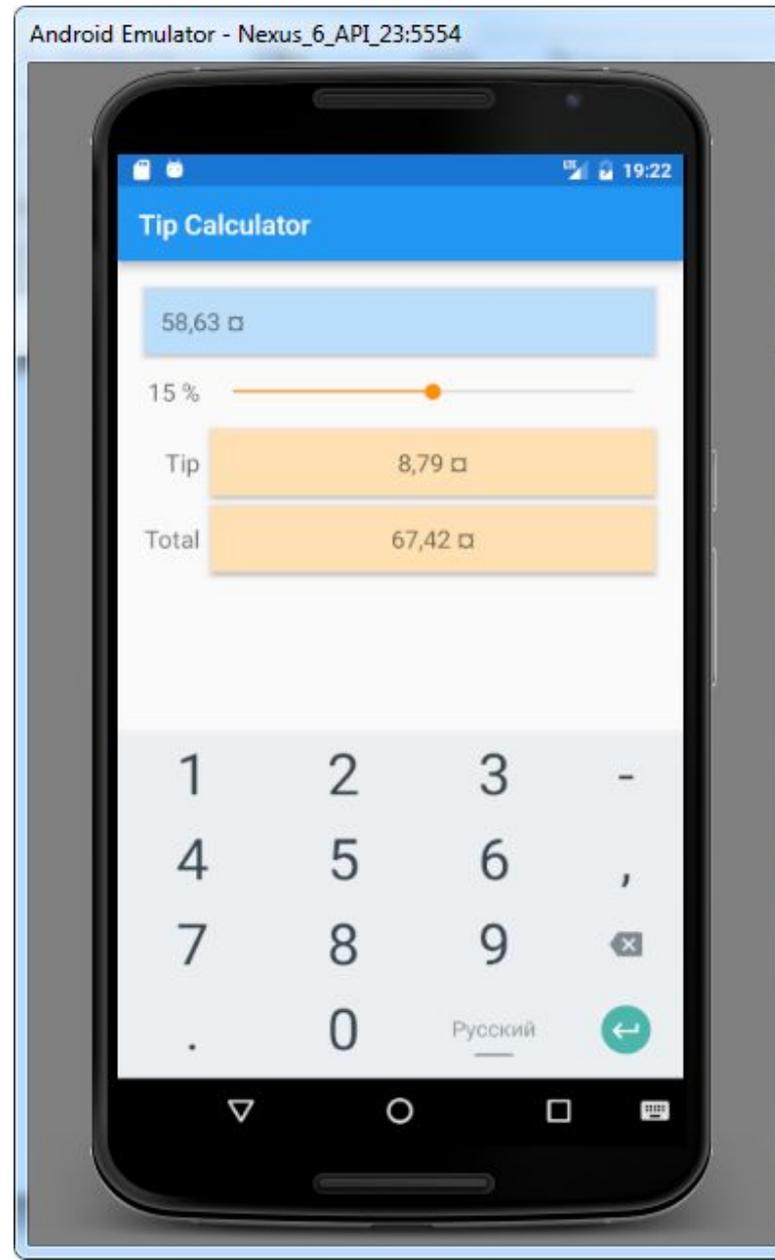


Android 6
Проект с базовой
ИНТЕРАКТИВНОСТЬЮ

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, subclasses 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, subclasses 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 потребуются 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 Android 5 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

The screenshot shows the Android Studio interface for an application named "L2Calculator". The main workspace is in "Design" mode, displaying a mobile device mockup with a blue header bar labeled "L2 Calculator" and a white content area. The "Properties" panel on the right shows the properties of the selected "GridLayout" component, with the "rowCount" property set to 4. The "Component Tree" on the left shows the "activity_main (GridLayout)" component selected. The "Widget Palette" on the left lists various UI components like TextView, Button, and ProgressBar. The "Properties" panel also shows other properties like "rotationY", "saveEnabled", "textDirection", "touchscreenBloc", "transformPivotX", "transformPivotY", "transitionGroup", "transitionName", "translationX", "translationY", "translationZ", "useDefaultMargin" (checked), "verticalScrollbar", and "visibility".

Properties:

- stateListAnimator
- rotationY
- rowCount: 4
- rowOrderPreserv
- saveEnabled
- textDirection
- touchscreenBloc
- transformPivotX
- transformPivotY
- transitionGroup
- transitionName
- translationX
- translationY
- translationZ
- useDefaultMargin:
- verticalScrollbar
- visibility

Component Tree:

- activity_main (GridLayout)

Построение графического интерфейса

The screenshot displays the Android Studio 2.2.3 interface for an application named "L2Calculator". The main workspace shows a design view of the "activity_main.xml" layout, which is a GridLayout. A red arrow points from the "amountEditText" component in the Component Tree to its corresponding entry in the Properties panel. The Properties panel shows the following layout properties for "amountEditText":

id	amountEditText
layout_width	wrap_content
layout_height	wrap_content
Layout_Margin	[?, ?, ?, ?]
Padding	[?, ?, ?, ?]
Theme	
elevation	
ems	10
layout_column	0
layout_columnSpan	2
accessibilityLabel	
allowUndo	<input type="checkbox"/>
alpha	
autoLink	<input type="checkbox"/>
autoText	<input type="checkbox"/>
background	?attr/editTextBackgri
backgroundTint	

The Component Tree on the left shows the "activity_main" (GridLayout) containing the "amountEditText" (EditText) component. The Properties panel also shows the "amountEditText" component selected, with its properties listed. A red box highlights the "layout_column" and "layout_columnSpan" properties in the Properties panel.

Построение графического интерфейса

The screenshot shows the Android Studio interface for an application named 'L2Calculator'. The main design view displays a calculator app with a blue header and a white content area. A red box highlights the 'amountTextView' widget in the Component Tree and its properties in the Properties panel. The Properties panel shows attributes like 'id', 'layout_width', 'layout_height', 'layout_column', and 'layout_row'. A red arrow points from the 'amountTextView' widget in the Component Tree to the 'amountTextView' property in the Properties panel.

Property	Value
id	amountTextView
layout_width	wrap_content
layout_height	wrap_content
Layout_Margin	[?, ?, ?, ?]
Padding	[?, ?, ?, ?]
Theme	
elevation	
text	TextView
accessibilityLiveRegion	
layout_column	0
layout_columnSpan	2
layout_columnWeig	
layout_gravity	[]
layout_row	0
autoText	[-]
background	
backgroundTint	
backgroundTintMoc	

Построение графического интерфейса

The screenshot displays the Android Studio 2.2.3 interface for developing an application named "L2 Calculator". The main design area shows a mobile device screen with a blue header "L2 Calculator" and a status bar at the top. The layout is a GridLayout with several components: a TextView, an EditText, and a SeekBar. Red arrows point from the Palette's "SeekBar" component to the SeekBar on the screen, and from the Component Tree's "percentSeekBar" to the same SeekBar. The Properties panel on the right lists the attributes of the selected SeekBar.

