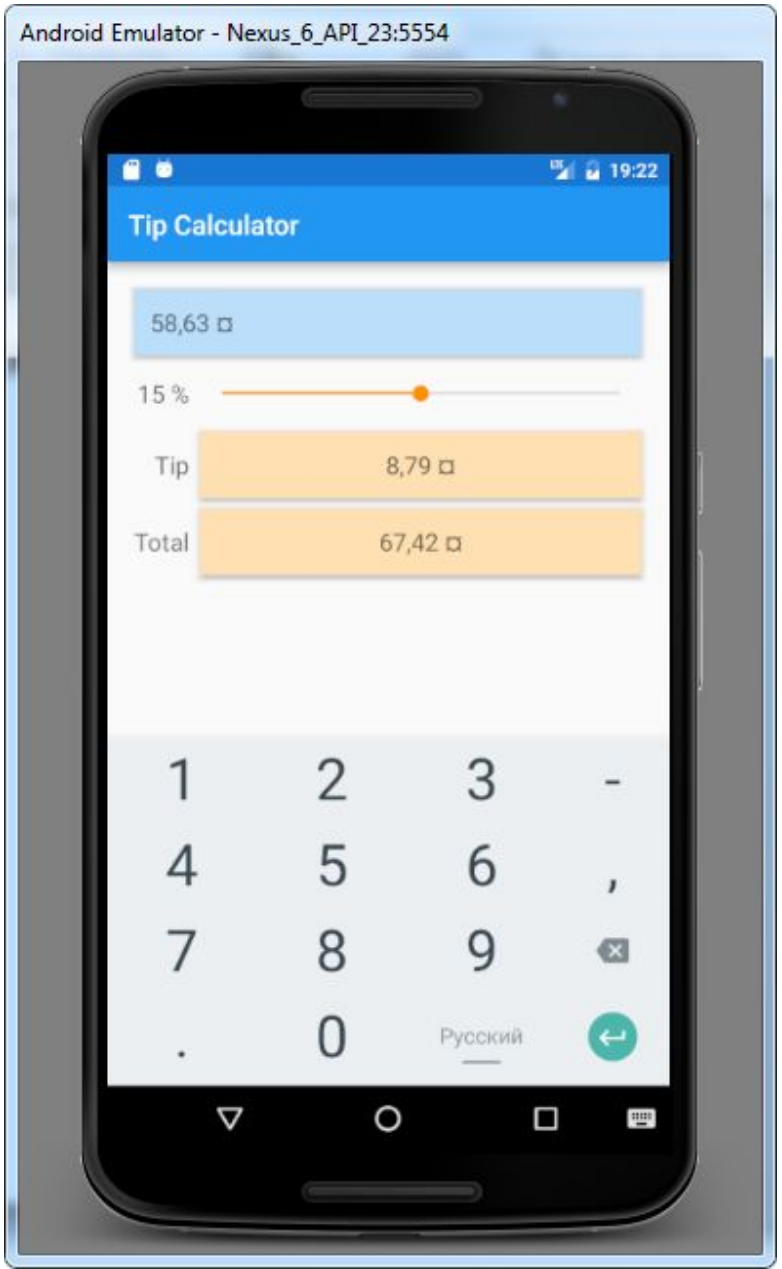
Android 6
Проект с базовой интерактивностью

Рассматриваемые вопросы

- Изменение темы графического интерфейса по умолчанию
- Настройка цветов темы
- Построение графического интерфейса с использованием компонента GridLayout
- Использование окна Component Tree для добавления представлений в GridLayout.
- Использование компонентов TextView, EditText и SeekBar
- Применение объектно-ориентированных возможностей Java, включая классы, объекты, интерфейсы, анонимные внутренние классы и наследование, для расширения функциональности приложений Android
- Программное изменение текста в TextView
- Использование обработки событий при взаимодействии с пользователем с помощью компонентов EditText и SeekBar
- Отображение виртуальной клавиатуры по умолчанию при выполнении приложения
- Ограничение приложения портретной ориентацией

Целевое приложение



Особенности

- приложения Android не содержат метода main()
- используются 4 типа исполняемых компонентов:
  - активности (activities, субклассы Activity, android.app)
  - службы (services)
  - провайдеры контента (content providers)
  - широковещательные приёмники (broadcast receivers)
- приложение может иметь много активностей
- пользователи взаимодействуют с активностями через представления (views, субклассы View, android.view)
- активность может управлять несколькими фрагментами (телефон: 1 экран = 1 фрагмент, планшет: 1 экран = несколько фрагментов)

Методы жизненного цикла активности

- **«Активная активность»** отображается на экране и «обладает фокусом» — то есть взаимодействует с пользователем.
- **Приостановленная активность** видна на экране, но *не обладает* фокусом (например, на время отображения диалогового окна с сообщением). Пользователь не может взаимодействовать с приостановленной активностью.
- **Остановленная активность** *не отображается* на экране и, вероятно, будет уничтожена системой, когда потребуется освободить занимаемую ею память. Активность *останавливается*, когда другая активность переходит в *активное* состояние. Например, остановка приложения при ответе на телефонный звонок.

Методы жизненного цикла активности

При переходах активности между состояниями исполнительная среда Android вызывает различные *методы жизненного цикла* (все эти методы определяются в классе Activity из пакета android.app).

Для *каждой* активности обычно переопределяется метод **onCreate**. Этот метод вызывается исполнительной системой Android при запуске активности — то есть когда ее графический интерфейс готов к отображению, чтобы пользователь мог взаимодействовать с активностью.

Другие методы жизненного цикла:

- onStart
- onPause
- onRestart
- onResume
- onStop
- onDestroy

Каждый переопределяемый метод жизненного цикла активности ***должен*** вызывать версию метода из суперкласса; в противном случае происходит исключение. Вызов версии суперкласса необходим, потому что каждый метод жизненного цикла в суперклассе Activity содержит код, который должен выполняться помимо кода, определяемого в переопределенных методах жизненного цикла.

Используемые компоненты

- GridLayout (размещение компонентов в ячейках прямоугольной таблицы, API14 и выше)
- TextView (см. предыдущую презентацию)
- EditText (субкласс TextView для вывода и ввода текста пользователем; позволяет ограничить диапазон ввода)
- SeekBar (целое число, настраиваемое интерфейсным элементом «ползунок»; диапазон по умолчанию: 0-100)
- класс NumberFormat (для форматирования чисел в соответствии с локальным контекстом)
- интерфейс TextWatcher (для реакции на события при изменении текста)
- интерфейс SeekBar.OnSeekBarChangeListener (для обработки перемещений ползунка)

Материальный дизайн

Тема приводит внешний вид приложения в соответствие с визуальным стилем Android. В проектах для Android5 и выше используются темы, отвечающие правилам материального дизайна Google.

Предопределённые темы:

- Тема «light» с белой панелью приложения, белым фоном и текстом черного или темно-серого цвета.
- Тема «light» с темной панелью приложения отличается от предыдущей тем, что панель приложения окрашена в черный цвет, а цвет по умолчанию выводится белым шрифтом.
- Тема «dark» с черной панелью приложения, темно-серым фоном и текстом белого или светло-серого цвета.

Каждая из тем существует в двух версиях:

- Theme.Material (например, Theme.Material.Light) для приложений, которые не используют библиотеки AppCompat и работают в Android 5 и выше
- Theme.AppCompat (например, Theme.AppCompat.Light) для приложений, которые используют библиотеки AppCompat и работают в Android 2.1 и выше.

Материальный дизайн

Рельеф и тени.

- Когда для представления задается свойство elevation, Android автоматически генерирует тень от этого представления.
- Чем больше значение elevation, тем четче выражена тень. Для диалоговых окон рекомендуемое значение elevation равно 24dp, для меню— 8dp.
- Другие рекомендации: *http://www.google.com/design/spec/what-is-material/elevation-shadows.html*

Цвета.

- Рекомендации материального дизайна Google советуют выбрать цветовую палитру, состоящую из основного цвета (не более чем с тремя оттенками) и акцентного цвета.
- Основные цвета обычно используются для окрашивания строки состояния и панели приложения в верхней части экрана; кроме того, они могут использоваться в графическом интерфейсе.
- Акцентный цвет предназначен для выделения оттенков в различных представлениях— таких, как SeekBar, CheckBox и RadioButton.
- Рекомендации по выбору цветов: *http://www.materialpalette.com/*

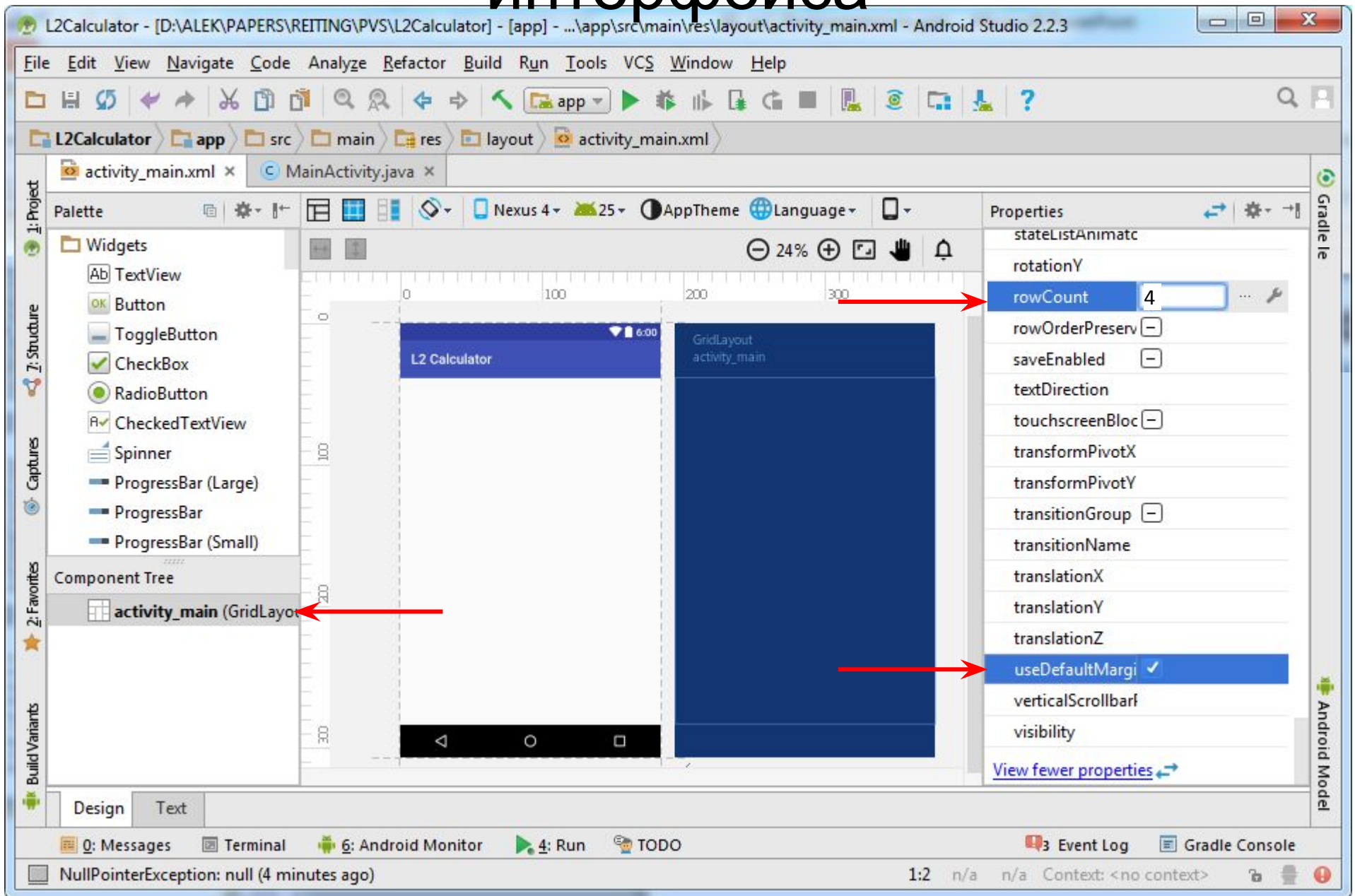
AndroidManifest.xml

- Файл AndroidManifest.xml генерируется средой разработки при создании нового проекта приложения.
- В файле хранятся настройки, вводимые в диалоговом окне Create New Project: имя приложения, имя пакета, имя класса активности (активностей) и т. д.
- В примере отредактируем разметку XML этого файла и добавим в неё новый параметр, включающий постоянное отображение виртуальной клавиатуры на экране.
- Будет указано, что приложение поддерживает только *портретную ориентацию* (то есть вертикальной является более длинная сторона).

