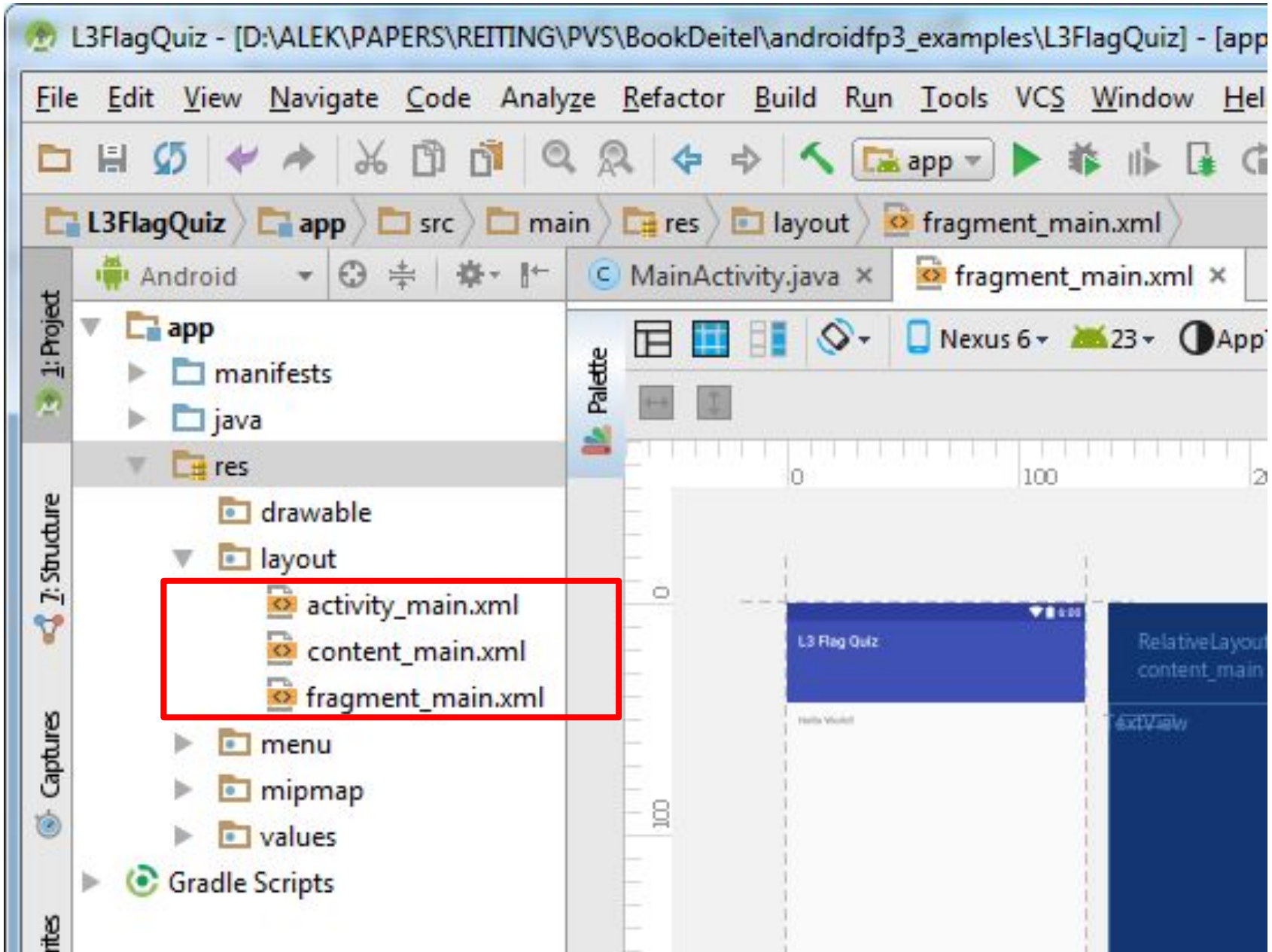
- класс **Configuration** (для определения размера экрана)
- класс **Toast** (для отображения временных второстепенных сообщений)
- объект **Handler** (для управления программными потоками)
- класс **Animation** (для анимации флагов при неверном ответе)
- окна **AlertDialog** (для вывода результатов в конце викторины)
- класс **Log** (для регистрации сообщений об исключениях)
- интенты (для запуска активностей в приложении и передачи данных между ними)
- особенности Java SE7 (выведение типов, конструкция try с ресурсами)

Создание проекта

- Имя проекта: L3 Flag Quiz
- Android 6, API 23
- Шаблон: **Basic Activity**
- флажок Use a Fragment
- Добавить значок в проект



Макеты приложения



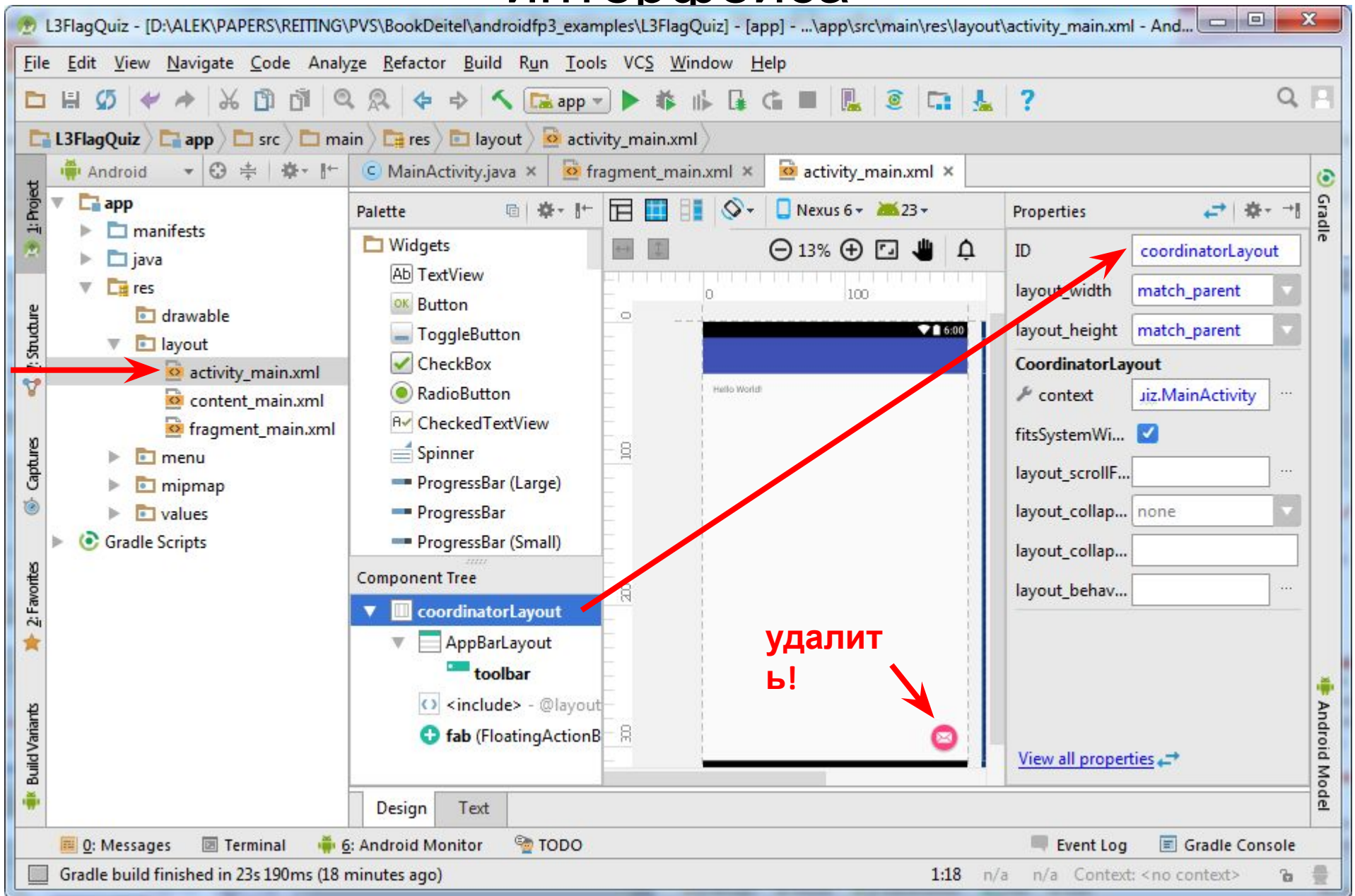
Макеты приложения

- **activity_main.xml**
  содержит компонент CoordinatorLayout с панелью приложения Toolbar

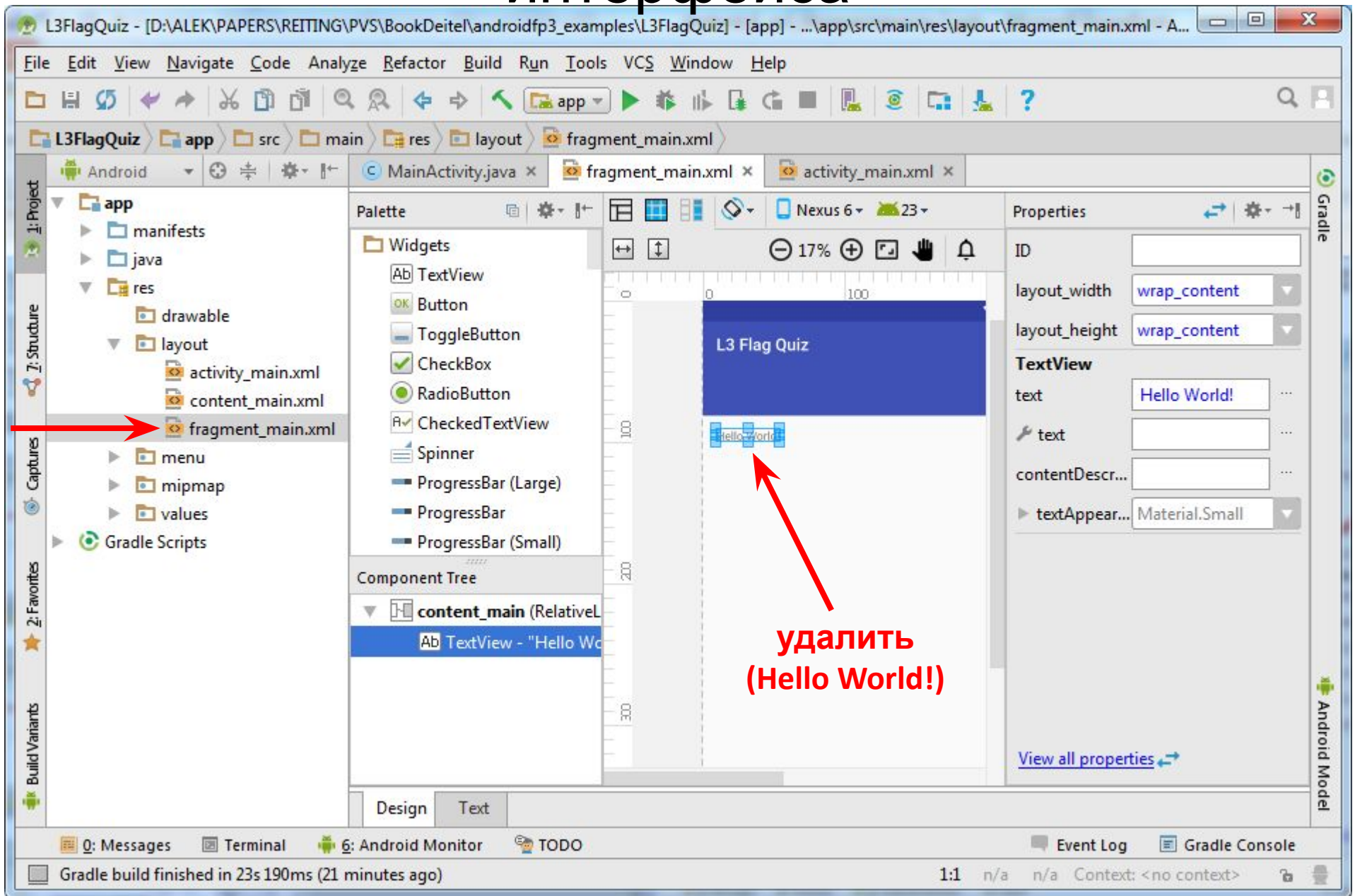
- **content_main.xml**
  определяет часть графического интерфейса MainActivity, которая отображается между панелью приложения (наверху) и системной панелью (внизу)

- **fragment_main.xml**
  при использовании фрагмента основной графический интерфейс определяется именно

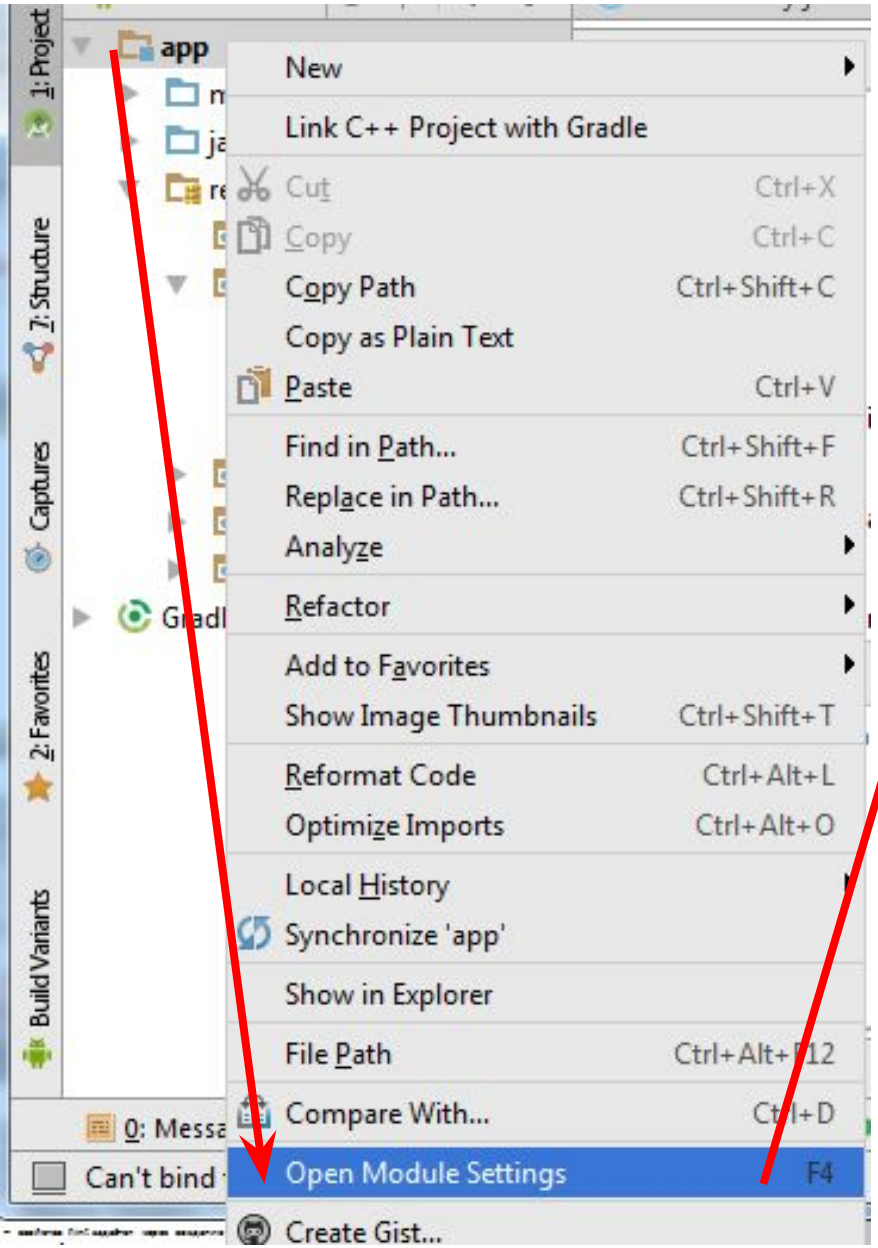
Подготовка к построению графического интерфейса