Attribute	Value
id	percentSeekBar
layout_width	wrap_content
layout_height	wrap_content
Layout_Margin	[?, ?, ?, ?]
Padding	[?, 16dip, ?, 16dip, ?]
Theme	
elevation	
accessibilityLiveRegi	
accessibilityTraversa	
accessibilityTraversa	
alpha	
animationResolutior	
background	@drawable/control_bac
backgroundTint	
backgroundTintMoc	
clickable	-
contentDescription	
contextClickable	-

Построение графического интерфейса

The screenshot displays the Android Studio 2.2.3 interface for an application named "L2Calculator". The main design area shows a blue-themed calculator layout with various text views and a seek bar. The Component Tree on the left lists several text views, with "totalTextView" highlighted in a red box. The Properties panel on the right shows the attributes for the selected "totalTextView" widget.

id	totalTextView
layout_width	wrap_content
layout_height	wrap_content
Layout_Margin	[?, ?, ?, ?]
Padding	[?, ?, ?, ?]
Theme	
elevation	
text	TextView
accessibilityLiveRegi	
accessibilityTraversa	
accessibilityTraversa	
allowUndo	-
alpha	
autoLink	[]
autoText	-
background	
backgroundTint	
backgroundTintMoc	

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
<code>percentTextView</code>	<code>tip_percentage</code>	15%
<code>tipLabelTextView</code>	<code>tip</code>	Tip
<code>totalLabelTextView</code>	<code>total</code>	Total

Построение графического интерфейса

The screenshot shows the Android Studio 2.2.3 interface for developing an Android application named "L2Calculator". The main workspace displays a design preview of the app's main activity, which is a calculator interface. The interface features a blue header with the text "L2 Calculator" and a status bar at the top showing the time as 6:00. Below the header, there are several input fields and buttons, including an "amount" field, a "percent" field, and a "total" field. The left sidebar shows the "Component Tree" with the following structure:

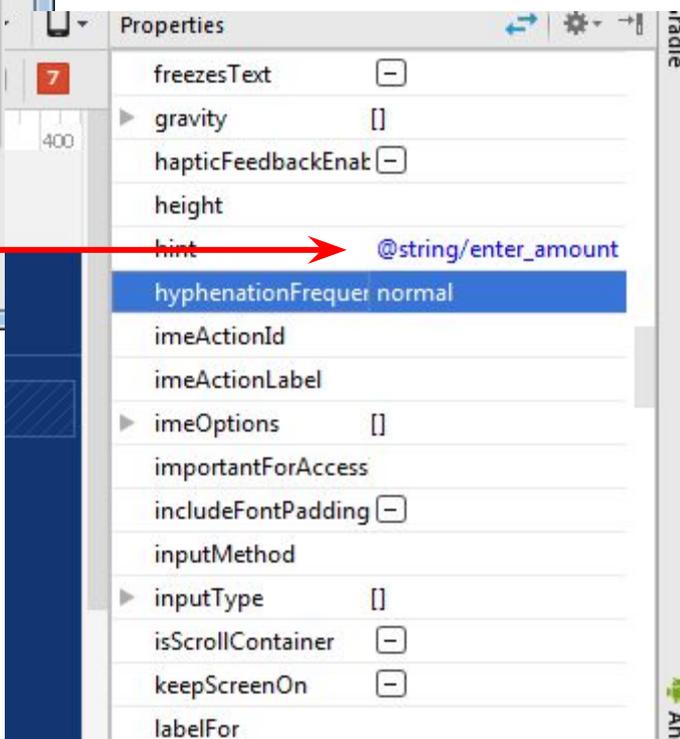
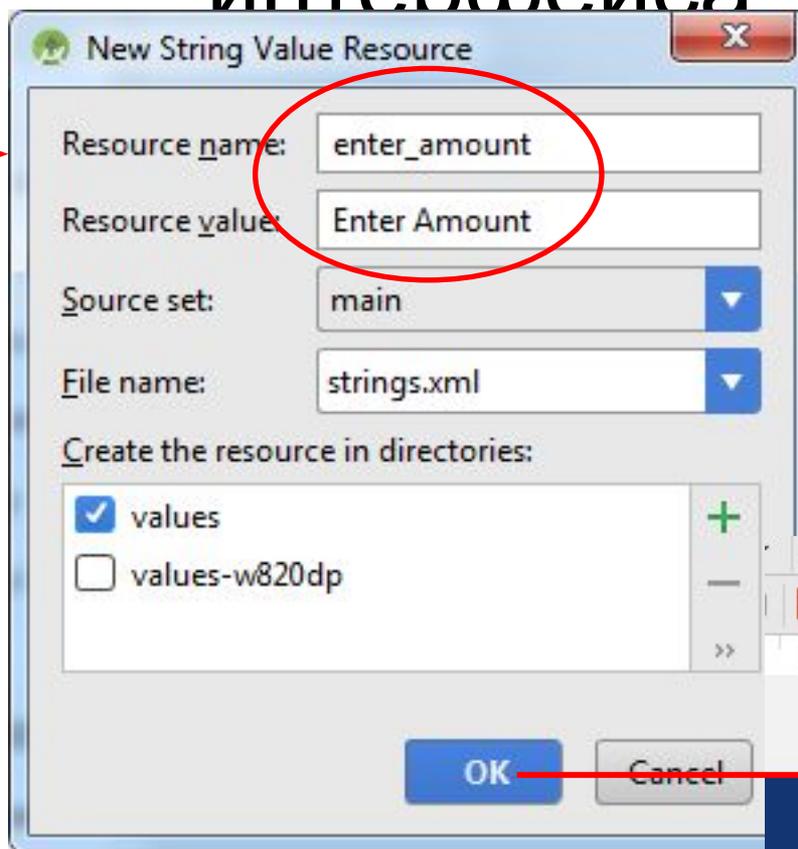
- activity_main (GridLayout)
 - amountEditText (EditText)
 - amountTextView (TextView)
 - percentTextView (TextView)
 - percentSeekBar (SeekBar)
 - tipLabelTextView (TextView)
 - tipTextView (TextView)
 - totalLabelTextView (TextView)
 - totalTextView (TextView)

The right sidebar shows the "Properties" panel for the selected component, with the following properties listed:

- freezesText
- gravity
- hapticFeedbackEnabled
- height
- hint
- hyphenationFrequency
- imeActionId
- imeActionLabel
- imeOptions
- importantForAccessibility
- includeFontPadding
- inputMethod
- inputType
- isScrollContainer
- keepScreenOn
- labelFor
- layerType
- layoutDirection

Red arrows indicate the relationship between the Component Tree and the Properties panel. One arrow points from the "amountTextView" component in the Component Tree to the "hint" property in the Properties panel. Another arrow points from the "hint" property in the Properties panel to the corresponding "hint" attribute in the XML layout file, which is visible in the background of the design preview.

