

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



GIÁO TRÌNH
THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1. Làm quen	3
Bài 1) Tạo ứng dụng đầu tiên	3
1.1) Android Studio và Hello World.....	3
1.2) Giao diện người dùng tương tác đầu tiên	34
1.3) Trình chỉnh sửa bố cục	69
1.4) Văn bản và các chế độ cuộn	109
1.5) Tài nguyên có sẵn	130
Bài 2) Activities	139
2.1) Activity và Intent	139
2.2) Vòng đời của Activity và trạng thái	139
2.3) Intent ngầm định.....	139
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ.....	139
3.1) Trình gỡ lỗi.....	139
3.2) Kiểm thử đơn vị.....	139
3.3) Thư viện hỗ trợ.....	139
CHƯƠNG 2. Trải nghiệm người dùng.....	140
Bài 1) Tương tác người dùng.....	140
1.1) Hình ảnh có thể chọn	140
1.2) Các điều khiển nhập liệu	140
1.3) Menu và bộ chọn	140
1.4) Điều hướng người dùng.....	140
1.5) RecyclerView.....	140
Bài 2) Trải nghiệm người dùng thú vị	140
2.1) Hình vẽ, định kiểu và chủ đề	140
2.2) Thẻ và màu sắc.....	140

2.3) Bố cục thích ứng.....	140
Bài 3) Kiểm thử giao diện người dùng	140
3.1) Espresso cho việc kiểm tra UI.....	140
CHƯƠNG 3. Làm việc trong nền	140
Bài 1) Các tác vụ nền	140
1.1) AsyncTask	140
1.2) AsyncTask và AsyncTaskLoader	140
1.3) Broadcast receivers	140
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	140
2.1) Thông báo.....	140
2.2) Trình quản lý cảnh báo	140
2.3) JobScheduler	140
CHƯƠNG 4. Lưu trữ dữ liệu người dùng	141
Bài 1) Tùy chọn và cài đặt.....	141
1.1) Shared preferences	141
1.2) Cài đặt ứng dụng	141
Bài 2) Lưu trữ dữ liệu với Room	141
2.1) Room, LiveData và ViewModel	141
2.2) Room, LiveData và ViewModel	141

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

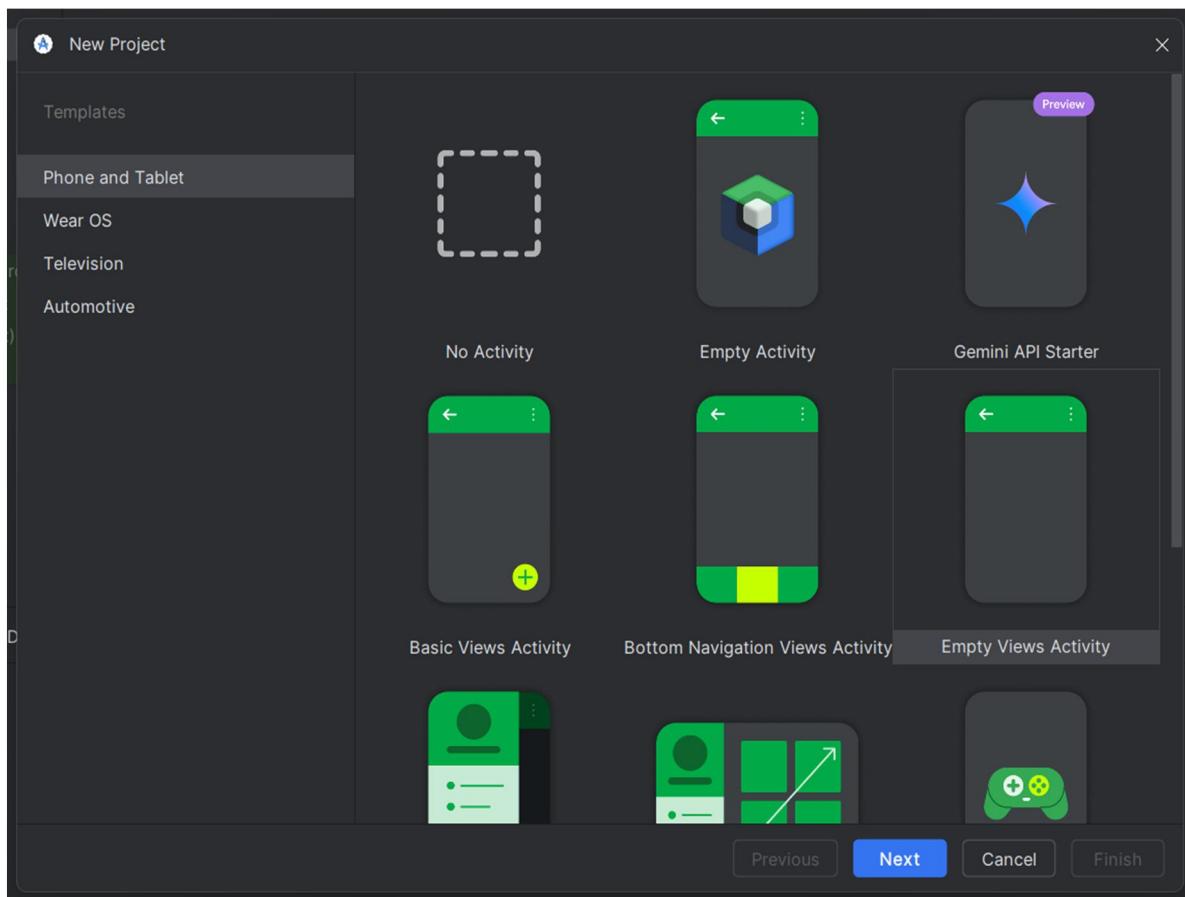
Giới thiệu

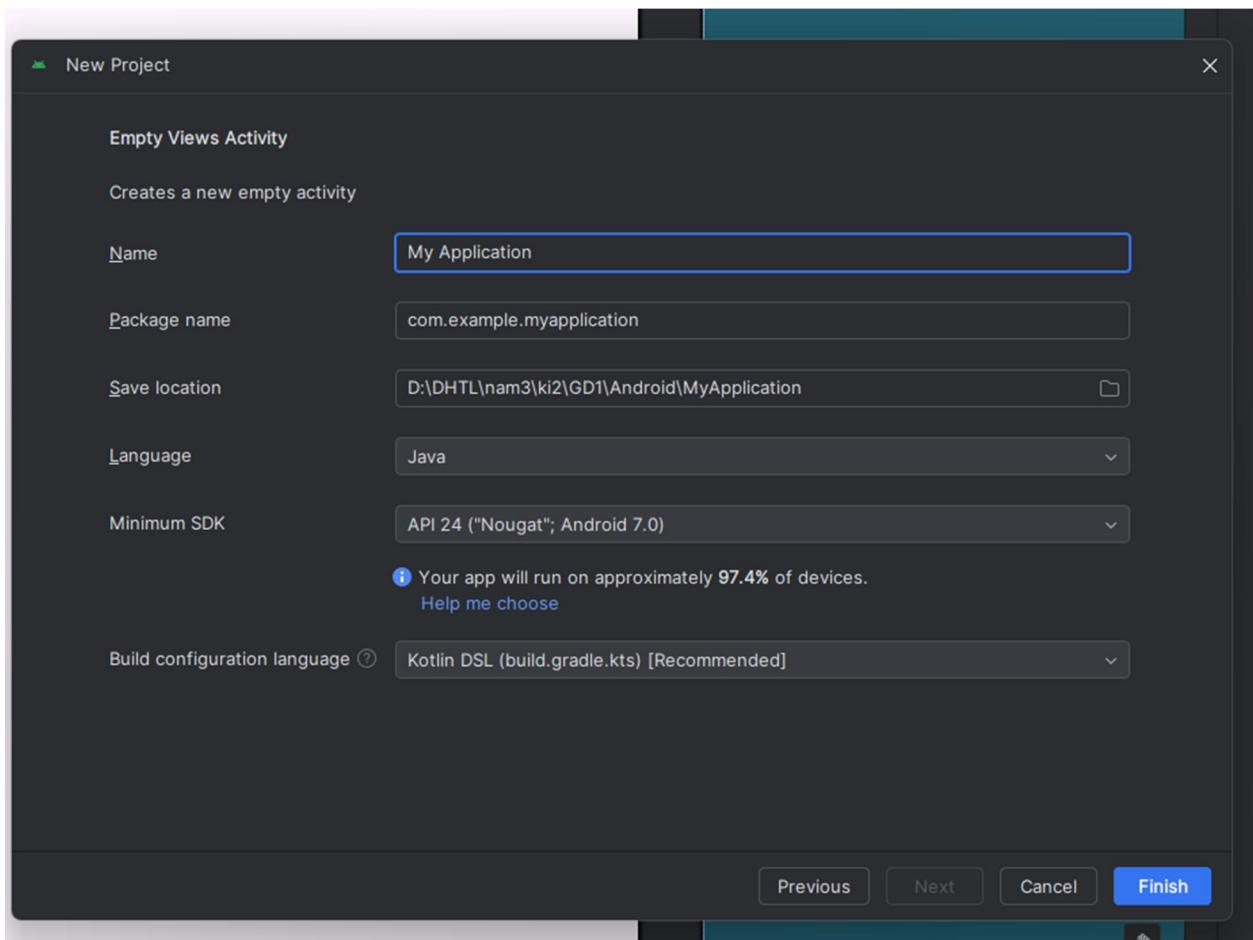
Trong bài thực hành này, bạn sẽ tìm hiểu cách cài đặt Android Studio, môi trường phát triển Android. Bạn cũng sẽ tạo và chạy ứng dụng Android đầu tiên của mình, Hello World, trên một trình giả lập và trên một thiết bị vật lý.

Những gì bạn nên biết

Bạn nên có khả năng:

- Hiểu quy trình phát triển phần mềm tổng quát cho các ứng dụng lập trình hướng đối tượng sử dụng một IDE (môi trường phát triển tích hợp) như Android Studio.
- Chứng minh rằng bạn có ít nhất 1-3 năm kinh nghiệm trong lập trình hướng đối tượng, với một phần trong số đó tập trung vào ngôn ngữ lập trình Java. (Các bài thực hành này sẽ không giải thích về lập trình hướng đối tượng hoặc ngôn ngữ Java).





Những gì bạn sẽ cần

- Một máy tính chạy Windows hoặc Linux, hoặc một Mac chạy macOS. Xem trang tải xuống Android Studio để biết yêu cầu hệ thống cập nhật.
- Truy cập Internet hoặc một phương pháp thay thế để tải các cài đặt mới nhất của Android Studio và Java lên máy tính của bạn.

Những thứ bạn sẽ tìm hiểu

- Cách cài đặt và sử dụng IDE Android Studio.
- Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android.
- Cách tạo một dự án Android từ một mẫu.
- Cách thêm thông điệp ghi lại vào ứng dụng của bạn để phục vụ mục đích gỡ lỗi.

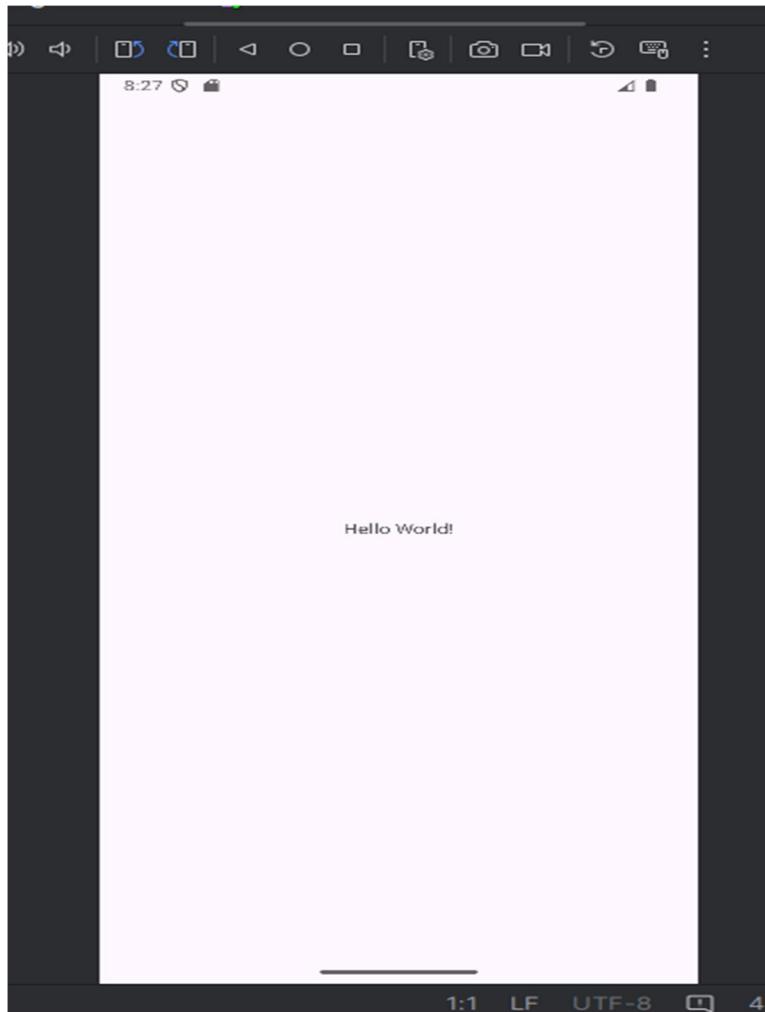
Những thứ bạn sẽ làm

- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.
- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.
- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

Tổng quan về ứng dụng

Sau khi hoàn tất cài đặt **Android Studio**, bạn sẽ tạo từ mẫu, 1 dự án mới cho ứng dụng **Hello World**. Đây là 1 dự án ứng dụng đơn giản hiển thị chuỗi “Hello World” trên màn hình của máy ảo thiết bị Android hoặc thiết bị vật lý.

Dưới đây là giao diện của ứng dụng sau khi đã hoàn thành:



Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp hoàn chỉnh (IDE), bao gồm một trình soạn thảo mã tiên tiến và bộ mẫu ứng dụng đa dạng. Ngoài ra, nó còn chứa các công cụ hỗ trợ phát triển, gỡ lỗi, kiểm thử và tối ưu hiệu suất, giúp quá trình phát triển ứng dụng trở nên nhanh chóng và dễ dàng hơn. Bạn có thể kiểm thử ứng dụng của bạn trên nhiều trình giả lập được cấu hình sẵn hoặc trực tiếp trên thiết bị di động của mình, xây dựng ứng dụng hoàn chỉnh và phát hành chúng trên cửa hàng Google Play.

Chú ý: Android Studio luôn được cải tiến liên tục. Để cập nhật thông tin chi tiết về yêu cầu cấu hình hệ thống cũng như hướng dẫn cài đặt, vui lòng tham khảo trang chủ của [Android Studio](#).

Android Studio có thể dùng cho các máy tính chạy hệ điều hành Windows hoặc Linux, và cho các máy Macs chạy macOS. Phiên bản OpenJDK (Java Development Kit) mới nhất được tích hợp sẵn trong Android Studio.

Để bắt đầu sử dụng Android Studio, trước tiên hãy kiểm tra các [yêu cầu hệ thống](#) để đảm bảo rằng máy tính của bạn đáp ứng đủ các tiêu chí để sử dụng. Quá trình cài đặt tương tự cho tất cả các nền tảng. Các khía cạnh khác nếu có sẽ được ghi chú bên dưới.

1. Truy cập [trang Android Developers](#) và làm theo hướng dẫn để tải về và [cài đặt Android Studio](#).
2. Chấp nhận các cấu hình mặc định cho tất cả các bước cài đặt, và đảm bảo rằng tất cả các thành phần cần thiết đều được chọn.
3. Sau khi cài đặt xong, Setup Wizard sẽ tự động tải về và cài đặt một số thành phần bổ sung, bao gồm Android SDK. Hãy kiên nhẫn, quá trình này có thể mất chút thời gian tùy thuộc vào tốc độ Internet của bạn, và một số bước có thể trông có vẻ thưa thớt.
4. Khi quá trình tải về hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng để tạo dự án đầu tiên của mình.

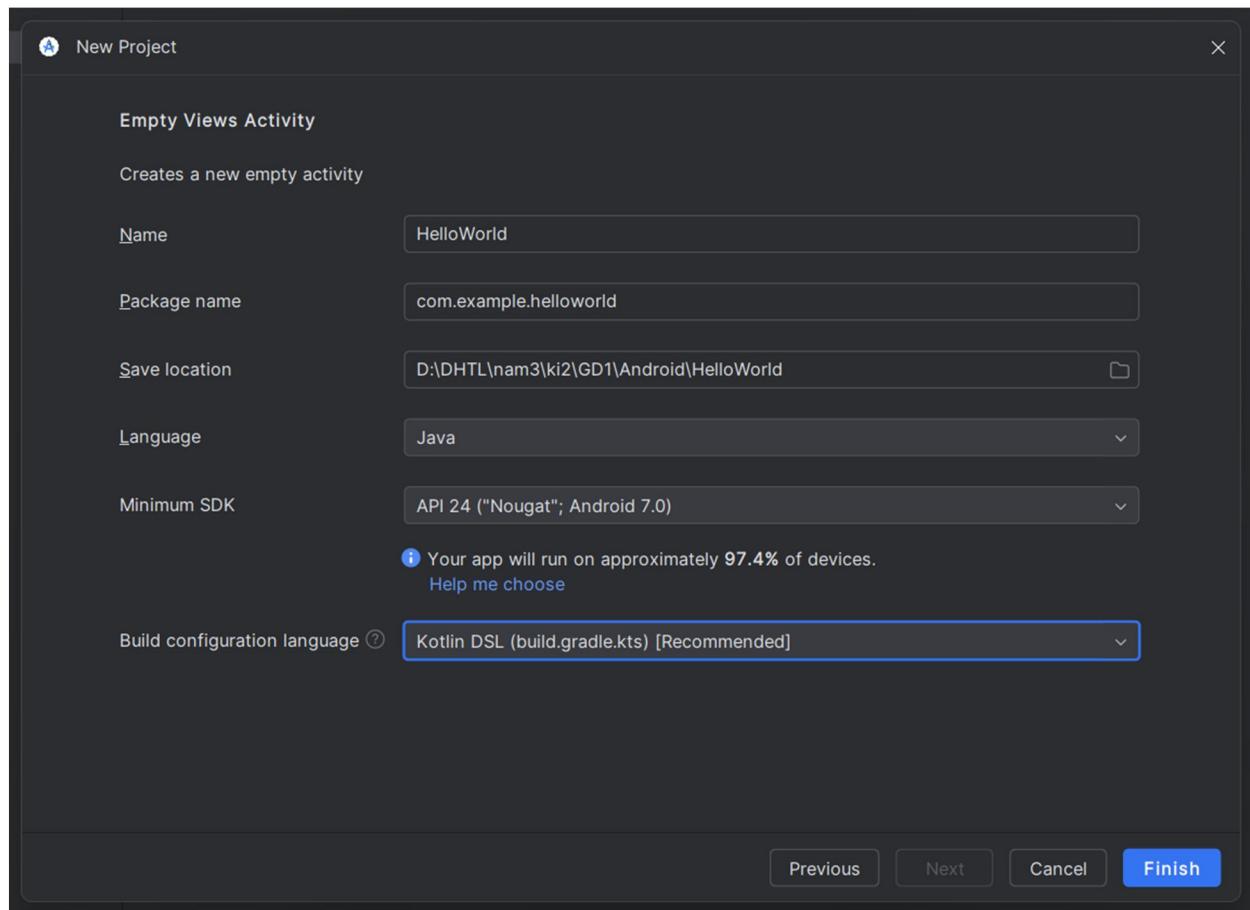
Khắc phục sự cố: Nếu bạn gặp vấn đề với quá trình cài đặt, hãy xem phần [ghi chú phát hành](#) của [Android Studio](#) hoặc tìm sự hỗ trợ từ người hướng dẫn của bạn.

Nhiệm vụ 2: Tạo 1 ứng dụng Hello World

Ở nhiệm vụ này, bạn sẽ tạo 1 ứng dụng và hiển thị "Hello World" nhằm xác nhận rằng Android Studio đã được cài đặt đúng cách, đồng thời làm quen với những kiến thức cơ bản về phát triển ứng dụng với Android Studio.

2.1. Tạo dự án

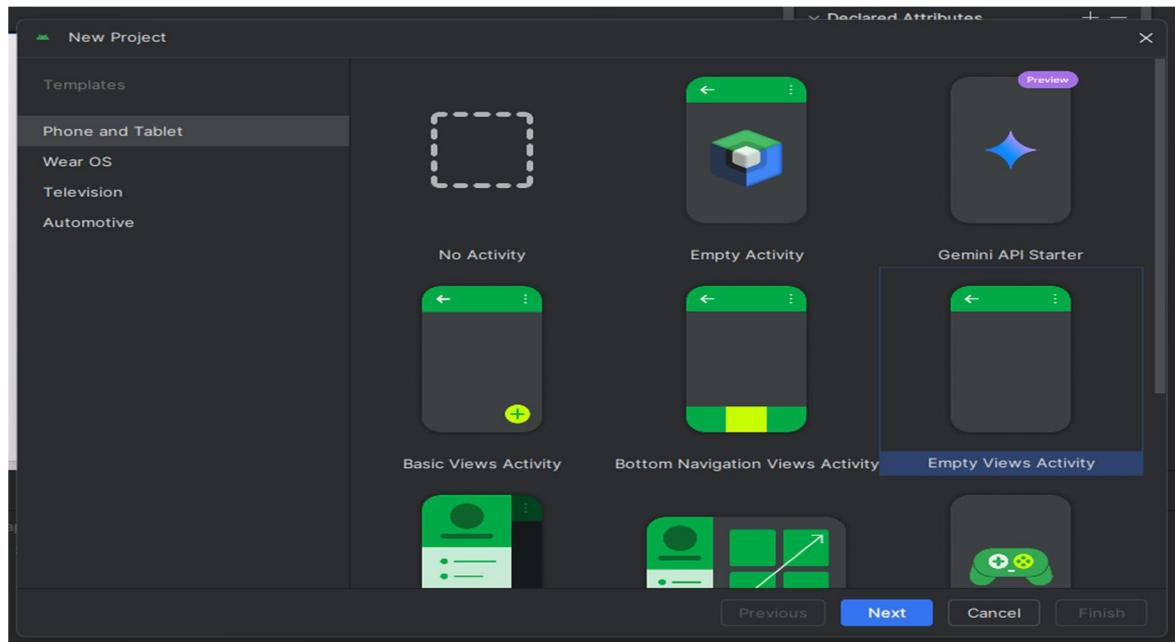
1. Mở Android Studio nếu nó chưa được mở
2. Trong màn hình chính hiển thị **Welcome to Android Studio**, hãy nhấp vào **Start a new Android Studio project**.
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** vào trường "Name".



4. Xác nhận rằng vị trí lưu trữ dự án (**Save location**) mặc định là nơi bạn muốn lưu trữ ứng dụng Hello World cũng như các dự án Android Studio khác, hoặc thay đổi thành thư mục mà bạn ưu tiên.
5. Xác nhận mặc định "android.example.com" cho trường Company Domain, hoặc tạo 1 tên miền khác riêng biệt.

Nếu bạn không có ý định phát hành ứng dụng, có thể chấp nhận giá trị mặc định. Lưu ý rằng thay đổi tên gói của ứng dụng của bạn sau này sẽ làm tăng khối lượng công việc

6. Không chọn tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấn **Next**.
7. Ở màn hình **Target Android Devices**, đảm bảo rằng lựa chọn **Phone and Tablet** đã được chọn. Kiểm tra xem Minimum SDK đã được đặt là **API 24: Android 7.0 Nougat**; nếu chưa, hãy sử dụng menu popup để cài đặt đúng.
Với cách thiết lập này được sử dụng bởi các ví dụ trong bài học của khóa học này. Tính đến thời điểm hiện tại, các thiết lập này giúp ứng dụng Hello World của bạn tương thích với 97% các thiết bị Android đang hoạt động trên Google Play Store.
8. Bỏ chọn tùy chọn **Include Instant App support** và tất cả các tùy chọn khác. Sau đó nhấn **Next**. Nếu dự án của bạn yêu cầu thêm các thành phần bổ sung cho target SDK mà bạn đã chọn, Android Studio sẽ tự động cài đặt chúng.\
9. Cửa sổ **Add an Activity** xuất hiện. Một Activity là một tác vụ đơn lẻ, tập trung mà người dùng có thể thực hiện và là thành phần thiết yếu của bất kỳ ứng dụng Android nào. Một Activity thường có kèm theo một bố cục (layout) xác định cách các thành phần giao diện người dùng xuất hiện trên màn hình. Android Studio cung cấp các mẫu Activity để giúp bạn bắt đầu. Đối với dự án Hello World, hãy chọn **Empty Activity** như hình minh họa bên dưới, và nhấn **Next**.



10. Màn hình **Configure Activity** xuất hiện (nó sẽ khác nhau tùy theo template bạn đã chọn ở bước trước). Theo mặc định, Activity rỗng được cung cấp bởi template có

tên là **MainActivity**. Bạn có thể thay đổi tên này nếu muốn, nhưng bài học này sử dụng **MainActivity**.

11. Hãy đảm bảo rằng tùy chọn **Generate Layout file** được đánh dấu. Tên layout mặc định là **activity_main**. Bạn có thể thay đổi điều này nếu muốn, nhưng bài học này sử dụng **activity_main**.
12. Đảm bảo rằng tùy chọn **Backwards Compatibility (App Compat)** được đánh dấu. Điều này giúp ứng dụng của bạn tương thích ngược với các phiên bản Android trước đó.
13. Nhấn **Finish**.

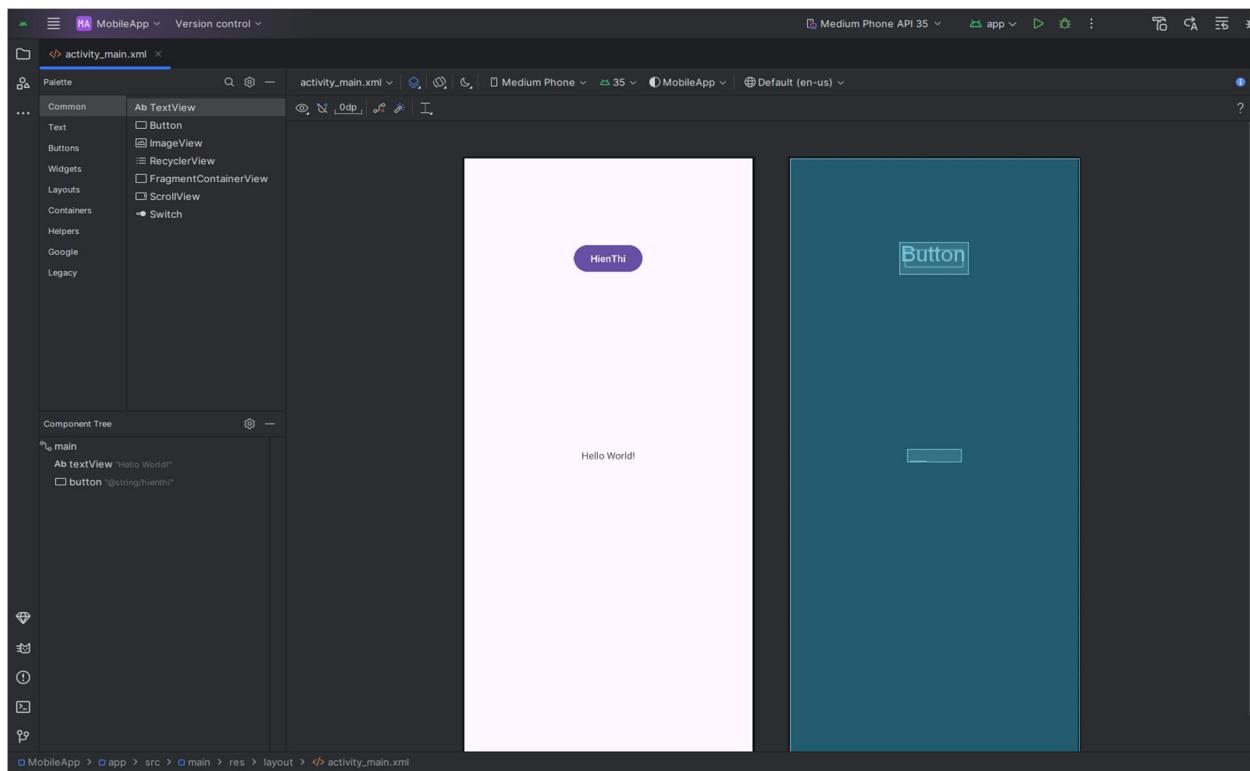
Android Studio sẽ tạo 1 thư mục chứa dự án của bạn, và xây dựng dự án sử dụng **Gradle**(quá trình này có thể mất một vài phút)

Mẹo: Tham khảo trang [Configure your build developer](#) của nhà phát triển để biết thêm thông tin chi tiết.

Bạn cũng có thể thấy một thông báo "Tip of the day" chứa các phím tắt và các mẹo hữu ích khác. Nhấn **Close** để đóng thông báo.

Trình chỉnh sửa soạn thảo của Android Studio xuất hiện. Thực hiện các bước sau:

1. Nhấp vào tab **activity_main.xml** để xem trình chỉnh sửa bố cục.
2. Nhấp vào tab **Design** trong trình chỉnh sửa bố cục (nếu chưa được chọn) để hiển thị phiên bản đồ họa của bố cục như hình minh họa bên dưới.



3. Chọn thẻ **MainActivity.java** để xem trình soạn thảo mã như được hiển thị bên dưới.

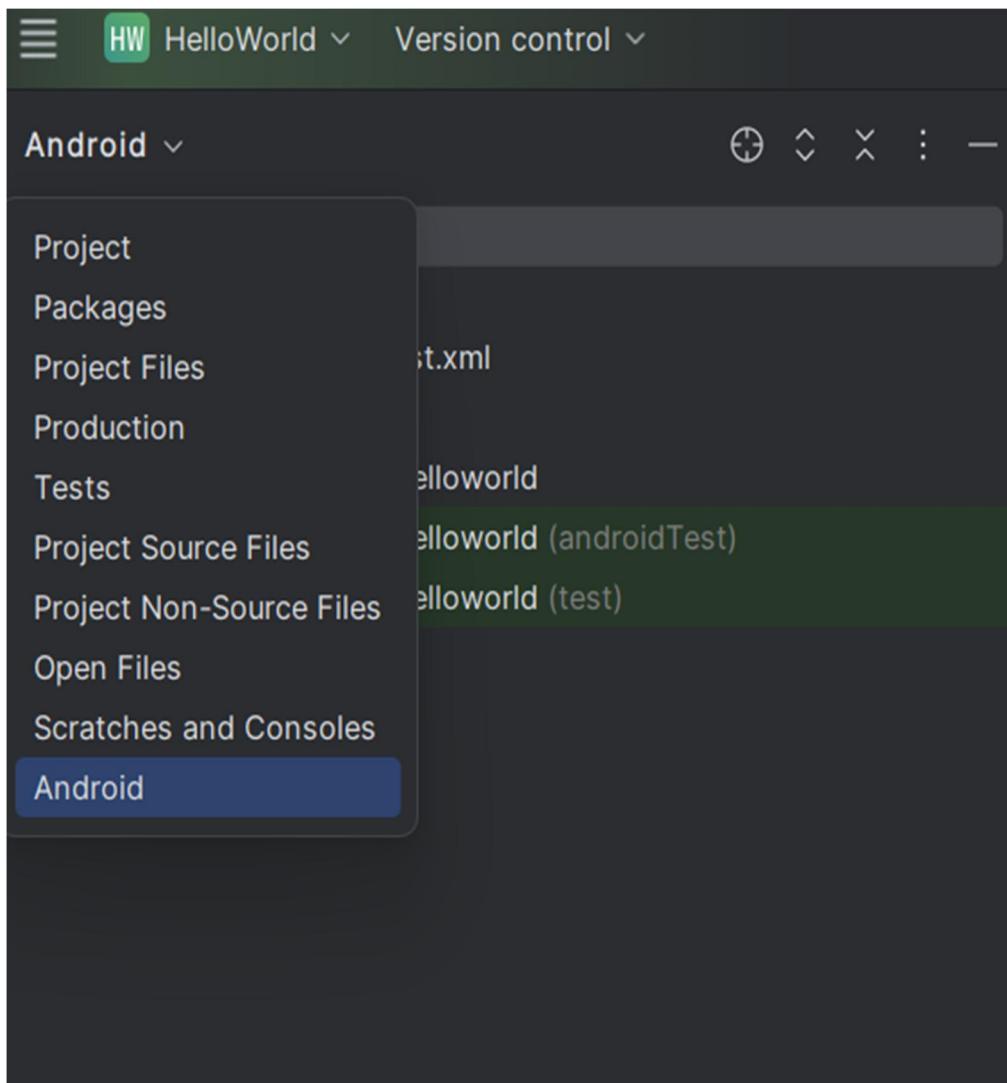
```
1 package com.example.mobileapp;
2
3 > import ...
4
5 >> public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        EdgeToEdge.enable( $thisEnableEdgeToEdge: this );
11        setContentView(R.layout.activity_main);
12        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
13            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
14            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
15            return insets;
16        });
17    }
18}
```

The screenshot shows the Android Studio code editor with the tab 'MainActivity.java' selected. The code defines a public class MainActivity that extends AppCompatActivity. The onCreate method is overridden to set the content view to R.layout.activity_main and handle edge insets using ViewCompat.setOnApplyWindowInsetsListener. The code uses Java 8 features like lambda expressions and the \$this variable reference style.

2.2. Khám phá Dự án > Bảng điều khiển Android

Trong bài thực hành này, bạn sẽ khám phá cách dự án được tổ chức trong Android Studio.

1. Nếu chưa được chọn, hãy nhấp vào tab **Project** ở cột tab dọc bên trái của cửa sổ Android Studio. Cửa sổ **Project** sẽ xuất hiện.
2. Để xem dự án theo cấu trúc tiêu chuẩn của Android, hãy chọn **Android** từ menu thả xuống ở phía trên của cửa sổ **Project**, như hình minh họa bên dưới.

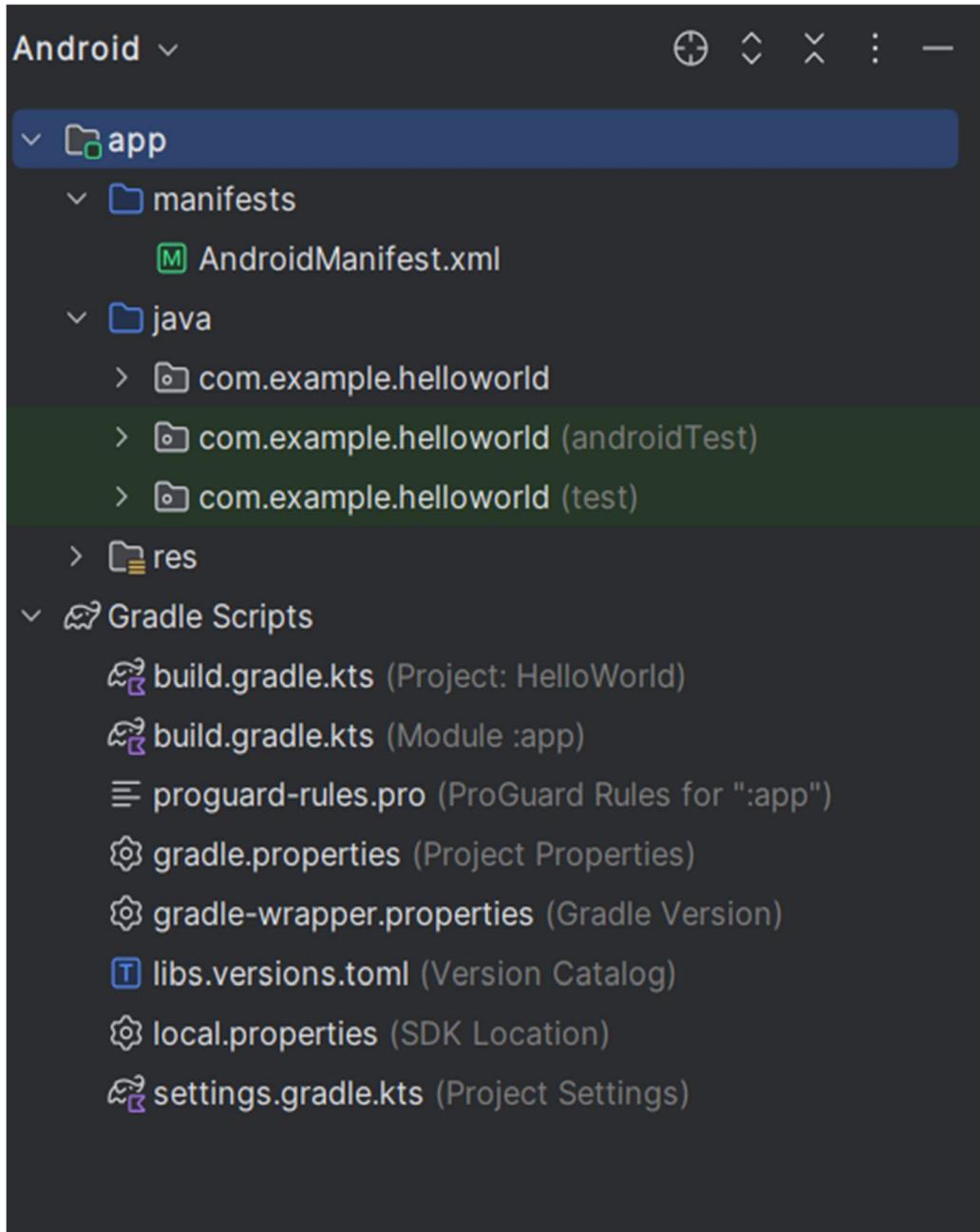


Chú ý: Chương này và các chương khác đề cập đến cửa sổ dự án, khi được đặt ở chế độ Android như là **Project > Android**.

2.3. Khám phá thư mục Gradle Scripts

thống xây dựng Gradle trong Android Studio giúp bạn dễ dàng tích hợp các tệp nhị phân bên ngoài hoặc các module thư viện khác vào quá trình xây dựng dưới dạng các phụ thuộc.

Khi bạn vừa tạo một dự án ứng dụng, cửa sổ **Project > Android** sẽ hiển thị với thư mục **Gradle Scripts** được mở rộng như hình minh họa bên dưới.

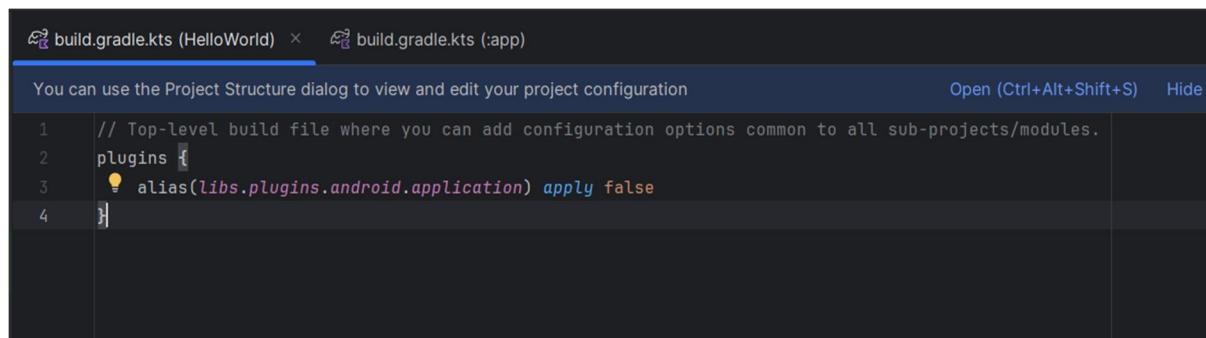


Thực hiện các bước sau để tìm hiểu về hệ thống **Gradle Scripts**:

1. Nếu thư mục **Gradle Scripts** chưa được mở, hãy nhấp vào biểu tượng hình tam giác để mở rộng nó. Thư mục này chứa tất cả các tệp cần thiết cho hệ thống build.
2. Nhìn vào tệp **build.gradle (Project: HelloWorld)**:

Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các module cấu thành dự án của bạn. Mỗi dự án Android Studio đều có một tệp build.gradle cấp cao. Hầu hết thời gian, bạn sẽ không cần chỉnh sửa bất kỳ tệp nào, nhưng việc hiểu rõ nội dung của nó vẫn rất hữu ích.

Mặc định, tệp build cấp cao sử dụng khái niệm **buildscript** để định nghĩa các kho lưu trữ và các phụ thuộc chung cho tất cả các module trong dự án. Khi phụ thuộc của bạn không phải là một thư viện nội bộ hay một tập tin cục bộ, Gradle sẽ tìm các tệp cần thiết ở các kho lưu trữ trực tuyến được chỉ định trong khái niệm **repositories** của tệp này. Mặc định, các dự án Android Studio mới khai báo **JCenter** và **Google** (bao gồm cả **Google Maven repository**) làm các vị trí kho lưu trữ:



```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
    alias(libs.plugins.android.application) apply false
}
```

3. Nhìn vào tệp **build.gradle(Module:app)**:

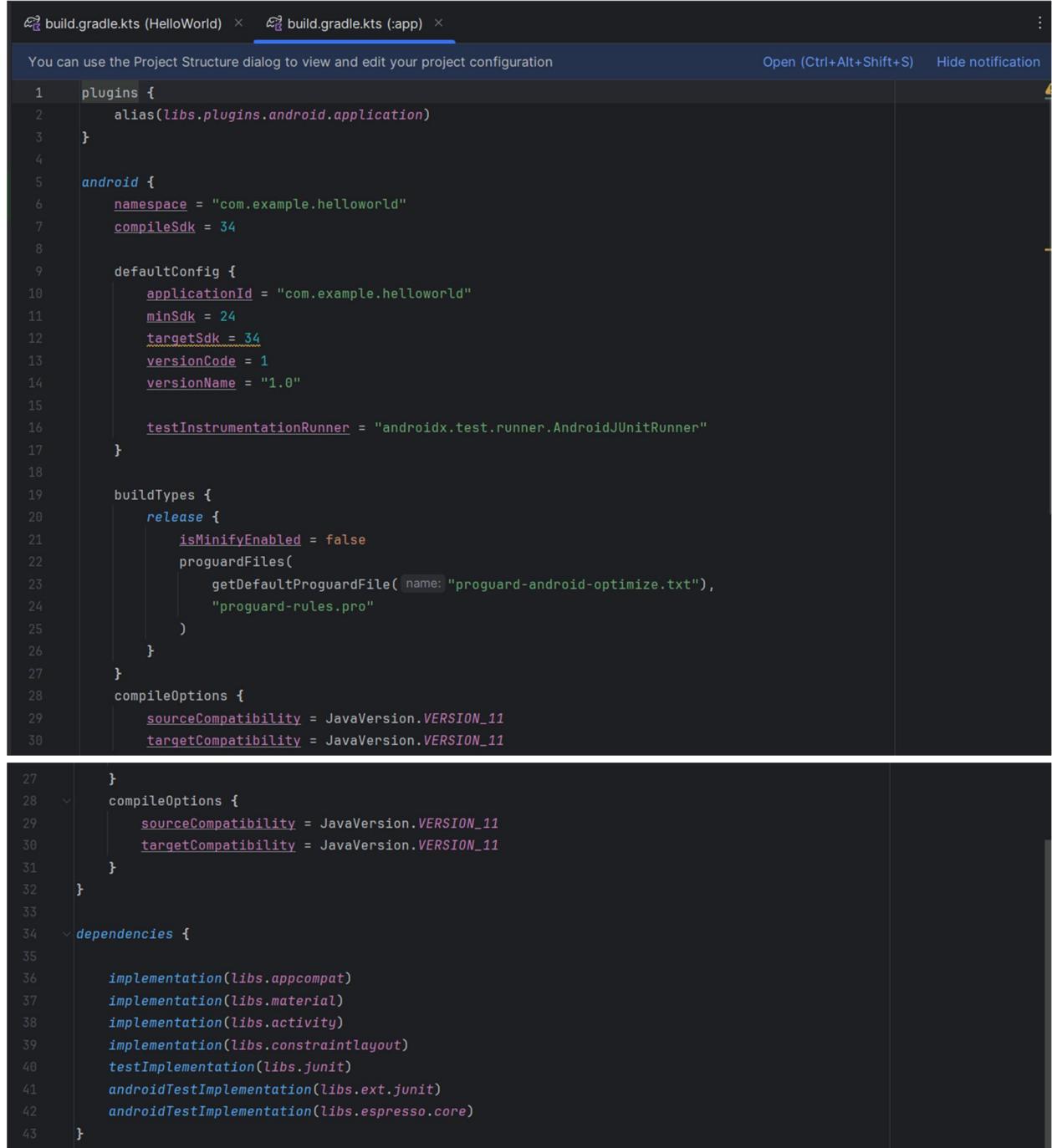
Bên cạnh tệp **build.gradle** cấp dự án, mỗi mô-đun đều có tệp build.gradle riêng, cho phép bạn định cấu hình cài đặt bản dựng cho từng mô-đun cụ thể (ứng dụng HelloWorld chỉ có một module duy nhất). Việc cấu hình các thiết lập build này giúp bạn cung cấp các tùy chọn đóng gói tùy chỉnh, chẳng hạn như thêm các kiểu xây dựng và các sản phẩm đi kèm. Bạn cũng có thể ghi đè các thiết lập có trong tệp **AndroidManifest.xml** hoặc tệp **build.gradle** cấp dự án.

Tệp này thường là tệp bạn sẽ chỉnh sửa khi thay đổi các cấu hình cấp ứng dụng, chẳng hạn như khai báo các phụ thuộc trong phần **dependencies**. Bạn có thể khai báo một phụ thuộc thư viện bằng cách sử dụng một trong các cấu hình phụ thuộc khác nhau; mỗi cấu hình phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện. Ví dụ, câu lệnh

```
implementation fileTree(dir: 'libs', include: ['*.jar'])
```

thêm 1 phụ thuộc cho tất cả các tệp “.jar” ở trong thư mục libs.