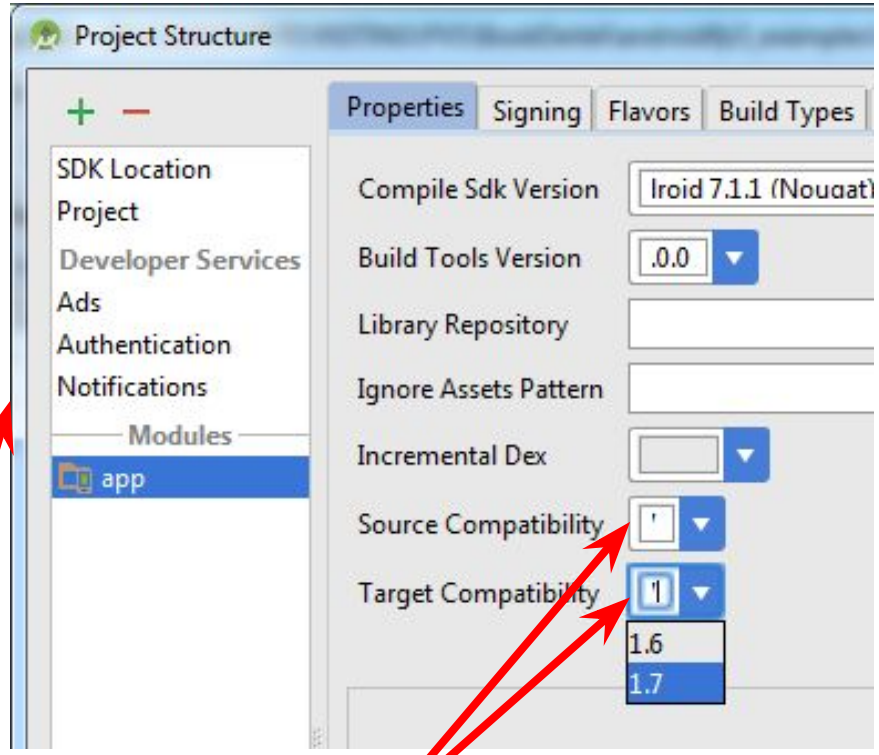


Подготовка к построению графического интерфейса



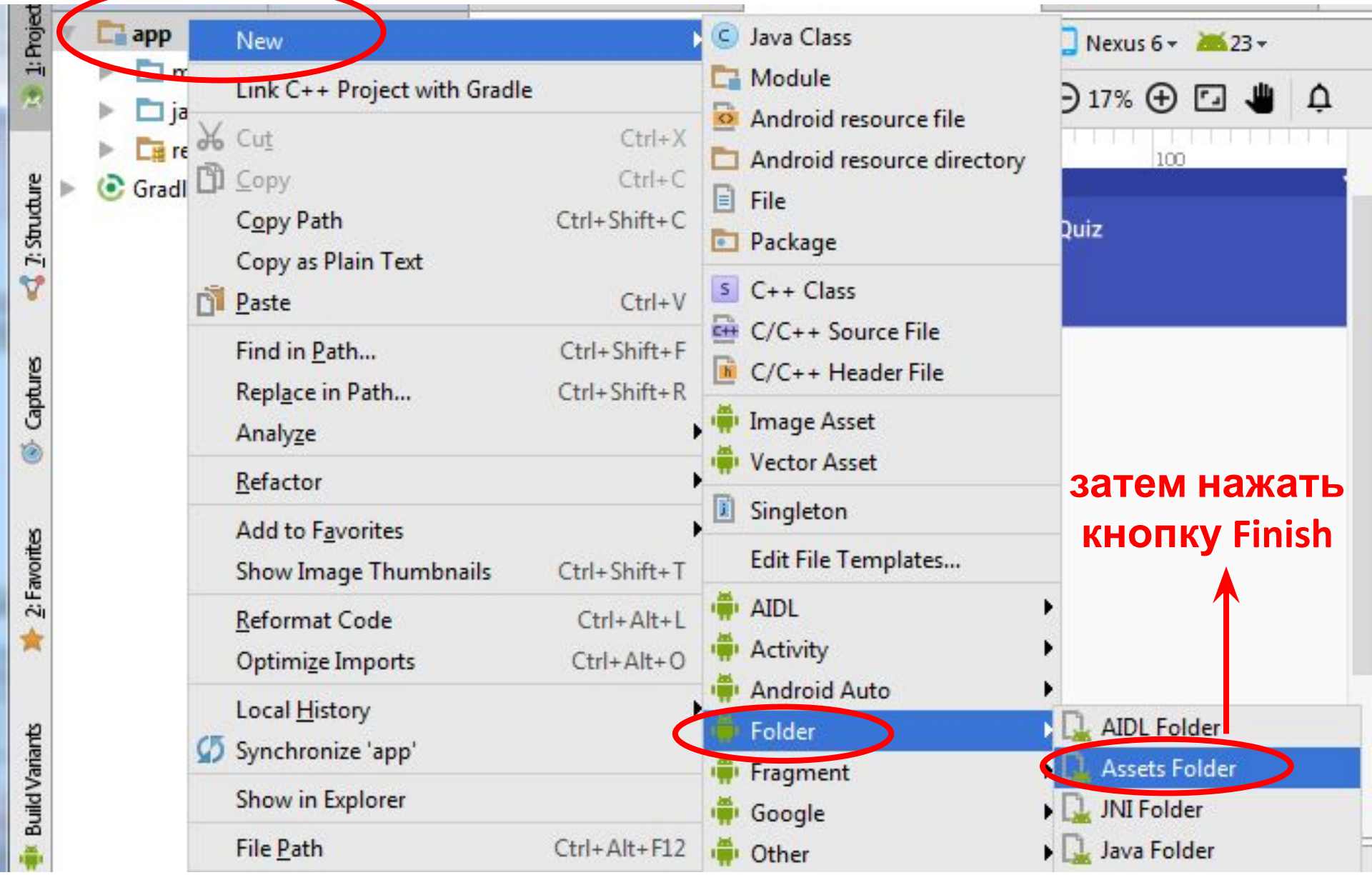
Настройка поддержки Java SE 7



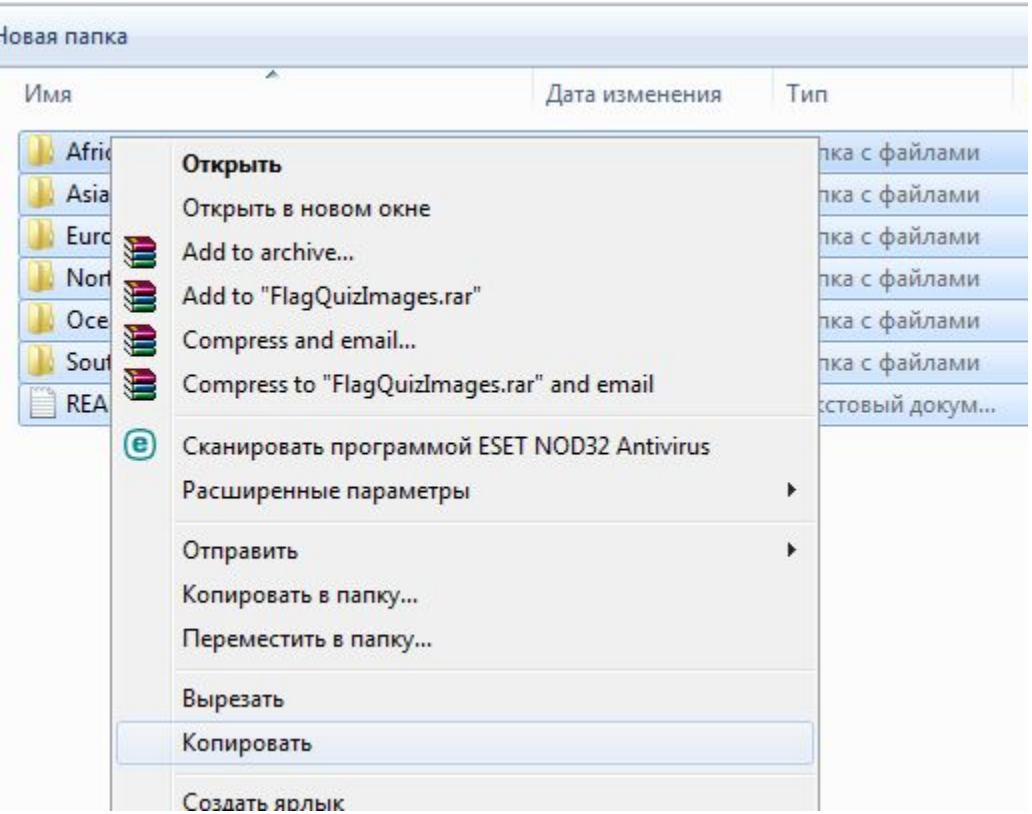


установить значение 1.7

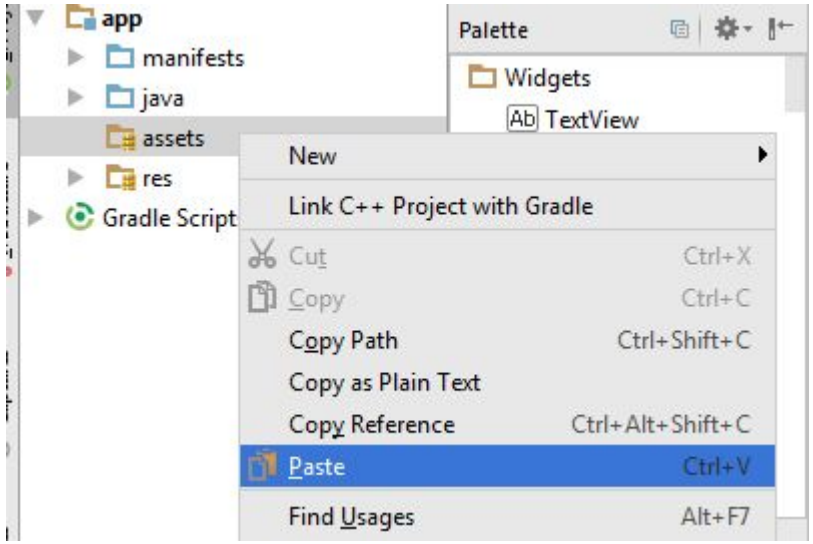
Добавление изображений флагов



Добавление изображений флагов

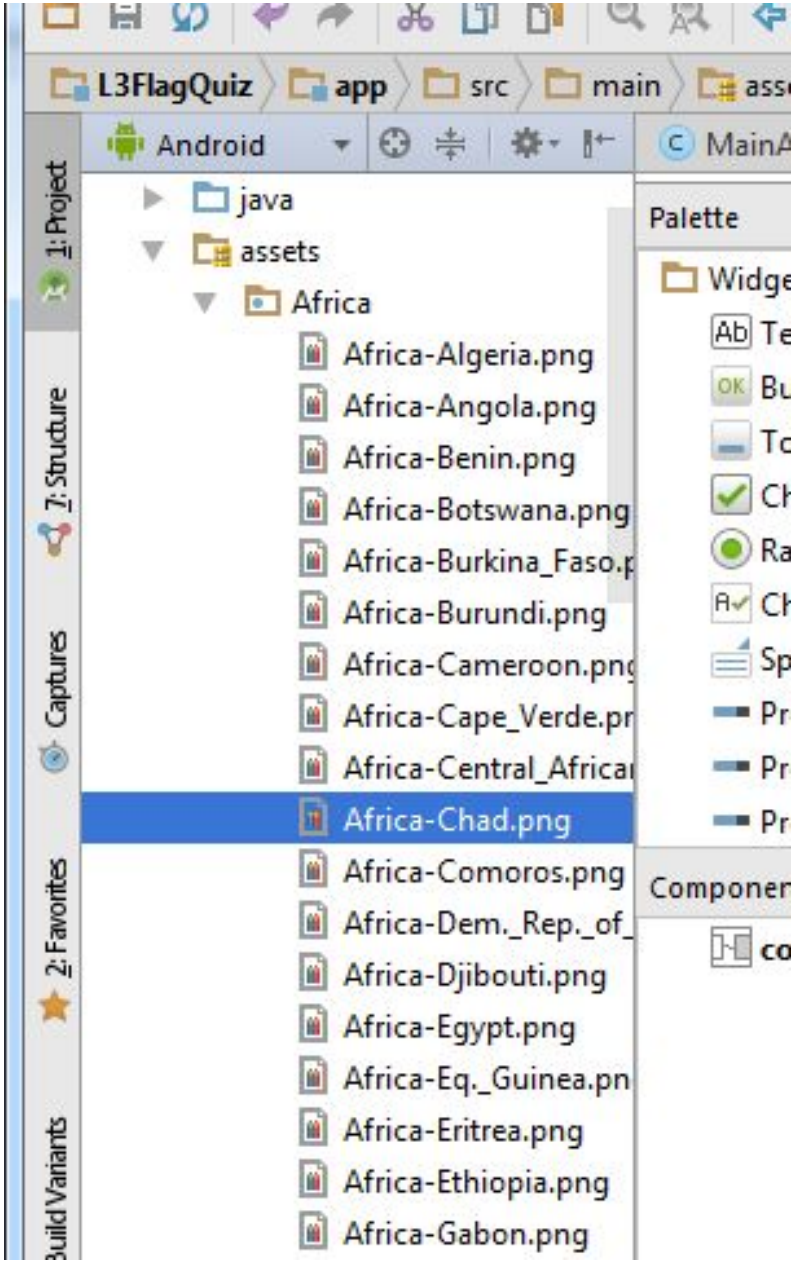


копируем папки в проводнике с изображениями флагов

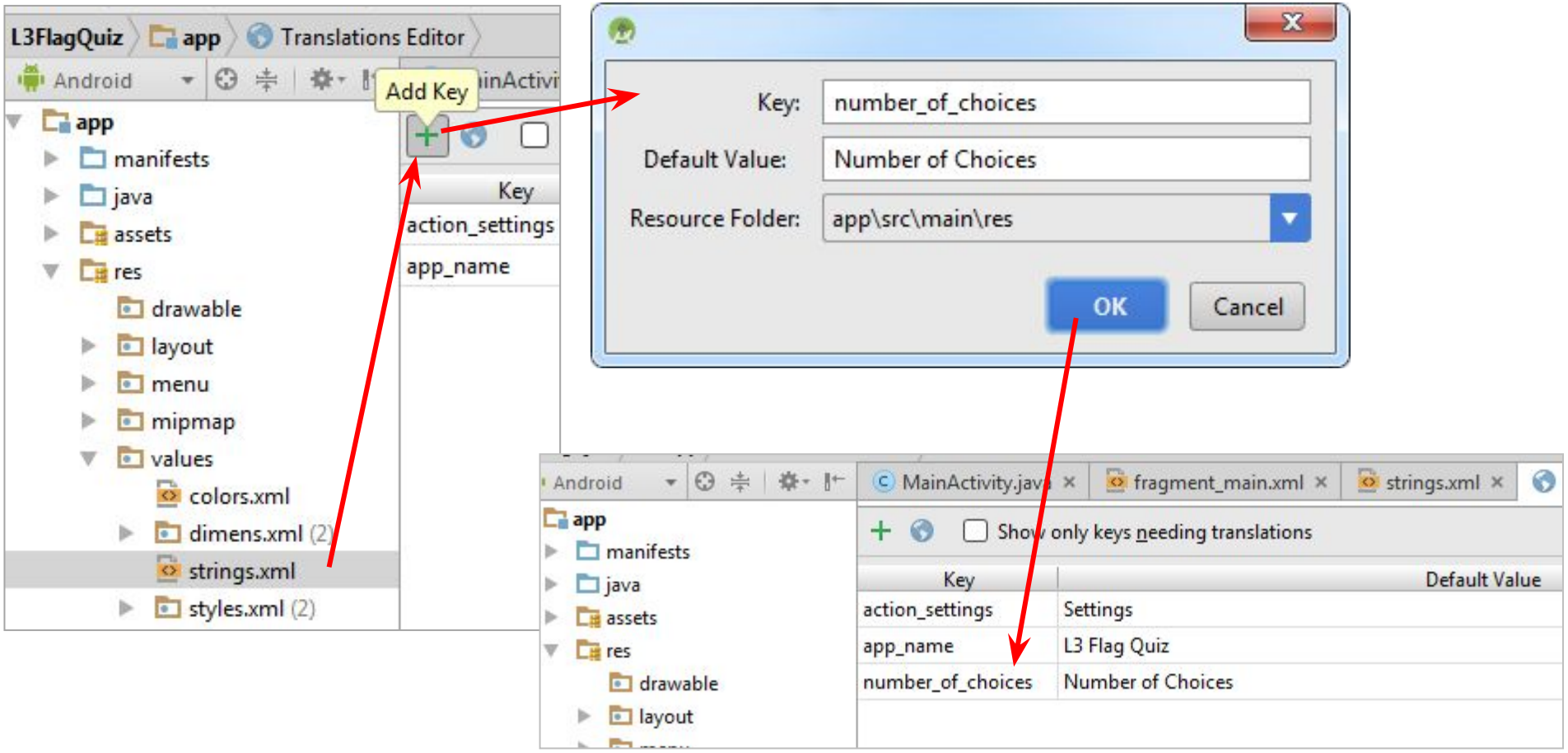


вставляем их в Android Studio в папку assets, затем нажимаем «ОК»

Добавление изображений флагов



Определение строковых ресурсов и форматных строк



аналогично для остальных строковых ресурсов

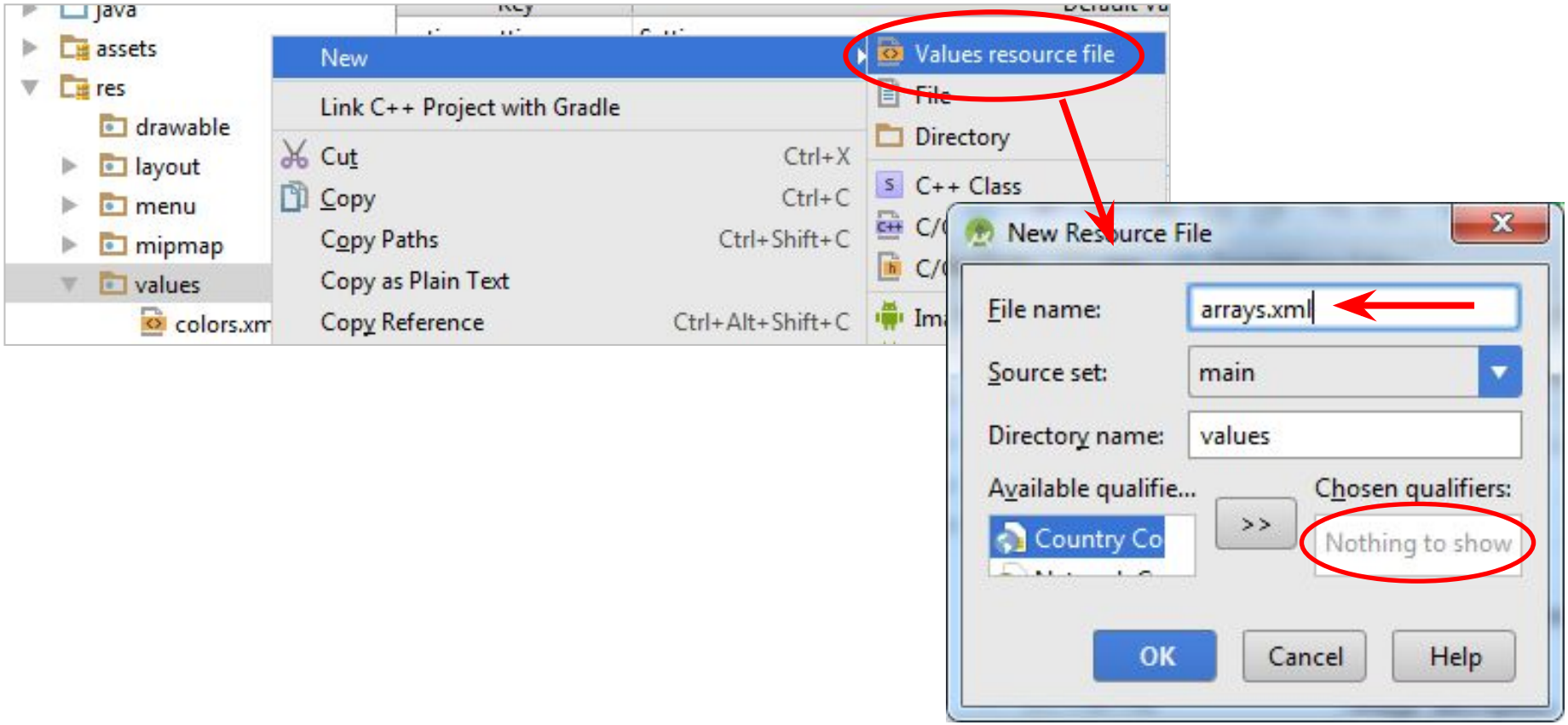
Определение строковых ресурсов и форматных строк

Имя ресурса	Значение
number_of_choices_description	Display 2, 4, 6 or 8 guess buttons
world_regions	Regions
world_regions_description	Regions to include in the quiz
guess_country	Guess the Country
results	%1$d guesses, %2$.02f%% correct
incorrect_answer	Incorrect!
default_region_message	One region must be selected. Setting North America as the default region
restarting_quiz	Quiz will restart with your new setting
question	Question %1$d of %2$d
reset_quiz	Reset Quiz
image_description	Image of the current flag in the quiz
default_region	North_America

форматные строки

Определение ресурсов массивов

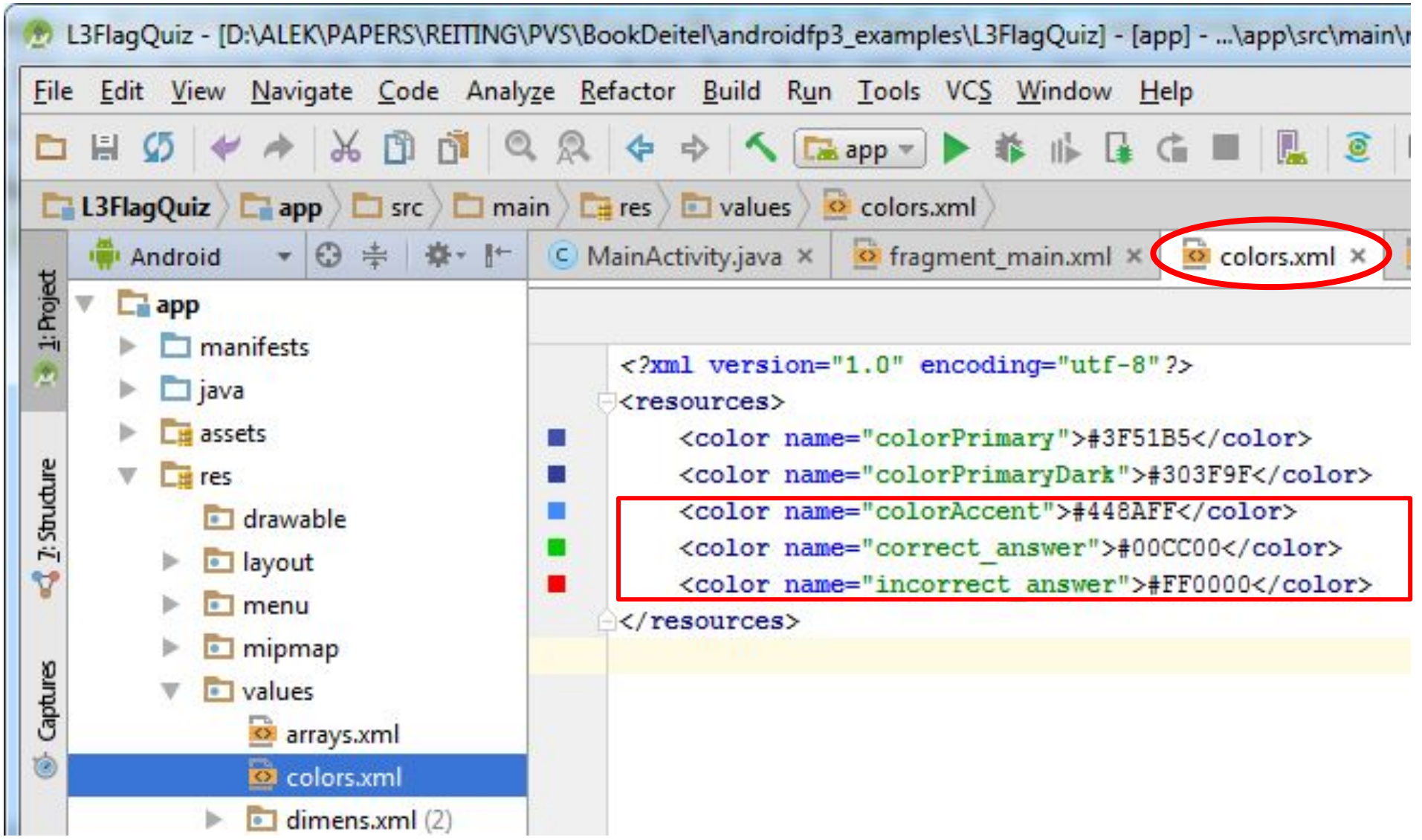
Имя массива ресурса	Значение
regions_list	Africa, Asia, Europe, North_America, Oceania, South_America
regions_list_for_settings	Africa, Asia, Europe, North America, Oceania, South America
guesses_list	2, 4, 6, 8