Построение графического интерфейса



Построение графического интерфейса

The screenshot shows the Android Studio interface for an application named "L2Calculator". The main workspace displays a design view of the "activity_main.xml" layout. The layout is a GridLayout with a blue header bar containing the text "L2 Calculator". Below the header, there are several UI elements: an "amount" input field, a "percent" input field, a "tip" input field, and a "total" input field. The Component Tree on the left shows the hierarchy of these elements, with several TextViews highlighted by red arrows. The Properties panel on the right shows the "layout_gravity" property set to "right".

Component Tree:

- activity_main (GridLayout)
 - amountEditText (EditText)
 - amountTextView (TextView)
 - percentTextView (TextView)
 - percentSeekBar (SeekBar)
 - tipLabelTextView (TextView)
 - tipTextView (TextView)
 - totalLabelTextView (TextView)
 - totalTextView (TextView)

Properties Panel:

Property	Value
layout_gravity	right
top	<input type="checkbox"/>
bottom	<input type="checkbox"/>
left	<input type="checkbox"/>
right	<input checked="" type="checkbox"/>
center_vertical	<input type="checkbox"/>
fill_vertical	<input type="checkbox"/>
center_horizontal	<input type="checkbox"/>
fill_horizontal	<input type="checkbox"/>
center	<input type="checkbox"/>
fill	<input type="checkbox"/>
clip_vertical	<input type="checkbox"/>
clip_horizontal	<input type="checkbox"/>
start	<input type="checkbox"/>
end	<input type="checkbox"/>
layout_row	
layout_rowSpan	
layout_rowWeight	

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

The screenshot displays the Android Studio 2.2.3 interface for an application named "L2Calculator". The main workspace shows a design view of the "activity_main.xml" layout, which is a GridLayout. The design features a blue header with the text "L2 Calculator", a light blue input field labeled "Enter Amount", and three orange buttons below it. The design is shown on a Nexus 4 device with a status bar at the top displaying the time 6:00. The Component Tree on the left lists the views in the hierarchy: activity_main (GridLayout), amountEditText (EditText), amountTextView (TextView), percentTextView (TextView), percentSeekBar (SeekBar), tipLabelTextView (TextView), tipTextView (TextView), totalLabelTextView (TextView), and totalTextView (TextView). The Properties panel on the right shows the attributes of the selected view, including layout_width, layout_height, elevation, text, accessibilityLiveRegion, accessibilityTraversal, allowUndo, alpha, autoLink, autoText, background, backgroundTint, backgroundTintMode, and breakStrategy.

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

L2Calculator [app] src main res layout activity_main.xml

activity_main.xml x MainActivity.java x

Palette

- Widgets
- TextView
- Button
- ToggleButton
- CheckBox
- RadioButton
- CheckedTextView
- Spinner

Component Tree

- activity_main (GridLayout)
- amountEditText (EditText)
- amountTextView (TextView)
- percentTextView (TextView)
- percentSeekBar (SeekBar)
- tipLabelTextView (TextView)
- tipTextView (TextView)
- totalLabelTextView (TextView)
- totalTextView (TextView)

Properties

- Layout_Margin [?, ?, ?, ?]
- Padding [12dp, ?, ?, ?]
- Theme
- layout_width wrap_content
- layout_height wrap_content
- elevation @dimen/elevation
- text
- accessibilityLiveRegion
- accessibilityTraversal
- accessibilityTraversal
- allowUndo [-]
- alpha
- autoLink []
- autoText [-]
- background @color/result_backg
- backgroundTint
- backgroundTintMode
- breakStrategy high_quality

Design Text

0: Messages Terminal 6: Android Monitor 4: Run TODO

3 Event Log Gradle Console

NullPointerException: null (yesterday 23:12)

1:1 n/a n/a Context: <no context>

Тема по умолчанию и настройка цветов темы

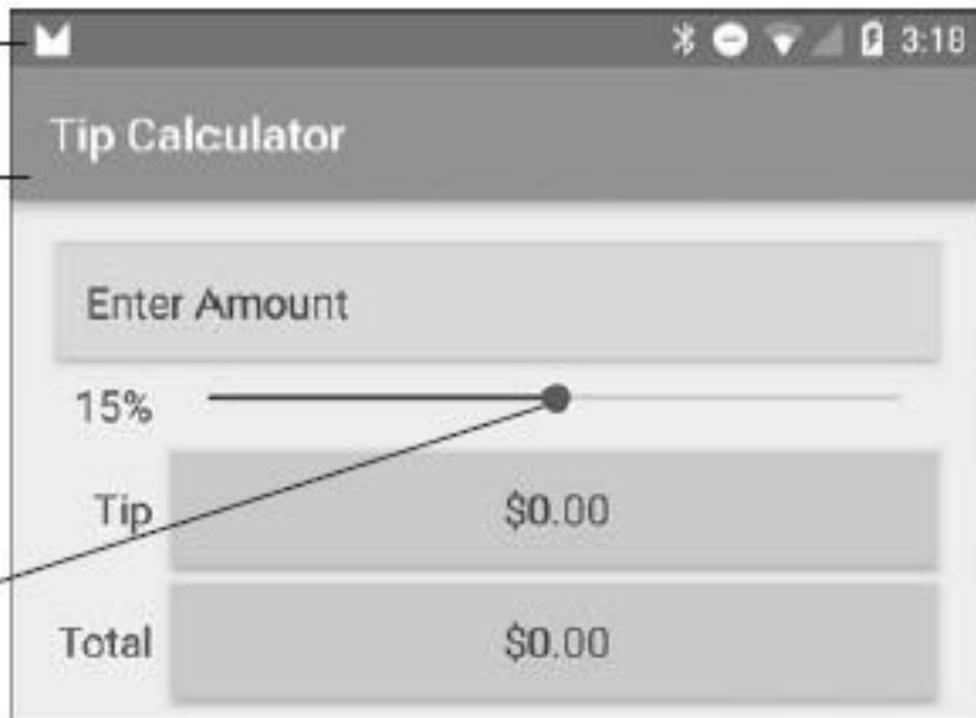
- У каждого приложения существует тема, определяющая оформление стандартных компонентов, которые используются.
- Тема приложения указывается в файле `AndroidManifest.xml` приложения.
- Можно настроить различные аспекты темы (например, составляющие цветовой схемы), определяя ресурсы в файле `styles.xml`,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

Тема по умолчанию и настройка цветов ТЕМЫ

`android:colorPrimaryDark`
используется в строке состояния

`android:colorPrimary` используется
в панели приложения

`android:colorAccent`
используется для придания
оттенков различным элементам
управления, включая `SeekBar`



ТЕМЫ

The screenshot displays the Android Studio 2.2.3 interface. The top toolbar includes icons for File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The breadcrumb navigation shows the path: L2Calculator > app > src > main > res > values > styles.xml. The left sidebar contains the Project view, Structure view, Captures, Favorites, and Build Variants. The Project view shows the following structure:

- app
 - manifests
 - AndroidManifest.xml
 - java
 - com.somewhere.l2calculat
 - com.somewhere.l2calculat
 - com.somewhere.l2calculat
 - res
 - drawable
 - layout
 - mipmap
 - values
 - colors.xml
 - dimens.xml (2)
 - strings.xml
 - styles.xml
 - Gradle Scripts

The main editor displays the `styles.xml` file with the following content:

```
<resources>  
  
  <!-- Base application theme. -->  
  <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">  
    <!-- Customize your theme here. -->  
    <item name="colorPrimary">@color/colorPrimary</item>  
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>  
    <item name="colorAccent">@color/colorAccent</item>  
  </style>  
  
</resources>
```

At the top of the editor, a notification reads: "Edit all themes in the project in the theme editor." with links for "Open editor" and "Hide notification". A red arrow points from the "Open editor" link to the right. Another red arrow points from the "styles.xml" file in the Project view to the main editor area.