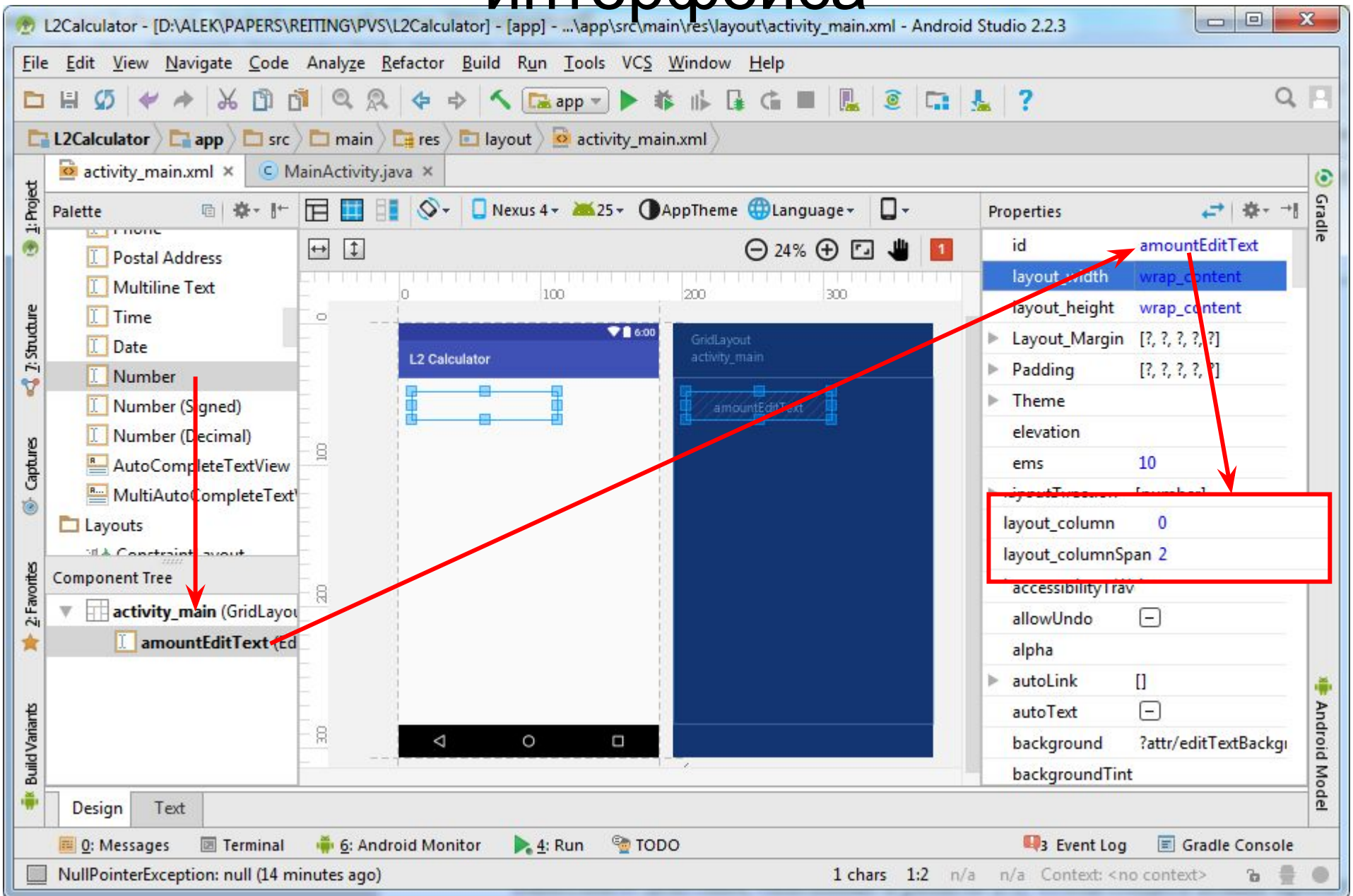
Создание проекта

- Имя проекта: L2Calculator
- Android 6, API 23
- Шаблон: Empty Activity
- Добавить значок в проект
- Удалить компонент TextView с текстом «Hello World!»
- Заменить android.support.constraint.ConstraintLayout на GridLayout
  (в файле activity_main.xml, режим «Text»)
- Вернуться в режим «Design»