Определение ресурсов массивов

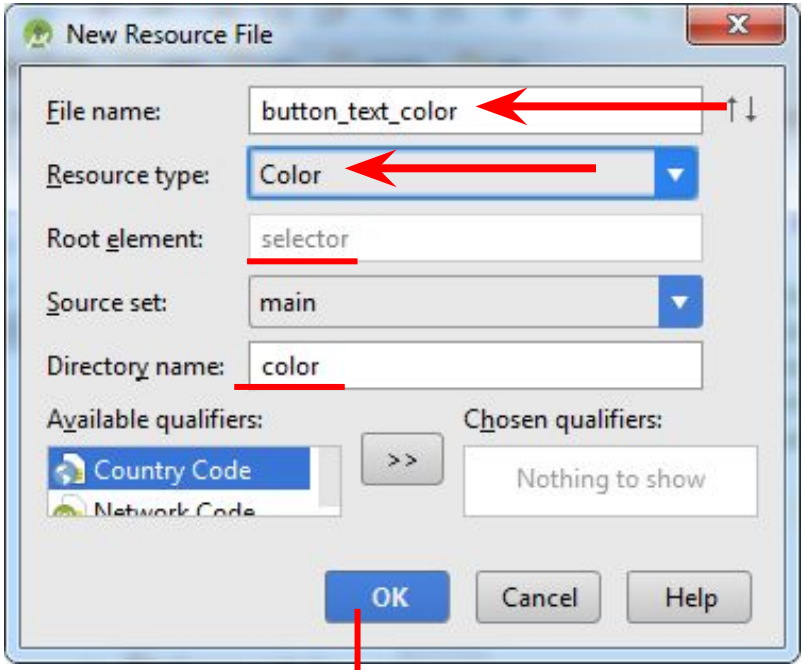
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string-array name="regions_list">
        <item>Africa</item>
        <item>Asia</item>
        <item>Europe</item>
        <item>North_America</item>
        <item>Oceania</item>
        <item>South_America</item>
    </string-array>

    <string-array name="regions_list_for_settings">
        <item>Africa</item>
        <item>Asia</item>
        <item>Europe</item>
        <item>North America</item>
        <item>Oceania</item>
        <item>South America</item>
    </string-array>

    <string-array name="guesses_list">
        <item>2</item>
        <item>4</item>
        <item>6</item>
        <item>8</item>
    </string-array>

</resources>
```

Определение цветовых ресурсов



Определение ресурсов цветов кнопок

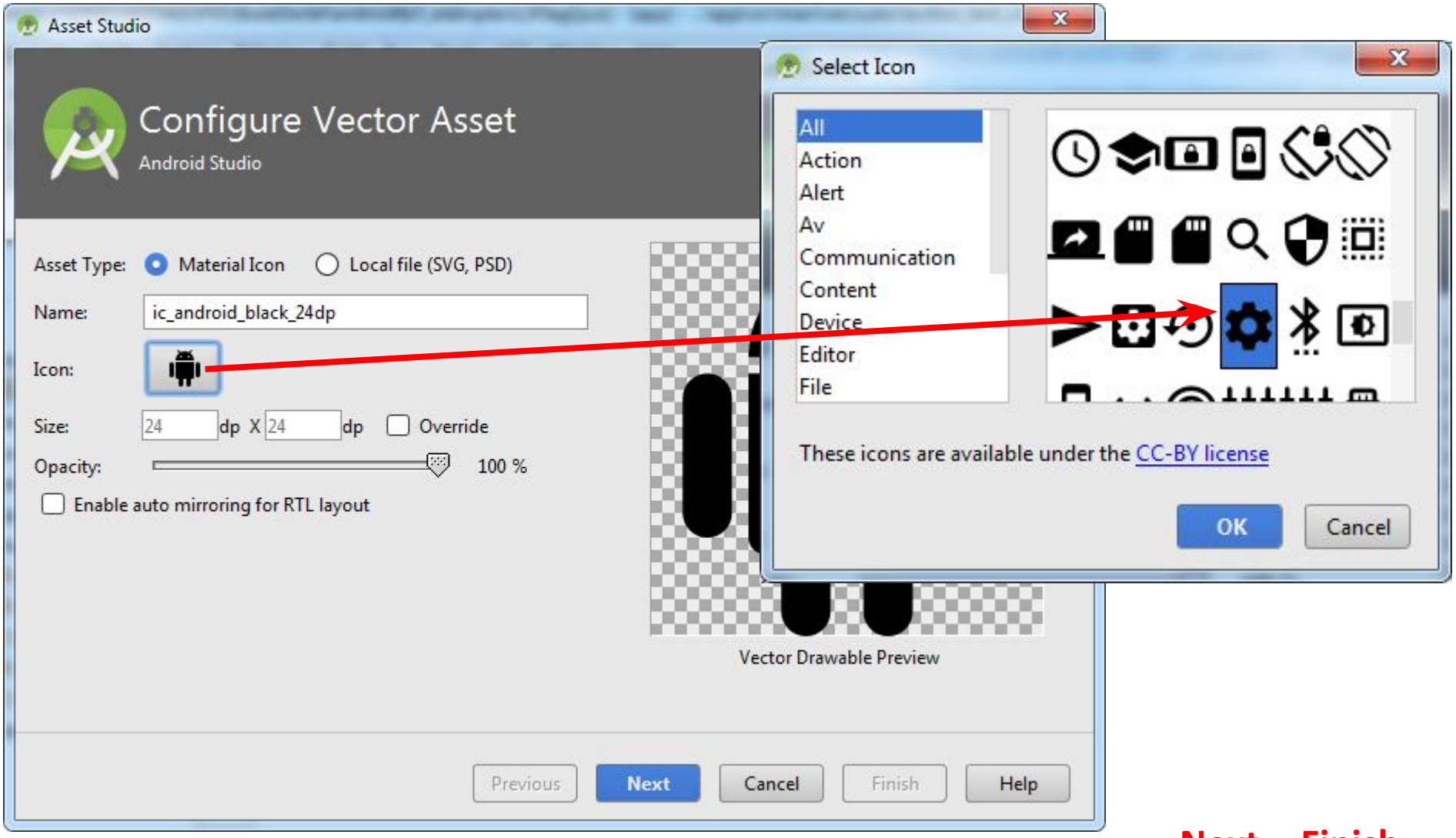
res →New→Android resource file



задаём цвет для доступной кнопки и для заблокированной

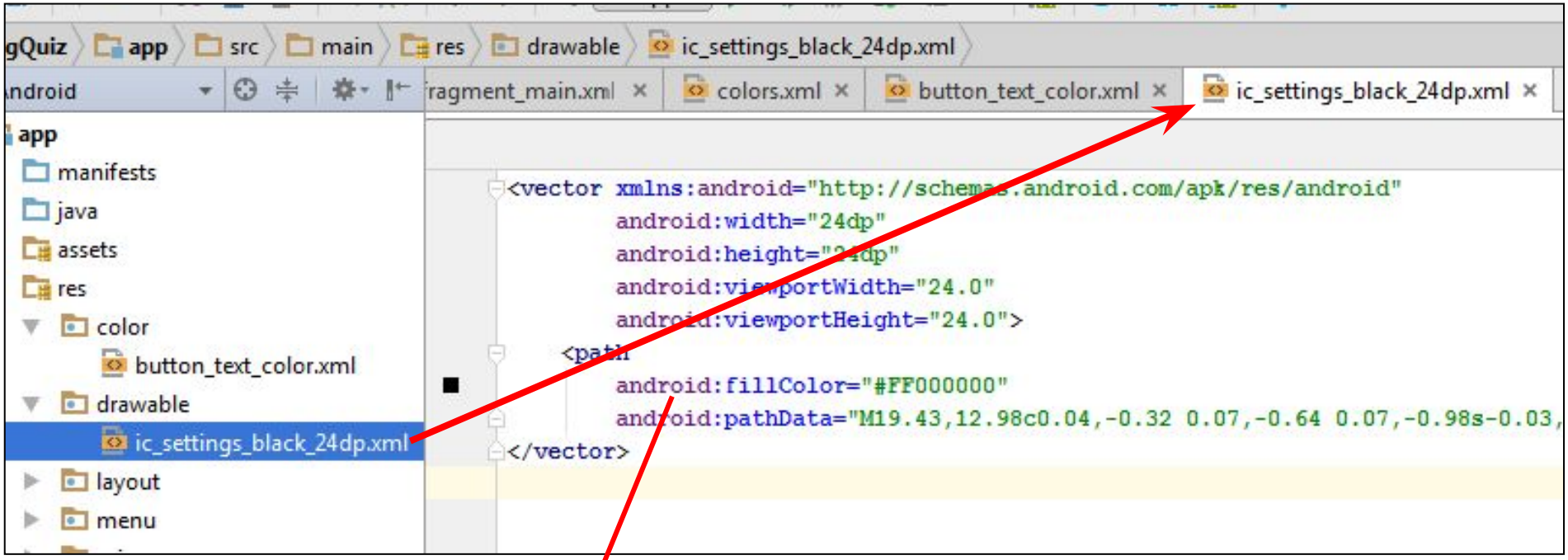
Редактирование ресурса меню

File→New→Vector Asset (при активной корневой папке



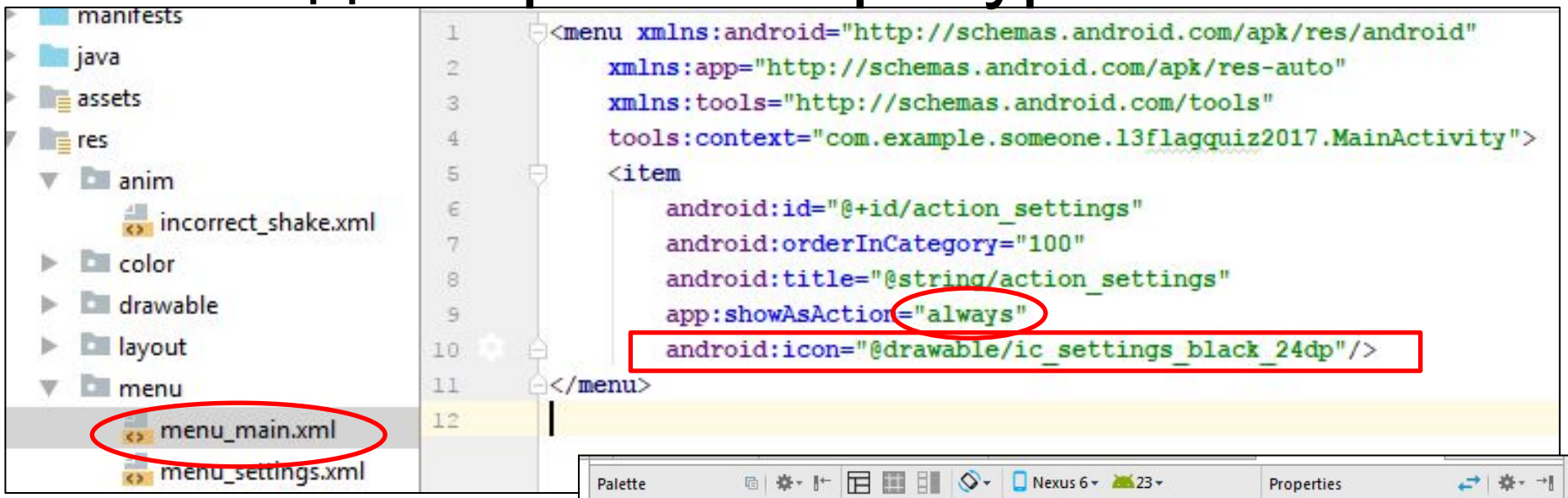
Next→Finish

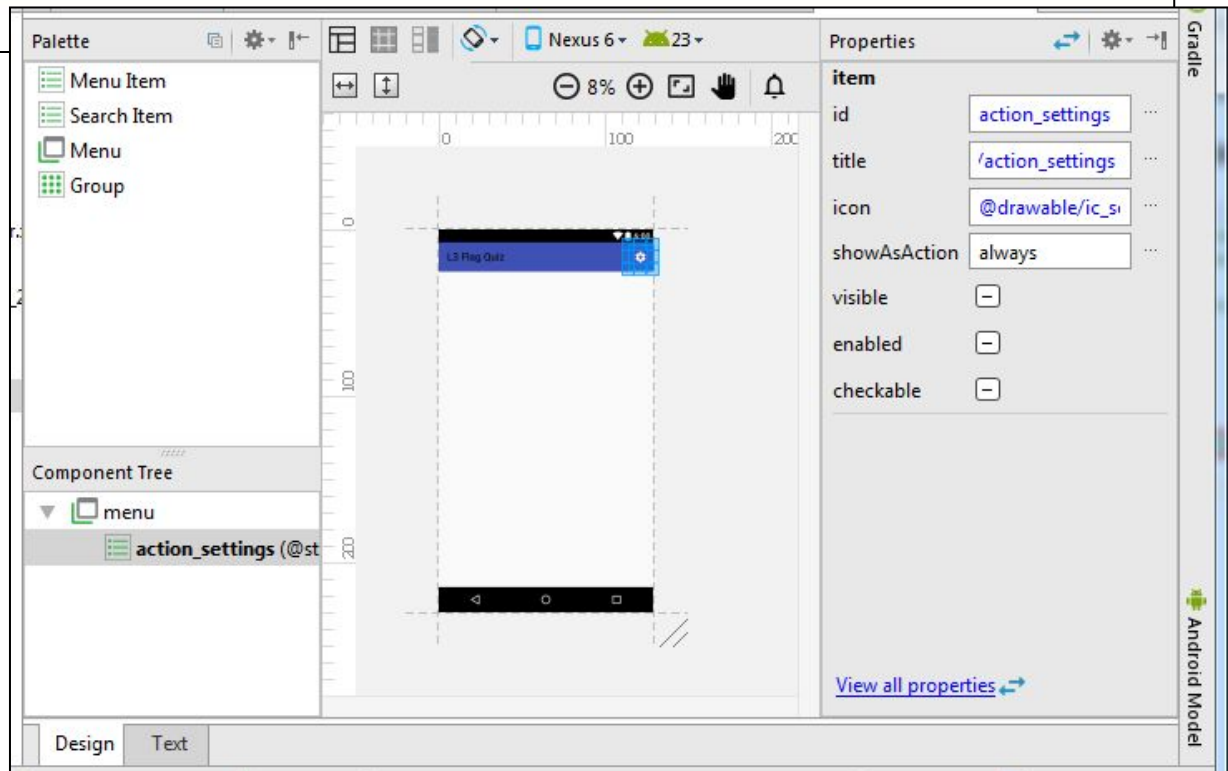
Редактирование ресурса меню




```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
        android:width="24dp"
        android:height="24dp"
        android:viewportWidth="24.0"
        android:viewportHeight="24.0">
    <path
        android:fillColor="@android:color/white"
        android:pathData="M19.43,12.98c0.04,-0.32 0.07,-0.64 0.07,-0.98s-0.03,-0.66 -0
</vector>
```

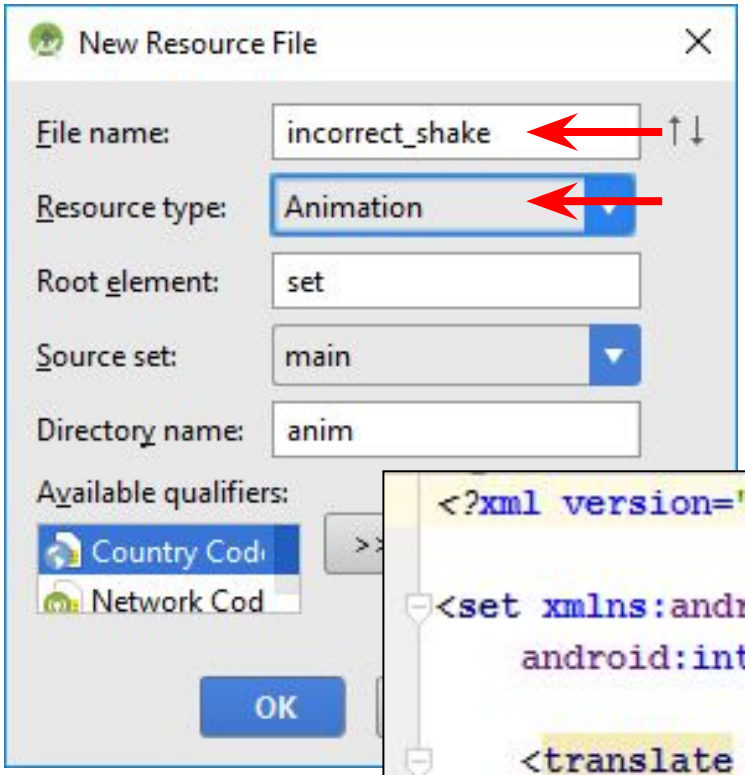
Редактирование ресурса меню





Создание анимации «качающегося» флага

res →New→Android resource file



Типы анимаций с переходами:

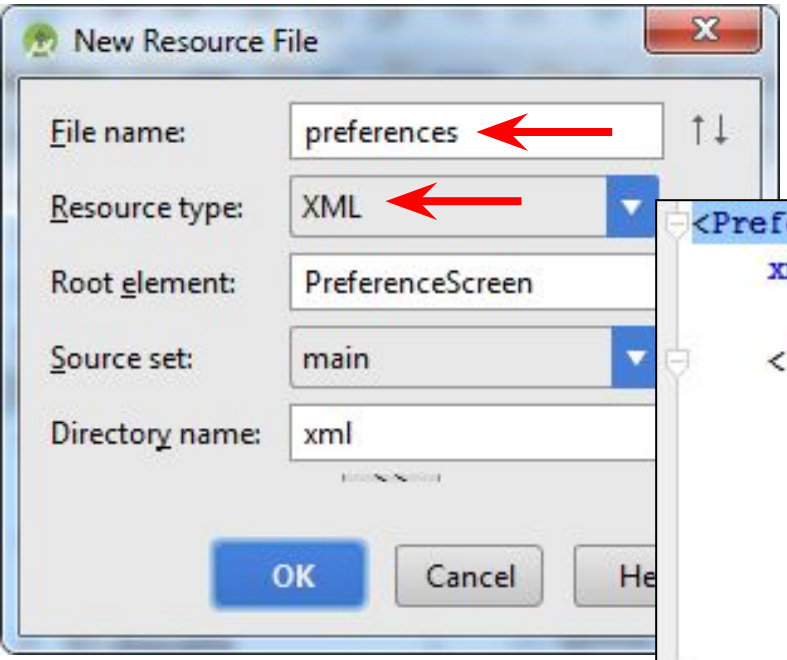
- alpha (прозрачность)
- scale (масштабирование)
- **translate** (перемещение)
- rotate (вращение)

Есть также анимации свойств

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/decelerate_interpolator">

    <translate android:duration="100" android:fromXDelta="0"
        android:toXDelta="-5%p"/>

    <translate android:duration="100" android:fromXDelta="-5%p"
        android:toXDelta="5%p" android:startOffset="100"/>

    <translate android:duration="100" android:fromXDelta="5%p"
        android:toXDelta="-5%p" android:startOffset="200"/>
</set>
```