Dưới đây là tệp **build.gradle (Module: app)** cho ứng dụng HelloWorld:



The screenshot shows the Android Studio interface with two tabs open: "build.gradle.kts (HelloWorld)" and "build.gradle.kts (:app)". The right tab is selected and displays the Gradle script code. The code defines various configurations like defaultConfig, buildTypes (release), compileOptions, and dependencies, along with implementation statements for various libraries.

```
plugins {
    alias(libs.plugins.android.application)
}

android {
    namespace = "com.example.helloworld"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.helloworld"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}

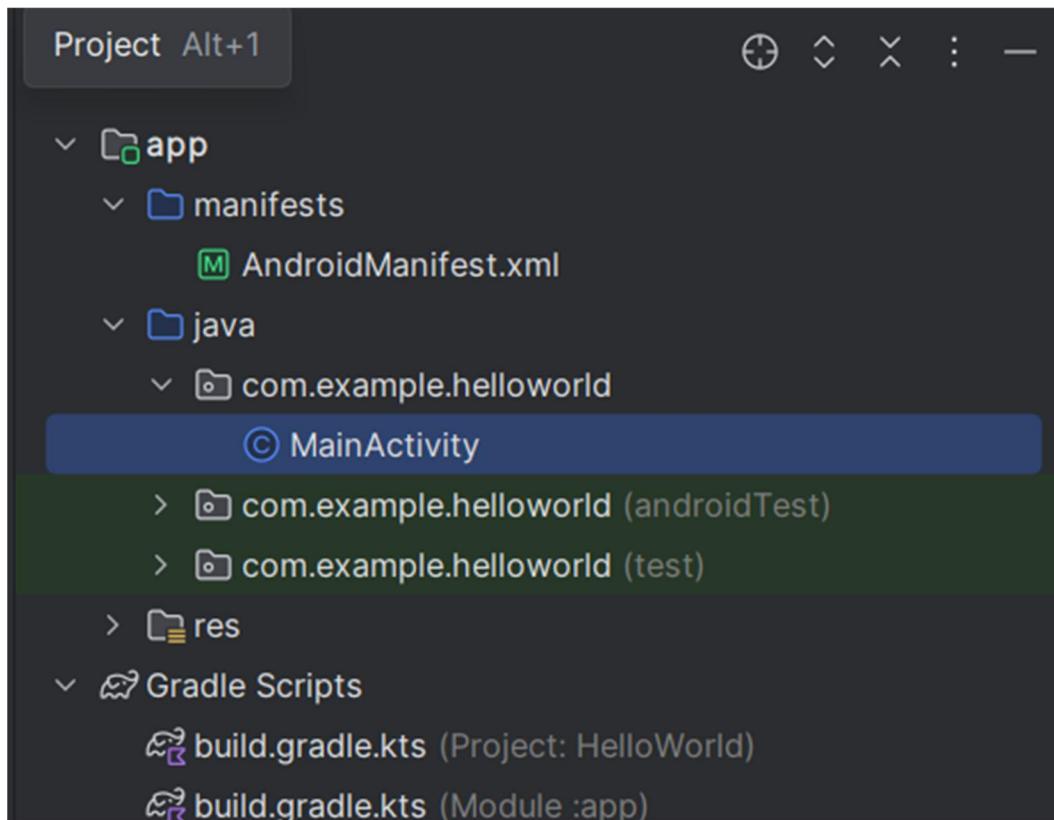
dependencies {
    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
}
```

4. Nhấp vào hình tam giác để đóng Gradle Scripts.

2.4. Khám phá các thư mục ứng dụng và thư mục res

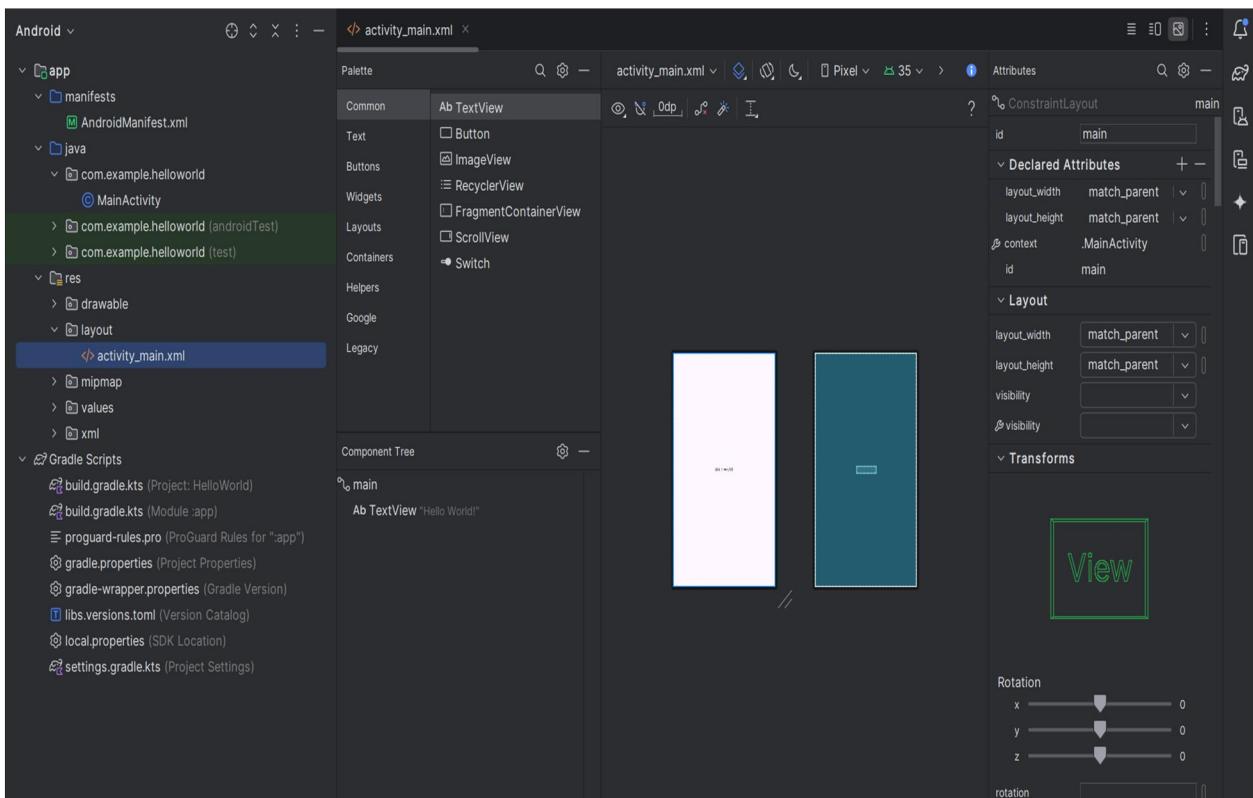
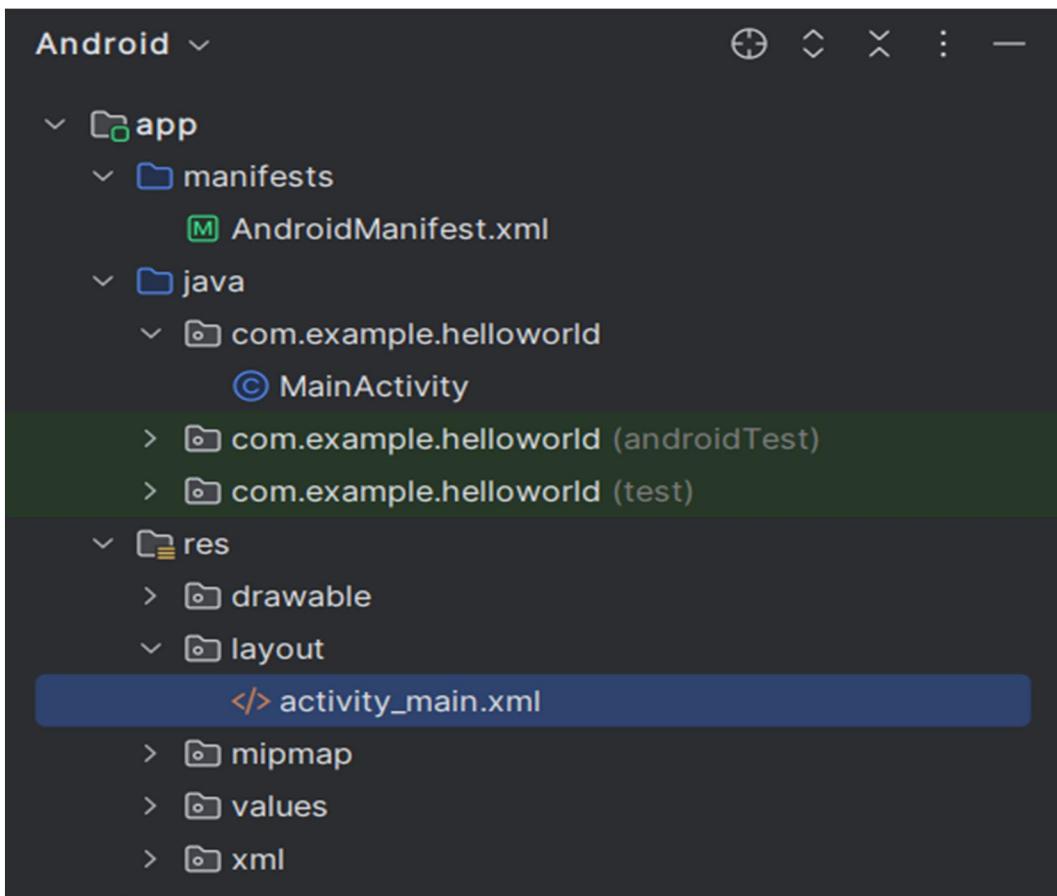
Tất cả code và tài nguyên cho ứng dụng đều nằm ở thư mục ứng dụng và thư mục res

1. Mở thư mục app, sau đó mở thư mục java, và cuối cùng là thư mục com.example.android.helloworld để thấy tệp MainActivity.java. Nhấp đúp vào tệp này sẽ mở nó trong trình soạn thảo mã nguồn.



Thư mục **java** chứa các tệp lớp Java được sắp xếp trong ba thư mục con, như hình minh họa ở trên. Thư mục **com.example.android.helloworld** (hoặc tên miền mà bạn đã chỉ định) chứa tất cả các tệp của một gói ứng dụng. Hai thư mục còn lại được sử dụng cho mục đích kiểm thử và sẽ được mô tả trong một bài học khác. Đối với ứng dụng Hello World, chỉ có một gói và nó chứa tệp **MainActivity.java**. Tên của Activity đầu tiên (màn hình) mà người dùng thấy, đồng thời khởi tạo các tài nguyên cho toàn bộ ứng dụng, thường được đặt là **MainActivity** (phần mở rộng tệp được bỏ qua trong cửa sổ **Project > Android**).

2. Mở thư mục res và sau đó mở thư mục layout, nháy đúp chuột và tệp **activity_main.xml** để mở nó trong trình chỉnh sửa bố cục.



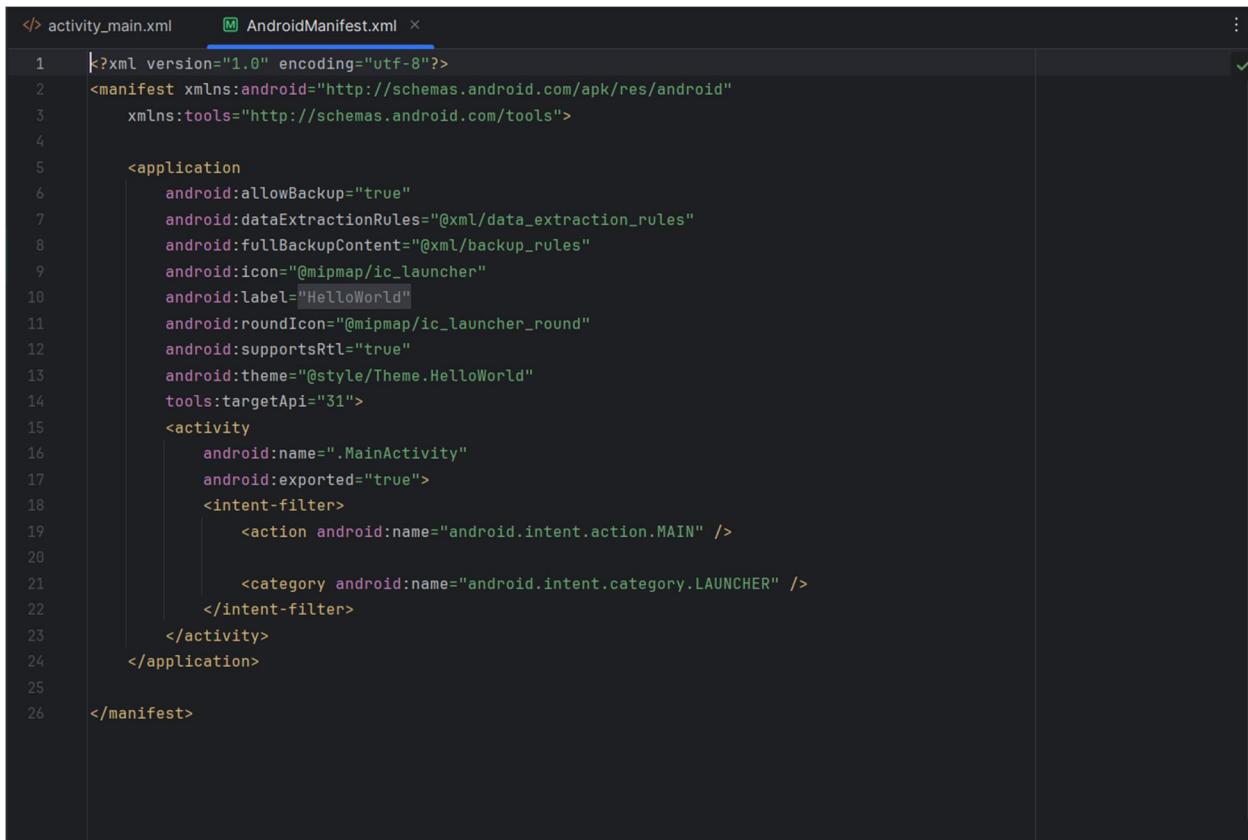
Thư mục **res** chứa các tài nguyên của ứng dụng, chẳng hạn như bố cục chuỗi ký tự và hình ảnh. Một **Activity** thường được liên kết với một bố cục giao diện người dùng được định nghĩa trong một tệp XML, và tệp này thường được đặt tên theo tên của **Activity** đó.

2.5. Khám phá thư mục manifests

Thư mục **manifests** chứa các tệp cung cấp thông tin thiết yếu về ứng dụng cho hệ thống Android – thông tin mà hệ thống cần có trước khi có thể chạy bất kỳ mã nào của ứng dụng.

1. Mở rộng thư mục **manifests**.

2. Mở tệp **AndroidManifest.xml**.



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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.HelloWorld"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Tệp **AndroidManifest.xml** mô tả tất cả các thành phần của ứng dụng Android của bạn. Mỗi thành phần của ứng dụng, chẳng hạn như từng **Activity**, đều phải được khai báo trong tệp XML này. Trong các bài học tiếp theo của khóa học, bạn sẽ chỉnh sửa tệp này để thêm các tính năng và cấp quyền truy cập cho các tính năng đó. Để tìm hiểu thêm, hãy xem mục [App Manifest Overview](#).

Nhiệm vụ 3: Sử dụng thiết bị ảo (trình giả lập)

Trong nhiệm vụ này, bạn sẽ sử dụng [Android Virtual Device \(AVD\) manager](#) để tạo một thiết bị ảo (còn được gọi là trình giả lập) mô phỏng cấu hình của một loại thiết bị Android cụ thể, và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng **Android Emulator** có các yêu cầu bổ sung ngoài các yêu cầu hệ thống cơ bản của Android Studio.

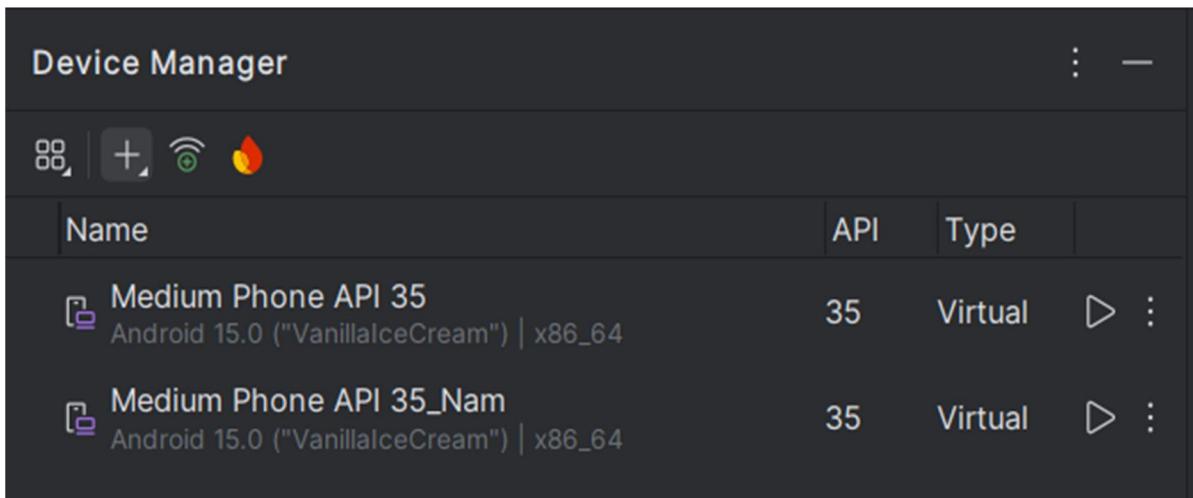
Khi sử dụng Trình quản lý AVD, bạn sẽ xác định các đặc điểm phần cứng của thiết bị, cấp độ API, bộ nhớ, giao diện và các thuộc tính khác của thiết bị và lưu dưới dạng thiết bị ảo. Với các thiết bị ảo, bạn có thể kiểm tra ứng dụng trên các cấu hình thiết bị khác nhau (như máy tính bảng và điện thoại) với các cấp độ API khác nhau mà không cần phải sử dụng thiết bị vật lý.

3.1. Tạo 1 thiết bị Andriod ảo (AVD)

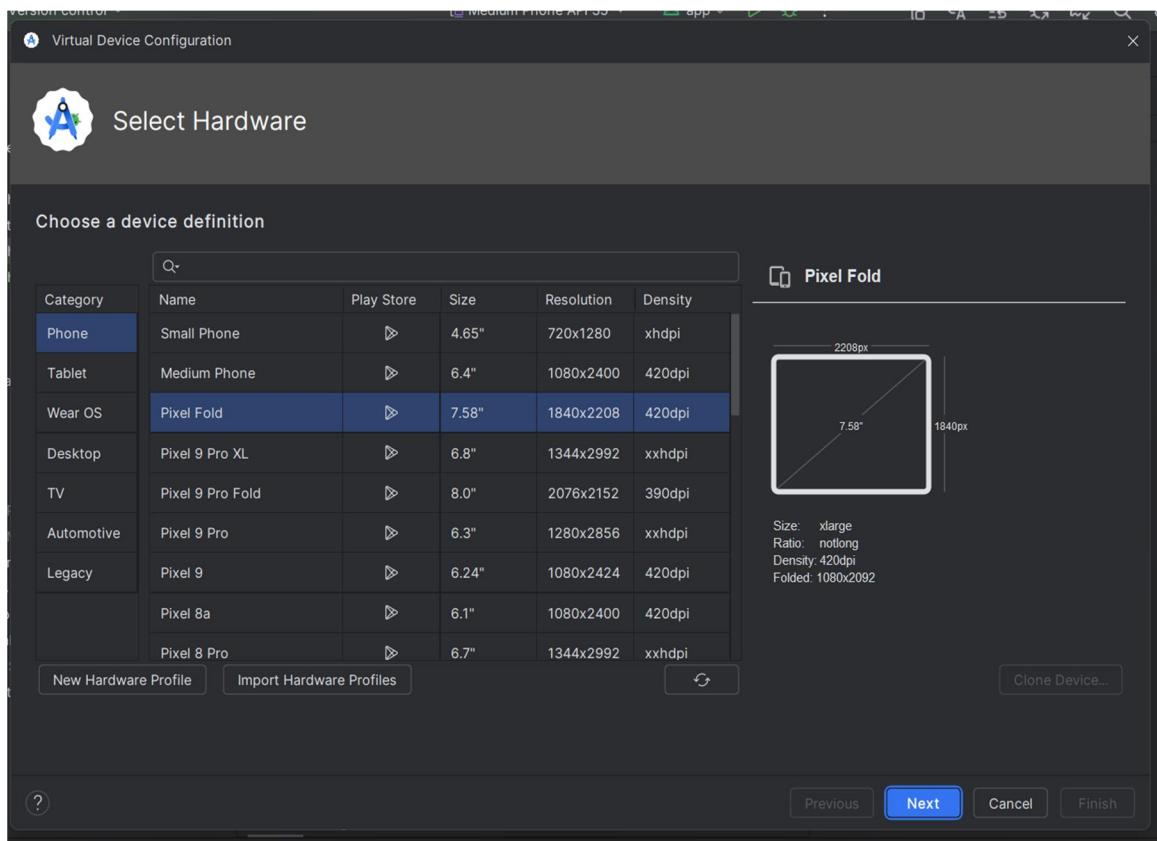
Để chạy emulator trên máy tính của bạn, bạn cần tạo một cấu hình mô tả thiết bị ảo.

- Ở trong **Android Studio**, chọn: **Tools > Android > AVD Manager**, hoặc nhập vào

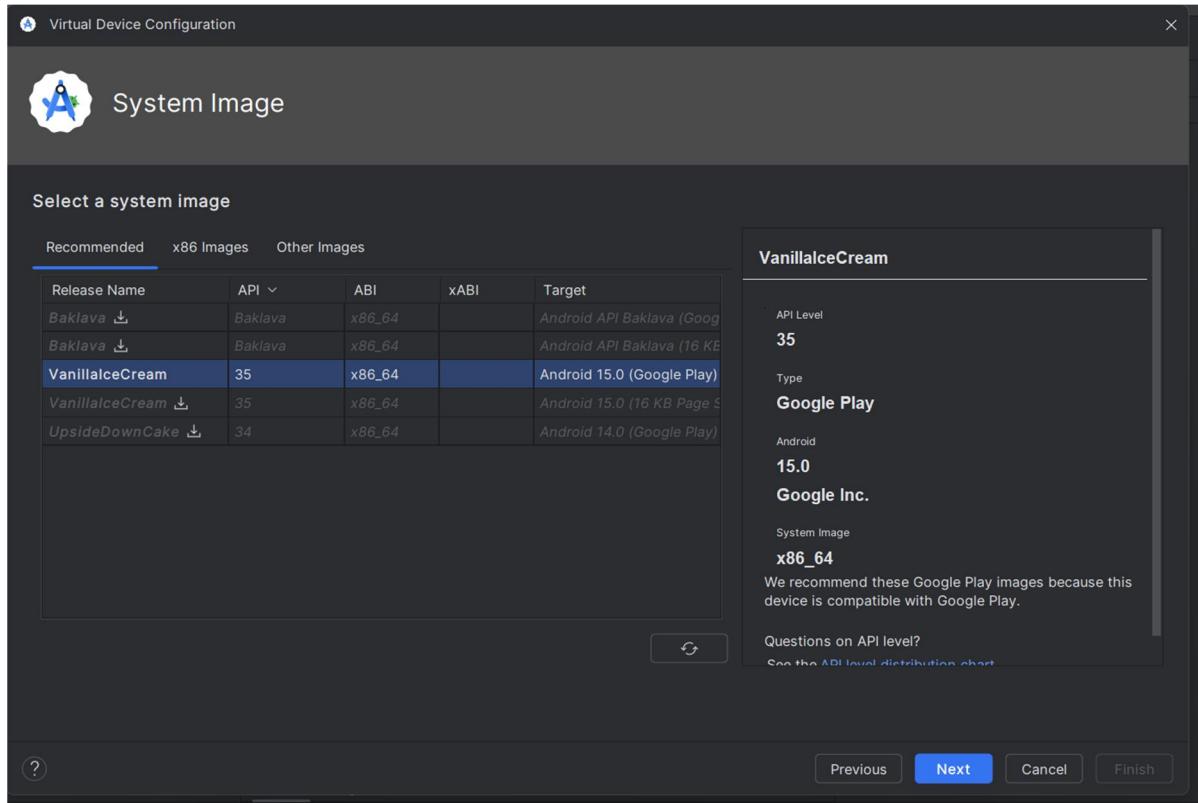
biểu tượng **Device Manager**  trên thanh công cụ. Màn hình **Device Manager** xuất hiện. Nếu bạn đã tạo thiết bị ảo trước đó, danh sách sẽ hiển thị chúng. Nếu chưa có, danh sách sẽ trống.



- Nhấp vào **+Create Virtual Device**. Cửa sổ Select Hardware sẽ xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng cung cấp một cột cho kích thước màn hình đường chéo (Size), độ phân giải màn hình tính bằng pixel (Resolution) và mật độ pixel(Density).



3. Chọn một thiết bị như **Pixel 9 Pro XL** hoặc **Pixel Fold** và nhấp vào **Next**. Màn hình ảnh hệ thống xuất hiện.
4. Nhấp vào thẻ **Recommended** nếu chưa chọn và chọn phiên bản hệ thống Android nào để chạy trên thiết bị ảo (chẳng hạn như **VanillaIceCream**).



Có nhiều phiên bản khả dụng hơn so với những phiên bản được hiển thị trong thẻ **Recommended**. Hãy xem các hệ điều hành **x86** và các hệ điều hành khác để xem chúng.

Nếu có liên kết **Download** bên cạnh một hệ điều hành bạn muốn sử dụng, điều đó có nghĩa là nó chưa được cài đặt. Nhấn vào liên kết để bắt đầu tải xuống, sau đó nhấn **Finish** khi quá trình tải hoàn tất.

- Sau cùng chọn hệ điều hành, nhấn **Next**. Cửa sổ **Android Virtual Device (AVD)** sẽ xuất hiện. Bạn cũng có thể đổi tên thiết bị ảo (AVD). Kiểm tra lại cấu hình của bạn và nhấn **Finish**.

3.2. Chạy ứng dụng trên thiết bị ảo

Ở nhiệm vụ này, bạn sẽ hoàn tất chạy ứng dụng Hello World của mình.

- Trong Android Studio, chọn **Run > Run app** hoặc nhấn vào biểu tượng **Run** trên thanh công cụ.
- Trong cửa sổ **Select Deployment Target**, dưới mục **Available Virtual Devices**, chọn thiết bị ảo mà bạn vừa tạo và nhấn **OK**.

Name	API	Type		
Medium Phone API 35 Android 15.0 ("VanillaIceCream") x86_64	35	Virtual	▷	⋮
Medium Phone API 35_Nam Android 15.0 ("VanillaIceCream") x86_64	35	Virtual	▷	⋮

Trình giả lập sẽ khởi động giống như một thiết bị thật. Tùy vào tốc độ của máy tính, quá trình này có thể mất một chút thời gian. Ứng dụng của bạn được xây dựng và khi trình giả lập đã sẵn sàng, Android Studio sẽ tải ứng dụng lên trình giả lập và chạy ứng dụng.

Bạn sẽ thấy ứng dụng Hello World như trong hình như sau



Mẹo: Khi thử nghiệm trên thiết bị ảo, bạn nên khởi động thiết bị một lần, ngay từ đầu phiên của bạn. Bạn không nên đóng thiết bị cho đến khi hoàn tất thử nghiệm ứng dụng, để ứng dụng không phải trải qua quá trình khởi động thiết bị một lần nữa. Để đóng thiết bị ảo, hãy nhấp vào nút X ở đầu trình giả lập, chọn **Thoát** từ menu hoặc nhấn **Control-Q** trong Windows hoặc **Command-Q** trong macOS.

Nhiệm vụ 4: (Tùy chọn) Sử dụng thiết bị vật lý

Trong nhiệm vụ cuối cùng này, bạn sẽ chạy ứng dụng của mình trên thiết bị di động vật lý như điện thoại hoặc máy tính bảng. Bạn luôn phải kiểm tra ứng dụng của mình trên cả thiết bị ảo và vật lý.

Những gì bạn cần:

- Một thiết bị Android, chẳng hạn như điện thoại hoặc máy tính bảng.
- Cáp dữ liệu để kết nối thiết bị Android của bạn với máy tính qua cổng USB.
- Nếu bạn đang sử dụng hệ thống Linux hoặc Windows, có thể cần thực hiện thêm một số bước để chạy trên thiết bị phần cứng. Kiểm tra tài liệu [Using Hardware Devices](#). Bạn cũng có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Đối với trình điều khiển USB trên Windows, xem [OEM USB Drivers](#).

4.1. Bật gỡ lỗi USB

Để Android Studio có thể giao tiếp với thiết bị của bạn, bạn phải bật **Gỡ lỗi USB** trên thiết bị Android. Tùy chọn này nằm trong **Cài đặt nhà phát triển** của thiết bị của bạn.

Trên Android 4.2 trở lên, màn hình **Cài đặt nhà phát triển** bị ẩn theo mặc định. Để hiển thị và bật **Gỡ lỗi USB**, hãy làm theo các bước sau:

1. Trên thiết bị của bạn, mở **Cài đặt**, tìm **Giới thiệu về điện thoại**, nhấn vào **Giới thiệu về điện thoại**, sau đó chạm vào **Số bản dựng bảy lần**.
2. Quay lại màn hình trước đó (**Cài đặt / Hệ thống**), bạn sẽ thấy mục **Tùy chọn nhà phát triển**. Nhấn vào **Tùy chọn nhà phát triển**.
3. Tìm và bật **Gỡ lỗi USB**.

4.2. Chạy ứng dụng ở trên thiết bị

Bây giờ bạn có thể kết nối thiết bị và chạy ứng dụng từ Android Studio.

1. Kết nối thiết bị của bạn với máy tính phát triển bằng cáp USB.

2. Nhấn vào nút **Run** trên thanh công cụ. Cửa sổ **Select Deployment Target** sẽ mở ra, hiển thị danh sách các trình giả lập và thiết bị đã kết nối.
3. Chọn thiết bị của bạn và nhấn **Ok**.

Android Studio sẽ cài đặt và chạy ứng dụng trên thiết bị của bạn.

Khắc phục sự cố

Nếu Android Studio không nhận diện được thiết bị của bạn, hãy thử các bước sau:

1. Rút và cắm lại thiết bị.
2. Khởi động lại Android Studio.

Nếu máy tính vẫn không tìm thấy thiết bị hoặc hiển thị là “**unauthorized**” (**chưa được ủy quyền**), hãy làm theo các bước sau:

1. Rút thiết bị ra.
2. Trên thiết bị, mở **Tùy chọn nhà phát triển** trong **Cài đặt**.
3. Nhấn vào **Thu hồi quyền gỡ lỗi USB**.
4. Kết nối lại thiết bị với máy tính.
5. Khi có thông báo yêu cầu quyền, hãy cấp quyền.

Bạn có lẽ cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Xem tài liệu sử dụng thiết bị phần cứng

Nhiệm vụ 5: Thay đổi cấu hình Gradle của ứng dụng

Trong nhiệm vụ này, bạn sẽ thay đổi một số cấu hình của ứng dụng trong tệp **build.gradle (Module: app)** để tìm hiểu cách thực hiện thay đổi và đồng bộ chúng với dự án Android Studio.

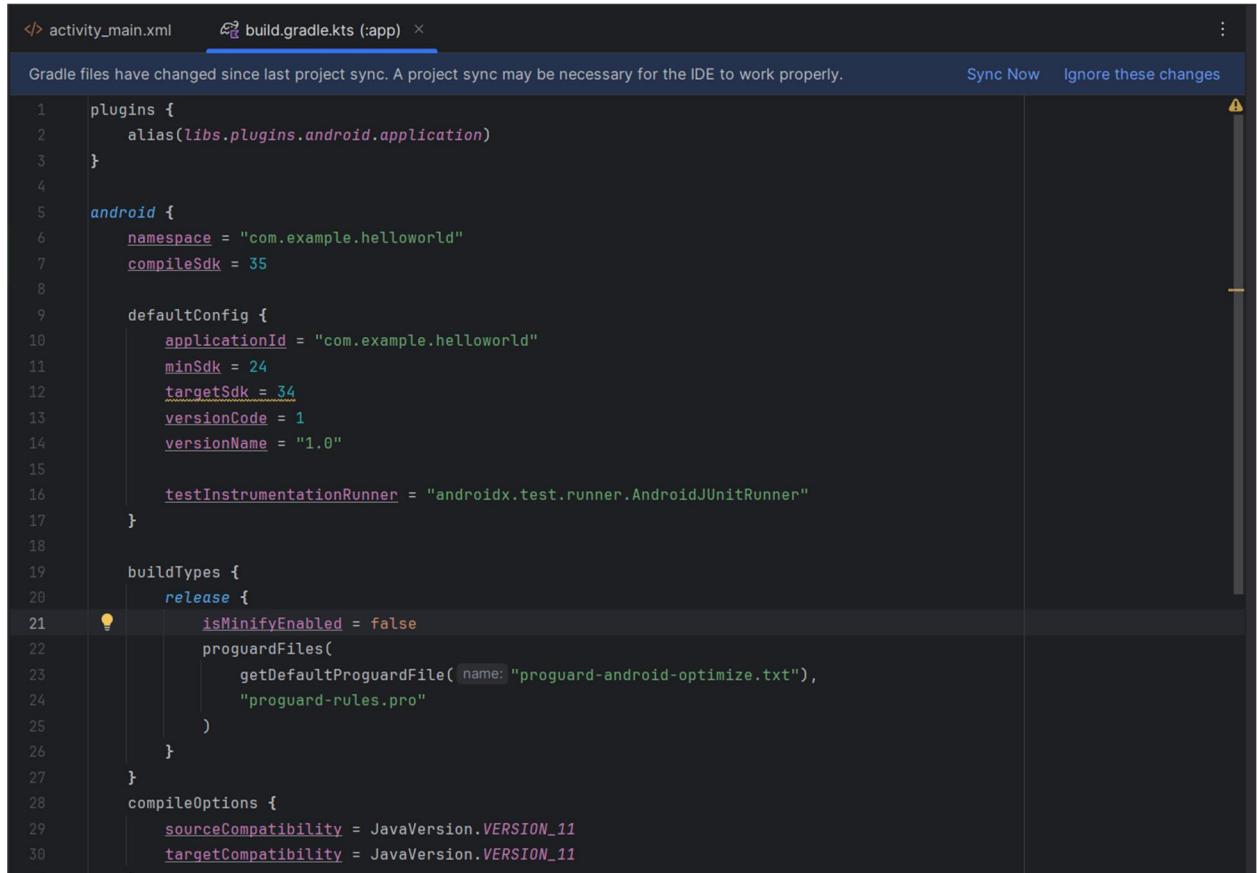
5.1. Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Làm theo các bước sau :

1. Mở rộng thư mục **Gradle Scripts** nếu thư mục này chưa mở và nhấp đúp vào tệp **build.gradle(Module:app)**.

Nội dung của tệp sẽ xuất hiện trong trình chỉnh sửa mã.

- Trong khối *defaultConfig*, hãy thay đổi giá trị của *minSdk* thành 24 như hiển thị bên dưới (ban đầu giá trị này được đặt thành 23).



```
plugins {
    alias(libs.plugins.android.application)
}

android {
    namespace = "com.example.helloworld"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.helloworld"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}
```

Trình chỉnh sửa mã hiển thị thanh thông báo ở trên cùng với liên kết **Sync Now** ngay.

5.2. Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi đối với các tệp cấu hình bản dựng trong một dự án, Android Studio yêu cầu bạn đồng bộ các tệp dự án để có thể nhập các thay đổi cấu hình bản dựng và chạy một số kiểm tra để đảm bảo cấu hình sẽ không tạo ra lỗi.

Để đồng bộ các tệp dự án, hãy nhấp vào **Sync Now** trên thanh thông báo xuất hiện khi thực hiện thay đổi (như được hiển thị trong hình trước) hoặc nhấp vào biểu tượng **Sync Project with Gradle Files** trên thanh công cụ.

Khi quá trình đồng bộ hóa Gradle hoàn tất, thông báo Gradle build finished sẽ xuất hiện ở góc dưới bên trái của cửa sổ Android Studio.

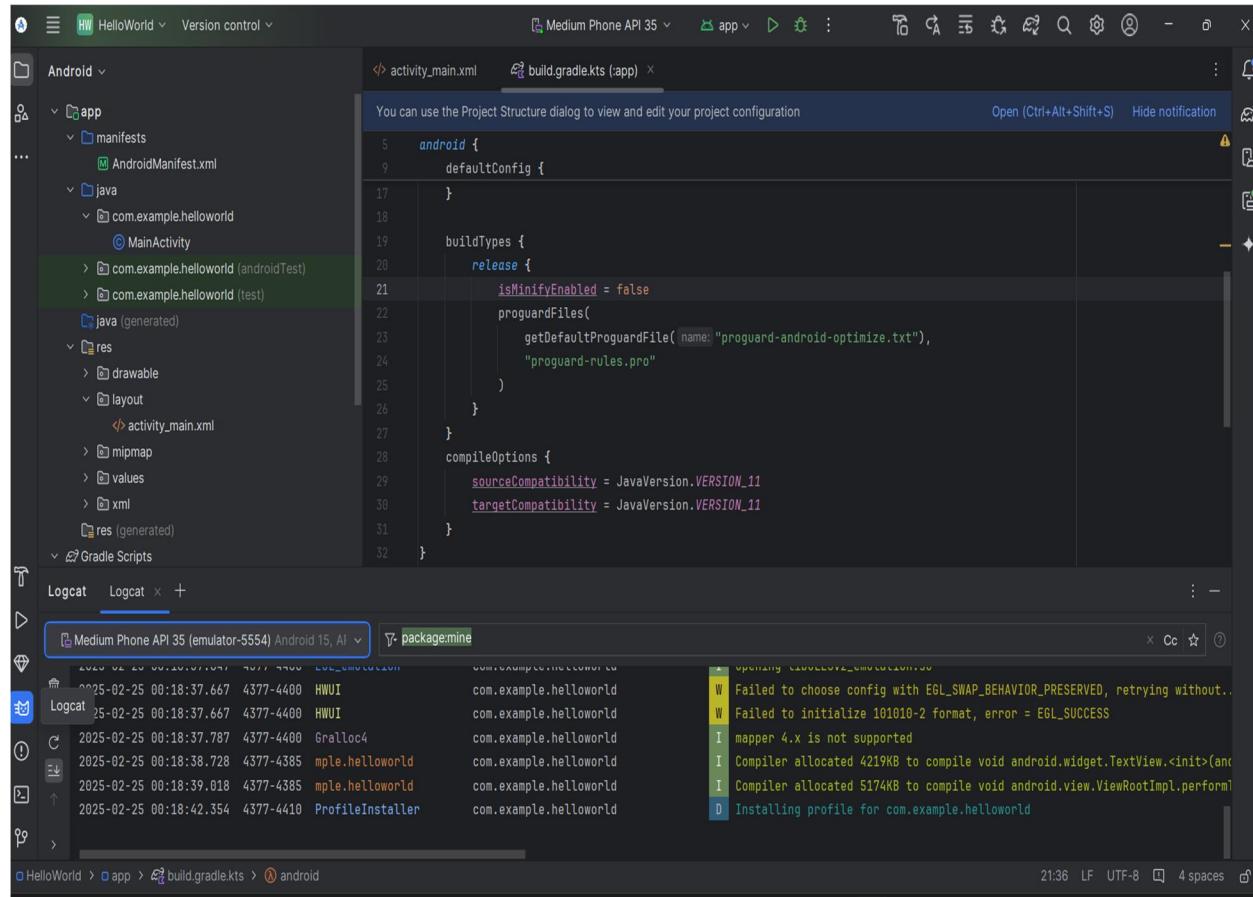
Để tìm hiểu sâu hơn về Gradle, hãy xem tài liệu [Tổng quan về hệ thống bản dựng \(Build System Overview\)](#) và [Cấu hình bản dựng Gradle](#)(Configuring Gradle Builds)

Nhiệm vụ 6: Thêm câu lệnh nhật ký vào ứng dụng của bạn

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh Log vào ứng dụng của mình, hiển thị các thông báo trong cửa sổ **Logcat**. Thông báo Log là một công cụ gỡ lỗi mạnh mẽ mà bạn có thể sử dụng để kiểm tra các giá trị, đường dẫn thực thi và báo cáo các ngoại lệ.

6.1. Cửa sổ Logcat

Để xem cửa sổ Logcat, hãy nhấp vào thẻ Logcat ở cuối cửa sổ Android Studio như được hiển thị trong hình bên dưới.



Trong hình trên:

- Thẻ **Logcat** để mở và đóng cửa sổ **Logcat**, hiển thị thông tin về ứng dụng của bạn khi ứng dụng đang chạy. Nếu bạn thêm các câu lệnh Log vào ứng dụng, các thông báo Log sẽ xuất hiện ở đây.

2. Bộ menu cấp độ Log được đặt thành **Verbose** (mặc định), hiển thị tất cả các thông báo Log. Các cài đặt khác bao gồm **Debug**, **Error**, **Info** và **Warn**.

6.2. Thêm các câu lệnh nhật ký vào ứng dụng của bạn

Các câu lệnh nhật ký trong mã ứng dụng của bạn hiển thị thông báo trong ngăn Logcat. Ví dụ:

```
Log.d("MainActivity", "Hello World");
```

Các phần của thông báo là:

- Log: Lớp Log để gửi thông báo nhật ký đến cửa sổ Logcat.
- d: Cài đặt mức Nhật ký gỡ lỗi để lọc hiển thị thông báo nhật ký trong cửa sổ Logcat. Các mức nhật ký khác là e cho Lỗi, w cho Cảnh báo và i cho Thông tin.
- "MainActivity": Đối số đầu tiên là một thẻ có thể được sử dụng để lọc thông báo trong cửa sổ Logcat. Đây thường là tên của Hoạt động mà thông báo bắt nguồn. Tuy nhiên, bạn có thể biến điều này thành bất kỳ thứ gì hữu ích cho bạn để gỡ lỗi.

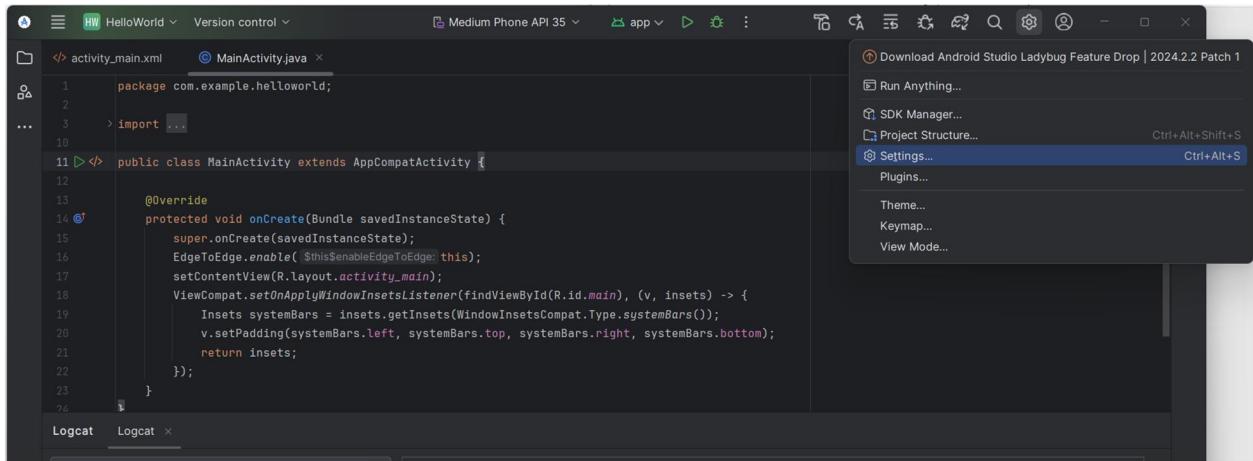
Theo quy ước, thẻ nhật ký được định nghĩa là hằng số cho hoạt động:

```
private static final String LOG_TAG = MainActivity.class.getSimpleName();
```

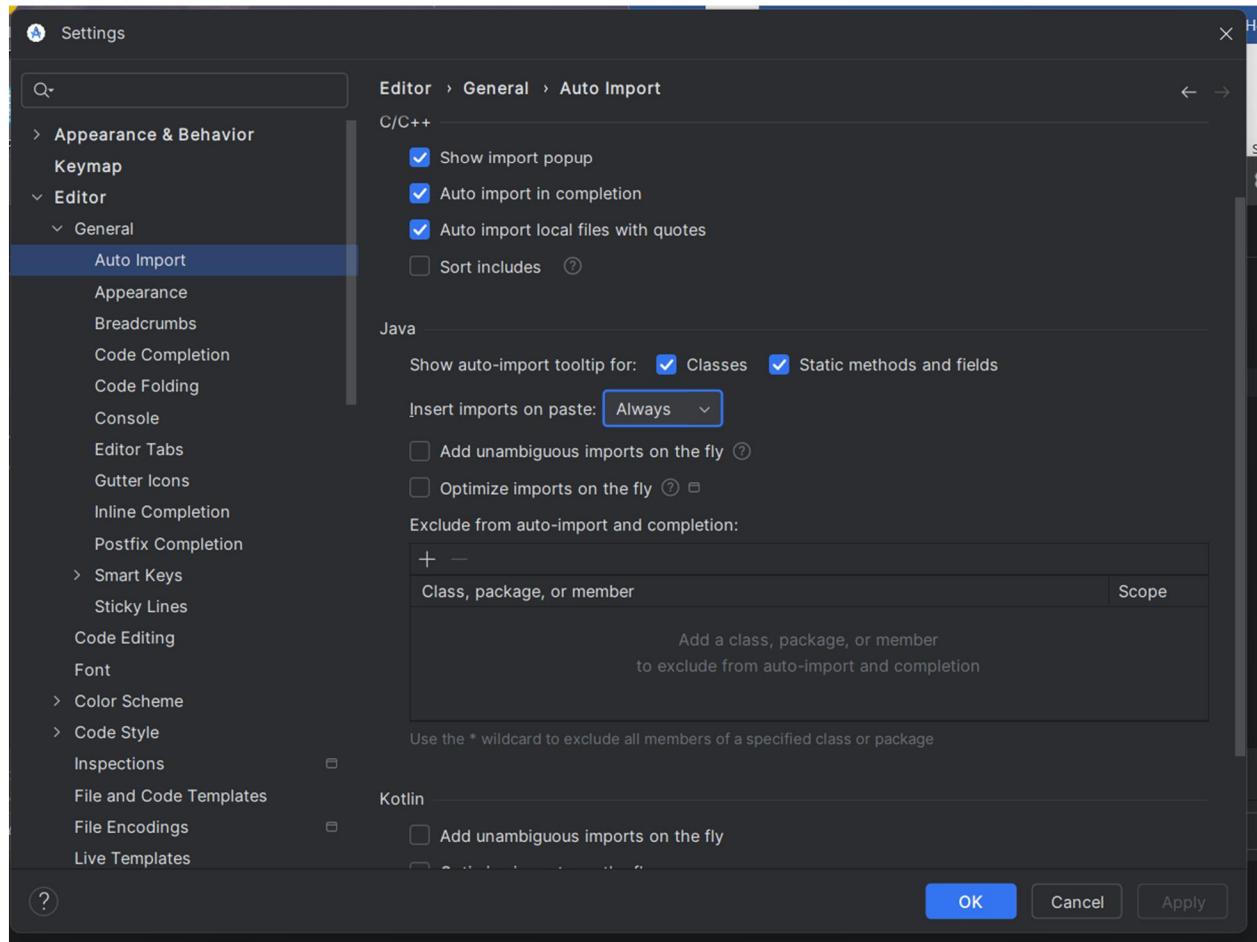
- "Hello world": Đối số thứ hai là thông điệp thực tế.