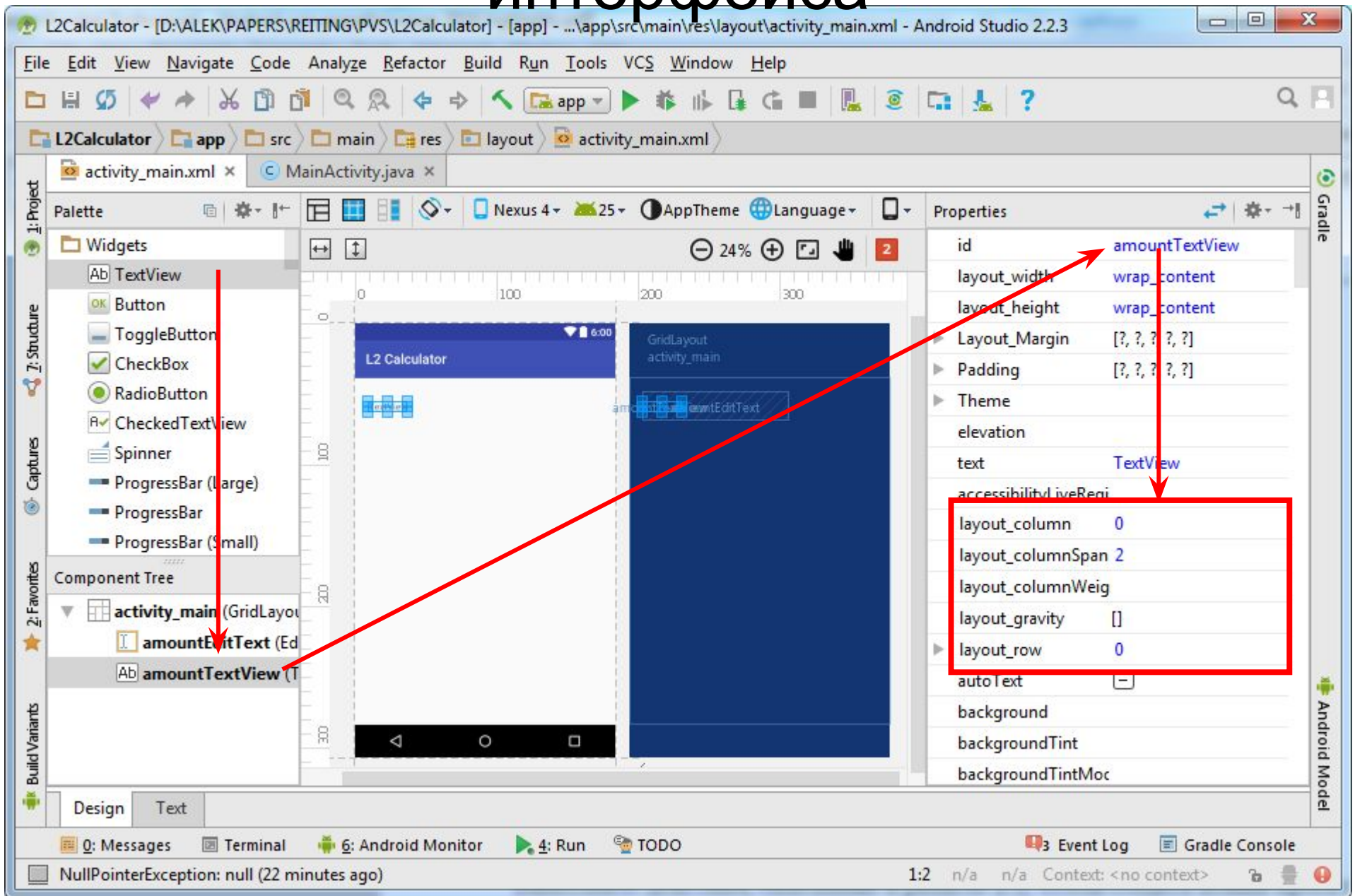
Построение графического интерфейса



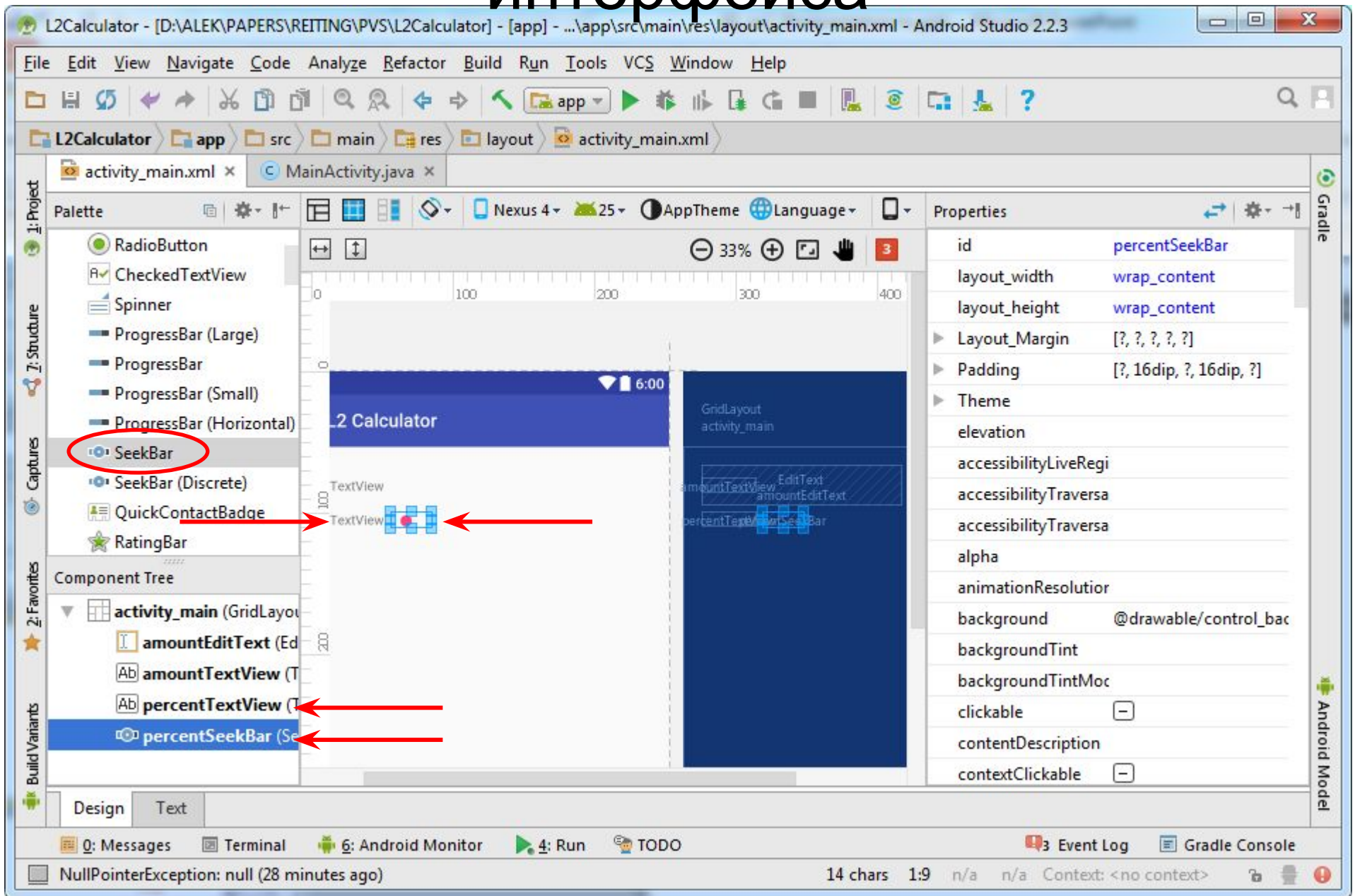
Построение графического интерфейса



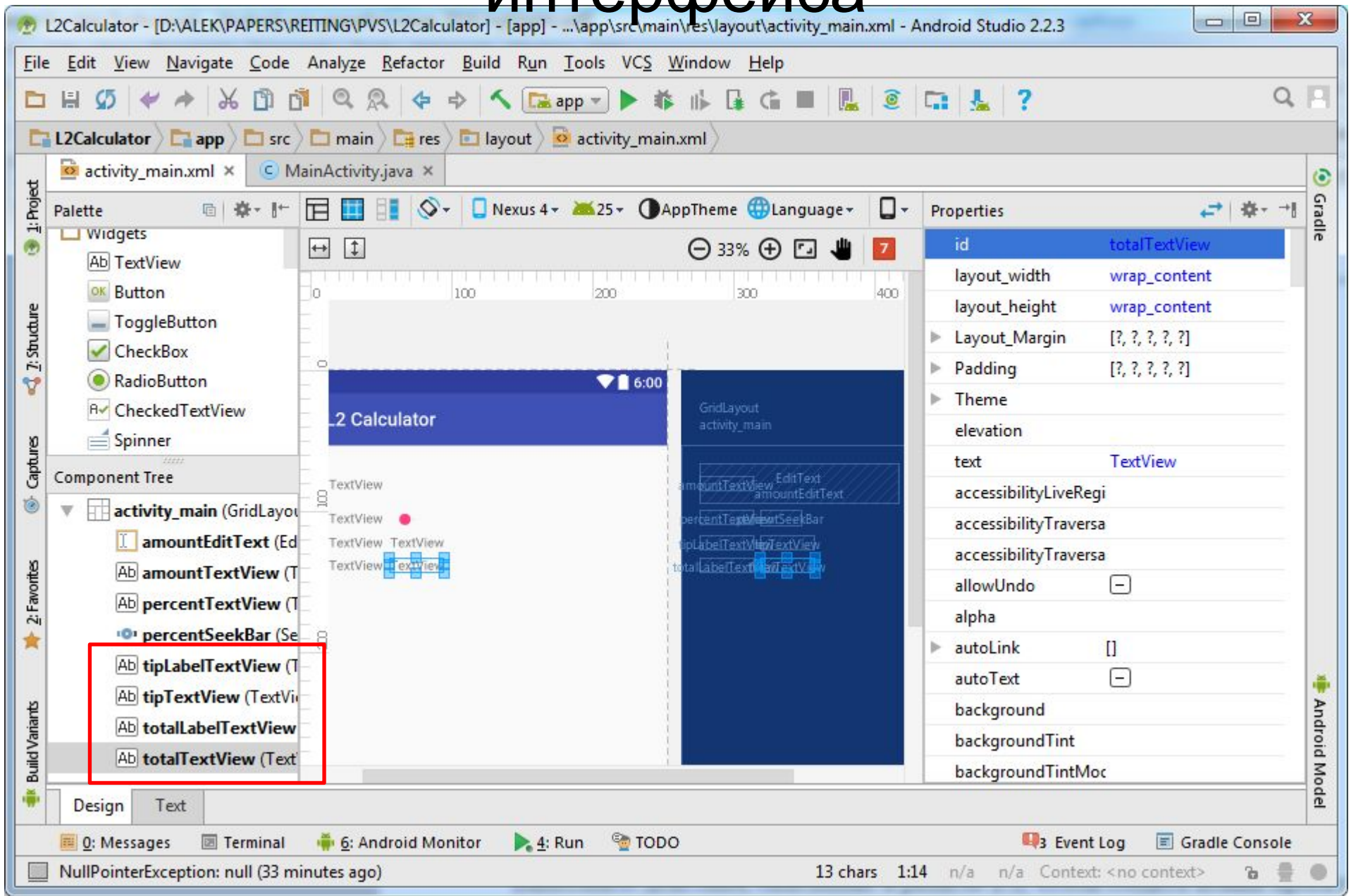
Построение графического интерфейса



Построение графического интерфейса



Построение графического интерфейса

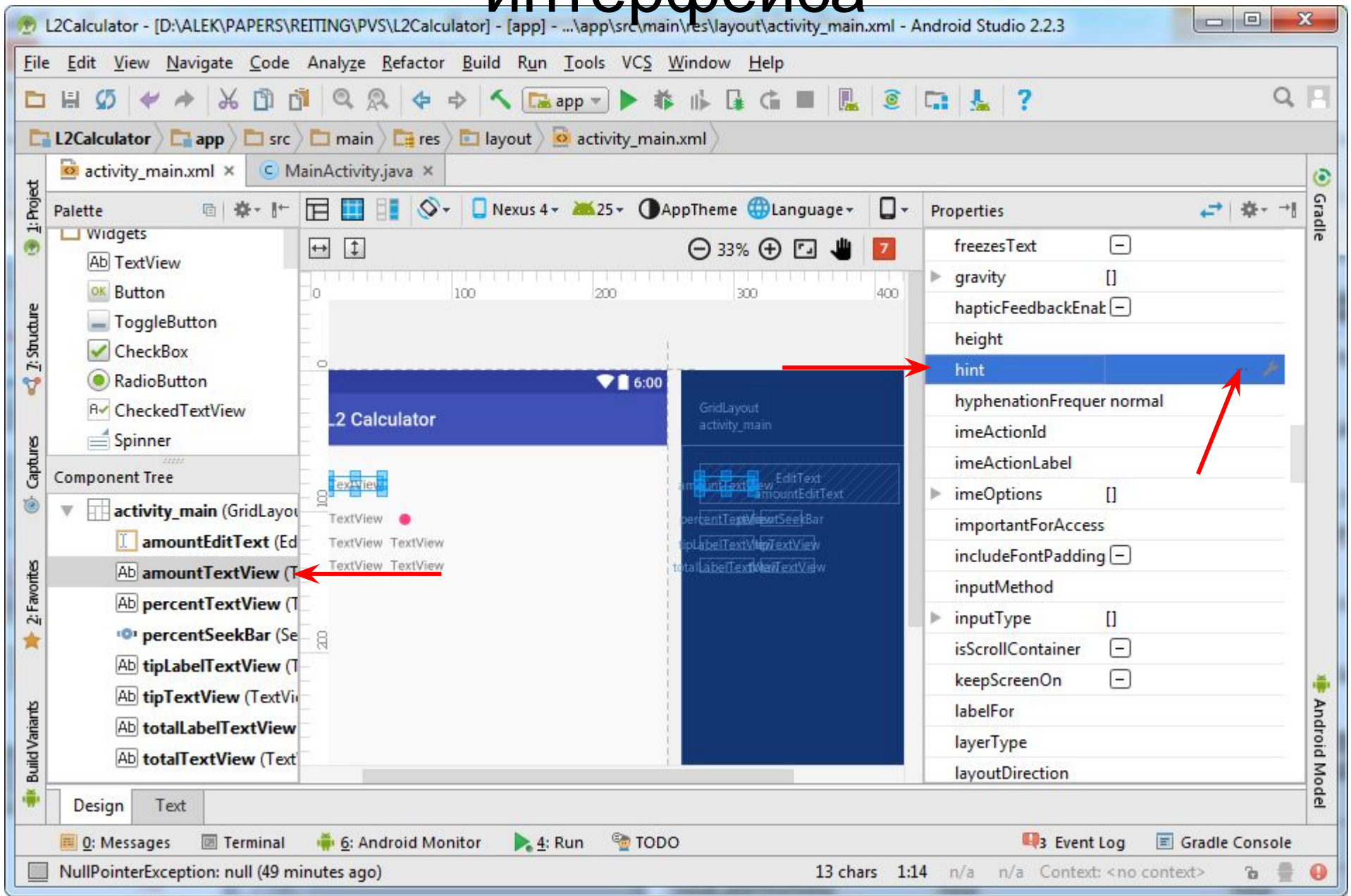


Построение графического интерфейса

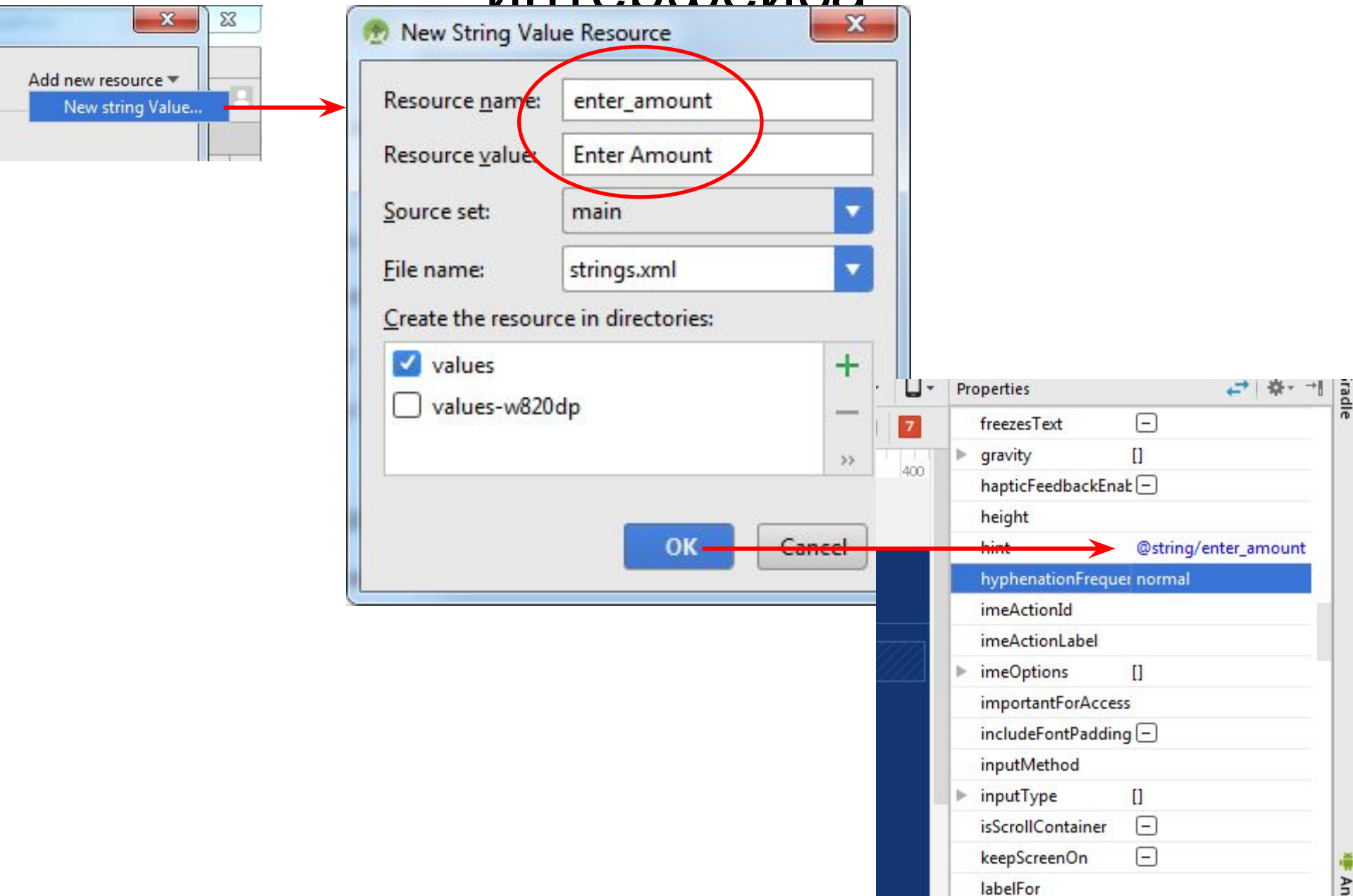
- настроим свойства компонентов и создадим несколько ресурсов для строк и метрик
- создадим несколько ресурсов со строками, размерами и цветами
- если свойство text пусто, выводится значение свойства hint
  - зададим hint для amountTextView, percentTextView, tipLabelTextView, totalLabelTextView
  - свойство hint задаётся через создание ресурсов строковых значений
  - для amountTextView показано на следующих слайдах, для остальных – см. таблицу ниже

Компонент	Имя ресурса	Значение ресурса
percentTextView	tip_percentage	15%
tipLabelTextView	tip	Tip
totalLabelTextView	total	Total