Определение конфигурации приложения

res →New→Android resource file




```
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">

    <ListPreference
        android:entries="@array/guesses_list"
        android:entryValues="@array/guesses_list"
        android:key="pref_numberOfChoices"
        android:title="@string/number_of_choices"
        android:summary="@string/number_of_choices_description"
        android:persistent="true"
        android:defaultValue="4" />

    <MultiSelectListPreference
        android:entries="@array/regions_list_for_settings"
        android:entryValues="@array/regions_list"
        android:key="pref_regionsToInclude"
        android:title="@string/world_regions"
        android:summary="@string/world_regions_description"
        android:persistent="true"
        android:defaultValue="@array/regions_list" />

</PreferenceScreen>
```

Определение конфигурации приложения

свойства ListPreference

Свойство	Значение	Описание
entries	@array/guesses_list	Массив строк, которые будут отображаться в виде списка вариантов
entryValues	@array/guesses_list	Массив значений, связанных с элементами из свойства Entries. Значение выбранного элемента будет храниться в объекте SharedPreferences
key	pref_numberOfChoices	Имя параметра, хранящегося в SharedPreferences
title	@string/number_of_choices	Заголовок параметра, отображаемый в GUI
summary	@string/number_of_ choices_description	Краткое описание параметра, отображаемое под заголовком
persistent	true	Признак сохранения параметра после завершения приложения — если он равен true, класс PreferenceFragment немедленно сохраняет значение параметра при каждом его изменении
defaultValue	4	Элемент из свойства Entries, которое выбирается по умолчанию

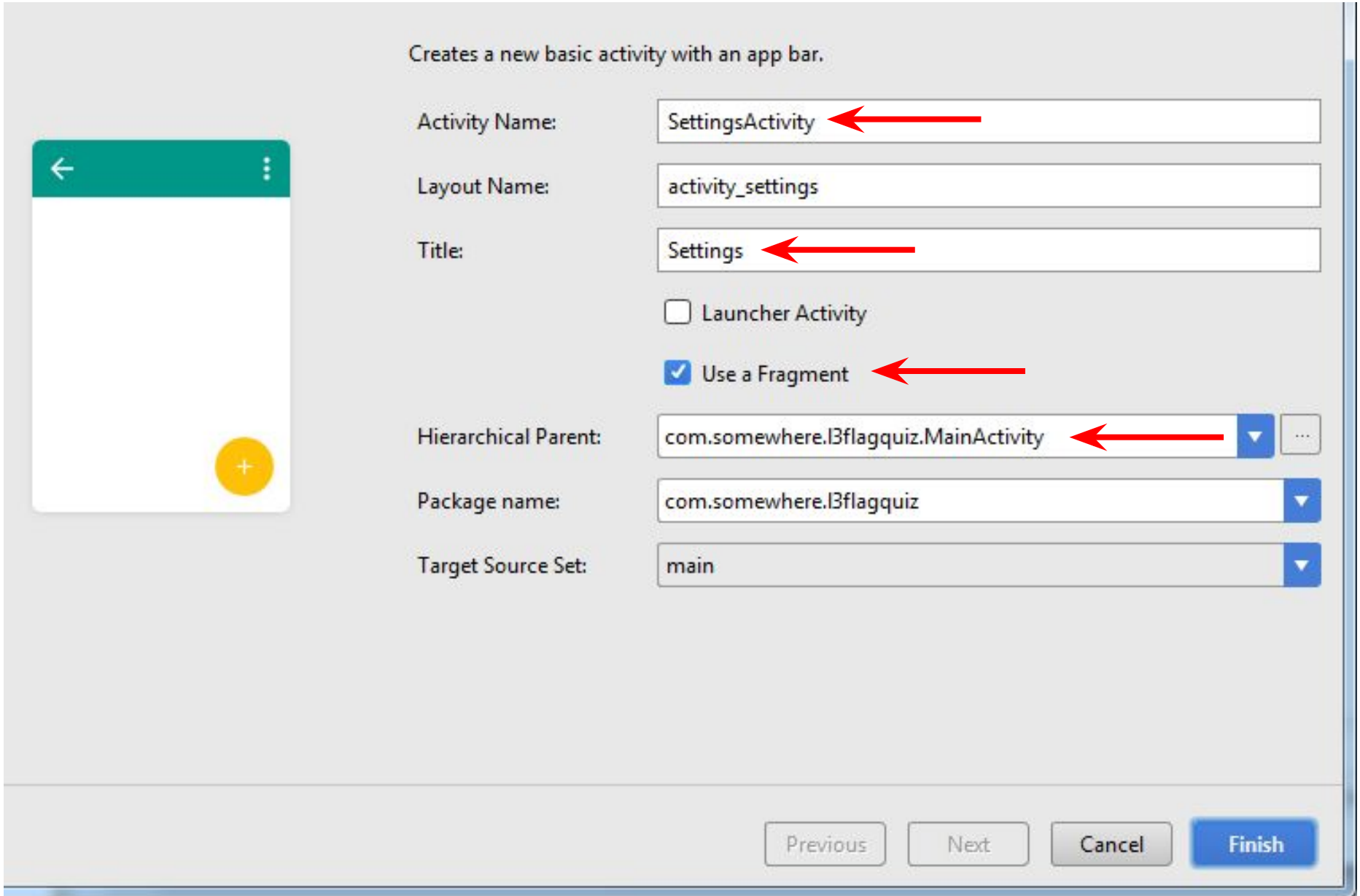
Определение конфигурации приложения

свойства

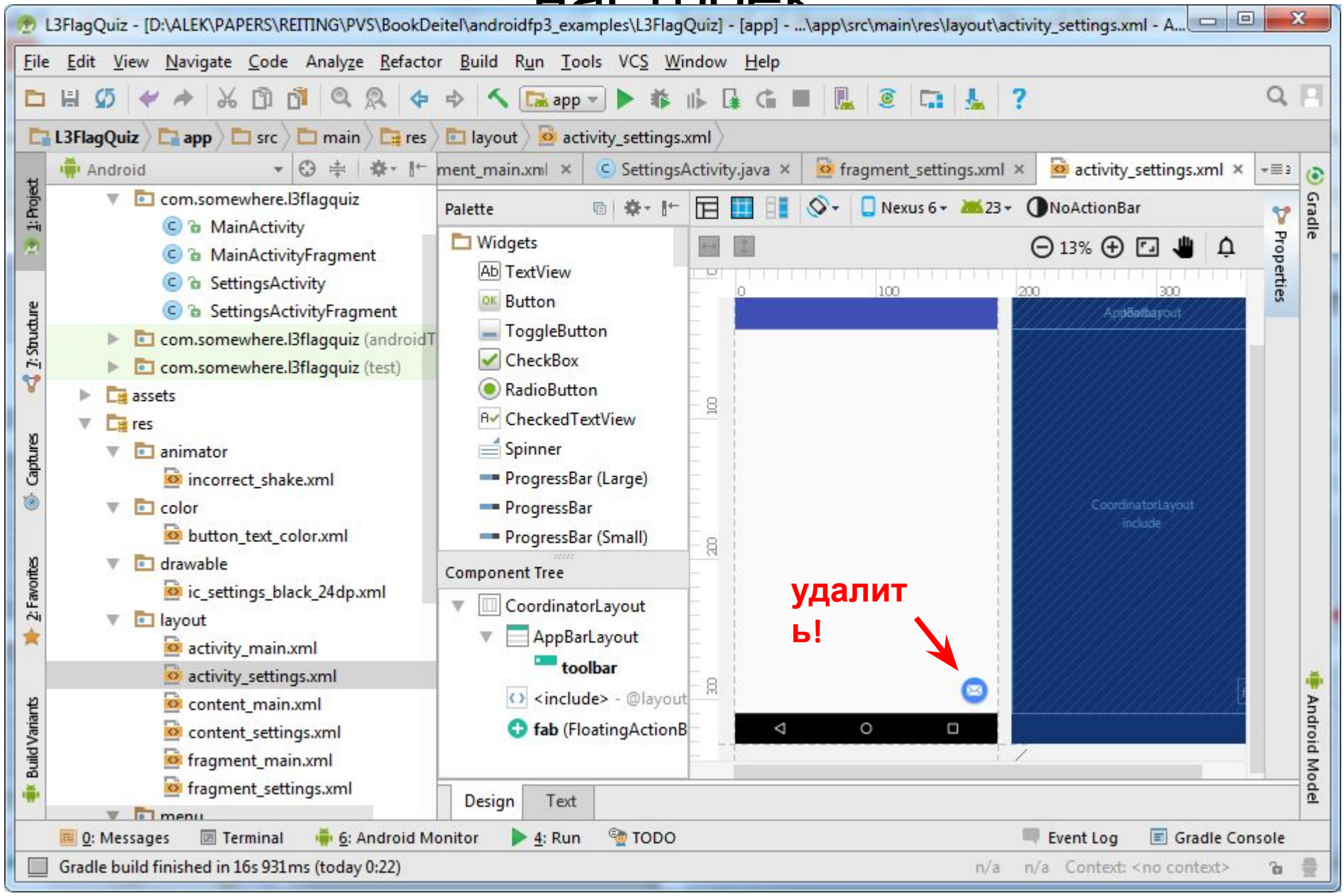
Свойство	Значение	Описание
entries	@array/regions_list_for_settings	Массив строк, которые будут отображаться в виде списка вариантов
entryValues	@array/regions_list	Массив значений, связанных с элементами из свойства Entries. Значения всех выбранных элементов будут храниться в объекте SharedPreferences
key	pref_regionsToInclude	Имя параметра, хранящегося в SharedPreferences
title	@string/world_regions	Заголовок параметра, отображаемый в GUI
summary	@string/world_regions_description	Краткое описание параметра, отображаемое под заголовком
persistent	true	Признак сохранения параметра после завершения приложения
defaultValue	@array/regions_list	Массив значений по умолчанию этого параметра — в данном случае по умолчанию будут выбраны все элементы

Добавление активностей для настроек

app →New →Activity →Basic Activity

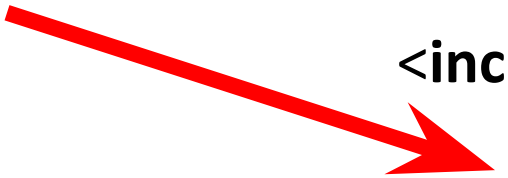


Добавление активностей для настроек



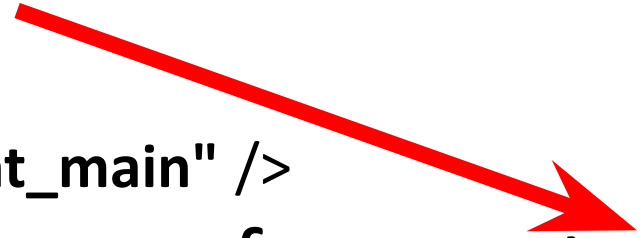
Построение графического интерфейса

activity_main



<include layout="@layout/content_main" />

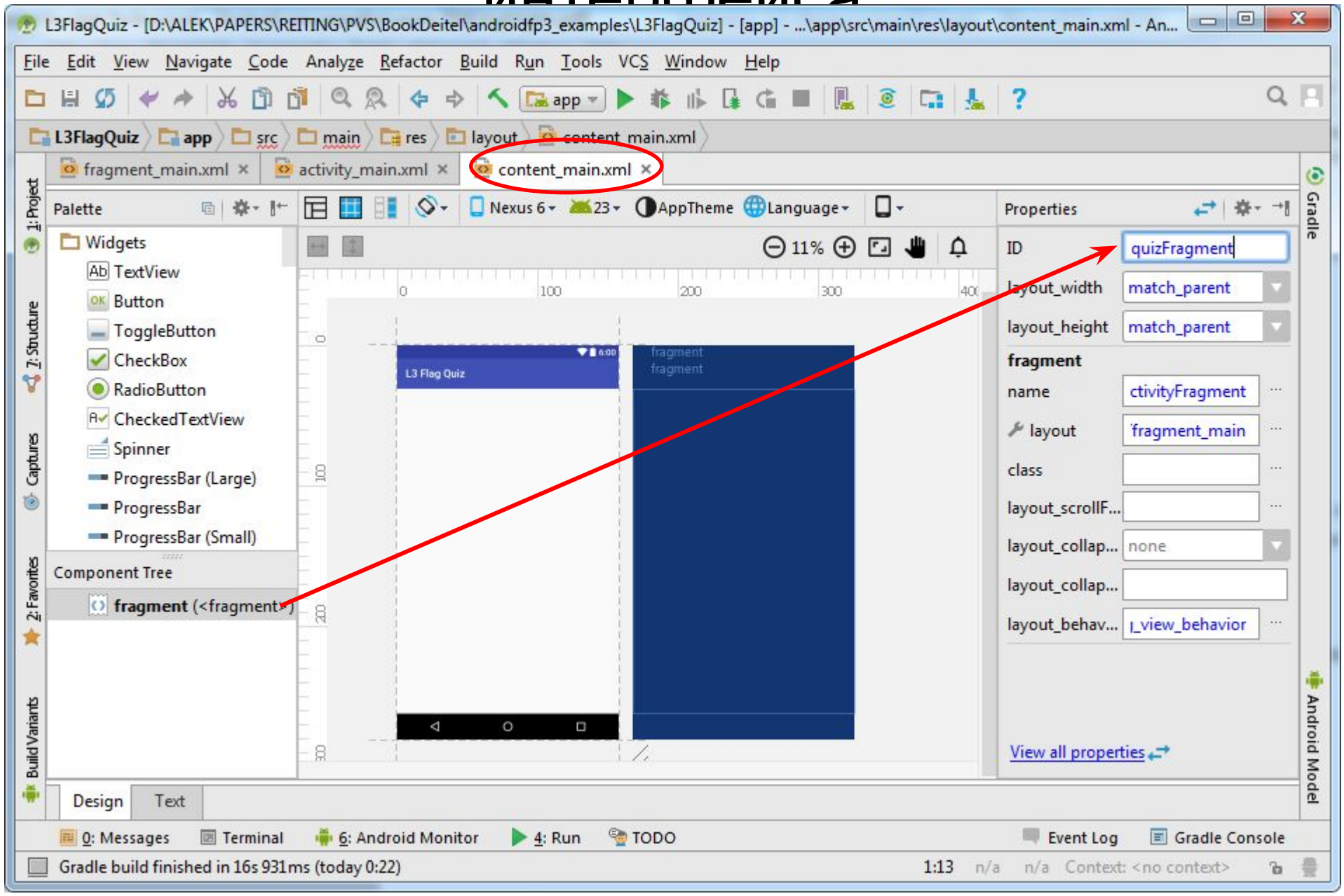
content_main



<fragment …
tools:layout="@layout/fragment_main" />

fragment_main

Построение графического интерфейса



Построение графического интерфейса

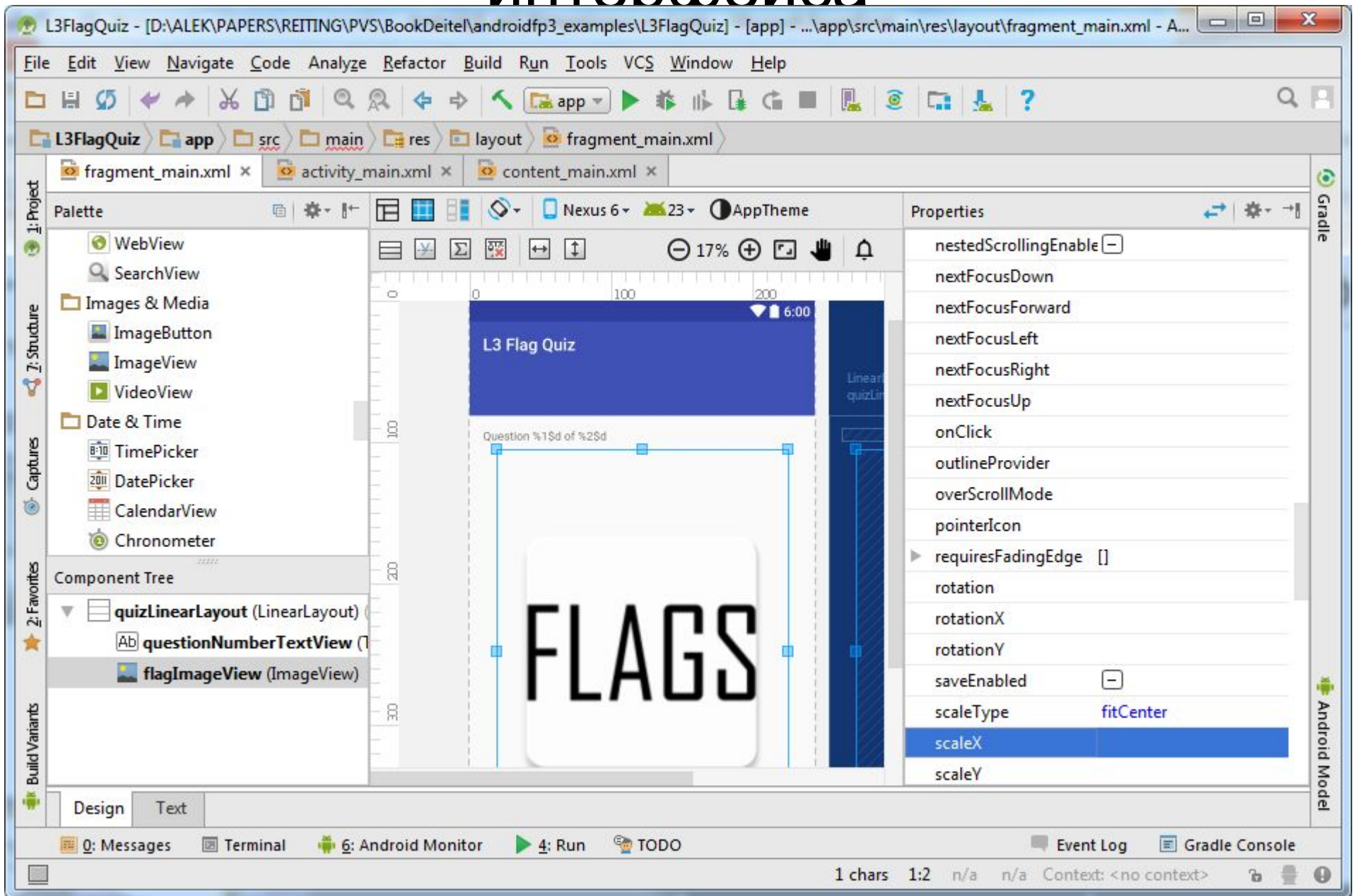
идентификаторы компонентов интерфейса MainActivityFragment

Построение графического интерфейса

Изменяем fragment_main.xml

- android.support.constraint.ConstraintLayout → LinearLayout
  - LinearLayout (fragment_main) .orientation=vertical
  - LinearLayout (fragment_main) .ID=quizLinearLayout
- TextView добавляем в LinearLayout
  - .ID=questionNumberTextView
  - .layout_gravity=center_horizontal
  - .layout_margin:bottom=@dimen/spacing (8dp)
  - .text=@string/question
- ImageView добавляем после questionNumberTextView
  (в диалоговом окне выбрать любой из ресурсов изображений
  - .ID=flagImageView
  - .layout_width=match_parent
  - .layout_height=0dp
  - .layout_gravity=[center]
  - .layout_margin:bottom=@dimen/spacing
  - .layout_margin:left=.layout_margin:right
    =@dimen/fab_margin (16dp)
  - .layout_weight=1
  - .adjustViewBounds=true
  - .contentDescription=@string/image_description
  - .scaleType=fitCenter