Thực hiện theo các bước sau:

1. Mở ứng dụng Hello World của bạn trong Android studio và mở MainActivity.
2. Để tự động thêm các mục nhập rõ ràng vào dự án của bạn (chẳng hạn như android.util.Log cần thiết để sử dụng Log), hãy chọn **File > Settings** trong Windows hoặc **Android Studio > Preferences** trong macOS.



3. Chọn **Editor > General >Auto Import**. Chọn tất cả các hộp và **đặt Insert imports on paste** thành **Always**.



4. Nhấp vào **Apply** rồi nhấp vào **OK**.

5. Trong phương thức `onCreate()` của `MainActivity`, hãy thêm câu lệnh sau:

`Log.d("MainActivity", "Hello World");`

Phương thức `onCreate()` bây giờ sẽ trông giống như đoạn mã sau:

```
package com.example.helloworld;
XML file
import android.os.Bundle;
import android.util.Log;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        Log.d( tag: "MainActivity", msg: "Hello World");
    }
}
```

6. Nếu cửa sổ Logcat chưa mở, hãy nhấp vào thẻ **Logcat** ở cuối Android Studio để mở nó.
7. Kiểm tra xem tên mục tiêu và tên gói của ứng dụng có đúng không.
8. Thay đổi mức Nhật ký trong cửa sổ **Logcat** thành **Debug** (hoặc để nguyên là **Verbose** vì có quá ít thông báo nhật ký).
9. Chạy ứng dụng của bạn.

Thông báo sẽ xuất hiện ở cửa sổ Logcat:



Thử thách mã hóa

Lưu ý: Tất cả các thử thách mã hóa đều là tùy chọn và không phải là điều kiện tiên quyết cho các bài học sau.

Thử thách: Bây giờ bạn đã thiết lập và quen thuộc với quy trình phát triển cơ bản, hãy thực hiện các bước sau:

1. Tạo một dự án mới trong Android Studio.
2. Đổi lời chào "Hello World" thành "Happy Birthday to " và tên của một người có sinh nhật gần đây.

3. (Tùy chọn) Chụp ảnh màn hình ứng dụng đã hoàn thành của bạn và gửi email cho một người có sinh nhật mà bạn đã quên.
4. Một cách sử dụng phổ biến của lớp Log là ghi lại các ngoại lệ Java khi chúng xảy ra trong chương trình của bạn. Có một số phương pháp hữu ích, chẳng hạn như Log.e(), mà bạn có thể sử dụng cho mục đích này. Khám phá phương pháp bạn có thể sử dụng để bao gồm ngoại lệ với thông báo Nhật ký. Sau đó, viết mã trong ứng dụng của bạn để kích hoạt và ghi nhật ký ngoại lệ.

Tóm tắt

- Để cài đặt Android Studio, truy cập [Android Studio](#) và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo ứng dụng mới, đảm bảo đặt **API 24: Android 7.0 Nougat** làm Minimum SDK.
- Để xem cấu trúc Android của ứng dụng trong **Project pane**, nhấp vào tab **Project** trong cột tab dọc, sau đó chọn **Android** trong menu bật lên ở trên cùng.
- Chỉnh sửa tệp **build.gradle(Module:app)** khi cần thêm thư viện mới hoặc thay đổi phiên bản thư viện.
- Tất cả mã nguồn và tài nguyên của ứng dụng nằm trong thư mục **app** và **res**. Thư mục java chứa các hoạt động, kiểm thử, và các thành phần mã Java khác. Thư mục res chứa tài nguyên như bố cục giao diện, chuỗi văn bản, hình ảnh.
- Chỉnh sửa tệp **AndroidManifest.xml** để thêm các thành phần và quyền cho ứng dụng Android.
- Sử dụng [Android Virtual Device \(AVD\) Manager](#) để tạo thiết bị ảo (emulator) chạy ứng dụng.
- Thêm câu lệnh [Log](#) vào ứng dụng để hiển thị thông điệp trong **Logcat pane** nhằm hỗ trợ gỡ lỗi.
- Để chạy ứng dụng trên thiết bị Android thật, bật **USB Debugging** trên thiết bị. Mở **Settings > About phone**, nhấn vào **Build number** bảy lần. Quay lại **Settings** màn hình, vào **Developer options**, bật **USB Debugging**.

Các khái niệm liên quan

Tài liệu về khái niệm liên quan có trong phiên bản 1.0: Giới thiệu về Android và phiên bản 1.1 Ứng dụng Android đầu tiên của bạn.

Tìm hiểu thêm

- Tài liệu Android Studio:
- Trang tải xuống Android Studio
- Ghi chú phát hành Android Studio
- Làm quen với Android Studio
- Công cụ dòng lệnh Logcat
- Trình quản lý Thiết bị ảo Android (AVD)
- Tổng quan về App Manifest
- Cấu hình bản dựng của bạn
- Lớp nhặt ký
- Tạo và quản lý Thiết bị ảo

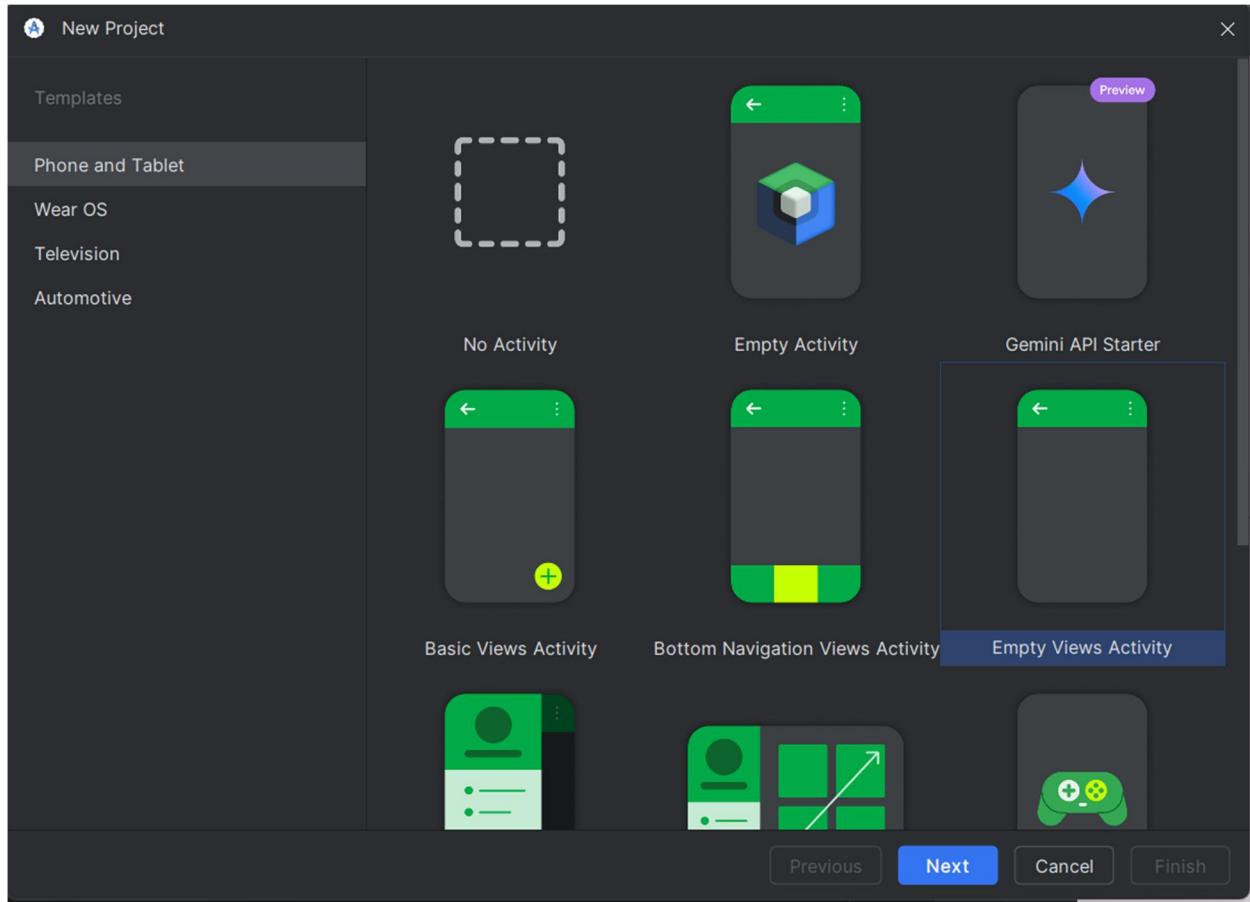
Khác:

- Làm thế nào để cài đặt Java?
- Cài đặt phần mềm JDK và thiết lập JAVA_HOME
- Trang Gradle
- Cú pháp Apache Groovy
- Trang Wikipedia Gradle

Bài tập

Xây dựng và chạy ứng dụng

- Tạo một dự án Android mới từ mẫu.



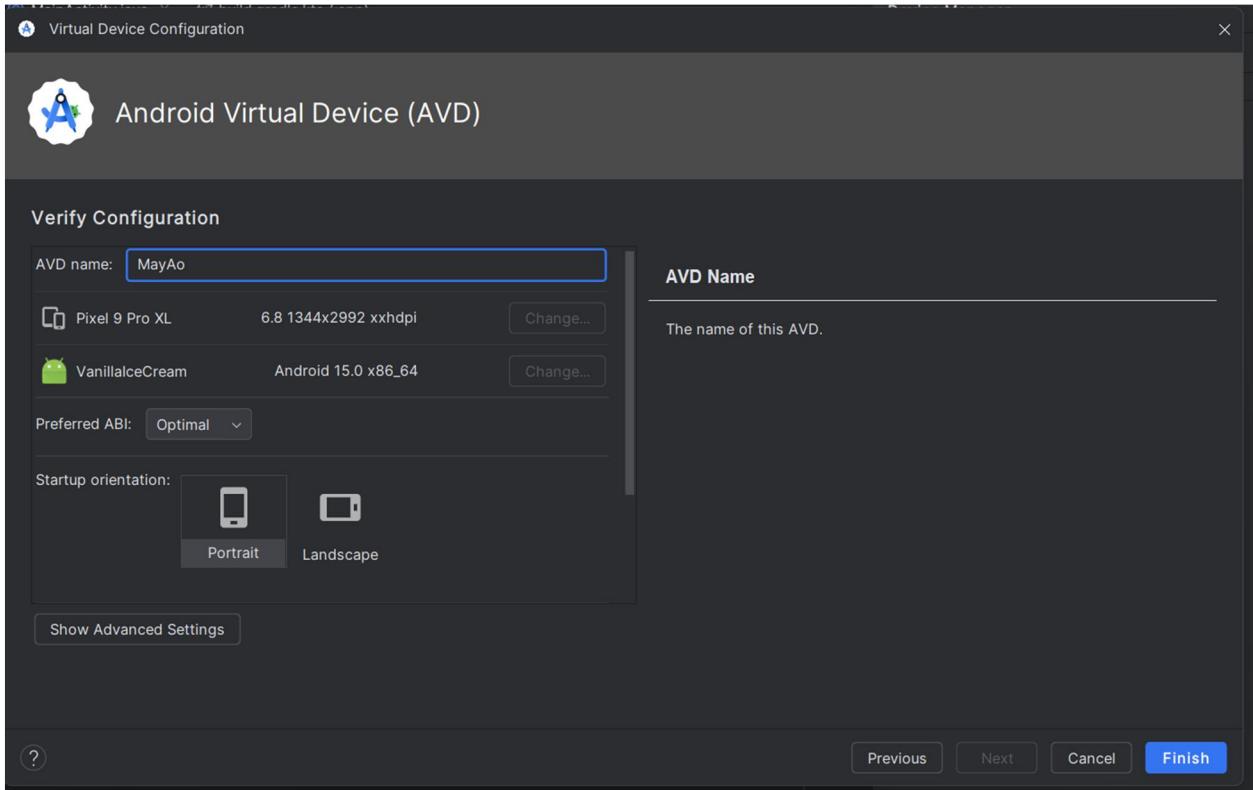
- Thêm các câu lệnh ghi nhật ký cho nhiều cấp độ nhật ký khác nhau trong onCreate() trong hoạt động chính.

```

</> activity_main.xml  MainActivity.java  build.gradle.kts (app)
 9 import androidx.core.view.ViewCompat;
10 import androidx.core.view.WindowInsetsCompat;
11
12 <> public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         EdgeToEdge.enable( $this$enableEdgeToEdge: this );
18         setContentView(R.layout.activity_main);
19         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
20             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
21             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
22             return insets;
23         });
24         Log.d( tag: "MainActivity", msg: "Hello World" );
25         Log.v( tag: "MainActivity", msg: "This is a VERBOSE log message." );
26         Log.i( tag: "MainActivity", msg: "This is an INFO log message." );
27         Log.w( tag: "MainActivity", msg: "This is a WARNING log message." );
28         Log.e( tag: "MainActivity", msg: "This is an ERROR log message." );
29         try {
30             int result = 10 / 0;
31         } catch (Exception e) {
32             Log.e( tag: "MainActivity", msg: "Exception occurred: ", e );
33         }
34     }
35 }

```

- Tạo trình giả lập cho thiết bị, nhắm mục tiêu đến bất kỳ phiên bản Android nào bạn thích và chạy ứng dụng.



- Sử dụng tính năng lọc trong **Logcat** để tìm các câu lệnh nhật ký của bạn và điều chỉnh các cấp độ để chỉ hiển thị các câu lệnh ghi nhật ký gỡ lỗi hoặc lỗi.

2025-02-25 10:51:45.843 7700-7700 mple.helloworld	com.example.helloworld		W Accessing hidden method Landroid/view/ViewGroup;.>makeOptionalFitsSystemWindow
2025-02-25 10:51:45.948 7700-7700 MainActivity	com.example.helloworld		D Hello World
2025-02-25 10:51:45.948 7700-7700 MainActivity	com.example.helloworld		V This is a VERBOSE log message.
2025-02-25 10:51:45.948 7700-7700 MainActivity	com.example.helloworld		I This is an INFO log message.
2025-02-25 10:51:45.949 7700-7700 MainActivity	com.example.helloworld		W This is a WARNING log message.
2025-02-25 10:51:45.949 7700-7700 MainActivity	com.example.helloworld		E This is an ERROR log message.
2025-02-25 10:51:45.951 7700-7700 MainActivity	com.example.helloworld		E Exception occurred:
			java.lang.ArithmetricException: divide by zero
			at com.example.helloworld.MainActivity.onCreate(MainActivity.java:30)
			at android.app.Activity.performCreate(Activity.java:9002)
			at android.app.Activity.performCreate(Activity.java:9880)
			at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1207)
			at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:1004)
			at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:1072)
			at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivi

Trả lời các câu hỏi sau

Câu hỏi 1: Tên của tệp bố cục cho hoạt động chính là gì?

- MainActivity.java
 - AndroidManifest.xml
 - activity_main.xml
 - build.gradle

Câu hỏi 2:Tên của tài nguyên chuỗi chỉ định tên ứng dụng là gì?

- app_name
- xmlns:app
- android:name
- applicationId

Câu hỏi 3: Bạn sử dụng công cụ nào để tạo trình giả lập mới?

- Android Device Monitor
- **AVD Manager**
- SDK Manager
- Theme Editor

Câu hỏi 4: Giả sử ứng dụng của bạn bao gồm câu lệnh ghi nhật ký này:

```
Log.i("MainActivity", "MainActivity layout is complete");
```

Bạn thấy câu lệnh "MainActivity layout is complete" trong ngăn Logcat nếu menu Log level được đặt thành tùy chọn nào sau đây? (Gợi ý: nhiều câu trả lời là được.)

- **Verbose (Chi tiết)**
- **Debug (Gỡ lỗi)**
- **Info (Thông tin)**
- Warn (Cảnh báo)
- Error (Lỗi)
- Assert (Khẳng định)

Nộp ứng dụng của bạn để chấm điểm

Kiểm tra để đảm bảo ứng dụng có những thông tin sau:

- Hoạt động hiển thị "Hello World" trên màn hình.
- Câu lệnh log trong onCreate() trong hoạt động chính.
- Mức log trong ngăn Logcat chỉ hiển thị câu lệnh ghi nhật ký gỡ lỗi hoặc lỗi.

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của một thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là *views* — mỗi phần tử trên màn hình đều là một View. Lớp View đại diện cho khối xây dựng cơ bản của tất cả các thành phần UI và là lớp cơ sở cho các lớp cung cấp các thành phần UI tương tác, chẳng hạn như nút bấm (Button), hộp kiểm (Checkbox), và trường nhập văn bản (Text Entry Field). Các lớp con của View được sử dụng phổ biến và sẽ được mô tả qua nhiều bài học bao gồm:

- [TextView](#) để hiển thị văn bản.
- [EditText](#) để cho phép người dùng nhập và chỉnh sửa văn bản.
- [Button](#) và các thành phần có thể nhấp khác (như [RadioButton](#), [CheckBox](#) và [Spinner](#)) để cung cấp hành vi tương tác.
- [ScrollView](#) và [RecyclerView](#) để hiển thị các mục có thể cuộn.
- [ImageView](#) để hiển thị hình ảnh.
- [ConstraintLayout](#) và [LinearLayout](#) để chứa các thành phần View khác và định vị chúng.

Mã Java hiển thị và điều khiển UI được chứa trong một lớp mở rộng Activity. Activity thường được liên kết với bố cục của các chế độ xem UI được xác định là tệp XML (Ngôn ngữ đánh dấu mở rộng). Tệp XML này thường được đặt tên theo Activity của nó và xác định bố cục của các thành phần View trên màn hình.

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị bố cục được xác định trong tệp bố cục activity_main.xml, bao gồm TextView có văn bản "Hello World".

Trong các ứng dụng phức tạp hơn, một Hoạt động có thể triển khai các hành động để phản hồi các lần chạm của người dùng, vẽ nội dung đồ họa hoặc yêu cầu dữ liệu từ cơ sở dữ liệu hoặc internet. Bạn sẽ tìm hiểu thêm về lớp Hoạt động trong một bài học khác.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên của mình-một ứng dụng cho phép người dùng tương tác. Bạn sẽ tạo một ứng dụng bằng mẫu Hoạt động trống. Bạn cũng sẽ học cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục và cách chỉnh sửa bố cục trong XML. Bạn cần phát triển các kỹ năng này để có thể hoàn thành các bài thực hành khác trong khóa học này.

Những gì bạn nên biết

Bạn nên có khả năng:

- Cách cài đặt và mở Android Studio.
- Cách tạo ứng dụng HelloWorld.
- Cách chạy ứng dụng HelloWorld.

Những thứ bạn sẽ tìm hiểu

- Cách tạo ứng dụng có hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục.
- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới. Hãy xem qua từ vựng và khái niệm trong bảng chú giải thuật ngữ để có các định nghĩa thân thiện

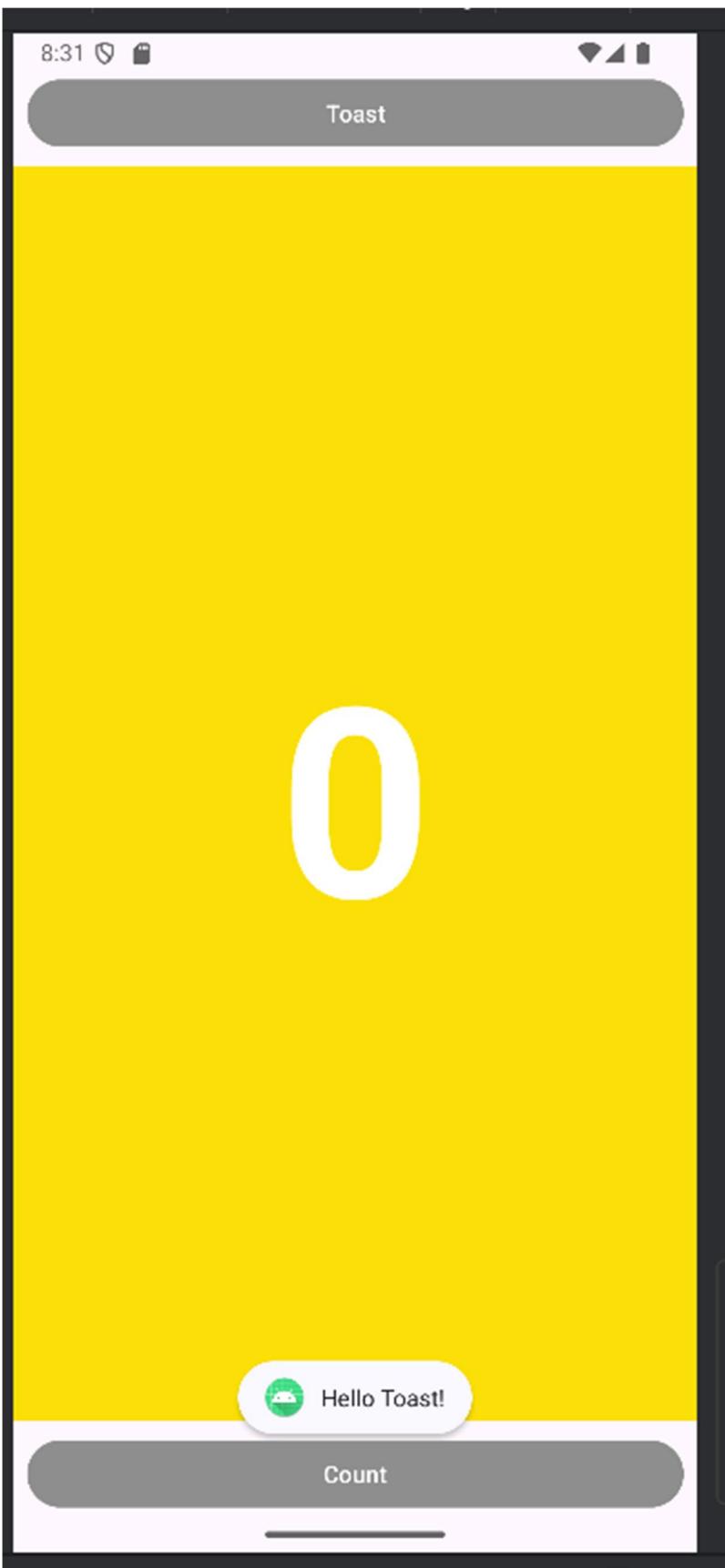
Những thứ bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử Button và một TextView vào bố cục.
- Thao tác từng phần tử trong [ConstraintLayout](#) để giới hạn chúng vào lề và các phần tử khác.
- Thay đổi các thuộc tính phần tử UI.
- Chỉnh sửa bố cục của ứng dụng trong XML.
- Trích xuất các chuỗi được mã hóa cứng thành các tài nguyên chuỗi.
- Triển khai các phương thức xử lý nhấp để hiển thị thông báo trên màn hình khi người dùng chạm vào mỗi Button.

Tổng quan về ứng dụng

Ứng dụng HelloToast bao gồm hai Button và một TextView. Khi người dùng nhấn vào Nút đầu tiên nó hiển thị một thông báo ngắn (**Toast**) trên màn hình. Nhấn nút thứ hai Tăng giá trị bộ đếm số lần nhấn và hiển thị trong TextView, bắt đầu từ 0.

Dưới đây là giao diện của ứng dụng đã hoàn thiện:



Nhiệm vụ 1: Tạo và khám phá một dự án mới

Trong bài thực hành này, bạn thiết kế và triển khai một dự án cho ứng dụng HelloToast. Một liên kết đến mã giải pháp được cung cấp ở cuối.

1.1 Tạo dự án Android Studio

1. Khởi động Android Studio và tạo một dự án mới với các tham số sau:

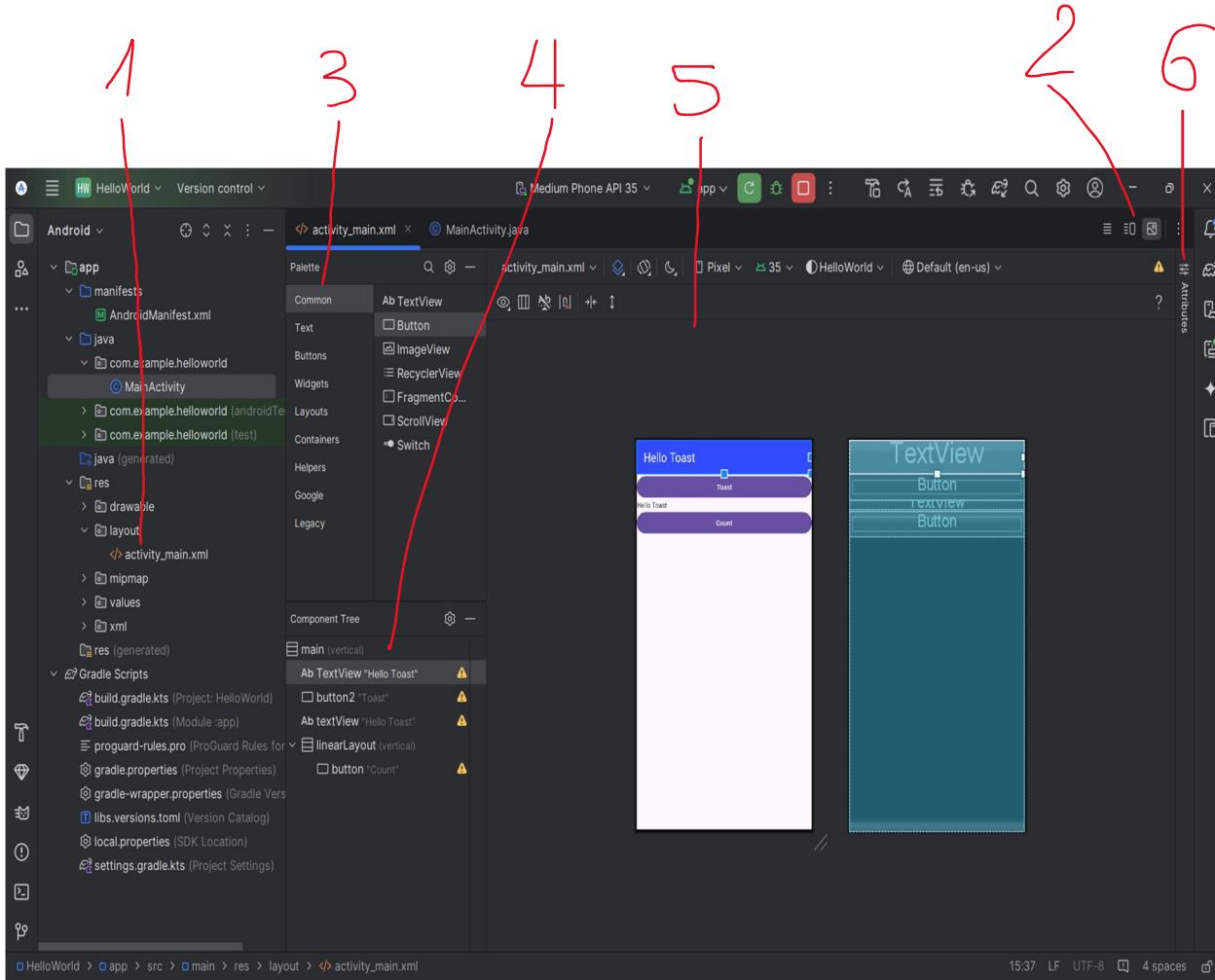
Giá trị thuộc tính	Giá trị
Tên ứng dụng Hello Toast	Hello Toast
Tên công ty	com.example.android (hoặc tên miền của riêng bạn)
Điện thoại và Máy tính bảng SDK tối thiểu	API 24: Android 7.0 Nougat
Mẫu	Empty View Activity
Hộp Tạo tệp Bố cục	Đã chọn
Hộp Tương Thích Ngược	Đã chọn

2. Chọn **Run > Run app** hoặc nhấp vào **biểu tượng Run** trên thanh công cụ để biên dịch và chạy ứng dụng trên trình giả lập hoặc thiết bị của bạn.

1.2 Khám phá trình chỉnh sửa bố cục (Layout Editor)

Android Studio cung cấp trình chỉnh sửa bố cục để nhanh chóng xây dựng bố cục của các thành phần giao diện người dùng (UI) của ứng dụng. Nó cho phép bạn kéo các thành phần vào chế độ xem thiết kế trực quan và bản thiết kế, định vị chúng trong bố cục, thêm ràng buộc và đặt thuộc tính. Ràng buộc xác định vị trí của thành phần UI trong bố cục. Ràng buộc thể hiện kết nối hoặc căn chỉnh với chế độ xem khác, bố cục cha hoặc hướng dẫn vô hình.

Khám phá trình chỉnh sửa bố cục và tham khảo hình bên dưới khi bạn làm theo các bước được đánh số:



1. Trong thư mục **app > res > layout** trong ngăn **Project > Android**, hãy nhấp đúp vào tệp **activity_main.xml** để mở tệp đó, nếu tệp đó chưa được mở.
2. Nhấp vào tab **Design** nếu tệp đó chưa được chọn. Bạn sử dụng tab **Design** để thao tác các phần tử và bố cục, và thẻ **Text** để chỉnh sửa mã XML cho bố cục.
3. Cửa sổ **Palettes** hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục của ứng dụng.
4. Ngăn **Component tree** hiển thị hệ thống phân cấp chế độ xem của các phần tử UI. Các phần tử View được tổ chức thành hệ thống phân cấp cây gồm các phần tử cha và con, trong đó phần tử con kế thừa các thuộc tính của phần tử cha. Trong hình trên, **TextView** là phần tử con của **ConstraintLayout**. Bạn sẽ tìm hiểu về các phần tử này sau trong bài học này.
5. Ngăn thiết kế và bản thiết kế của trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình trên, bố cục chỉ hiển thị một phần tử: **TextView** hiển thị "Hello World".
6. Thẻ **Attributes** hiển thị ngăn **Attributes** để thiết lập thuộc tính cho phần tử UI.

Mẹo: Xem Xây dựng giao diện người dùng bằng Trình chỉnh sửa bố cục để biết chi tiết về cách sử dụng trình chỉnh sửa bố cục và [Meet Android Studio](#) để biết tài liệu đầy đủ về Android Studio.

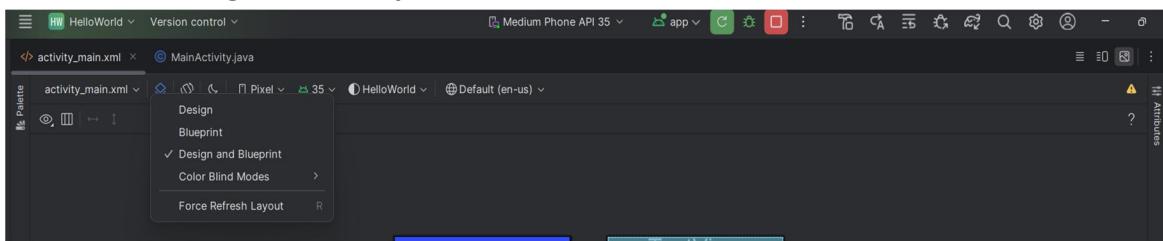
Nhiệm vụ 2: Thêm các phần tử View trong Layout Editor

Trong phần này, bạn sẽ tạo bố cục UI cho ứng dụng HelloToast bằng cách sử dụng [ConstraintLayout](#). Bạn có thể thiết lập ràng buộc thủ công hoặc tự động bằng công cụ [Autoconnect](#).

2.1 Kiểm tra ràng buộc của phần tử UI

Làm theo các bước sau:

1. Mở `activity_main.xml` từ ngăn Project > Android nếu nó chưa mở. Nếu thẻ **Design** chưa được chọn, hãy nhấp vào tab đó.
Nếu không có bản thiết kế, hãy nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design and Blueprint**.

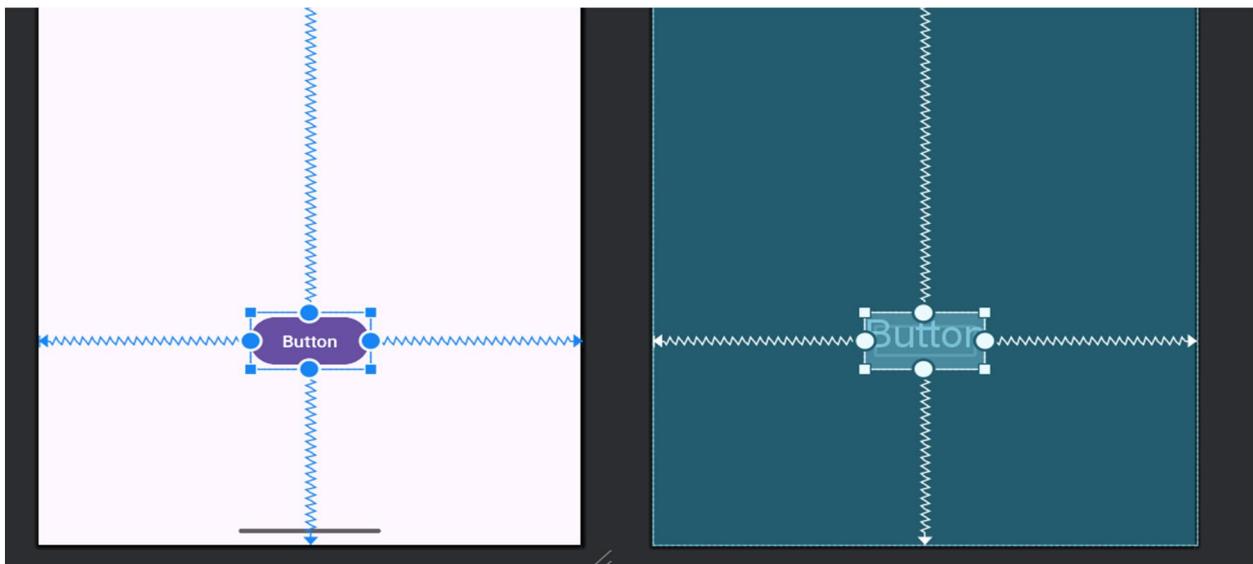


2. Công cụ **Autoconnect**  cũng nằm trên thanh công cụ. Theo mặc định, công cụ này được bật. Đối với bước này, hãy đảm bảo rằng công cụ không bị tắt.
3. Nhấp vào nút phóng to để phóng to các ngăn thiết kế và bản thiết kế để xem cận cảnh.



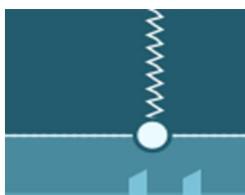
4. Chọn **TextView** trong ngăn Component Tree. TextView "Hello World" được tô sáng trong các ngăn thiết kế và bản thiết kế và các ràng buộc cho phần tử có thể nhìn thấy được.

5. Tham khảo hình ảnh động bên dưới để biết bước này. Nhấp vào tay cầm hình tròn ở bên phải của TextView để xóa ràng buộc theo chiều ngang liên kết chế độ xem với bên phải của bố cục. TextView nhảy sang bên trái vì nó không còn bị ràng buộc ở bên phải nữa. Để thêm lại ràng buộc theo chiều ngang, hãy nhấp vào tay cầm tương tự và kéo một đường sang bên phải của bố cục.



Trong bản thiết kế hoặc ngăn thiết kế, các nút điều khiển sau xuất hiện trên phần tử TextView:

- **Constraint handle:** Để tạo một ràng buộc như trong hình động ở trên, hãy nhấp vào một constraint handle, được hiển thị dưới dạng một vòng tròn ở bên cạnh một phần tử. Sau đó, kéo handle đến một constraints khác hoặc đến ranh giới parent. Một đường ngoằn ngoèo biểu thị cho constraints.



- **Resizing handle:** Để thay đổi kích thước phần tử, hãy kéo các nút điều chỉnh thay đổi kích thước hình vuông. Nút điều chỉnh sẽ thay đổi thành một góc nghiêng khi bạn kéo nó.

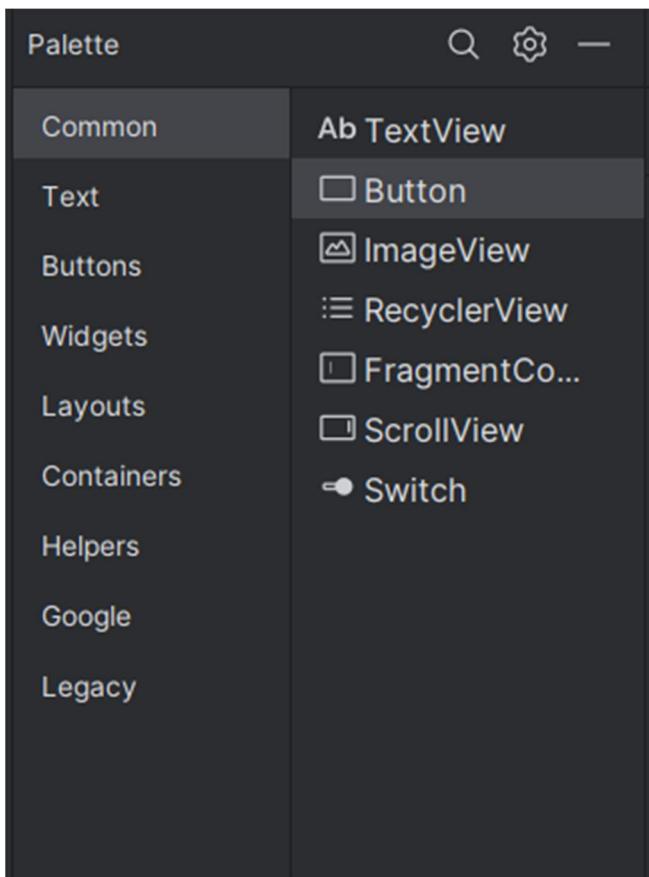


2.2 Thêm 1 nút vào bố cục

Khi được bật, công cụ **Autoconnect** sẽ tự động tạo hai hoặc nhiều ràng buộc cho một thành phần UI vào bố cục cha. Sau khi bạn kéo thành phần vào bố cục, công cụ này sẽ tạo ra các ràng buộc dựa trên vị trí của thành phần.

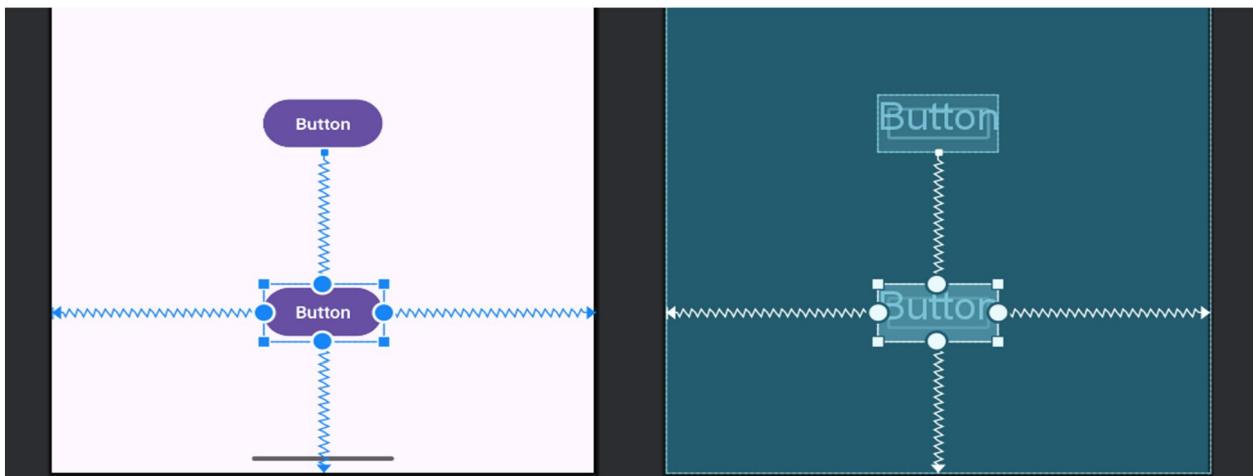
Thực hiện theo các bước sau để thêm Nút:

1. Bắt đầu với một bảng trắng. Thành phần TextView không cần thiết, vì vậy khi thành phần này vẫn được chọn, hãy nhấn phím **Delete** hoặc chọn **Edit > Delete**. Bây giờ bạn có một trang bố cục hoàn toàn trống.
2. Kéo **Button** từ **Palette** đến bất kỳ vị trí nào trong bố cục. Nếu bạn thả Nút vào khu vực giữa trên cùng của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc lên trên cùng, bên trái và bên phải của bố cục như minh họa trong hình động bên dưới.



2.3 Thêm nút thứ hai vào bố cục

1. Kéo **Button** khác từ ngăn **Palette** màu vào giữa bố cục như thể hiện trong hình động bên dưới. Autoconnect có thể cung cấp các ràng buộc theo chiều ngang cho bạn (nếu không, bạn có thể tự kéo chúng).
2. Kéo một ràng buộc theo chiều dọc xuống dưới cùng của bố cục (tham khảo hình bên dưới).



Bạn có thể xóa ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di chuột qua

Clear Constraints of Selection

để hiển thị nút **Clear Constraints**. Nhấp vào nút này để xóa **tất cả** ràng buộc trên phần tử đã chọn.

Để xóa một ràng buộc cụ thể, hãy nhấp vào nút tròn tương ứng của ràng buộc đó.

Để xóa tất cả ràng buộc trong toàn bộ bố cục, nhấp vào công cụ **Clear All Constraints** trên thanh công cụ. Công cụ này hữu ích nếu bạn muốn thiết lập lại toàn bộ ràng buộc trong bố cục của mình.

Nhiệm vụ 3: Thay đổi thuộc tính phần tử UI

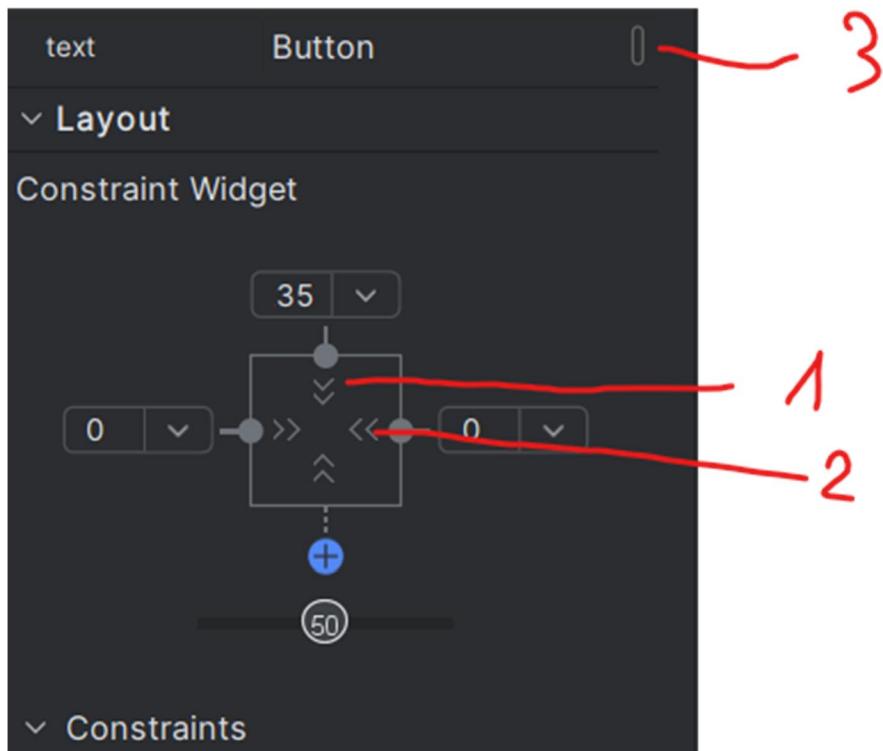
Ngăn Attributes cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho phần tử UI. Bạn có thể tìm thấy các thuộc tính (được gọi là thuộc tính) chung cho tất cả các chế độ xem trong tài liệu lớp View.

Trong nhiệm vụ này, bạn nhập các giá trị mới và thay đổi các giá trị cho các thuộc tính Button quan trọng, có thể áp dụng cho hầu hết các loại View.

3.1 Thay đổi kích thước Nút

Trình chỉnh sửa bố cục cung cấp các nút điều chỉnh thay đổi kích thước ở bốn góc của một View, cho phép bạn thay đổi kích thước nhanh chóng. Bạn có thể kéo các nút điều chỉnh ở mỗi góc của View để điều chỉnh kích thước, nhưng điều này sẽ gán kích thước cố định cho chiều rộng và chiều cao. Tránh đặt kích thước cố định cho hầu hết các phần tử View, vì các kích thước này không thể tự động điều chỉnh theo nội dung hoặc kích thước màn hình khác nhau.

Thay vào đó, sử dụng bảng **Attributes** pane ở bên phải của Layout Editor để chọn chế độ kích thước không dùng giá trị cố định. Bảng **Attributes** có một bảng định cỡ hình vuông ở trên cùng, trong đó các biểu tượng bên trong đại diện cho các cài đặt chiều cao và chiều rộng như sau:



Trong hình trên:

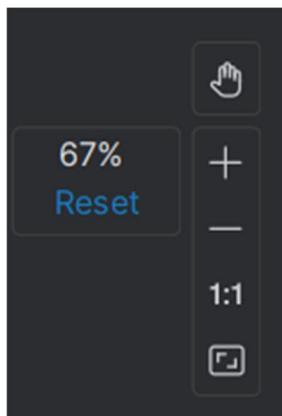
1. **Kiểm soát chiều cao.** Kiểm soát này chỉ định thuộc tính `layout_height` và xuất hiện trong hai phân đoạn ở phía trên và phía dưới của hình vuông. Các góc cho biết rằng kiểm soát này được đặt thành `wrap_content`, nghĩa là View sẽ mở rộng theo chiều dọc khi cần để vừa với nội dung của nó. "35" cho biết lề chuẩn được đặt thành 35dp.
2. **Kiểm soát chiều rộng.** Kiểm soát này chỉ định `layout_width` và xuất hiện trong hai phân đoạn ở bên trái và bên phải của hình vuông. Các góc cho biết rằng kiểm soát

này được đặt thành wrap_content, nghĩa là View sẽ mở rộng theo chiều ngang khi cần để vừa với nội dung của nó, tối đa là lề 0dp.