The bottom status bar shows: 0: Messages, Terminal, Android Monitor, 4: Run, TODO, 3: Event Log, Gradle Console, 1:1 CRLF+ UTF-8 Context: <no context>

ТЕМЫ

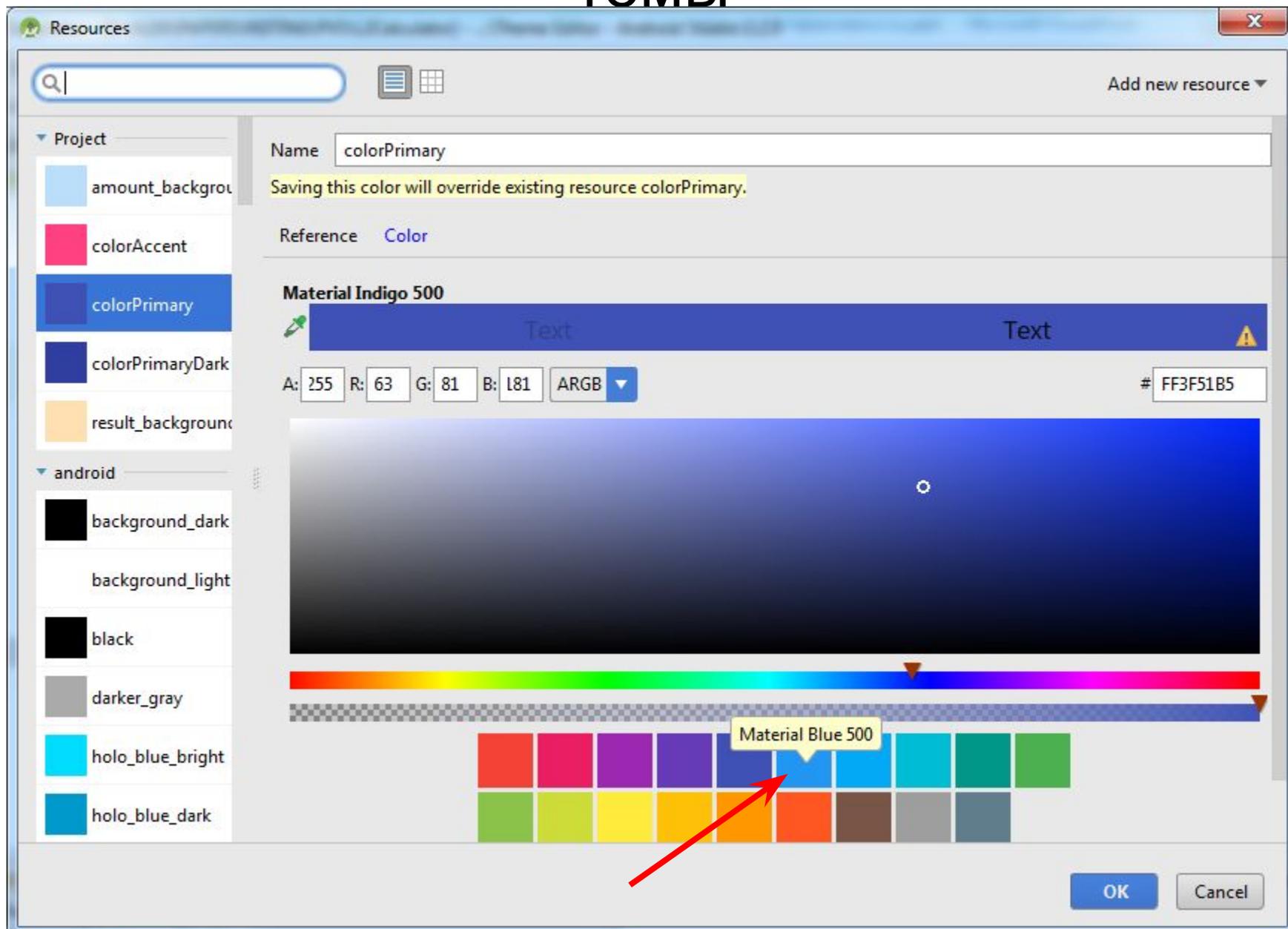
The screenshot displays the Android Studio 2.2.3 Theme Editor interface. The main window shows a preview of the application theme, which includes a blue toolbar with a back arrow and the text "Toolbar", and a raised button with the text "NORMAL" and "DISABLED". The theme is currently set to "AppTheme".

The right-hand panel, titled "Theme", shows the configuration for the selected theme. It includes the following settings:

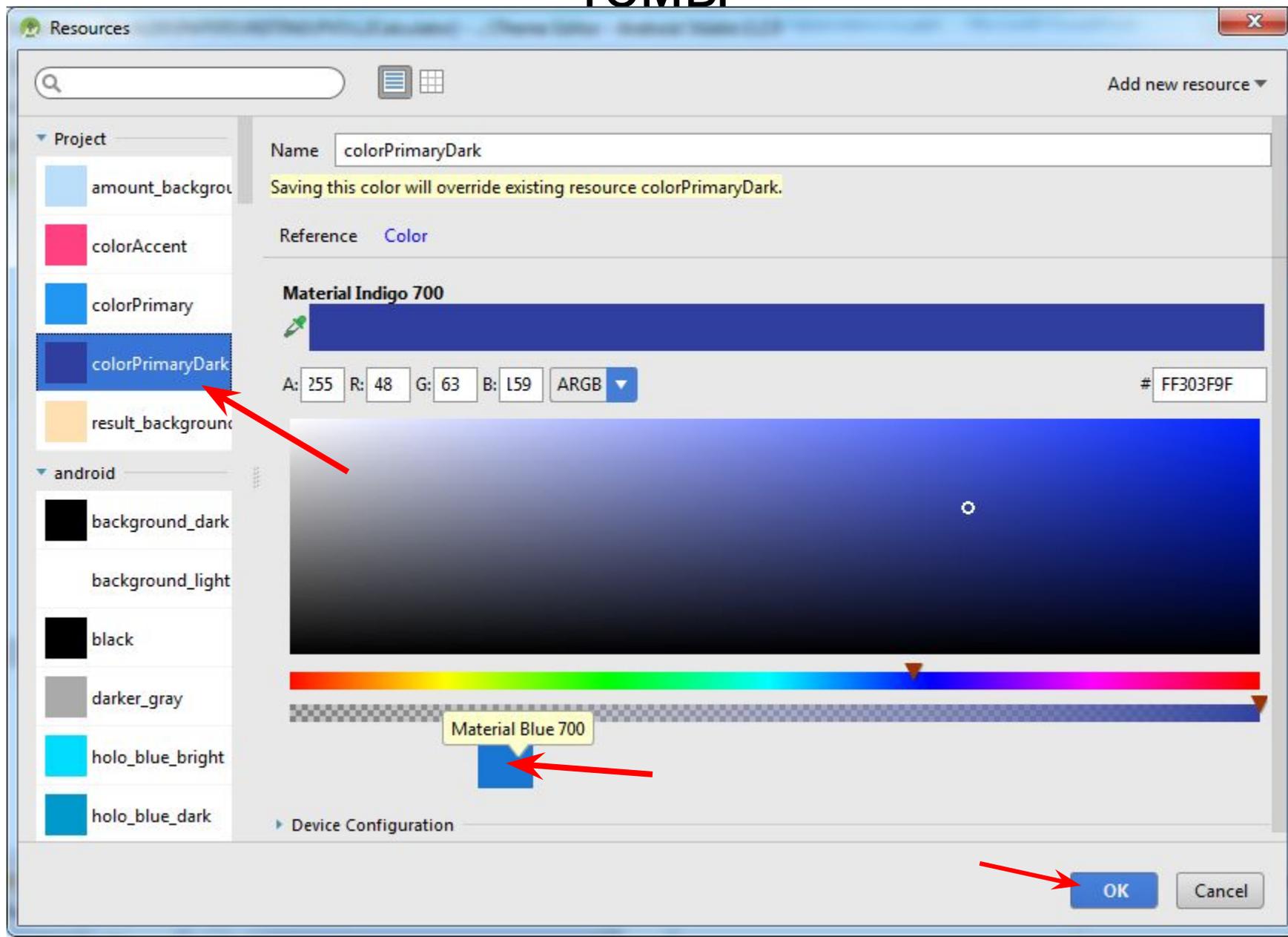
- Theme: AppTheme
- Theme parent: default (AppCompat Light [Theme])
- colorPrimary: default (@color/colorPrimary) - A red arrow points to this setting.
- colorPrimaryDark: default (@color/colorPrimar...)
- colorAccent: default (@color/colorAccent)

The bottom status bar shows the following information: "NullPointerException: null (yesterday 23:12)", "n/a n/a Context: <no context>", and "Event Log" and "Gradle Console" tabs.