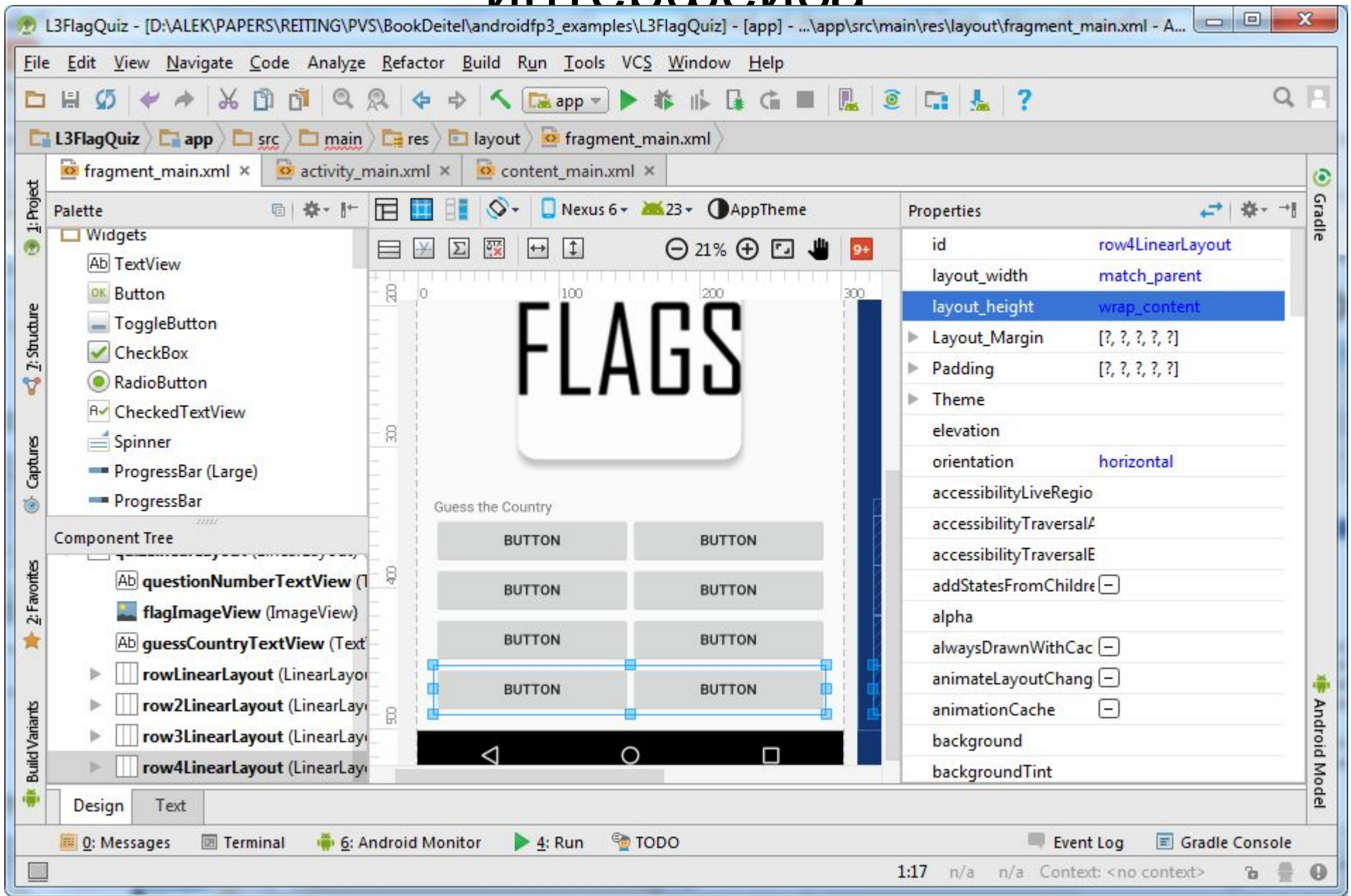
Построение графического интерфейса



Построение графического интерфейса

- TextView добавляем после flagImageView
  - .ID=guessCountryTextView
  - .layout_gravity=center_horizontal
  - .text=@string/guess_country
- добавляем кнопки
  - перетаскиваем LinearLayout (horizontal)
  - .ID=row1LinearLayout
  - row1LinearLayout.layout_height=wrap_context
  - перетаскиваем два раза компонент Button на row1LinearLayout (ID не задаём)
  - создаём аналогично ещё три строки кнопок (row2LinearLayout, row3LinearLayout, row4LinearLayout)

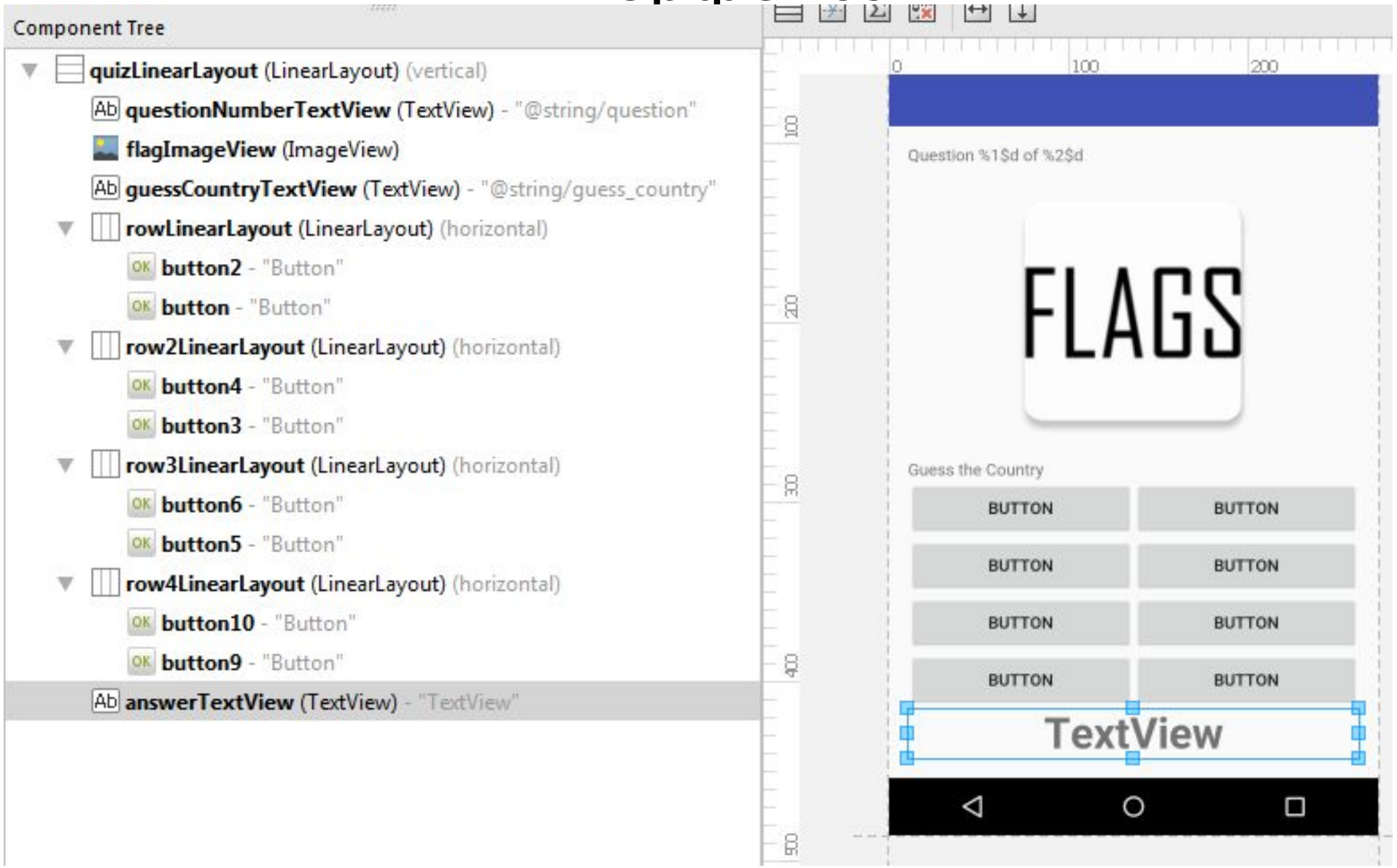
Построение графического интерфейса



Построение графического интерфейса

- TextView добавляем после row4LinearLayout
  - .ID=answerTextView
  - .layout_gravity=[bottom, center_horizontal]
  - .gravity=center_horizontal
  - .textSize=@dimen/answer_size (36sp)
  - .textStyle=bold
  - .text будет задаваться программно

Построение графического интерфейса

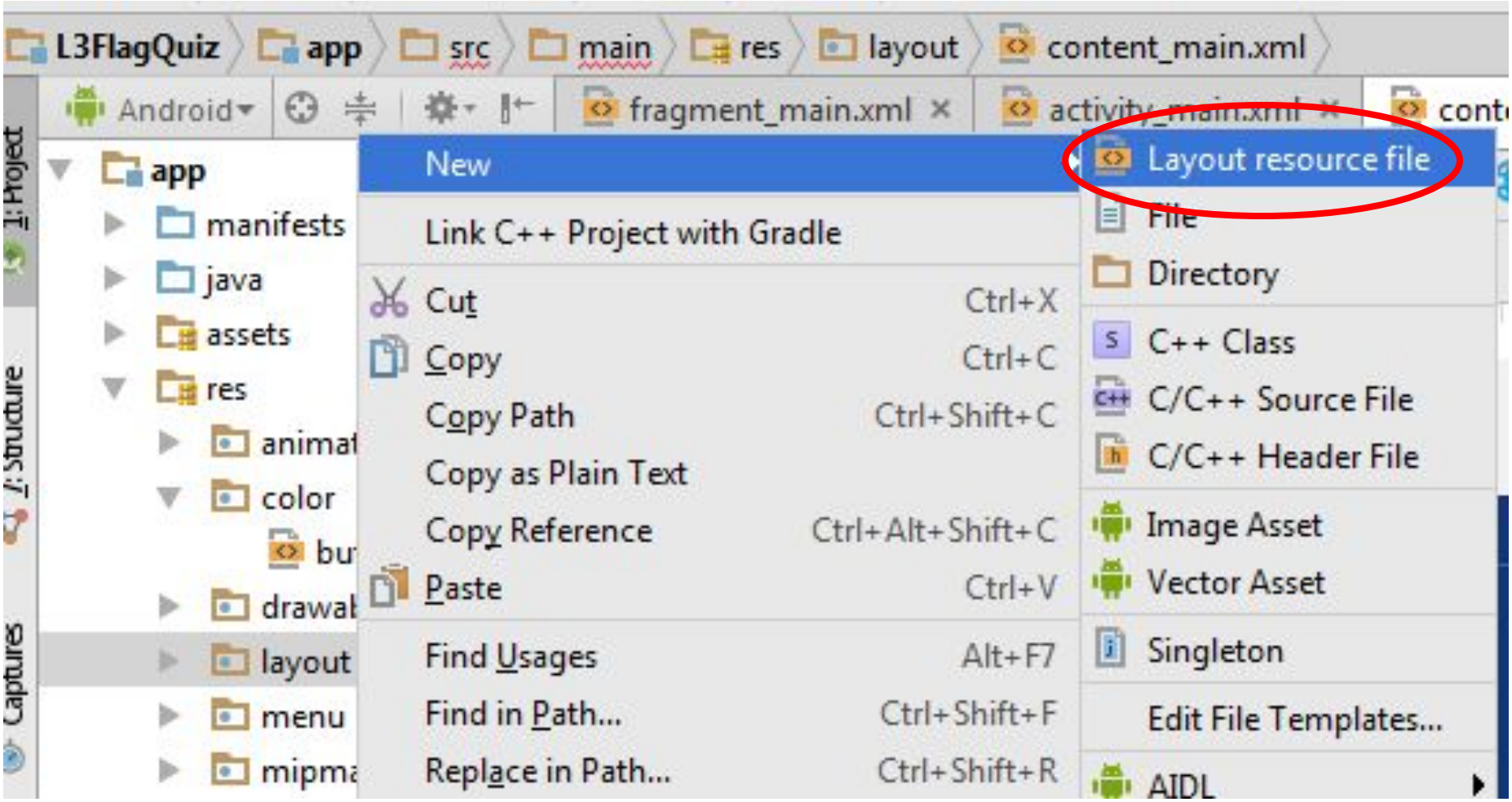


Построение графического интерфейса

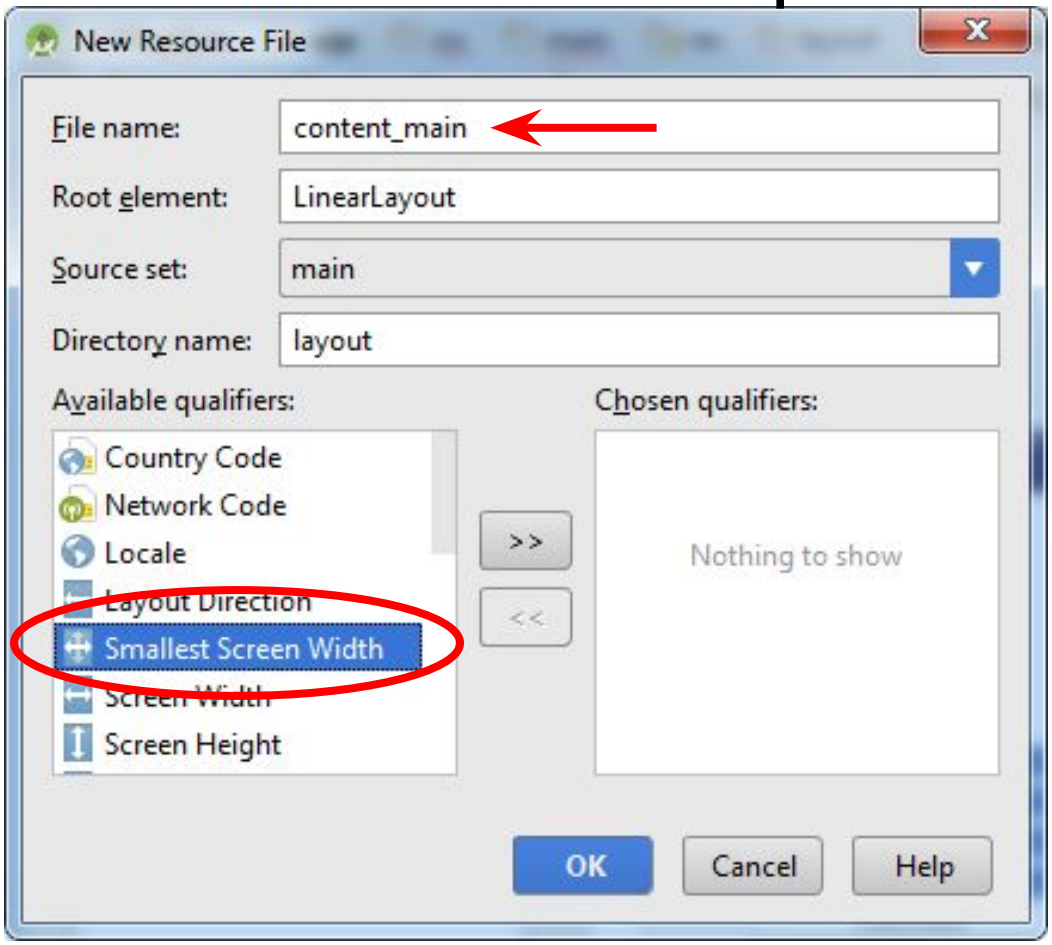
- задаём свойства кнопок
  - .layout_weight=1
  - .layout_height=match_parent
  - .layout_width=0dp
  - .lines=2 (для названий стран разной длины)
  - .style=@android:style/Widget.Material.Button.Colored
  - .textColor=@color/button_text_color (список цветов состояний)

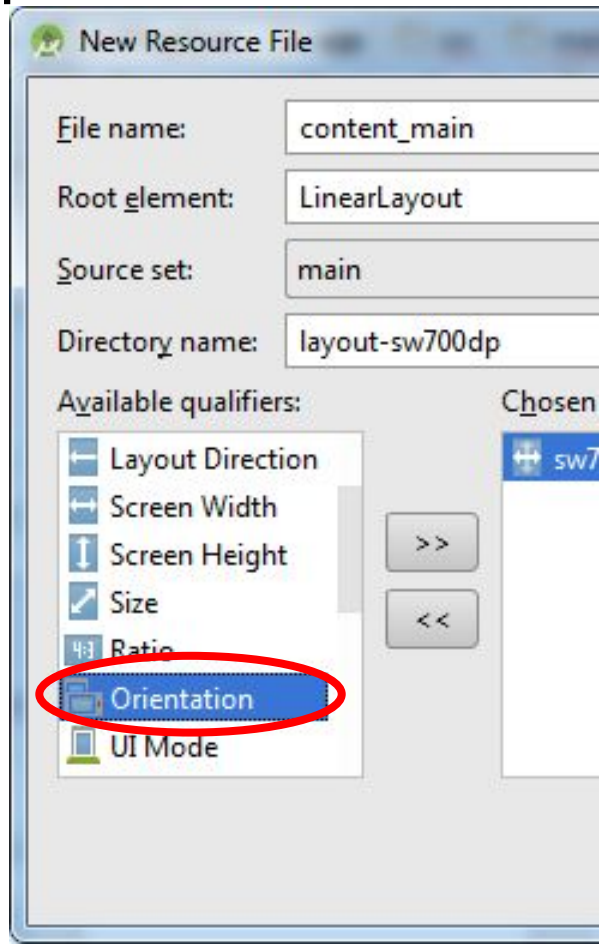
Макет для планшетов в альбомной ориентации

- фрагменты SettingsActivityFragment и MainActivityFragment должны отображаться одновременно



