

**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 .....	29
1.3)	Trình chỉnh sửa bố cục .....	64
1.4)	Văn bản và các chế độ cuộn .....	86
1.5)	Tài nguyên có sẵn .....	86
Bài 2)	Activities .....	86
2.1)	Activity và Intent .....	86
2.2)	Vòng đời của Activity và trạng thái .....	86
2.3)	Intent ngầm định .....	86
Bài 3)	Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ .....	86
3.1)	Trình gỡ lỗi .....	86
3.2)	Kiểm thử đơn vị .....	86
3.3)	Thư viện hỗ trợ .....	86
CHƯƠNG 2.	Trải nghiệm người dùng .....	87
Bài 1)	Tương tác người dùng .....	87
1.1)	Hình ảnh có thể chọn .....	87
1.2)	Các điều khiển nhập liệu .....	87
1.3)	Menu và bộ chọn .....	87
1.4)	Điều hướng người dùng .....	87
1.5)	RecyclerView .....	87
Bài 2)	Trải nghiệm người dùng thú vị .....	87
2.1)	Hình vẽ, định kiểu và chủ đề .....	87
2.2)	Thẻ và màu sắc .....	87

2.3)	Bố cục thích ứng .....	87
Bài 3)	Kiểm thử giao diện người dùng .....	87
3.1)	Espresso cho việc kiểm tra UI .....	87
CHƯƠNG 3.	Làm việc trong nền .....	87
Bài 1)	Các tác vụ nền .....	87
1.1)	AsyncTask .....	87
1.2)	AsyncTask và AsyncTaskLoader .....	87
1.3)	Broadcast receivers .....	87
Bài 2)	Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền .....	87
2.1)	Thông báo .....	87
2.2)	Trình quản lý cảnh báo .....	87
2.3)	JobScheduler .....	87
CHƯƠNG 4.	Lưu dữ liệu người dùng .....	88
Bài 1)	Tùy chọn và cài đặt .....	88
1.1)	Shared preferences .....	88
1.2)	Cài đặt ứng dụng .....	88
Bài 2)	Lưu trữ dữ liệu với Room .....	88
2.1)	Room, LiveData và ViewModel .....	88
2.2)	Room, LiveData và ViewModel .....	88

3.1) Trính gowx loi .....

## 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 gì bạn sẽ học

- 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.