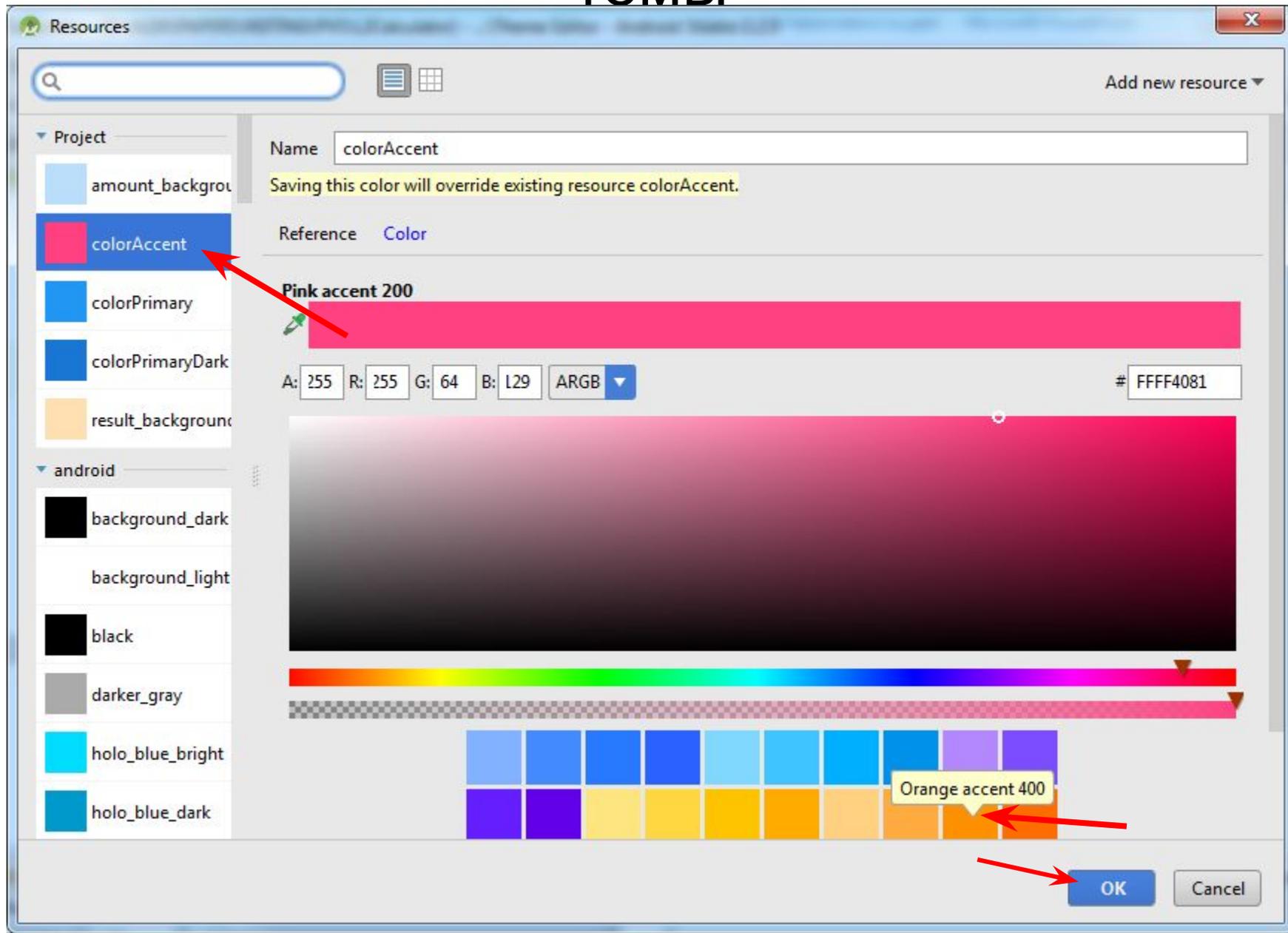
ТЕМЫ



ТЕМЫ



ТЕМЫ

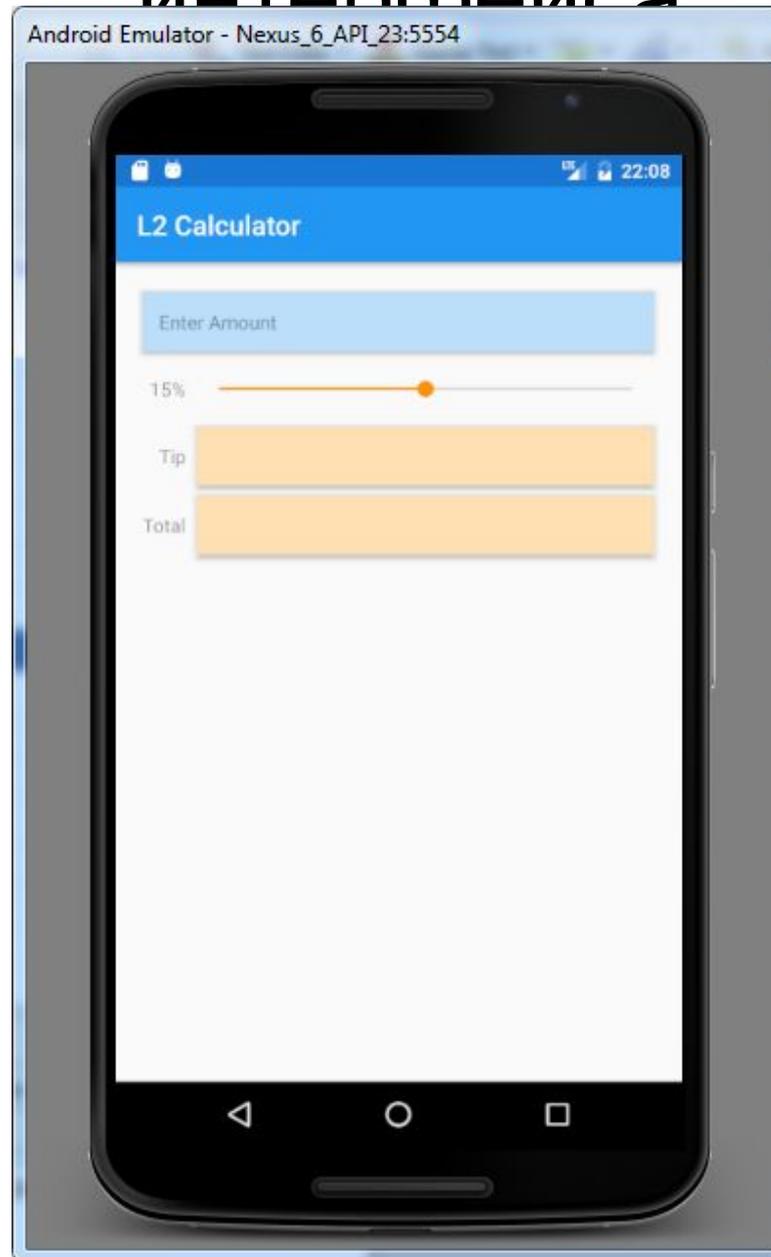


ТЕМЫ

The screenshot displays the Android Studio interface for an application named 'L2Calculator'. The main design view shows a mobile app layout with a blue header bar containing the text 'L2 Calculator'. Below the header is a light blue input field labeled 'Enter Amount', followed by two orange buttons. A red rectangular box highlights this design view. On the left, the 'Palette' and 'Component Tree' are visible. The 'Component Tree' shows 'activity_main (GridLayout)'. On the right, the 'Properties' panel lists various attributes for the selected component, such as 'id', 'layout_width', 'layout_height', and 'padding'. The 'Theme' property is set to 'AppTheme'. The bottom status bar shows a 'NullPointerException: null (yesterday 23:12)' error.

Property	Value
id	activity_main
layout_width	match_parent
layout_height	match_parent
Layout_Margin	[?, ?, ?, ?]
Padding	[?, 16dp, 16dp, 16]
Theme	AppTheme
elevation	
columnCount	2
context	com.somewhere.l2calcu
rowCount	2
useDefaultMargins	<input checked="" type="checkbox"/>
accessibilityLiveRegi	
accessibilityTraversa	
accessibilityTraversa	
actionBarNavMo	
addStatesFromChild	<input type="checkbox"/>
alignmentMode	
alpha	