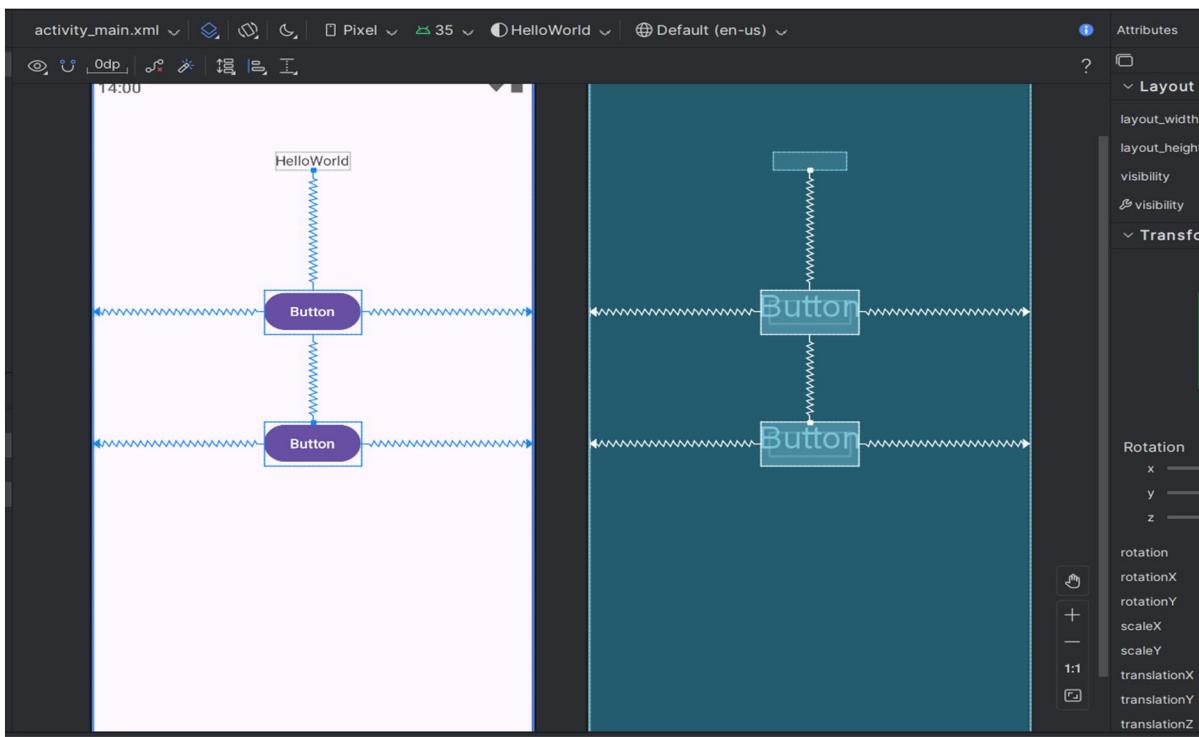
3. **Nút đóng ngăn Thuộc tính.** Nhấp để đóng ngăn.

Làm theo các bước sau:

1. Chọn Nút trên cùng trong ngăn Component Tree.
2. Nhấp vào thẻ Attributes ở phía bên phải của cửa sổ trình chỉnh sửa bố cục.

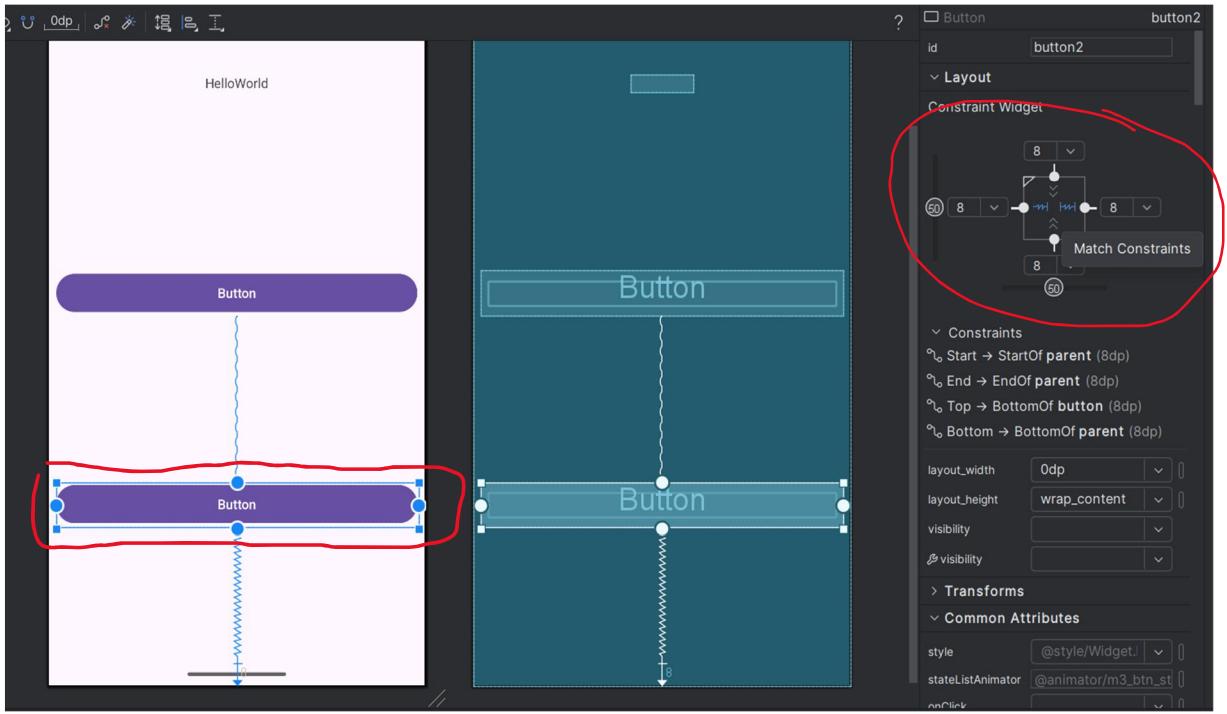


3. Nhấp vào điều khiển chiều rộng hai lần - lần nhấp đầu tiên sẽ thay đổi thành **Fixed** với các đường thẳng và lần nhấp thứ hai sẽ thay đổi thành Match Constraints với các ràng buộc với các thanh cuộn lò xo, như được hiển thị trong hình động bên dưới.



Kết quả của việc thay đổi điều khiển chiều rộng, thuộc tính `layout_width` trong ngăn **Attributes** hiển thị giá trị `match_constraint` và phần tử Button kéo dài theo chiều ngang để lấp đầy khoảng trống giữa bên trái và bên phải của bố cục.

4. Chọn Nút thứ hai và thực hiện các thay đổi tương tự đối với `layout_width` như trong bước trước, như thể hiện trong hình bên dưới.



Như đã trình bày trong các bước trước, các thuộc tính `layout_width` và `layout_height` trong ngăn Attributes thay đổi khi bạn thay đổi các điều khiển chiều cao và chiều rộng trong trình kiểm tra. Các thuộc tính này có thể lấy một trong ba giá trị cho bối cục, đó là `ConstraintLayout`:

- Thiết lập `match_constraint` mở rộng phần tử View để lấp đầy phần tử cha của nó theo chiều rộng hoặc chiều cao -lên đến một lề, nếu có. Phần tử cha trong trường hợp này là `ConstraintLayout`. Bạn tìm hiểu thêm về `ConstraintLayout` trong các nhiệm vụ tiếp theo.
- Thiết lập `wrap_content` thu nhỏ kích thước của phần tử View để nó chỉ đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, phần tử View sẽ trở nên vô hình.
- Để chỉ định kích thước cố định điều chỉnh theo kích thước màn hình của thiết bị, hãy sử dụng số lượng cố định pixel không phụ thuộc vào mật độ (đơn vị dp). Ví dụ: 16dp nghĩa là 16 pixel không phụ thuộc vào mật độ

Mẹo: Nếu bạn thay đổi thuộc tính `layout_width` bằng menu bật lên của nó, thuộc tính `layout_width` sẽ được đặt thành 0 vì không có kích thước nào được đặt. Thiết lập này giống như `match_constraint` - chế độ xem có thể mở rộng tối đa có thể để đáp ứng các ràng buộc và thiết lập lề.

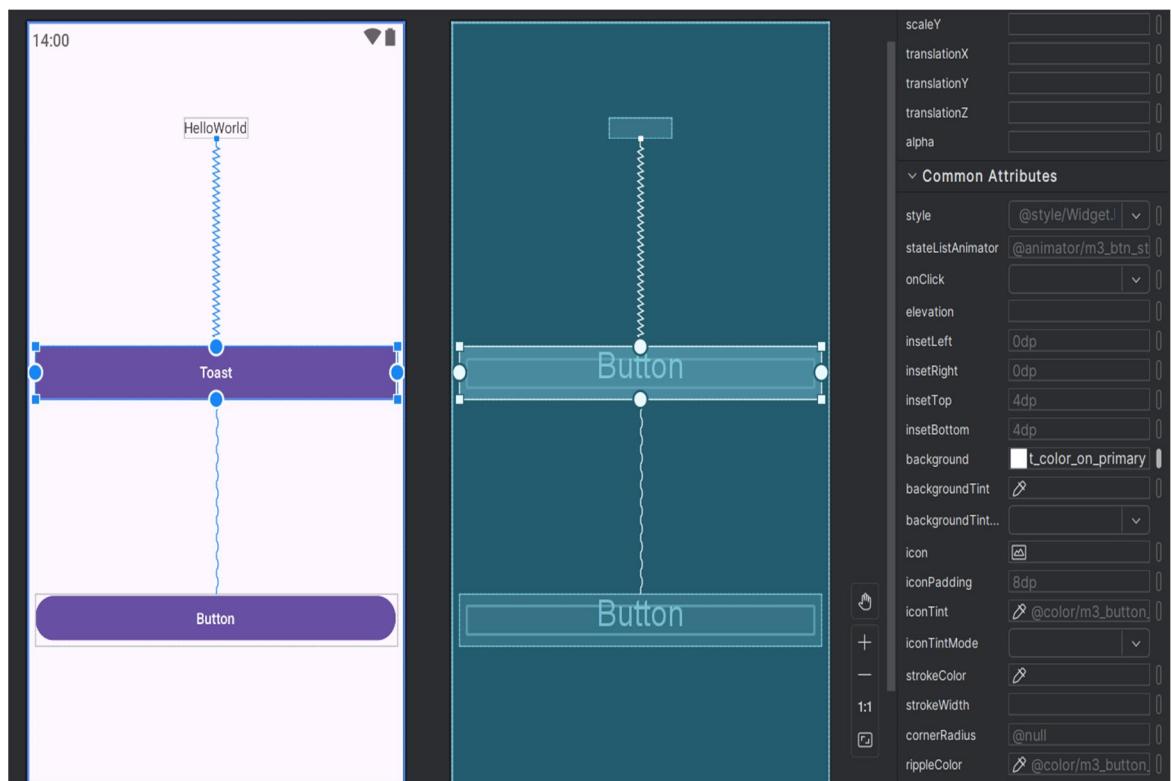
3.2. Thay đổi thuộc tính của nút

Để xác định duy nhất từng View trong bố cục Activity, mỗi View hoặc lớp con View (chẳng hạn như Button) cần một ID duy nhất. Và để có thể sử dụng, các phần tử Button cần có văn bản. Các phần tử View cũng có thể có nền có thể là màu hoặc hình ảnh.

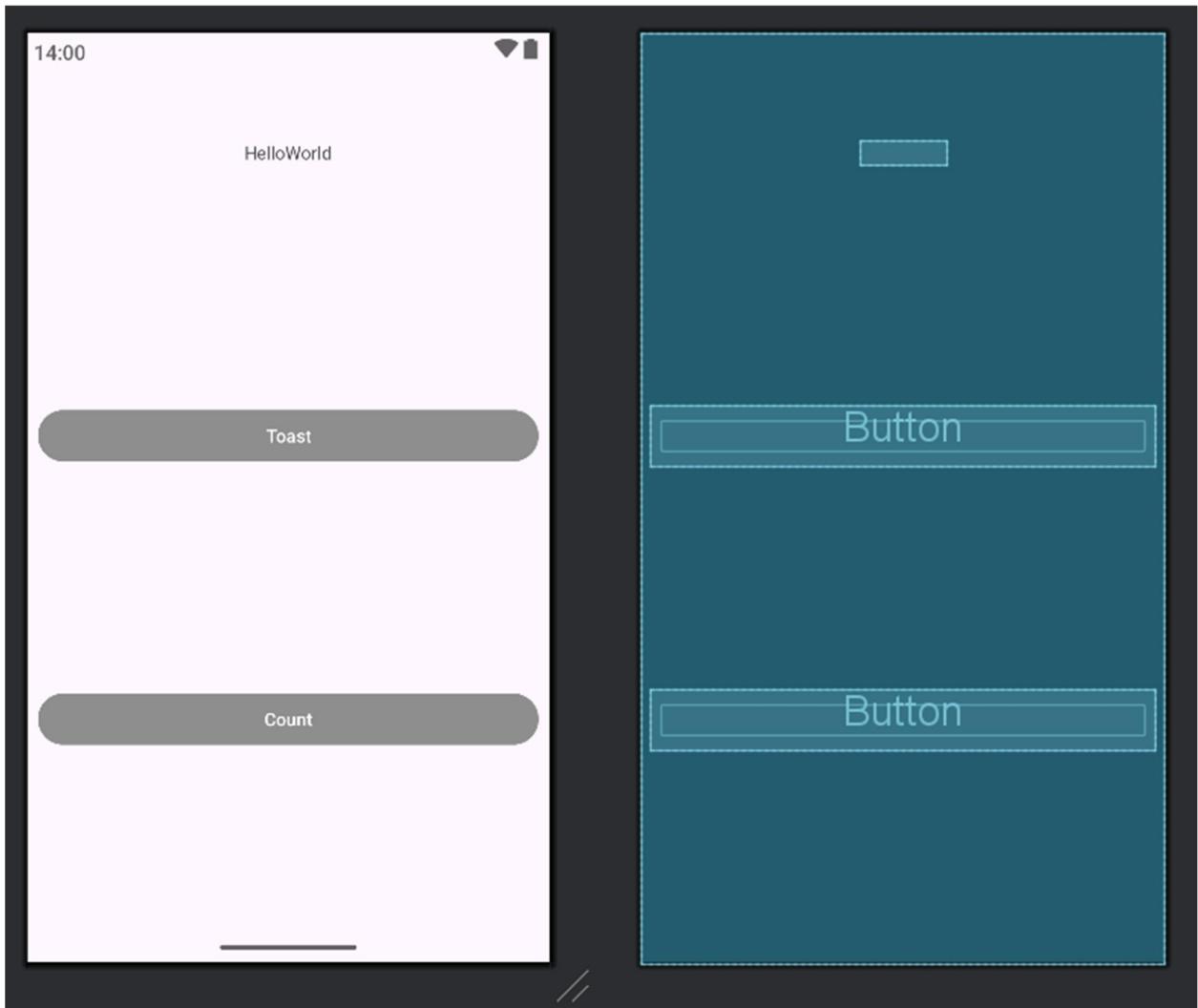
Ngăn Attributes cung cấp quyền truy cập vào tất cả các thuộc tính mà bạn có thể gán cho một phần tử View. Bạn có thể nhập giá trị cho từng thuộc tính, chẳng hạn như các thuộc tính android:id, background, textColor và text

Hoạt động sau đây minh họa cách thực hiện các bước này:

1. Sau khi chọn Button đầu tiên, hãy chỉnh sửa trường ID ở đầu ngăn Attributes thành **button_toast** cho thuộc tính android:id, được sử dụng để xác định phần tử trong bố cục.
2. Đặt thuộc tính background thành **@color/design_default_color_on_primary**. (Khi bạn nhập **@c**, các lựa chọn sẽ xuất hiện để dễ dàng lựa chọn.)
3. Đặt thuộc tính textColor thành **@android:color/white**.
4. Chỉnh sửa thuộc tính văn bản thành **Toast**.



- Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng **button_count** làm ID, **Count** cho thuộc tính text và cùng màu cho nền và text như các bước trước.



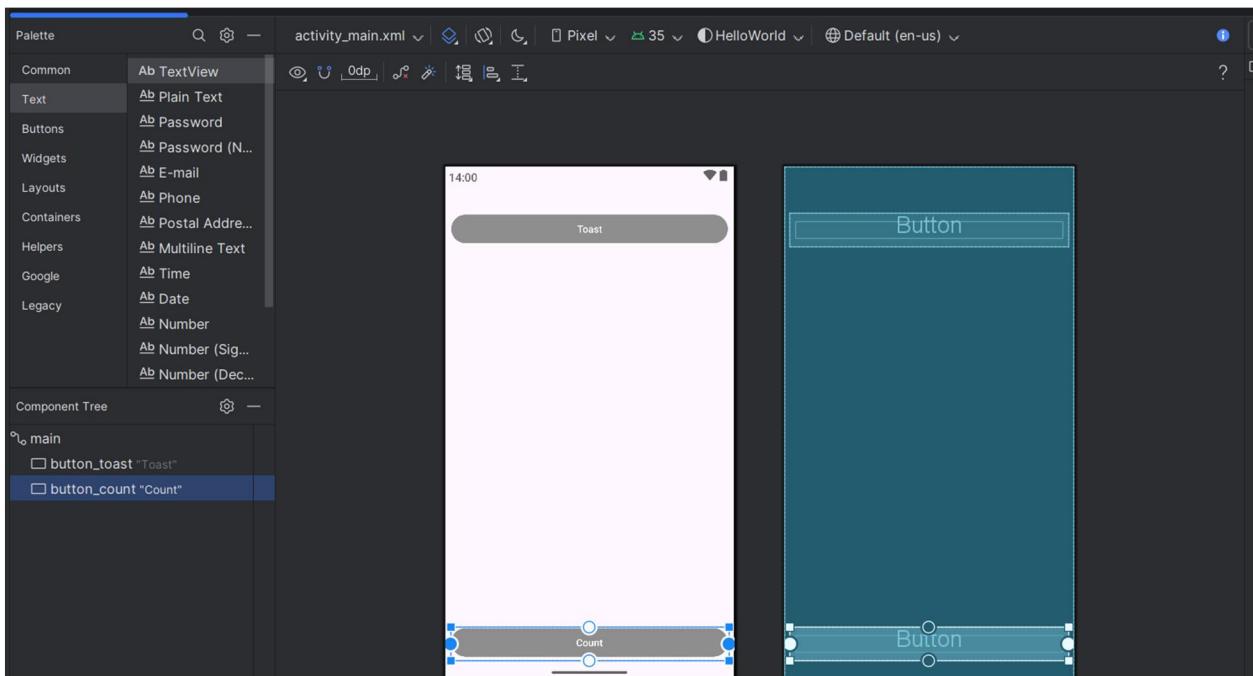
Màu *ColorPrimary* là màu chủ đạo của giao diện ứng dụng, một trong những màu cơ sở chủ đạo được xác định trước được định nghĩa trong tệp tài nguyên colors.xml. Nó được sử dụng cho thanh ứng dụng. Sử dụng màu cơ sở cho các thành phần UI khác sẽ tạo ra một UI thống nhất. Bạn sẽ tìm hiểu thêm về chủ đề ứng dụng và Material Design trong một bài học khác

Nhiệm vụ 4: Thêm TextEdit và thiết lập các thuộc tính của nó

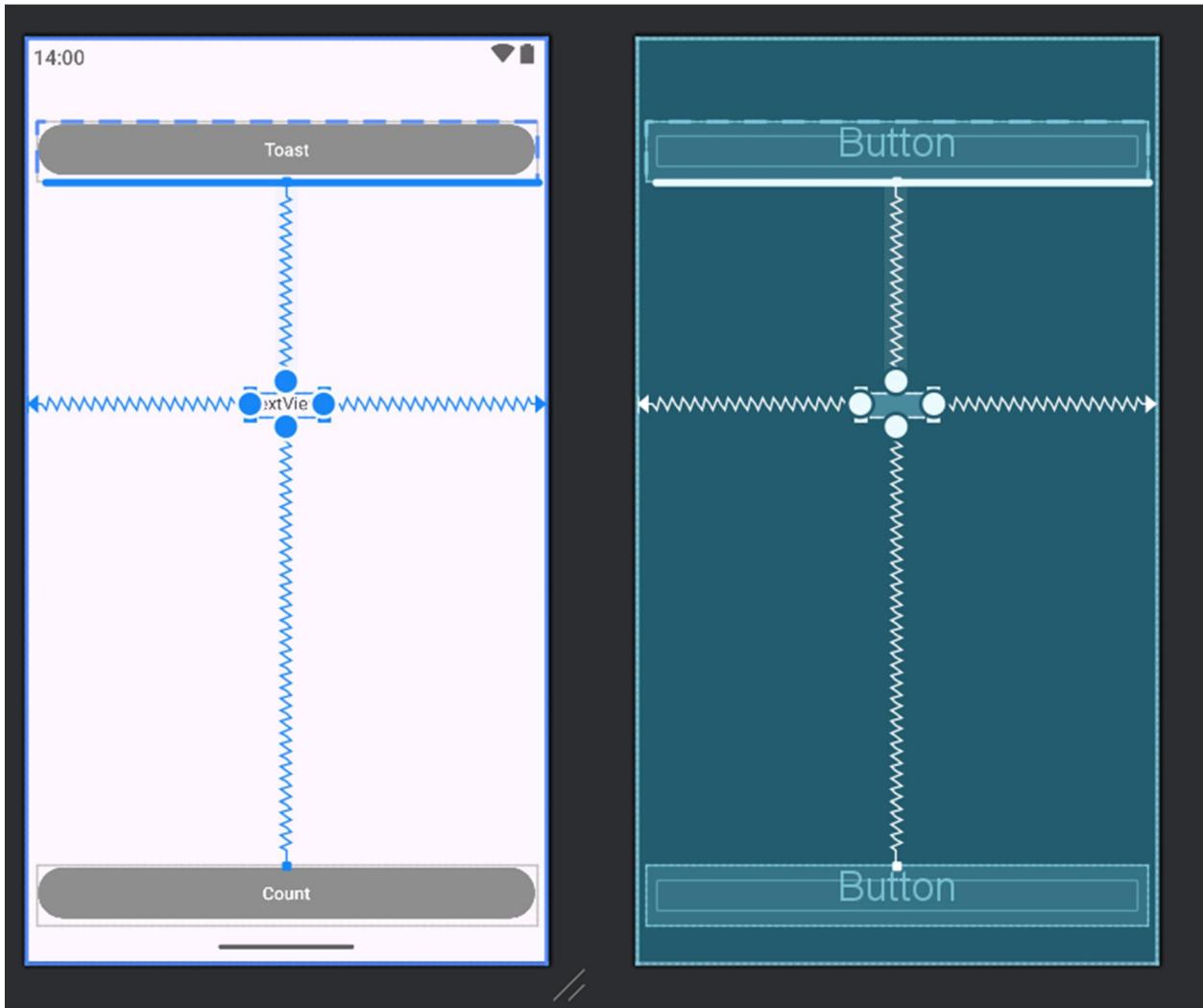
Một trong những lợi ích của [ConstraintLayout](#) là khả năng căn chỉnh hoặc hạn chế các phần tử so với các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm TextView vào giữa bố cục và hạn chế theo chiều ngang với các lề và theo chiều dọc với hai phần tử Button. Sau đó, bạn sẽ thay đổi các thuộc tính cho TextView trong ngăn **Attributes**.

4.1 Thêm TextView và các ràng buộc

1. Như được hiển thị trong hình bên dưới, hãy kéo TextView từ ngăn **Palette** đến phần trên của bố cục và kéo một hạn chế từ đầu TextView đến tay cầm ở phía dưới của nút **Toast**. Thao tác này hạn chế TextView nằm bên dưới Button



2. Như hiển thị trong hình động bên dưới, kéo một ràng buộc từ cạnh dưới của TextView đến tay cầm ở phía trên của nút **Count**, và từ hai bên của TextView đến hai bên của bố cục. Điều này ràng buộc TextView nằm ở giữa bố cục giữa hai phần tử Button.

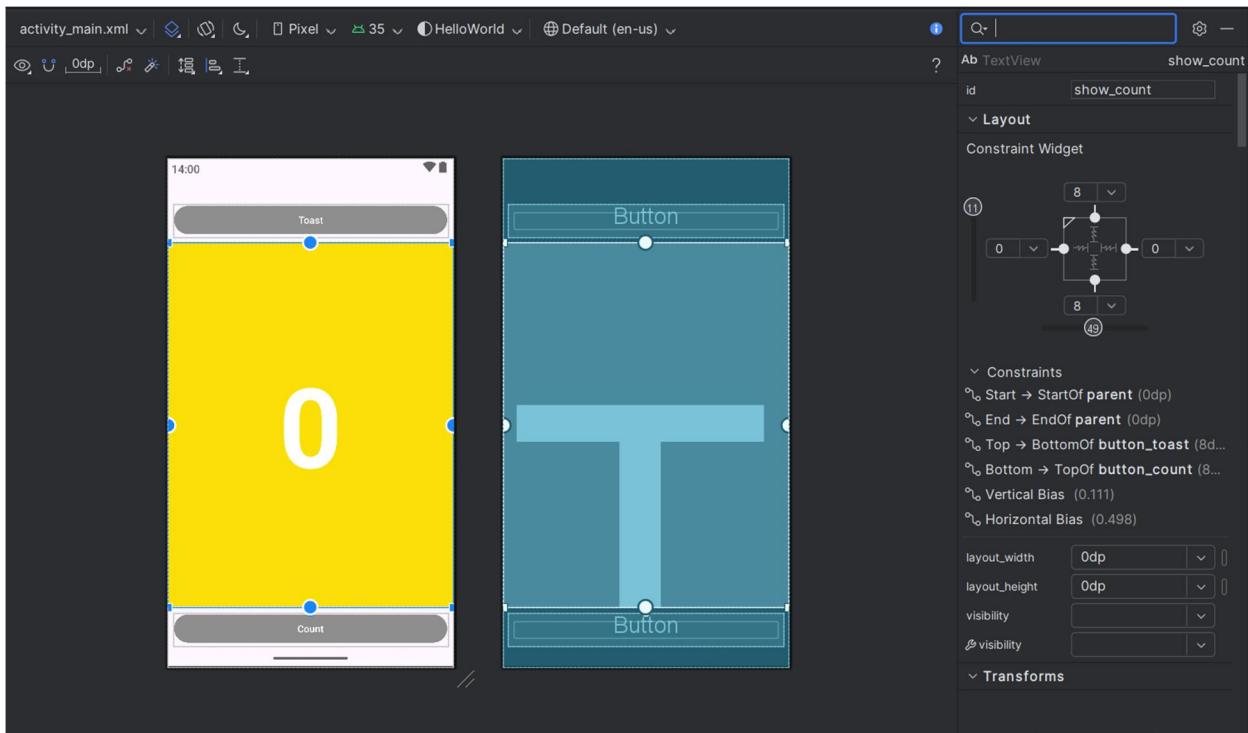


4.2 Thiết lập các thuộc tính của TextView

Với **TextView** được chọn, mở bảng **Attributes** nếu nó chưa được mở. Đặt các thuộc tính cho **TextView** như hiển thị trong hình bên dưới. Các thuộc tính mà bạn chưa gặp sẽ được giải thích sau hình:

1. Đặt **ID** thành **show_count**.
2. Đặt **text** thành **0**.
3. Đặt **textSize** thành **160sp**.
4. Đặt **textStyle** thành **B (bold)** và **textAlignment** thành **CENTER** (căn giữa đoạn văn).
5. Thay đổi điều khiển kích thước ngang và dọc của chế độ xem (**layout_width** và **layout_height**) thành **match_constraint**.

6. Đặt `textColor` thành `@color/design_default_color_on_primary`.
7. Cuộn xuống ngăn và nhấp vào **View all attributes**, cuộn xuống trang thứ hai của thuộc tính đến **background**, sau đó nhập `#FBE008` để có màu vàng.
8. Cuộn xuống **gravity**, mở rộng gravity và chọn **center_ver** (cho center-vertical).



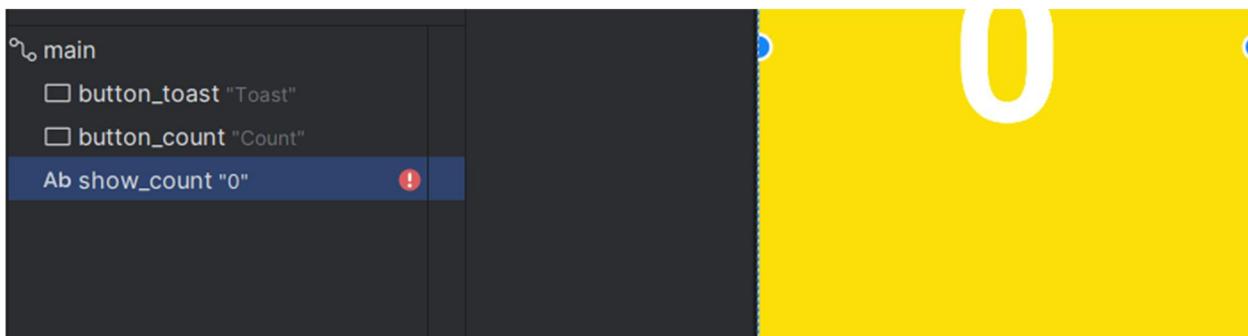
- **textSize**: Kích thước chữ của **TextView**. Trong bài học này, kích thước được đặt là **160sp**. **sp** là viết tắt của **scale-independent pixel** và giống như **dp**, đây là đơn vị tỉ lệ theo mật độ màn hình và cài đặt kích thước chữ của người dùng. Hãy sử dụng đơn vị **dp** khi chỉ định kích thước phông chữ để đảm bảo kích thước được điều chỉnh phù hợp với cả mật độ màn hình và tùy chọn của người dùng.
- **textStyle** và **textAlignment**: Kiểu chữ, được đặt thành **B (bold)** trong bài học này, và căn chỉnh văn bản, được đặt thành **CENTER** (căn giữa đoạn văn).
- **gravity**: Thuộc tính **gravity** xác định cách một View được căn chỉnh bên trong View hoặc ViewGroup cha của nó. Trong bước này, bạn căn giữa TextView theo chiều dọc bên trong ConstraintLayout cha.

Bạn có thể nhận thấy rằng thuộc tính `background` xuất hiện trên trang đầu tiên của **Attributes** đối với **Button**, nhưng lại nằm trên trang thứ hai đối với **TextView**. Bảng **Attributes** thay đổi tùy theo từng loại **View**: Các thuộc tính phổ biến nhất của loại View sẽ xuất hiện trên trang đầu tiên, và các thuộc tính còn lại sẽ được liệt kê trên trang thứ

hai. Để quay lại trang đầu tiên của bảng **Attributes**, hãy nhấp vào biểu tượng  trên thanh công cụ ở đầu bảng.

Nhiệm vụ 5: Chỉnh sửa bố cục trong XML

Bố cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử giao diện người dùng trong *Component Tree*. Di chuột qua các dấu chấm than này để xem thông báo cảnh báo, như hiển thị bên dưới. Cùng một cảnh báo xuất hiện cho cả ba phần tử: các chuỗi được mã hóa cứng nên sử dụng tài nguyên.



Cách dễ nhất để khắc phục các vấn đề về bố cục là chỉnh sửa bố cục ở trong XML. Mặc dù trình chỉnh sửa bố cục là một công cụ mạnh mẽ, nhưng một số thay đổi sẽ dễ thực hiện hơn trực tiếp trong mã nguồn XML.

5.1 Mở mã XML cho bố cục

Trong nhiệm vụ này, mở tệp `activity_main.xml` nếu nó chưa được mở, và click và thẻ **Text**



ở cuối trình chỉnh sửa bố cục.

Trình chỉnh sửa XML xuất hiện, thay thế các bảng thiết kế và bản vẽ bố cục. Như bạn có thể thấy trong hình bên dưới, hiển thị một phần mã XML của bố cục, các cảnh báo được đánh dấu—các chuỗi mã hóa cứng "**Toast**" và "**Count**". (Chuỗi mã hóa cứng "**0**" cũng được đánh dấu nhưng không hiển thị trong hình). Di chuột qua chuỗi mã hóa cứng "**Toast**" để xem thông báo cảnh báo.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layoutDescription="@xml/activity_main_scene"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button_toast"
        android:layout_width="395dp"
        android:layout_height="wrap_content"
        android:backgroundTint="#8E8E8E"
        android:text="Toast"
        android:textColor="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="DuplicateSpeakableTextCheck,HardcodedText,VisualLintButtonSize,TextContrastCheck,DuplicateClickableBoundsCheck,VisualLintOverlap"
        tools:layout_editor_absoluteY="65dp" />

    <Button
        android:id="@+id/button_count"
        android:layout_width="395dp"
        android:layout_height="wrap_content"
        android:backgroundTint="#8E8E8E"
        android:text="Count"
        android:textColor="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="HardcodedText,VisualLintButtonSize,TextContrastCheck" />

```

5.2 Trích xuất tài nguyên chuỗi

Thay vì mã hóa cứng chuỗi, một phương pháp tốt nhất là sử dụng tài nguyên chuỗi, đại diện cho các chuỗi văn bản. Việc lưu trữ chuỗi trong một tệp riêng giúp dễ dàng quản lý chúng hơn, đặc biệt khi bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và bản địa hóa ứng dụng, vì bạn cần tạo một tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhấp một lần vào từ "**Toast**" (cảnh báo đầu tiên được đánh dấu).
2. Nhấn **Alt + Enter** trên Windows hoặc **Option + Enter** trên macOS, sau đó chọn **Extract string resource** từ mục lục bật lên.

The screenshot shows the code editor for `activity_main.xml`. A context menu is open over the line `16 android:text="Toast"`. The menu items include:

- Extract string resource
- Create new scratch file from selection
- Override Resource in Other Configuration...
- Rearrange tag attributes
- Remove attribute

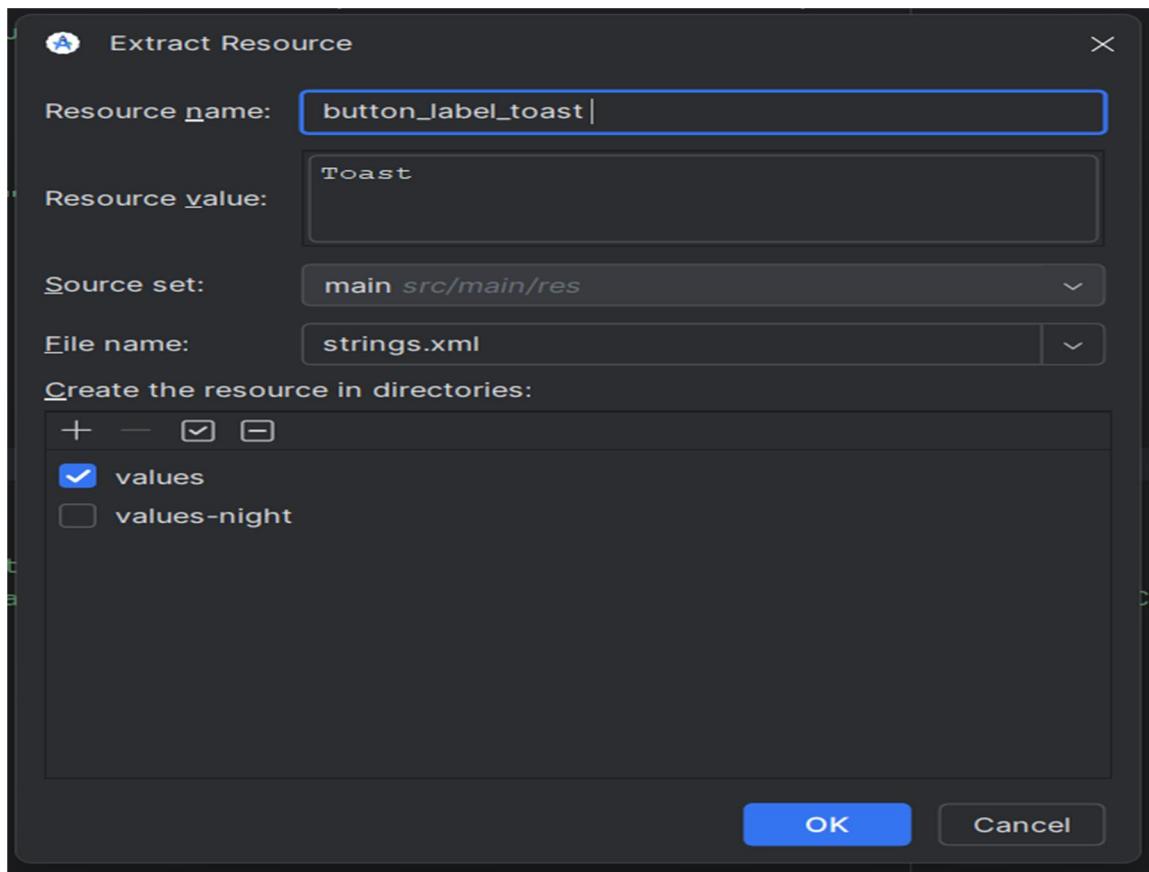
The text "Toast" is highlighted in the code, and a tooltip shows the full value: `16 android:text="@string/toast"`.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layoutDescription="@xml/activity_main_scene"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button_toast"
        android:layout_width="395dp"
        android:layout_height="wrap_content"
        android:backgroundTint="#8E8E8E"
        android:text="Toast"
        android:textColor="@color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="HardcodedText,VisualLintButtonSize,TextContrastCheck" />

    <Button
        android:id="@+id/bu
    
```

3. Nhập `button_label_toast` vào trường Resource name.



4. Nhấp **OK**. Một tài nguyên chuỗi sẽ được tạo trong tệp `values/res/strings.xml`, và chuỗi trong mã sẽ được thay thế bằng tham chiếu đến tài nguyên:

`@string/button_label_toast`

5. Thực hiện tương tự với các chuỗi còn lại: `button_label_count` cho "Count", và `count_initial_value` cho "0"
6. Trong **Project > Android**, mở rộng thư mục **values** bên trong **res**. Nhấp đúp vào `strings.xml` để xem các tài nguyên chuỗi trong tệp `strings.xml`:

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
</resources>
```

7. Bạn cần thêm một chuỗi khác để sử dụng trong một nhiệm vụ tiếp theo nhằm hiển thị thông báo. Hãy thêm vào tệp `strings.xml` một tài nguyên chuỗi mới có tên `toast_message` với nội dung "Hello Toast!" như sau:

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
    <string name="toast_message">Hello Toast!</string>
</resources>
```

Mẹo: Tài nguyên chuỗi bao gồm tên ứng dụng, xuất hiện trên thanh ứng dụng ở đầu màn hình nếu bạn bắt đầu dự án ứng dụng của mình bằng mẫu trống. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa tài nguyên `app_name`.

Nhiệm vụ 6: Thêm sự kiện onClick cho nút

Trong nhiệm vụ này, bạn thêm một phương thức Java cho mỗi Nút trong `MainActivity` để thực thi khi người dùng nhấn vào Nút.

6.1 Thêm thuộc tính onClick and trình xử lí sự kiện cho mỗi Nút

Một trình xử lý sự kiện nhấp là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử giao diện có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick ở bảng **Attributes** của thẻ **Design**. Bạn cũng có thể chỉ định tên phương thức xử lý trong trình chỉnh sửa XML bằng cách thêm thuộc tính android:onClick vào thẻ Button. Bạn sẽ sử dụng phương pháp thứ hai vì bạn chưa tạo các phương thức xử lý, và trình chỉnh sửa XML cung cấp cách tự động tạo các phương thức đó.

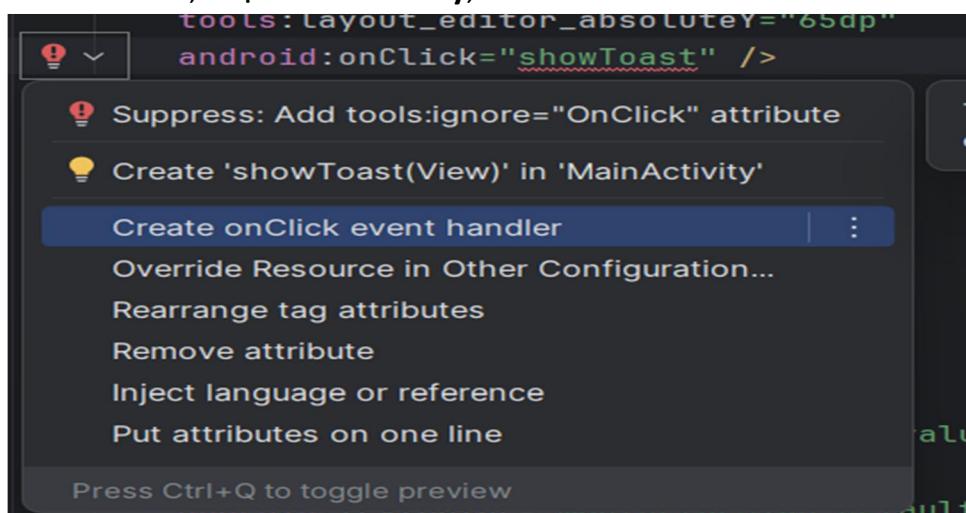
1. Với trình chỉnh sửa XML đang mở (**thẻ Text**), tìm phần tử **Button** có thuộc tính **android:id** được đặt là **button_toast**.

```
<Button  
    android:id="@+id/button_toast"  
    android:layout_width="395dp"  
    android:layout_height="wrap_content"  
    android:backgroundTint="#8E8E8E"  
    android:text="@string/button_label_toast"  
    android:textColor="@color/white"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    tools:ignore="DuplicateSpeakableTextCheck,HardcodedText,VisualLintButtonSize,TextContrastCheck"  
    tools:layout_editor_absoluteY="65dp" />
```

2. Thêm thuộc tính **android:onClick** vào cuối phần tử **button_toast**, ngay trước dấu **/>**, như sau:

```
        android:onClick="showToast" />
```

3. Nhấp vào biểu tượng bóng đèn màu đỏ xuất hiện bên cạnh thuộc tính. Chọn **Create click handler**, chọn **MainActivity**, rồi nhấn **OK**.



- Nếu biểu tượng bóng đèn không xuất hiện, nhấp vào tên phương thức (" showToast"), nhấn **Alt + Enter** (Windows) hoặc **Option + Enter** (Mac), chọn **Create 'showToast(view)' in MainActivity**, rồi nhấn **OK**.
 - Hành động này sẽ tạo một phương thức **showToast()** trong **MainActivity**.
4. Lặp lại hai bước cuối cùng với button_count Button: Thêm thuộc tính **android:onClick** vào cuối và thêm trình xử lý nhấp chuột:
- ```
 android:onClick="countUp" />
```

Mã XML cho các thành phần UI trong ConstraintLayout hiện trông như thế này:

```
<Button
 android:id="@+id/button_toast"
 android:layout_width="395dp"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:text="@string/button_label_toast"
 android:textColor="@color/white"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 tools:ignore="DuplicateSpeakableTextCheck,HardcodedText,VisualLintButtonSize,TextContrastCheck"
 tools:layout_editor_absoluteY="65dp"
 android:onClick="showToast" />

<Button
 android:id="@+id/button_count"
 android:layout_width="395dp"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:text="@string/button_label_count"
 android:textColor="@color/white"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 tools:ignore="HardcodedText,VisualLintButtonSize,TextContrastCheck"
 tools:layout_editor_absoluteY="653dp"
 android:onClick="countUp" />
```

5. Nếu MainActivity.java chưa được mở, hãy mở **java** trong chế độ xem Project > Android, mở **com.example.android.hellotoast**, rồi nhấp đúp vào **MainActivity**.  
Trình biên tập mã xuất hiện với mã trong MainActivity:

```
package com.example.helloworld;

import android.os.Bundle;
import android.util.Log;
import android.view.View;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

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

 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.button), new ViewCompat.OnApplyWindowInsetsListener() {
 @Override
 public WindowInsetsCompat onApplyWindowInsets(View v, WindowInsetsCompat insets) {
 if (insets.isFromUser()) {
 Log.d("MainActivity", "User is interacting with the screen");
 }
 return insets;
 }
 });
 }

 public void showToast(View view) {
 Toast.makeText(this, "Hello Toast!", Toast.LENGTH_SHORT).show();
 }

 public void countUp(View view) {
 int count = 0;
 while (true) {
 count++;
 if (count == 10) {
 break;
 }
 }
 }
}
```

## 6.2 Chỉnh sửa phương thức showToast()

Bây giờ bạn sẽ thực hiện chỉnh sửa phương thức showToast() - trình xử lý nhấp chuột Nút Toast trong MainActivity - để nó hiển thị một thông báo. Toast cung cấp một cách để hiển thị một thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ lấp đầy lượng không gian cần thiết cho thông báo. Hoạt động hiện tại vẫn hiển thị và tương tác. Toast có thể

hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn - thêm một thông báo Toast để hiển thị kết quả của việc chạm vào Nút hoặc thực hiện một hành động.

Làm theo các bước sau để chỉnh sửa trình xử lý nhấp chuột Nút Toast:

1. Xác định vị trí phương thức showToast() mới được tạo.

```
usage
public void showToast(View view) {
}
```

2. Để tạo một phiên bản của Toast, hãy gọi phương thức makeText() trên lớp Toast.

```
public void showToast(View view) {
 Toast toast = Toast.makeText();
}
```

Câu lệnh này chưa hoàn chỉnh cho đến khi bạn hoàn thành tất cả các bước.