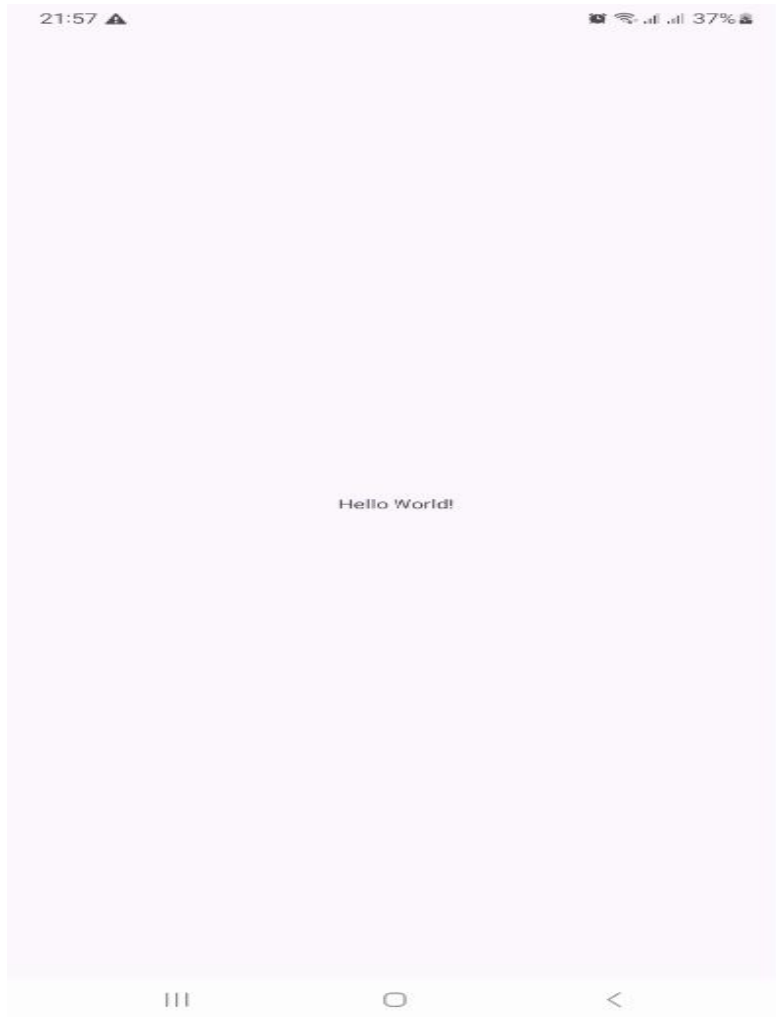
## Những gì bạn sẽ làm

- 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 bạn cài đặt thành công **Android Studio**, bạn sẽ tạo một dự án mới từ mẫu (**template**) cho ứng dụng **Hello World**. Ứng dụng đơn giản này sẽ hiển thị chuỗi "**Hello World**" trên màn hình của thiết bị **ảo** hoặc **thực** chạy Android.

Đây là giao diện của ứng dụng sau khi hoàn thành:



## Nhiệm vụ 1: Cài đặt Android Studio

**Android Studio** cung cấp một **môi trường phát triển tích hợp (IDE)** hoàn chỉnh, bao gồm **trình soạn thảo mã nâng cao** và một bộ **mẫu ứng dụng**. Ngoài ra, nó còn chứa các công cụ hỗ trợ **phát triển, gỡ lỗi, kiểm thử và tối ưu hóa hiệu suất**, giúp quá trình phát triển ứng dụng nhanh hơn và dễ dàng hơn.

Bạn có thể:

- ✓ **Kiểm thử ứng dụng** trên nhiều **trình giả lập** đã được cấu hình sẵn hoặc trên **thiết bị di động thực tế**.
- ✓ **Xây dựng ứng dụng** để phát hành chính thức.
- ✓ **Đăng tải ứng dụng** lên **Google Play Store**.

**Lưu ý:** **Android Studio** liên tục được cập nhật và cải tiến. Để biết thông tin mới nhất về **yêu cầu hệ thống** và **hướng dẫn cài đặt**, hãy xem tại [Android Studio](#).

**Android Studio** có sẵn cho **Windows**, **Linux** và **macOS**. Phiên bản mới nhất của **OpenJDK (Java Development Kit)** cũng được tích hợp sẵn trong **Android Studio**.

Các bước cài đặt **Android Studio**:

- ✓ **Bước 1:** Truy cập trang web dành cho [nhà phát triển Android](#) và làm theo hướng dẫn để tải xuống & cài đặt **Android Studio**.
- ✓ **Bước 2:** Chấp nhận **cấu hình mặc định** cho tất cả các bước và đảm bảo rằng **tất cả các thành phần cần thiết** được chọn để cài đặt.
- ✓ **Bước 3:** Sau khi hoàn tất cài đặt, **Setup Wizard** sẽ tải xuống và cài đặt **các thành phần bổ sung**, bao gồm **Android SDK**. Quá trình này có thể mất một khoảng thời gian tùy thuộc vào tốc độ Internet của bạn. Một số bước có thể trông giống nhau, nhưng hãy kiên nhẫn.
- ✓ **Bước 4:** Khi quá trình tải xuống hoàn tất, **Android Studio** sẽ tự động khởi động, và bạn đã sẵn sàng tạo **dự án đầu tiên**.

Nếu gặp vấn đề trong quá trình cài đặt, hãy kiểm tra **ghi chú phát hành của Android Studio** hoặc nhờ sự trợ giúp từ **giảng viên** của bạn.