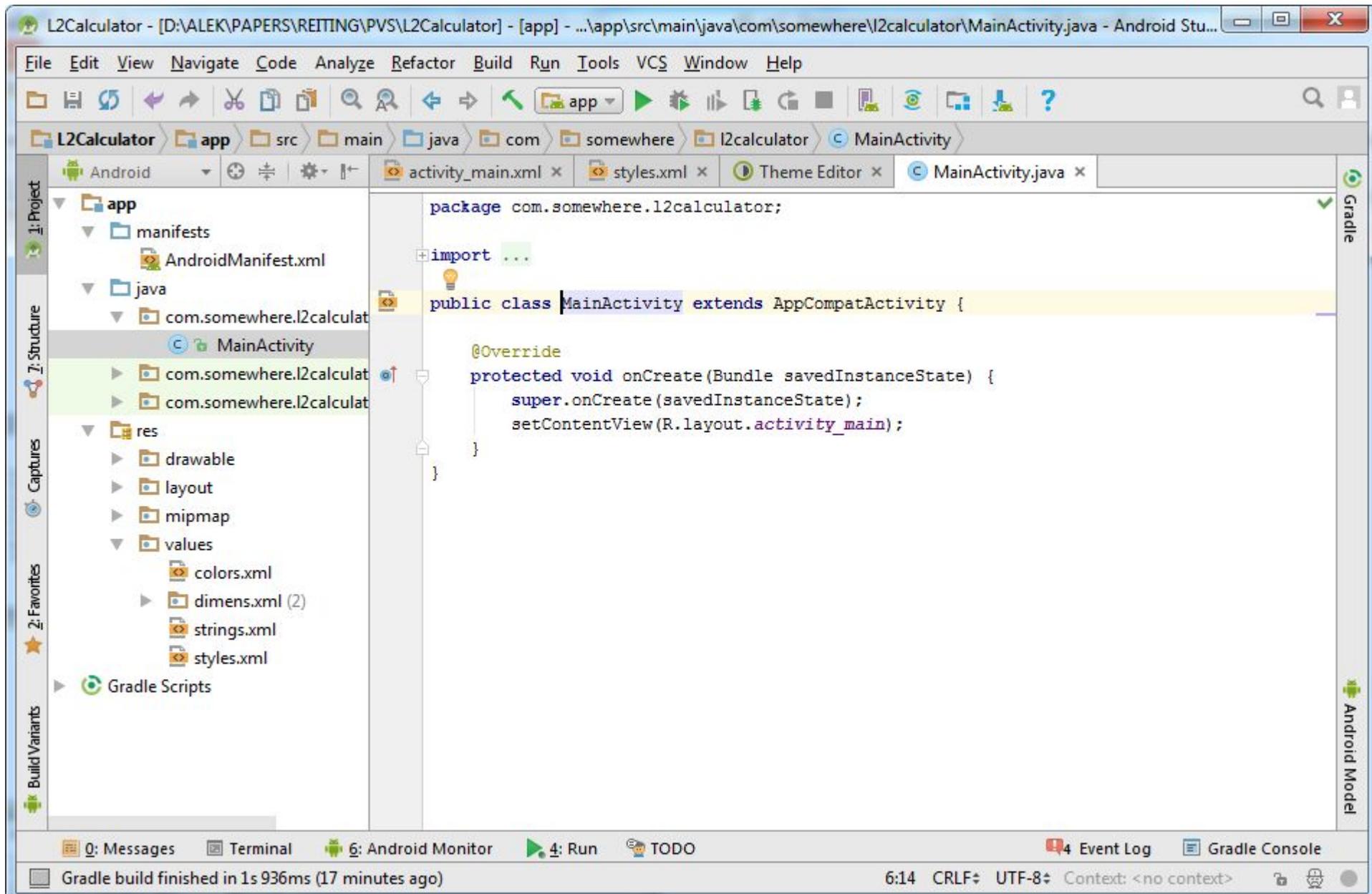
Построение графического интерфейса



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



The screenshot displays an IDE window for an Android project named "L2Calculator". The main editor shows the source code for MainActivity.java, which is a Java class extending AppCompatActivity. The code includes package declarations, imports, and an overridden onCreate method that calls super.onCreate and setContentView.

```
package com.somewhere.l2calculator;

import ...

public class MainActivity extends AppCompatActivity {

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

The left sidebar shows the project structure, including the 'app' folder, 'manifests' (AndroidManifest.xml), 'java' (com.somewhere.l2calculator), 'res' (drawable, layout, mipmap, values), and 'Gradle Scripts'. The bottom status bar indicates "Gradle build finished in 1s 936ms (17 minutes ago)" and shows system information like "6:14 CRLF UTF-8 Context: <no context>".

Класс MainActivity

```
activity_main.xml x  styles.xml x  Theme Editor x  MainActivity.java x  
  
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

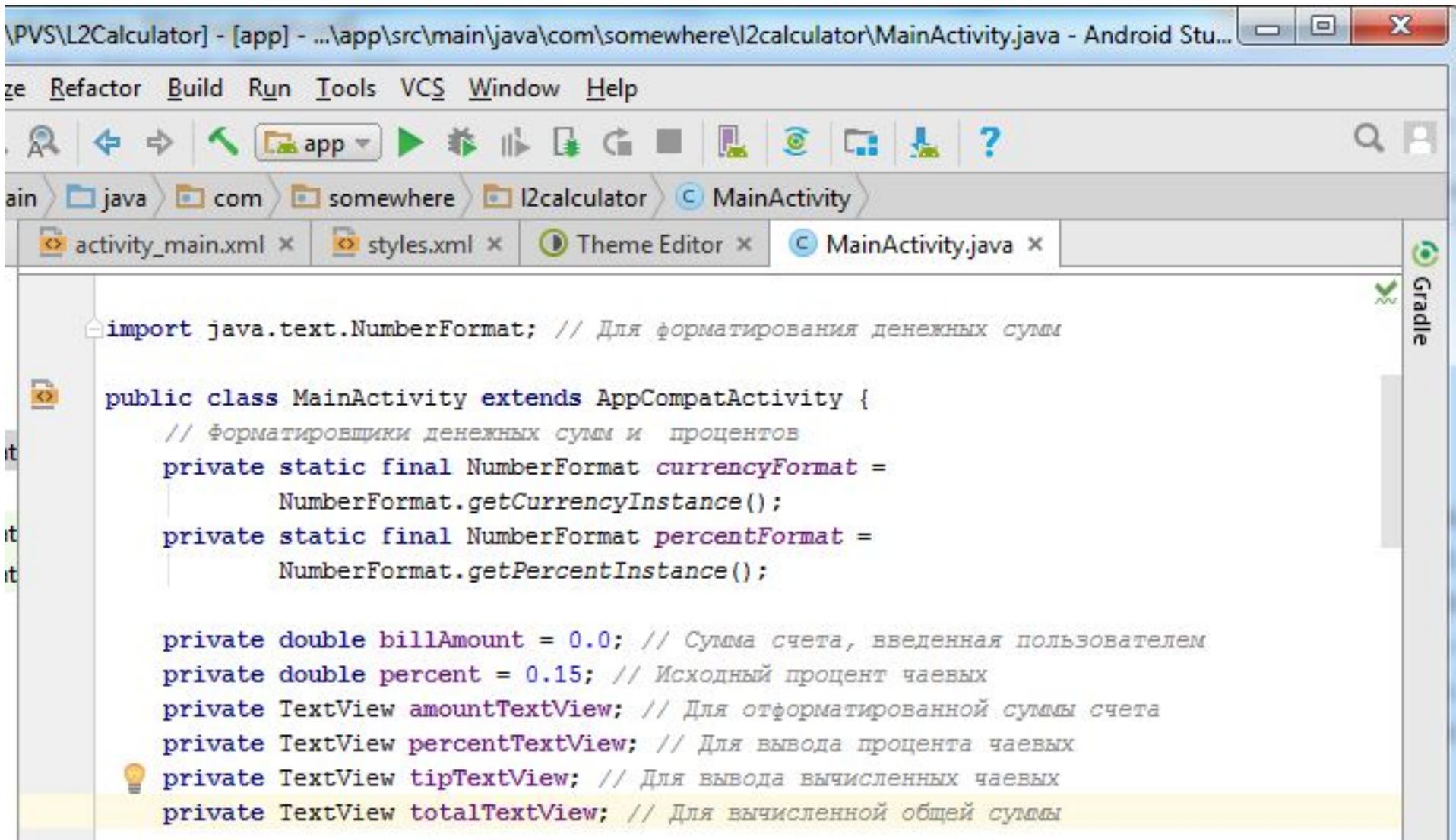
- Пакет `android.widget` содержит *виджеты* (визуальные компоненты) и макеты, используемые в графическом интерфейсе Android. В приложении используются виджеты `EditText`, `SeekBar` и `TextView`.
- Интерфейс `SeekBar.OnSeekBarChangeListener` из пакета `android.widget` реализуется для обработки событий,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

Класс MainActivity

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

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

Поля класса MainActivity



The screenshot shows an IDE window for MainActivity.java. The code defines several private fields: currencyFormat, percentFormat, billAmount, percent, amountTextView, percentTextView, tipTextView, and totalTextView. Each field is accompanied by a comment in Russian explaining its purpose.