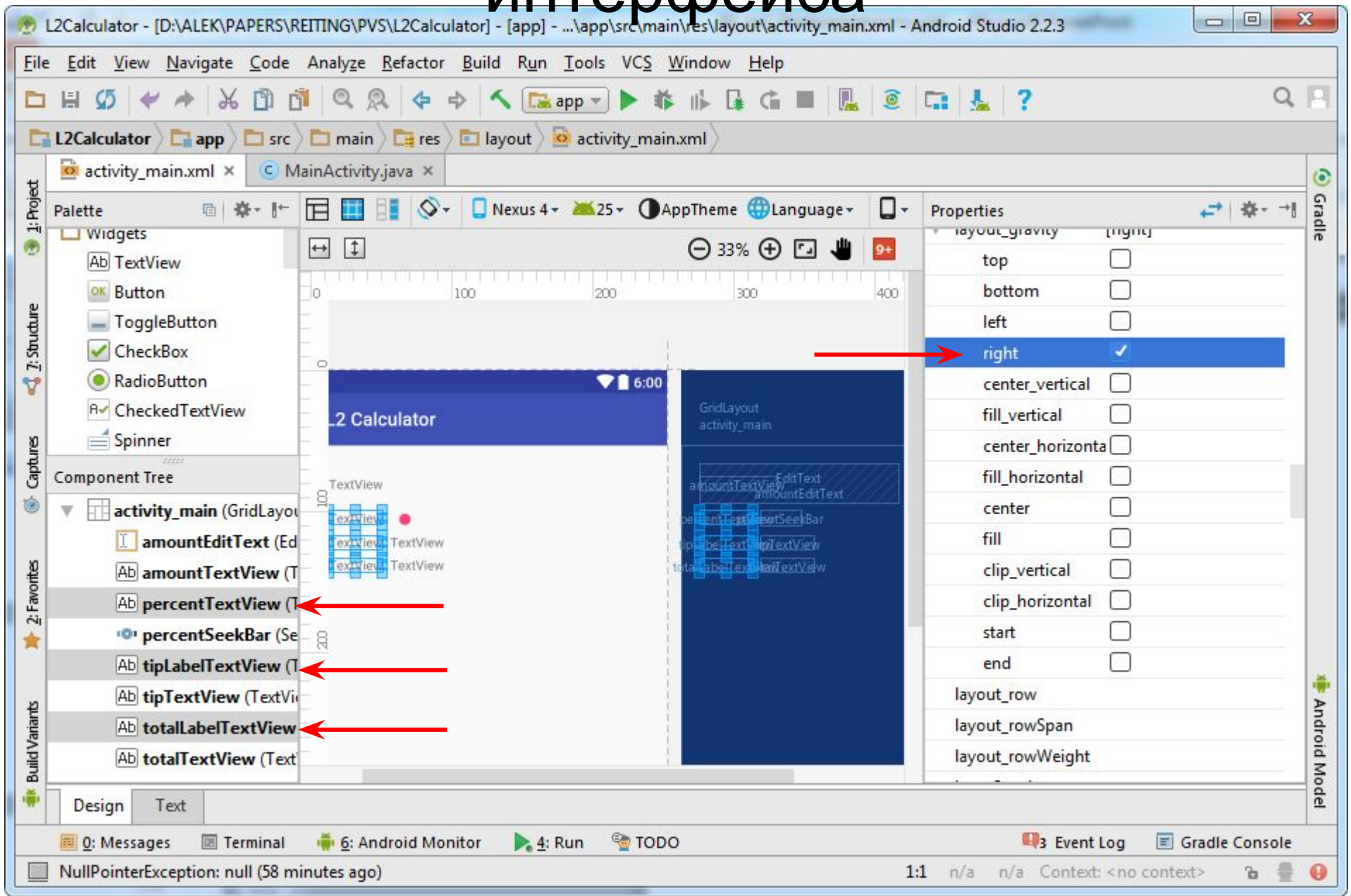
Построение графического интерфейса



Построение графического интерфейса



Построение графического интерфейса



Построение графического интерфейса

- разрешаем вводить только цифры
  amountEditText.digits=0123456789

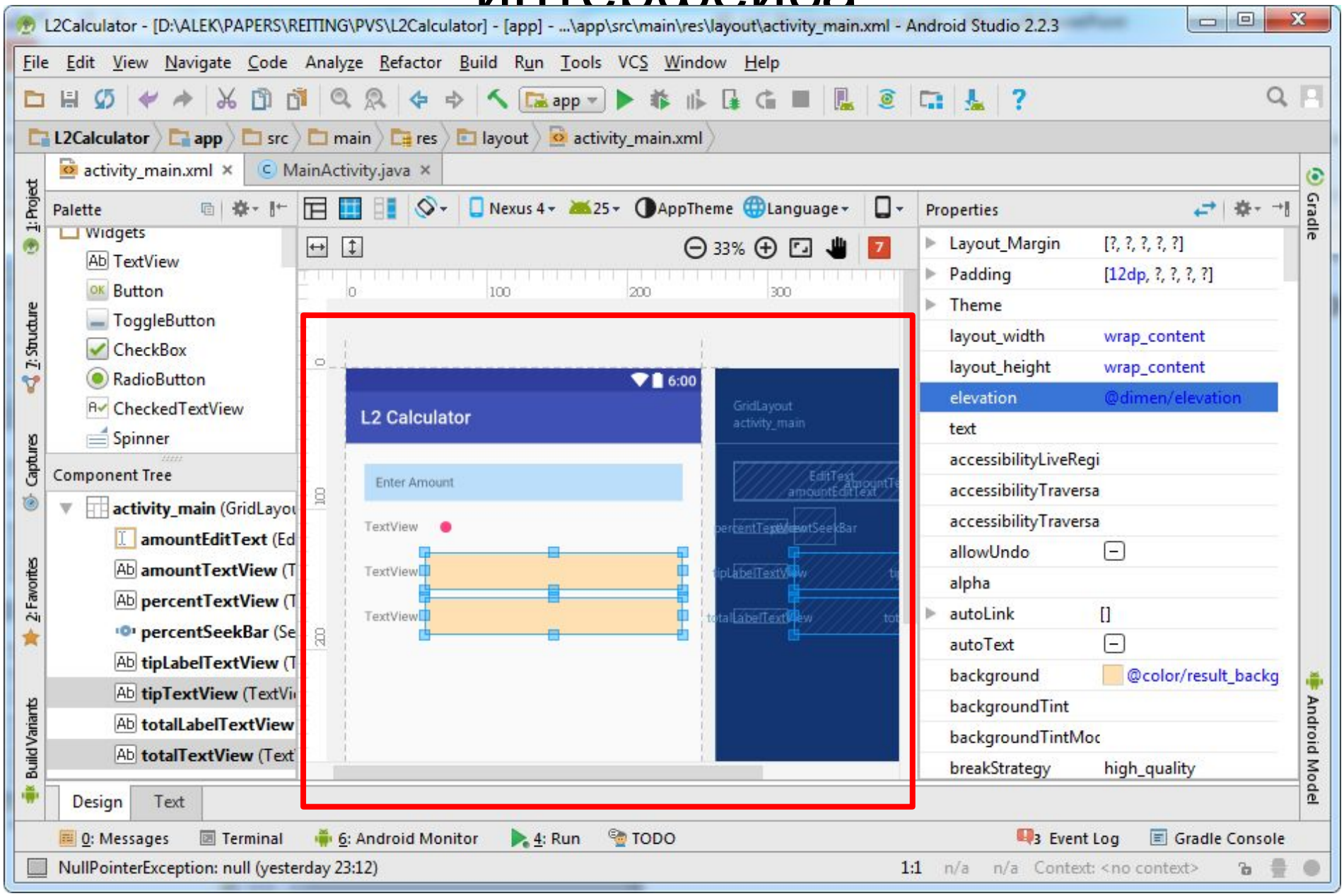
- максимальная длина суммы счёта
  amountEditText.maxLength=6

- настраиваем amountTextView
  .text="" (удаляем текст по умолчанию)
  .layout_gravity.fill_horizontal=true
  .background=@color/amount_background (#BBDEFB)
  .Padding.all=@dimen/textview_padding (12dp)
  .elevation=@dimen/elevation (4dp)

Построение графического интерфейса

- настраиваем percentTextView
  .layout_gravity.center_vertical=true
- настраиваем percentSeekBar
  .max=30 (максимальное значение)
  .progress=15 (значение по умолчанию)
  .layout_gravity.fill_horizontal=true
  .layout_height=@dimen/seekbar_height (40dp)
- настраиваем tipTextView и totalTextView
  .text="" (удаляем текст по умолчанию)
  .layout_gravity.fill_horizontal=true
  .background=@color/result_background (#FFE0B2)
  .gravity.center=true
  .Padding.all=@dimen/textview_padding (12dp)
  .elevation=@dimen/elevation (4dp)

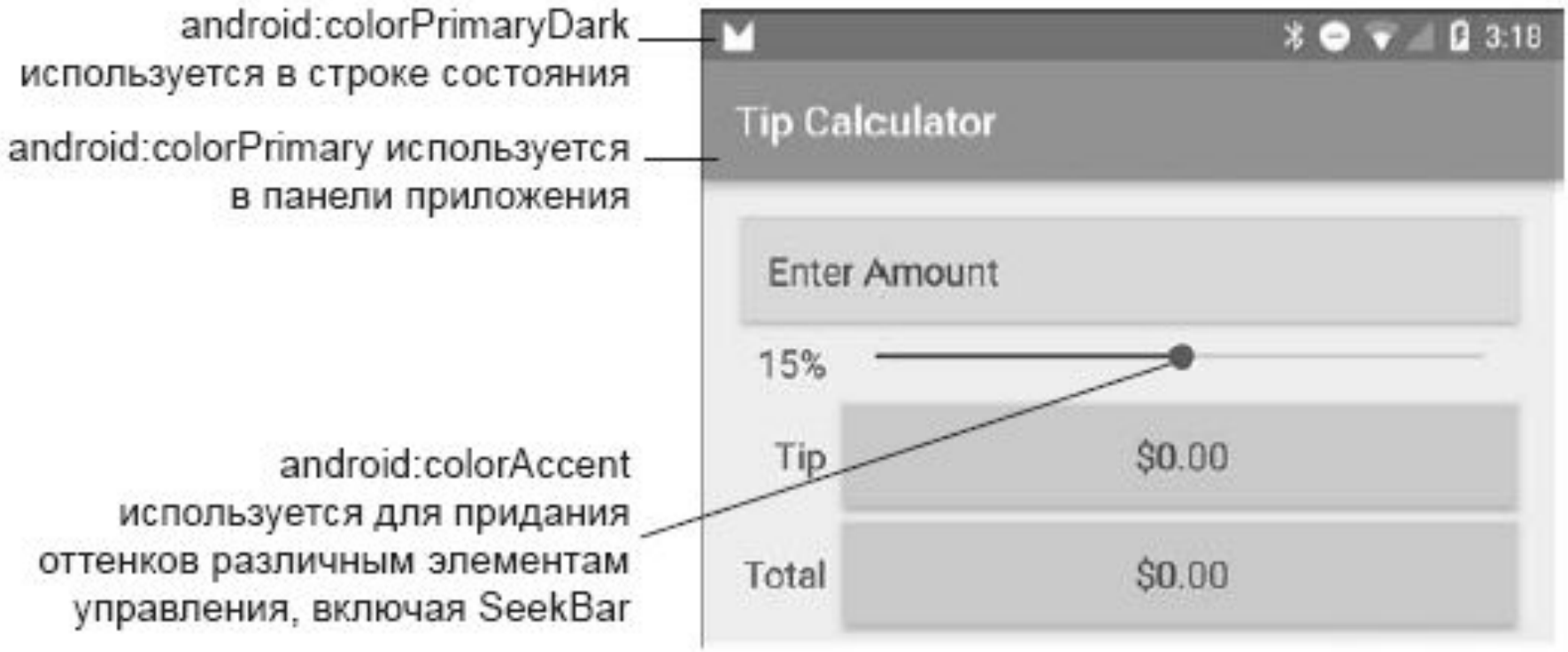
Построение графического интерфейса



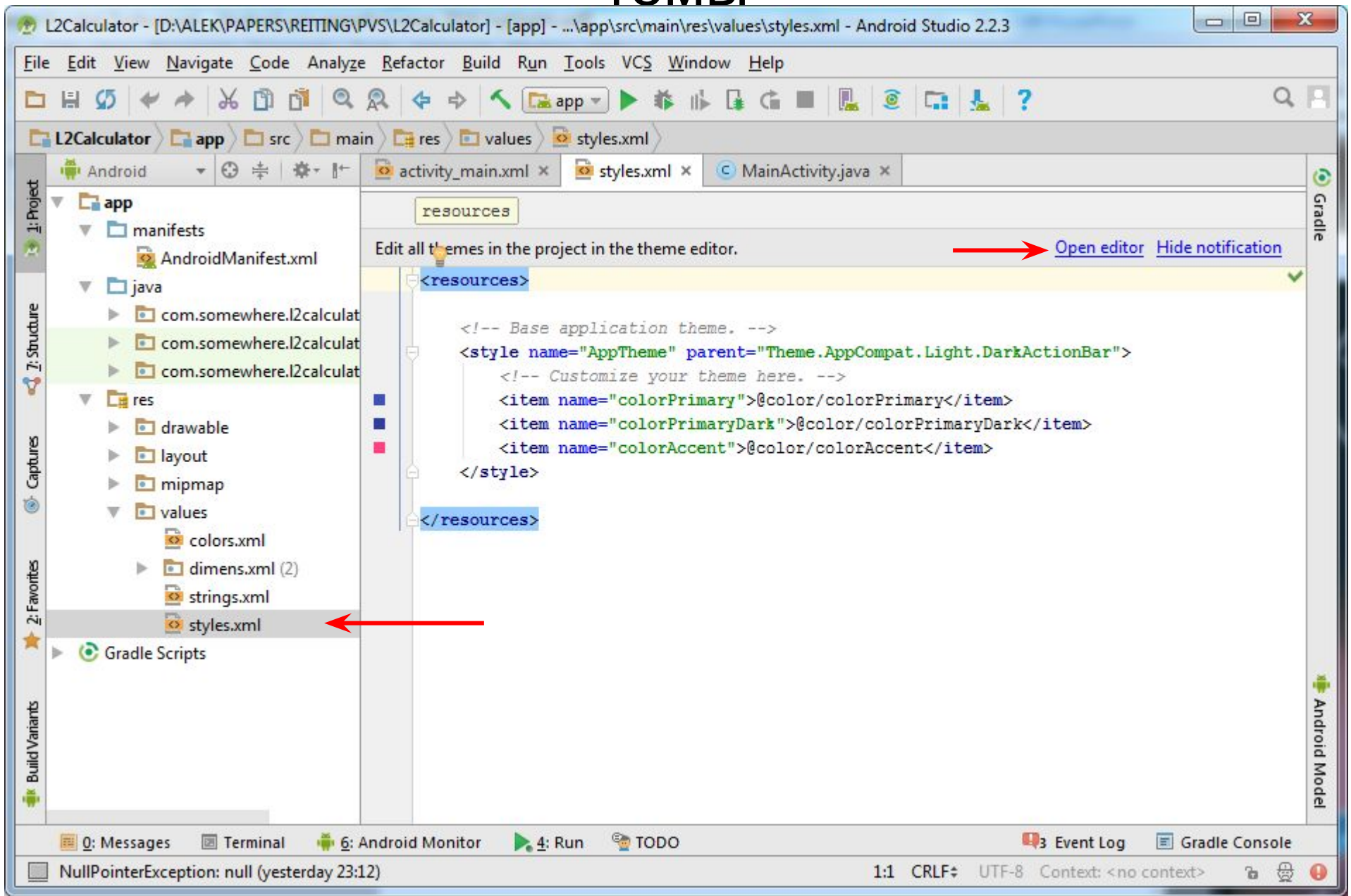
Тема по умолчанию и настройка цветов темы

- У каждого приложения существует тема, определяющая оформление стандартных компонентов, которые используются.
- Тема приложения указывается в файле AndroidManifest.xml приложения.
- Можно настроить различные аспекты темы (например, составляющие цветовой схемы), определяя ресурсы в файле styles.xml, находящемся в папке res/values приложения.

