МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КІБЕРБЕЗПЕКИ

ЗВІТ ПРО ВИКОНАННЯ
ЛАБОРАТОРНОЇ РОБОТИ №2

із дисципліни «Програмування для мобільних пристроїв»

Виконав студент групи КБ-01
Д.О. Борщ

Перевірив В.В. Нагорний

Суми – 2022

Лабораторна робота 2

Завдання No1. Your first interactive UI (Task 1–6).

Task 1: Create and explore a new project

Для початку роботи відкриваємо Android Studio та створюємо пустий проект за параметрами, що вказані в туторіалі.

1.1 Create the Android Studio project

1. Start Android Studio and create a new project with the following parameters:

Attribute	Value
Template	**Empty Activity**
Name	**Hello Toast**
Package name	**com.example.hellotoast**
Language	**Java**
Minimum SDK	**API 21: Android 5.0 (Lollipop)**
Use legacy android.support libraries	Unchecked

Рис. 1 - Задані параметри.

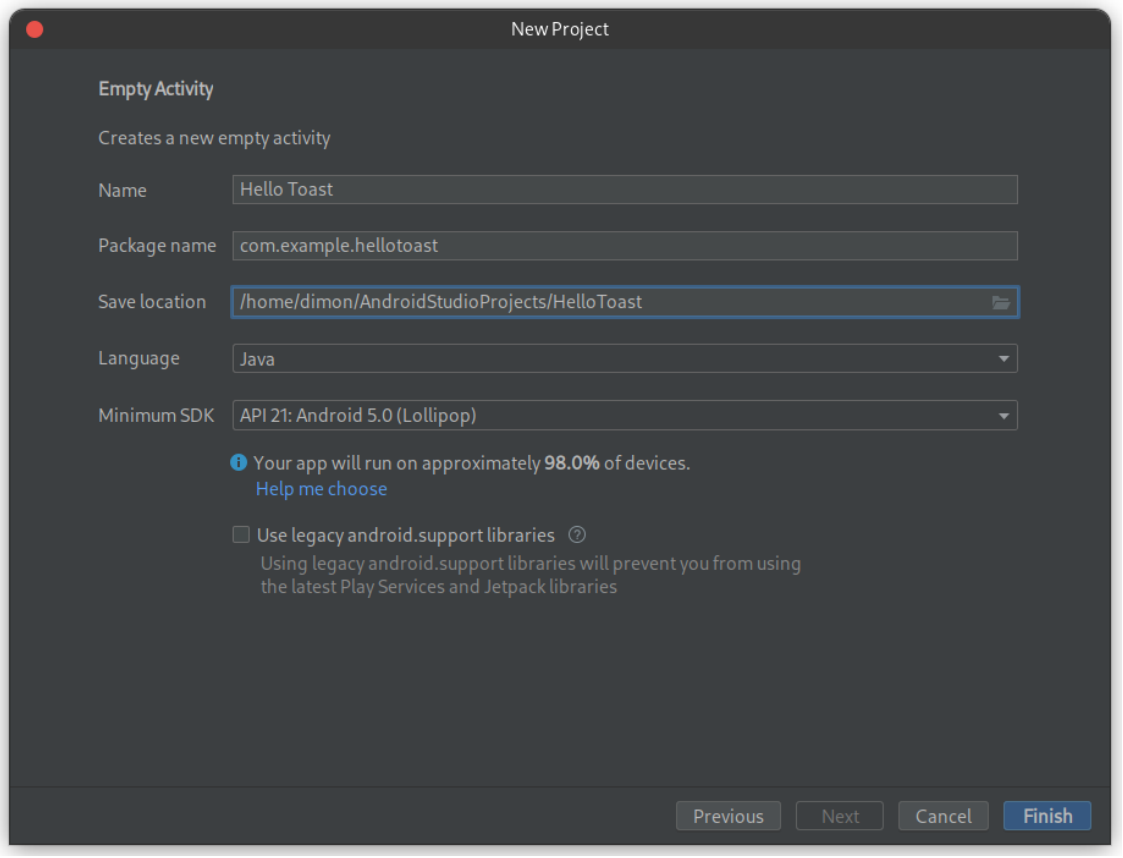


Рис. 2 - Створення нового проекту.

Task 2: Add View elements in the layout editor

2.1 Examine the element constraints

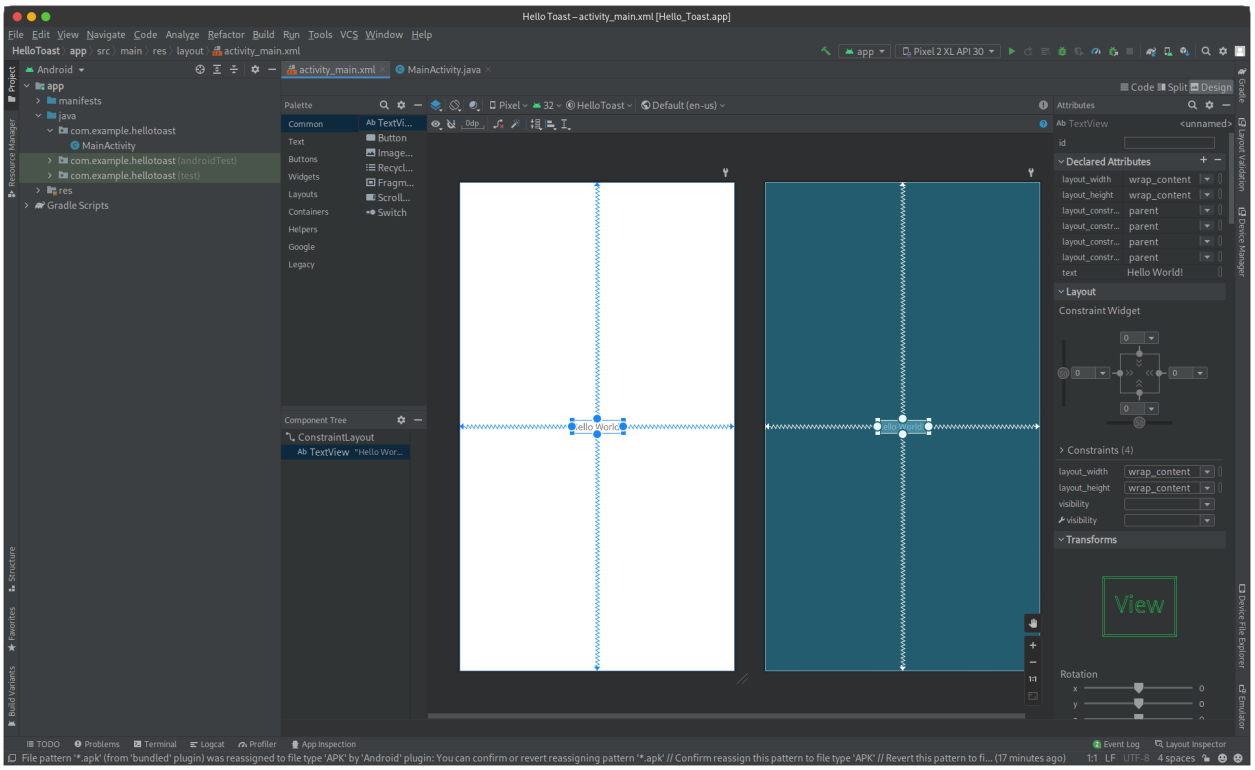


Рис. 3 - Досліджуємо constraints елемента.

2.2 Add a Button to the layout

За вказівками видалимо TextView та додамо кнопку, що буде прив’язана до верху нашого лайоуту та центрована по горизонталі.

2.3 Add a second Button to the layout

Додаємо другу кнопку до лайоуту, але прив’язуємо її до нижнього краю.

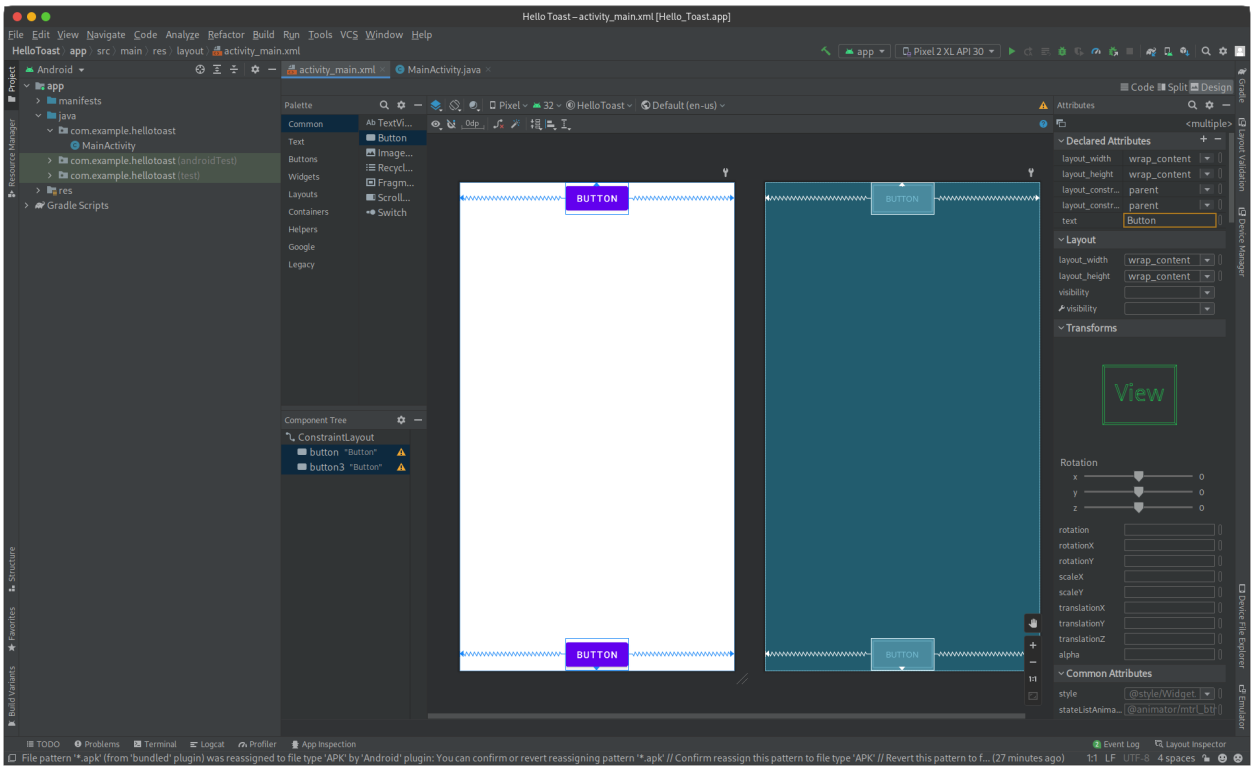


Рис. 4 - Кнопки з налаштованими constraints.

Task 3: Change UI element attributes

3.1 Change the Button size
Follow these steps:

1. Select the top Button in the Component Tree pane.
2. Click the Attributes tab on the right side of the layout editor window.
3. Click the width control twice—the first click changes it to Fixed with straight lines, and the second click changes it to Match Constraints with spring coils, as shown in the animated figure below.

As a result of changing the width control, the layout_width attribute in the Attributes pane shows the value match_constraint and the Button element stretches horizontally to fill the space between the left and right sides of the layout.

4. Select the second Button, and make the same changes to the layout_width as in the previous step, as shown in the figure below.

Зробимо обидві кнопки по ширині лайоуту за наведеною вище інструкцією.

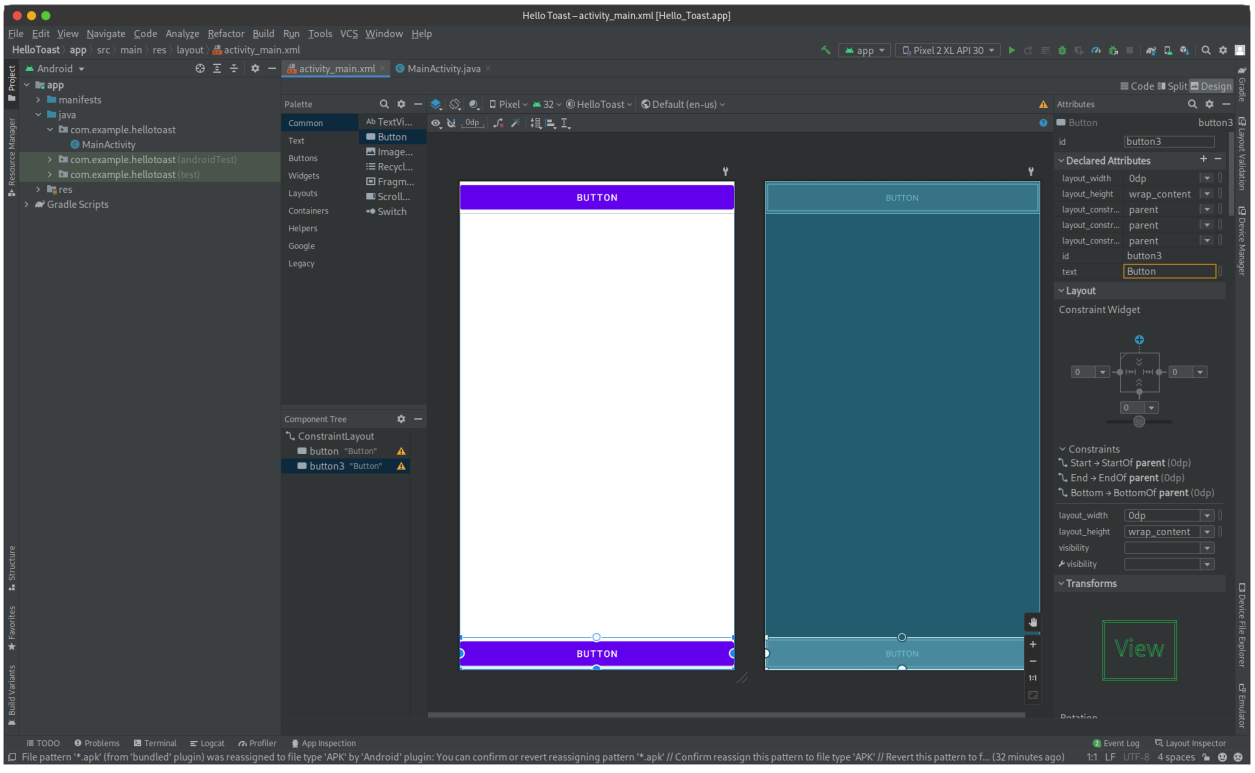


Рис. 5 - Змінення розміру кнопок.

3.2 Change the Button attributes
Perform these steps:

1. After selecting the first Button, edit the ID field at the top of the Attributes pane to **button_toast** for the android:id attribute, which is used to identify the element in the layout. When the Rename dialog box appears, select Refactor. This changes all instances of the ID across the project to the new name.
2. Set the backgroundTint attribute to **@color/purple_200**. (As you enter @c, choices appear for easy selection.)

Tip: If you don't see an attribute you're looking for, try using the search function as demonstrated in the animated figure below.

3. Set the textColor attribute to **@android:color/black.**
4. Edit the text attribute to **Toast**.

Changing other UI element attributes

5. Perform the same attribute changes for the second Button, using button_count as the ID, Count for the text attribute, and the same colors for the backgroundTint and text as the previous steps.

Змінимо ID, підписи та кольори кнопок за наведеними вище вказівками.

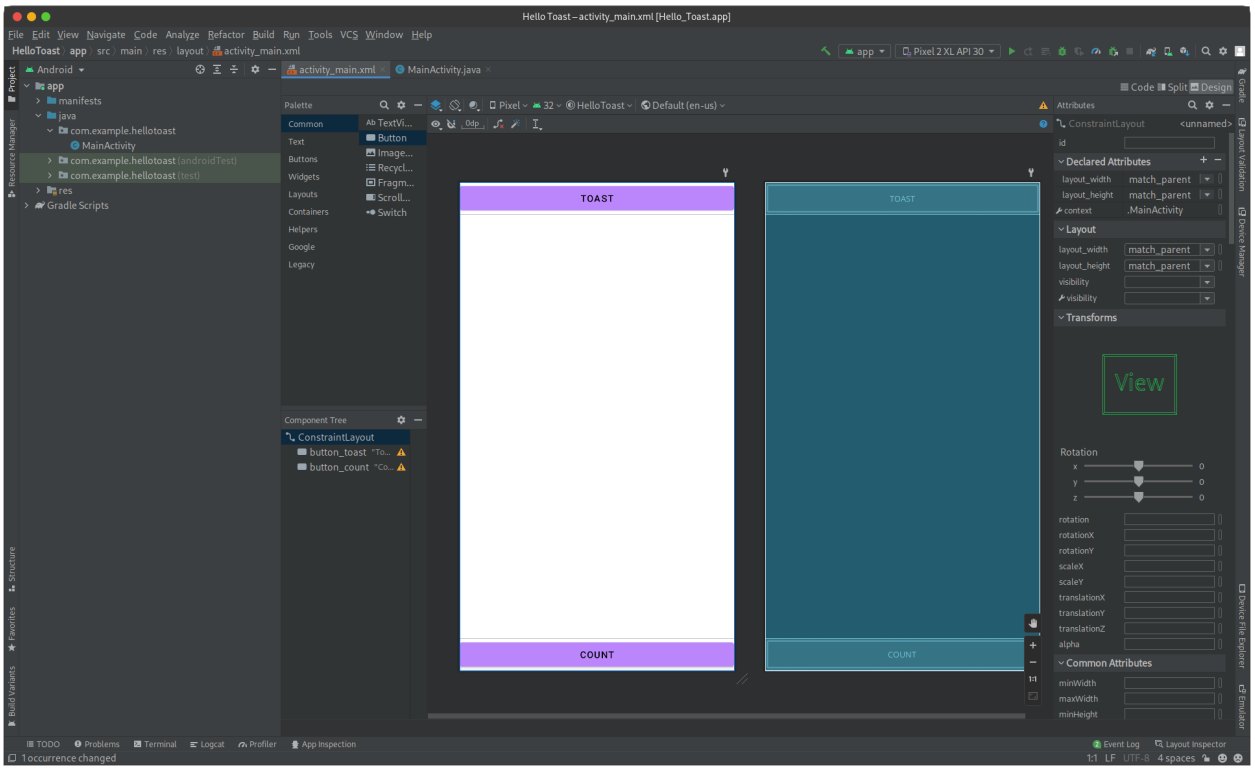


Рис. 6 - Змінення атрибутів кнопок.

Task 4: Add a TextEdit and set its attributes

4.1 Add a TextView and constraints

Drag a TextView from the Palette pane to the upper part of the layout, and drag a constraint from the top of the TextView to the handle on the bottom of the Toast Button. This constrains the TextView to be underneath the Button

Додамо TextView та налаштуємо constraints за вказівкою.