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

public class MainActivity extends AppCompatActivity {
    // ФОРМАТИРОВЩИКИ денежных сумм и процентов
    private static final NumberFormat currencyFormat =
        NumberFormat.getCurrencyInstance();
    private static final NumberFormat percentFormat =
        NumberFormat.getPercentInstance();

    private double billAmount = 0.0; // Сумма счета, введенная пользователем
    private double percent = 0.15; // Исходный процент чаевых
    private TextView amountTextView; // Для отформатированной суммы счета
    private TextView percentTextView; // Для вывода процента чаевых
    private TextView tipTextView; // Для вывода вычисленных чаевых
    private TextView totalTextView; // Для вычисленной общей суммы
}
```

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.12calculator">  
  
    <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>
```

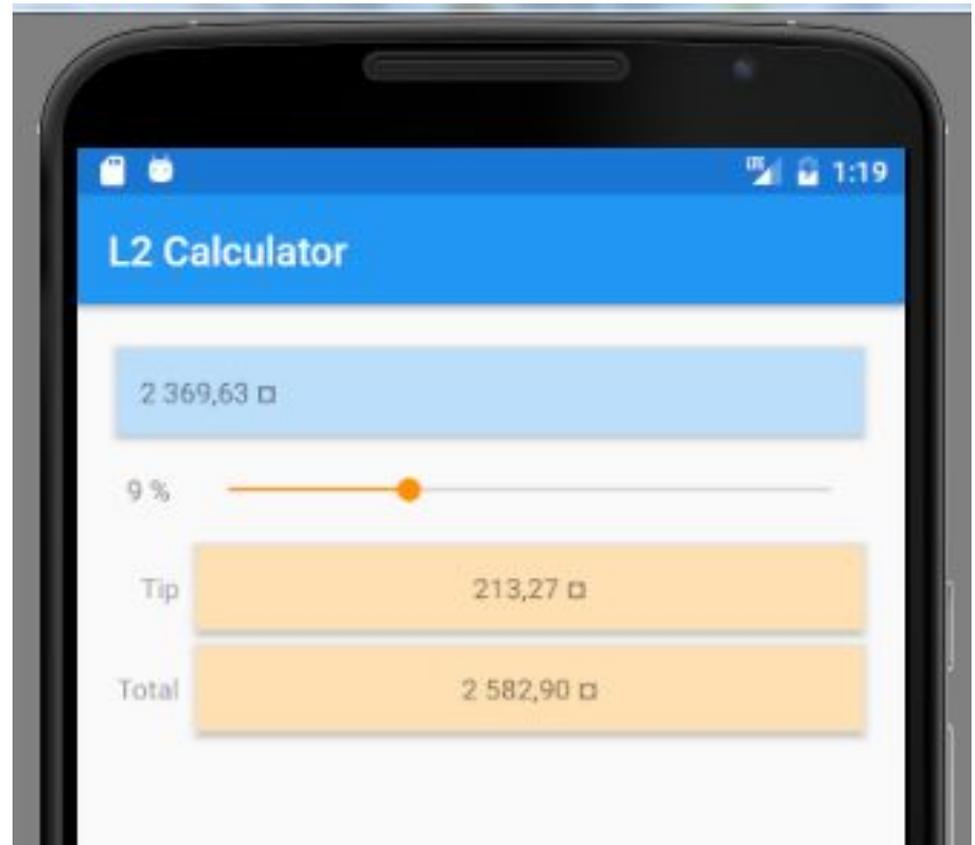
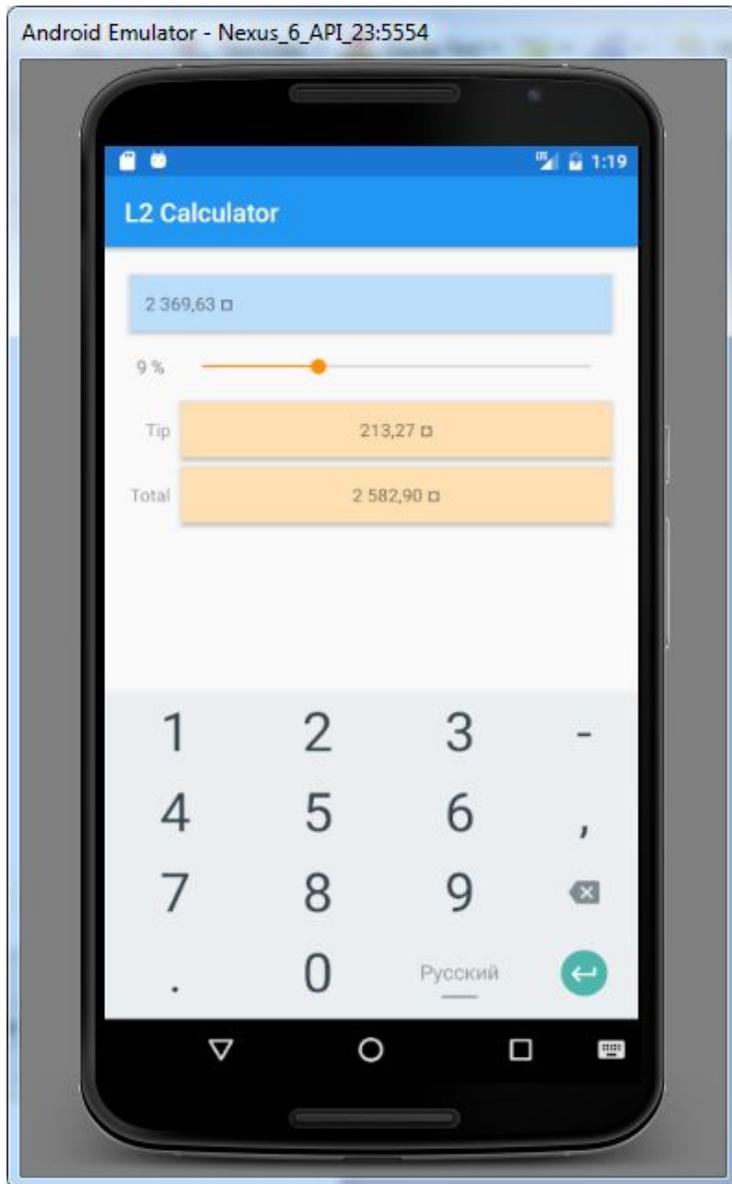
- `android:name` — имя класса активности (можно полностью `com.somewhere.l2calculator.MainActivity`)
- `android:screenOrientation` — переход устройства в альбомную ориентацию на типичном телефоне приведет к тому, что цифровая клавиатура закроет большую часть графического интерфейса. По этой причине задаем 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

Show only keys needing translations
 [Order a translation...](#)

Key	Default Value	Untra...	Russian (ru)
app_name	L2 Calculator	<input checked="" type="checkbox"/>	<u>L2 Calculator</u>
enter_amount	Enter Amount	<input type="checkbox"/>	Счёт
tip	Tip	<input type="checkbox"/>	Чаевые
tip_percentage	15%	<input type="checkbox"/>	Процент
total	Total	<input type="checkbox"/>	Общая сумма

Key:

Default Value:

Translation:

Локализация