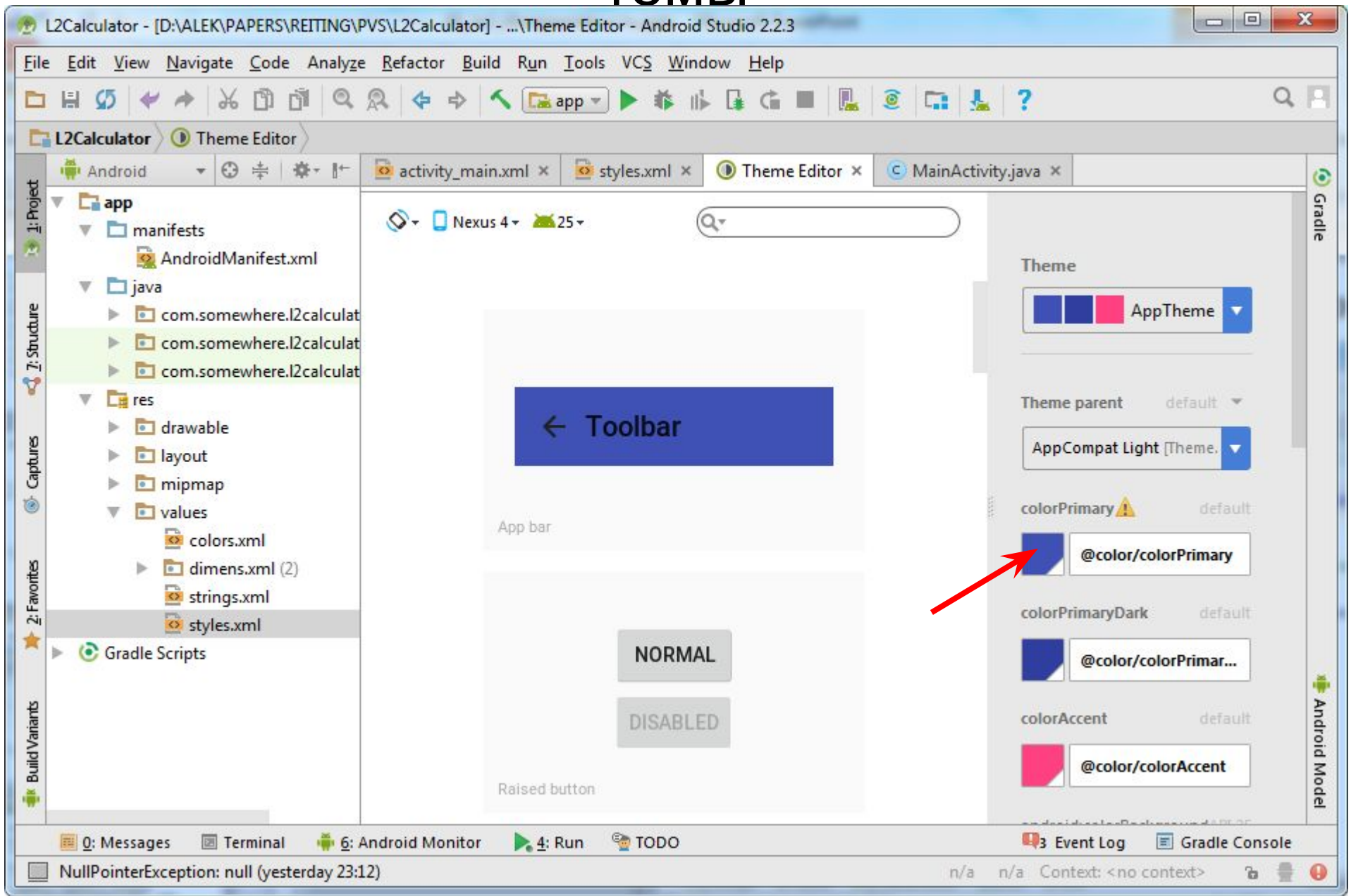
Тема по умолчанию и настройка цветов темы



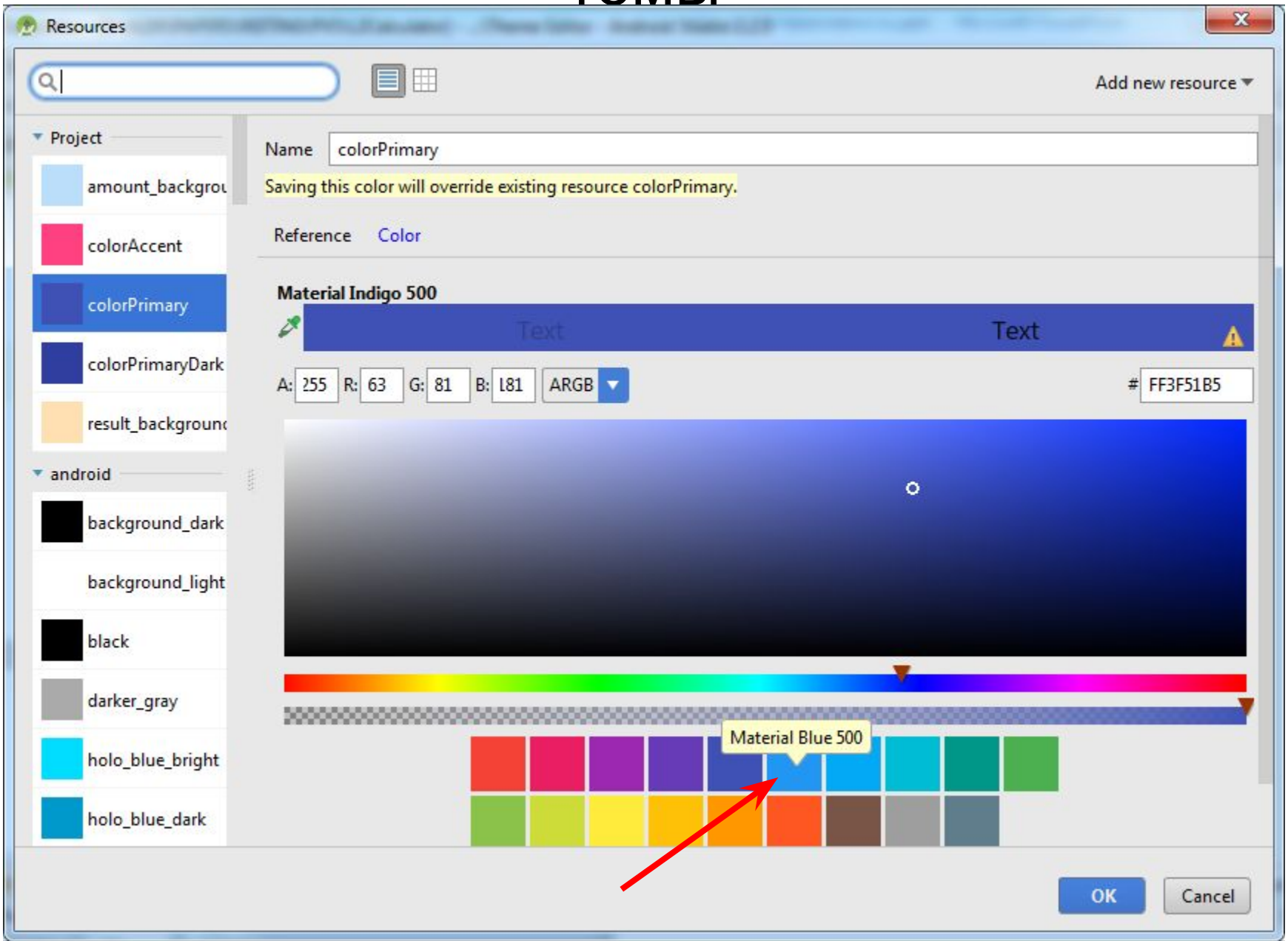
Тема по умолчанию и настройка цветов темы



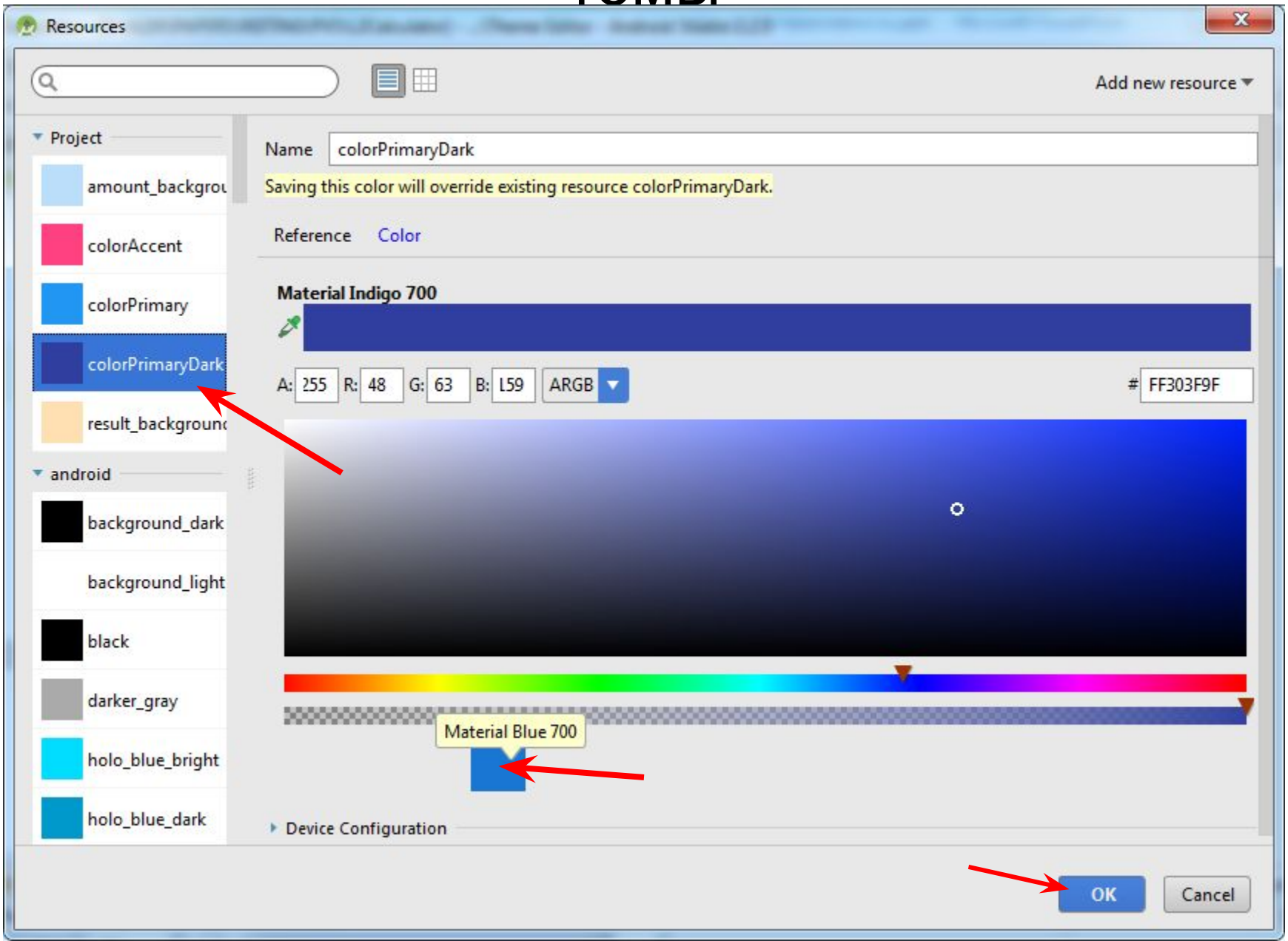
Тема по умолчанию и настройка цветов темы