3. Cung cấp `context` của Activity của ứng dụng. Vì Toast hiển thị trên giao diện Activity, hệ thống cần thông tin về Activity hiện tại. Khi bạn đang ở trong Activity cần sử dụng context, có thể dùng `this` như một cách viết tắt.

```
public void showToast(View view) {
 Toast toast = Toast.makeText(this,);
}
```

4. Cung cấp thông báo để hiển thị, chẳng hạn như một string resource (biến `toast_message` mà bạn đã tạo ở bước trước). Tài nguyên chuỗi `toast_message` được xác định bởi `R.string`

```
public void showToast(View view) {
 Toast toast = Toast.makeText(this, R.string.toast_message,);
}
```

5. Cung cấp thời lượng hiển thị. Ví dụ: `Toast.LENGTH_SHORT` hiển thị toast trong thời gian tương đối ngắn.

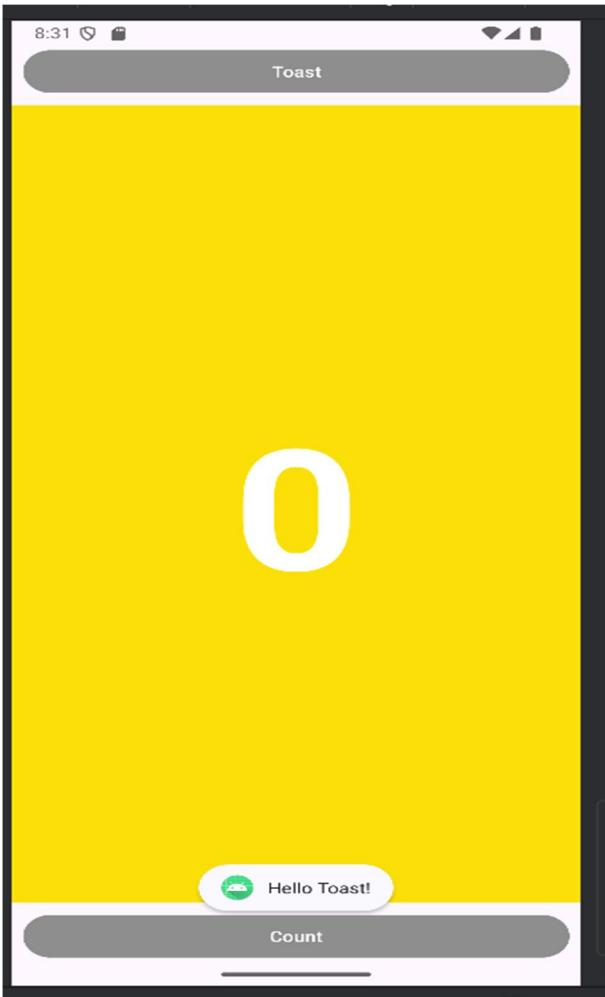
```
public void showToast(View view) {
 Toast toast = Toast.makeText(this, R.string.toast_message, Toast.LENGTH_SHORT);
}
```

Thời lượng hiển thị Toast có thể là `Toast.LENGTH_LONG` hoặc `Toast.LENGTH_SHORT`. Độ dài thực tế là khoảng 3,5 giây cho Toast dài và 2 giây cho Toast ngắn.

6. Hiển thị Toast bằng cách gọi `show()`. Sau đây là toàn bộ phương thức `showToast()`:

```
public void showToast(View view) {
 Toast toast = Toast.makeText(this, R.string.toast_message,
 Toast.LENGTH_SHORT);
 toast.show();
}
```

Chạy ứng dụng và xác minh rằng thông báo Toast xuất hiện khi bạn nhấn vào nút Toast.



### 6.3 Chỉnh sửa trình xử lý nút đếm

Bây giờ bạn sẽ chỉnh sửa phương thức **countUp()** - trình xử lý sự kiện khi nhấn nút **Count** trong **MainActivity** - để hiển thị số đếm hiện tại sau mỗi lần nhấn. Mỗi lần nhấn sẽ tăng số đếm lên một đơn vị.

Mã xử lý sự kiện phải:

- Theo dõi sự thay đổi của số đếm.
- Gửi số đếm đã cập nhật đến **TextView** để hiển thị.

Thực hiện các bước sau để chỉnh sửa trình xử lý sự kiện của nút **Count**:

1. Xác định vị trí phương thức **countUp()** mới được tạo.

```
public void countUp(View view) {
}
```

2. Để theo dõi số đếm, bạn cần một biến thành viên **private**. Mỗi lần nhấn nút **Count**, giá trị của biến này sẽ tăng lên. Nhập nội dung sau, nội dung này sẽ được tô sáng màu đỏ và hiển thị biểu tượng bóng đèn màu đỏ:

```
public void countUp(View view) {
 mCount++;
}
```

Nếu biểu tượng bóng đèn màu đỏ không xuất hiện, hãy chọn biểu thức `mCount++`. Cuối cùng, bóng đèn màu đỏ sẽ xuất hiện

3. Nhấp vào biểu tượng bóng đèn màu đỏ và chọn **Tạo trường 'mCount'** từ menu bật lên. Thao tác này sẽ tạo một biến thành viên riêng tư ở đầu `MainActivity` và `Android Studio` giả định rằng bạn muốn biến đó là số nguyên (`int`):

```
public class MainActivity extends AppCompatActivity {
 private int mCount;
```

4. Thay đổi khai báo biến thành viên cục bộ để khởi tạo biến với giá trị ban đầu là **0**:

```
public class MainActivity extends AppCompatActivity {
 private int mCount = 0;
```

5. Cùng với biến trên, bạn cũng cần một biến thành viên cục bộ để tham chiếu đến `TextView` có ID là `show_count`, biến này là biến mà bạn sẽ thêm vào trình xử lý nhấp chuột. Gọi biến này là `mShowCount`:

```
public class MainActivity extends AppCompatActivity {
 private int mCount = 0;
 private TextView mShowCount;
```

6. Nay bạn đã có `mShowCount`, bạn có thể lấy tham chiếu đến `TextView` bằng ID bạn đã đặt trong tệp bố cục. Để chỉ lấy tham chiếu này một lần, hãy chỉ định tham chiếu đó trong phương thức `onCreate()`. Như bạn đã tìm hiểu trong bài học khác, phương thức `onCreate()` được sử dụng để làm tải bố cục, có nghĩa là đặt chế độ xem nội dung của màn hình thành bố cục XML. Bạn cũng có thể sử dụng phương thức này để lấy tham chiếu đến các thành phần UI khác trong giao diện, chẳng hạn như `TextView`. Xác định vị trí phương thức `onCreate()` trong `MainActivity`:

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

7. Thêm câu lệnh `findViewById` vào cuối phương thức:

```
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 EdgeToEdge.enable(this);
 setContentView(R.layout.activity_main);
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
 Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
 v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
 return insets;
 });
 mShowCount = (TextView) findViewById(R.id.show_count);
```

Một View, giống như một chuỗi, là một tài nguyên có thể có một id. Lệnh gọi `findViewById` lấy ID của một view làm tham số của nó và trả về View. Vì phương thức trả về View, nên bạn phải ép kết quả thành kiểu view mà bạn mong đợi, trong trường hợp này là (`TextView`).

8. Bây giờ bạn đã gán `TextView` cho `mShowCount`, bạn có thể sử dụng biến để đặt văn bản trong `TextView` thành giá trị của biến `mCount`. Thêm nội dung sau vào phương thức `countUp()`

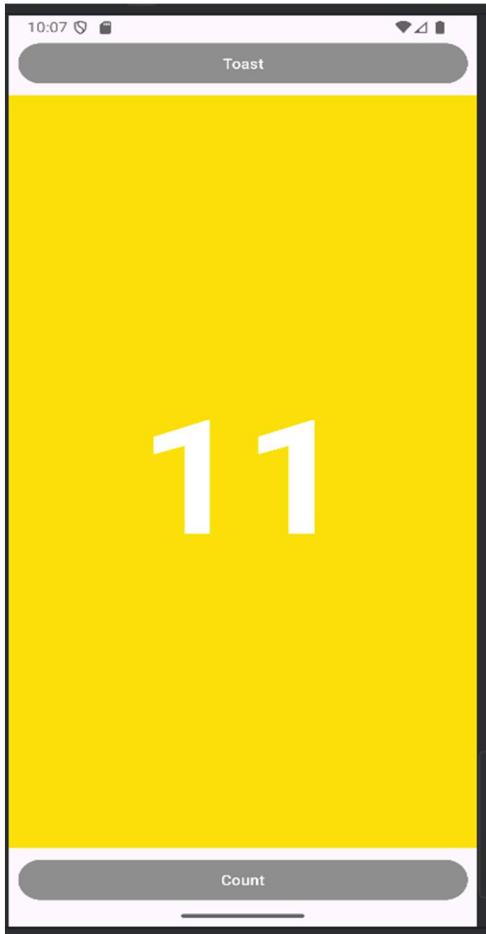
```
if(mShowCount != null)
 mShowCount.setText(Integer.toString(mCount));
```

Toàn bộ phương thức `countUp()` hiện trông như thế này:

```
public void countUp(View view) {
 mCount++;
 if(mShowCount != null)
 mShowCount.setText(Integer.toString(mCount));
```

```
}
```

9. Chạy ứng dụng để xác minh số đếm tăng lên khi bạn chạm vào nút Đếm.



**Mẹo:** Để biết hướng dẫn chi tiết về cách sử dụng ConstraintLayout, hãy xem Codelab Sử dụng ConstraintLayout để thiết kế chế độ xem của bạn.

## Mã giải pháp

Dự án Android Studio: [HelloToast](#)

## Tóm tắt

View, ViewGroup và layout:

- Tất cả các thành phần UI đều là lớp con của lớp View và do đó kế thừa nhiều thuộc tính của lớp siêu View.
- Các thành phần View có thể được nhóm bên trong ViewGroup, hoạt động như một vùng chứa. Mỗi quan hệ là cha-con, trong đó cha là ViewGroup và con là View hoặc một ViewGroup khác.
- Phương thức `onCreate()` được sử dụng để làm phông layout, nghĩa là đặt chế độ xem nội dung của màn hình thành layout XML. Bạn cũng có thể sử dụng phương thức này để lấy tham chiếu đến các thành phần UI khác trong layout.
- View, giống như một chuỗi, là một tài nguyên có thể có một id. Lệnh gọi `findViewById` lấy ID của một view làm tham số của nó và trả về View .

Sử dụng trình chỉnh sửa bố cục:

- Nhấp vào thẻ **Design** để thao tác các thành phần và bố cục, và thẻ **Text** để chỉnh sửa mã XML cho bố cục.
- Trong tab **Thiết kế**, ngăn **Palettes** hiển thị các thành phần UI mà bạn có thể sử dụng trong bố cục của ứng dụng và ngăn **Component tree** hiển thị hệ thống phân cấp chế độ xem của các thành phần UI.
- Ngăn thiết kế và bản thiết kế của trình chỉnh sửa bố cục hiển thị các thành phần UI trong bố cục.
- Thẻ **Attributes** hiển thị ngăn **Attributes** để thiết lập các thuộc tính cho một thành phần UI.
- Tay cầm ràng buộc: Nhấp vào tay cầm ràng buộc, được hiển thị dưới dạng một vòng tròn ở mỗi bên của một thành phần, và sau đó kéo đến tay cầm ràng buộc khác hoặc đến ranh giới cha để tạo một ràng buộc. Ràng buộc được biểu thị bằng đường ngoằn ngoèo.
- Tay cầm thay đổi kích thước: Bạn có thể kéo tay cầm thay đổi kích thước hình vuông để thay đổi kích thước phần tử. Trong khi kéo, tay cầm sẽ thay đổi thành góc nghiêng.
- Khi được bật, công cụ **Tự động kết nối** sẽ tự động tạo hai hoặc nhiều ràng buộc cho một phần tử UI vào bố cục cha. Sau khi bạn kéo phần tử vào bố cục, công cụ này sẽ tạo ra các ràng buộc dựa trên vị trí của phần tử.

- Bạn có thể xóa các ràng buộc khỏi phần tử bằng cách chọn phần tử và di con trỏ lên phần tử đó để hiển thị nút Xóa ràng buộc. Nhấp vào nút này để xóa tất cả các ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc duy nhất, hãy nhấp vào tay cầm cụ thể đặt ràng buộc đó.
- **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Ngăn này cũng bao gồm một bảng điều khiển kích thước hình vuông được gọi là thanh tra chế độ xem ở trên cùng. Các biểu tượng bên trong hình vuông biểu thị các thiết lập chiều cao và chiều rộng.

Thiết lập chiều rộng và chiều cao bố cục:

Các thuộc tính layout\_width và layout\_height thay đổi khi bạn thay đổi chiều cao và kích thước chiều rộng các điều khiển trong trình kiểm tra chế độ xem. Các thuộc tính này có thể lấy một trong ba giá trị cho một ConstraintLayout:

- Thiết lập match\_constraint mở rộng chế độ xem để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao - lên đến một lề, nếu có.
- Thiết lập wrap\_content thu nhỏ kích thước chế độ xem để chế độ xem chỉ đủ lớn để bao gồm nội dung của nó. Nếu không có nội dung, chế độ xem sẽ trở nên vô hình.
- Sử dụng số lượng dp cố định (pixel không phụ thuộc mật độ) để chỉ định kích thước cố định, được điều chỉnh cho kích thước màn hình của thiết bị.

Trích xuất tài nguyên chuỗi:

Thay vì mã hóa cứng chuỗi, cách tốt nhất là sử dụng tài nguyên chuỗi, biểu diễn các chuỗi. Thực hiện theo các bước sau:

1. Nhấp một lần vào chuỗi được mã hóa cứng để trích xuất, nhấn **Alt-Enter (Option-Enter** trên máy Mac), và chọn Trích xuất tài nguyên chuỗi từ mục lục bật lên.
2. Đặt tên Tài nguyên.
3. Nhấp vào OK. Thao tác này sẽ tạo một tài nguyên chuỗi trong tệp values/res/string.xml và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: @string/button\_label\_toast

Xử lý nhấp chuột:

- Trình xử lý nhấp chuột là phương thức được gọi khi người dùng nhấp hoặc chạm vào phần tử UI.

- Chỉ định trình xử lý nhấp cho một phần tử UI như Button bằng cách nhập tên của nó vào trường onClick trong ngăn Thuộc tính của tab Thiết kế hoặc trong trình soạn thảo XML bằng cách thêm thuộc tính android:onClick vào một phần tử UI như Button.
- Tạo trình xử lý nhấp trong Hoạt động chính bằng cách sử dụng tham số View. Ví dụ: public void showToast(View view) {/...}.
- Bạn có thể tìm thông tin về tất cả các thuộc tính Button trong tài liệu lớp Button và tất cả các thuộc tính TextView trong tài liệu lớp TextView.

Hiển thị thông báo Toast:

[Toast](#) cung cấp một cách để hiển thị một thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ lấp đầy lượng không gian cần thiết cho thông báo. Để tạo một phiên bản của Toast, hãy làm theo các bước sau:

1. Gọi phương thức nhà máy [makeText\(\)](#) trên lớp [Toast](#).
2. Cung cấp ngữ cảnh của Hoạt động ứng dụng và thông báo để hiển thị (chẳng hạn như một chuỗi tài nguyên).
3. Cung cấp thời lượng hiển thị, ví dụ [Toast.LENGTH\\_SHORT](#) trong một khoảng thời gian ngắn. Thời lượng có thể là [Toast.LENGTH\\_LONG](#) hoặc [Toast.LENGTH\\_SHORT](#).
4. Hiển thị Toast bằng cách gọi [show\(\)](#).

## Các khái niệm liên quan

Tài liệu về khái niệm liên quan có trong [1.2: Bố cục và tài nguyên cho UI](#).

## Tìm hiểu thêm

Tài liệu dành cho nhà phát triển Android:

- [Android Studio](#)
- [Xây dựng giao diện người dùng bằng Trình chỉnh sửa bố cục](#)
- [Xây dựng giao diện người dùng phản hồi bằng ConstraintLayout](#)
- [Bố cục](#)
- [Giao diện](#)
- [Nút](#)

- [TextView](#)
- [Tài nguyên Android](#)
- [Tài nguyên R.color chuẩn của Android](#)
- [Hỗ trợ các mật độ khác nhau](#)
- [Sự kiện đầu vào Android](#)
- [Context](#)

Khác:

- Codelab: Sử dụng ConstraintLayout để thiết kế chế độ xem của bạn
- Từ vựng và thuật ngữ thuật ngữ

### 1.3) Trình chỉnh sửa bố cục

## Giới thiệu

Như bạn đã học trong [Phần 1.2 Giao diện tương tác đầu tiên](#), bạn có thể xây dựng giao diện người dùng (UI) bằng [ConstraintLayout](#) trong trình chỉnh sửa bố cục (layout editor). ConstraintLayout sắp xếp các phần tử giao diện trong bố cục bằng cách sử dụng các ràng buộc (constraint connections) với các phần tử khác và với các cạnh của bố cục. ConstraintLayout được thiết kế để giúp bạn dễ dàng kéo các phần tử UI vào trình chỉnh sửa bố cục.

ConstraintLayout là một [ViewGroup](#), tức là một View đặc biệt có thể chứa các đối tượng View khác (gọi là children hoặc child views). Bài thực hành này giới thiệu thêm các tính năng của ConstraintLayout và trình chỉnh sửa bố cục (layout editor).

Bài thực hành này cũng giới thiệu hai lớp con khác của [ViewGroup](#):

- [LinearLayout](#): Một nhóm căn chỉnh các phần tử View bên trong nó theo chiều ngang hoặc chiều dọc.
- [RelativeLayout](#): Một nhóm các phần tử View, trong đó mỗi phần tử View được định vị và căn chỉnh tương đối với các phần tử View khác trong ViewGroup. Vị trí của các phần tử View con được mô tả dựa trên mối quan hệ với nhau hoặc với ViewGroup cha.

## Những gì bạn nên biết

Bạn nên có khả năng:

- Tạo ứng dụng Hello World bằng Android Studio.

- Chạy ứng dụng trên trình giả lập hoặc thiết bị.
- Tạo bối cảnh đơn giản cho ứng dụng bằng ConstraintLayout.
- Trích xuất và sử dụng tài nguyên chuỗi.

## Những thứ bạn sẽ tìm hiểu

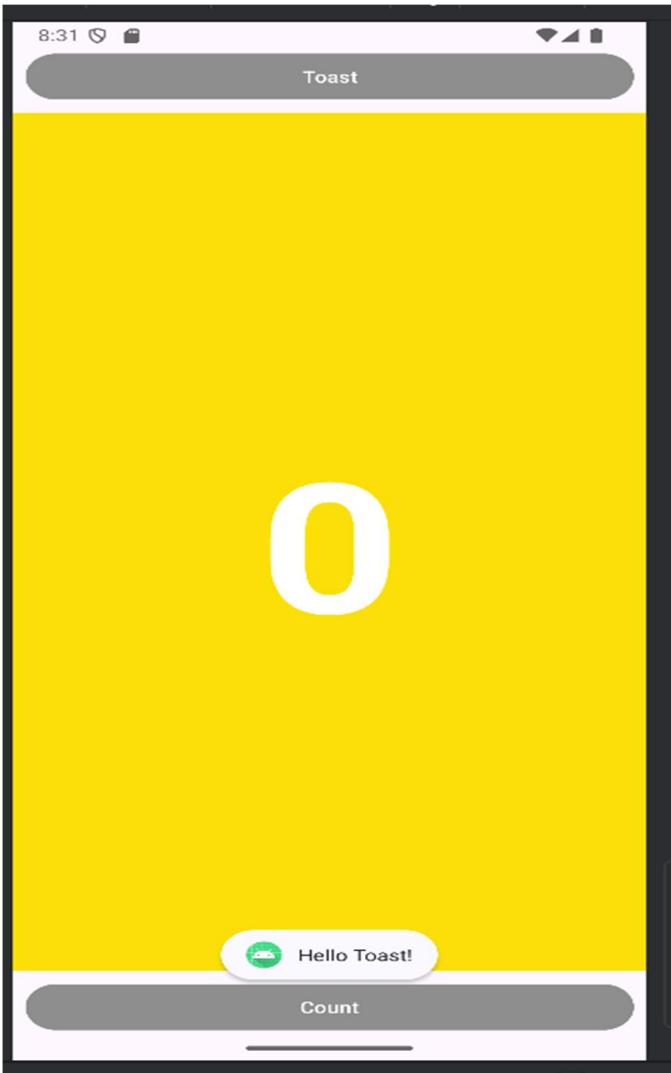
- Làm sao để tạo biến thể bối cảnh theo hướng ngang (ngang).
- Làm thế nào để tạo một biến thể bối cảnh cho máy tính bảng và màn hình lớn.
- Làm thế nào để sử dụng ràng buộc đường cơ sở để căn chỉnh các phần tử UI với văn bản
- Làm thế nào sử dụng các nút căn chỉnh và gói để sắp xếp các phần tử trong bối cảnh.
- Làm thế nào định vị các view trong LinearLayout.
- Làm thế nào định vị các view trong RelativeLayout

## Những thứ bạn sẽ làm

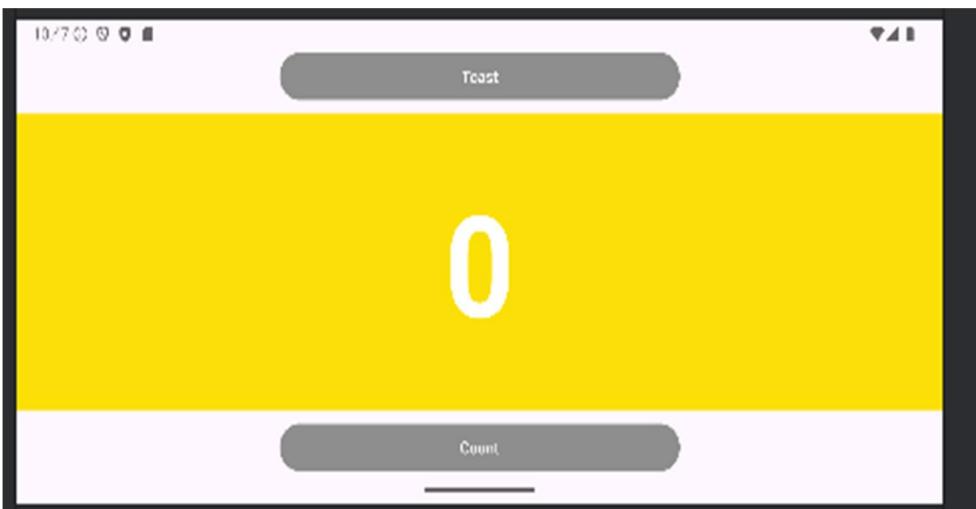
- Tạo một biến thể bối cảnh cho màn hình ngang.
- Tạo một biến thể bối cảnh cho máy tính bảng và màn hình lớn hơn.
- Chính sửa bối cảnh để thêm ràng buộc cho các phần tử UI.
- Sử dụng **ConstraintLayout** với ràng buộc đường cơ sở để căn chỉnh phần tử với văn bản.
- Sử dụng **ConstraintLayout** với các nút căn chỉnh và gói để sắp xếp phần tử.
- Thay đổi bối cảnh để sử dụng **LinearLayout**.
- Định vị các phần tử trong **LinearLayout**.
- Thay đổi bối cảnh để sử dụng **RelativeLayout**.
- Sắp xếp lại các view trong bối cảnh chính để liên kết tương đối với nhau.

## Tổng quan về ứng dụng

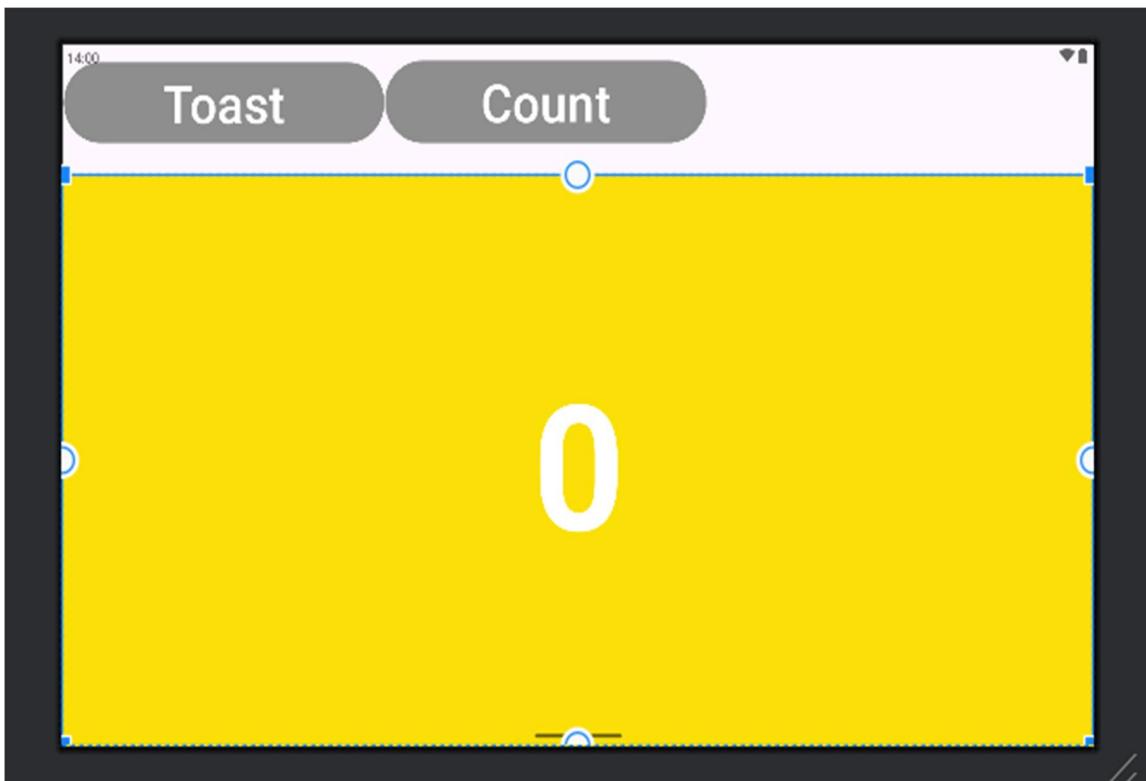
Ứng dụng Hello Toast ở trong bài học trước sử dụng **ConstraintLayout** để vẽ các phần tử UI ở trong bối cảnh Activity, hiển thị như hình ảnh bên dưới



Để thực hành nhiều hơn với ConstraintLayout, bạn sẽ tạo một phiên bản khác của bố cục này cho chế độ ngang như minh họa trong hình bên dưới.



Bạn cũng sẽ học cách sử dụng các ràng buộc cơ sở và một số tính năng căn chỉnh của ConstraintLayout bằng cách tạo một biến thể bố cục khác cho màn hình máy tính bảng.

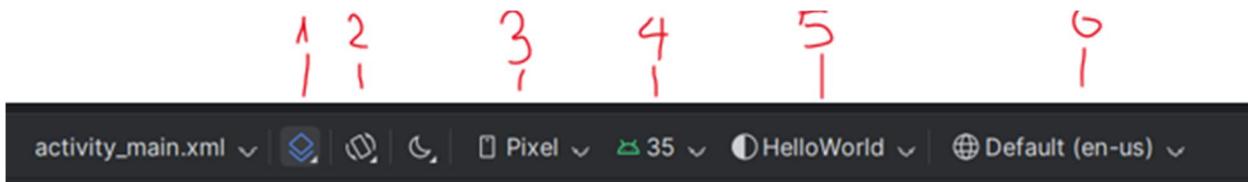


Bạn cũng tìm hiểu về các lớp con ViewGroup khác như LinearLayout và RelativeLayout và thay đổi bố cục ứng dụng Hello Toast để sử dụng chúng.

## Nhiệm vụ 1: Tạo các biến thể bố cục

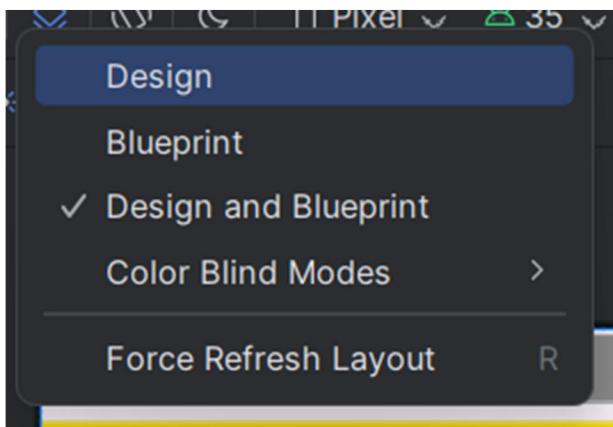
Trong bài học trước, thử thách mã hóa yêu cầu thay đổi bố cục của ứng dụng Hello Toast để nó có thể vừa vận theo hướng ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo các biến thể bố cục của mình theo hướng ngang (còn gọi là ngang) và hướng dọc (còn gọi là dọc) cho điện thoại và cho màn hình lớn hơn như máy tính bảng.

Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trong hai thanh công cụ trên cùng của trình chỉnh sửa bố cục. Thanh công cụ trên cùng cho phép bạn định cấu hình giao diện của bản xem trước bố cục trong trình chỉnh sửa bố cục:



Trong hình trên:

- Chọn **Design Surface**: Chọn **Design** để hiển thị bản xem trước màu của bố cục hoặc **Blueprint** để chỉ hiển thị các phác thảo cho từng thành phần UI. Để xem cả hai ngang cạnh nhau, hãy chọn **Design + Blueprint**.



- Orientation in Editor**: Chọn **Portrait** hoặc **Landscape** để hiển thị bản xem trước theo hướng dọc hoặc ngang. Điều này hữu ích khi xem trước bố cục mà không cần phải chạy ứng dụng trên trình giả lập hoặc thiết bị. Để tạo bố cục thay thế, hãy chọn **Create Landscape Variation** hoặc các biến thể khác.
- Device in Editor**: Chọn loại thiết bị (điện thoại/máy tính bảng, Android TV hoặc Android Wear).
- API Version in Editor**: Chọn phiên bản Android để sử dụng để hiển thị bản xem trước.
- Theme in Editor**: Chọn một chủ đề (như AppTheme) để áp dụng cho bản xem trước.
- Locale in Editor**: Chọn ngôn ngữ và locale cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có sẵn trong tài nguyên chuỗi (xem bài học về bản địa hóa để biết chi tiết về cách thêm ngôn ngữ). Bạn cũng có thể chọn Xem trước là Phải sang Trái để xem bố cục như thể đã chọn ngôn ngữ RTL.

Thanh công cụ thứ hai cho phép bạn cấu hình giao diện của các thành phần UI trong ConstraintLayout và để phóng to và thu nhỏ bản xem trước:



Trong hình minh họa trên:

1. **Hiển thị:** Chọn **Show Constraints** và **Show Margins** để hiển thị hoặc ẩn các ràng buộc và lề trong bản xem trước.
2. **Tự động kết nối:** Bật hoặc tắt **Autoconnect**. Khi được bật, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như **Button**) đến bất kỳ vị trí nào trong bố cục để tự động tạo ràng buộc với bố cục cha.
3. **Xóa tất cả ràng buộc:** Xóa tất cả các ràng buộc trong toàn bộ bố cục.
4. **Suy luận ràng buộc:** Tạo ràng buộc bằng cách suy luận từ vị trí hiện tại của các phần tử.
5. **Lề mặc định:** Đặt giá trị lề mặc định.
6. **Gói gọn:** Gói hoặc mở rộng các phần tử đã chọn.
7. **Căn chỉnh:** Căn chỉnh các phần tử đã chọn.
8. **Đường hướng dẫn:** Thêm đường hướng dẫn dọc hoặc ngang.
9. **Điều khiển thu phóng/dịch chuyển:** Phóng to hoặc thu nhỏ.

**Mẹo:** Để tìm hiểu thêm về cách sử dụng **Layout Editor**, hãy xem **Build a UI with Layout Editor**. Để biết thêm chi tiết về cách xây dựng bố cục với **ConstraintLayout**, hãy tham khảo **Build a Responsive UI with ConstraintLayout**.

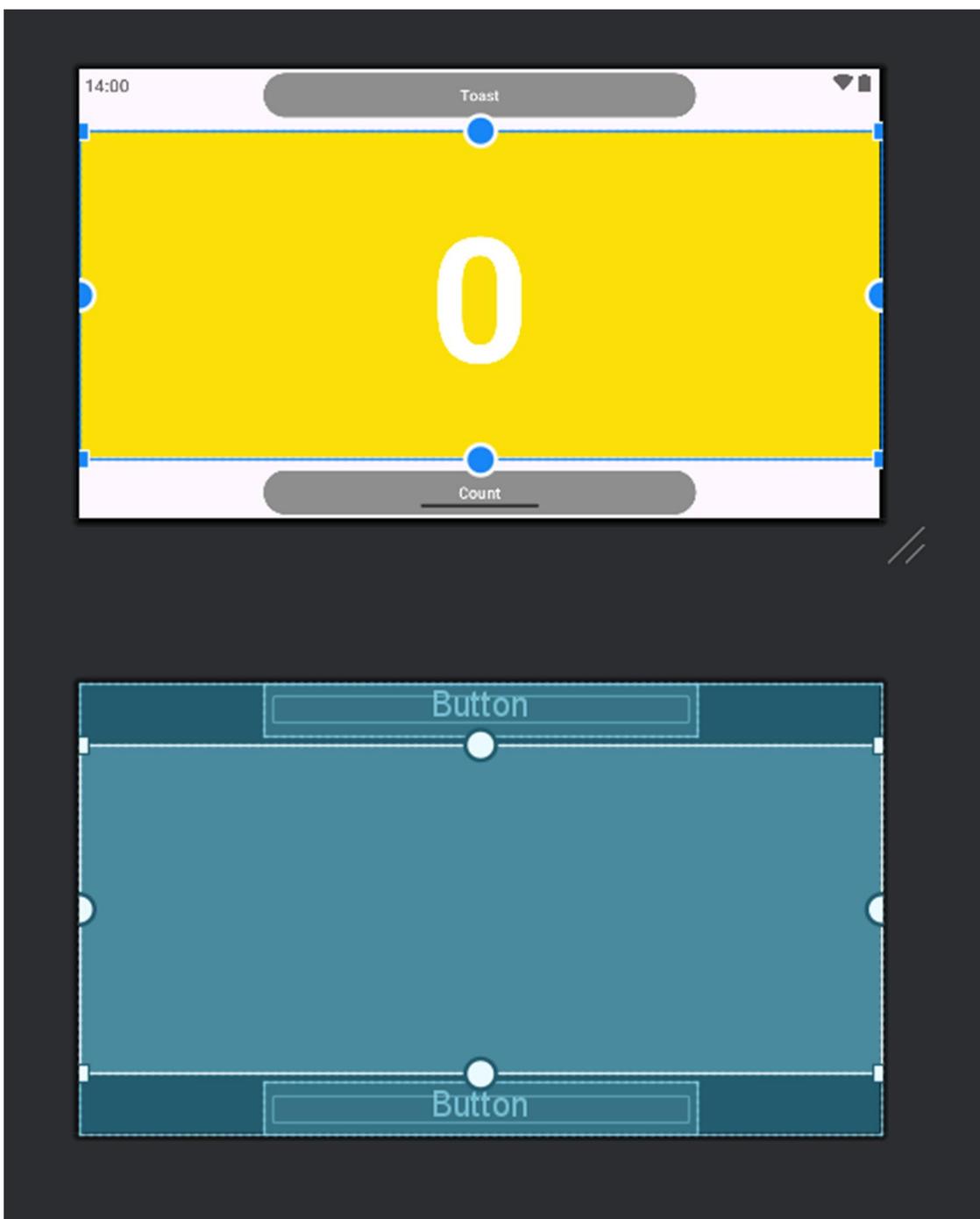
## 1.1 Xem trước bố cục ở chế độ ngang

Để xem trước bố cục của ứng dụng **Hello Toast** trong chế độ ngang, hãy làm theo các bước sau:

1. Mở ứng dụng **Hello Toast** từ bài học trước.

**Lưu ý:** Nếu bạn đã tải xuống mã hoàn chỉnh của HelloToast, bạn cần xóa các bố cục cho chế độ ngang và màn hình lớn mà bạn sẽ tạo lại trong bài này. Chuyển từ **Project > Android sang Project > Project Files** trong **Project pane**. Mở app > **src/main > res**, chọn cả thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**. Chuyển Project pane về **Project > Android**.

2. Mở tệp **activity\_main.xml**. Nhấp vào tab **Design** nếu nó chưa được chọn.
3. Nhấp vào nút **Orientation in Editor**  trên thanh công cụ trên cùng.
4. Chọn **Switch to Landscape** trong menu thả xuống. Bố cục sẽ hiển thị ở chế độ ngang như hình minh họa bên dưới. Để quay lại chế độ dọc, chọn **Switch to Portrait**.



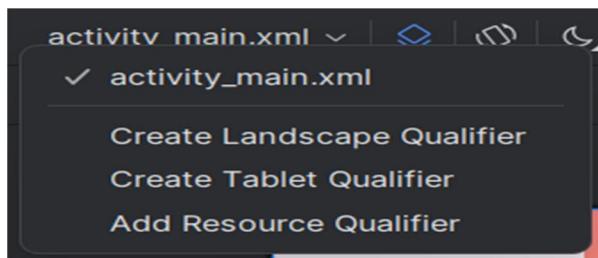
## 1.2 Tạo biến thể bố cục cho chế độ ngang

Sự khác biệt trực quan giữa bố cục ở chế độ dọc và ngang là chữ số **(0)** trong phần tử TextView show\_count hiển thị quá thấp trong chế độ ngang - quá gần với nút Đếm. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử TextView có thể quá lớn hoặc không được căn giữa vì kích thước văn bản được cố định ở mức 160sp.

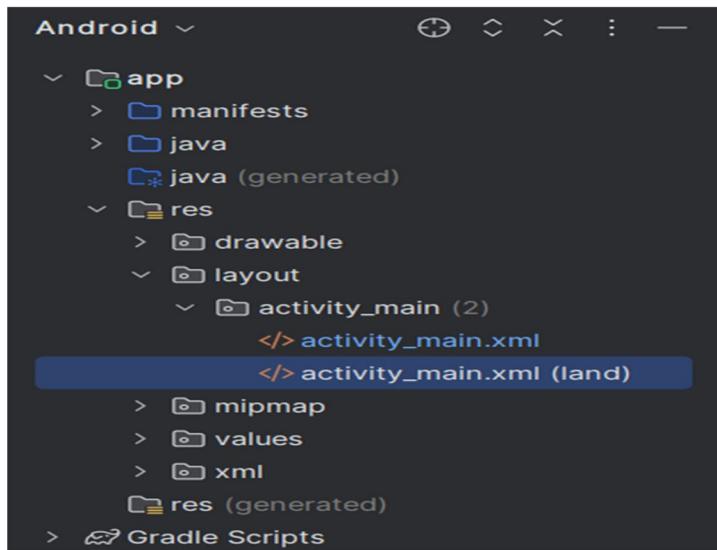
Để khắc phục điều này đối với hướng ngang trong khi vẫn giữ nguyên hướng dọc, bạn có thể tạo biến thể của bố cục ứng dụng Hello Toast khác với hướng ngang. Thực hiện theo các bước sau:

1. Nhấp vào nút **Orientation in Editor**  ở thanh công cụ trên cùng.
2. Chọn **Create Landscape Variation.**

Một cửa sổ trình chỉnh sửa mới mở ra với tab **land/activity\_main.xml** hiển thị bố cục cho hướng phong cảnh (ngang). Bạn có thể thay đổi bố cục này, dành riêng cho hướng ngang, mà không cần thay đổi hướng chân dung (dọc) ban đầu.



3. Trong **Project > Android**, hãy xem bên trong thư mục **res > layout** và bạn sẽ thấy rằng Android Studio đã tự động tạo biến thể cho bạn, có tên là **activity\_main.xml (land)**.



### 1.3 Xem trước bố cục trên các thiết bị khác nhau

Bạn có thể xem trước bố cục trên nhiều thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thực hoặc trình giả lập. Hãy làm theo các bước sau:

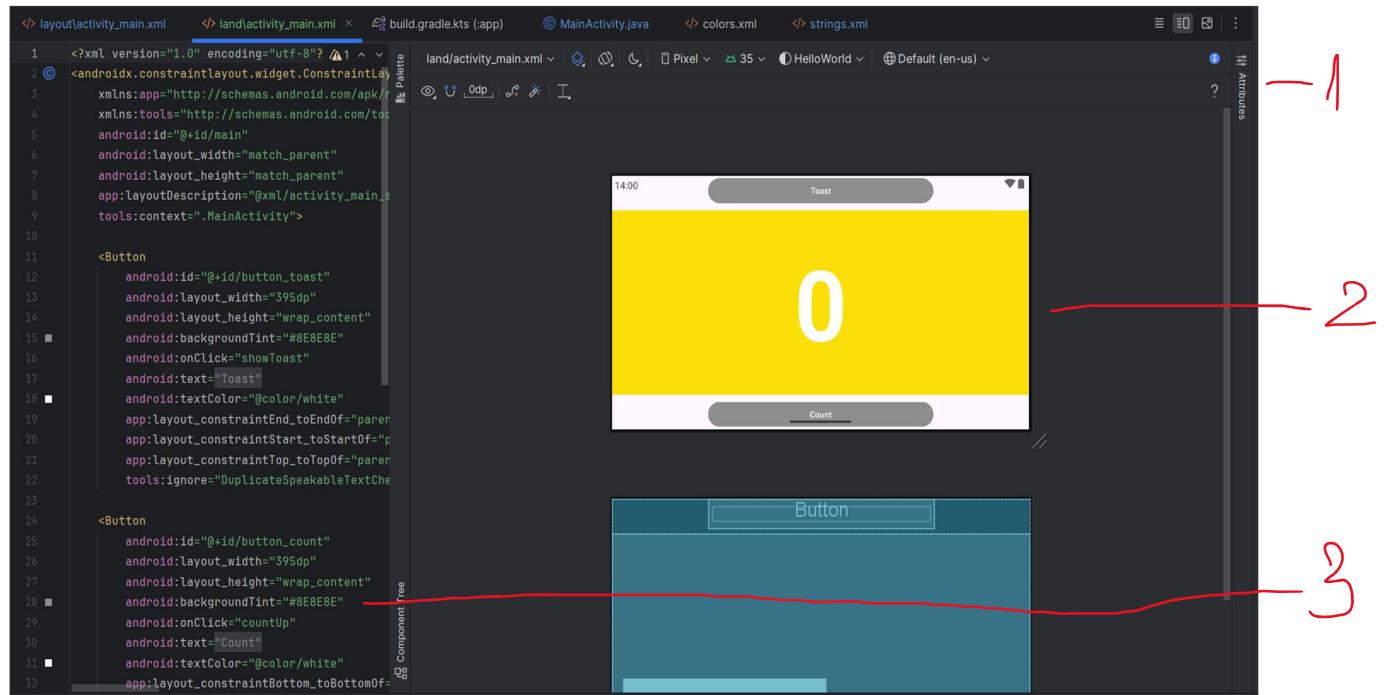
1. Thẻ **land/activity\_main.xml** vẫn nên mở trong **Layout Editor**. Nếu chưa mở, hãy double-click vào tệp **activity\_main.xml (land)** trong thư mục **layout**.

2. Nhấp vào nút **Device in Editor**  Pixel ▾ trên thanh công cụ trên cùng.

3. Chọn một thiết bị khác từ menu thả xuống. Ví dụ: Chọn **Nexus 4**, sau đó **Nexus 5**, và cuối cùng là **Pixel** để xem sự khác biệt trong bản xem trước. Những khác biệt này xảy ra do kích thước văn bản cố định trong **TextView**.

## 1.4 Thay đổi bố cục cho chế độ ngang

Bạn có thể sử dụng bảng Attributes trong thẻ **Design** để thiết lập hoặc thay đổi thuộc tính, nhưng đôi khi chỉnh sửa trực tiếp mã XML trong tab **Text** sẽ nhanh hơn. Thẻ **Text** hiển thị mã XML và cung cấp một tab **Preview** ở bên phải cửa sổ để xem trước bố cục, như minh họa trong hình bên dưới.