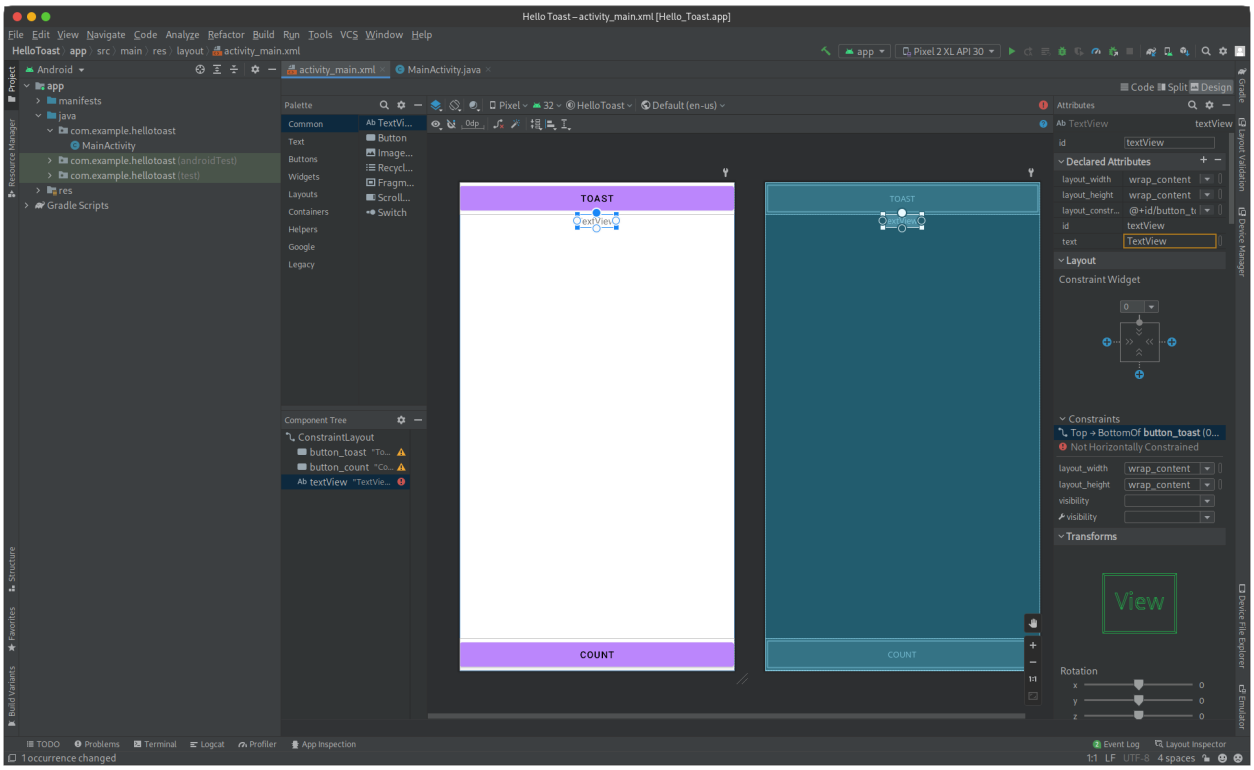


Рис. 7 - Додаємо TextView.

Drag a constraint from the bottom of the TextView to the handle on the top of the Count Button, and from the sides of the TextView to the sides of the layout. This constrains the TextView to be in the middle of the layout between the two Button elements.

Центруємо TextView за допомогою constraints.

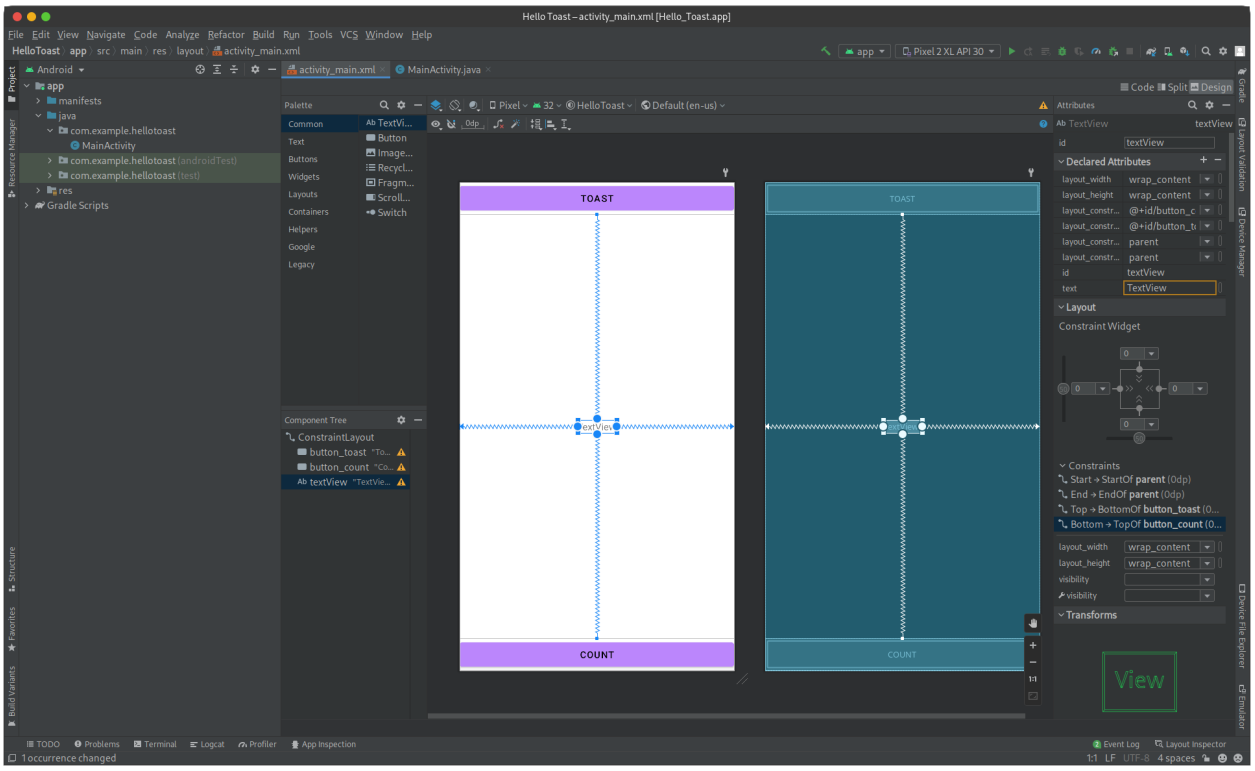


Рис. 8 - Налаштовані constraints для TextView.

4.2 Set the TextView attributes

With the TextView selected, open the **Attributes** pane, if it is not already open. Set attributes for the TextView as described below. The attributes you haven't encountered yet are explained after the instructions:

1. Set the ID to **show_count**.
2. Change the horizontal and vertical view size controls (layout_width and layout_height) to **match_constraint**.
3. Set the text to **0**.
4. Scroll down the pane to **Common Attributes** and expand the textAppearance attribute.
5. Set the textSize to **160sp**.
6. Set the textColor to **@color/purple_500**.
7. Set the textStyle to **B** (bold).
8. Scroll down the pane to **All Attributes**, and set the background to **#FFFF00** for a shade of yellow.
9. Scroll down to gravity, expand gravity, and select **center**.

Налаштуємо атрибути TextView за вказівками, поданими вище.

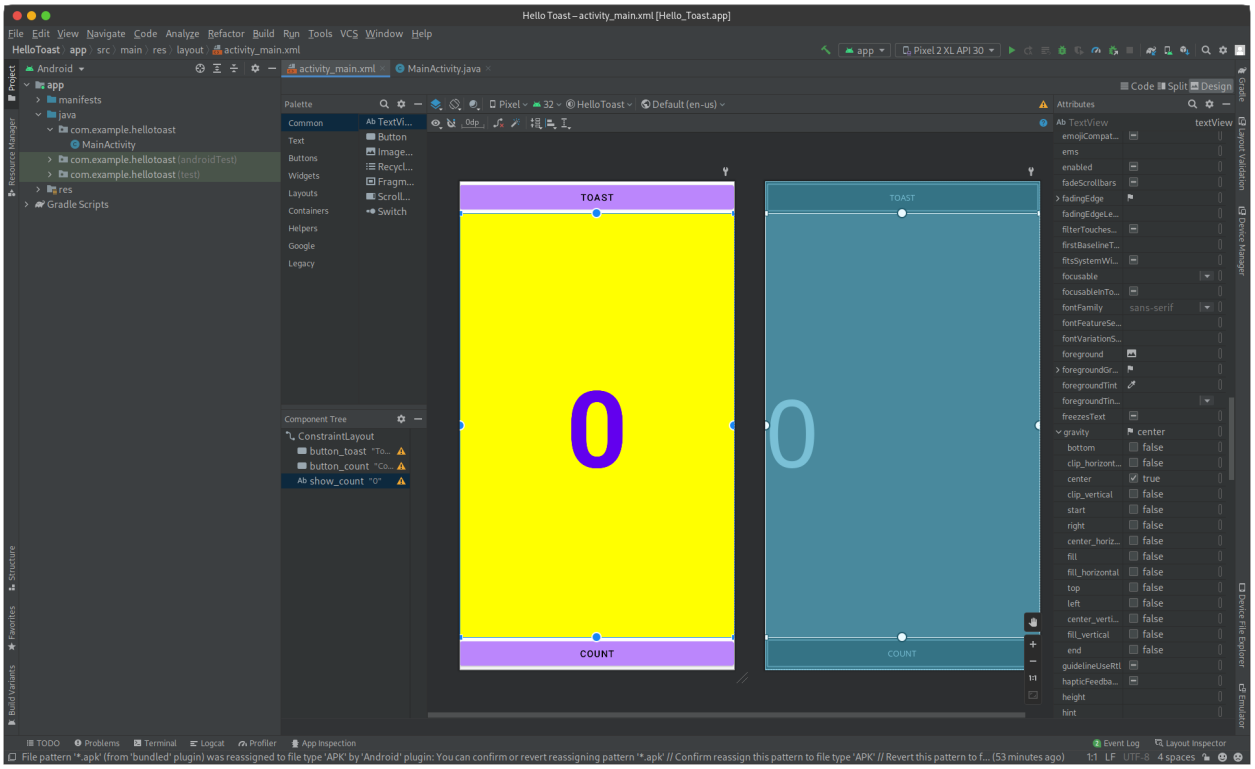


Рис. 9 - Налаштовані атрибути TextView.

Task 5: Edit the layout in XML

5.1 Open the XML code for the layout

Відкриваємо activity_main.xml в представленні Code.

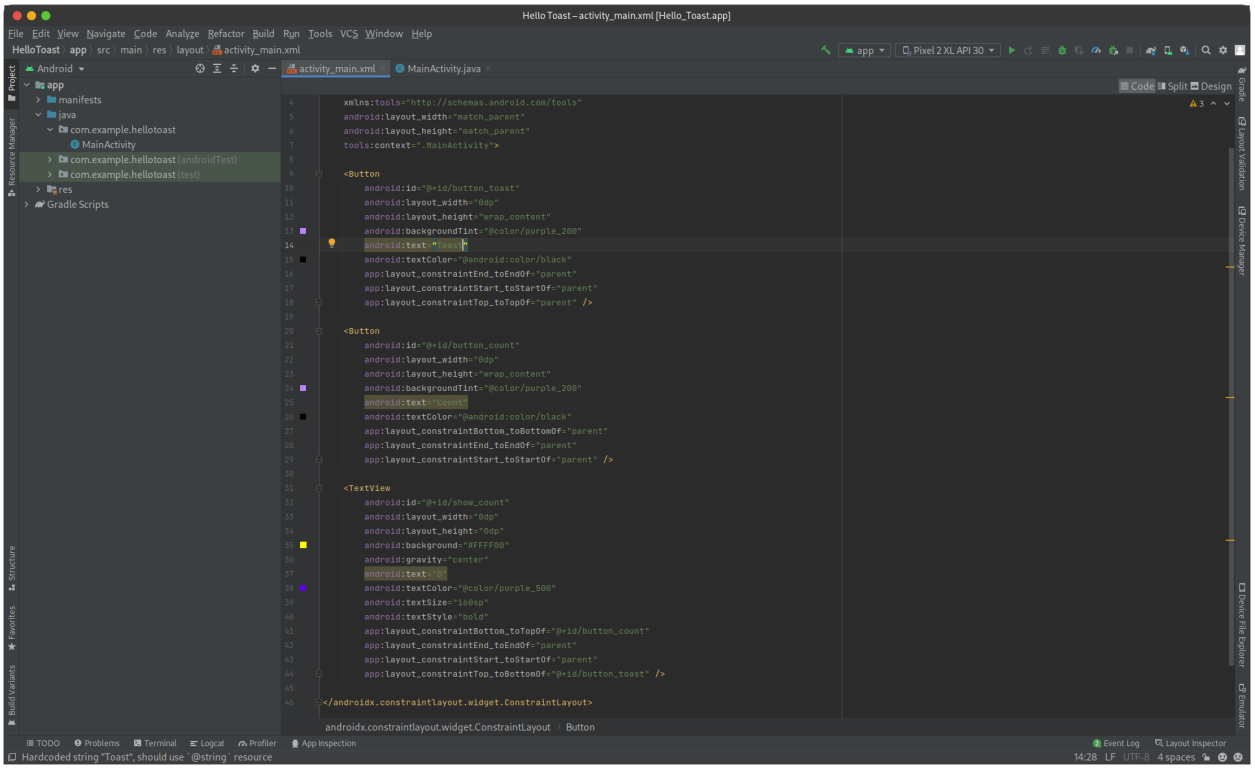


Рис. 10 - Файл activity_main.xml в представленні Code.

5.2 Extract string resources

1. Click once on the word "Toast"(the first highlighted warning).

2. Click on the lightbulb that appears at the beginning of the line  and select **Extract string resource** from the popup menu.
3. Enter **button_label_toast** for the **Resource name**.
4. Click **OK**. A string resource is created in the res/values/strings.xml file, and the string in your code is replaced with a reference to the resource
5. Extract the remaining strings: button_label_count for "Count", and count_initial_value for "0".
6. In the **Project > Android** pane, expand **values** within **res**, and then double-click **strings.xml** to see your string resources in the strings.xml file.
7. You need another string to use in a subsequent task that displays a message. Add to the strings.xml file another string resource named toast_message for the phrase "Hello Toast!".

За вказівками додаємо записи в strings.xml та замінюємо hardcoded текст на посилання на ці ресурси.

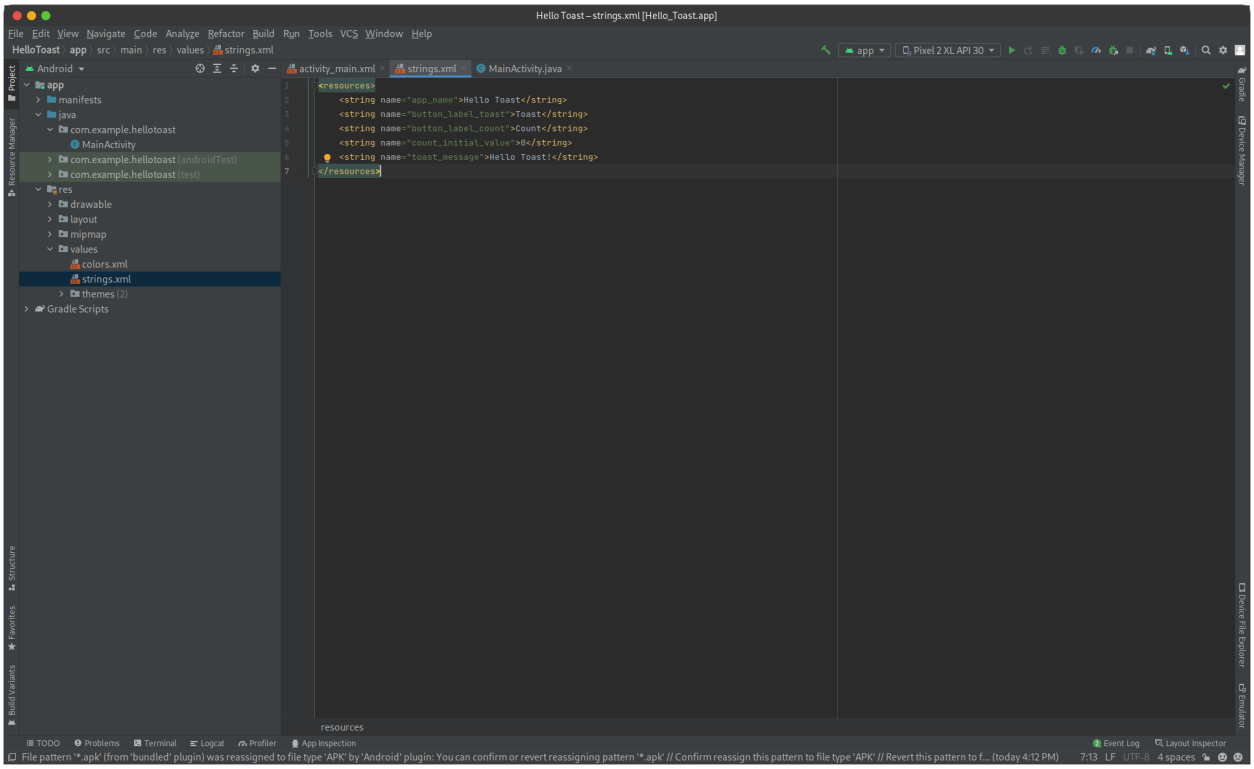


Рис. 11 - Файл strings.xml.