Макет для планшетов в альбомной ориентации

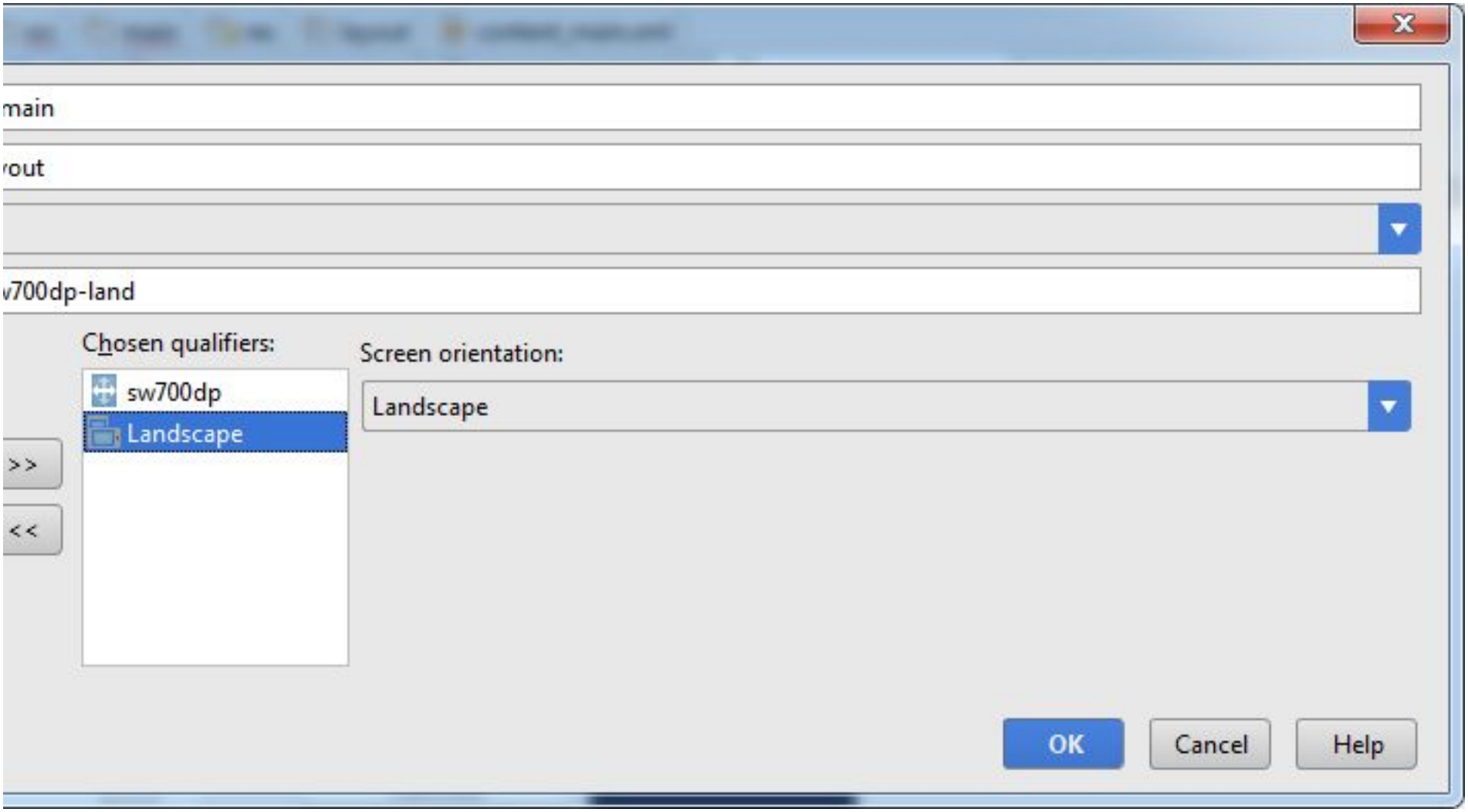




присвоить значение 700

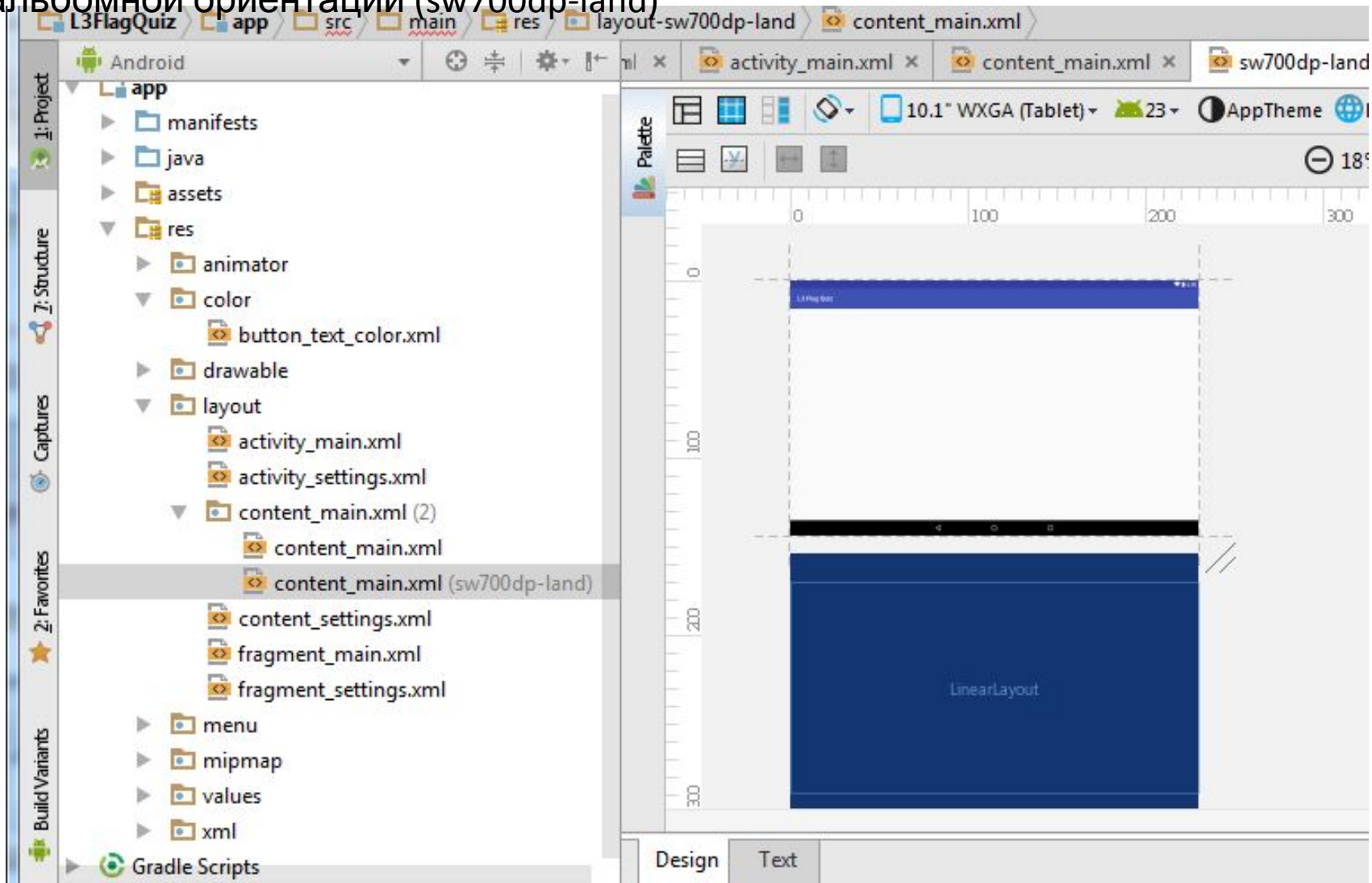
присвоить значение Landscape

Макет для планшетов в альбомной ориентации

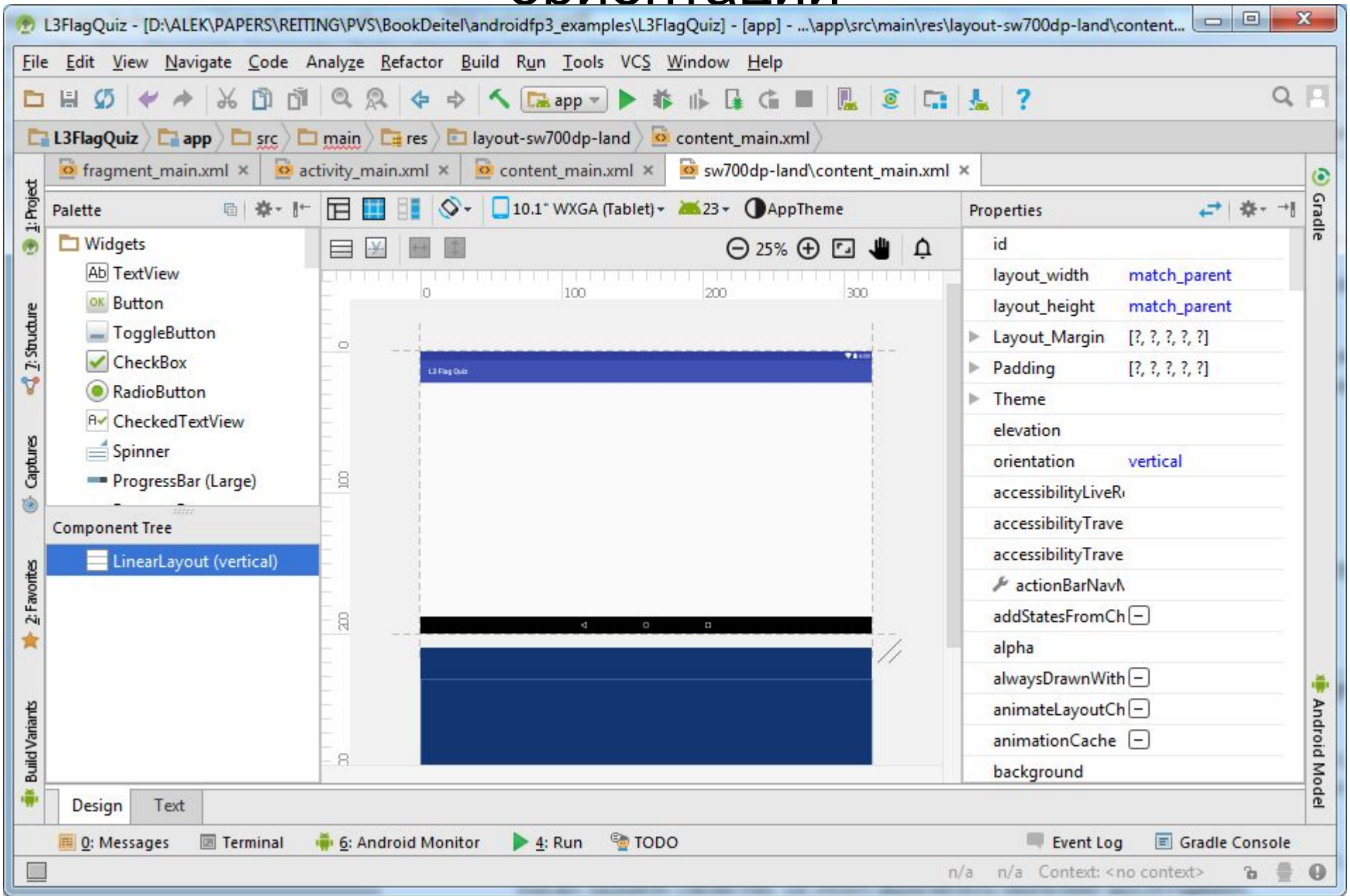


Макет для планшетов в альбомной ориентации

Макет будет использоваться на устройствах с минимальной шириной экрана 700dp
в альбомной ориентации (sw700dp-land)

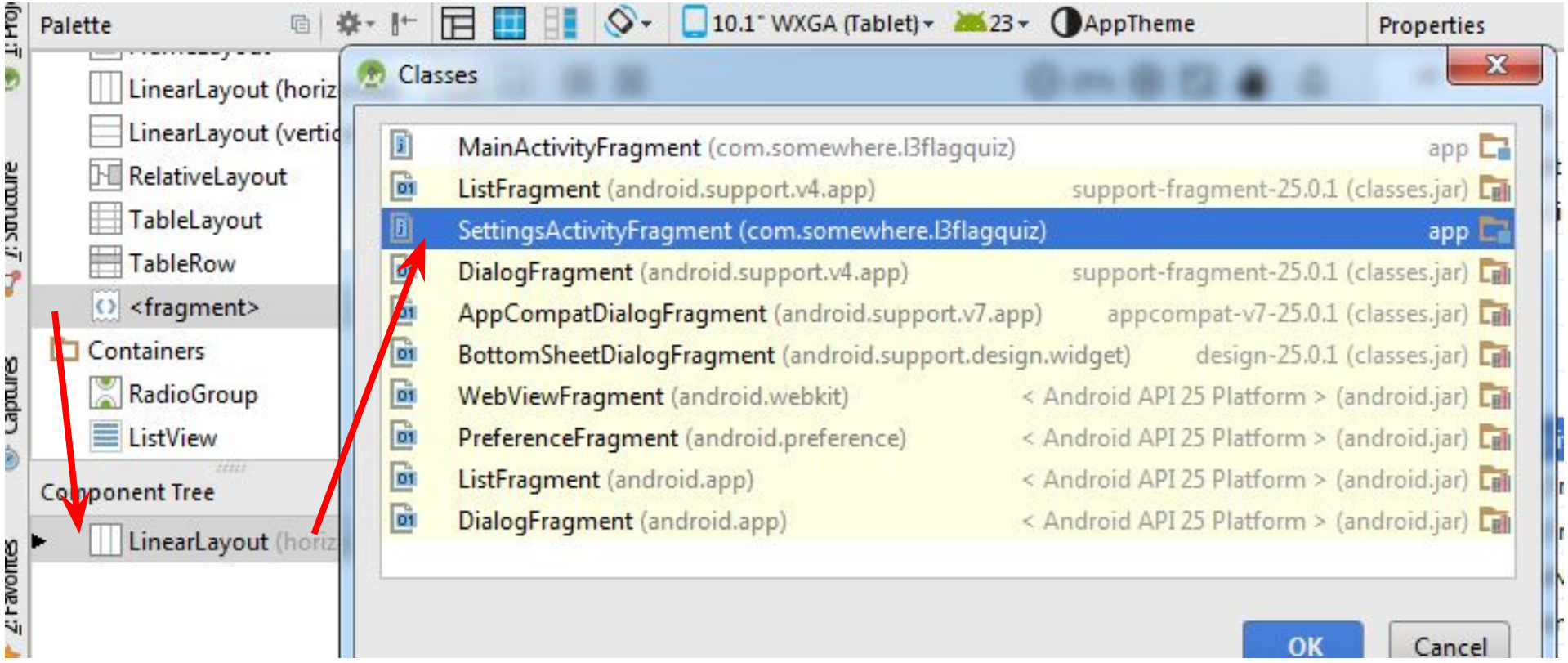


Макет для планшетов в альбомной ориентации



Макет для планшетов в альбомной ориентации

- LinearLayout (vertical) . orientation = horizontal
- добавляем фрагменты settingsActivityFragment (id= settingsActivityFragment) и MainActivityFragment (id=quizFragment)



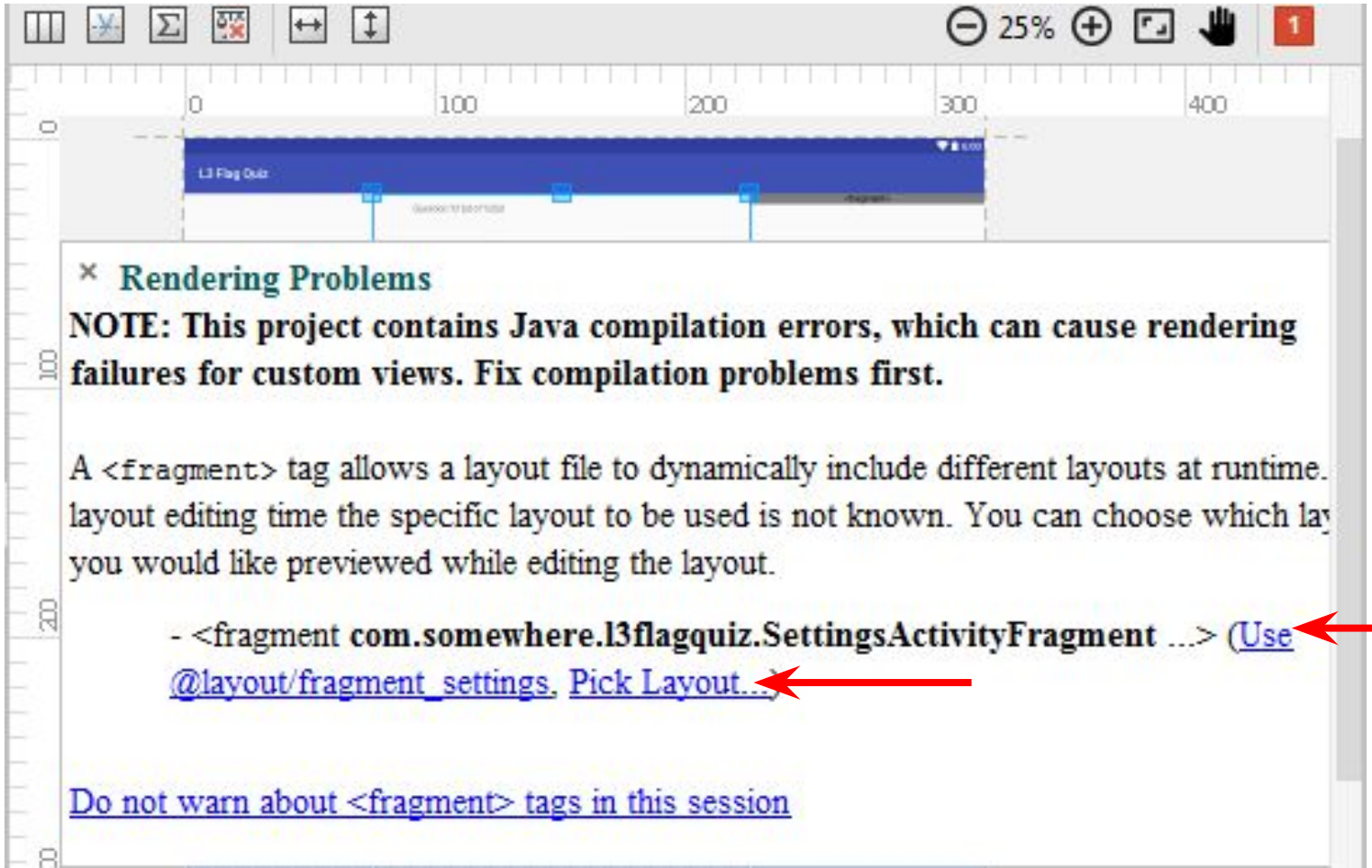
Макет для планшетов в альбомной ориентации

- задаём свойства settingsActivityFragment
  - .layout_width=0dp
  - .layout_height=match_parent
  - .layout_weight=1 (1/3 пространства)
- задаём свойства quizFragment
  - .layout_width=0dp
  - .layout_height=match_parent
  - .layout_weight=2 (2/3 пространства)
- в режиме Text изменяем тег LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="horizontal" android:layout_width="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    android:layout_height="match_parent">
```

Макет для планшетов в альбомной ориентации

- В случае проблем с отображением макета фрагмента в режиме Design необходимо явно указать необходимый фрагмент



Класс MainActivity

- Класс MainActivity управляет фрагментом quizFragment при выполнении приложения в портретной ориентации и фрагментами settingsActivityFragment и quizFragment — когда приложение выполняется на планшете в альбомной ориентации.

Класс MainActivity

Команды package, import и переменные экземпляров

```
// MainActivity.java
// Управляет фрагментом MainActivityFragment на телефоне и  фрагментами
// MainActivityFragment и  SettingsActivityFragment на планшете
package com.somewhere.l3flagquiz;

import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

import java.util.Set;
```

Класс MainActivity

Пол

```
public class MainActivity extends AppCompatActivity {
    // Ключи для чтения данных из SharedPreferences
    public static final String CHOICES = "pref_numberOfChoices";
    public static final String REGIONS = "pref_regionsToInclude";

    private boolean phoneDevice = true; // Включение портретного режима
    private boolean preferencesChanged = true; // Настройки изменились?
```

- **CHOICES**, **REGIONS** – константы для ключей параметров, которые будут использоваться для обращения к соответствующим значениям параметров
- **phoneDevice** – выполняется ли приложение на телефоне (тогда только портретная ориентация)
- **preferencesChanged** – изменились ли настройки приложения (тогда метод onStart вызовет методы изменения конфигурации викторины)

Класс MainActivity

Переопределение метода

```
// Настройка MainActivity
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    // Задание значений по умолчанию в  файле SharedPreferences
    PreferenceManager.setDefaultValues(this, R.xml.preferences, false);
    // Регистрация слушателя для изменений SharedPreferences
    PreferenceManager.getDefaultSharedPreferences(this).
            registerOnSharedPreferenceChangeListener(
                    preferencesChangeListener);
    // Определение размера экрана
    int screenSize = getResources().getConfiguration().screenLayout &
            Configuration.SCREENLAYOUT_SIZE_MASK;
    // Для планшетного устройства phoneDevice присваивается false
    if (screenSize == Configuration.SCREENLAYOUT_SIZE_LARGE ||
            screenSize == Configuration.SCREENLAYOUT_SIZE_XLARGE)
        phoneDevice = false; // not a phone-sized device
    // На телефоне разрешена только портретная ориентация
    if (phoneDevice)
        setRequestedOrientation(
                ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
}
```

Класс MainActivity

Переопределение метода

```
// Настройка MainActivity
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
```

- **setContentView()** – назначение графического интерфейса MainActivity (выбор встраиваемого content_main.xml происходит автоматически на основе аппаратных характеристик устройства)
- **toolbar, setSupportActionBar()** – назначение объекта Toolbar как панели приложения (ранее - панели действия), обеспечение обратной совместимости

Класс MainActivity

Переопределение метода **onCreate**

```
// Задание значений по умолчанию в  файле SharedPreferences
PreferenceManager.setDefaultValues(this, R.xml.preferences, false);
// Регистрация слушателя для изменений SharedPreferences
PreferenceManager.getDefaultSharedPreferences(this).
        registerOnSharedPreferenceChangeListener(
                preferencesChangeListener);
```

- **setDefaultValues()** – задание параметров конфигурации приложения при установке и первом запуске приложения; создаётся и инициализируется файл SharedPreferences; параметры: (1) – окружение, в котором выполняется приложение; (2) – идентификатор ресурса; (3) – значения должны сбрасываться только при первом вызове метода
- **preferencesChangeListener** – регистрация слушателя события изменения настроек

Класс MainActivity

Переопределение метода **onCreate**