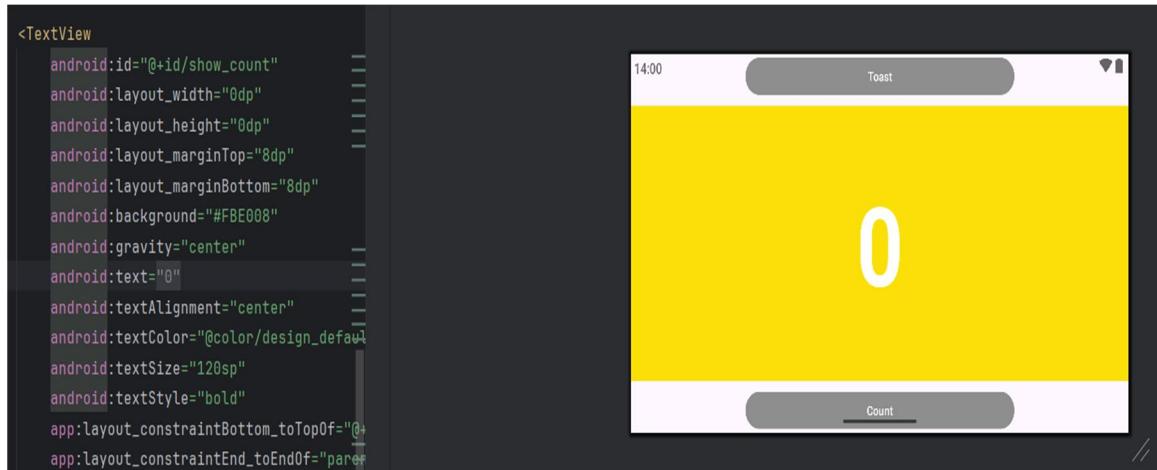


Hình ở trên hiển thị những nội dung sau:

1. Thẻ Xem trước, bạn sử dụng để hiển thị ngăn xem trước
2. Ngăn xem trước
3. Mã XML

Để thay đổi bố cục, hãy làm theo các bước sau:

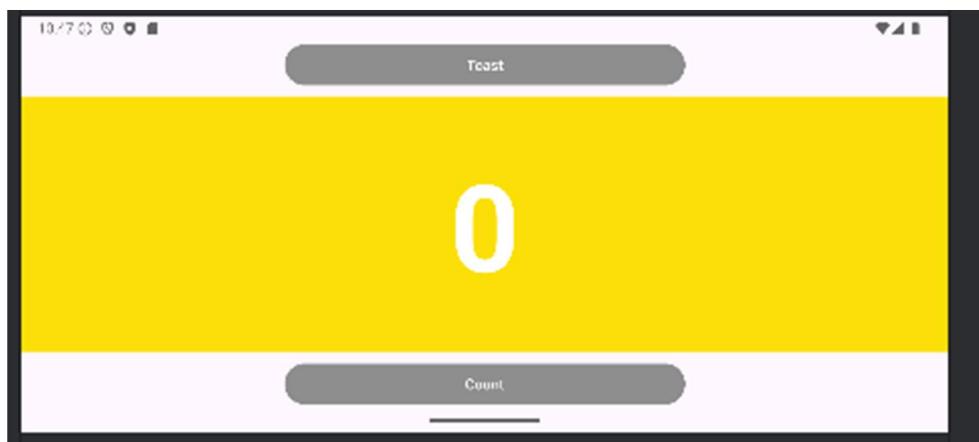
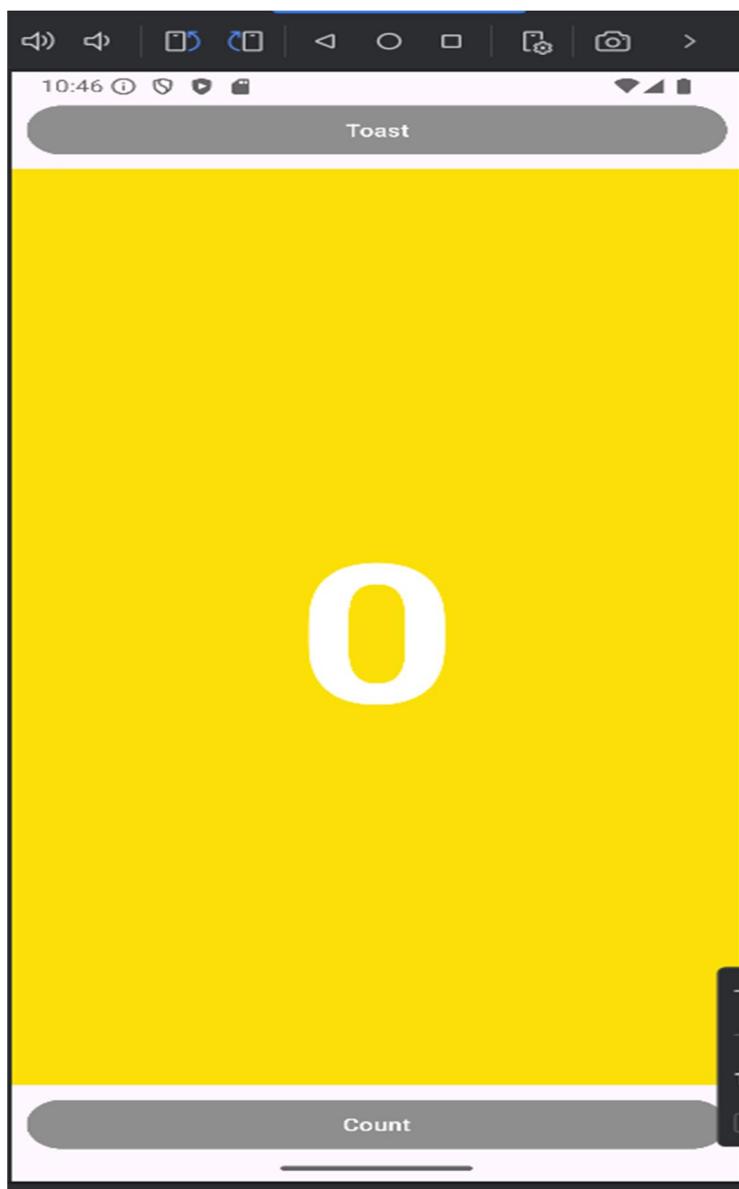
1. Thẻ **land/activity\_main.xml** vẫn phải mở trong trình chỉnh sửa bố cục; nếu không, hãy nhấp đúp vào tệp **activity\_main.xml (land)** trong thư mục bố cục.
2. Nhấp vào thẻ **Văn bản** và thẻ **Xem trước** (nếu chưa chọn).
3. Tìm phần tử **TextView** trong mã XML.
4. Thay đổi thuộc tính **android:textSize="160sp"** thành **android:textSize="120sp"**. Bản xem trước bố cục cho thấy kết quả:



5. Chọn các thiết bị khác nhau trong menu thả xuống **Device in Editor** để xem bố cục trông như thế nào trên các thiết bị khác nhau theo hướng ngang.

Trong ngăn trình chỉnh sửa, thẻ **land/activity\_main.xml** hiển thị bố cục theo hướng ngang. Thẻ **activity\_main.xml** hiển thị bố cục không thay đổi theo hướng dọc. Bạn có thể chuyển đổi qua lại bằng cách nhấp vào các thẻ.

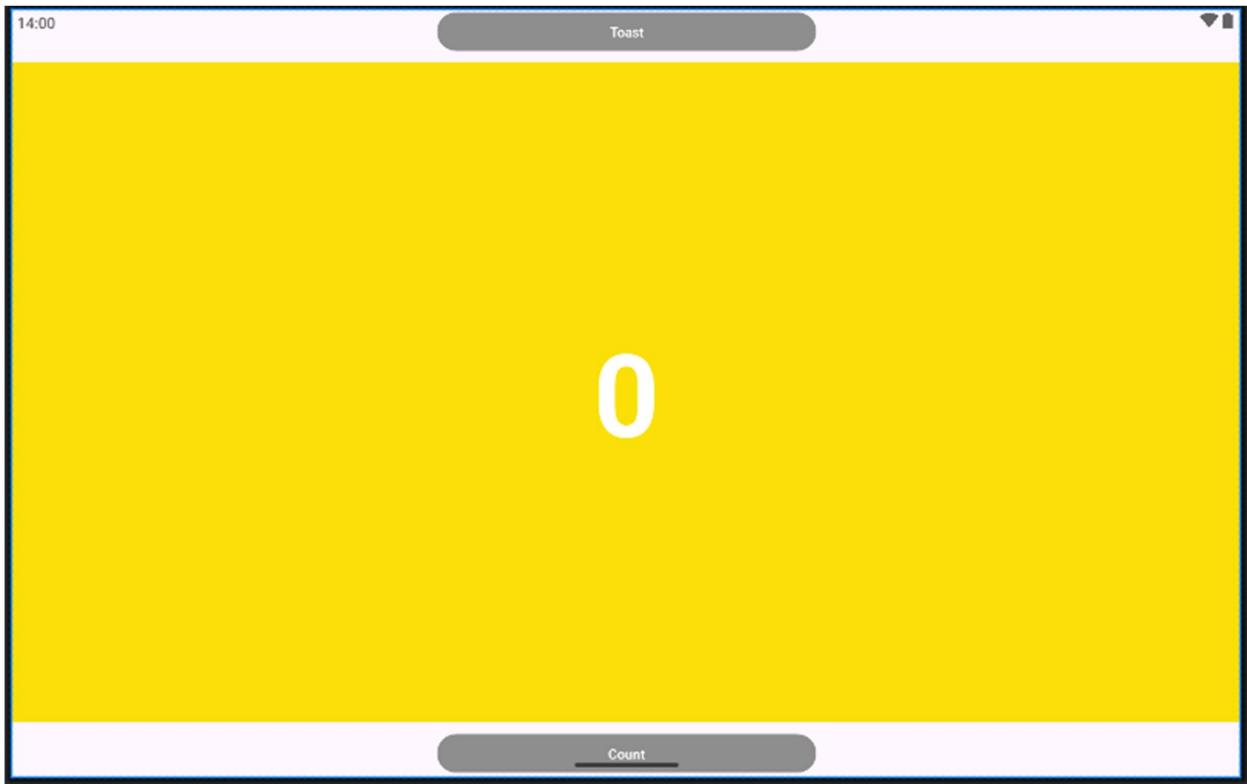
6. Chạy ứng dụng trên trình giả lập hoặc thiết bị và chuyển hướng từ dọc sang ngang để xem cả hai bố cục.



## 1.5 Tạo biến thể bố cục cho máy tính bảng

Như bạn đã tìm hiểu trước đó, bạn có thể xem trước bố cục trên các thiết bị khác nhau

bằng cách nhấp vào nút **Device in Editor** Pixel trên thanh công cụ phía trên. Nếu bạn chọn một thiết bị như **Nexus 10** (máy tính bảng) từ menu, bạn sẽ thấy rằng bố cục không phù hợp với màn hình máy tính bảng—văn bản của mỗi **Button** quá nhỏ và cách sắp xếp các phần tử **Button** ở trên và dưới không lý tưởng cho một màn hình lớn.



Để khắc phục vấn đề này trên máy tính bảng trong khi vẫn giữ nguyên bố cục cho điện thoại ở cả chế độ ngang và dọc, bạn có thể tạo một biến thể của bố cục dành riêng cho máy tính bảng. Hãy làm theo các bước sau:

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị bảng thiết kế và sơ đồ.
2. Nhấp vào nút **Orientation in Editor** trên thanh công cụ phía trên.
3. Chọn **Create layout x-large Variation**.

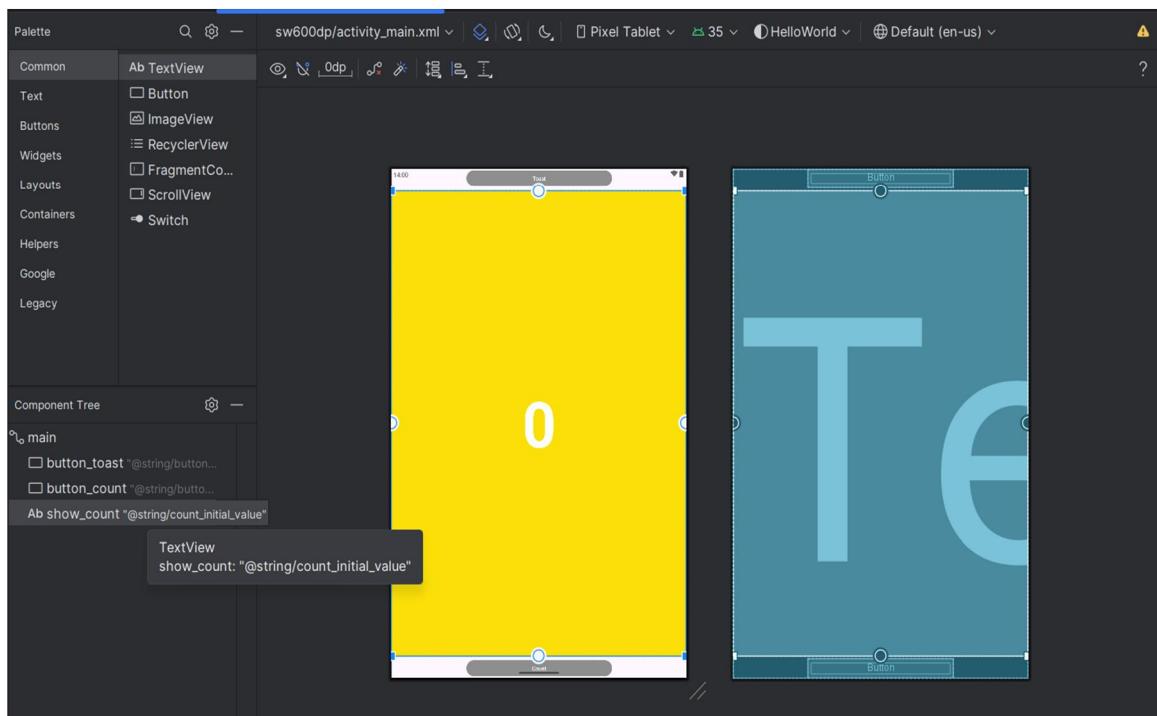
Một cửa sổ chỉnh sửa mới sẽ mở ra với tab **xlarge/activity\_main.xml**, hiển thị bố cục dành cho thiết bị có màn hình lớn như máy tính bảng. Trình chỉnh sửa cũng sẽ tự động chọn một thiết bị máy tính bảng, chẳng hạn như Nexus 9 hoặc Nexus 10, để hiển thị bản xem

trước. Bạn có thể thay đổi bố cục này dành riêng cho máy tính bảng mà không ảnh hưởng đến các bố cục khác.

## 1.6 Thay đổi biến thể bố cục cho máy tính bảng

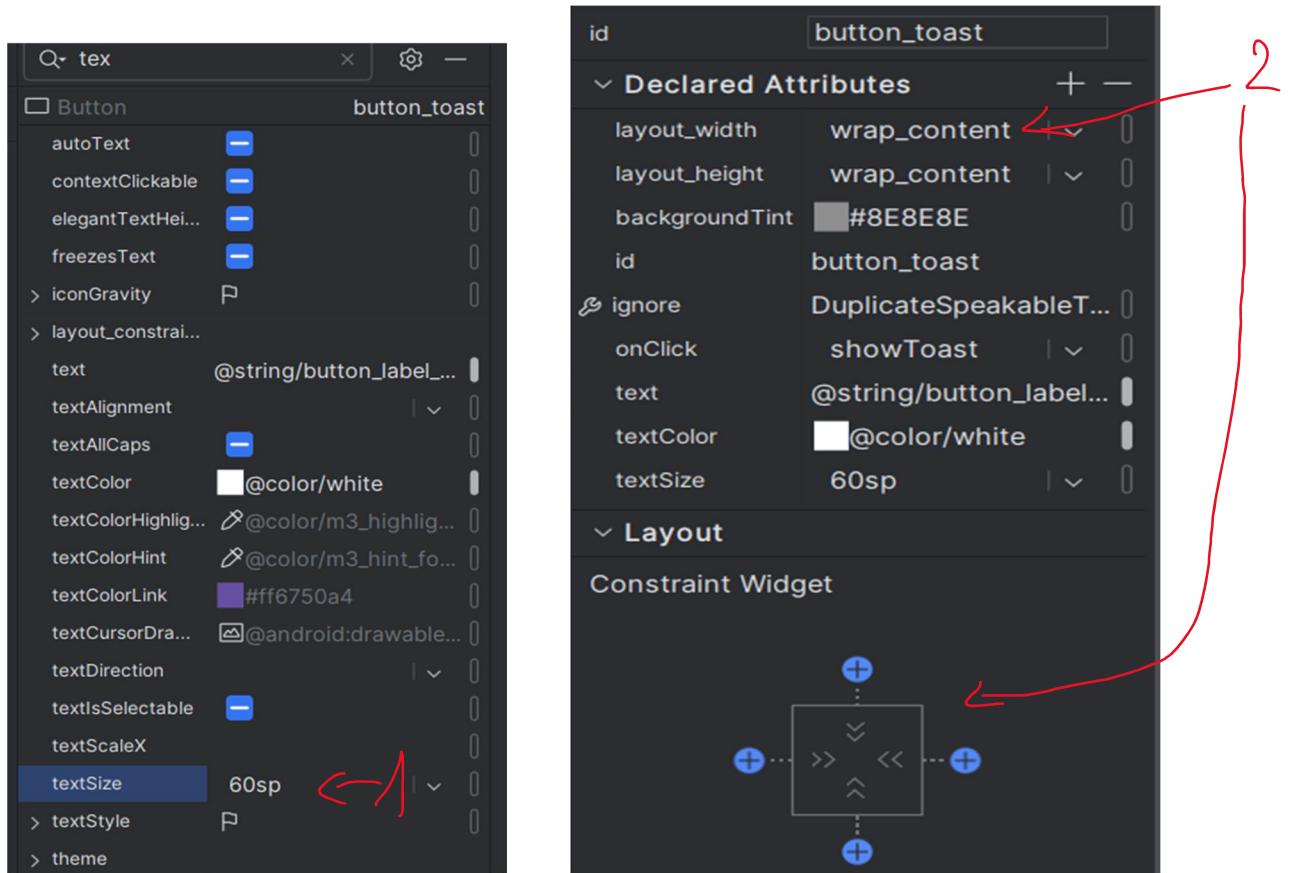
Bạn có thể sử dụng bảng **Attributes** trong tab **Design** để thay đổi các thuộc tính của bố cục này. Hãy làm theo các bước sau:

1. Tắt công cụ Autoconnect trên thanh công cụ. Đảm bảo rằng công cụ này đã được vô hiệu hóa .
2. Xóa tất cả các ràng buộc trong bố cục bằng cách nhấp vào nút **Clear All Constraints**  trên thanh công cụ.
- Sau khi xóa ràng buộc, bạn có thể di chuyển và thay đổi kích thước các phần tử trong bố cục một cách tự do.
3. Trình chỉnh sửa bố cục cung cấp các nút điều chỉnh kích thước ở cả bốn góc của một phần tử để thay đổi kích thước của phần tử đó. Trong **Component Tree**, hãy chọn **TextView** có tên là `show_count`. Để **TextView** không cản trở bạn có thể tự do kéo các phần tử **Button**, hãy kéo một góc của phần tử đó để thay đổi kích thước, như được hiển thị trong hình động bên dưới.



Việc thay đổi kích thước phần tử sẽ đặt cứng các giá trị chiều rộng và chiều cao. Tránh đặt kích thước cố định cho hầu hết các phần tử, vì bạn không thể dự đoán cách chúng sẽ hiển thị trên các màn hình có kích thước và mật độ khác nhau. Hiện tại, bạn chỉ đang làm điều này để di chuyển phần tử ra khỏi vị trí ban đầu, và bạn sẽ thay đổi kích thước của nó ở bước tiếp theo.

- Chọn Nút button\_toast trong **Component Tree**, nhấp vào thẻ **Attributes** để mở ngăn **Thuộc tính** và thay đổi textSize thành **60sp** (#1 trong hình bên dưới) và layout\_width thành **wrap\_content** (#2 trong hình bên dưới).



Như được hiển thị ở phía bên phải của hình trên (2), bạn có thể nhấp vào bộ điều khiển chiều rộng trong view inspector, xuất hiện dưới dạng hai đoạn ở bên trái và bên phải của hình vuông, cho đến khi nó hiển thị Wrap Content. Ngoài ra, bạn cũng có thể chọn **wrap\_content** từ bảng chọn layout\_width.

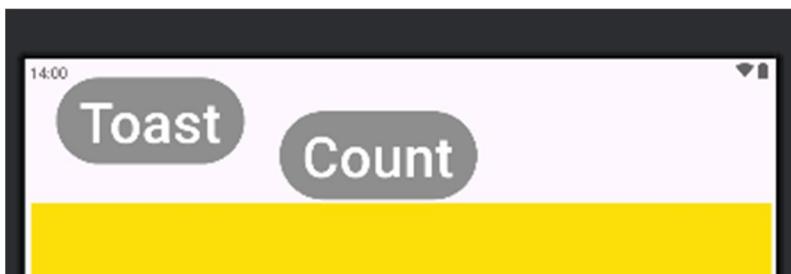
Bạn sử dụng **wrap\_content** để đảm bảo rằng nếu văn bản của nút được chuyển sang một ngôn ngữ khác, nút sẽ tự động mở rộng hoặc thu hẹp để phù hợp với từ ngữ trong ngôn ngữ đó.

- Chọn Button\_count trong **Component Tree**, thay đổi textSize thành **60sp** và layout\_width thành **wrap\_content**, rồi kéo Button lên trên TextView đến một khoảng trống trong layout.

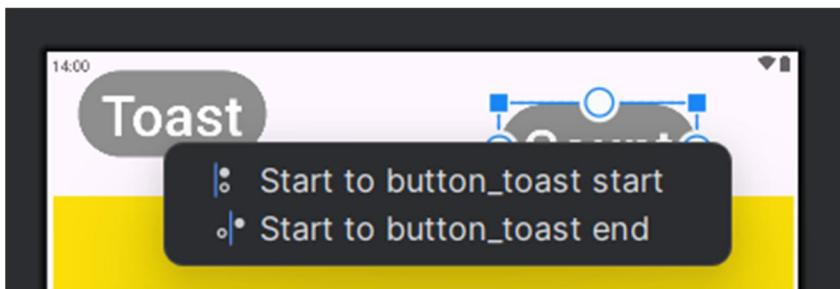
## 1.7 Sử dụng ràng buộc đường cơ sở

Bạn có thể căn chỉnh một phần tử giao diện chứa văn bản, chẳng hạn như **TextView** hoặc **Button**, với một phần tử khác có văn bản. Ràng buộc Baseline Constraint giúp bạn căn chỉnh các phần tử sao cho đường cơ sở của văn bản trùng khớp nhau.

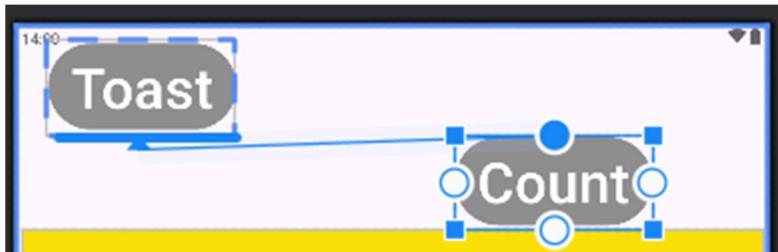
- Ràng buộc nút button\_toast ở phía trên và bên trái của bố cục, kéo nút button\_count đến khoảng trống gần nút button\_toast và ràng buộc nút button\_count ở phía bên phải của nút button\_toast, như thể hiện trong hình động:



- Sử dụng ràng buộc đường cơ sở, bạn có thể ràng buộc button\_count sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của nút Button\_toast. Chọn phần tử button\_count, sau đó di con trỏ qua phần tử cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.



- Nhấp vào nút ràng buộc đường cơ sở. Tay cầm đường cơ sở xuất hiện, nhấp nháy màu xanh lá cây như được hiển thị trong hình động. Nhấp và kéo một đường ràng buộc đường cơ sở đến đường cơ sở của phần tử button\_toast.

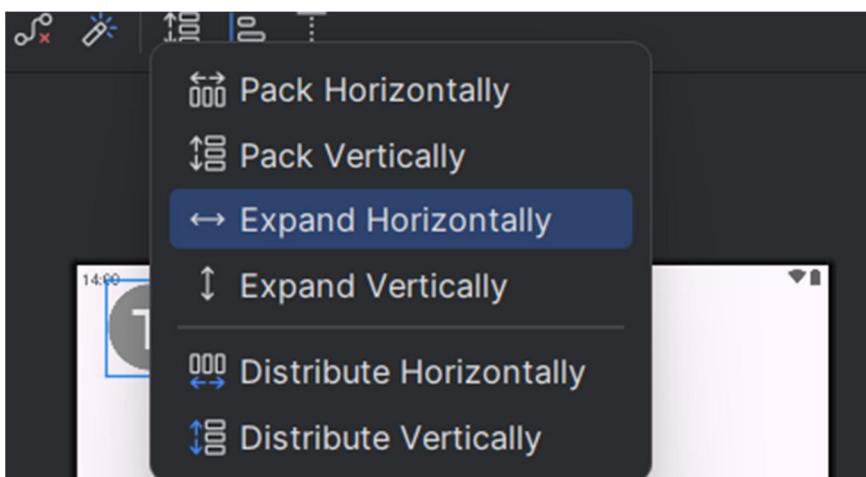


## 1.8 Mở rộng các nút theo chiều ngang



Nút đóng gói  trên thanh công cụ cung cấp các tùy chọn để sắp xếp hoặc mở rộng các phần tử giao diện người dùng được chọn. Bạn có thể sử dụng nó để sắp xếp đều các nút Button theo chiều ngang trên bố cục.

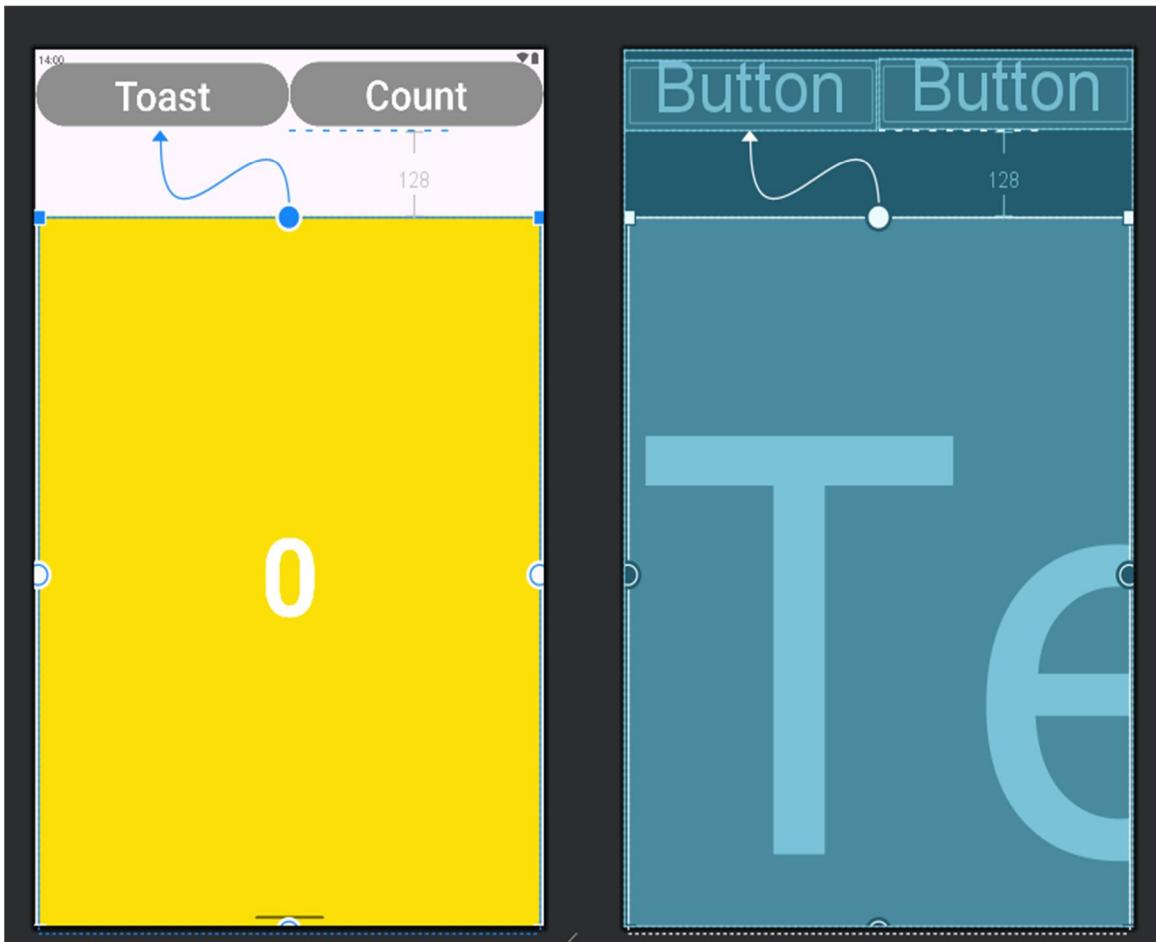
1. Chọn **button\_count** trong **Component Tree**, sau đó nhấn **Shift** và chọn **button\_toast** để chọn cả hai nút.
2. Nhấp vào nút đóng gói  trên thanh công cụ và chọn **Expand Horizontally**, như minh họa trong hình bên dưới.



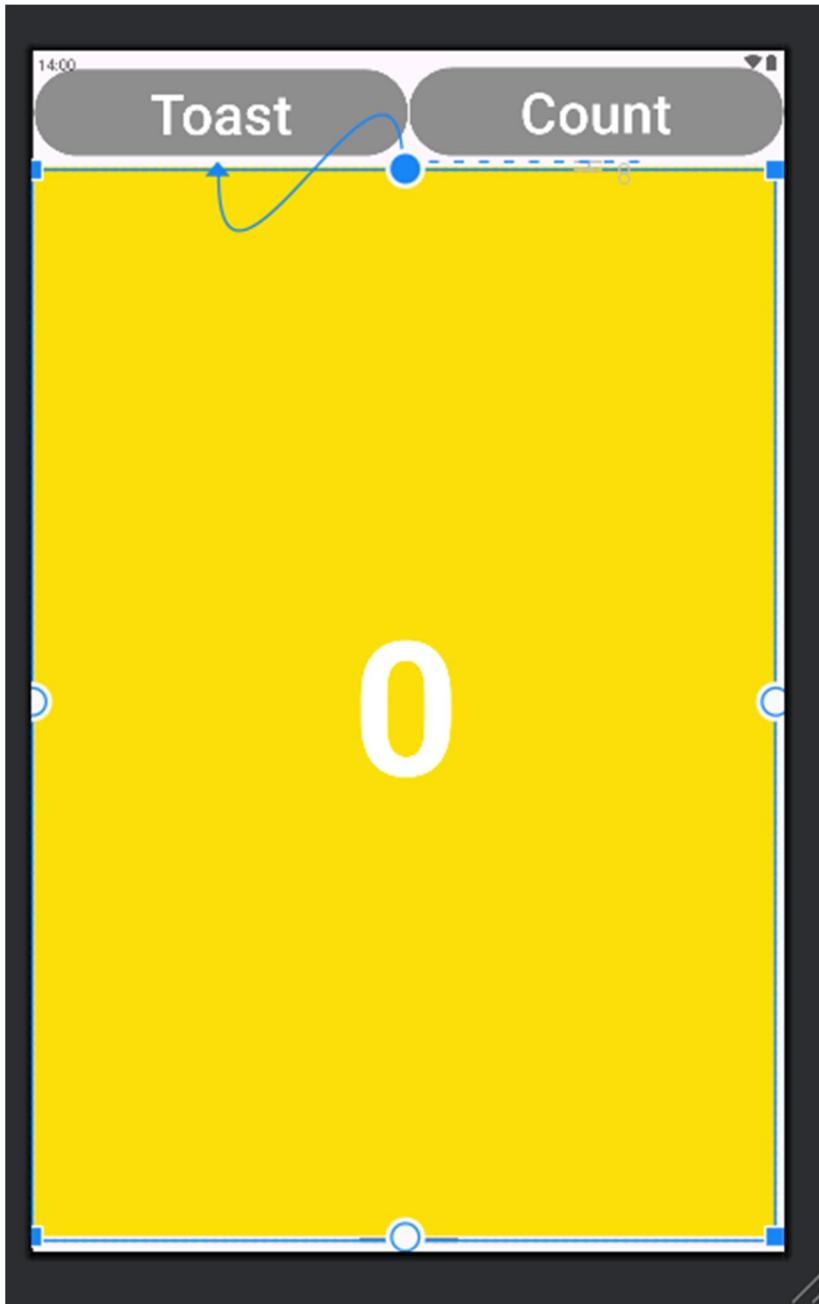
Các thành phần Nút mở rộng theo chiều ngang để lấp đầy bố cục như hiển thị bên dưới.



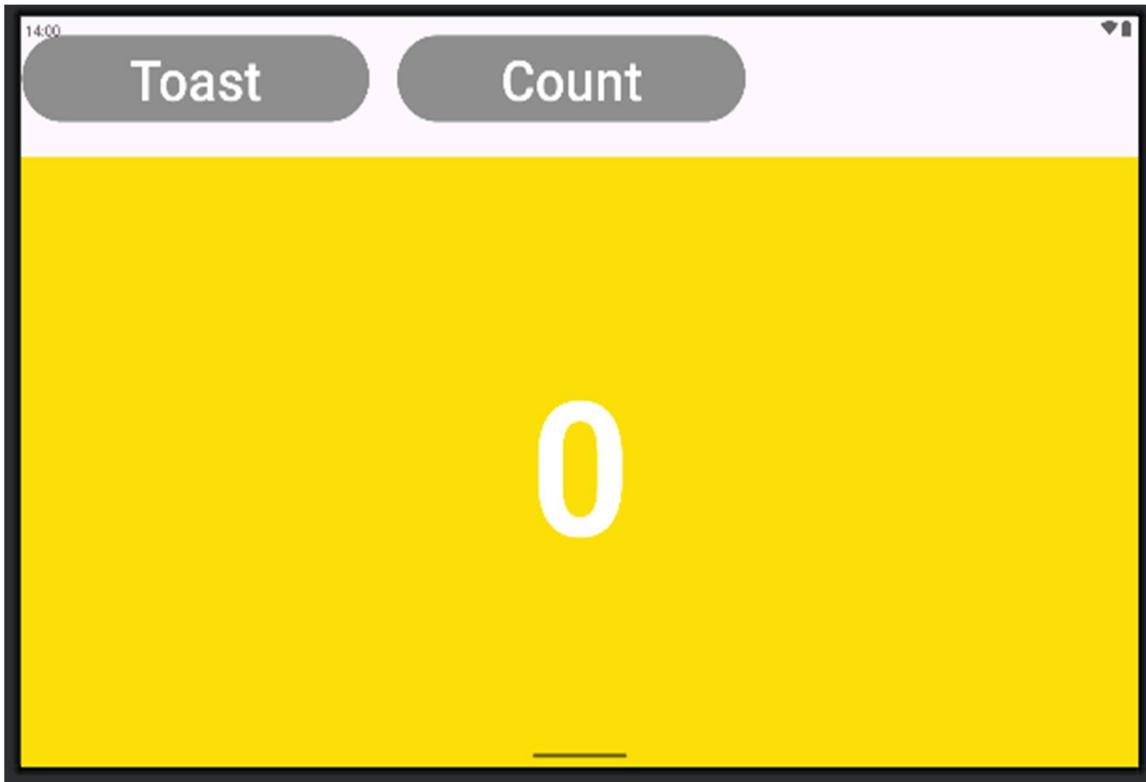
3. Để hoàn thành bố cục, hãy thêm ràng buộc cho TextView `show_count` để nằm dưới của button\_toast và các cạnh và đáy của bố cục, sau khi hoàn tất, `show_count` sẽ được đặt đúng vị trí như trong hình minh họa động bên dưới.



4. Bước cuối cùng là thay đổi `layout_width` và `layout_height` của TextView `show_count` thành **Match Constraints** và `textSize` thành **200sp**. Bố cục cuối cùng trông giống như hình bên dưới.



5. Nhấp vào nút **Orientation in Editor**  ở thanh công cụ trên cùng và chọn **Switch to Landscape**. Bố cục máy tính bảng xuất hiện theo hướng ngang như minh họa bên dưới. (Bạn có thể chọn **Switch to Portrait** để trở về hướng dọc).



- Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng sau khi chạy ứng dụng để xem ứng dụng trông như thế nào trên các loại thiết bị khác nhau. Bạn đã tạo thành công một ứng dụng có thể chạy với giao diện người dùng phù hợp trên điện thoại và máy tính bảng có kích thước và mật độ màn hình khác nhau.

**Mẹo:** Để biết hướng dẫn chi tiết về cách sử dụng ConstraintLayout, hãy xem [Sử dụng ConstraintLayout](#) để thiết kế chế độ xem của bạn.

### **Mã giải pháp nhiệm vụ 1**

Dự án Android Studio: [HelloToast](#)

## Nhiệm vụ 2: Thay đổi bố cục thành LinearLayout

[LinearLayout](#) là một ViewGroup sắp xếp tập hợp các View theo hàng ngang hoặc hàng dọc. LinearLayout là một trong những bố cục phổ biến nhất vì nó đơn giản và nhanh chóng. Nó thường được sử dụng bên trong một ViewGroup khác để sắp xếp các phần tử giao diện người dùng theo chiều ngang hoặc chiều dọc.

Một LinearLayout bắt buộc phải có các thuộc tính sau:

- `layout_width`
- `layout_height`
- `orientation`

Hai thuộc tính `layout_width` và `layout_height` có thể nhận một trong các giá trị sau:

- **match\_parent**: `match_parent` : Mở rộng chế độ xem để lấp đầy chế độ xem cha theo chiều rộng hoặc chiều cao. Khi LinearLayout là chế độ xem gốc, chế độ xem này sẽ mở rộng theo kích thước của màn hình (chế độ xem cha).
- **wrap\_content**: Thu nhỏ kích thước chế độ xem để chế độ xem chỉ đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, chế độ xem sẽ trở nên vô hình.
- Số dp cố định (pixel không phụ thuộc vào mật độ): Chỉ định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ: `16dp` nghĩa là 16 pixel không phụ thuộc vào mật độ.

Các giá trị của `orientation`:

- **horizontal**: Sắp xếp các phần tử từ trái sang phải.
- **vertical**: Sắp xếp các phần tử từ trên xuống dưới.

Trong nhiệm vụ này bạn sẽ thay đổi từ chế độ xem gốc `ConstraintLayout` cho ứng dụng Hello Toast thành `LinearLayout` để bạn có thể thực hành sử dụng `LinearLayout`.

### 2.1 Thay đổi nhóm View gốc thành LinearLayout

1. Mở ứng dụng Hello Toast từ bài trước.
2. Mở tệp `activity_main.xml` (nếu chưa mở), sau đó nhấn vào thẻ **Text** ở cuối khung chỉnh sửa để xem mã XML. Ở đầu mã XML, bạn sẽ thấy dòng thẻ sau:

```
<androidx.constraintlayout.widget.ConstraintLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Thay đổi `<androidx.constraintlayout.widget.ConstraintLayout` thành `<LinearLayout` lúc này đoạn mã sẽ trông như thế này:

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

4. Đảm bảo rằng thẻ đóng ở cuối mã đã thay đổi thành `</LinearLayout>` (Android Studio sẽ tự động thay đổi thẻ đóng nếu bạn thay đổi thẻ mở). Nếu nó không tự động thay đổi, hãy chỉnh sửa thủ công.  
5. Dưới dòng thẻ `<LinearLayout`, thêm thuộc tính sau ngay sau thuộc tính `android:layout_height`:

```
 android:orientation="vertical"
```

Sau khi thực hiện các thay đổi này, một số thuộc tính XML của các phần tử khác có thể bị gạch chân màu đỏ vì chúng được sử dụng với `ConstraintLayout` và không phù hợp với `LinearLayout`.

## 2.2 Thay đổi thuộc tính phần tử cho `LinearLayout`

Hãy làm theo các bước sau để thay đổi thuộc tính của các phần tử giao diện sao cho phù hợp với `LinearLayout`:

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp **activity\_main.xml** (nếu chưa mở), sau đó nhấn vào thẻ **Text**
3. Tìm phần tử nút `button_toast`, sau đó thay đổi thuộc tính sau:

| Nguyên bản                              | Đổi thành                                        |
|-----------------------------------------|--------------------------------------------------|
| <code>android:layout_width="0dp"</code> | <code>android:layout_width="match_parent"</code> |

4. Xóa các thuộc tính sau khỏi phần tử `button_toast`:

```
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
```

```
<Button
 android:id="@+id/button_toast"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:onClick="showToast"
 android:text="@string/button_label_toast"
 android:textColor="@color/white"
 app:layout_constraintEnd_toEndOf="parent"-->
 app:layout_constraintStart_toStartOf="parent"-->
 app:layout_constraintTop_toTopOf="parent"-->
```

5. Tìm phần tử nút button\_count, và sau đó thay đổi thuộc tính sau:

| Nguyên bản                 | Đổi thành                           |
|----------------------------|-------------------------------------|
| android:layout_width="0dp" | android:layout_width="match_parent" |

```
<Button
 android:id="@+id/button_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
```

6. Xóa các thuộc tính sau khỏi phần tử button\_count:

```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
```

```
<Button
 android:id="@+id/button_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:onClick="countUp"
 android:text="Count"
 android:textColor="@color/white"

 tools:ignore="HardcodedText,VisualLintButtonSize,TextContrastCheck" />
 app:layout_constraintBottom_toBottomOf="parent"-->
 app:layout_constraintEnd_toEndOf="parent"-->
 app:layout_constraintStart_toStartOf="parent"-->
```

7. Tìm phần tử TextView show\_count, và thay đổi các thuộc tính sau:

| Nguyên bản                  | Đổi thành                            |
|-----------------------------|--------------------------------------|
| android:layout_width="0dp"  | android:layout_width="match_parent"  |
| android:layout_height="0dp" | android:layout_height="wrap_content" |

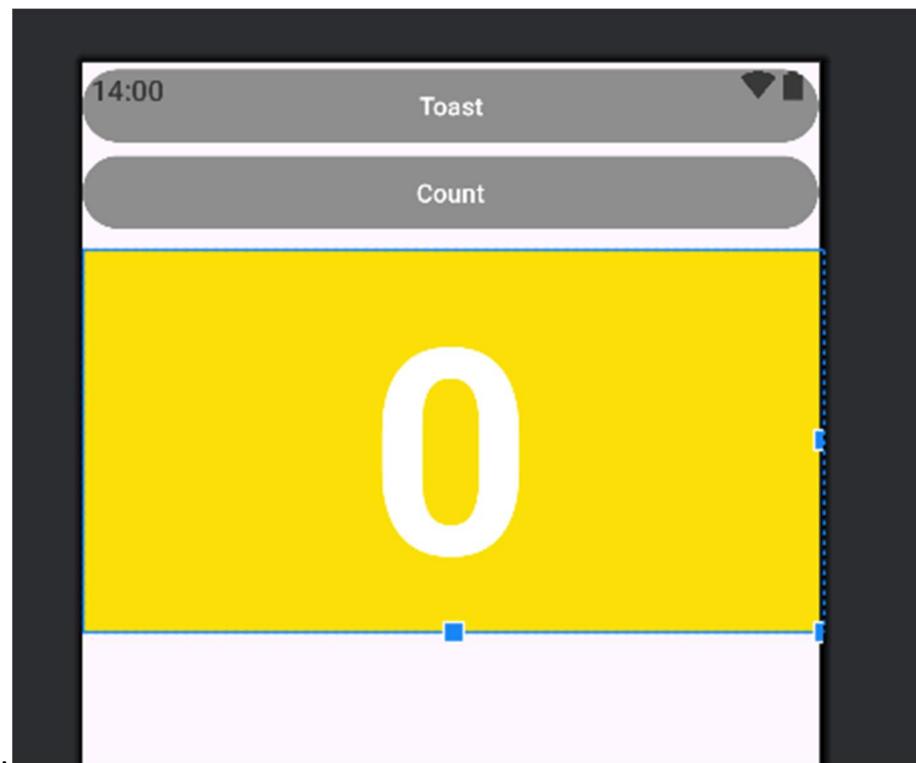
```
<Button
 android:id="@+id/button_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:background="#000000" style="color:#000000"/>
```

8. Xóa các thuộc tính sau khỏi phần tử show\_count:

```
app:layout_constraintBottom_toTopOf="@+id/button_count"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/button_toast"
```

```
app:layout_constraintBottom_toTopOf="@+id/button_count"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.498"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/button_toast"
```

9. Chọn thẻ **Preview** ở bên phải cửa sổ Android Studio (nếu chưa chọn) để xem bản xem trước của bố cục cho đến thời điểm này:



10.

### 2.3 Thay đổi vị trí các phần tử trong LinearLayout

`LinearLayout` sắp xếp các phần tử theo hàng ngang hoặc dọc. Bạn đã thêm thuộc tính `android:orientation="vertical"`, vì vậy các phần tử sẽ được xếp chồng lên nhau theo chiều dọc như trong hình trước đó.

Để thay đổi vị trí sao cho nút **Count** nằm ở phía dưới, hãy làm theo các bước sau:

1. Mở ứng dụng Hello Toast từ bài trước.
2. Mở tệp **activity\_main.xml** (nếu chưa mở), sau đó nhấn vào thẻ **Text** để chỉnh sửa XML.
3. Chọn nút `button_count` và tất cả các thuộc tính của nó, từ thẻ `<Button` cho đến và bao gồm thẻ đóng `>`, và chọn **Edit > Cut**.
4. Nhấp vào sau thẻ đóng `>` của phần tử `TextView` nhưng trước thẻ đóng `</LinearLayout>`, và chọn **Edit > Paste**.
5. (Tùy chọn) Để sửa bất kỳ vấn đề thụt lề hoặc khoảng cách nào vì mục đích thẩm mỹ, hãy chọn **Code > Reformat Code** để định dạng lại mã XML với khoảng cách và thụt lề phù hợp.

Mã XML cho các phần tử UI bây giờ sẽ trong như bên dưới:

```
<Button
 android:id="@+id/button_toast"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:onClick="showToast"
 android:text="Toast"
 android:textColor="@color/white"

 tools:ignore="DuplicateSpeakableTextCheck,HardcodedText,VisualLintButtonSize,TextContrastCheck,DuplicateClickableBoundsCheck,VisualLintOverlap" />

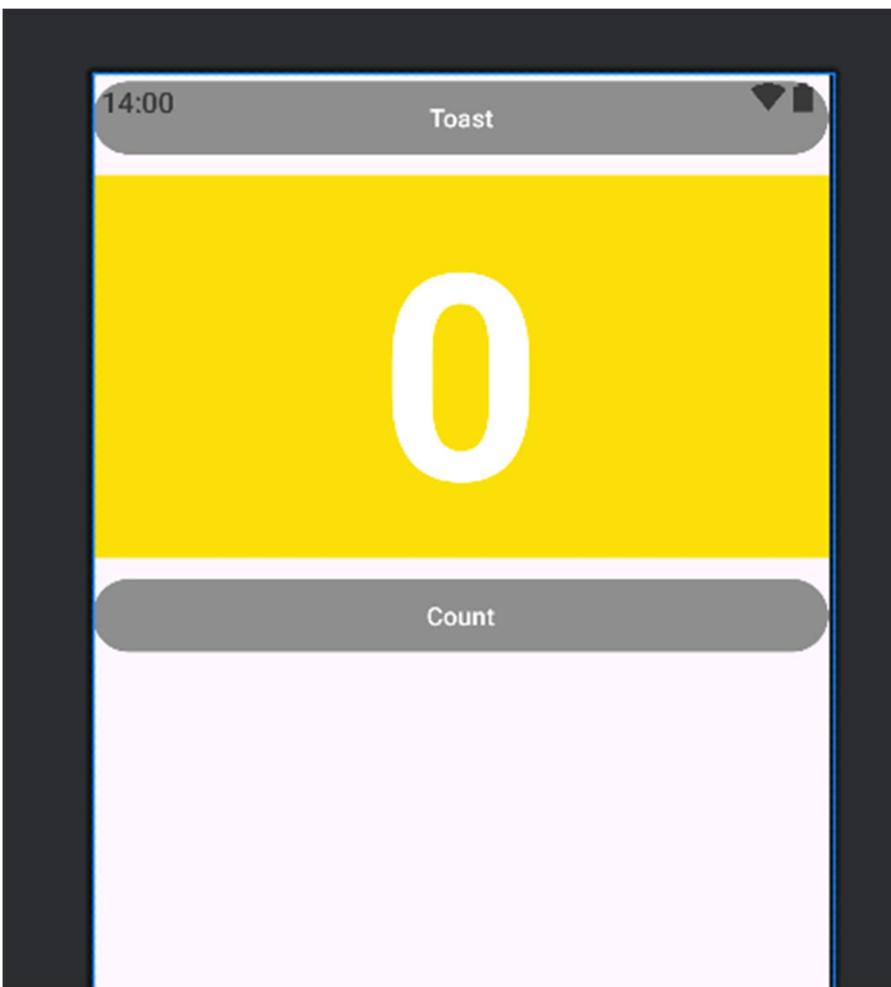
<TextView
 android:id="@+id/show_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginTop="8dp"
 android:layout_marginBottom="8dp"
 android:background="#FBE008"
 android:gravity="center"
 android:text="0"
 android:textAlignment="center"
 android:textColor="@color/design_default_color_on_primary"
 android:textSize="160sp"
 android:textStyle="bold"
 app:layout_constraintHorizontal_bias="0.498"

 app:layout_constraintVertical_bias="0.111"
 tools:ignore="HardcodedText,TextContrastCheck" />
```

```
<Button
 android:id="@+id/button_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:onClick="countUp"
 android:text="Count"
 android:textColor="@color/white"

 tools:ignore="HardcodedText,VisualLintButtonSize,TextContrastCheck" />
```

Bằng cách di chuyển nút button\_count bên dưới TextView, bố cục hiện gần giống với những gì bạn đã có trước đó, với nút **Count** ở phía dưới. Bản xem trước của bố cục hiện trông như sau:



## 2.4 Thêm độ rộng cho phần tử TextView

Việc chỉ định các thuộc tính gravity và weight giúp bạn kiểm soát tốt hơn cách sắp xếp các View và nội dung trong LinearLayout.

Thuộc tính android:gravity xác định căn chỉnh nội dung bên trong một **View**. Trong bài học trước, bạn đã thiết lập thuộc tính này cho TextView show\_count để căn giữa nội dung (chữ số 0) bên trong TextView.