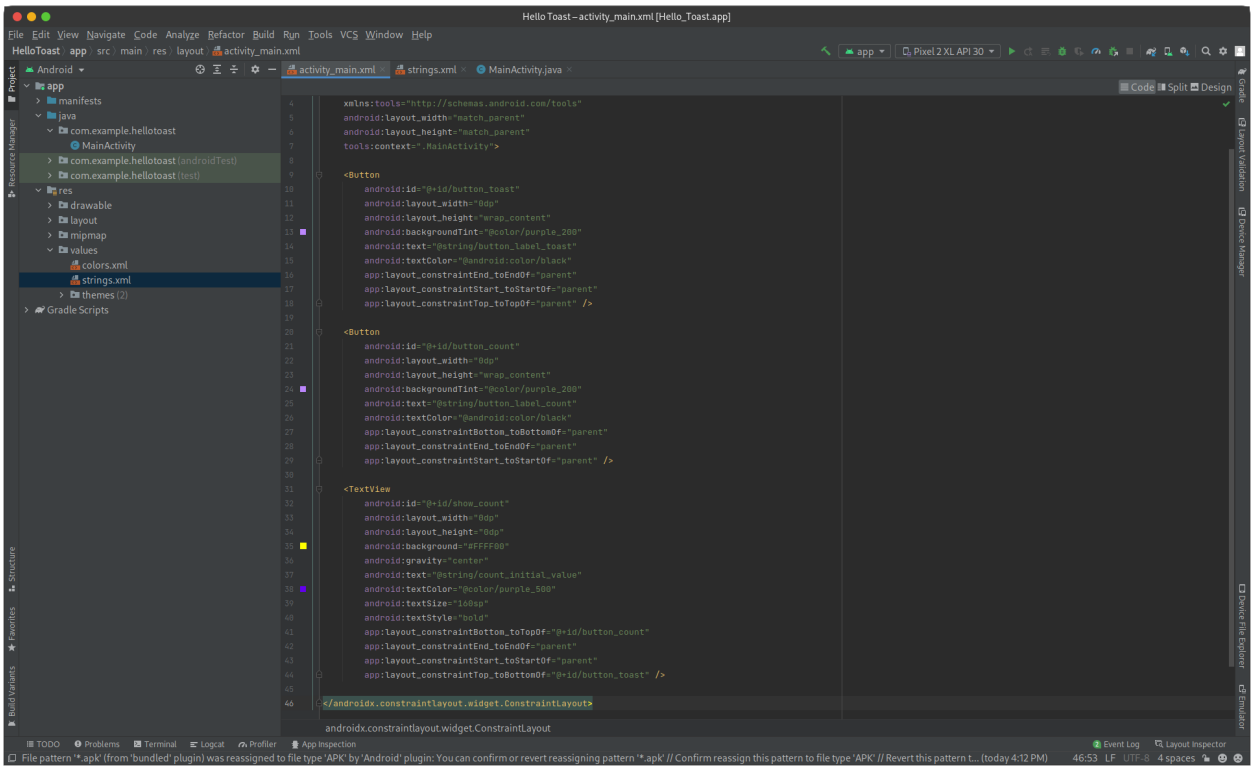


Рис. 12 - Виправлений файл activity_main.xml.

Task 6: Add onClick handlers for the buttons

6.1 Add the onClick attribute and handler to each Button

1. With the XML editor open (the Code tab), find the Button with the android:id set to button_toast.
2. Add the android:onClick attribute to the end of the button_toast element after the last attribute and before the /> end indicator:
3. Click the red bulb icon that appears next to the attribute. Select **Create ‘showToast(View)' in ‘MainActivity'**.

If the red bulb icon doesn't appear, click the method name ("showToast"). Press **Alt-Enter** (**Option-Enter** on the Mac), select **Create ‘showToast(view)' in MainActivity**, and click **OK**.

This action creates a placeholder method stub for the showToast() method in MainActivity, and opens MainActivity.java.

4. Return to activity_main.xml and repeat the last two steps with the button_count Button. Add the android:onClick attribute to the end, and add the click handler.

За вказівками додамо атрибути onClick до кнопок та створимо для початку пусту реалізацію цих функцій.

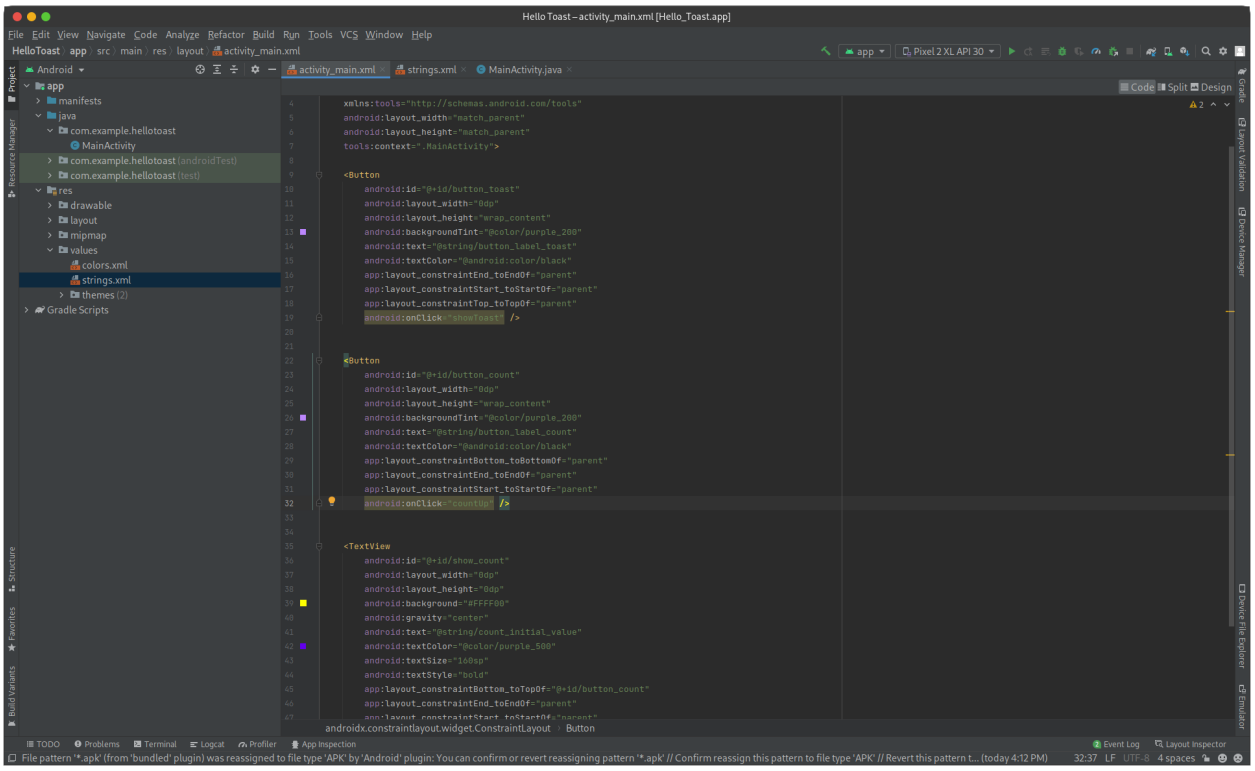


Рис. 13 - Додавання атрибутів.

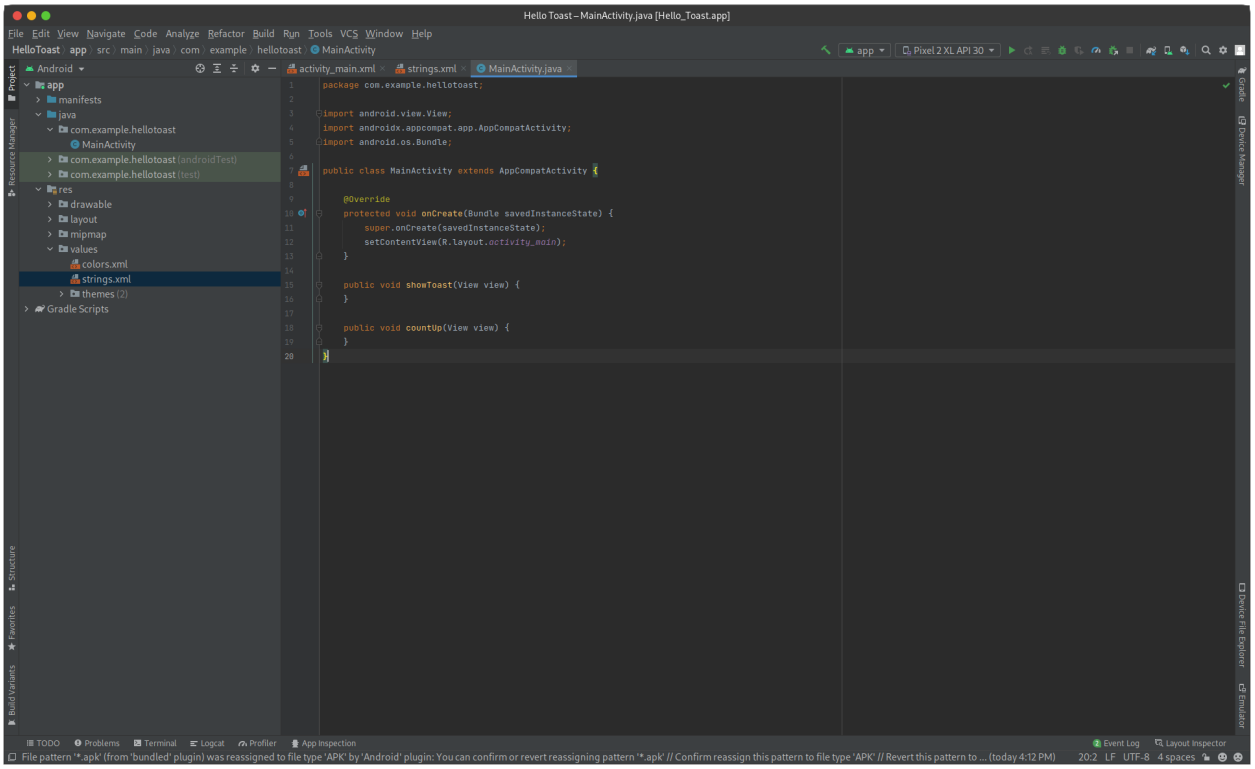


Рис. 14 - Реалізація функцій хендлерів.

6.2 Edit the Toast Button handler

Follow these steps to edit the **Toast** Button click handler:

1. Locate the newly created showToast() method.
2. To create an instance of a Toast, call the makeText() factory method on the Toast class.

This statement is incomplete until you finish all of the steps.

3. Supply the context of the app Activity. Because a Toast displays on top of the Activity UI, the system needs information about the current Activity. When you are already within the context of the Activity whose context you need, use this as a shortcut.
4. Supply the message to display, such as a string resource (the toast_message you created in a previous step). The string resource toast_message is identified by R.string.
5. Supply a duration for the display. For example, Toast.LENGTH_SHORT displays the toast for a relatively short time.

The duration of a Toast display can be either Toast.LENGTH_LONG or Toast.LENGTH_SHORT. The actual lengths are about 3.5 seconds for the long Toast and 2 seconds for the short Toast.

6. Show the Toast by calling show(). The following is the entire showToast() method

Run the app and verify that the Toast message appears when the **Toast** button is tapped.

Створимо реалізацію хендлера кнопки Toast, яка буде викликати toast повідомлення, яка ми записали в resources раніше. Перевіримо її в емуляторі.

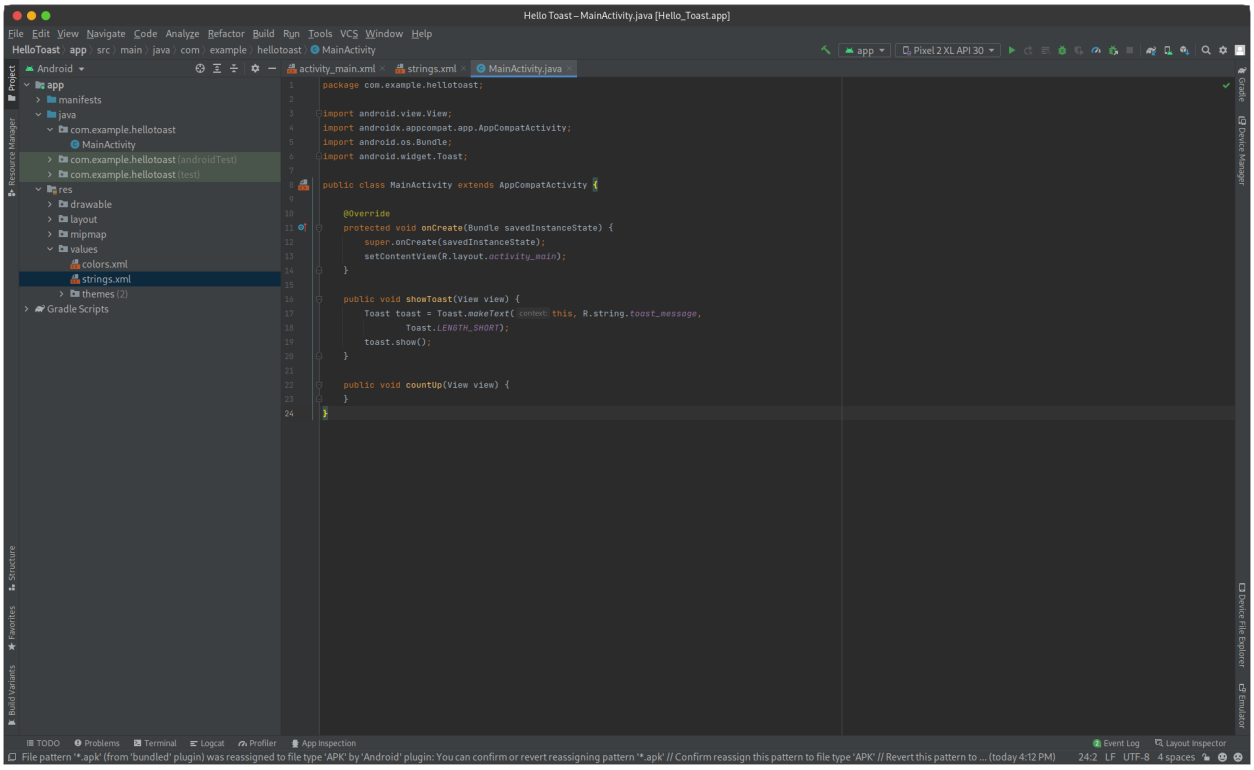


Рис. 15 - Реалізація функції showToast.

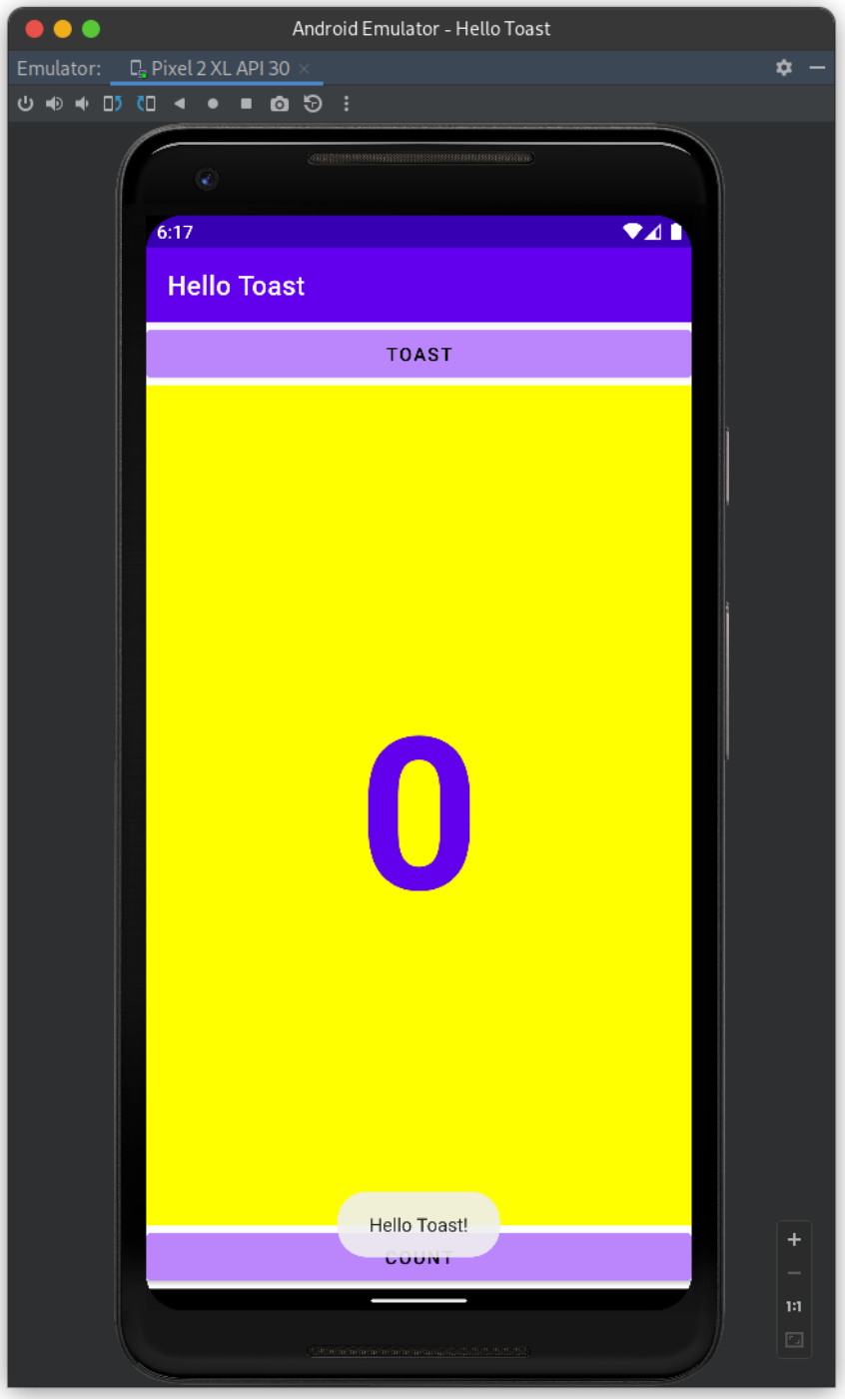


Рис. 16 - Перевірка роботи програми в емуляторі.

6.3 Edit the Count Button handler

Follow these steps to edit the **Count** Button click handler:

1. Locate the newly created countUp() method.
2. To keep track of the count, you need a private member variable. Each tap of the **Count** button increases the value of this variable. Enter the following, which will be highlighted in red and show a red bulb icon

If the red bulb icon doesn't appear, select the mCount++ expression. The red bulb eventually appears.

3. Click the red bulb icon and choose **Create field 'mCount' in 'MainActivity'** from the popup menu. This creates a private member variable at the top of MainActivity, and Android Studio assumes that you want it to be an integer (int).
4. Change the private member variable statement to initialize the variable to zero.
5. Along with the variable above, you also need a private member variable for the reference of the show_count TextView, which you will add to the click handler. Call this variable mShowCount.
6. Now that you have mShowCount, you can get a reference to the TextView using the ID you set in the layout file. In order to get this reference only once, specify it in the onCreate() method. As you will learn in another lesson, the onCreate() method is used to *inflate the layout*, which means to set the content view of the screen to the XML layout. You can also use it to get references to other UI elements in the layout, such as the TextView. Locate the onCreate() method in MainActivity
7. Add the findViewById statement to the end of the method.

A View, like a string, is a resource that can have an id. The findViewById call takes the ID of a view as its parameter and returns the View. Because the method returns a View, you have to cast the result to the view type you expect, in this case (TextView).

8. Now that you have assigned to mShowCount the TextView, you can use the variable to set the text in the TextView to the value of the mCount variable. Add the following to the countUp() method.

Користуючись вказівками реалізуємо метод countUp та перевіряємо його роботу.

Додамо глобальні змінні для лічильнику та TextView. Також допрацюємо метод, що викликається при ініціалізації класу.