```
// Определение размера экрана
int screenSize = getResources().getConfiguration().screenLayout &
        Configuration.SCREENLAYOUT_SIZE_MASK;
// Для планшетного устройства phoneDevice присваивается false
if (screenSize == Configuration.SCREENLAYOUT_SIZE_LARGE ||
        screenSize == Configuration.SCREENLAYOUT_SIZE_XLARGE)
    phoneDevice = false; // not a phone-sized device
// На телефоне разрешена только портретная ориентация
if (phoneDevice)
    setRequestedOrientation(
            ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
```

- определение размера экрана (screenSize)
- проверка того, является ли он большим; если да, то это не телефон
- если телефон, то установка портретной ориентации

Класс MainActivity

Переопределение метода **onStart**

Вызов:

- при первом запуске приложения после onCreate()
- при портретной ориентации после отображения настроек

```
// Вызывается после завершения выполнения onCreate
@Override
protected void onStart() {
    super.onStart();

    if (preferencesChanged) {
        // После задания настроек по умолчанию инициализировать
        // MainActivityFragment и  запустить викторину
        MainActivityFragment quizFragment = (MainActivityFragment)
                getSupportFragmentManager().findFragmentById(
                        R.id.quizFragment);
        quizFragment.updateGuessRows(
                PreferenceManager.getDefaultSharedPreferences(this));
        quizFragment.updateRegions(
                PreferenceManager.getDefaultSharedPreferences(this));
        quizFragment.resetQuiz();
        preferencesChanged = false;
    }
}
```

Класс MainActivity

Переопределение метода
onCreateOptionsMenu
Инициализирует стандартное меню активности

```
// Меню отображается на телефоне или планшете в  портретной ориентации
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Получение текущей ориентации устройства
    int orientation = getResources().getConfiguration().orientation;

    // Отображение меню приложения только в  портретной ориентации
    if (orientation == Configuration.ORIENTATION_PORTRAIT) {
        // Заполнение меню
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
    else
        return false;
}
```

Класс MainActivity

Переопределение метода **onOptionsItemSelected**

Вызывается при выборе команды меню (Settings)

Создаёт явный интент для запуска SettingsActivity и передаёт его унаследованному методу StartActivity

```
// Отображает SettingsActivity при запуске на телефоне
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Intent preferencesIntent = new Intent(this, SettingsActivity.class);
    startActivity(preferencesIntent);
    return super.onOptionsItemSelected(item);
}
```

Класс MainActivity

Прослушивание изменений в конфигурации

```
// Слушатель изменений в конфигурации SharedPreferences приложения
private OnSharedPreferenceChangeListener preferencesChangeListener =
    new OnSharedPreferenceChangeListener() {
    // Вызывается при изменении настроек приложения
    @Override
    public void onSharedPreferenceChanged(
            SharedPreferences sharedPreferences, String key) {
        preferencesChanged = true; // Пользователь изменил настройки

        MainActivityFragment quizFragment = (MainActivityFragment)
                getSupportFragmentManager().findFragmentById(
                R.id.quizFragment);

        if (key.equals(CHOICES)) { // Изменилось число вариантов
            quizFragment.updateGuessRows(sharedPreferences);
            quizFragment.resetQuiz();
            }
        else if (key.equals(REGIONS)) { // Изменились регионы
            Set<String> regions =
                    sharedPreferences.getStringSet(REGIONS, null);

            if (regions != null && regions.size() > 0) {
                quizFragment.updateRegions(sharedPreferences);
                quizFragment.resetQuiz();
                }
            else {
                // Хотя бы один регион - по умолчанию Северная Америка
                SharedPreferences.Editor editor =
                        sharedPreferences.edit();
```

Класс MainActivity

Прослушивание изменений в конфигурации

```
        if (regions != null && regions.size() > 0) {
            quizFragment.updateRegions(sharedPreferences);
            quizFragment.resetQuiz();
        }
        else {
            // Хотя бы один регион - по умолчанию Северная Америка
            SharedPreferences.Editor editor =
                    sharedPreferences.edit();
            regions.add(getString(R.string.default_region));
            editor.putStringSet(REGIONS, regions);
            editor.apply();

            Toast.makeText(MainActivity.this,
                    R.string.default_region_message,
                    Toast.LENGTH_SHORT).show();
        }
    }

    Toast.makeText(MainActivity.this,
            R.string.restarting_quiz,
            Toast.LENGTH_SHORT).show();
};
```

Класс MainActivityFragment

Строит графический интерфейс игры и реализует её логику

```
// MainActivityFragment.java
// Класс содержит логику приложения Flag Quiz
package com.somewhere.l3flagquiz;

import java.io.IOException;
import java.io.InputStream;
import java.security.SecureRandom;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Set;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.content.res.AssetManager;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Handler;
import android.support.v4.app.DialogFragment;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewAnimationUtils;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
```

Класс MainActivityFragment

Статические поля и поля экземпляров

```
public class MainActivityFragment extends Fragment {
    // Строка, используемая при регистрации сообщений об ошибках
    private static final String TAG = "FlagQuiz Activity";

    private static final int FLAGS_IN_QUIZ = 10;

    private List<String> fileNameList; // Имена файлов с  флагами
    private List<String> quizCountriesList; // Страны текущей викторины
    private Set<String> regionsSet; // Регионы текущей викторины
    private String correctAnswer; // Правильная страна для текущего флага
    private int totalGuesses; // Количество попыток
    private int correctAnswers; // Количество правильных ответов
    private int guessRows; // Количество строк с  кнопками вариантов
    private SecureRandom random; // Генератор случайных чисел
    private Handler handler; // Для задержки загрузки следующего флага
    private Animation shakeAnimation; // Анимация неправильного ответа

    private LinearLayout quizLinearLayout; // Макет с  викториной
    private TextView questionNumberTextView; // Номер текущего вопроса
    private ImageView flagImageView; // Для вывода флага
    private LinearLayout[] guessLinearLayouts; // Строки с  кнопками
    private TextView answerTextView; // Для правильного ответа
```

Класс MainActivityFragment

Переопределение метода onCreateView

- заполнение графического интерфейса
- динамическая загрузка анимации

```
// Настройка MainActivityFragment при создании представления
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    View view =
            inflater.inflate(R.layout.fragment_main, container, false);

    fileNameList = new ArrayList<>(); // Оператор <>
    quizCountriesList = new ArrayList<>();
    random = new SecureRandom();
    handler = new Handler();

    // Загрузка анимации для неправильных ответов
    shakeAnimation = AnimationUtils.loadAnimation(getActivity(),
            R.anim.incorrect_shake);
    shakeAnimation.setRepeatCount(3); // Анимация повторяется 3 раза
```

Класс MainActivityFragment

Переопределение метода onCreateView

- получение ссылок на интерфейсные элементы

```
// Получение ссылок на компоненты графического интерфейса
quizLinearLayout =
        (LinearLayout) view.findViewById(R.id.quizLinearLayout);
questionNumberTextView =
        (TextView) view.findViewById(R.id.questionNumberTextView);
flagImageView = (ImageView) view.findViewById(R.id.flagImageView);
guessLinearLayouts = new LinearLayout[4];
guessLinearLayouts[0] =
        (LinearLayout) view.findViewById(R.id.row1LinearLayout);
guessLinearLayouts[1] =
        (LinearLayout) view.findViewById(R.id.row2LinearLayout);
guessLinearLayouts[2] =
        (LinearLayout) view.findViewById(R.id.row3LinearLayout);
guessLinearLayouts[3] =
        (LinearLayout) view.findViewById(R.id.row4LinearLayout);
answerTextView = (TextView) view.findViewById(R.id.answerTextView);
```

Класс MainActivityFragment

Переопределение метода onCreateView

- назначение слушателей нажатиям кнопок ответов
- отображение номера вопроса
- возврат сформированного графического интерфейса

```
    // Настройка слушателей для кнопок ответов
    for (LinearLayout row : guessLinearLayouts) {
        for (int column = 0; column < row.getChildCount(); column++) {
            Button button = (Button) row.getChildAt(column);
            button.setOnClickListener(guessButtonListener);
        }
    }

    // Назначение текста questionNumberTextView
    questionNumberTextView.setText(
            getString(R.string.question, 1, FLAGS_IN_QUIZ));
    return view; // Возвращает представление фрагмента для вывода
}
```

Класс MainActivityFragment

Метод updateGuessRows

- вызывается из MainActivity при запуске приложения и при изменении количества вариантов ответа
- вычисление количества рядов кнопок
- скрытие всех рядов (View.GONE) и отображение нужных

```
// Обновление guessRows на основании значения SharedPreferences
public void updateGuessRows(SharedPreferences sharedPreferences) {
    // Получение количества отображаемых вариантов ответа
    String choices =
            sharedPreferences.getString(MainActivity.CHOICES, null);
    guessRows = Integer.parseInt(choices) / 2;

    // Все компоненты LinearLayout скрываются
    for (LinearLayout layout : guessLinearLayouts)
        layout.setVisibility(View.GONE);

    // Отображение нужных компонентов LinearLayout
    for (int row = 0; row < guessRows; row++)
        guessLinearLayouts[row].setVisibility(View.VISIBLE);
}
```

Класс MainActivityFragment

Метод updateRegions

- вызывается из MainActivity при запуске приложения и при изменении набора регионов
- получение списка отмеченных пользователем регионов

```
// Обновление выбранных регионов по данным из SharedPreferences
public void updateRegions(SharedPreferences sharedPreferences) {
    regionsSet =
            sharedPreferences.getStringSet(MainActivity.REGIONS, null);
}
```

Класс MainActivityFragment

Метод resetQuiz

- настраивает и запускает викторину
- получение имён файлов изображений
- регистрация в журнале в случае ошибки

```
// Настройка и  запуск следующей серии вопросов
public void resetQuiz() {
    // Использование AssetManager для получения имен файлов изображений
    AssetManager assets = getActivity().getAssets();
    fileNameList.clear(); // Пустой список имен файлов изображений

    try {
        // Перебрать все регионы
        for (String region : regionsSet) {
            // Получить список всех имён файлов изображений флагов
            // для данного региона
            String[] paths = assets.list(region);

            for (String path : paths)
                fileNameList.add(path.replace(".png", ""));
        }
    }
    catch (IOException exception) {
        Log.e(TAG, "Error loading image file names", exception);
    }
```

Класс MainActivityFragment

Метод resetQuiz

- обнуление счётчиков попыток (правильных и общего количества)

```
correctAnswers = 0; // Сброс количества правильных ответов
totalGuesses = 0; // Сброс общего количества попыток
quizCountriesList.clear(); // Очистка предыдущего списка стран
```

Класс MainActivityFragment

Метод resetQuiz

- формирование случайного списка флагов
- загрузка первого флага (запуск викторины)

```
    int flagCounter = 1;
    int numberOfFlags = fileNameList.size();

    // Добавление FLAGS_IN_QUIZ случайных файлов в  quizCountriesList
    while (flagCounter <= FLAGS_IN_QUIZ) {
        int randomIndex = random.nextInt(numberOfFlags);

        // Получение случайного имени файла
        String filename = fileNameList.get(randomIndex);

        // Если регион включен, но еще не был выбран
        if (!quizCountriesList.contains(filename)) {
            quizCountriesList.add(filename); // Добавить файл в  список
            ++flagCounter;
        }
    }

    loadNextFlag(); // Запустить викторину загрузкой первого флага
}
```

Класс MainActivityFragment

Метод loadNextFlag

- загружает и отображает следующий флаг и варианты ответа
- в списке quizCountriesList хранятся имена файлов в формате *регион-страна* без расширения .png; если *регион* или *страна* содержат несколько слов, то слова разделяются символом подчеркивания (_)
- получение следующего флага и удаление его из списка
- обновление правильного ответа

```
// Загрузка следующего флага после правильного ответа
private void loadNextFlag() {
    // Получение имени файла следующего флага и  удаление его из списка
    String nextImage = quizCountriesList.remove(0);
    correctAnswer = nextImage; // Обновление правильного ответа
    answerTextView.setText(""); // Очистка answerTextView

    // Отображение номера текущего вопроса
    questionNumberTextView.setText(getString(
            R.string.question, (correctAnswers + 1), FLAGS_IN_QUIZ));
```

Класс MainActivityFragment

Метод loadNextFlag

- загрузка изображения
- загрузка флага из ресурса с контролем исключений

```
// Извлечение региона из имени следующего изображения
String region = nextImage.substring(0, nextImage.indexOf('-'));

// Использование AssetManager для загрузки следующего изображения
AssetManager assets = getActivity().getAssets();

// Получение объекта InputStream для ресурса следующего флага
// и  попытка использования InputStream
try (InputStream stream =
             assets.open(region + "/" + nextImage + ".png")) {
    // Загрузка графики в  виде объекта Drawable и  вывод на flagImageVie
    Drawable flag = Drawable.createFromStream(stream, nextImage);
    flagImageView.setImageDrawable(flag);

    animate(false); // Анимация появления флага на экране
}
catch (IOException exception) {
    Log.e(TAG, "Error loading " + nextImage, exception);
}
```

Класс MainActivityFragment

Метод loadNextFlag

- перестановка списка имён файлов в случайном порядке
- поиск правильного ответа correctAnswer и перемещение его в конец fileNameList — позднее этот ответ будет случайно помещен на одну из кнопок с вариантами

```
Collections.shuffle(fileNameList); // Перестановка имен файлов

// Помещение правильного ответа в  конец fileNameList
int correct = fileNameList.indexOf(correctAnswer);
fileNameList.add(fileNameList.remove(correct));
```

Класс MainActivityFragment

Метод loadNextFlag

- добавление кнопок с названиями стран

```
// Добавление 2, 4, 6 или 8 кнопок в  зависимости от значения guessRows
for (int row = 0; row < guessRows; row++) {
    // Размещение кнопок в  currentTableRow
    for (int column = 0;
         column < guessLinearLayouts[row].getChildCount();
         column++) {
        // Получение ссылки на Button
        Button newGuessButton =
                (Button) guessLinearLayouts[row].getChildAt(column);
        newGuessButton.setEnabled(true);

        // Назначение названия страны текстом newGuessButton
        String filename = fileNameList.get((row * 2) + column);
        newGuessButton.setText(getCountryName(filename));
    }
}
```

Класс MainActivityFragment

Метод loadNextFlag

- размещение правильного ответа на одну из кнопок случайным образом

```
    // Случайная замена одной кнопки правильным ответом
    int row = random.nextInt(guessRows); // Выбор случайной строки
    int column = random.nextInt(2); // Выбор случайного столбца
    LinearLayout randomRow = guessLinearLayouts[row]; // Получение строки
    String countryName = getCountryName(correctAnswer);
    ((Button) randomRow.getChildAt(column)).setText(countryName);
}
```

Метод getCountryName

- выделяет название страны из имени файла с изображением

```
// Извлекает имя файла с флагом и возвращает название страны
private String getCountryName(String name) {
    return name.substring(name.indexOf('-') + 1).replace('_', ' ');
}
```

Класс MainActivityFragment

Метод animate

- **применяет анимацию кругового раскрытия со всем макетом (quizLinearLayout) при переходе к следующему вопросу**

```
// Весь макет quizLinearLayout появляется или исчезает с  экрана
private void animate(boolean animateOut) {
    // Предотвращение анимации интерфейса для первого флага
    if (correctAnswers == 0)
        return;

    // Вычисление координат центра
    int centerX = (quizLinearLayout.getLeft() +
            quizLinearLayout.getRight()) / 2; // calculate center x
    int centerY = (quizLinearLayout.getTop() +
            quizLinearLayout.getBottom()) / 2; // calculate center y

    // Вычисление радиуса анимации
    int radius = Math.max(quizLinearLayout.getWidth(),
            quizLinearLayout.getHeight());
```

Класс MainActivityFragment

Метод animate

- исчезновение quizLinearLayout с экрана (animateOut==true)

```
Animator animator;

// Если изображение должно исчезать с  экрана
if (animateOut) {
    // Создание круговой анимации
    animator = ViewAnimationUtils.createCircularReveal(
            quizLinearLayout, centerX, centerY, radius, 0);
    animator.addListener(
            new AnimatorListenerAdapter() {
                // Вызывается при завершении анимации
                @Override
                public void onAnimationEnd(Animator animation) {
                    loadNextFlag();
                }
            }
    );
}
```

Класс MainActivityFragment

Метод animate

- анимация появления quizLinearLayout на экране вначале следующего вопроса (animateOut==false)
- задание продолжительности анимации
- запуск анимации

```
    else { // Если макет quizLinearLayout должен появиться
        animator = ViewAnimationUtils.createCircularReveal(
                quizLinearLayout, centerX, centerY, 0, radius);
    }

    animator.setDuration(500); // Анимация продолжительностью 500 мс
    animator.start(); // Начало анимации
}
```

Обработчик нажатия кнопки, реализующий интерфейс OnClickListener

- вызывается при нажатии кнопки ответа
- получение ссылки на нажатую кнопку
- определение выбранной страны и правильной страны
- увеличение общего счётчика ответов

```
// Вызывается при нажатии кнопки ответа
private final ThreadLocal<OnClickListener> guessButtonListener =
        new ThreadLocal<OnClickListener>() {
    @Override
    protected OnClickListener initialValue() {
        return new OnClickListener() {
            @Override
            public void onClick(View v) {
                Button guessButton = ((Button) v);
                String guess = guessButton.getText().toString();
                String answer = getCountryName(correctAnswer);
                ++totalGuesses; // Увеличение количества попыток пользователя
```

Обработчик нажатия кнопки, реализующий интерфейс OnClickListener

Ответ правильный!

- увеличение счётчика правильных ответов
- вывод его зелёным цветом
- блокировка всех кнопок во избежание дальнейших ответов

```
if (guess.equals(answer)) { // Если ответ правилен
    ++correctAnswers; // Увеличить количество правильных ответов

    // Правильный ответ выводится зеленым цветом
    answerTextView.setText(answer + "!");
    answerTextView.setTextColor(
            getResources().getColor(R.color.correct_answer,
                    getContext().getTheme()));

    disableButtons(); // Блокировка всех кнопок ответов
```

Обработчик нажатия кнопки, реализующий интерфейс OnClickListener

Ответ правильный и викторина завершена!

- создание диалога вывода статистики

```
// Если пользователь правильно угадал FLAGS_IN_QUIZ флагов
if (correctAnswers == FLAGS_IN_QUIZ) {
    // DialogFragment для вывода статистики и перезапуска
    DialogFragment quizResults = new DialogFragment() {
                // Создание окна AlertDialog
                @Override
                public Dialog onCreateDialog(Bundle bundle) {
                    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
                    builder.setMessage(
                            getString(R.string.results,
                                    totalGuesses,
                                    (1000 / (double) totalGuesses))
                    );
                    // Кнопка сброса "Reset Quiz"
                    builder.setPositiveButton(R.string.reset_quiz, new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog,
                                                    int id) {
                                    resetQuiz();
                                }
                            }
                    );
                    return builder.create(); // Вернуть AlertDialog
                }
            };
```

Обработчик нажатия кнопки, реализующий интерфейс OnClickListener

Ответ правильный и викторина завершена!

- отображение диалога вывода статистики

Ответ правильный и викторина не закончена!

- скрытие текущего флага для загрузки следующего через 2 секунды

```
    // Использование FragmentManager для вывода DialogFragment
    quizResults.setCancelable(false);
    quizResults.show(getFragmentManager(), "quiz results");
} else { // Ответ правильный, но викторина не закончена
    // Загрузка следующего флага после двухсекундной задержки
    handler.postDelayed(
            new Runnable() {
                @Override
                public void run() {
                    animate(true); // Анимация исчезновения флага
                }
            }, 2000); // 2000 миллисекунд для двухсекундной задержки
}
```

Обработчик нажатия кнопки, реализующий интерфейс OnClickListener

Ответ неправильный

- «встряхивание» флага
- сообщение о ошибке цветом R.color.incorrect_answer
- блокировка неправильного ответа

```
        else { // Неправильный ответ
            flagImageView.startAnimation(shakeAnimation); // Встряхивание

            // Сообщение "Incorrect!" выводится красным шрифтом
            answerTextView.setText(R.string.incorrect_answer);
            answerTextView.setTextColor(getResources().getColor(
                    R.color.incorrect_answer, getContext().getTheme()));
            guessButton.setEnabled(false); // Блокировка неправильного ответа
        }
};
```

Класс MainActivityFragment

Метод disableButtons

- вызывается при правильном ответе
- перебирает кнопки с вариантами ответов и блокирует их