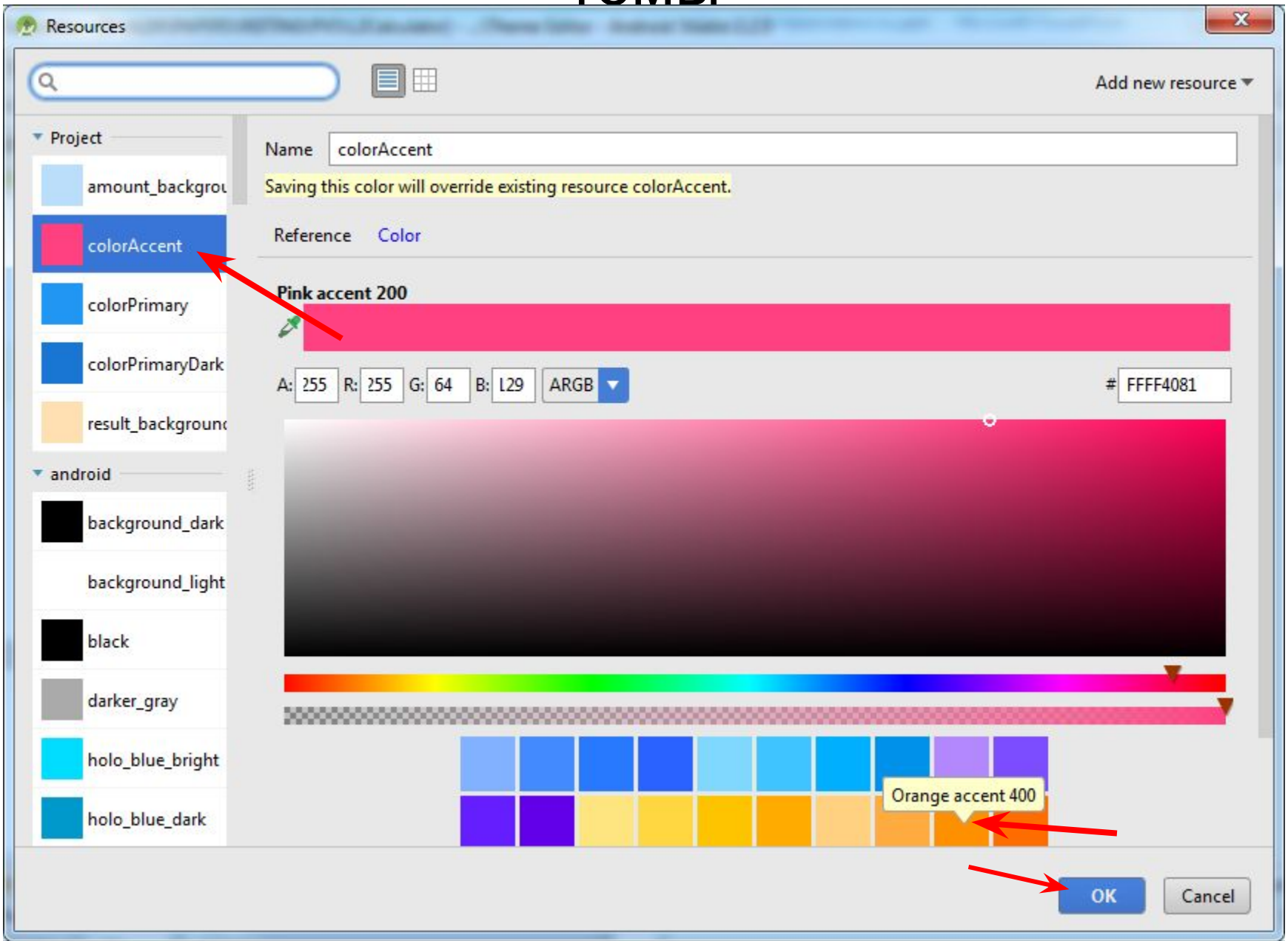
Тема по умолчанию и настройка цветов темы



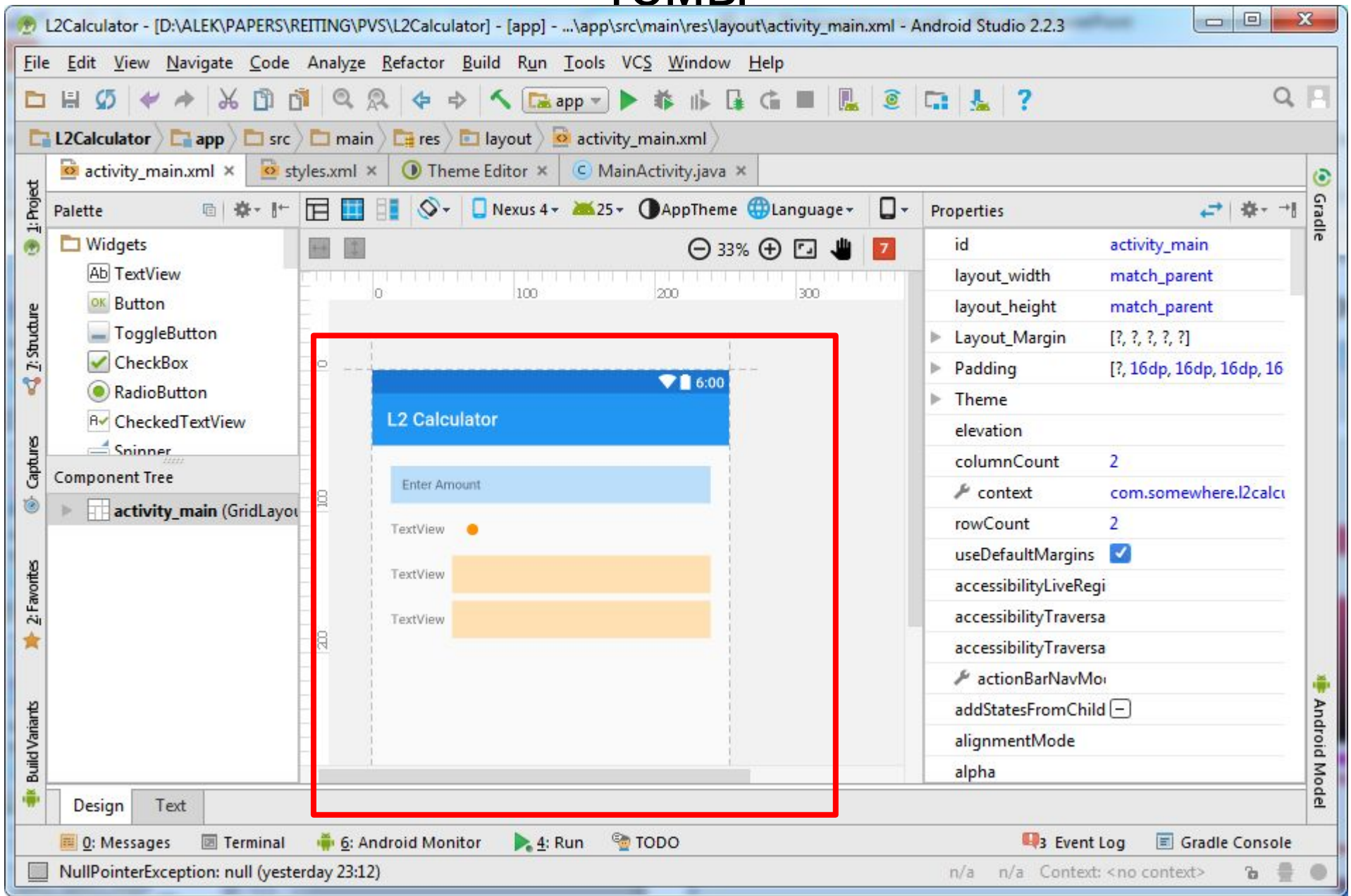
Тема по умолчанию и настройка цветов темы



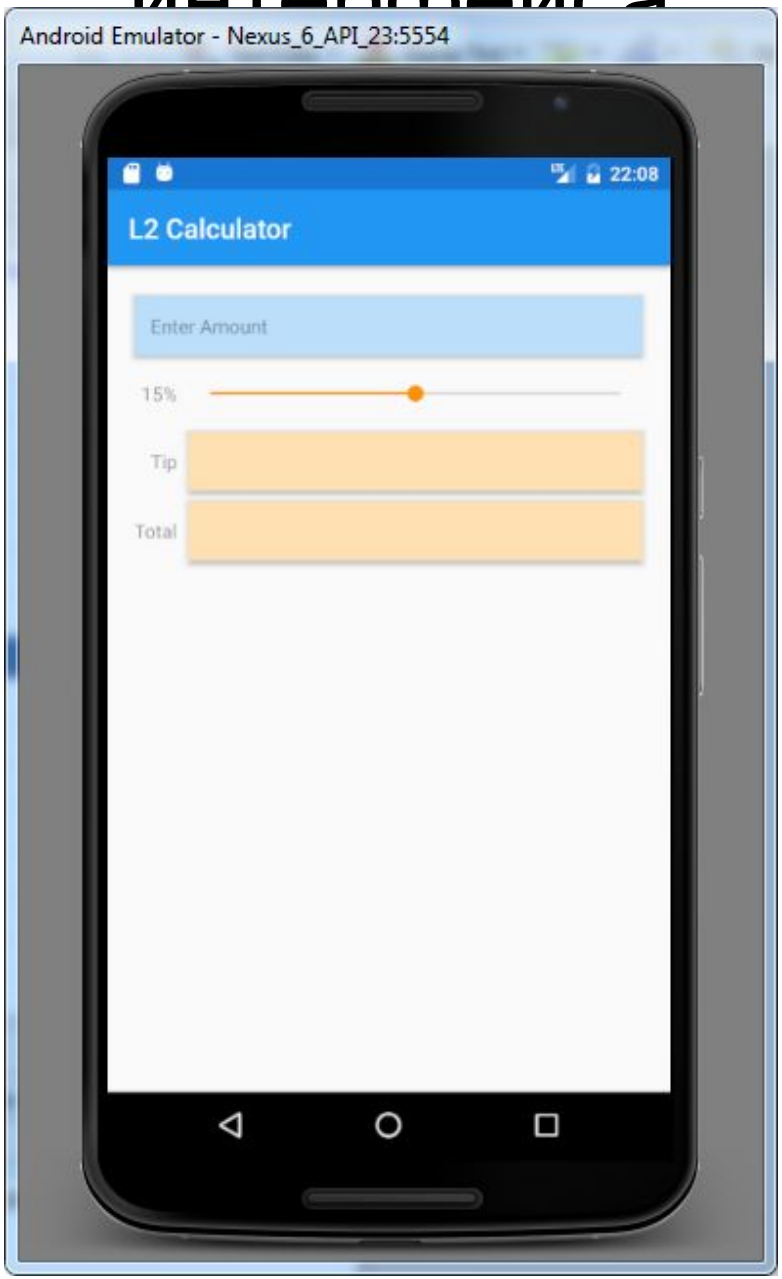
Тема по умолчанию и настройка цветов темы



Тема по умолчанию и настройка цветов темы



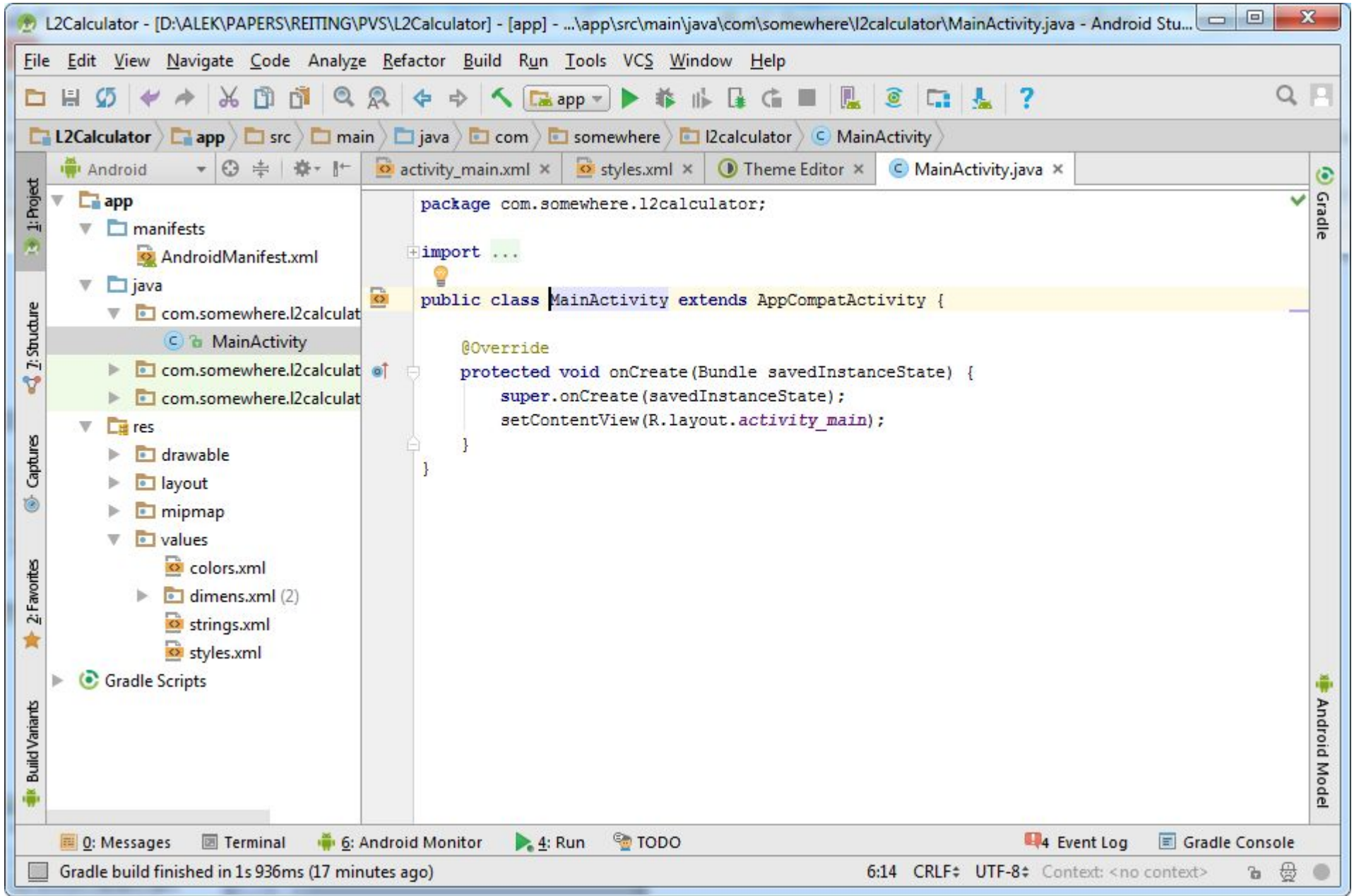
Построение графического интерфейса



Класс MainActivity

- реализует функциональность приложения
- вычисляет чаевые и общую сумму и выводит результаты в формате денежной суммы для действующего локального контекста
- чтобы просмотреть файл, в окне Project следует открыть папку app/Java/com.somewhere.l2calculator/ и сделать двойной щелчок на файле MainActivity.java

Класс MainActivity



Класс MainActivity

activity_main.xml × | styles.xml × | Theme Editor × | MainActivity.java ×

```
package com.somewhere.l2calculator;

import android.os.Bundle; // Для хранения информации состояния
import android.support.v7.app.AppCompatActivity; // Базовый класс
import android.text.Editable; // Для обработки событий EditText
import android.text.TextWatcher; // Слушатель EditText
import android.widget.EditText; // Для ввода счета
import android.widget.SeekBar; // Для изменения процента чаевых
import android.widget.SeekBar.OnSeekBarChangeListener; // Слушатель SeekBar
import android.widget.TextView; // Для вывода текста

import java.text.NumberFormat; // Для форматирования денежных сумм
```

Класс MainActivity

- Класс Bundle из пакета android.os хранит набор пар «ключ—значение» — обычно эти пары представляют информацию состояния приложения или данные, передаваемые между активностями. Android дает приложению возможность сохранить свое состояние в Bundle перед тем, как другое приложение появится на экране. После этого исполнительная среда Android может закрыть приложение, то есть освободить занимаемую им память. Когда приложение возвращается на экран, исполнительная среда Android передает объект Bundle с ранее сохраненным состоянием методу onCreate активности.