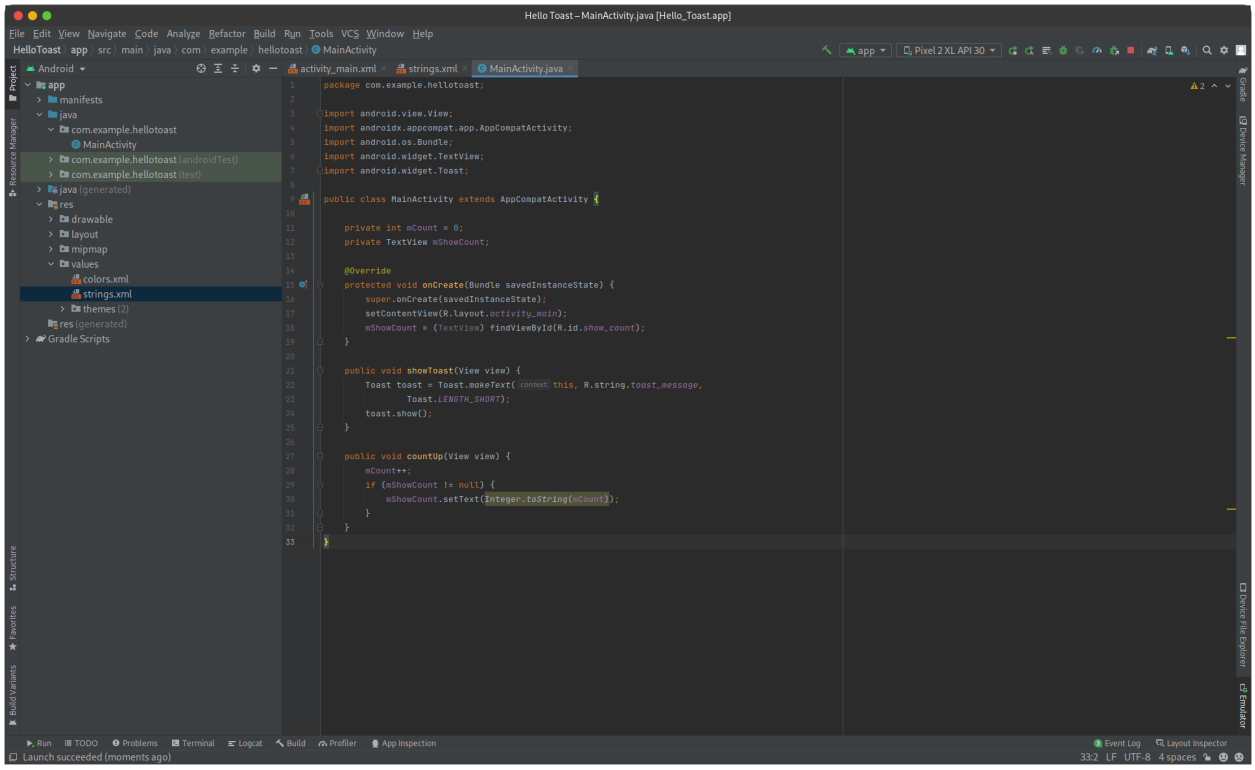


Рис. 17 - Допрацювання головного класу програми.

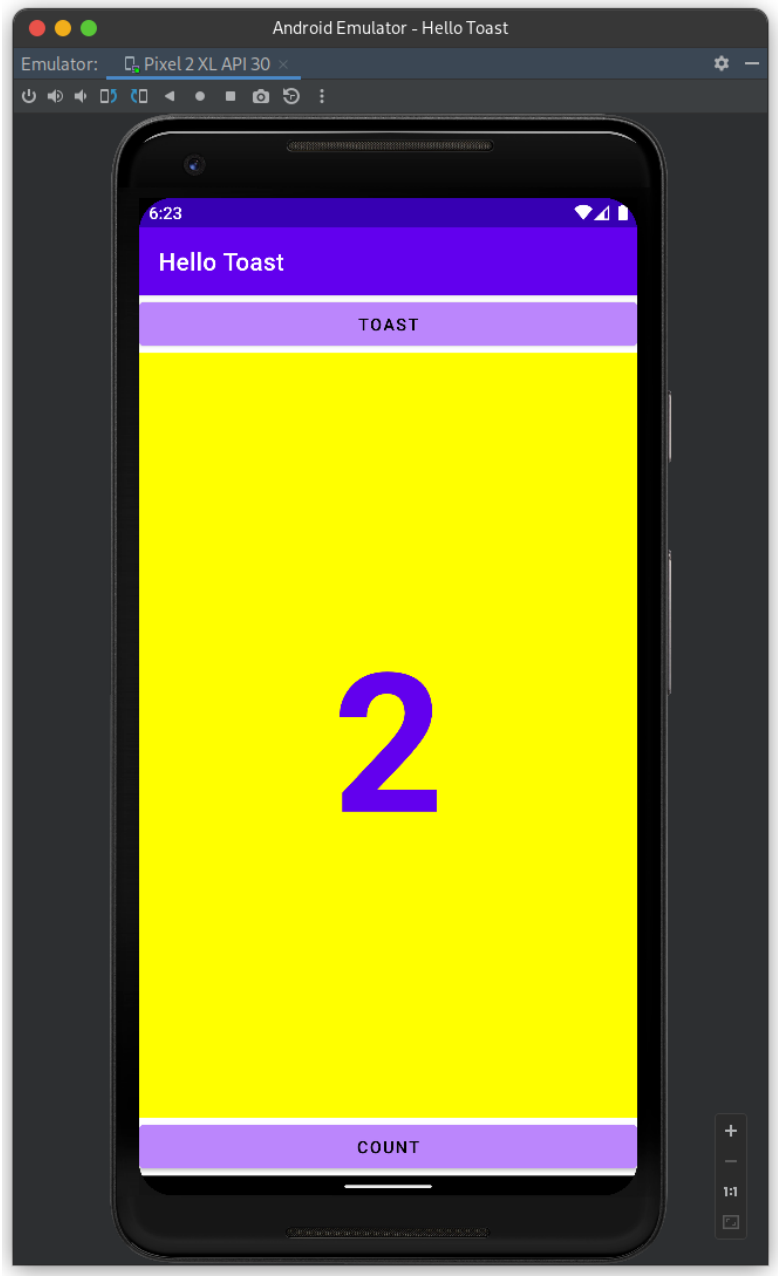


Рис. 18 - Демонстрація роботи програми.

Завдання No2. Part B: The layout editor

Task 1: Create layout variants

1.1 Preview the layout in a horizontal orientation

To preview the Hello Toast app layout with a horizontal orientation, follow these steps:

1. Open the Hello Toast app from the previous lesson.
2. Open the **activity_main.xml** layout file. Click the **Design** tab if it is not already selected.
3. Click the **Orientation in Editor** button  in the top toolbar.
4. Select **Switch to Landscape** in the dropdown menu. The layout appears in horizontal orientation as shown below. To return to vertical orientation, select **Switch to Portrait**.

Переглянемо наш лайоут в горизонтальному вигляді.

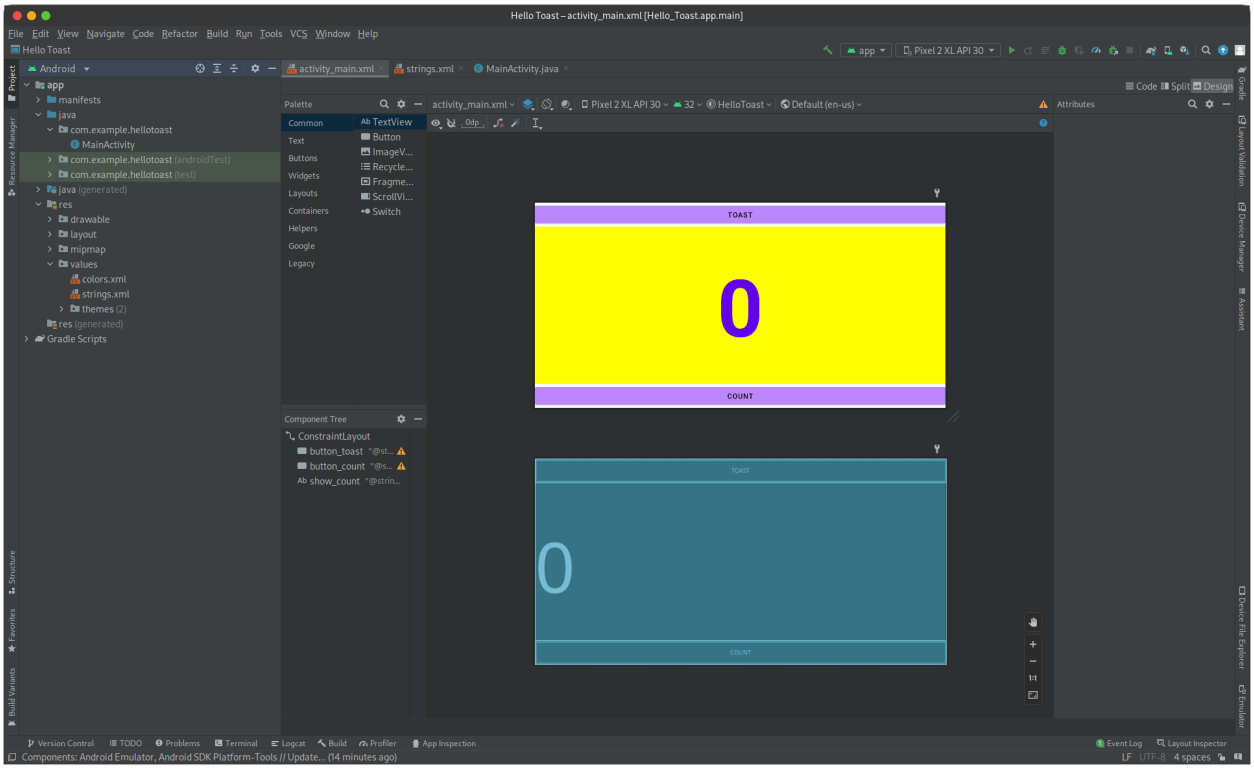


Рис. 19 - Горизонтальний лайоут

1.2 Create a layout variant for horizontal orientation

1. Click the **Orientation in Editor** button  in the top toolbar.
2. Choose **Create Landscape Variation**.

A new editor window opens with the **land/activity_main.xml** tab showing the layout for the landscape (horizontal) orientation. You can change this layout, which is specifically for horizontal orientation, without changing the original portrait (vertical) orientation.

3. In the **Project > Android** pane, look inside the **res > layout** directory, and you will see that Android Studio automatically created the variant for you, called activity_main.xml (land).

Створимо варіант лайоуту для ландшафтної орієнтації.

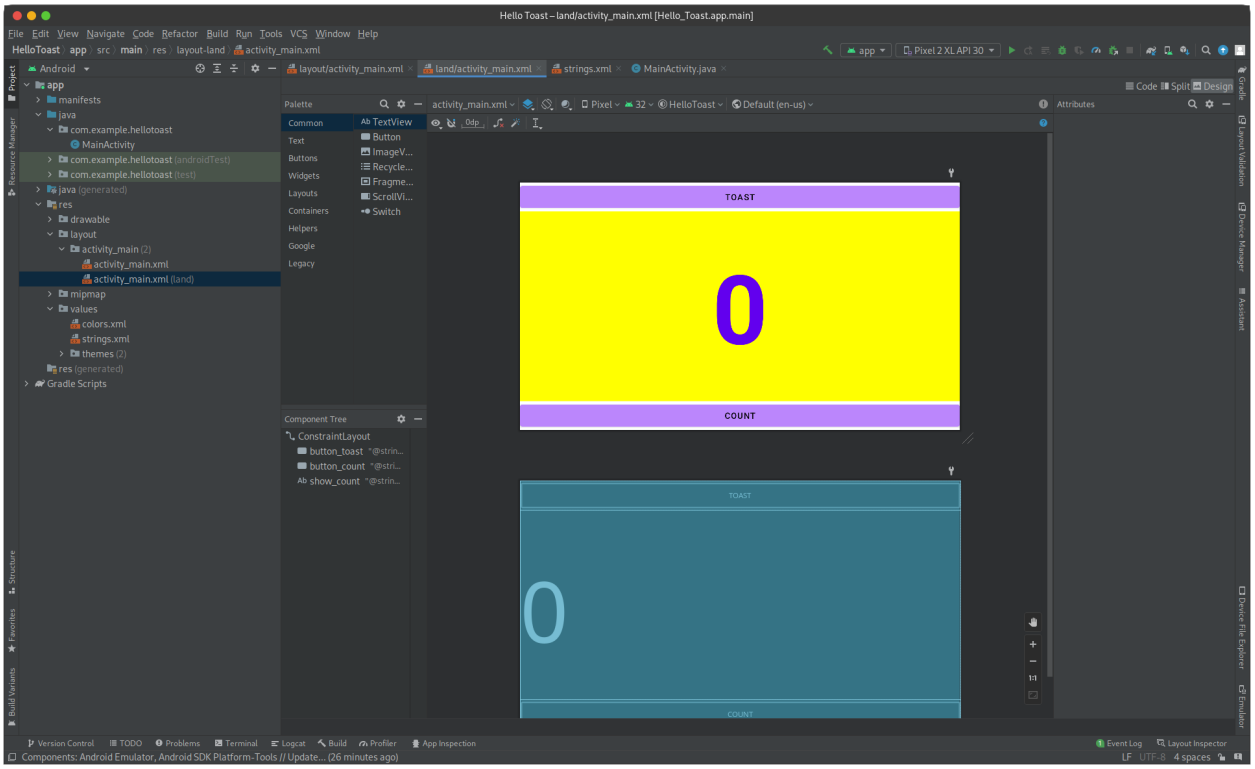
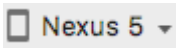


Рис. 20 - Ландшафтний варіант.

1.3 Preview the layout for different devices

You can preview the layout for different devices without having to run the app on the device or emulator. Follow these steps:

1. The **land/activity_main.xml** tab should still be open in the layout editor; if not, double-click the **activity_main.xml (land)** file in the **layout** directory.
2. Click the **Device in Editor** button 

in the top toolbar.
3. Choose a different device in the dropdown menu. For example, choose **Nexus 4**, **Nexus 5**, and then **Pixel** to see differences in the previews. These differences are due to the fixed text size for the TextView.

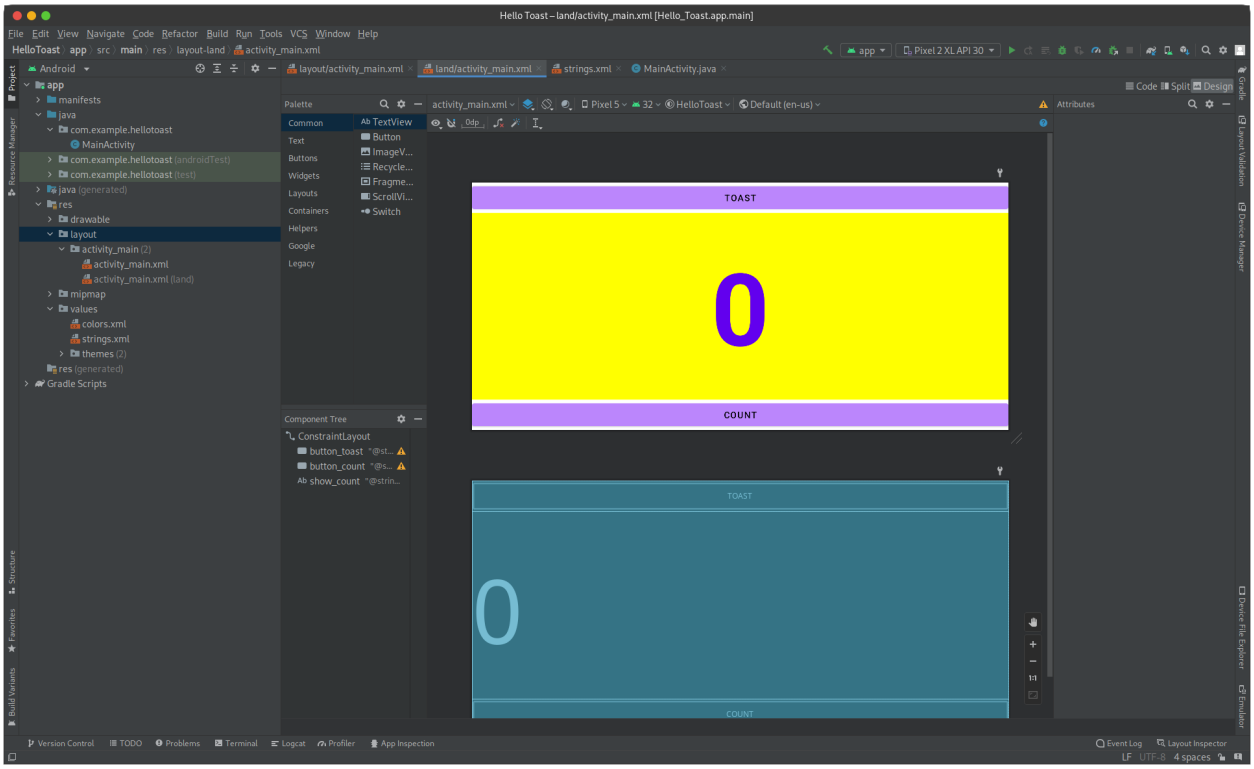


Рис. 21 - Вигляд на пристрої Pixel 5.

1.4 Change the layout for horizontal orientation
To change the layout, follow these steps:

1. The **land/activity_main.xml** tab should still be open in the layout editor; if not, double-click the **activity_main.xml (land)** file in the **layout** directory.
2. Click the **Text** tab and the **Preview** tab (if not already selected).
3. Find the TextView element in the XML code.
4. Change the android:textSize="160sp" attribute to android:textSize="120sp".
5. Choose different devices in the **Device in Editor** dropdown menu to see how the layout looks on different devices in horizontal orientation.

In the editor pane, the **land/activity_main.xml** tab shows the layout for horizontal orientation. The **activity_main.xml** tab shows the unchanged layout for vertical orientation. You can switch back and forth by clicking the tabs.

6. Run the app on an emulator or device, and switch the orientation from vertical to horizontal to see both layouts.

Змінимо лайоут для ландшафтного вигляду.