```
    // Вспомогательный метод, блокирующий все кнопки
    private void disableButtons() {
        for (int row = 0; row < guessRows; row++) {
            LinearLayout guessRow = guessLinearLayouts[row];
            for (int i = 0; i < guessRow.getChildCount(); i++)
                guessRow.getChildAt(i).setEnabled(false);
        }
    }
}
```

Класс SettingsActivity

- управляет фрагментом настроек при портретной ориентации
- заполняет интерфейс, отображает Toolbar, добавляет

```
package com.somewhere.l3flagquiz;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

public class SettingsActivity extends AppCompatActivity {
    // Заполняет GUI, отображает Toolbar и  добавляет кнопку "up"
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }
}
```

Класс SettingsActivityFragment

- управление настройками приложения
- заполняет интерфейс на базе preferences.xml
- настройки автоматически сохраняются в файле SharedPreferences на устройстве
- если файл не существует, он создается; в противном случае он обновляется

```
package com.somewhere.l3flagquiz;

import android.os.Bundle;
import android.preference.PreferenceFragment;

public class SettingsActivityFragment extends PreferenceFragment {
    // Создание GUI настроек по файлу preferences.xml из res/xml
    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        addPreferencesFromResource(R.xml.preferences); // Загрузить из XML
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.somewhere.l3flagquiz">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="L3 Flag Quiz"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="L3 Flag Quiz"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

AndroidManifest.xml

```
        <activity
            android:name=".SettingsActivity"
            android:label="Settings"
            android:parentActivityName=".MainActivity"
            android:theme="@style/AppTheme.NoActionBar">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.somewhere.l3flagquiz.MainActivity" />
        </activity>
    </application>

</manifest>
```

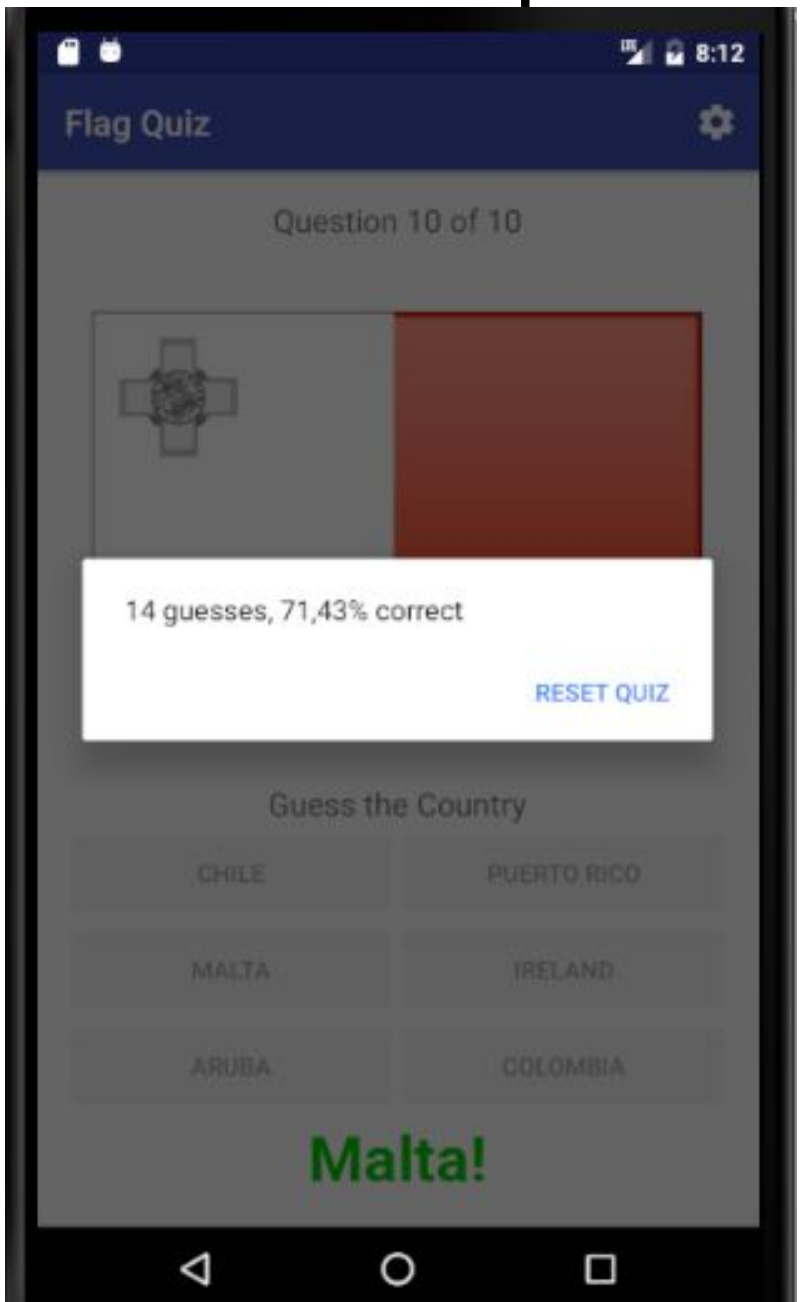
Файл сгенерирован автоматически

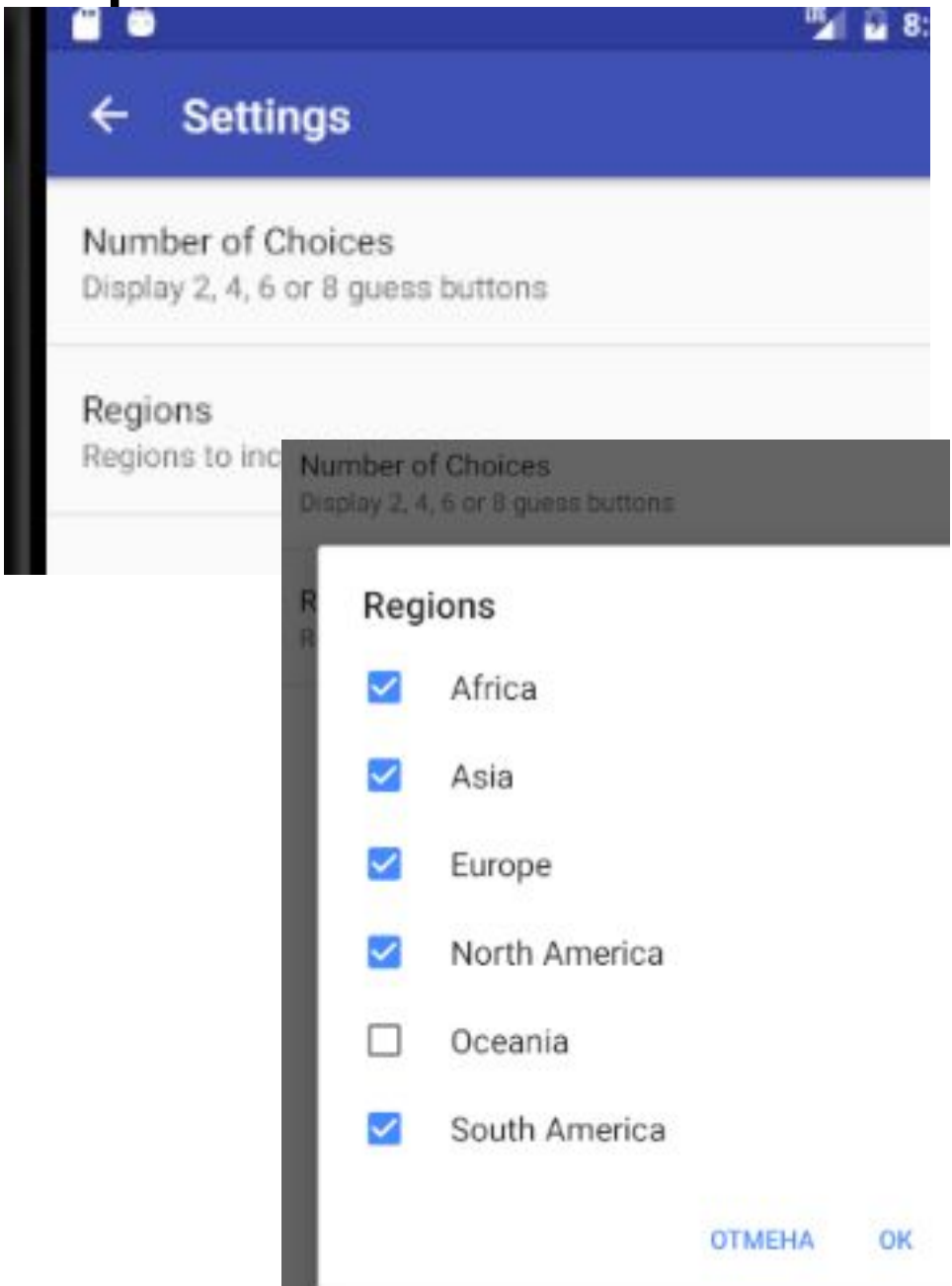
Тестирование приложения

Тестирование приложения

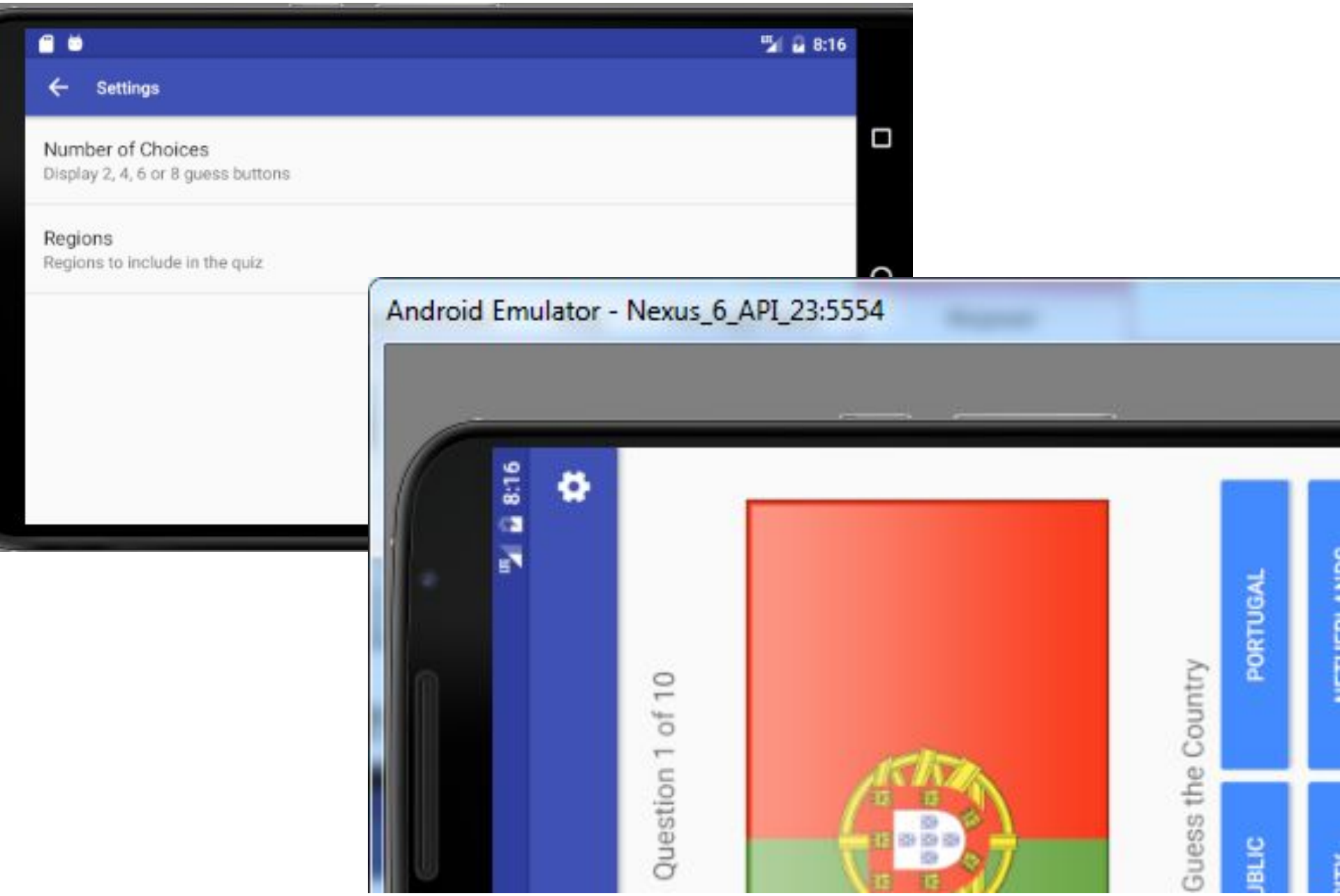
Тестирование приложения

Тестирование приложения

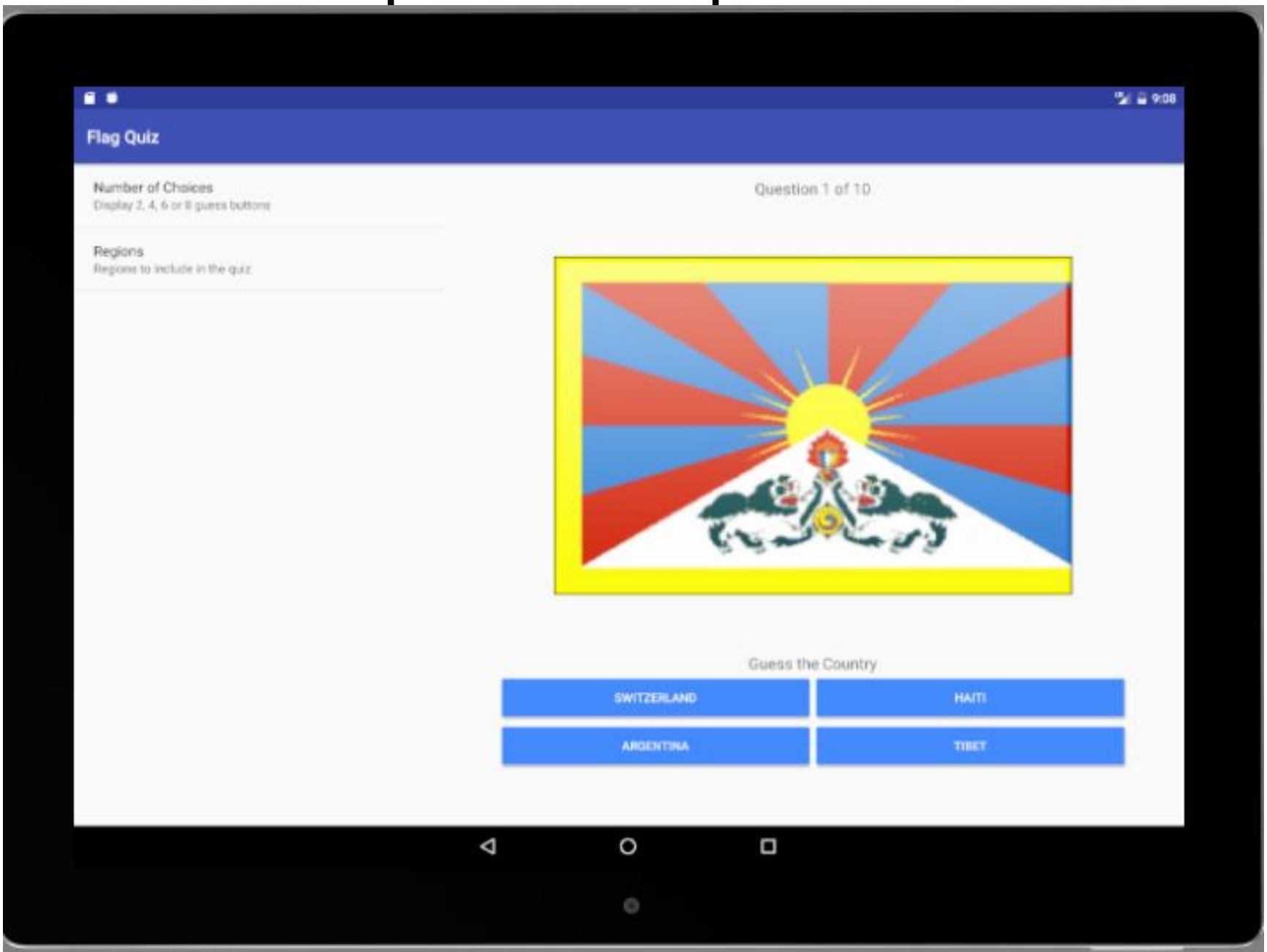




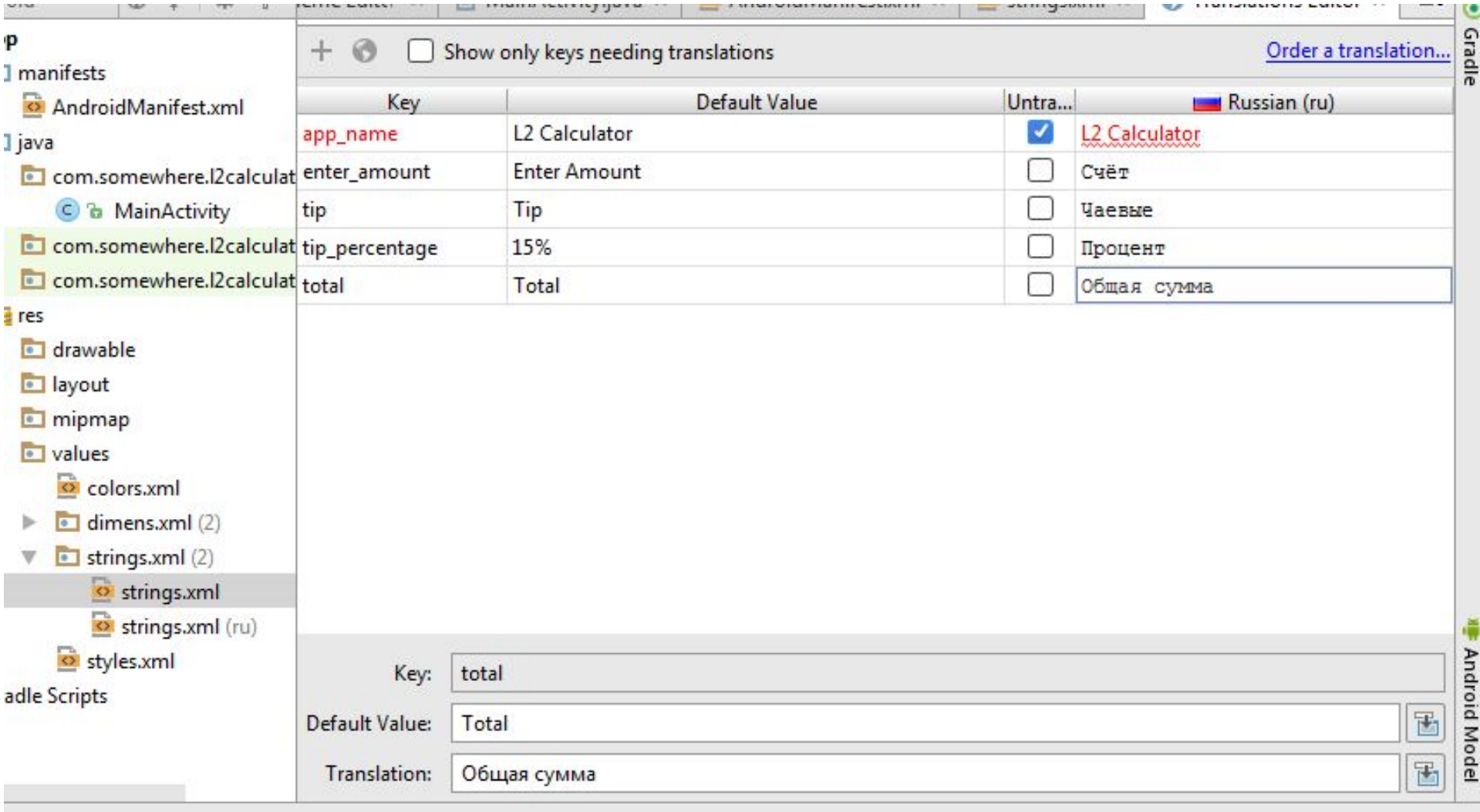
Тестирование приложения



Тестирование приложения



Локализация



Локализация