Класс MainActivity

- Класс AppCompatActivity из пакета android.support.v7.app предоставляет основные методы жизненного цикла приложения. Класс AppCompatActivity (непрямой субкласс класса Activity из пакета android.app) обеспечивает поддержку новой функциональности в приложениях, работающих на старых платформах Android.
- Интерфейс Editable из пакета android.text позволяет изменять содержимое и разметку текста в графическом интерфейсе.
- Интерфейс TextWatcher из пакета android.text реализуется для обработки событий при изменении пользователем текста в EditText.

Класс MainActivity

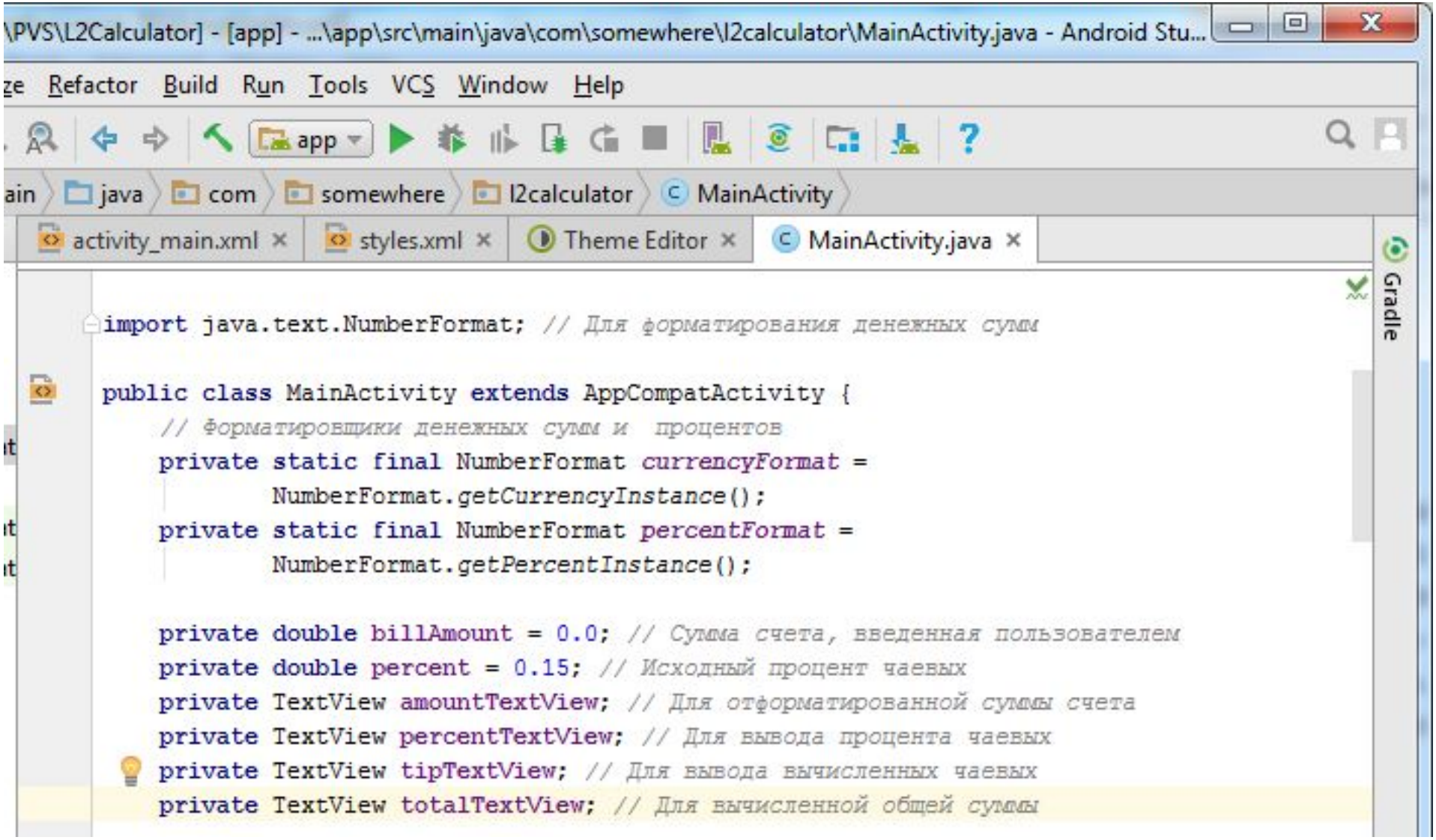
- Пакет android.widget содержит *виджеты* (визуальные компоненты) и макеты, используемые в графическом интерфейсе Android. Вприложении используются виджеты EditText, SeekBar и TextView.
- Интерфейс SeekBar.OnSeekBarChangeListener из пакета android.widget реализуется для обработки событий, возникающих при перемещении ползунка SeekBar.
- Класс NumberFormat из пакета java.text предоставляет средства числового форматирования (например, форматы денежных сумм и процентов по правилам локального

Класс MainActivity

Класс MainActivity является основным классом активности приложения. При создании проекта среда разработки сгенерировала этот класс как субкласс AppCompatActivity (непрямого субкласса Activity) и предоставила переопределенную версию унаследованного от Activity метода onCreate. Каждый субкласс Activity *должен* переопределять этот метод.

```
// Класс MainActivity приложения Tip Calculator
public class MainActivity extends AppCompatActivity {
```

Поля класса MainActivity



Переопределение метода onCreate()

- Метод генерируется автоматически и вызывается после запуска класса активности
- Обычно инициализирует поля экземпляра Activity и компоненты пользовательского интерфейса
- Должен быть предельно упрощён для быстрой загрузки
- Если загрузка превышает 5 секунд, операционная система отображает диалоговое окно «Приложение не отвечает» с возможностью принудительного завершения приложения

Переопределение метода onCreate()

```
// Вызывается при первом создании активности
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); // Вызов версии суперкласса
    setContentView(R.layout.activity_main); // Заполнение GUI

    // Получение ссылок на компоненты TextView, с  которыми
    // MainActivity взаимодействует на программном уровне
    amountTextView = (TextView) findViewById(R.id.amountTextView);
    percentTextView = (TextView) findViewById(R.id.percentTextView);
    tipTextView = (TextView) findViewById(R.id.tipTextView);
    totalTextView = (TextView) findViewById(R.id.totalTextView);
    tipTextView.setText(currencyFormat.format(0)); // Заполнить 0
    totalTextView.setText(currencyFormat.format(0)); // Заполнить 0

    // Назначение слушателя TextWatcher для amountEditText
    EditText amountEditText =
            (EditText) findViewById(R.id.amountEditText);
    amountEditText.addTextChangedListener(amountEditTextWatcher);

    // Назначение слушателя OnSeekBarChangeListener для percentSeekBar
    SeekBar percentSeekBar =
            (SeekBar) findViewById(R.id.percentSeekBar);
    percentSeekBar.setOnSeekBarChangeListener(seekBarListener);
}
```

Переопределение метода onCreate()

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); // Вызов версии суперкласса
    setContentView(R.layout.activity_main); // Заполнение GUI
```

- аргумент Bundle передаёт сохранённое состояние активности (если имеется)
- вызов super.onCreate() обязателен при переопределении метода
- класс R генерируется автоматически и содержит статические вложенные классы, представляющие каждый тип ресурсов из папки res проекта
- метод setContentView использует константу, определяющую XML-файл с графическим интерфейсом MainActivity, для *заполнения* (inflating) графического интерфейса

Переопределение метода onCreate()

```
amountTextView = (TextView) findViewById(R.id.amountTextView);
percentTextView = (TextView) findViewById(R.id.percentTextView);
tipTextView = (TextView) findViewById(R.id.tipTextView);
totalTextView = (TextView) findViewById(R.id.totalTextView);
tipTextView.setText(currencyFormat.format(0)); // Заполнить 0
totalTextView.setText(currencyFormat.format(0)); // Заполнить 0
```

- ссылки на виджеты получаем для выполнения программных операций с ними
- свойства text компонентов tipTextView и totalTextView заполняются исходным значением 0, отформатированным по правилам денежных сумм для текущего локального контекста

Переопределение метода onCreate()

```
    // Назначение слушателя TextWatcher для amountEditText
    EditText amountEditText =
            (EditText) findViewById(R.id.amountEditText);
    amountEditText.addTextChangedListener(amountEditTextWatcher);

    // Назначение слушателя OnSeekBarChangeListener для percentSeekBar
    SeekBar percentSeekBar =
            (SeekBar) findViewById(R.id.percentSeekBar);
    percentSeekBar.setOnSeekBarChangeListener(seekBarListener);
}
```

- получаем ссылку на компонент amountEditText и вызываем его метод addTextChangedListener для регистрации слушателя TextWatcher, который будет реагировать на изменение текста в EditText
- получаем ссылку на компонент percentSeekBar и вызываем его метод setOnSeekBarChangeListener для регистрации слушателя OnSeekBarChangeListener, который будет реагировать на перемещение ползунка SeekBar

Метод calculate()

- вызывается слушателями EditText и SeekBar для обновления компонентов TextView с чаевыми и общей суммой счета каждый раз, когда пользователь *изменяет* выставленный счет

```
// Вычисление и  вывод чаевых и  общей суммы
private void calculate() {
    // Форматирование процентов и  вывод в  percentTextView
    percentTextView.setText(percentFormat.format(percent));

    // Вычисление чаевых и  общей суммы
    double tip = billAmount * percent;
    double total = billAmount + tip;

    // Вывод чаевых и  общей суммы в  формате денежной величины
    tipTextView.setText(currencyFormat.format(tip));
    totalTextView.setText(currencyFormat.format(total));
}
```

Слушатель для percentSeekBar

```
// Объект слушателя для событий изменения состояния SeekBar
private final OnSeekBarChangeListener seekBarListener =
        new OnSeekBarChangeListener() {
            // Обновление процента чаевых и  вызов calculate
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
                                          boolean fromUser) {
                percent = progress / 100.0; // Назначение процента чаевых
                calculate(); // Вычисление и  вывод чаевых и  суммы
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) { }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) { }
        };
```

Слушатель для percentSeekBar

```
// Обновление процента чаевых и  вызов calculate
@Override
public void onProgressChanged(SeekBar seekBar, int progress,
                              boolean fromUser) {
    percent = progress / 100.0; // Назначение процента чаевых
    calculate(); // Вычисление и  вывод чаевых и  суммы
}
```

- переопределение метода onProgressChanged интерфейса OnSeekBarChangeListener
- метод onProgressChanged вызывается при каждом изменении позиции ползунка SeekBar
- вычисляется percent на основании параметра progress, представляющего позицию ползунка
- вызывается метод calculate для пересчета и отображения чаевых и общей суммы

Слушатель для percentSeekBar

```
@Override
public void onStartTrackingTouch(SeekBar seekBar) { }

@Override
public void onStopTrackingTouch(SeekBar seekBar) { }
```

- в языке Java *каждый* метод реализуемого интерфейса должен переопределяться в программе
- предоставляем пустое тело для методов onStartTrackingTouch и onStopTrackingTouch , так как нам не нужно знать, когда пользователь начинает или прекращает перемещать ползунок
- можно автоматизировать переопределение методов интерфейса (курсор внутри класса, меню Code|Override Methods...)

Слушатель для amountEditText