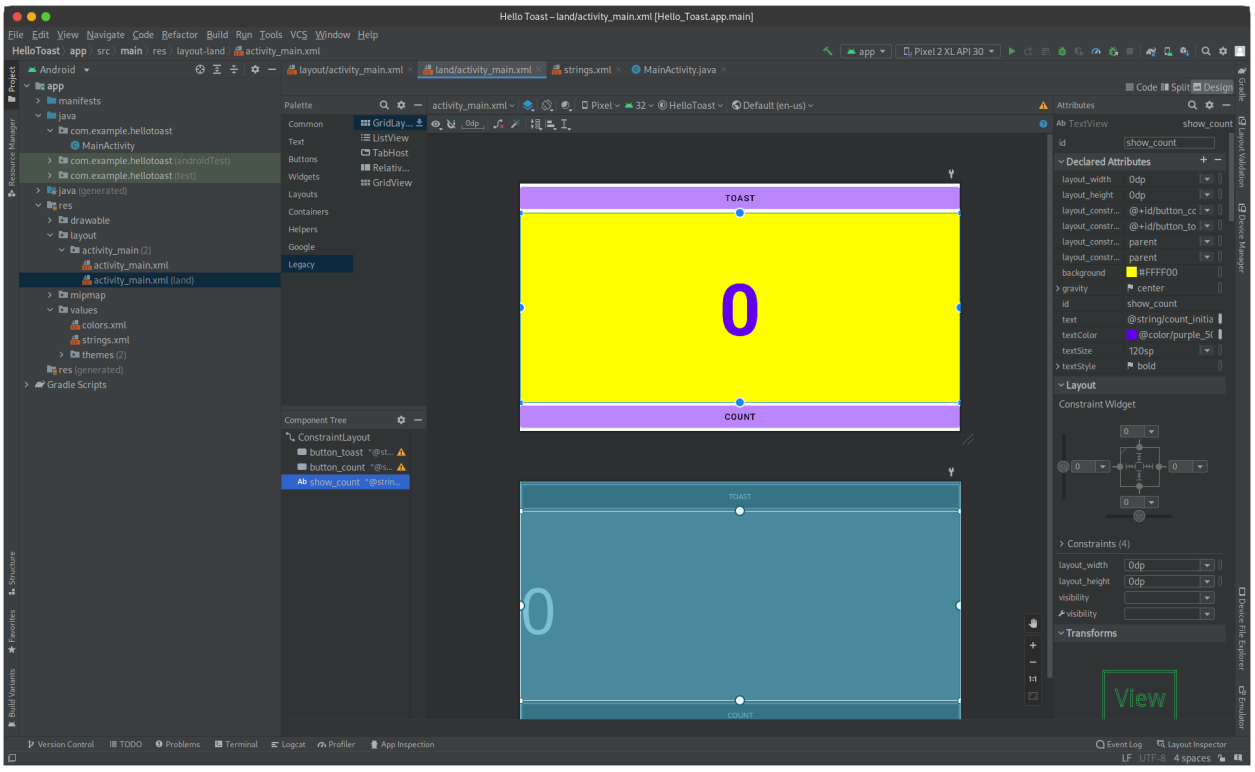


Рис. 22 - Зменшений текст для ландшафтного лайоуту.

1.5 Create a layout variant for tablets

To fix this for tablets while leaving the phone-size horizontal and vertical orientations alone, you can create variant of the layout that is completely different for tablets. Follow these steps:

1. Click the **Design** tab (if not already selected) to show the design and blueprint panes.
2. Click the **Orientation in Editor** button  in the top toolbar.
3. Choose **Create layout x-large Variation**.

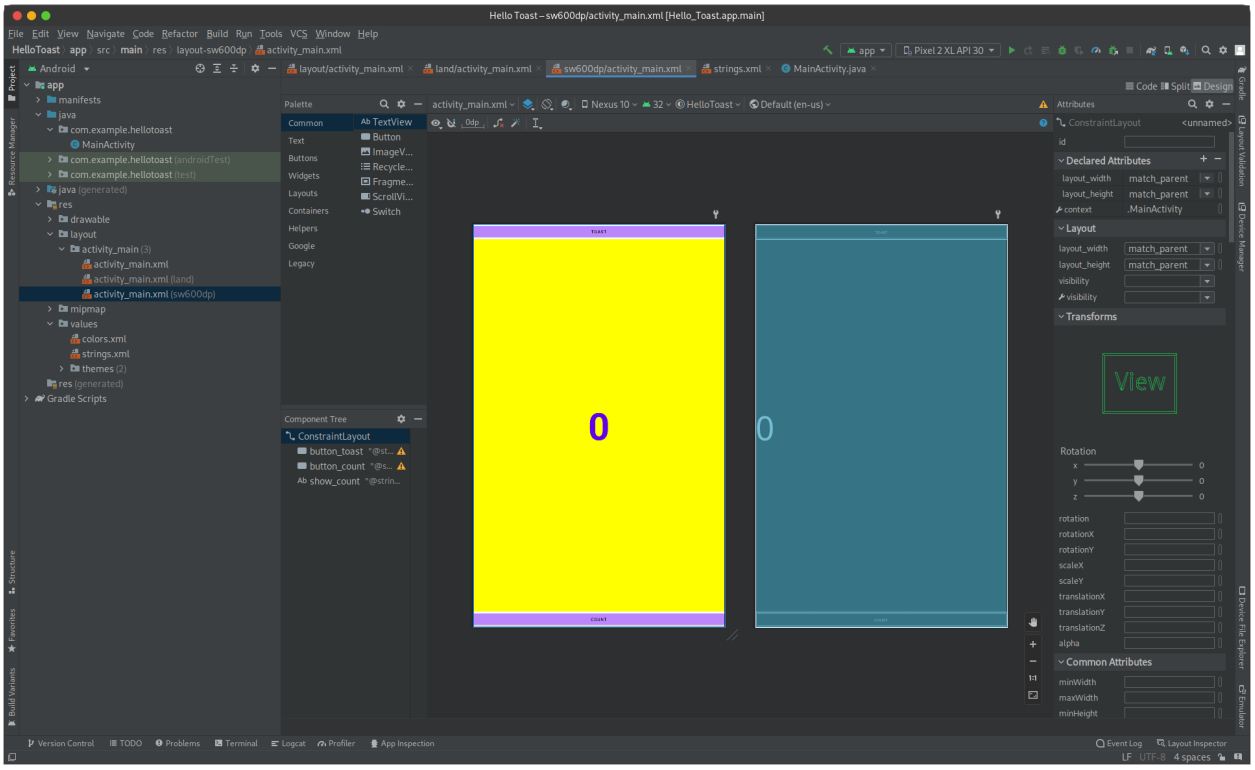


Рис. 23 - Лайоут для планшетів.

1.6 Change the layout variant for tablets

You can use the Attributes pane in the **Design** tab to change attributes for this layout.

1. Turn off the Autoconnect tool in the toolbar. For this step, ensure that the tool is disabled: 
2. Clear all constraints in the layout by clicking the **Clear All Constraints.**
3. The layout editor offers resizing handles on all four corners of an element to resize it. In the **Component Tree**, select the TextView called show_count. To get the TextView out of the way so that you can freely drag the Button elements, drag a corner of it to resize it, as shown in the animated figure below.
4. Select the button_toast Button in the **Component Tree**, click the **Attributes** tab to open the **Attributes** pane, and change the textSize to **60sp** (#1 in the figure below) and the layout_width to **wrap_content** (#2 in the figure below).
5. Select the button_count Button in the **Component Tree**, change the textSize to **60sp** and the layout_width to **wrap_content**, and drag the Button above the TextView to an empty space in the layout.

Змінюємо розмір тексту та лайоут розміру для кнопок за вказівками, поданими вище.

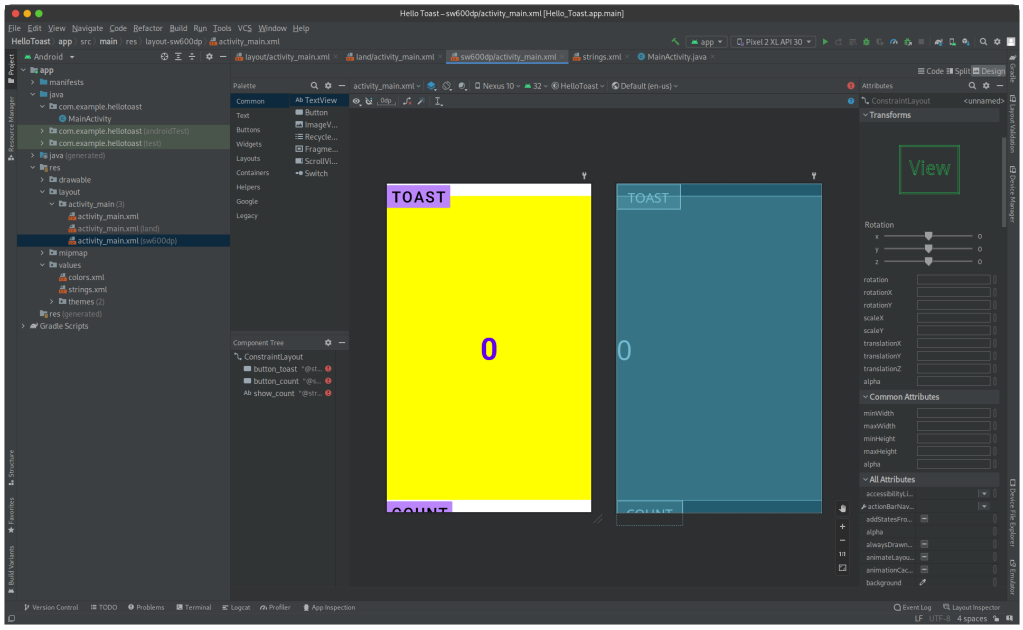


Рис. 24 - Редагування лайоуту для планшету.

1.7 Use a baseline constraint

1. Constrain the button_toast Button to the top and left side of the layout, drag the button_count Button to a space near the button_toast Button, and constrain the left side of the button_count Button to the right side of the button_toast Button.
2. Using a *baseline constraint*, you can constrain the button_count Button so that its text baseline matches the text baseline of the button_toast Button. Select the button_count element, and then hover your pointer over the element until the baseline constraint button  appears underneath the element.
3. Click the baseline constraint button. The baseline handle appears, blinking in green as shown in the animated figure. Click and drag a baseline constraint line to the baseline of the button_toast element.

Налаштуємо constraint для кнопок за вказівками.

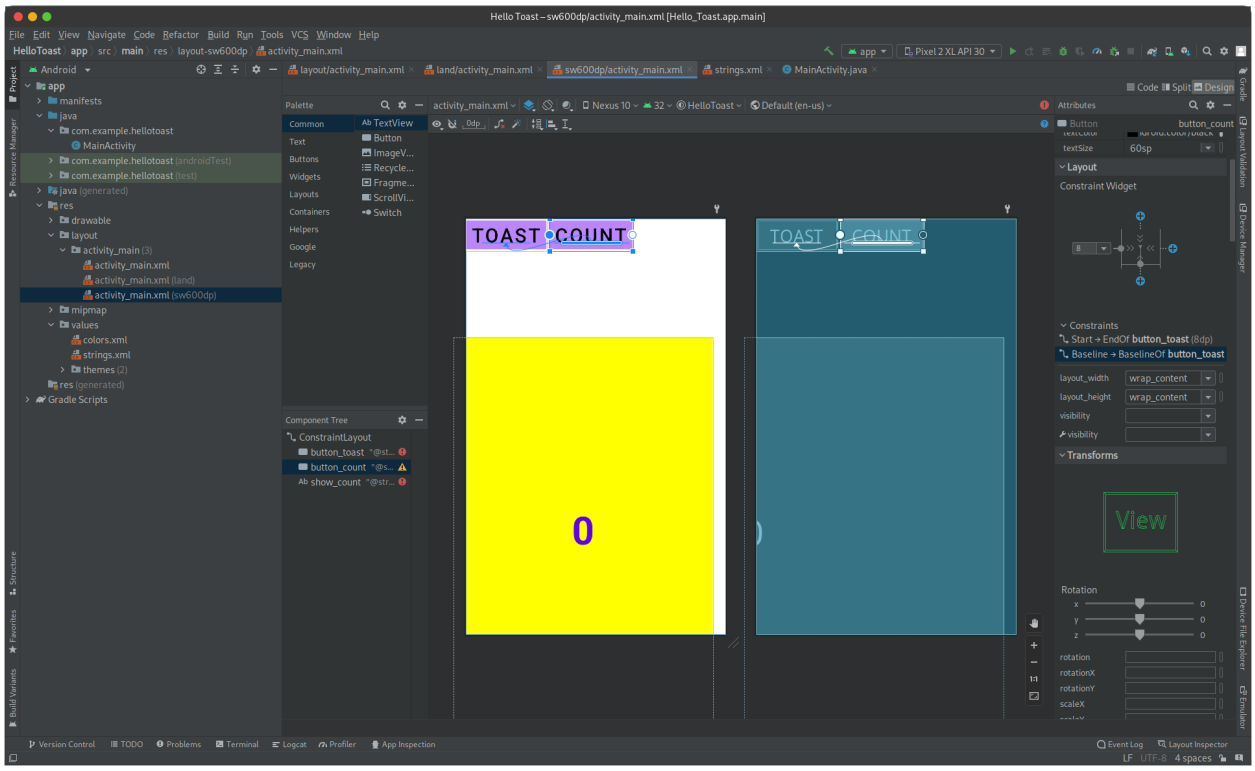


Рис. 25 - Вирівнювання кнопок за baseline.

1.8 Expand the buttons horizontally

1. Select the button_count Button in the **Component Tree**, and Shift-select the button_toast Button so that both are selected.
2. Click the pack button  in the toolbar, and choose **Expand Horizontally** as shown in the figure below.

The Button elements expand horizontally to fill the layout as shown below.

3. To finish the layout, constraint the show_count TextView to the bottom of the button_toast Button and to the sides and bottom of the layout, as shown in the animated figure below.
4. The final steps are to change the show_count TextView layout_width and layout_height to **Match Constraints** and the textSize to **200sp**. The final layout looks like the figure below.
5. Click the **Orientation in Editor** button  in the top toolbar and choose **Switch to Landscape**. The tablet layout appears in horizontal orientation as shown below. (You can choose **Switch to Portrait** to return to vertical orientation.).
6. Run the app on different emulators, and change the orientation after running the app, to see how it looks on different types of devices. You have successfully created an app that can run with a proper UI on phones and tablets that have different screen sizes and densities.

Завершуємо налаштування лайоуту для планшету за вказівками та перевіряємо в різних положеннях.

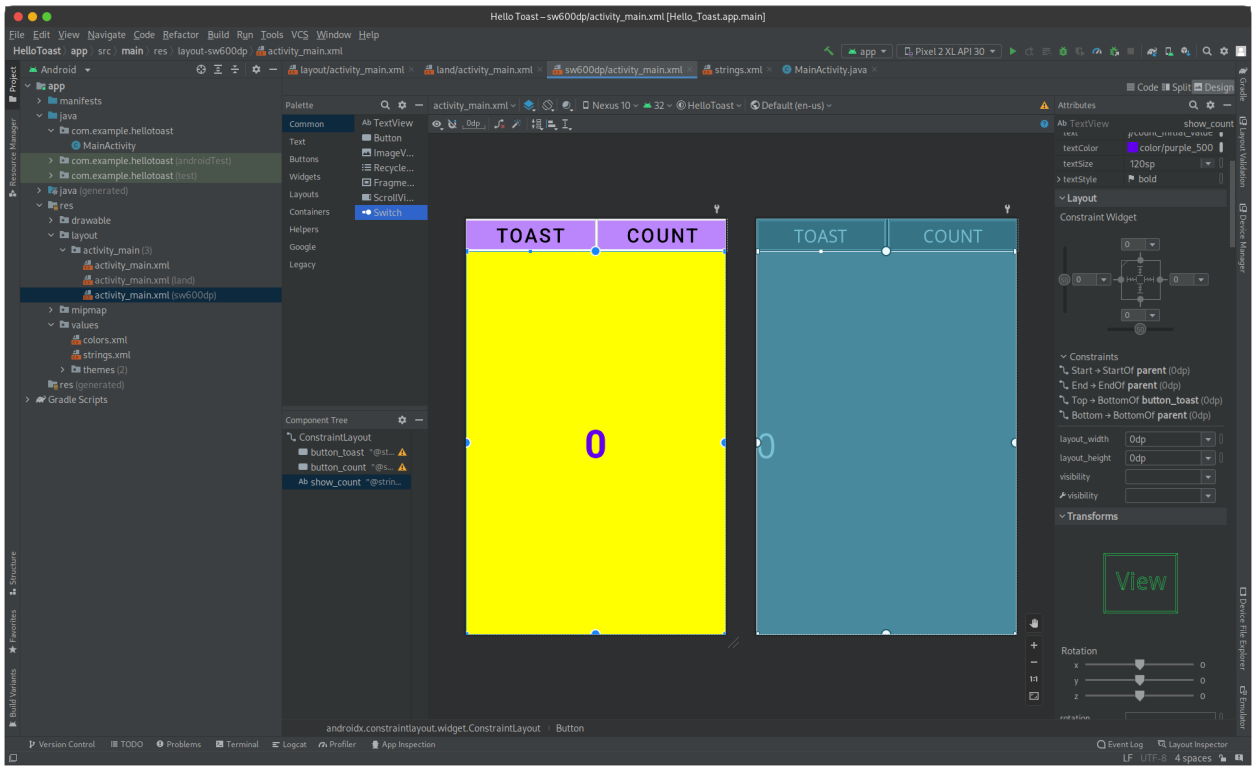


Рис. 26 - Завершений лайоут в звичайній орієнтації.

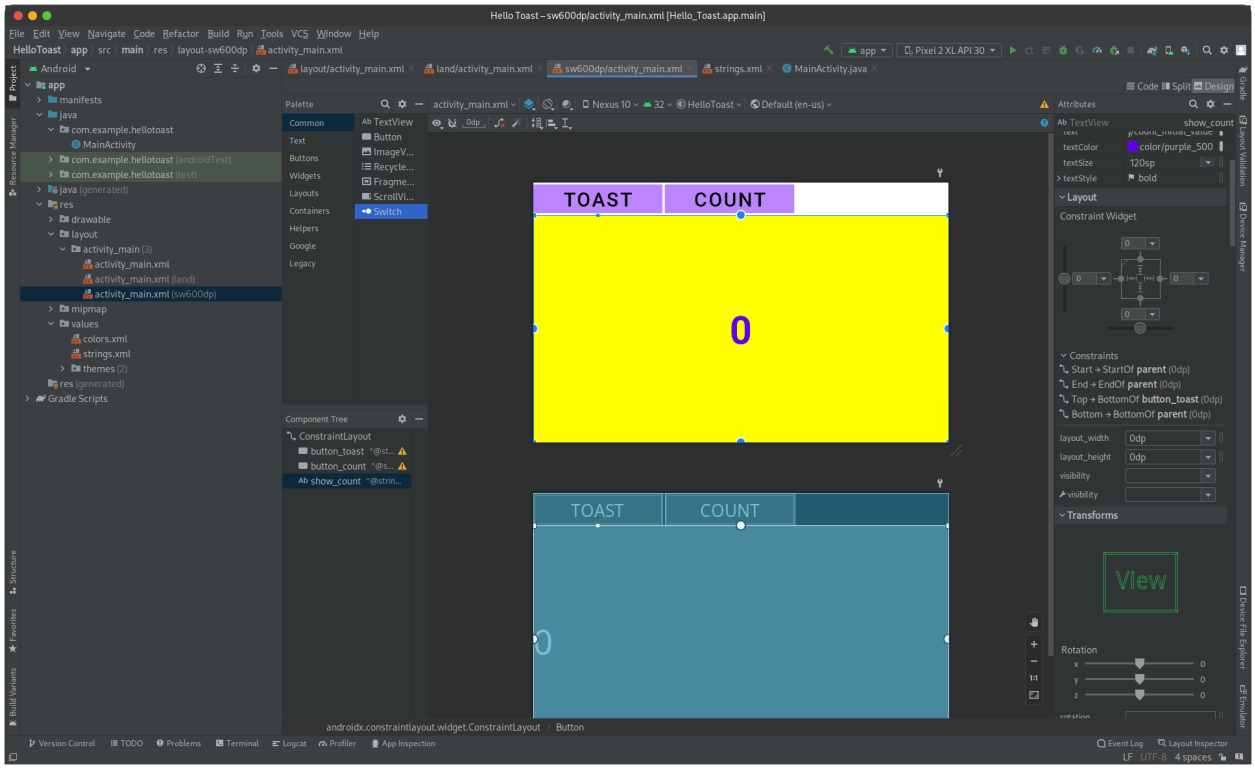


Рис. 27 - Завершений лайоут в ландшафтній орієнтації.