```
android:gravity="center_vertical"
```

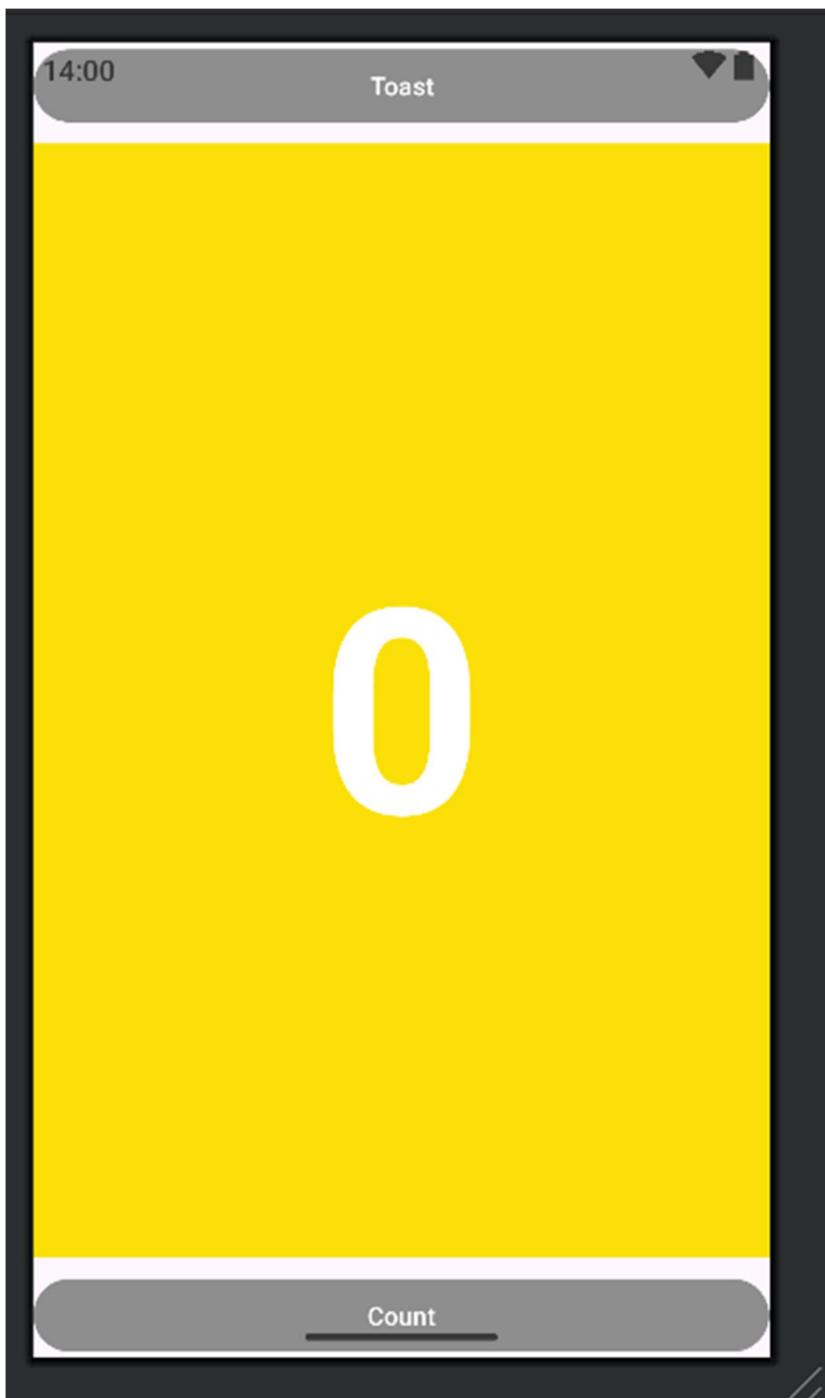
Thuộc tính android:layout\_weight chỉ ra lượng không gian bổ sung trong LinearLayout sẽ được phân bổ cho View. Nếu chỉ có một View có thuộc tính này, View sẽ nhận được toàn bộ không gian màn hình bổ sung. Đối với nhiều phần tử View, không gian sẽ được tính theo tỷ lệ. Ví dụ: nếu mỗi phần tử Button có trọng số là 1 và TextView là 2, tổng cộng là 4, thì mỗi phần tử Button sẽ nhận được  $\frac{1}{4}$  không gian và TextView sẽ nhận được một nửa.

Trên các thiết bị khác nhau, bố cục có thể hiển thị phần tử TextView show\_count lấp đầy một phần hoặc hầu hết không gian giữa các nút **Toast** và **Count**. Để mở rộng TextView để lấp đầy không gian khả dụng bất kể sử dụng thiết bị nào, hãy chỉ định thuộc tính android:gravity cho TextView. Thực hiện theo các bước sau:

1. Mở ứng dụng Hello Toast từ tác vụ trước đó.
2. Mở tệp bố cục activity\_main.xml (nếu tệp này chưa được mở) và nhấp vào tab Text.
3. Tìm phần tử TextView show\_count và thêm thuộc tính sau:

android:layout\_weight="1"

Bản xem trước bây giờ trông giống như hình sau.



Phần tử `TextView` `show_count` chiếm hết không gian giữa các nút. Bạn có thể xem trước bối cảnh cho các thiết bị khác nhau, như bạn đã làm trong tác vụ trước đó bằng cách nhấp vào nút `Device in Editor` ở thanh công cụ trên cùng của ngăn xem trước và chọn một thiết bị khác. Bất kể bạn chọn thiết bị nào để xem trước, phần tử `TextView` `show_count` sẽ chiếm hết không gian giữa các nút.

## Mã nhiệm vụ 2

Mã XML ở trong activity\_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 app:layoutDescription="@xml/activity_main_scene"
 tools:context=".MainActivity">

 <Button
 android:id="@+id/button_toast"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:onClick="showToast"
 android:text="@string/button_label_toast"
 android:textColor="@color/white"

 tools:ignore="DuplicateSpeakableTextCheck,HardcodedText,VisualLintButtonSize
 ,TextContrastCheck,DuplicateClickableBoundsCheck,VisualLintOverlap" />
 <!-- app:layout_constraintEnd_toEndOf="parent"-->
 <!-- app:layout_constraintStart_toStartOf="parent"-->
 <!-- app:layout_constraintTop_toTopOf="parent"-->

 <TextView
 android:id="@+id/show_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginTop="8dp"
 android:layout_marginBottom="8dp"
 android:background="#FBE008"
 android:gravity="center"
 android:text="@string/count_initial_value"
 android:textAlignment="center"
 android:textColor="@color/design_default_color_on_primary"
 android:textSize="160sp"
 android:textStyle="bold"
 app:layout_constraintHorizontal_bias="0.498"
 android:layout_weight="1"

 app:layout_constraintVertical_bias="0.111"
 tools:ignore="HardcodedText,TextContrastCheck" />
 <!-- app:layout_constraintBottom_toTopOf="@+id/button_count"-->
 <!-- app:layout_constraintEnd_toEndOf="parent"-->
 <!-- app:layout_constraintStart_toStartOf="parent"-->
 <!-- app:layout_constraintTop_toBottomOf="@+id/button_toast"-->
```

```
<Button
 android:id="@+id/button_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:onClick="countUp"
 android:text="@string/button_label_count"
 android:textColor="@color/white"

 tools:ignore="HardcodedText,VisualLintButtonSize,TextContrastCheck"
/>
 <!-- app:layout_constraintBottom_toBottomOf="parent"-->
 <!-- app:layout_constraintEnd_toEndOf="parent"-->
 <!-- app:layout_constraintStart_toStartOf="parent"-->

</LinearLayout>
```

## Nhiệm vụ 3: Thay đổi bố cục thành RelativeLayout

**RelativeLayout** là một nhóm chế độ xem trong đó mỗi chế độ xem được định vị và căn chỉnh theo các chế độ xem khác trong nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục bằng **RelativeLayout**.

### 3.1 Thay đổi LinearLayout thành RelativeLayout

Một cách dễ dàng để thay đổi **LinearLayout** thành **RelativeLayout** là thêm các thuộc tính XML trong thẻ **Text**.

1. Mở tệp **activity\_main.xml**, sau đó nhấn vào thẻ **Text** ở cuối khung chỉnh sửa để xem mã XML.
2. Thay đổi **<LinearLayout** ở đầu tệp thành **<RelativeLayout** để dòng lệnh trông như sau:

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

3. Cuộn xuống để đảm bảo rằng thẻ đóng **</LinearLayout>** cũng đã được thay đổi thành **</RelativeLayout>**. Nếu nó chưa thay đổi, hãy chỉnh sửa thủ công.

### 3.2 Sắp xếp lại các View trong RelativeLayout

Một cách dễ dàng để sắp xếp và định vị các view ở trong **RelativeLayout** là thêm các thuộc tính XML trong thẻ **Text**.

1. Nhấp vào tab **Preview** ở bên cạnh trình chỉnh sửa (nếu chưa được chọn) để xem bản xem trước bố cục, bây giờ sẽ trông giống như hình minh họa bên dưới.



Với thay đổi sang **RelativeLayout**, trình chỉnh sửa bố cục cũng đã thay đổi một số thuộc tính của các view. Ví dụ:

- Nút **Count** (`button_count`) chồng lên nút **Toast** (`button_toast`), khiến bạn không nhìn thấy nút **Toast**.
  - Phần trên của **TextView** (`show_count`) chồng lên các nút **Button**.
2. Thêm thuộc tính **android:layout\_below** vào nút **button\_count** để đặt nút này ngay bên dưới **TextView** `show_count`. Đây là một trong những thuộc tính quan trọng của **RelativeLayout** - giúp bạn định vị **View** dựa trên vị trí của các **View** khác.

```
 android:layout_below="@+id/show_count"
```

3. Thêm thuộc tính `android:layout_centerHorizontal` vào cùng **Button** để căn giữa nút này theo chiều ngang với cha của nó, vì **RelativeLayout** là **ViewGroup** cha trong trường hợp này.

```
 android:layout_centerHorizontal="true"
```

Mã XML đầy đủ cho Nút button\_count như sau:

```
<Button
 android:id="@+id/button_count"
 android:layout_below="@+id/show_count"
 android:layout_centerHorizontal="true"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:onClick="countUp"
 android:text="Count"
 android:textColor="@color/white"

 tools:ignore="HardcodedText,VisualLintButtonSize,TextContrastCheck" />
```

4. Thêm thuộc tính như sau vào `TextView` show\_count:

```
 android:layout_below="@+id/button_toast"
 android:layout_alignParentLeft="true"
 android:layout_alignParentStart="true"
```

`android:layout_alignParentLeft` căn chỉnh chế độ xem sang phía trái của nhóm chế độ xem cha `RelativeLayout`. Trong khi thuộc tính này tự nó đủ để căn chỉnh chế độ xem sang phía trái, nhưng bạn có thể muốn chế độ xem căn chỉnh sang phía bên phải nếu ứng dụng đang chạy trên thiết bị sử dụng ngôn ngữ từ phải sang trái. Do đó, thuộc tính `android:layout_alignParentStart` làm cho cạnh "bắt đầu" của chế độ xem này khớp với cạnh bắt đầu của cha. Điểm bắt đầu là cạnh trái của màn hình nếu tùy chọn là từ trái sang phải hoặc là cạnh phải của màn hình nếu tùy chọn là từ phải sang trái.

5. Xóa thuộc tính `android:layout_weight="1"` từ `TextView show_count` , thuộc tính này không liên quan đến `RelativeLayout` . Bản xem trước bố cục hiện trông giống như hình sau:



**Mẹo:** RelativeLayout giúp bạn dễ dàng sắp xếp các phần tử giao diện người dùng trong bố cục một cách nhanh chóng. Để tìm hiểu thêm về cách định vị View trong RelativeLayout, hãy xem phần "Positioning Views" trong chủ đề "Relative Layout" của API Guide.

### Mã nhiệm vụ 3

Mã XML ở trong activity\_main.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 app:layoutDescription="@xml/activity_main_scene"
 tools:context=".MainActivity">

 <Button
 android:id="@+id/button_toast"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:onClick="showToast"
 android:text="Toast"
 android:textColor="@color/white"

 tools:ignore="DuplicateSpeakableTextCheck,HardcodedText,VisualLintButtonSize,TextContrastCheck,DuplicateClickableBoundsCheck,VisualLintOverlap" />

 <TextView
 android:id="@+id/show_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginTop="8dp"
 android:layout_marginBottom="8dp"
 android:background="#FBE008"
 android:gravity="center"
 android:text="0"
```

```

<TextView
 android:text="0"
 android:textAlignment="center"
 android:textColor="@color/design_default_color_on_primary"
 android:textSize="160sp"
 android:textStyle="bold"
 app:layout_constraintHorizontal_bias="0.498"

 android:layout_below="@+id/button_toast"
 android:layout_alignParentLeft="true"
 android:layout_alignParentStart="true"

 app:layout_constraintVertical_bias="0.111"
 tools:ignore="HardcodedText,TextContrastCheck" />

<Button
 android:id="@+id/button_count"
 android:layout_below="@+id/show_count"
 android:layout_centerHorizontal="true"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:backgroundTint="#8E8E8E"
 android:onClick="countUp"
 android:text="Count"
 android:textColor="@color/white"

 tools:ignore="HardcodedText,VisualLintButtonSize,TextContrastCheck" />

```

## Tóm tắt

Sử dụng trình chỉnh sửa bố cục để xem trước và tạo biến thể:

- Để xem trước bố cục ứng dụng ở **chế độ ngang** trong trình chỉnh sửa bố cục, nhấp vào nút **Orientation in Editor**  trên thanh công cụ và chọn **Switch to Landscape**. Chọn **Switch to Portrait** để quay lại chế độ dọc.
- Để tạo **biến thể bố cục** khác cho **chế độ ngang**, nhấp vào nút **Orientation in Editor** và chọn **Create Landscape Variation**. Một cửa sổ trình chỉnh sửa mới sẽ mở ra với tab **land/activity\_main.xml**, hiển thị bố cục cho chế độ ngang.
- Để xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị hoặc trình giả lập, nhấp vào nút **Device in Editor**  Pixel  trên thanh công cụ và chọn một thiết bị.
- Để tạo **biến thể bố cục** khác dành cho **máy tính bảng (màn hình lớn)**, nhấp vào nút **Orientation in Editor** và chọn **Create layout x-large Variation**. Một cửa sổ trình chỉnh sửa mới sẽ mở ra **xlarge/activity\_main.xml** để hiển thị bố cục phù hợp với kích thước máy tính bảng.

## Sử dụng ConstraintLayout:

- Để xóa tất cả ràng buộc trong bố cục có **ConstraintLayout** làm gốc, nhấn vào nút  **Clear All Constraints** trên thanh công cụ.
- Bạn có thể căn chỉnh một phần tử UI chứa văn bản (chẳng hạn như TextView hoặc Button) với một phần tử UI khác chứa văn bản. *Baseline constraint* cho phép bạn ràng buộc các phần tử để các dòng cơ sở của văn bản khớp nhau.
- Để tạo baseline constraint, di chuột qua phần tử UI cho đến khi nút ràng buộc dòng cơ sở  xuất hiện bên dưới phần tử.
- Nút đóng gói  trên thanh công cụ cung cấp các tùy chọn để đóng gói hoặc mở rộng các phần tử UI đã chọn. Bạn có thể sử dụng nút này để sắp xếp các nút một cách đều nhau theo chiều ngang trong bố cục.

## Sử dụng LinearLayout:

- LinearLayout** là một **ViewGroup** sắp xếp các phần tử con theo hàng ngang hoặc dọc.
- Một **LinearLayout** bắt buộc phải có các thuộc tính `layout_width`, `layout_height`, và `orientation`.
- `match_parent` cho `layout_width` hoặc `layout_height`: Mở rộng View để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao. Khi **LinearLayout** là View gốc, nó sẽ mở rộng theo kích thước của màn hình (View cha).
- `wrap_content` cho `layout_width` hoặc `layout_height`: Thu nhỏ kích thước để View vừa với nội dung bên trong. Nếu không có nội dung, View trở nên vô hình.
- Kích thước cố định theo dp (**density-independent pixels**) cho `layout_width` hoặc `layout_height`: Xác định một kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ: **16dp** có nghĩa là **16 pixel** độc lập với mật độ.
- `orientation` của **LinearLayout** có thể là: horizontal sắp xếp các phần tử từ trái sang phải, hoặc vertical sắp xếp các phần tử từ trên xuống dưới.
- Chỉ định các thuộc tính `gravity` và `weight` cung cấp cho bạn quyền kiểm soát bổ sung đối với việc sắp xếp các chế độ xem và nội dung trong **LinearLayout**.

- Thuộc tính android:gravity chỉ định căn chỉnh nội dung của một View trong chính View đó.
- Thuộc tính android:layout\_weight chỉ ra lượng không gian bổ sung trong LinearLayout sẽ được phân bổ cho Chế độ xem. Nếu chỉ có một Chế độ xem có thuộc tính này, thì nó sẽ nhận được toàn bộ không gian màn hình bổ sung. Đối với nhiều phần tử Chế độ xem, không gian được chia theo tỷ lệ. Ví dụ: nếu hai phần tử Nút mỗi phần tử có trọng số là 1 và một TextView là 2, tổng cộng là 4, thì các phần tử Nút mỗi phần tử sẽ nhận được  $\frac{1}{4}$  không gian và TextView nhận được một nửa.

Sử dụng RelativeLayout:

- RelativeLayout là một ViewGroup trong đó mỗi chế độ xem được định vị và căn chỉnh theo các chế độ xem khác trong nhóm.
- Sử dụng android:layout\_alignParentTop để căn chỉnh Chế độ xem vào đầu của phần tử cha.
- Sử dụng android:layout\_alignParentLeft để căn chỉnh View về phía bên trái của phần tử cha.
- Sử dụng android:layout\_alignParentStart để làm cho cạnh bắt đầu của View khớp với cạnh bắt đầu của phần tử cha. Thuộc tính này hữu ích nếu bạn muốn ứng dụng của mình hoạt động trên các thiết bị sử dụng các tùy chọn ngôn ngữ hoặc địa phương khác nhau. Điểm bắt đầu là cạnh trái của màn hình nếu tùy chọn là từ trái sang phải hoặc là cạnh phải của màn hình nếu tùy chọn là từ phải sang trái.

## Các khái niệm liên quan

Tài liệu về khái niệm liên quan có trong 1.2: Bố cục và tài nguyên cho UI.

## Tìm hiểu thêm

Tài liệu Android Studio:

- [Làm quen với Android Studio](#)
- [Tạo biểu tượng ứng dụng bằng Image Asset Studio](#)

Tài liệu dành cho nhà phát triển Android:

- [Bố cục](#)
- [Xây dựng giao diện người dùng bằng Layout Editor](#)
- [Xây dựng giao diện người dùng phản hồi bằng ConstraintLayout](#)
- [Bố cục](#)
- [LinearLayout](#)
- [RelativeLayout](#)
- [View](#)
- [Button](#)
- [TextView](#)
- [Hỗ trợ nhiều mật độ điểm ảnh khác nhau](#)

Khác:

- [Codelab: Sử dụng ConstraintLayout để thiết kế chế độ xem Android của bạn](#)
- [Từ vựng và thuật ngữ thuật ngữ](#)

## Bài tập

### Thay đổi ứng dụng

Mở ứng dụng HelloToast.

1. Đổi tên dự án thành **HelloConstraint** và sắp xếp lại dự án thành Hello Constraint. (Để biết hướng dẫn về cách sao chép và sắp xếp lại dự án, hãy xem Phụ lục: Tiện ích.)
2. Sửa đổi bố cục activity\_main.xml để căn chỉnh các phần tử **Toast** và **Count** Button dọc bên trái của TextView show\_count hiển thị "0". Tham khảo các hình bên dưới để biết bố cục.
3. Bao gồm một nút thứ ba có tên là **Zero** xuất hiện giữa các phần tử nút **Toast** và **Count**.
4. Phân phối các phần tử Button theo chiều dọc giữa phần trên cùng và phần dưới cùng của TextView show\_count.
5. Đặt **Zero** nút ban đầu có nền **màu xám**.
6. Đảm bảo rằng bạn bao gồm Zero Button cho hướng ngang trong activity\_main.xml (land) và cho màn hình có kích thước máy tính bảng trong activity\_main (xlarge).

7. Làm cho nút **Zero** thay đổi giá trị trong TextView `show_count` thành 0.
8. Cập nhật trình xử lý nhấp chuột cho nút **Count** để nó thay đổi màu nền của chính nó, tùy thuộc vào việc số đếm mới là lẻ hay chẵn.

Gợi ý: **Không** sử dụng `findViewById` để tìm nút **Count**. Bạn có thể sử dụng cách nào khác không?

Hãy thoải mái sử dụng hằng số trong lớp Color cho hai màu nền khác nhau.

9. Cập nhật trình xử lý nhấp chuột cho nút **Count** để đặt màu nền cho nút **Zero** thành màu khác ngoài màu xám để hiển thị nút hiện đang hoạt động. Gợi ý: Bạn có thể sử dụng `findViewById` trong trường hợp này.
10. Cập nhật trình xử lý nhấp chuột cho nút **Zero** để đặt lại màu thành màu xám, sao cho nút có màu xám khi số đếm bằng không.

## Trả lời các câu hỏi sau

**Câu hỏi 1:** Hai thuộc tính ràng buộc nào trên nút **Zero** giúp căn chỉnh nó theo khoảng cách dọc đều giữa hai Button khác? (*Chọn 2 đáp án.*)

- `app:layout_constraintBottom_toTopOf="@+id/button_count"`
- `android:layout_marginBottom="8dp"`
- `android:layout_marginStart="16dp"`
- `app:layout_constraintTop_toBottomOf="@+id/button_toast"`
- `android:layout_marginTop="8dp"`

**Câu hỏi 2:** Thuộc tính ràng buộc nào giúp căn chỉnh nút **Zero** theo phương ngang để thẳng hàng với hai Button khác?

- `app:layout_constraintLeft_toLeftOf="parent"`
- `app:layout_constraintBottom_toTopOf="@+id/button_count"`
- `android:layout_marginBottom="8dp"`
- `app:layout_constraintTop_toBottomOf="@+id/button_toast"`

**Câu hỏi 3** Phương thức nào đúng khi sử dụng với thuộc tính `android:onClick` trong XML?

- `public void callMethod()`
- `public void callMethod(View view)`
- `private void callMethod(View view)`

- public boolean callMethod(View view)

Câu hỏi 4: Trình xử lý nhấp chuột cho nút **Count** bắt đầu bằng phương thức sau:

```
public void countUp(View view)
```

Kỹ thuật nào sau đây hiệu quả hơn khi sử dụng trong trình xử lý này để thay đổi màu nền của phần tử Button? Chọn một:

- Sử dụng `findViewById` để tìm nút **Count**. Gán kết quả cho một biến View, sau đó sử dụng `setBackgroundColor()`.
- Sử dụng tham số `view` được truyền cho trình xử lý nhấp chuột với `setBackgroundColor() : view.setBackgroundColor()`

## 1.4) Văn bản và các chế độ cuộn

### Giới thiệu

Lớp **TextView** là một lớp con của **View**, dùng để hiển thị văn bản trên màn hình. Bạn có thể kiểm soát cách văn bản xuất hiện bằng các thuộc tính **TextView** trong tệp XML giao diện. Phần thực hành này hướng dẫn cách làm việc với nhiều phần tử **TextView**, bao gồm cả một phần tử có thể cuộn nội dung theo chiều dọc.

Nếu có nhiều thông tin hơn mức có thể hiển thị trên màn hình thiết bị, bạn có thể tạo một chế độ cuộn để người dùng có thể **vuốt lên/xuống (dọc)** hoặc **vuốt trái/phải (ngang)** để xem nội dung.

Bạn thường sử dụng chế độ cuộn cho các bài báo, tin tức, hoặc bất kỳ đoạn văn bản dài nào không thể hiển thị hoàn toàn trên màn hình. Ngoài ra, bạn cũng có thể sử dụng chế độ cuộn để cho phép người dùng nhập nhiều dòng văn bản, hoặc để kết hợp các phần tử UI (chẳng hạn như trường nhập văn bản và nút bấm) trong một chế độ cuộn.

Lớp **ScrollView** cung cấp bố cục cho chế độ cuộn. ScrollView là một lớp con của **FrameLayout**. Chỉ đặt một phần tử view làm con bên trong nó—phần tử này chứa toàn bộ nội dung cần cuộn. Phần tử con này có thể là một **ViewGroup** (chẳng hạn như **LinearLayout**) chứa các thành phần UI.

Các bố cục phức tạp có thể gấp vấn đề về hiệu suất nếu chứa các phần tử con như hình ảnh. Một lựa chọn tốt cho một View bên trong ScrollView là **LinearLayout** với hướng dọc, trình bày các phần tử mà người dùng có thể cuộn qua (chẳng hạn như **TextView**).

Với ScrollView, tất cả các phần tử UI đều được lưu trong bộ nhớ và tồn tại trong cây giao diện, ngay cả khi chúng không được hiển thị trên màn hình. Điều này làm cho ScrollView trở thành lựa chọn lý tưởng để cuộn các trang văn bản tự do một cách mượt mà, vì văn bản đã có sẵn trong bộ nhớ. Tuy nhiên, ScrollView có thể tiêu tốn nhiều bộ nhớ, ảnh hưởng đến hiệu suất của ứng dụng. Để hiển thị danh sách dài mà người dùng có thể thêm, xóa, hoặc chỉnh sửa, hãy cân nhắc sử dụng [RecyclerView](#), được mô tả trong một bài học riêng.

## Những gì bạn nên biết

Bạn nên có khả năng:

- Tạo ứng dụng **Hello World** với **Android Studio**.
- Chạy ứng dụng trên **trình giả lập (emulator)** hoặc **thiết bị thực tế**.
- Triển khai một **TextView** trong bố cục của ứng dụng.
- Tạo và sử dụng **tài nguyên chuỗi (string resources)**.

## Những thứ bạn sẽ tìm hiểu

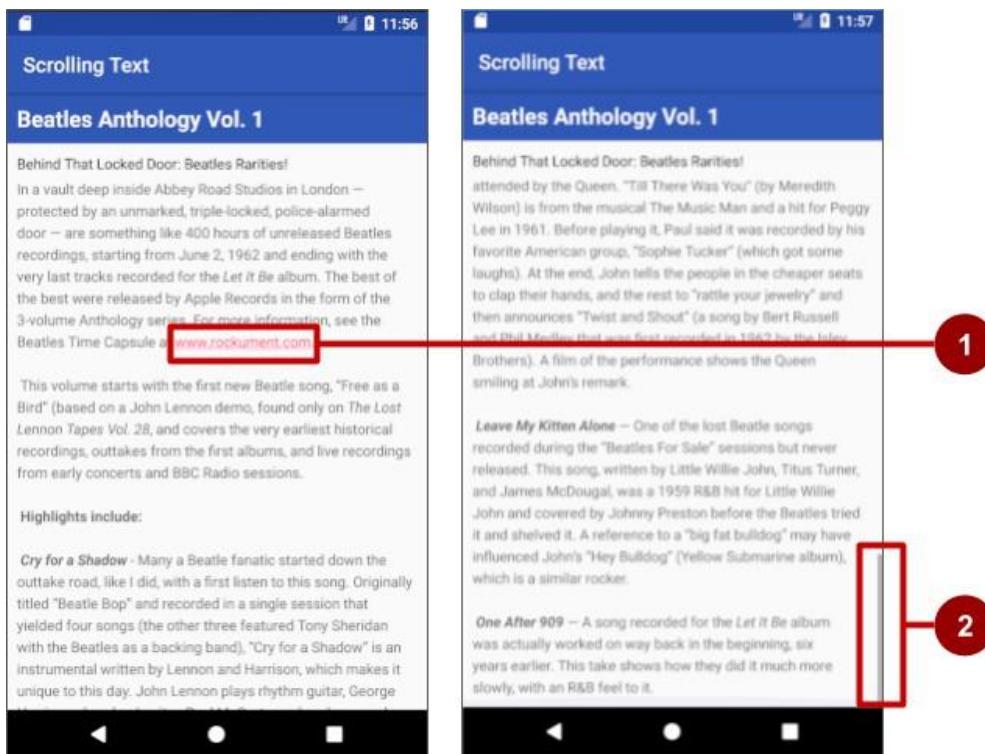
- Cách sử dụng mã XML để thêm nhiều phần tử TextView.
- Cách sử dụng mã XML để định nghĩa một View có thể cuộn.
- Cách hiển thị văn bản tự do (free-form text) với một số thẻ định dạng HTML.
- Cách tùy chỉnh màu nền và màu chữ của TextView.
- Cách chèn liên kết web vào văn bản.

## Những thứ bạn sẽ làm

- Tạo ứng dụng ScrollingText.
- Thay đổi ViewGroup ConstraintLayout thành RelativeLayout.
- Thêm hai phần tử TextView cho tiêu đề và tiêu đề phụ của bài viết.
- Sử dụng TextAppearance, styles, và màu sắc cho tiêu đề và tiêu đề phụ.
- Sử dụng thẻ HTML trong chuỗi văn bản để kiểm soát định dạng.
- Sử dụng thuộc tính lineSpacingExtra để thêm khoảng cách dòng, giúp dễ đọc hơn.
- Thêm ScrollView vào bố cục để cho phép cuộn trong phần tử TextView.
- Thêm thuộc tính autoLink để các URL trong văn bản có thể nhấp vào và mở trực tiếp.

## Tổng quan về ứng dụng

Ứng dụng Scrolling Text minh họa thành phần giao diện **ScrollView**. ScrollView là một ViewGroup, trong ví dụ này chứa một TextView. Nó hiển thị một trang văn bản dài—cụ thể là bài đánh giá album nhạc—mà người dùng có thể cuộn dọc để đọc bằng cách vuốt lên và xuống. Một thanh cuộn xuất hiện ở lề phải. Ứng dụng cho thấy cách sử dụng văn bản được định dạng với thẻ HTML tối giản để đặt chữ đậm hoặc nghiêng, cũng như kí tự xuống dòng để tách đoạn văn. Ngoài ra, bạn có thể chèn liên kết web có thể nhấp vào văn bản.

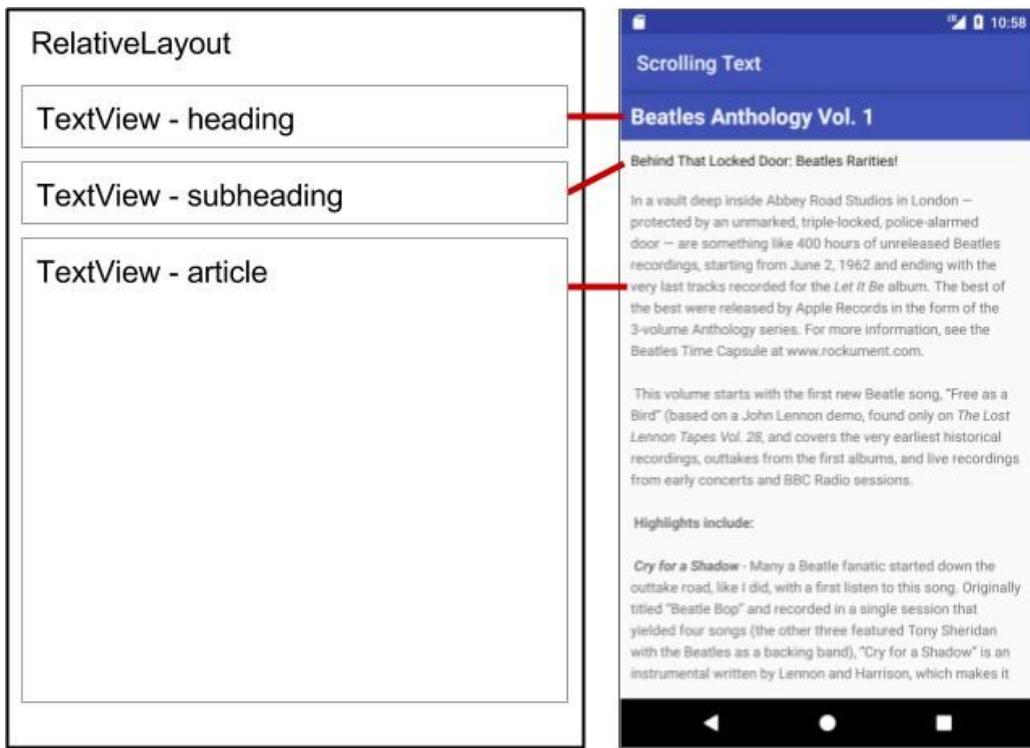


Ở trong hình trên, xuất hiện các thành phần sau:

1. Một liên kết web có thể nhấp được nhúng trong văn bản tự do.
2. Thanh cuộn xuất hiện khi cuộn văn bản.

## Nhiệm vụ 1: Thêm và chỉnh sửa các phần tử TextView

Trong bài thực hành này, bạn sẽ tạo một dự án Android cho ứng dụng ScrollingText, thêm các phần tử TextView vào bố cục để hiển thị tiêu đề và phụ đề của bài viết, đồng thời thay đổi phần tử TextView hiện có từ “Hello World” thành một bài viết dài. Hình dưới đây là sơ đồ bố cục của giao diện.



Bạn sẽ thực hiện tất cả các thay đổi này trong mã XML và tệp ***strings.xml***. Bạn sẽ chỉnh sửa mã XML cho bố cục trong Text pane, bằng cách nhấp vào thẻ **Text**, thay vì nhấp vào thẻ Design để mở **Design** trang. Một số thay đổi đối với phần tử UI và thuộc tính sẽ dễ thực hiện hơn trực tiếp trong Text trang bằng cách sử dụng mã nguồn XML.

## 1.1 Tạo dự án và các phần tử TextView

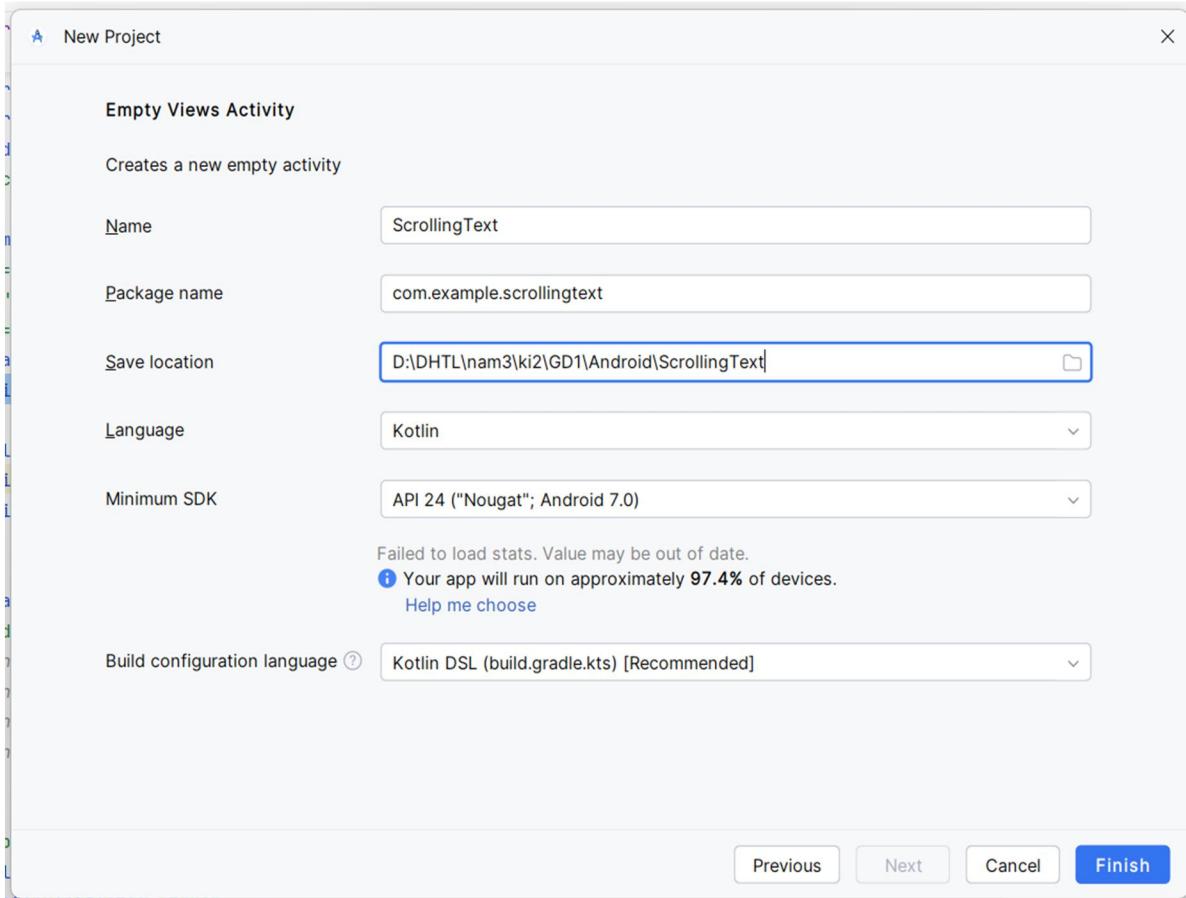
Trong nhiệm vụ này, bạn sẽ tạo dự án và các phần tử TextView, đồng thời sử dụng các thuộc tính của **TextView** để tạo kiểu cho văn bản và nền.

**Mẹo:** Để tìm hiểu thêm về các thuộc tính này, hãy tham khảo tài liệu **TextView reference**.

- Trong Android Studio, tạo một dự án mới với các tham số sau:

Thuộc tính	Giá trị
Application Name	Scrolling Text
Company Name	com.example.scollingtext
Phone and Tablet Minimum SDK	API 24: Android 7.0 Nougat
Template	Empty Views Activity
Generate Layout File checkbox	Chọn

Backwards Compatibility (AppCompat) checkbox	Chọn
-------------------------------------------------	------



2. Trong thư mục **app > res > layout** của **Project > Android pane**, mở tệp **activity\_main.xml**, sau đó nhấn vào tab **Text** để xem mã XML.

Ở phần đầu (hay **gốc**) của **View hierarchy** là **ViewGroup ConstraintLayout**:

```
<androidx.constraintlayout.widget.ConstraintLayout>
```

3. Thay đổi **ViewGroup** này thành **RelativeLayout**. Dòng thứ hai của mã bây giờ sẽ trông như sau:

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

**RelativeLayout** cho phép bạn đặt các phần tử giao diện người dùng (UI) tương đối với nhau hoặc tương đối với chính **RelativeLayout** cha.

Phần tử TextView mặc định “Hello World” được tạo bởi mẫu Empty Layout vẫn có các thuộc tính ràng buộc (ví dụ: layout\_constraintBottom\_toBottomOf="parent").  
Đừng lo lắng - bạn sẽ xóa chúng trong bước tiếp theo.

4. Xóa dòng mã XML sau, dòng này liên quan đến **ConstraintLayout**.

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

Khối mã XML ở đầu tệp bây giờ trông như sau:

```
<?xml version="1.0" encoding="utf-8"?>
© <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".MainActivity">
 ...
</RelativeLayout>
```

5. Thêm một phần tử TextView phía trên TextView “Hello World” bằng cách nhập `<TextView>`. Một khối TextView sẽ xuất hiện, kết thúc bằng `</TextView>` và hiển thị các thuộc tính `layout_width` và `layout_height`, đây là các thuộc tính bắt buộc cho TextView.
6. Nhập các thuộc tính sau cho TextView. Khi nhập từng thuộc tính và giá trị, các gợi ý sẽ xuất hiện để hoàn thành tên hoặc giá trị của thuộc tính.

TextView #1 attribute	Giá trị
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:id	"@+id/article_heading"
android:background	"@color/colorPrimary"
android:textColor	"@android:color/white"
android:padding	"10dp"
android:textAppearance	"@android:style/TextAppearance.DeviceDefault.Large"
android:textStyle	"bold"
android:text	"Article Title"

7. Trích xuất tài nguyên chuỗi cho thuộc tính android:text với chuỗi mã cứng "Article Title" trong TextView để tạo một mục nhập cho nó trong **strings.xml**.  
Đặt con trỏ vào chuỗi mã cứng, nhấn **Alt-Enter** (hoặc **Option-Enter** trên Mac), sau đó chọn **Extract string resource**. Đảm bảo rằng tùy chọn **Create the resource in directories** được chọn, sau đó chỉnh sửa tên tài nguyên thành **article\_title**. Tài nguyên chuỗi được mô tả chi tiết trong **String Resources**.
8. Trích xuất tài nguyên kích thước cho thuộc tính android:padding với giá trị mã cứng "10dp" trong TextView để tạo dimens.xml và thêm một mục nhập vào đó.  
Đặt con trỏ vào chuỗi mã cứng, nhấn **Alt-Enter** (hoặc **Option-Enter** trên Mac), sau đó chọn **Extract dimension resource**. Đảm bảo rằng tùy chọn **Create the resource in directories** được chọn, sau đó chỉnh sửa tên tài nguyên thành **padding\_regular**.
9. Thêm một phần tử TextView khác phía trên TextView "Hello World" và bên dưới TextView mà bạn đã tạo trong các bước trước. Thêm các thuộc tính sau vào TextView:

<b>TextView #2 attribute</b>	<b>Giá trị</b>
layout_width	"match_parent"
layout_height	"wrap_content"
android:id	"@+id/article_subheading"
android:layout_below	"@+id/article_heading"
android:padding	"@dimen/padding_regular"
android:textAppearance	"@android:style/TextAppearance.DeviceDefault"
android:text	"Article Subtitle"

Vì bạn đã trích xuất tài nguyên kích thước cho chuỗi "10dp" thành **padding\_regular** trong TextView đã tạo trước đó, bạn có thể sử dụng "**@dimen/padding\_regular**" cho thuộc tính android:padding trong TextView này.

10. Trích xuất tài nguyên chuỗi cho thuộc tính android:text với chuỗi mã cứng "Article Subtitle" trong TextView thành **article\_subtitle**.

11. Trong phần tử TextView chứa "Hello World", xóa tất cả các thuộc tính layout\_constraint:

```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"
```

12. Thêm các thuộc tính TextView sau vào phần tử TextView chứa "Hello World", và thay đổi thuộc tính android:text:

TextView Attribute	Giá trị
android:id	"@+id/article"
android:layout_below	"@+id/article_subheading"
android:lineSpacingExtra	"5sp"
android:padding	"@dimen/padding_regular"
android:text	Change to "Article text"

13. Trích xuất tài nguyên chuỗi cho "Article text" thành **article\_text** và trích xuất tài nguyên kích thước cho "5sp" thành **line\_spacing**.

14. Định dạng lại và căn chỉnh mã bằng cách chọn **Code > Reformat Code**. Thực hành tốt là định dạng lại và căn chỉnh mã của bạn để bạn và những người khác dễ hiểu hơn.

## 1.2 Thêm nội dung bài viết

Trong một ứng dụng thực sự truy cập các bài báo trên tạp chí hoặc báo, các bài báo xuất hiện có thể đến từ một nguồn trực tuyến thông qua nhà cung cấp nội dung hoặc có thể được lưu trước trong cơ sở dữ liệu trên thiết bị.

Đối với thực tế này, bạn sẽ tạo bài viết dưới dạng một chuỗi dài duy nhất trong tài nguyên strings.xml.

1. Trong thư mục **app > res > values**, hãy mở **strings.xml**.
2. Mở bất kỳ tệp văn bản nào có một lượng lớn văn bản hoặc mở tệp strings.xml của ứng dụng ScrollingText đã hoàn thành.
3. Nhập các giá trị cho các chuỗi **article\_title** và **article\_subtitle** với tiêu đề và phụ đề được tạo ra hoặc sử dụng các giá trị trong tệp strings.xml của đã hoàn thành

Ứng dụng ScrollingText. Đặt giá trị chuỗi thành văn bản một dòng không có thẻ HTML hoặc nhiều dòng.

#### 4. Nhập hoặc sao chép và dán văn bản cho chuỗi **article\_text**.

Bạn có thể sử dụng văn bản trong tệp văn bản của mình, hoặc sử dụng văn bản được cung cấp cho chuỗi **article\_text** trong tệp strings.xml của ứng dụng ScrollingText đã hoàn thành. Yêu cầu duy nhất cho tác vụ này là văn bản phải đủ dài để không vừa với màn hình.

Hãy ghi nhớ những điều sau (tham khảo hình bên dưới để biết về 1 ví dụ):

- Khi bạn nhập hoặc dán văn bản vào tệp *strings.xml*, các dòng văn bản không ngắt quãng đến dòng tiếp theo - chúng kéo dài ra ngoài lề bên phải. Đây là hành vi chính xác - mỗi dòng văn bản mới bắt đầu từ lề trái đại diện cho toàn bộ đoạn văn. Nếu bạn muốn văn bản trong *strings.xml* được bao bọc, bạn có thể nhấn **Return** để nhập kết thúc dòng cứng hoặc định dạng văn bản trước tiên trong trình soạn thảo văn bản có kết thúc dòng cứng.
- Nhập **\n** để biểu thị phần cuối của một dòng và một **\n** khác để biểu thị một dòng trống. Bạn cần thêm các ký tự cuối dòng để giữ cho các đoạn văn không dính sát vào nhau.
- Nếu bạn có dấu nháy đơn ('') trong văn bản của mình, bạn phải thoát khỏi nó bằng cách trước nó cho dấu gạch chéo ngược (\'). Nếu bạn có dấu ngoặc kép trong văn bản của mình, bạn cũng phải thoát nó (\") . Bạn cũng phải thoát khỏi bất kỳ ký tự không phải ASCII nào khác. Xem phần **Định dạng và tạo kiểu** của [tài nguyên Chuỗi](#) để biết thêm chi tiết.
- Nhập thẻ HTML **<b>** và **</b>** xung quanh các từ nên được in đậm.
- Nhập thẻ HTML **<i>** và **</i>** xung quanh các từ phải in nghiêng. Nếu bạn sử dụng dấu nháy đơn cong trong cụm từ nghiêng, hãy thay thế chúng bằng dấu nháy đơn thẳng.
- Bạn có thể kết hợp in đậm và in nghiêng bằng cách kết hợp các thẻ, như trong **<b><i>... words...</i></b>**. Các thẻ HTML khác bị bỏ qua.
- Đặt toàn bộ văn bản trong **<string name="article\_text"> </string>** trong tệp strings.xml.

- Bao gồm một liên kết web để kiểm tra, chẳng hạn như [www.google.com](http://www.google.com). (Ví dụ dưới đây sử dụng [www.rockument.com](http://www.rockument.com). ) Không sử dụng thẻ HTML, vì bất kỳ thẻ HTML nào ngoại trừ thẻ in đậm và nghiêng đều bị bỏ qua và được trình bày dưới dạng văn bản, đây không phải là điều bạn muốn.

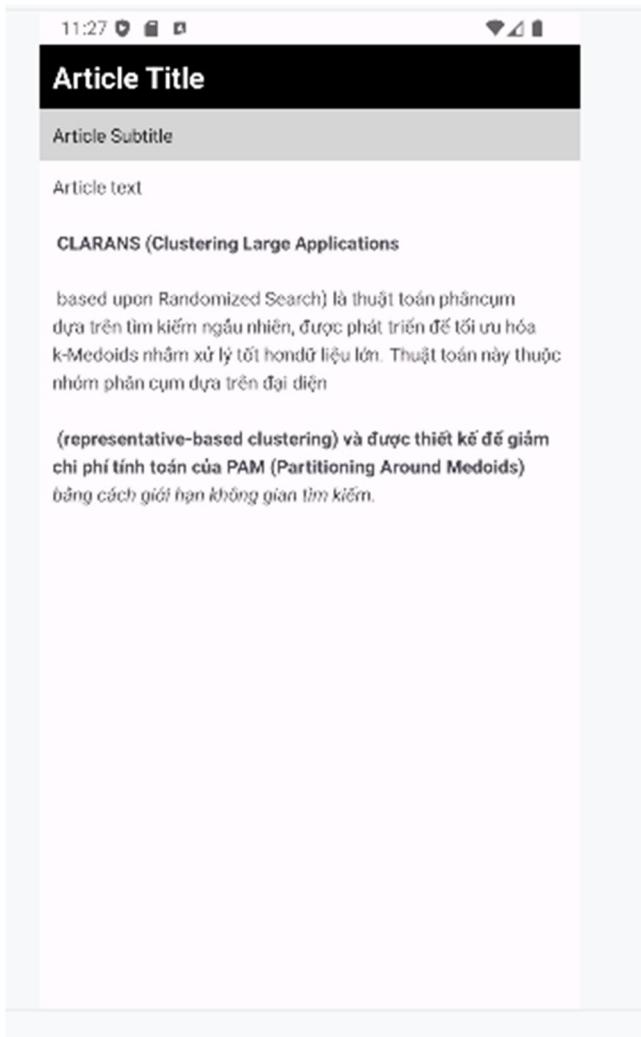
```

<resources>
 <string name="app_name">ScrollingText</string>
 <string name="article_title">Article Title</string>
 <string name="article_subtitle">Article Subtitle</string>
 <string name="article_text">Article text
 \n\n CLARANS (Clustering Large Applications)
 \n\n based upon Randomized Search) là thuật toán phân cụm dựa trên tìm kiếm ngẫu nhiên,
 được phát triển để tối ưu hóa k-Medoids nhằm xử lý tốt hơn dữ liệu lớn. Thuật toán này thuộc nhóm phân cụm dựa trên đại diện
 \n\n (representative-based clustering) và được thiết kế để giảm chi phí tính toán của PAM (Partitioning Around Medoids)
 <i>bằng cách giới hạn không gian tìm kiếm.</i>
 </string>
</resources>

```

### 1.3 Chạy ứng dụng

Chạy ứng dụng. Bài viết xuất hiện, nhưng người dùng không thể cuộn bài viết vì bạn chưa bao gồm ScrollView (Bạn sẽ thực hiện trong tác vụ tiếp theo). Cũng lưu ý rằng nhấn vào liên kết web hiện không làm được gì. Bạn cũng sẽ khắc phục điều đó trong nhiệm vụ tiếp theo.



## Mã giải pháp nhiệm vụ 1

Bố cục tệp activity\_main.xml trông như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".MainActivity">

 <TextView
 android:id="@+id/article_heading"
```

```
 android:layout_width="match_parent"
 android:layout_height="wrap_content"

 android:background="@color/material_dynamic_primary0"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_title"

 android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
 android:textColor="@color/white"
 android:textStyle="bold" />

<TextView
 android:id="@+id/article_subheading"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_below="@id/article_heading"
 android:background="#D5D5D5"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_subtitle"

 android:textAppearance="@android:style/TextAppearance.DeviceDefault" />

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

</RelativeLayout>
```

## Nhiệm vụ 2: Thêm ScrollView và liên kết web đang hoạt động

Trong nhiệm vụ trước, bạn đã tạo ứng dụng ScrollingText với các phần tử TextView cho tiêu đề bài viết, phụ đề và văn bản bài viết dài. Bạn cũng đã bao gồm một liên kết web, nhưng liên kết đó chưa hoạt động. Bạn sẽ thêm mã để làm cho nó hoạt động.