```
// Объект слушателя для событий изменения текста в  EditText
private final TextWatcher amountEditTextWatcher = new TextWatcher() {
    // Вызывается при изменении пользователем величины счета
    @Override
    public void onTextChanged(CharSequence s, int start,
                              int before, int count) {

        try { // Получить счет и  вывести в  формате денежной суммы
            billAmount = Double.parseDouble(s.toString()) / 100.0;
            amountTextView.setText(currencyFormat.format(billAmount));
        }
        catch (NumberFormatException e) { // Если s пусто или не число
            amountTextView.setText("");
            billAmount = 0.0;
        }

        calculate(); // Обновление полей с  чаевыми и  общей суммой
    }

    @Override
    public void afterTextChanged(Editable s) { }

    @Override
    public void beforeTextChanged(
            CharSequence s, int start, int count, int after) { }
};
```

Слушатель для amountEditText

```
public void onTextChanged(CharSequence s, int start,
                          int before, int count) {

    try { // Получить счет и  вывести в  формате денежной суммы
        billAmount = Double.parseDouble(s.toString()) / 100.0;
        amountTextView.setText(currencyFormat.format(billAmount));
    }
    catch (NumberFormatException e) { // Если s пусто или не число
        amountTextView.setText("");
        billAmount = 0.0;
    }

    calculate(); // Обновление полей с  чаевыми и  общей суммой
}
```

- onTextChanged вызывается при каждом изменении текста в компоненте amountEditText.
- параметр CharSequence s содержит копию текста
- другие параметры сообщают о том, что текст длиной count заменил фрагмент прежнего текста длины before, начинающийся в позиции start

AndroidManifest.xml

- отредактируем файл AndroidManifest.xml, чтобы указать, что
  - активность приложения поддерживает только портретную ориентацию устройства
  - виртуальная клавиатура должна отображаться постоянно после появления активности на экране или возвращения к ней
- подробная информация о содержании манифеста: *http://developer.android.com/guide/topics/manifest/manifest-intro.html*

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.somewhere.l2calculator">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="L2 Calculator"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateAlwaysVisible">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.somewhere.l2calculator">

    <application...>

</manifest>
```

- элемент <manifest> указывает, что содержимое файла XML представляет манифест приложения
- атрибут package этого элемента задает имя пакета Java, введенное при создании проекта приложения (для приложений, отправляемых в магазин Google Play, имя пакета используется в качестве уникального идентификатора приложения)
- элемент <application> задаёт атрибуты приложения

AndroidManifest.xml

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="L2 Calculator"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity"...>
</application>
```

- android:allowBackup — флаг автоматического резервного копирования данных приложения
- android:icon— значок, используемый в лаунчере для запуска приложения
- android:label — название приложения, которое обычно отображается под значком в лаунчере
- android:supportsRtl — флаг возможности горизонтального отражения интерфейса приложения для поддержки языков, в которых буквы пишутся справа налево (арабский, иврит)
- android:theme — тема, описывающая оформление компонентов приложения по умолчанию

AndroidManifest.xml

```
<activity android:name=".MainActivity">
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateAlwaysVisible">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

- android:name — имя класса активности (можно полностью com.somewhere.l2calculator.MainActivity)
- android:screenOrientation — переход устройства в альбомную ориентацию на типичном телефоне приведет к тому, что цифровая клавиатура закроет большую часть графического интерфейса. По этой причине задаем этому свойству значение "portrait"
- android:windowSoftInputMode — виртуальная клавиатура должна появиться сразу же после запуска приложения и снова появляться каждый раз, когда пользователь возвращается к приложению. По этой причине свойству задается значение "stateAlwaysVisible". При наличии аппаратной клавиатуры в этом

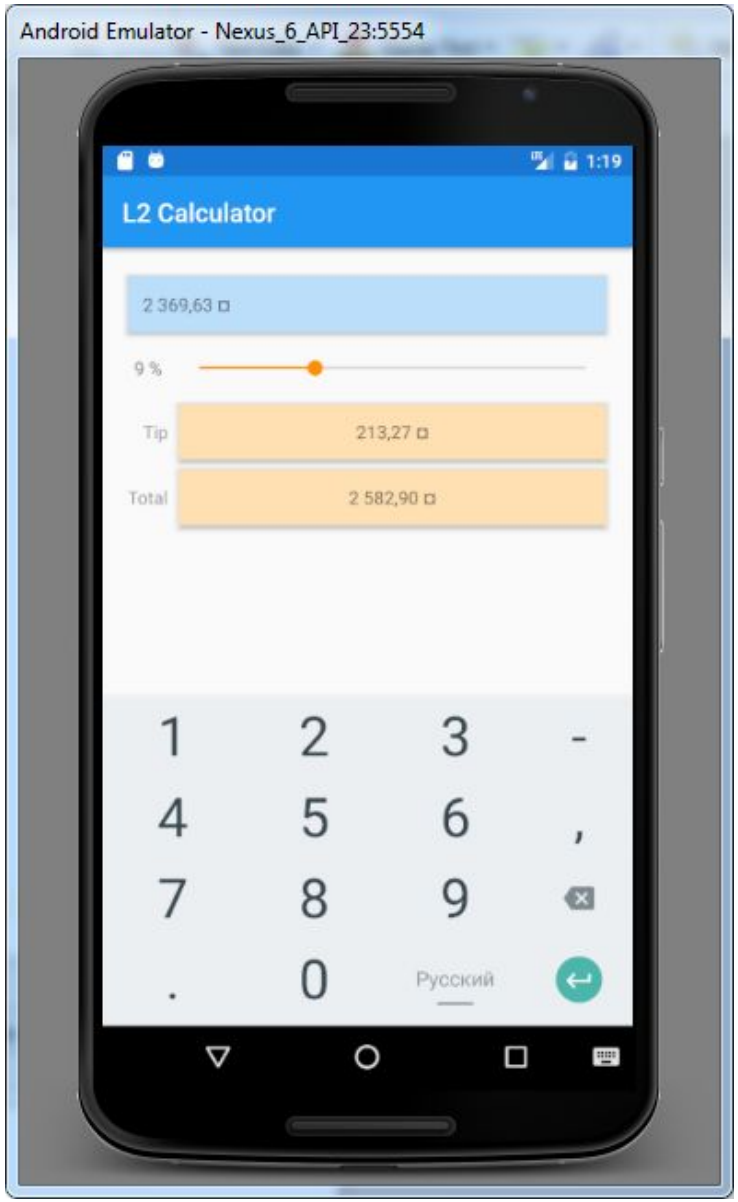
AndroidManifest.xml

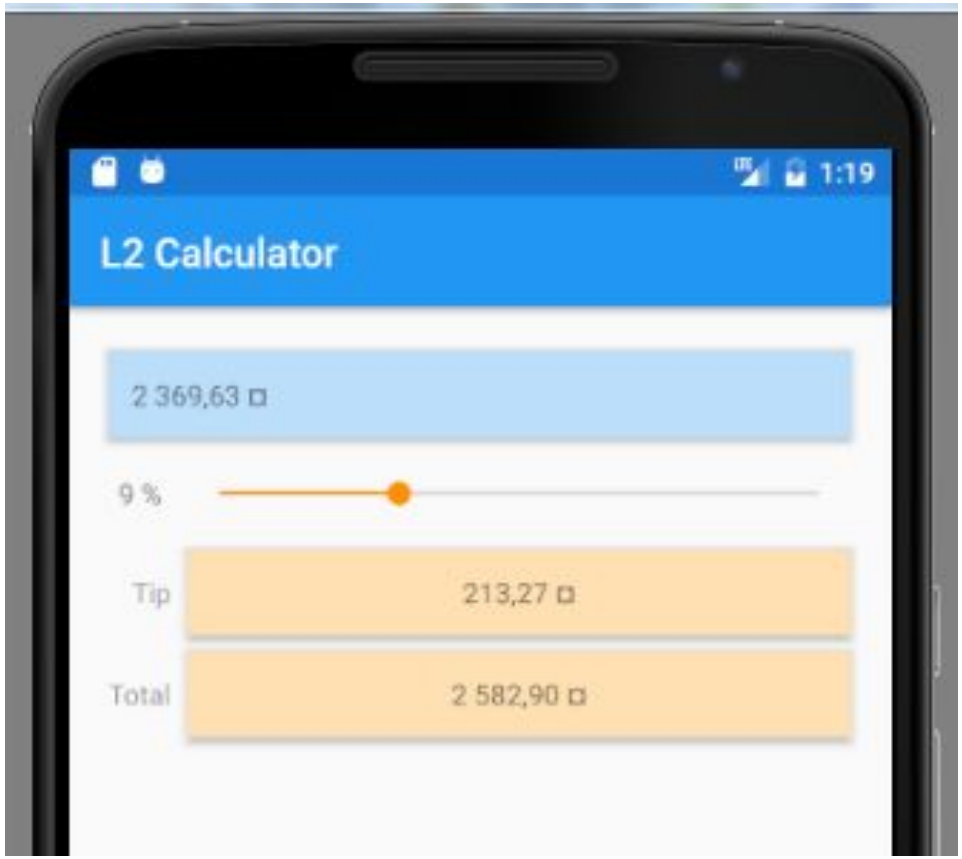
```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

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

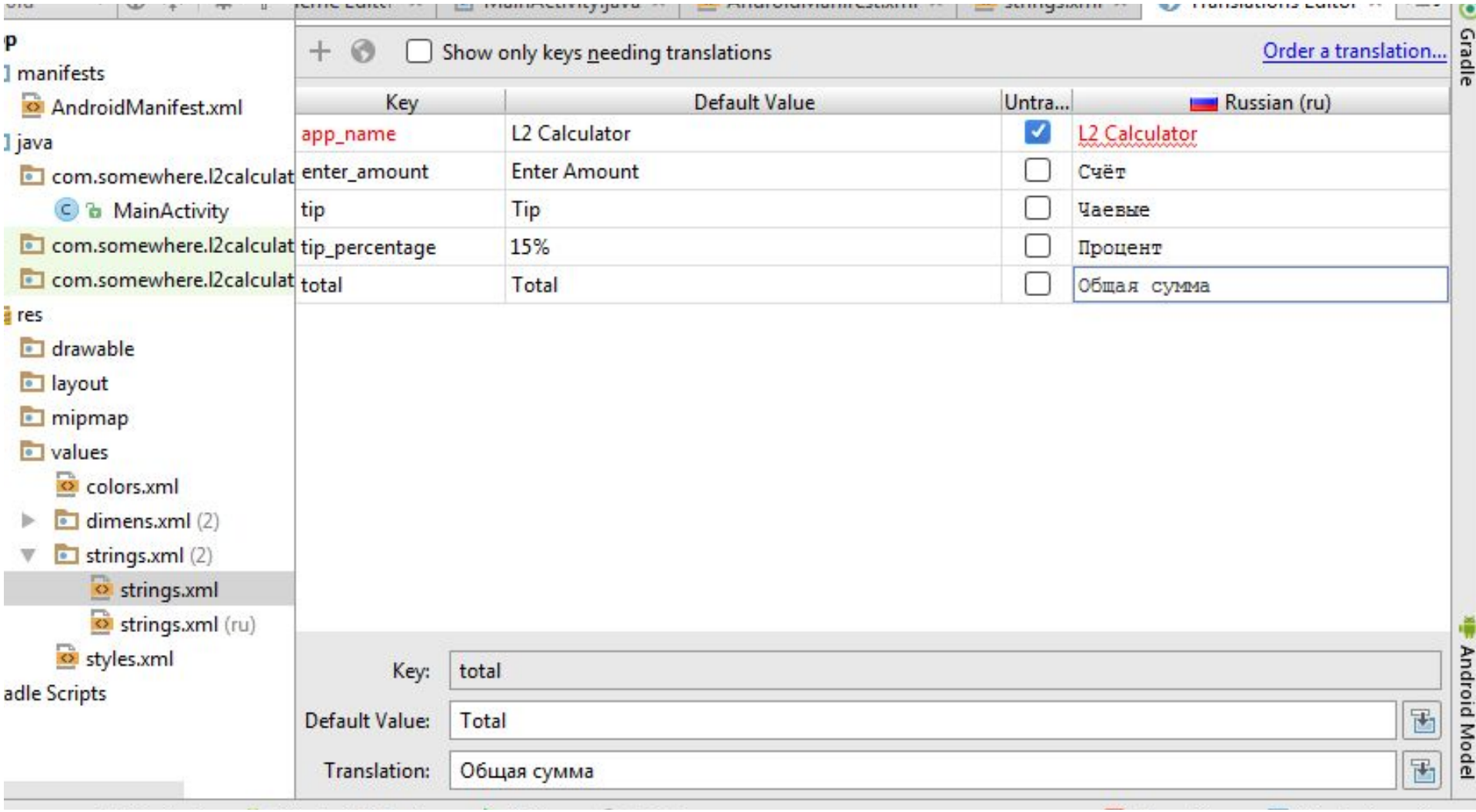
- механизм *интентов* (intent) используется в Android для организации взаимодействия между выполняемыми компонентами — активностями, фоновыми службами и операционной системой
- элемент intent-filter должен содержать один или несколько элементов action
- android.intent.action.MAIN означает, что MainActivity является главной активностью, которая должна выполняться при запуске приложения
- category (не обязательно) указывает, что является источником интента — для "android.intent.category.LAUNCHER" это лаунчер устройства. Данное значение также указывает, что активность должна отображаться в виде значка в лаунчере устройства вместе со значками других установленных приложений

Тестирование приложения





Локализация



Локализация