Task 2: Change the layout to LinearLayout

2.1 Change the root view group to LinearLayout

1. Open the Hello Toast app from the previous task.
2. Open the activity_main.xml layout file (if it is not already open), and click the **Text** tab at the bottom of the editing pane to see the XML code.
3. Change the <android.support.constraint.ConstraintLayout tag to <**LinearLayout**.
4. Make sure the closing tag at the end of the code has changed to </LinearLayout> (Android Studio automatically changes the closing tag if you change the opening tag). If it hasn't changed automatically, change it manually.
5. Under the <LinearLayout tag line, add the following attribute after the android:layout_height attribute

Змінюємо тип лайоуту в файлі activity_main.xml на LinearLayout.

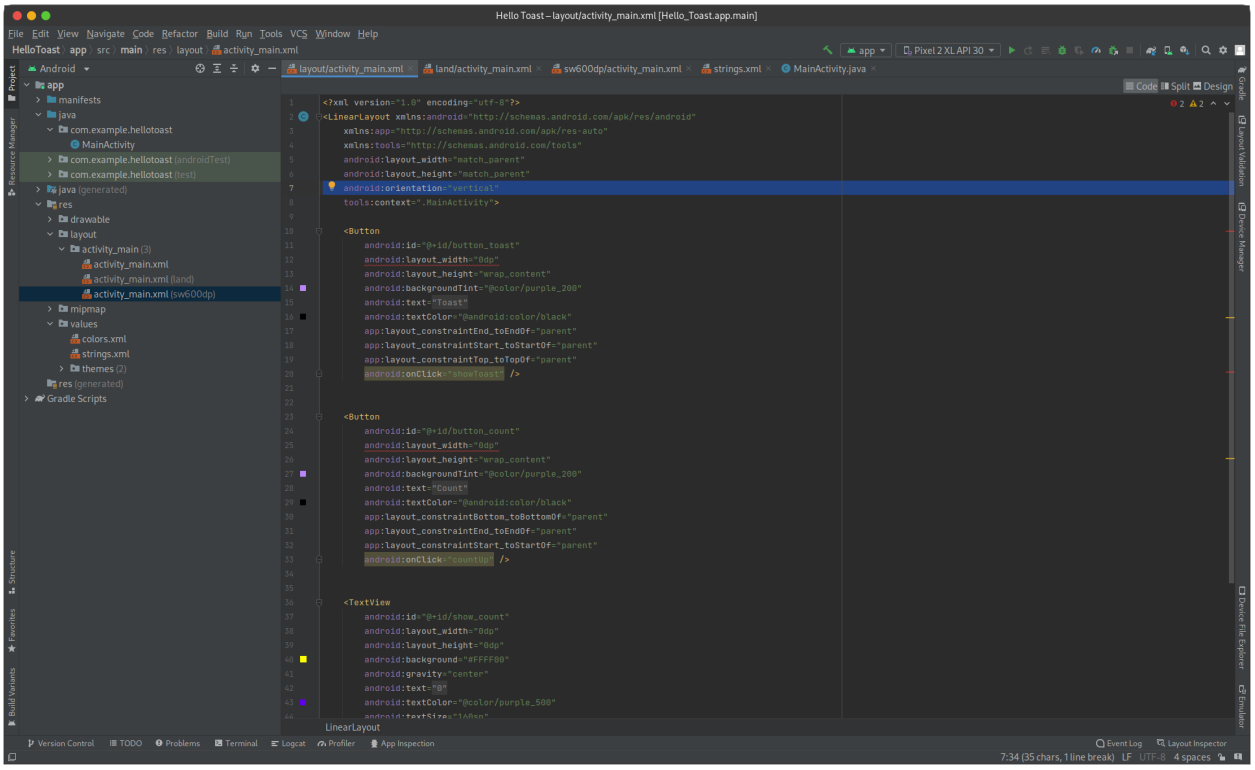


Рис. 28 - Відредагований activity_main.xml.

2.2 Change element attributes for the LinearLayout

1. Open the Hello Toast app from the previous task.
2. Open the activity_main.xml layout file (if it is not already open), and click the **Text** tab.
3. Find the button_toast Button element, and change the following attribute
4. Delete the following attributes from the button_toast element
5. Find the button_count Button element, and change the following attribute
6. Delete the following attributes from the button_count element
7. Find the show_count TextView element, and change the following attributes
8. Delete the following attributes from the show_count element
9. Click the **Preview** tab on the right side of the Android Studio window (if it is not already selected) to see a preview of the layout thus far

Змінюємо атрибути нашого лайоуту.

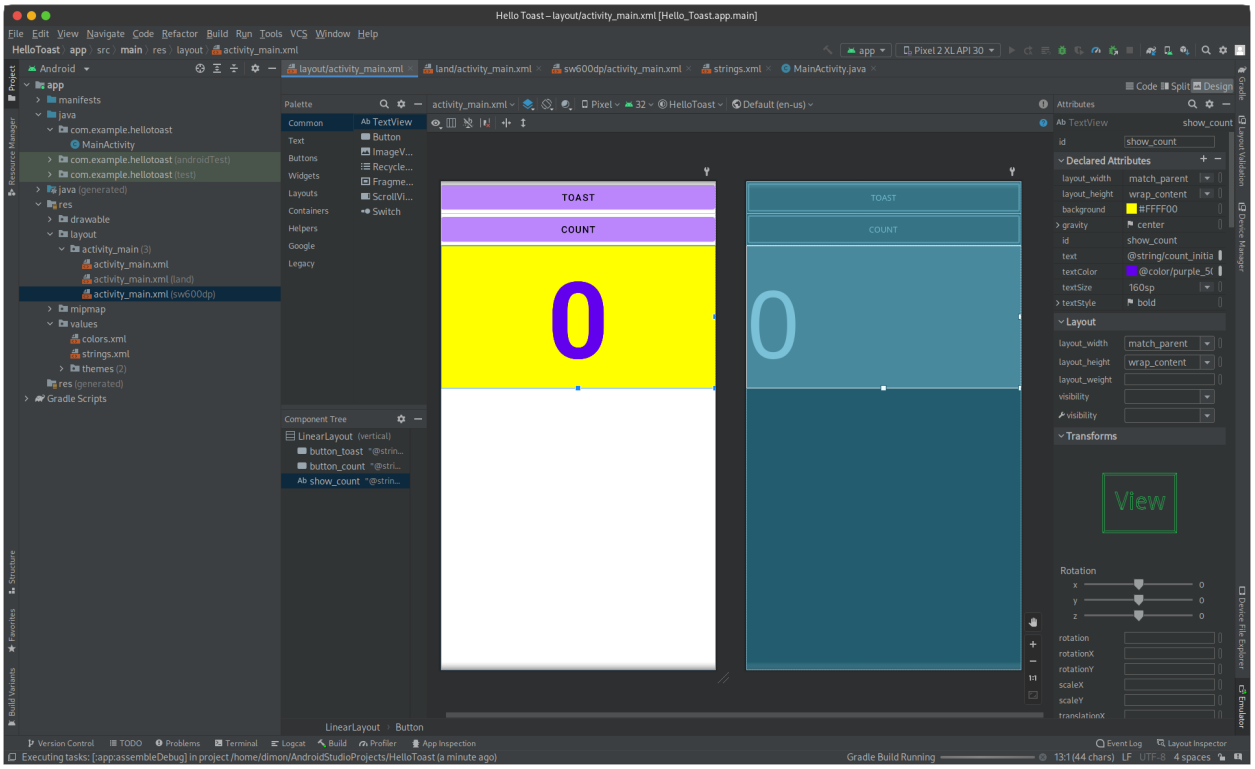


Рис. 29 - Оновлений лайоут.

2.3 Change the positions of elements in the LinearLayout

1. Open the Hello Toast app from the previous task.
2. Open the activity_main.xml layout file (if it is not already open), and click the **Text** tab.
3. Select the button_count Button and all of its attributes, from the <Button tag up to and including the closing /> tag, and choose **Edit > Cut**.
4. Click after the closing /> tag of the TextView element but before the closing </LinearLayout> tag, and choose **Edit > Paste**.
5. (Optional) To fix any indents or spacing issues for cosmetic purposes, choose **Code > Reformat Code** to reformat the XML code with proper spacing and indents.

Змінимо положення об’єктів в .xml файлі.

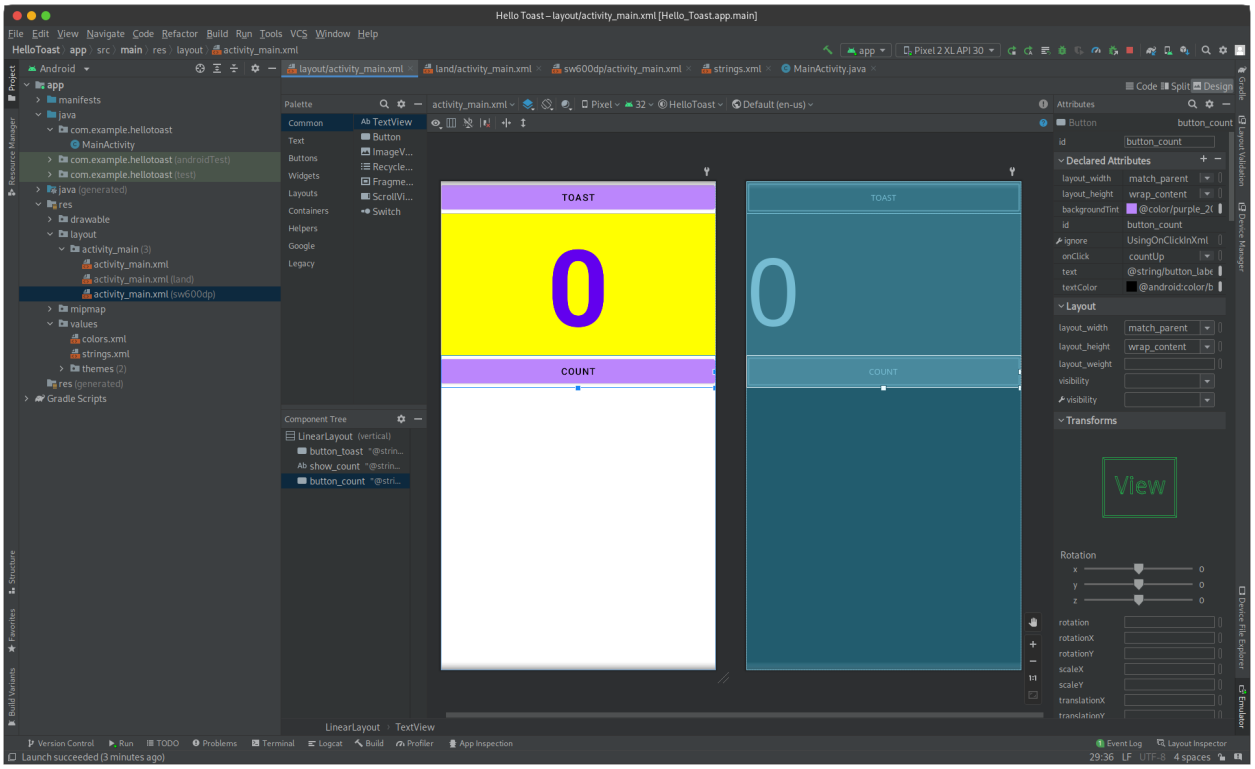


Рис. 30 - Об’єкти в правильному порядку.

2.4 Add weight to the TextView element

1. Open the Hello Toast app from the previous task.
2. Open the activity_main.xml layout file (if it is not already open), and click the **Text** tab.
3. Find the show_count TextView element, and add the following attribute:

android:layout_weight="1"

Додамо цей параметр, щоб вказати, скільки свободного місця займав елемент TextView. 1 означає 100% вільного місця.

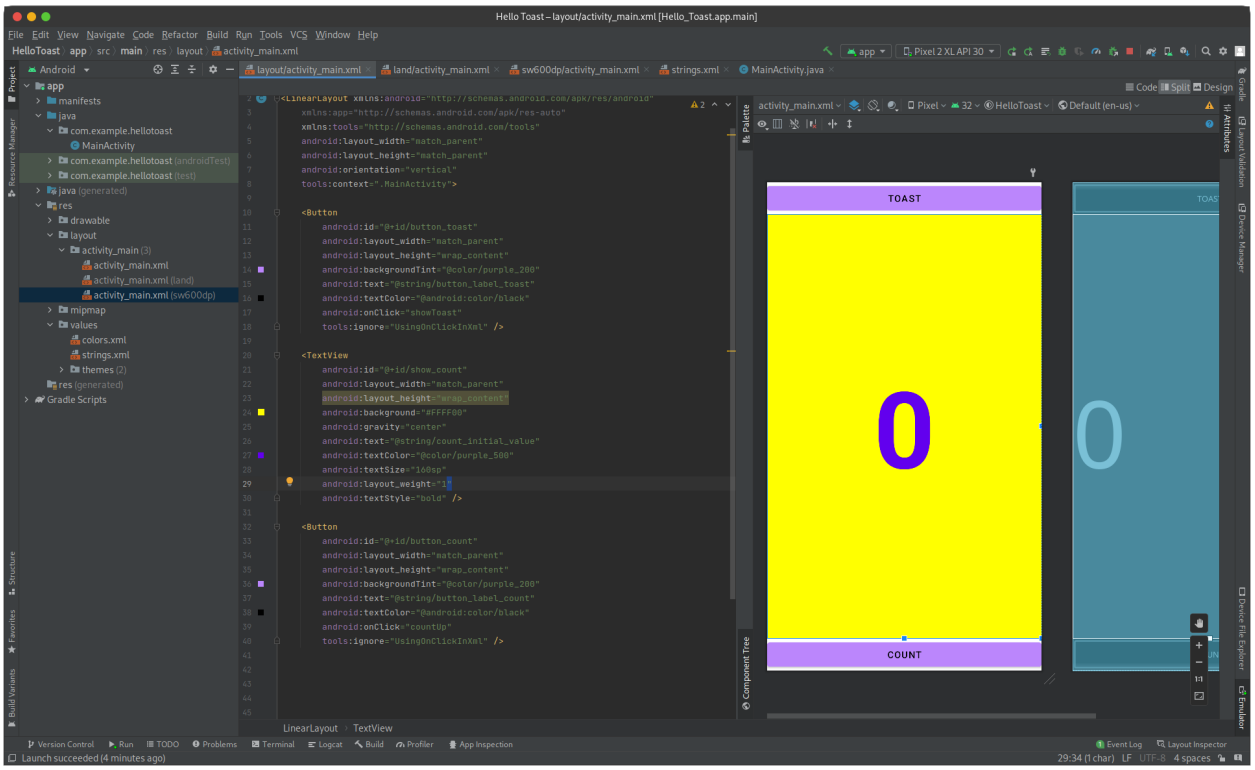


Рис. 31 - Відредагований лайоут.

Task 3: Change the layout to RelativeLayout

3.1 Change LinearLayout to RelativeLayout

An easy way to change the LinearLayout to a RelativeLayout is to add XML attributes in the **Text** tab.

1. Open the **activity_main.xml** layout file, and click the **Text** tab at the bottom of the editing pane to see the XML code.
2. Change the <LinearLayout at the top to <**RelativeLayout** so that the statement looks like this:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

3. Scroll down to make sure that the ending tag </LinearLayout> has also changed to </RelativeLayout>; if it hasn't, change it manually.

Змінимо наш лайоут на Relative.

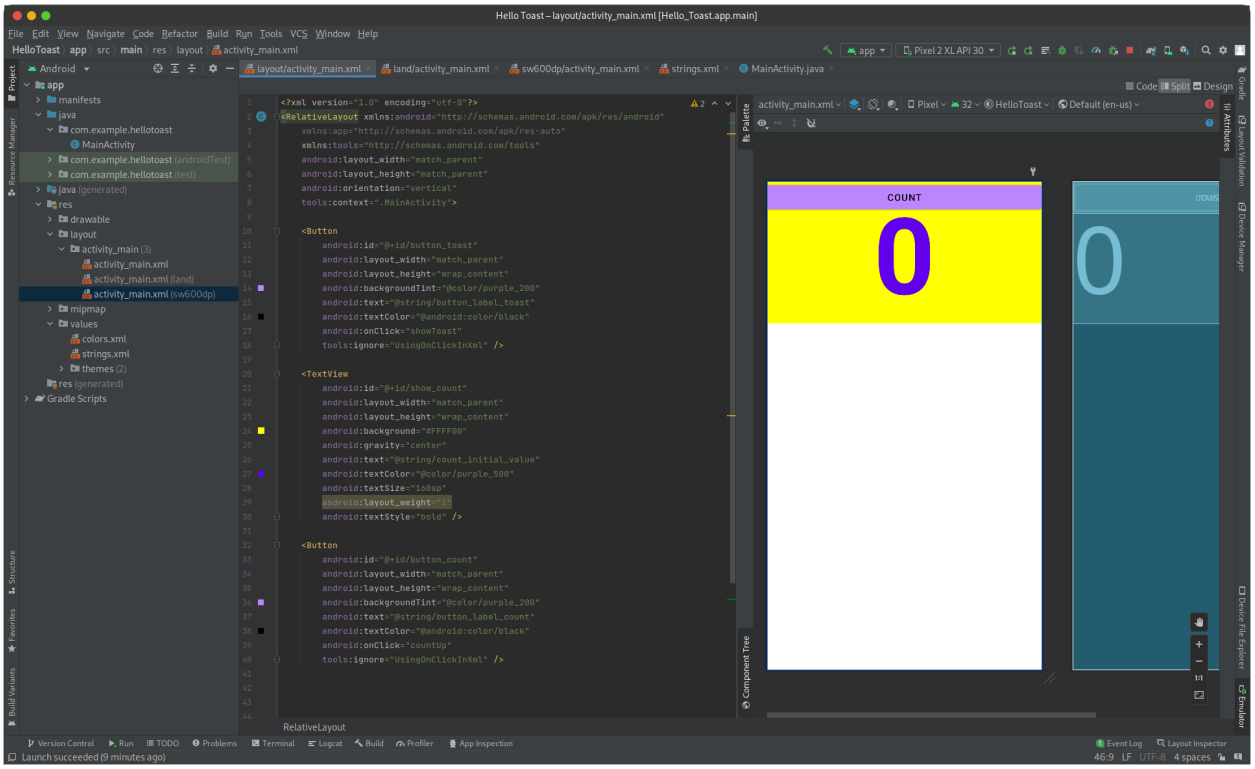


Рис. 32 - Змінений лайоут.