Ngoài ra, bản thân TextView không thể cho phép người dùng cuộn văn bản bài viết để xem tất cả. Bạn sẽ thêm một ViewGroup mới có tên là ScrollView vào bố cục XML sẽ làm cho TextView có thể cuộn được.

## 2.1 Thêm thuộc tính autoLink cho các liên kết web đang hoạt động

Thêm thuộc tính android:autoLink="web" vào phần tử TextView. Mã XML cho TextView này bây giờ trông như sau:

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

## 2.2 Thêm ScrollView vào bố cục

Để làm cho View (chẳng hạn như TextView) có thể cuộn được, hãy nhúng View bên trong một ScrollView.

1. Thêm một ScrollView giữa article\_subheading TextView và article TextView. Khi bạn vào <ScrollView>, Android Studio sẽ tự động thêm </ScrollView> ở cuối và hiển thị các thuộc tính android:layout\_width và android:layout\_height kèm theo các gợi ý.
2. Chọn **wrap\_content** từ các gợi ý cho cả hai thuộc tính.

Mã cho hai phần tử TextView và ScrollView bây giờ trông như thế này:

```
<TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:id="@+id/article_subheading"
```

```
 android:layout_below="@+id/article_heading"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_subtitle"
 android:textAppearance=
 "@android:style/TextAppearance.DeviceDefault"/>

<ScrollView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"></ScrollView>

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

3. Di chuyển phần kết thúc **</ScrollView>** mã sau **article** TextView để các thuộc tính TextView của **article** hoàn toàn nằm bên trong **ScrollView**.
4. Xóa thuộc tính sau khỏi **article** TextView và thêm nó vào ScrollView:

```
 android:layout_below="@+id/article_subheading"
```

Với thuộc tính trên, phần tử ScrollView sẽ xuất hiện bên dưới tiêu đề phụ của bài viết. Bài viết nằm bên trong phần tử ScrollView.

5. Chọn **Code > Reformat Code** để định dạng lại mã XML để article TextView hiện xuất hiện thực lề bên trong mã **<ScrollView>**.
6. Nhấp vào thẻ **Preview** ở phía bên phải của trình chỉnh sửa bố cục để xem trước bố cục.

Bố cục bây giờ trông giống như phía bên phải của hình sau:



## 2.3 Chạy ứng dụng

Để kiểm tra cách văn bản cuộn:

- Chạy ứng dụng trên thiết bị hoặc trình mô phỏng.  
Vuốt lên và xuống để cuộn bài viết. Thanh cuộn xuất hiện ở lề phải khi bạn cuộn. Nhấn vào web liên kết để chuyển đến web trang. Thuộc tính android:autoLink biến bất kỳ URL dễ nhận biết nào trong TextView (chẳng hạn như [www.rockument.com](http://www.rockument.com)) thành một liên kết web.
- Xoay thiết bị hoặc trình mô phỏng của bạn trong khi chạy ứng dụng. Lưu ý cách chế độ xem cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.
- Chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng. Lưu ý cách chế độ xem cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.

# Article Title

Article Subtitle

Article text <https://github.com/>

## CLARANS (Clustering Large Applications)

based upon Randomized Search) là thuật toán phân cụm dựa trên tìm kiếm ngẫu nhiên, được phát triển để tối ưu hóa k-Medoids nhằm xử lý tốt hơn dữ liệu lớn. Thuật toán này thuộc nhóm phân cụm dựa trên đại diện

(representative-based clustering) và được thiết kế để giảm chi phí tính toán của PAM (Partitioning Around Medoids) bằng cách giới hạn không gian tìm kiếm.

## CLARANS (Clustering Large Applications)

based upon Randomized Search) là thuật toán phân cụm dựa trên tìm kiếm ngẫu nhiên, được phát triển để tối ưu hóa k-Medoids nhằm xử lý tốt hơn dữ liệu lớn. Thuật toán này thuộc nhóm phân cụm dựa trên đại diện

Trong hình trên, xuất hiện như sau:

1. Một liên kết web đang hoạt động được nhúng trong văn bản dạng tự do
2. Thanh cuộn xuất hiện khi cuộn văn bản

## Mã cho nhiệm vụ 2

Mã XML cho bố cục với chế độ xem cuộn như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
```

```
 android:id="@+id/main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".MainActivity">

 <TextView
 android:id="@+id/article_heading"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:background="@color/material_dynamic_primary0"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_title"
 android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
 android:textColor="@color/white"
 android:textStyle="bold" />

 <TextView
 android:id="@+id/article_subheading"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_below="@+id/article_heading"
 android:background="#D5D5D5"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_subtitle"
 android:textAppearance="@android:style/TextAppearance.DeviceDefault" />

 <ScrollView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/article_subheading">

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

```

```
</ScrollView>

</RelativeLayout>
```

## Nhiệm vụ 3: Cuộn nhiều phần tử

Như đã lưu ý trước đây, một ScrollView chỉ có thể chứa một View con (chẳng hạn như một `Article TextView` mà bạn đã tạo). Tuy nhiên, View đó có thể là một ViewGroup khác chứa các phần tử View, chẳng hạn như [LinearLayout](#). Bạn có thể tìm một ViewGroup như `LinearLayout` với ScrollView, do đó cuộn mọi thứ bên trong `LinearLayout`.

Ví dụ: nếu bạn muốn tiêu đề phụ của bài viết cuộn cùng với bài viết, hãy thêm `LinearLayout` trong `ScrollView` và di chuyển tiêu đề phụ và bài viết vào `LinearLayout`. `LinearLayout` trở thành View con duy nhất trong `ScrollView` như trong hình bên dưới và người dùng có thể cuộn toàn bộ `LinearLayout`: tiêu đề phụ và bài viết.

### 3.1 Thêm `LinearLayout` vào `ScrollView`

1. Mở tệp `activity_main.xml` của dự án ứng dụng `ScrollingText` và chọn tab Văn bản để chỉnh sửa mã XML (nếu chưa được chọn).
2. Thêm `LinearLayout` phía trên `TextView` bài viết trong `ScrollView`. Khi bạn nhập `<LinearLayout`, Android Studio sẽ tự động thêm `</LinearLayout>` vào cuối, đồng thời hiển thị các thuộc tính `android:layout_width` và `android:layout_height` kèm theo các đề xuất. Chọn `match_parent` và `wrap_content` từ các đề xuất về chiều rộng và chiều cao của nó, tương ứng. Mã ở đầu `ScrollView` bây giờ trông như thế này:

```
<ScrollView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/article_subheading">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"></LinearLayout>

 <TextView
 android:id="@+id/article"
```

Bạn sử dụng `match_parent` để khớp với chiều rộng của `ViewGroup` gốc. Bạn sử dụng `wrap_content` để thay đổi kích thước `LinearLayout` sao cho nó vừa đủ lớn để bao bọc nội dung của nó.

3. Di chuyển mã kết thúc `</LinearLayout>` sau bài viết `TextView` nhưng trước khi kết thúc `</ScrollView>`.

`LinearLayout` hiện bao gồm bài viết `TextView` và hoàn toàn nằm bên trong Chế độ xem cuộn.

4. Thêm thuộc tính `android:orientation="vertical"` vào `LinearLayout` để đặt hướng của nó thành dọc.
5. Chọn **Code > Reformat Code** để thuần lề mã một cách chính xác.

`LinearLayout` bây giờ trông như thế này:

```
<ScrollView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/article_subheading">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical">

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

</ScrollView>
```

### 3.2 Di chuyển các thành phần giao diện người dùng trong `LinearLayout`

LinearLayout hiện chỉ có một phần tử giao diện người dùng — bài viết TextView. Bạn muốn bao gồm article\_subheading TextView trong LinearLayout để cả hai đều cuộn.

1. Để di chuyển TextView article\_subheading, hãy chọn mã, chọn Edit > Cut, nhấp vào phía trên TextView của bài viết bên trong LinearLayout và chọn Edit > Paste.
2. Xoá thuộc tính android:layout\_below="@+id/article\_heading" khỏi thuộc tính article\_subheading TextView. Vì TextView này hiện nằm trong LinearLayout nên thuộc tính này sẽ xung đột với các thuộc tính LinearLayout.
3. Thay đổi thuộc tính bố cục ScrollView từ android:layout\_below="@+id/article\_subheading" thành android:layout\_below="@+id/article\_heading". Bây giờ tiêu đề phụ là một phần của LinearLayout, ScrollView phải được đặt bên dưới tiêu đề, không phải tiêu đề phụ.

Mã XML cho ScrollView bây giờ là như sau:

```
<ScrollView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/article_heading">

<LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical">

<TextView
 android:id="@+id/article_subheading"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_subtitle"
 android:textAppearance=
 "@android:style/TextAppearance.DeviceDefault" />

<TextView
 android:id="@+id/article"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:autoLink="web"
```

```
 android:lineSpacingExtra="@dimen/line_spacing"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_text" />
 </LinearLayout>

</ScrollView>
```

#### 4. Chạy ứng dụng

Vuốt lên và xuống để cuộn bài viết và lưu ý rằng tiêu đề phụ bây giờ cuộn cùng với bài viết trong khi tiêu đề vẫn giữ nguyên.

## Mã giải pháp

Dự án Android Studio: [ScrollingText](#)

### Tóm tắt

- Sử dụng ScrollView để cuộn một View con (chẳng hạn như TextView). ScrollView chỉ có thể chứa một View hoặc ViewGroup con.
- Sử dụng ViewGroup chẳng hạn như LinearLayout làm View con trong ScrollView để cuộn nhiều phần tử View. Bao gồm các phần tử trong LinearLayout.
- Hiển thị văn bản dạng tự do trong TextView với các thẻ định dạng HTML cho in đậm và in nghiêng.
- Sử dụng \n làm ký tự cuối dòng trong văn bản dạng tự do để giữ cho một đoạn văn không chạy vào đoạn tiếp theo.
- Sử dụng thuộc tính android:autoLink="web" để làm cho các liên kết web trong văn bản có thể nhấp được.

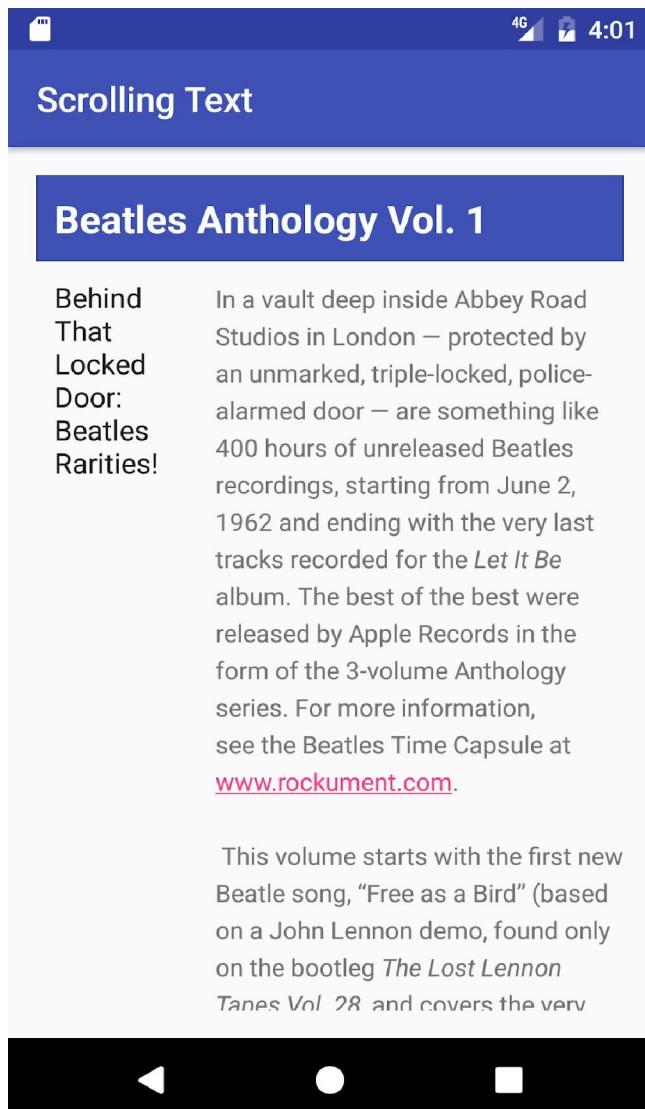
## Bài tập

### Thay đổi ứng dụng

Mở ứng dụng ScrollingText2 mà bạn đã tạo trong bài học Làm việc với các phần tử TextView.

1. Thay đổi tiêu đề phụ để nó bao bọc trong một cột ở bên trái rộng 100 dp, như hình dưới đây.

2. Đặt văn bản của bài viết ở bên phải của tiêu đề phụ như hình dưới đây.



### 1.5) Tài nguyên có sẵn

## Giới thiệu

### Những điều bạn nên biết

Bạn sẽ có thể:

- Tìm hiểu quy trình làm việc cơ bản của Android Studio.
- Tạo ứng dụng từ đầu bằng cách sử dụng mẫu Hoạt động trống.
- Sử dụng trình chỉnh sửa bố cục.

## Những gì bạn sẽ học

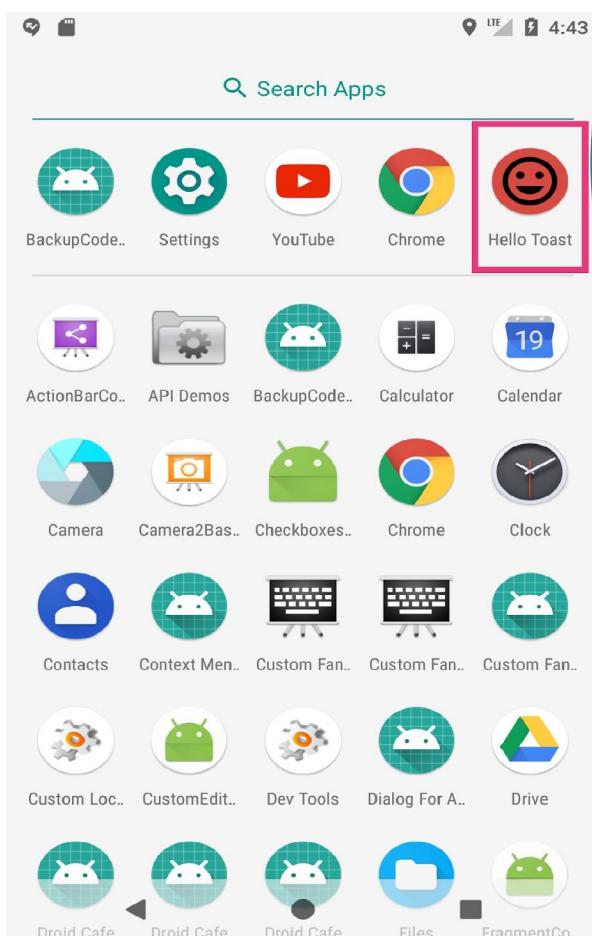
- Tìm thông tin và tài nguyên dành cho nhà phát triển ở đâu.
- Cách thêm biểu tượng trình khởi chạy vào ứng dụng của bạn.
- Cách tìm kiếm trợ giúp khi bạn đang phát triển ứng dụng Android.

## Bạn sẽ làm gì

- Khám phá một số tài nguyên có sẵn cho các nhà phát triển Android ở mọi cấp độ.
- Thêm biểu tượng trình chạy cho ứng dụng của bạn.

## Tổng quan về ứng dụng

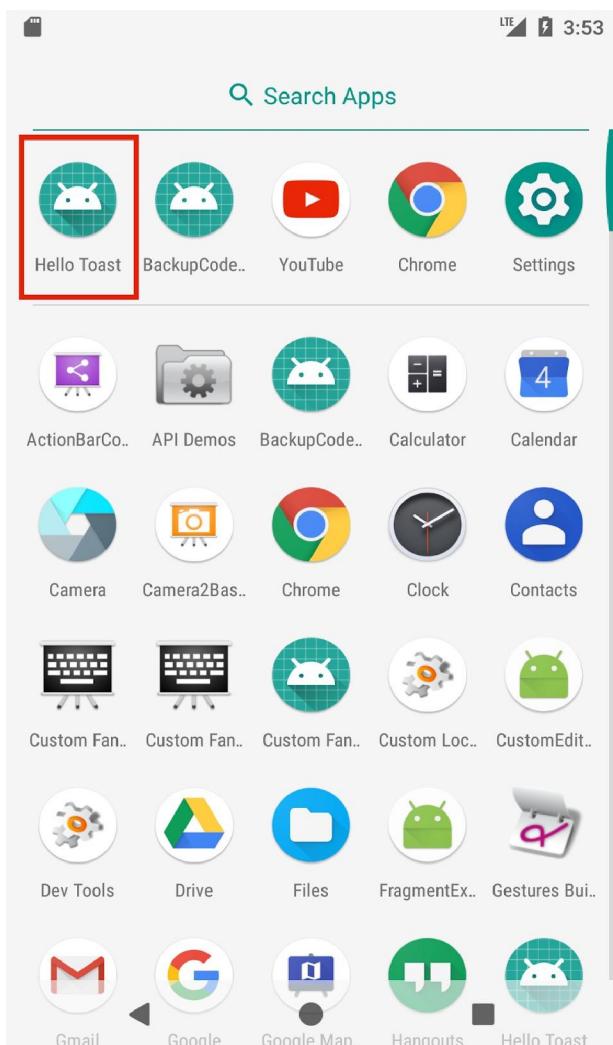
Bạn sẽ thêm biểu tượng launcher vào ứng dụng HelloToast mà bạn đã tạo trước đó hoặc vào một ứng dụng mới.



## Nhiệm vụ 1: Thay đổi biểu tượng trình khởi chạy

Mỗi ứng dụng mới bạn tạo bằng Android Studio đều bắt đầu bằng một biểu tượng trình chạy tiêu chuẩn đại diện cho ứng dụng. Biểu tượng trình chạy xuất hiện trong danh sách cửa hàng Google Play. Khi người dùng tìm kiếm trên cửa hàng Google Play, biểu tượng cho ứng dụng của bạn sẽ xuất hiện trong kết quả tìm kiếm.

Khi người dùng đã cài đặt ứng dụng, biểu tượng trình khởi chạy sẽ xuất hiện trên thiết bị ở nhiều nơi khác nhau bao gồm màn hình chính và màn hình Tìm kiếm Ứng dụng. Ví dụ: ứng dụng HelloToast xuất hiện trong màn hình Tìm kiếm ứng dụng của trình mô phỏng với biểu tượng tiêu chuẩn cho các dự án ứng dụng mới, như minh họa bên dưới.



Thay đổi biểu tượng trình chạy là một quy trình từng bước đơn giản giới thiệu cho bạn các tính năng nội dung hình ảnh của Android Studio. Trong nhiệm vụ này, bạn cũng tìm hiểu thêm về cách truy cập tài liệu chính thức của Android.

## 1.1 Khám phá tài liệu chính thức của Android

Bạn có thể tìm thấy tài liệu chính thức dành cho nhà phát triển Android tại [developer.android.com](https://developer.android.com). Tài liệu này chứa rất nhiều thông tin được Google cập nhật.

1. Đi đến [developer.android.com/design/](https://developer.android.com/design/).

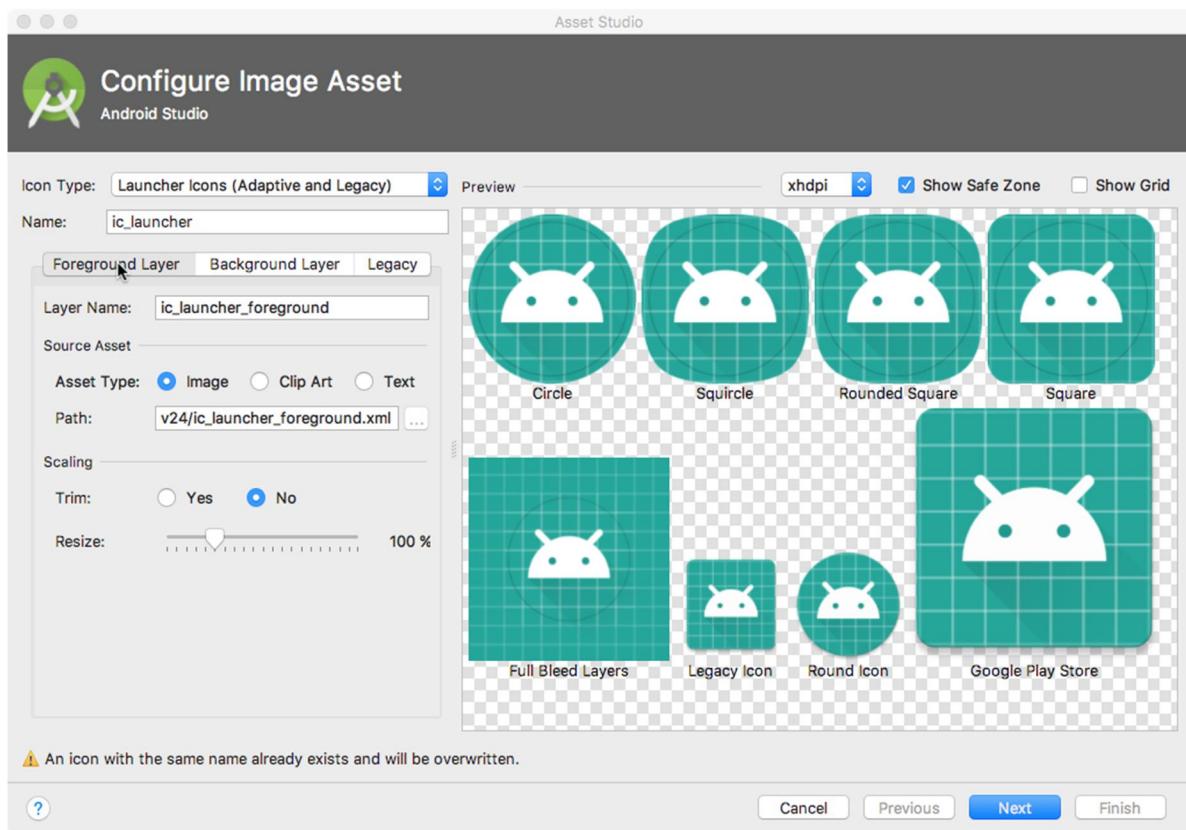
Phần này nói về Material Design, là một triết lý thiết kế khái niệm phác thảo cách các ứng dụng nên trông và hoạt động trên thiết bị di động. Điều hướng các liên kết để tìm hiểu thêm về Material Design. Ví dụ: truy cập phần Phong cách để tìm hiểu thêm về cách sử dụng màu sắc và các chủ đề khác.

2. Truy cập [developer.android.com/docs/](https://developer.android.com/docs/) để tìm thông tin API, tài liệu tham khảo, hướng dẫn, hướng dẫn công cụ và mẫu mã.
3. Truy cập [developer.android.com/distribute/](https://developer.android.com/distribute/) để tìm thông tin về cách đưa ứng dụng lên Google Play, hệ thống phân phối kỹ thuật số của Google dành cho các ứng dụng được phát triển bằng Android SDK. Sử dụng Google Play Console để phát triển cơ sở người dùng của bạn và bắt đầu kiếm tiền.

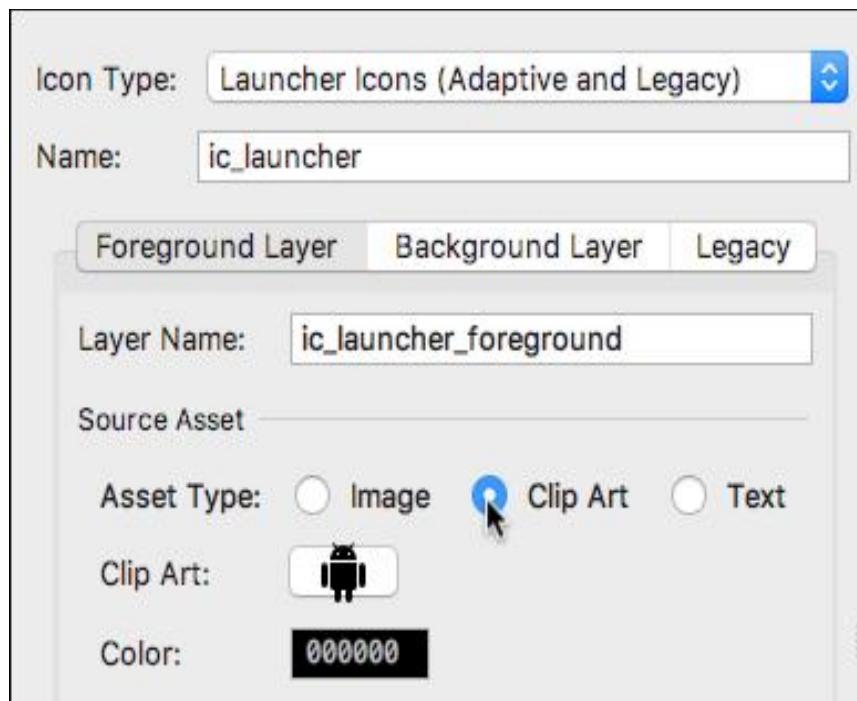
## 1.2 Thêm nội dung hình ảnh cho biểu tượng trình chạy

Để thêm hình ảnh clip-art làm biểu tượng trình khởi chạy, hãy làm theo các bước sau:

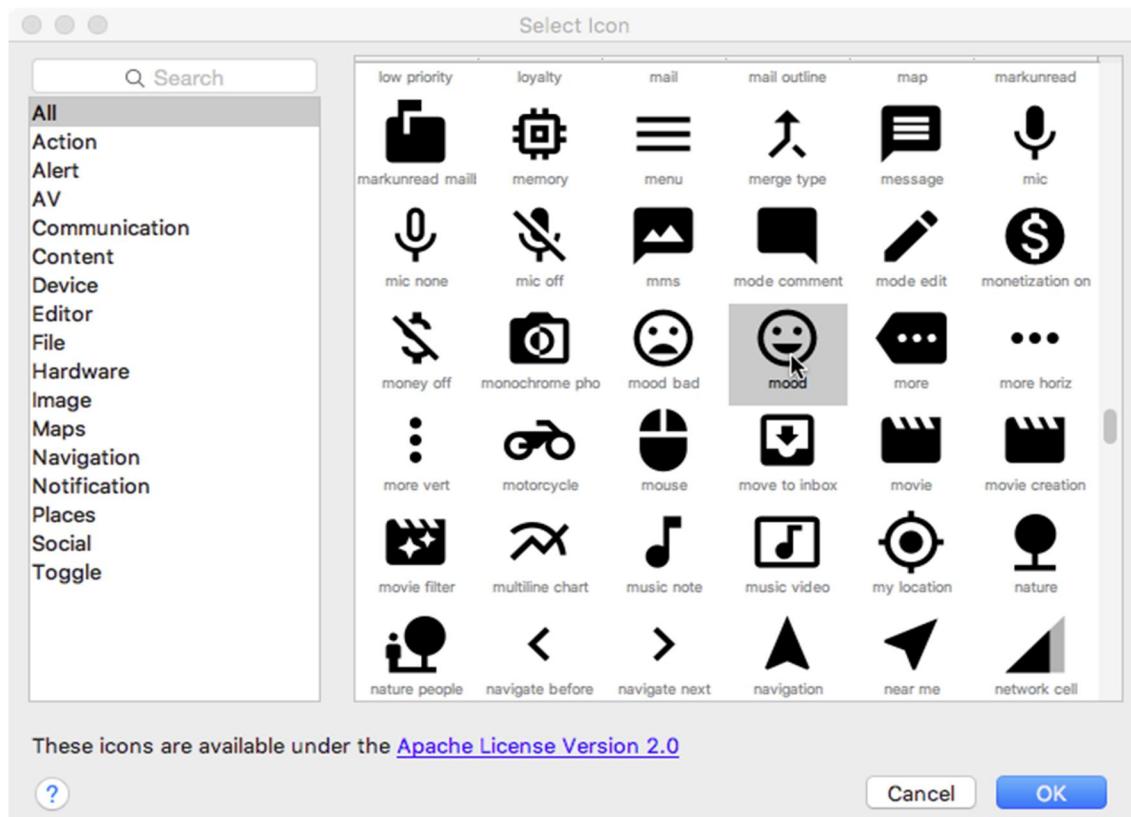
1. Mở dự án ứng dụng HelloToast từ bài học trước về cách sử dụng trình chỉnh sửa bố cục hoặc tạo dự án ứng dụng mới.
2. Trong ngăn Project > Android, nhấp chuột phải (hoặc Control khi nhấp vào) thư mục res và chọn New > Nội dung hình ảnh. Cửa sổ Định cấu hình nội dung hình ảnh xuất hiện.



3. Trong trường **Icon Type**, chọn **Launcher Icons (Adaptive & Legacy)** nếu nó chưa được chọn.
4. Nhấp vào tab **Foreground Layer**, chọn **Clip Art** cho Loại nội dung.

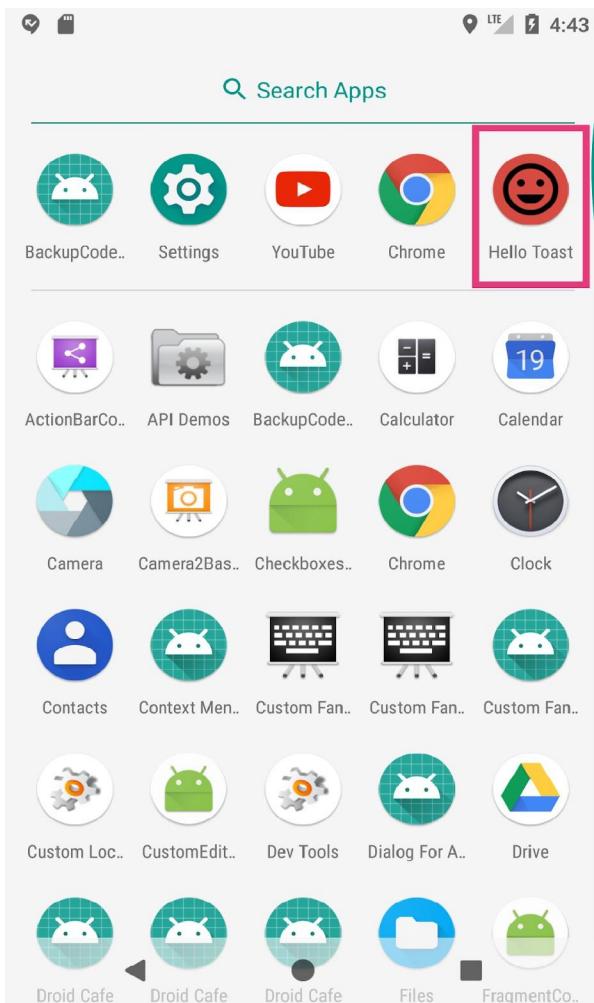


- Nhấp vào biểu tượng trong trường Clip Art. Các biểu tượng xuất hiện từ bộ biểu tượng thiết kế vật liệu.
- Duyệt qua cửa sổ Chọn Biểu tượng, chọn một biểu tượng thích hợp (chẳng hạn như biểu tượng tâm trạng để gợi ý tâm trạng tốt), sau đó bấm OK.



- Nhấp vào tab **Background Layer**, chọn **Color** làm **Asset Type**, sau đó nhấp vào chip màu để chọn màu để sử dụng làm layer nền.
- Nhấp vào tab **Legacy** và xem lại cài đặt mặc định. Xác nhận rằng bạn muốn tạo biểu tượng kế thừa, tròn và Cửa hàng Google Play. Nhấp vào Tiếp theo khi hoàn tất.
- Chạy ứng dụng.

Android Studio tự động thêm hình ảnh trình khởi chạy vào thư mục mipmap cho các mật độ khác nhau. Do đó, biểu tượng khởi chạy ứng dụng sẽ thay đổi thành biểu tượng mới sau khi bạn chạy ứng dụng, như hình dưới đây.



Mẹo: Xem Biểu tượng trình chạy để tìm hiểu thêm về cách thiết kế biểu tượng trình chạy hiệu quả.

## Nhiệm vụ 2: Sử dụng mẫu dự án

Android Studio cung cấp các mẫu cho các thiết kế hoạt động và ứng dụng phổ biến và được đề xuất. Sử dụng các mẫu tích hợp giúp tiết kiệm thời gian và giúp bạn tuân theo các phương pháp thiết kế tốt nhất.

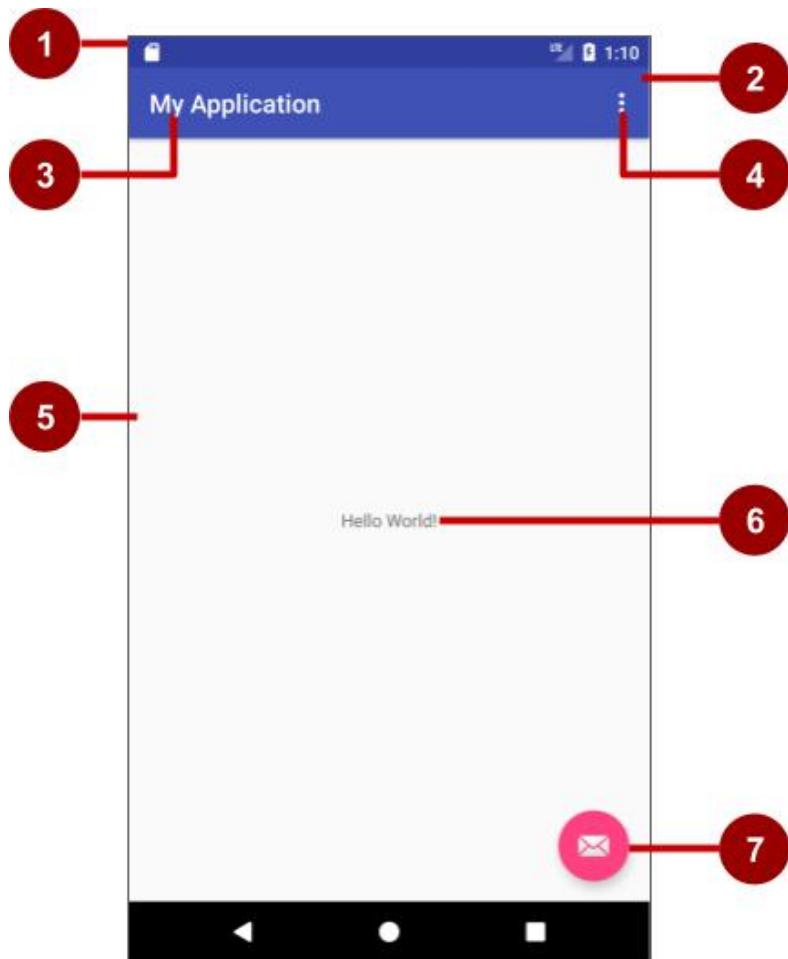
Mỗi mẫu kết hợp một hoạt động khung và giao diện người dùng. Bạn đã sử dụng mẫu Hoạt động trống. Mẫu Hoạt động cơ bản có nhiều tính năng hơn và kết hợp các tính năng ứng dụng được đề xuất, chẳng hạn như menu tùy chọn xuất hiện trong thanh ứng dụng.

### 2.1 Khám phá cấu trúc hoạt động cơ bản

Mẫu Basic Activity là một mẫu đa năng do Android Studio cung cấp để hỗ trợ bạn bắt đầu phát triển ứng dụng của mình.

1. Trong Android Studio, hãy tạo một dự án mới bằng mẫu Basic Activity.
2. Xây dựng và chạy ứng dụng.
3. Xác định các bộ phận được dán nhãn trong hình và bảng bên dưới. Tìm các ứng dụng tương đương trên màn hình thiết bị hoặc trình mô phỏng của bạn. Kiểm tra mã Java tương ứng và các tệp XML được mô tả trong bảng.

Làm quen với mã nguồn Java và các tệp XML sẽ giúp bạn mở rộng và tùy chỉnh mẫu này cho nhu cầu của riêng bạn.



## Kiến trúc của mẫu Hoạt động cơ bản

#	Mô tả giao diện người dùng	Tham khảo mã
1	Thanh trạng thái Hệ thống Android cung cấp và kiểm soát thanh trạng thái.	Không hiển thị trong mã mẫu. Bạn có thể truy cập nó từ hoạt động của bạn. Ví dụ: bạn có thể <u>ẩn thanh trạng thái</u> , nếu cần.
2	AppBarLayout > Thanh công cụ Thanh ứng dụng (còn được gọi là thanh hành động) cung cấp cấu trúc trực quan, các yếu tố hình ảnh được chuẩn hóa và điều hướng. Đối với ngược tương thích, AppBarLayout trong mẫu nhúng Thanh công cụ có chức năng tương tự như ActionBar.	Trong activity_main.xml, hãy tìm android.support.v7.widget.Thanh công cụ bên trong android.support.design.widget.AppBarLayout. Thay đổi thanh công cụ để thay đổi giao diện của thanh ứng dụng. Đối với người yêu cùample, hãy xem Hướng dẫn thanh ứng dụng.
3	Tên ứng dụng Điều này bắt nguồn từ tên gói của bạn, nhưng có thể là bất cứ thứ gì bạn chọn.	Trong AndroidManifest.xml: android:label="@string/app_name"
4	Nút tràn menu tùy chọn Các mục menu cho hoạt động, cũng như các tùy chọn chung, chẳng hạn như Tìm kiếm và Cài đặt cho ứng dụng. Các mục menu ứng dụng của bạn sẽ đi vào menu này.	Trong MainActivity.java: onOptionsItemSelected() thực hiện những gì xảy ra khi một mục menu được chọn.  > menu_main.xml menu res > Tài nguyên chỉ định các mục menu cho menu tùy chọn.
5	Nhóm chế độ xem bố cục CoordinatorLayout ViewGroup là một bố cục giàu tính năng cung cấp cơ chế cho các phần tử Chế độ xem (UI) tương tác. Giao diện người dùng của ứng dụng nằm bên trong tệp content_main.xml có trong ViewGroup này.	Trong activity_main.xml: Không có chế độ xem nào được chỉ định trong bố cục này; thay vào đó, nó bao gồm một bố cục khác với một lệnh bố cục bao gồm để bao gồm @layout/content_main nơi các chế độ xem được chỉ định. Điều này tách chế độ xem hệ thống khỏi chế độ xem duy nhất cho ứng dụng của bạn.

6	Chế độ xem văn bản  Trong ví dụ, được sử dụng để hiển thị "Hello World". Thay thế điều này bằng các thành phần giao diện người dùng cho ứng dụng của bạn.	Trong content_main.xml:  Tất cả các thành phần giao diện người dùng của ứng dụng được xác định trong tệp này.
7	Nút hành động nổi (FAB)	Trong activity_main.xml như một phần tử giao diện người dùng sử dụng biểu tượng clip-art. MainActivity.java bao gồm một sơ khai trong onCreate() để đặt trình nghe onClick() cho FAB.

## Bài 2) Activities

- 2.1) Activity và Intent**
- 2.2) Vòng đời của Activity và trạng thái**
- 2.3) Intent ngầm định**

## Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ

- 3.1) Trình gỡ lỗi**
- 3.2) Kiểm thử đơn vị**
- 3.3) Thư viện hỗ trợ**

## **CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG**

### **Bài 1) Tương tác người dùng**

- 1.1) Hình ảnh có thể chọn**
- 1.2) Các điều khiển nhập liệu**
- 1.3) Menu và bộ chọn**
- 1.4) Điều hướng người dùng**
- 1.5) RecyclerView**

### **Bài 2) Trải nghiệm người dùng thú vị**

- 2.1) Hình vẽ, định kiểu và chủ đề**
- 2.2) Thẻ và màu sắc**
- 2.3) Bố cục thích ứng**

### **Bài 3) Kiểm thử giao diện người dùng**

- 3.1) Espresso cho việc kiểm tra UI**

## **CHƯƠNG 3. LÀM VIỆC TRONG NỀN**

### **Bài 1) Các tác vụ nền**

- 1.1) AsyncTask**
- 1.2) AsyncTask và AsyncTaskLoader**
- 1.3) Broadcast receivers**

### **Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền**

- 2.1) Thông báo**
- 2.2) Trình quản lý cảnh báo**
- 2.3) JobScheduler**

## **CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG**

**Bài 1) Tùy chọn và cài đặt**

**1.1) Shared preferences**

**1.2) Cài đặt ứng dụng**

**Bài 2) Lưu trữ dữ liệu với Room**

**2.1) Room, LiveData và ViewModel**

**2.2) Room, LiveData và ViewModel**