3.2 Rearrange views in a RelativeLayout

An easy way to rearrange and position views in a RelativeLayout is to add XML attributes in the **Text** tab.

1. Click the **Preview** tab at the side of the editor (if it is not already selected) to see the layout preview, which now looks like the figure below.
2. Add the android:layout_below attribute to the button_count Button to position the Button directly below the show_count TextView. This attribute is one of several attributes for positioning views within a RelativeLayout—you place views in relation to other views.
3. Add the android:layout_centerHorizontal attribute to the same Button to center the view horizontally within its parent, which in this case is the RelativeLayout view group.
4. Add the following attributes to the show_count TextView
5. Delete the android:layout_weight="1" attribute from the show_count TextView, which is not relevant for a RelativeLayout.

Переробимо наш лайоут на Relative.

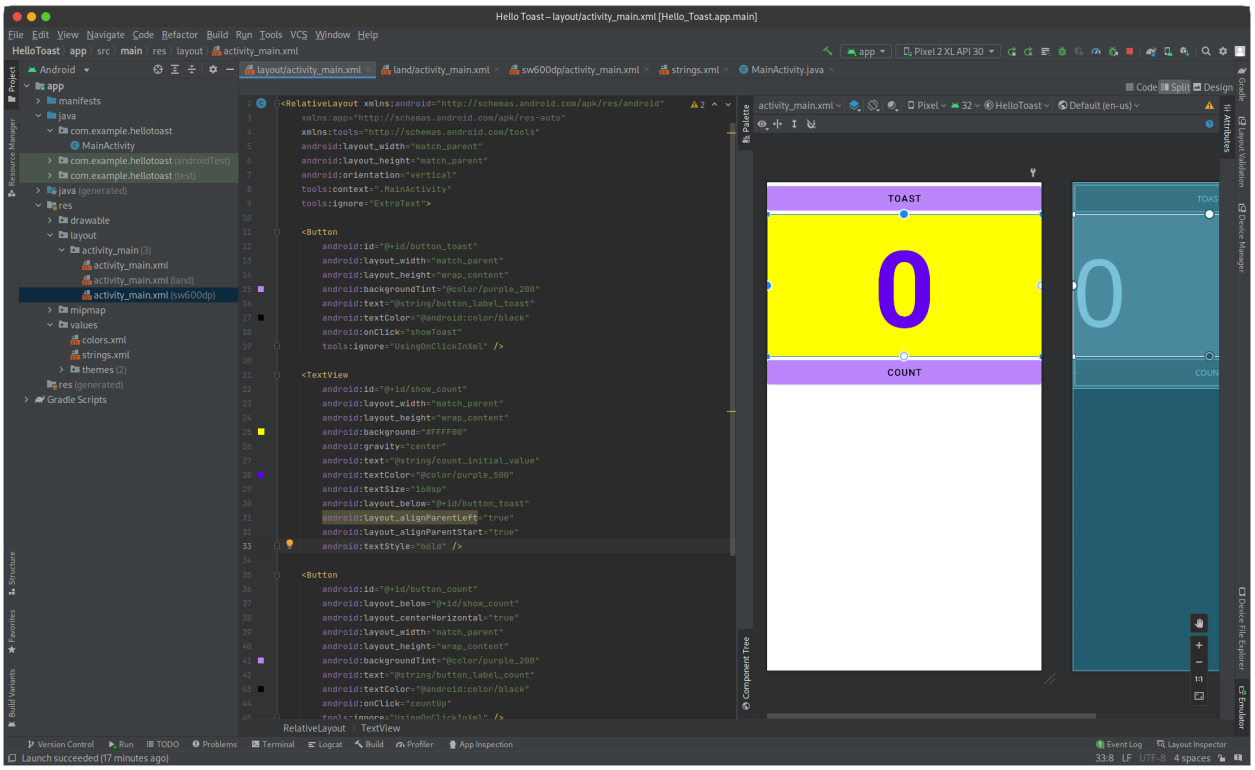


Рис. 33 - Relative layout.

«Homework»

Change an app

Open the HelloToast app.

1. Change the name of the project to **HelloConstraint**, and refactor the project to Hello Constraint. (For instructions on how to copy and refactor a project, see Appendix: Utilities.)
2. Modify the activity_main.xml layout to align the **Toast** and **Count** Button elements along the left side of the show_count TextView that shows "0". Refer to the figures below for the layout.
3. Include a third Button called **Zero** that appears between the **Toast** and **Count** Button elements.
4. Distribute the Button elements vertically between the top and bottom of the show_count TextView.
5. Set the **Zero** Button to initially have a gray background.
6. Make sure that you include the **Zero** Button for the landscape orientation in activity_main.xml (land), and also for a tablet-sized screen in activity_main (xlarge).
7. Make the **Zero** Button change the value in the show_count TextView to 0.
8. Update the click handler for the **Count** Button so that it changes its *own* background color, depending on whether the new count is odd or even.
9. Update the click handler for the **Count** Button to set the background color for the **Zero** Button to something other than gray to show it is now active. Hint: You can use findViewById in this case.
10. Update the click handler for the **Zero** Button to reset the color to gray, so that it is gray when the count is zero.

В домашній роботі нам пропонується додати третю кнопку “zero”, яка буде обнуляти наше значення. Виконуємо за вказівками.

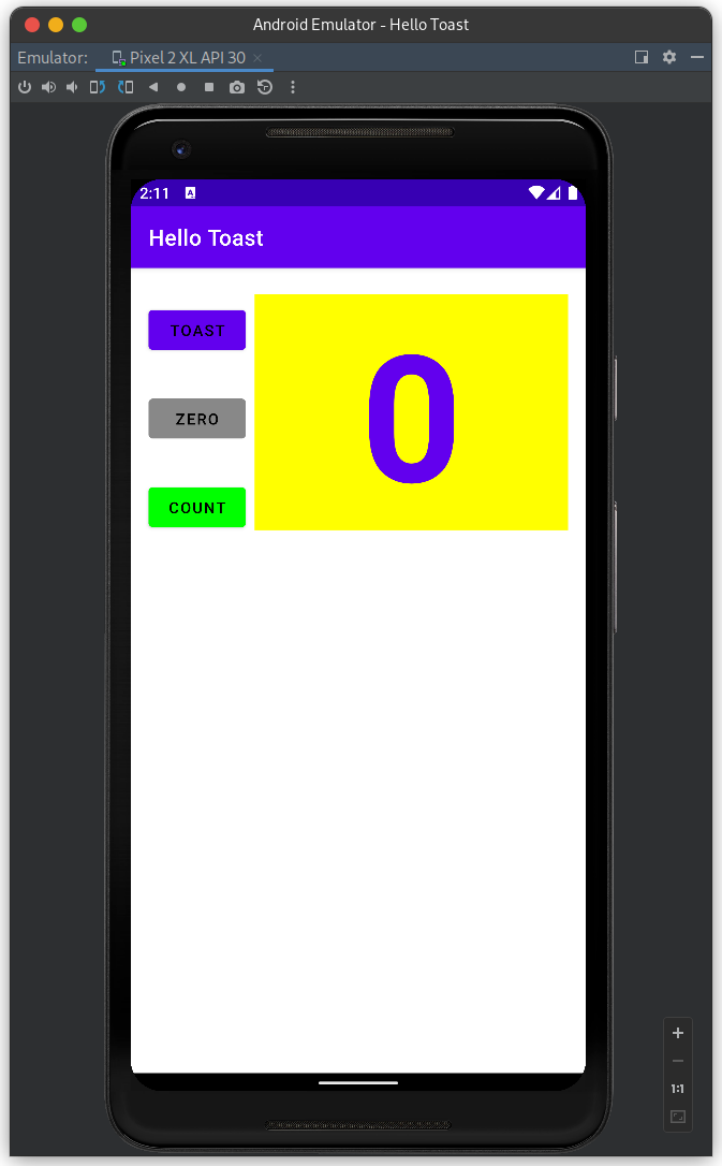


Рис. 34 - Допрацьований інтерфейс програми.

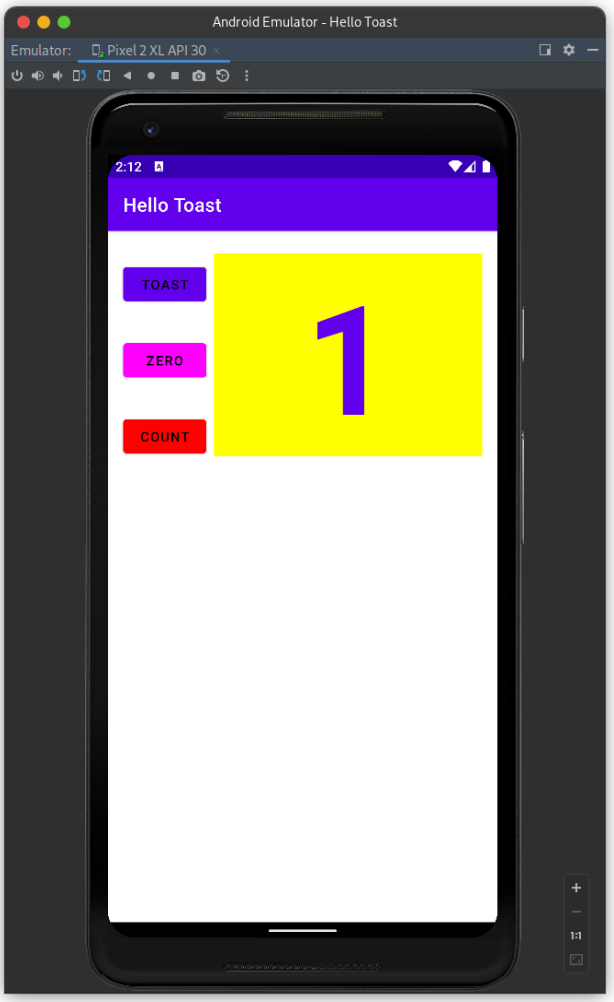


Рис. 35 - Зміна кнопок при натисканні.

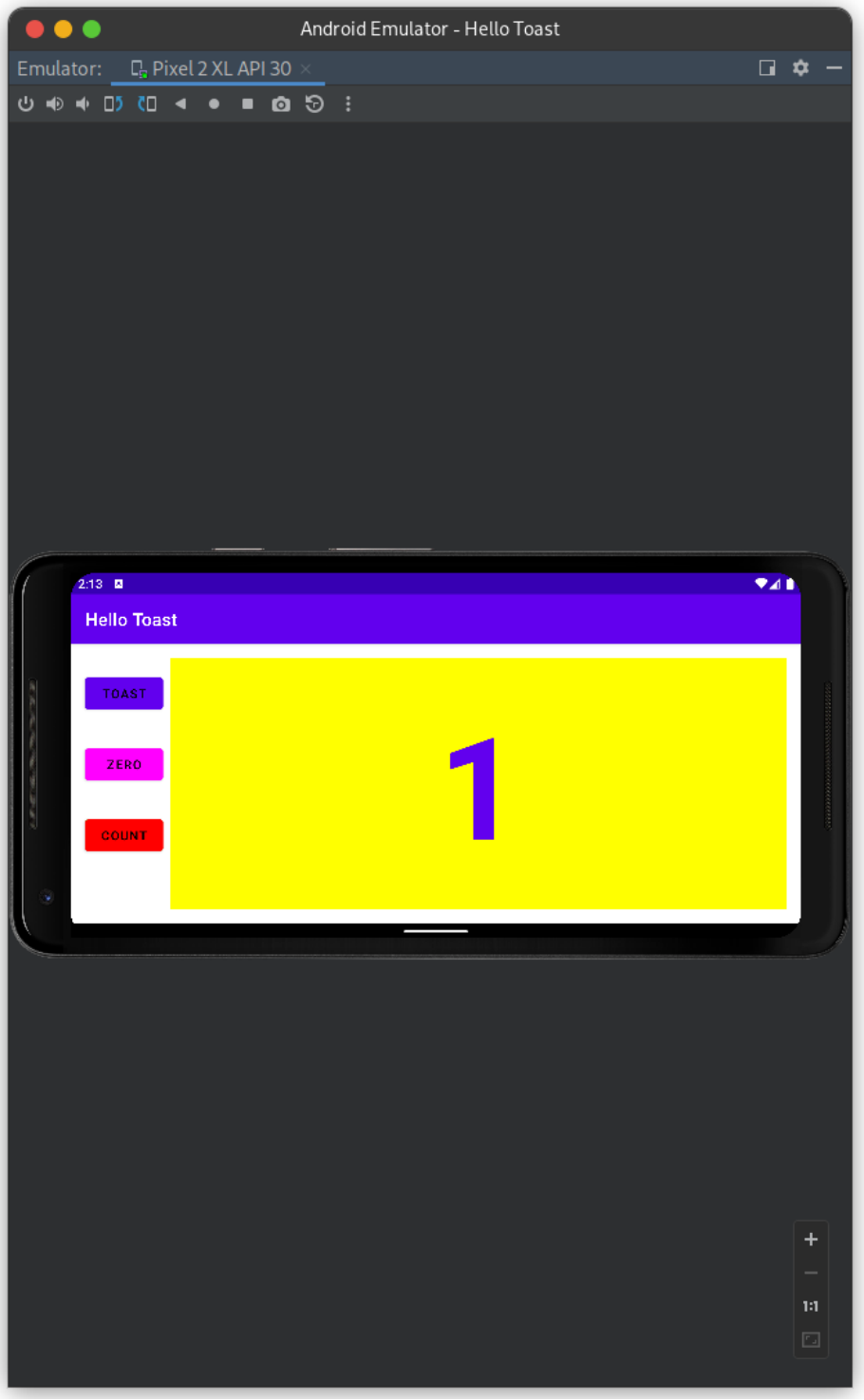


Рис. 36 - Горизонтальний лайоут.

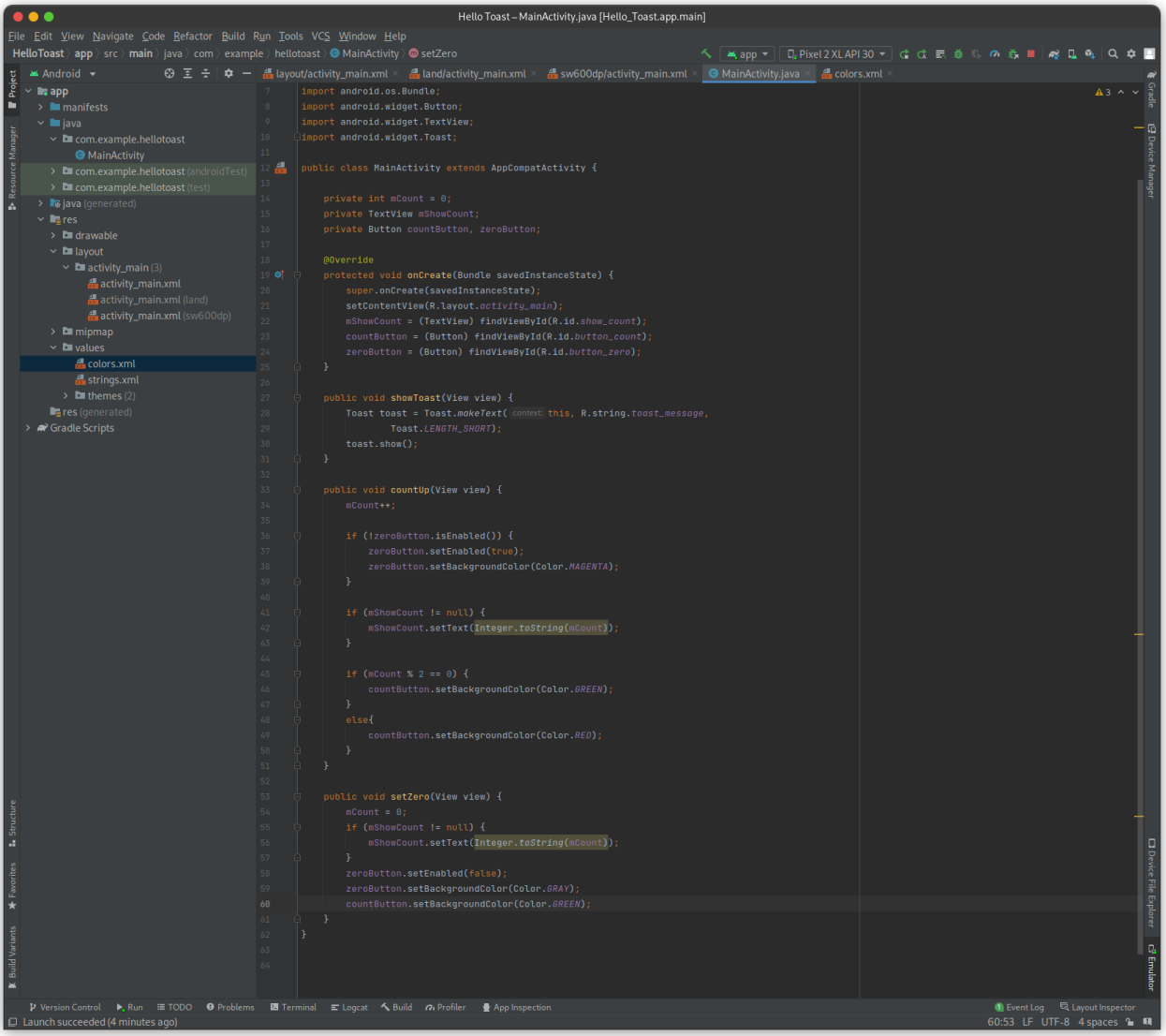


Рис. 37 - Доопрацьована реалізація функцій хендлерів.

Завдання No3. Text and scrolling views

Завдання для закріплення теоретичного матеріалу (Task 1–3). Виконайте завдання з розділу «Homework». За результатами виконання оформіть звіт з лабораторної роботи.

Task 1: Add and edit TextView elements

1.1 Create the project and TextView elements
Створюємо пустий проект з заданими параметрами.

Attribute	Value
Application Name	Scrolling Text
Company Name	android.example.com (or your own domain)
Phone and Tablet Minimum SDK	API15: Android 4.0.3 IceCreamSandwich
Template	Empty Activity
Generate Layout File checkbox	Selected
Backwards Compatibility (AppCompat) checkbox	Selected

Рис. 38 - Параметри проекту.

Далі створюємо лайоут за аналогією, як ми це робили в попередніх завданнях, за вказівками з лабораторної роботи.

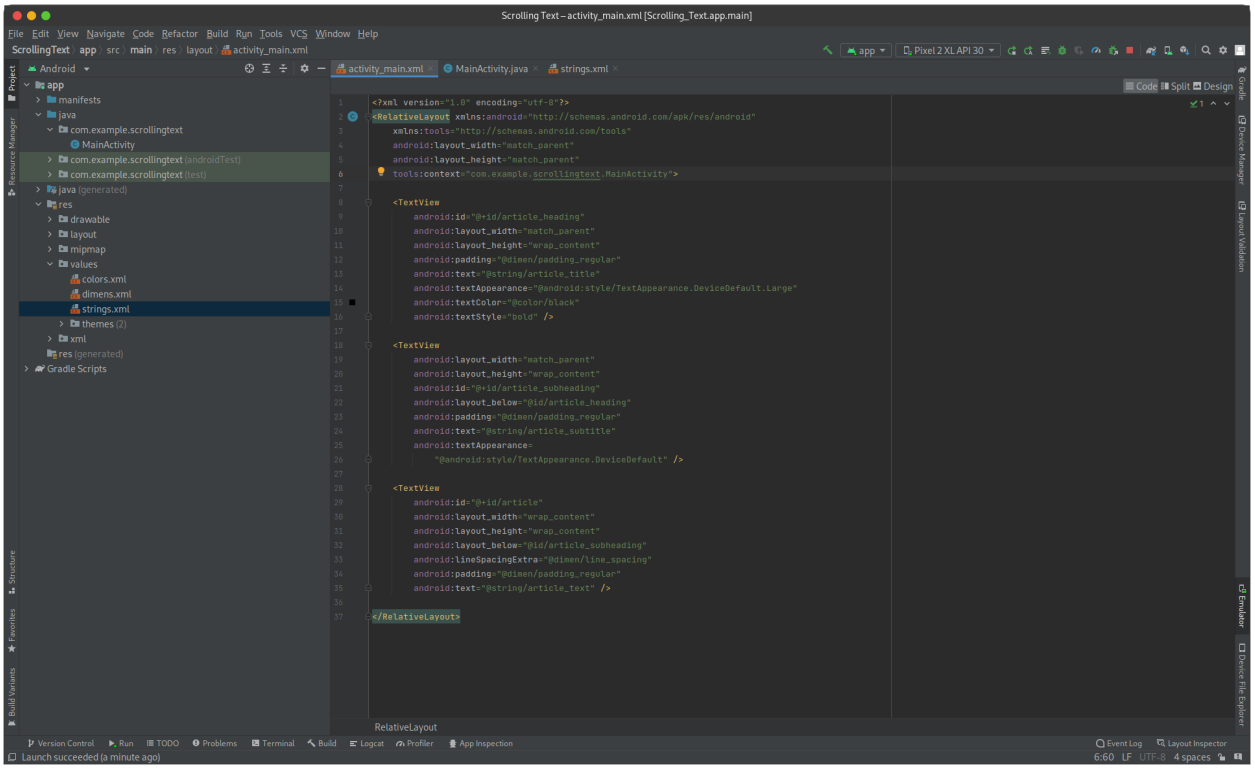


Рис. 39 - Новий лайоут.

1.2 Add the text of the article.

Додаємо поля до strings.xml, які будуть містити ресурси для TextView.

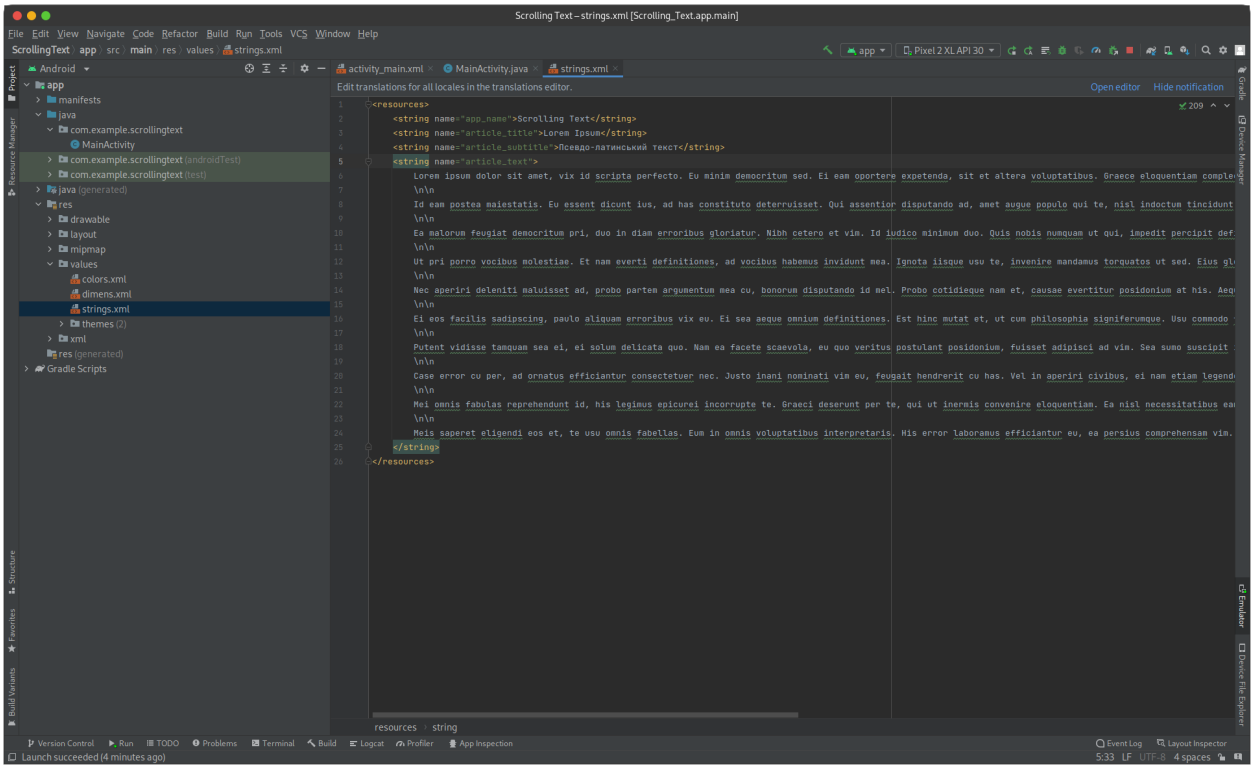


Рис. 40 - Файл strings.xml.

Рис. 41 - Запущена програма.

2.1 Add the autoLink attribute for active web links

Add the android:autoLink="web" attribute to the article TextView.

```
<TextView
    android:id="@+id/article"
    android:autoLink="web"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />
```

Рис. 42- Доданий атрибут autoLink.

2.2 Add a ScrollView to the layout.

To make a View (such as a TextView) scrollable, embed the View *inside* a ScrollView.

1. Add a ScrollView between the article_subheading TextView and the article TextView. As you enter **<ScrollView**, Android Studio automatically adds </ScrollView> at the end, and presents the android:layout_width and android:layout_height attributes with suggestions.
2. Choose **wrap_content** from the suggestions for both attributes.

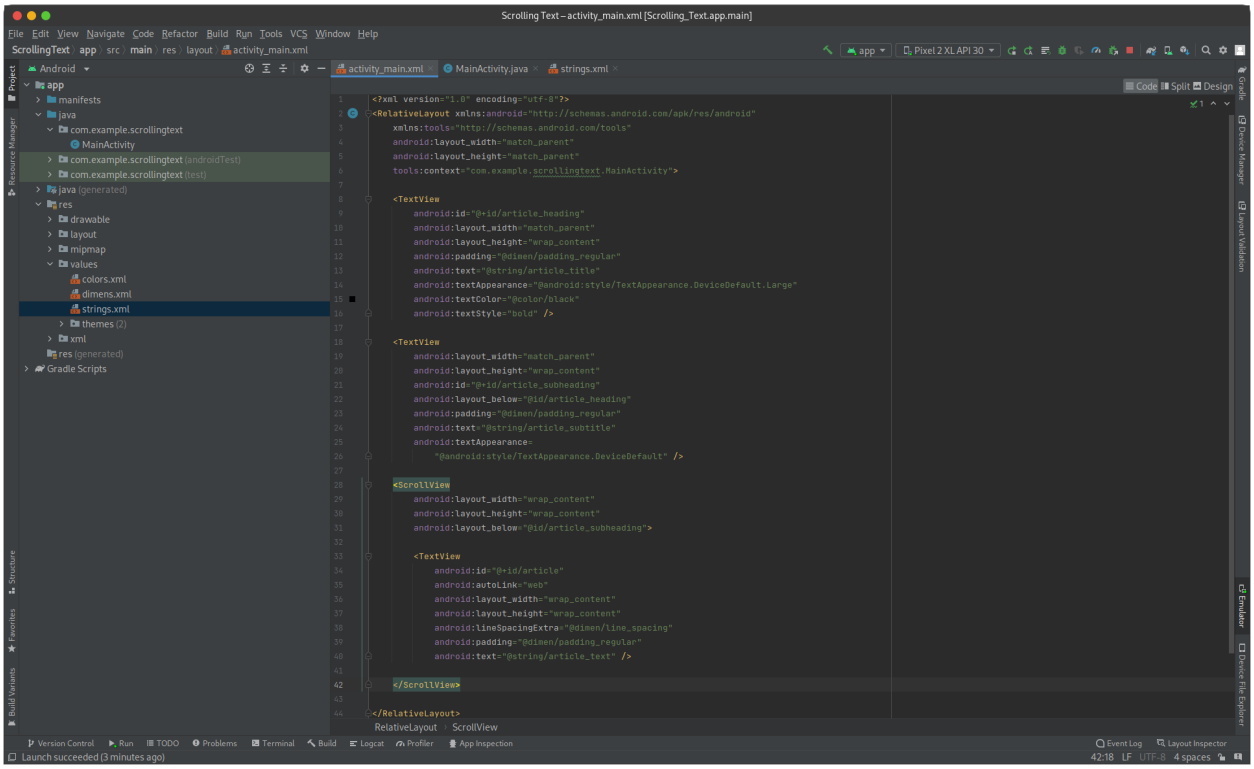


Рис. 43 - Готовий лайоут.

2.3 Run the app

Протестуємо програму в емуляторі.

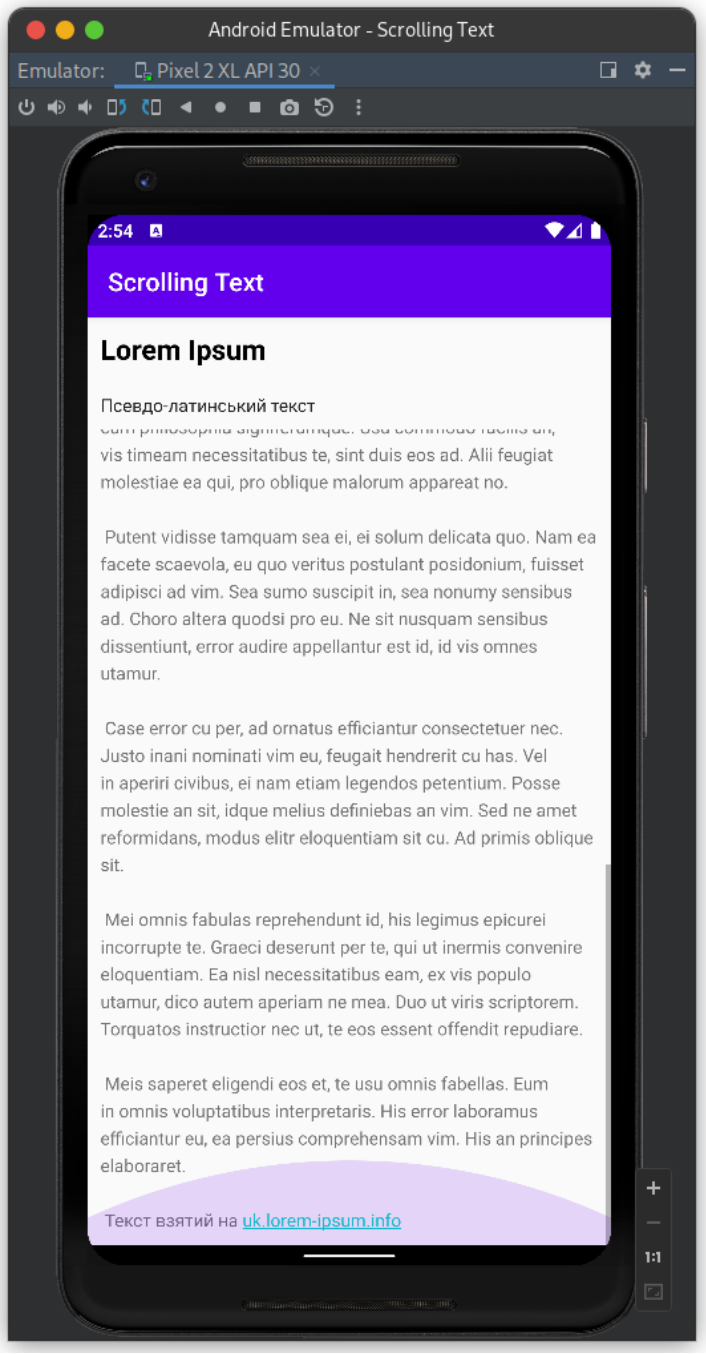


Рис. 44 - Запущена программа. Працює скролінг та посилання.

Task 3: Scroll multiple elements

As noted before, a ScrollView can contain only one child View (such as the article TextView you created). However, that View can be another ViewGroup that contains View elements, such as LinearLayout. You can *nest* a ViewGroup such as LinearLayout *within* the ScrollView, thereby scrolling everything that is inside the LinearLayout.

For example, if you want the subheading of the article to scroll along with the article, add a LinearLayout within the ScrollView, and move the subheading and article into the LinearLayout. The LinearLayout becomes the single child View in the ScrollView as shown in the figure below, and the user can scroll the entire LinearLayout: the subheading and the article.

Створимо LinearLayout всередині ScrollView за вказівками, щоб зробити скролінг декількох елементів.

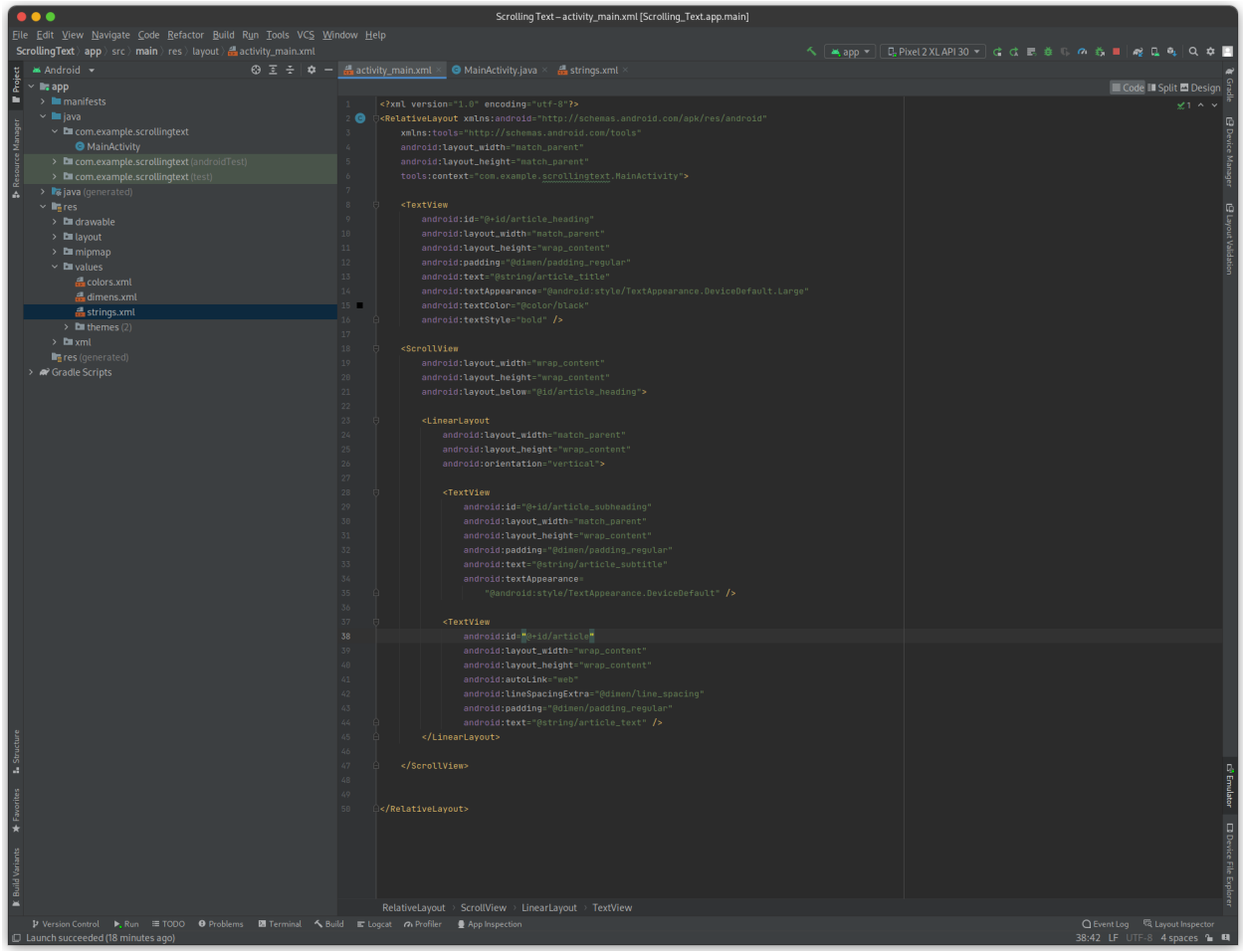


Рис. 45 - Оновлений лайоут.

Запускаємо та бачимо, що subtitle також тепер скролиться.

Рис. 46 - Запуск програми.

ВИСНОВОК

В процесі виконання лабораторної роботи вивчили основні три типи лайоутів(Constraint, Linear та Relative). Спробували створити кожен з них, робили адаптацію під різну орієнтацію екрану та розмір. Також досліджували різноманітні атрибути елементів, за допомогою java коду взаємодіяли з ними. Ознайомилися з елементом Scrollview та його атрибутами.