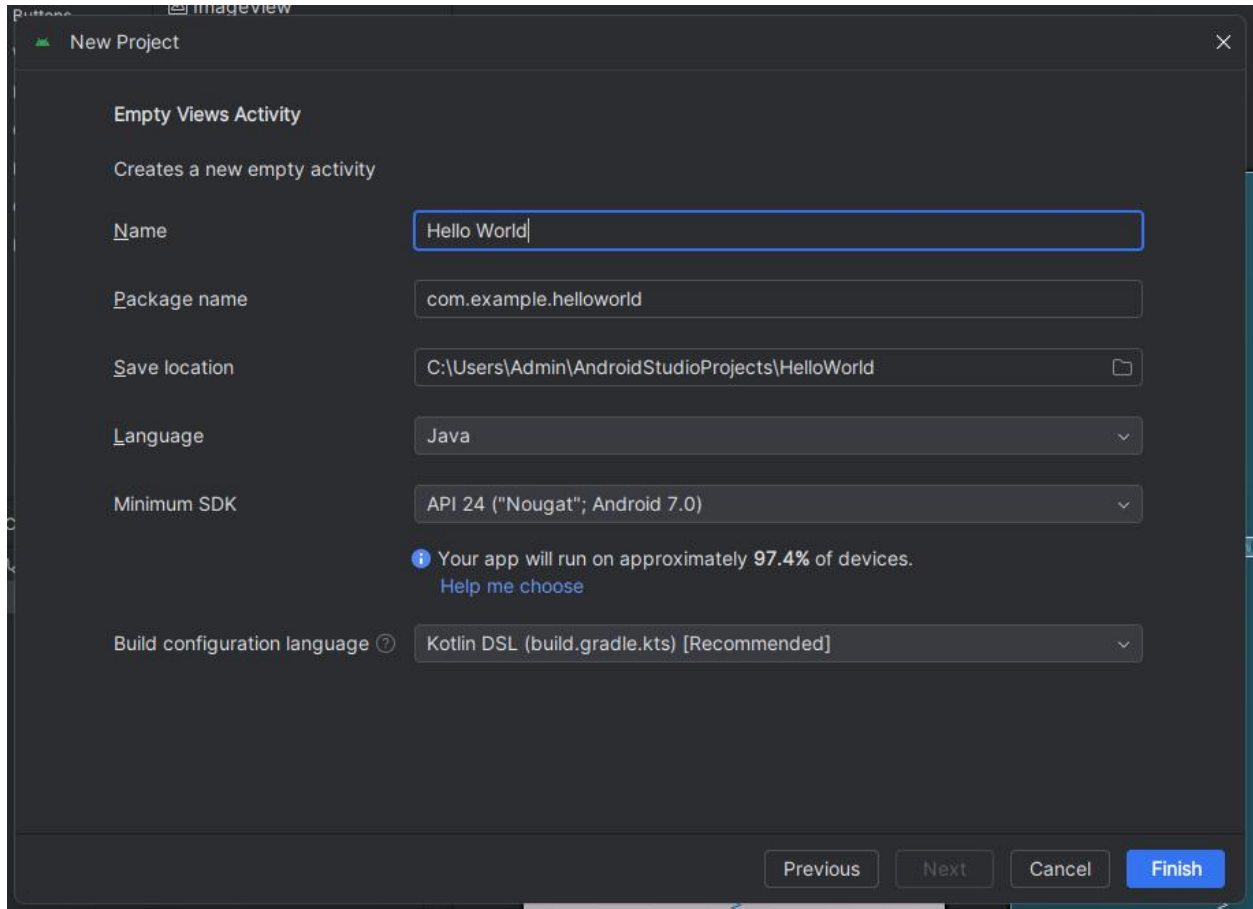
## Nhiệm vụ 2: Tạo ứng dụng Hello World

Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị **"Hello World"** để xác minh rằng **Android Studio** đã được cài đặt đúng cách và làm quen với các bước phát triển ứng dụng cơ bản.

### 2.1 Tạo dự án ứng dụng

- ✓ **Bước 1:** Mở **Android Studio** (nếu chưa mở).
- ✓ **Bước 2:** Trong cửa sổ **Welcome to Android Studio**, nhấp vào **"Start a new Android Studio project"** để tạo một dự án mới.

✓ **Bước 3:** Trong cửa sổ **Create Android Project**, nhập **Hello World** vào ô **Application name**.



✓ **Bước 4:** Xác minh rằng thư mục trong **Project location** là nơi bạn muốn lưu trữ ứng dụng **Hello World** và các dự án Android Studio khác. Nếu cần, hãy thay đổi sang thư mục bạn mong muốn.

✓ **Bước 5:** Chấp nhận giá trị mặc định **android.example.com** cho **Company Domain**, hoặc nhập một tên miền riêng nếu bạn có.

- Nếu không có kế hoạch xuất bản ứng dụng, bạn có thể giữ nguyên mặc định.
- Hãy lưu ý rằng **thay đổi tên package** sau này sẽ tốn công hơn.

✓ **Bước 6:** Bỏ chọn các tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấp **Next**.

✓ **Bước 7:** Ở màn hình **Target Android Devices**, đảm bảo **Phone and Tablet** được chọn.

- Kiểm tra **Minimum SDK** được đặt thành **API 15: Android 4.0.3 IceCreamSandwich**. Nếu chưa đúng, hãy sử dụng menu thả xuống để chọn.

Hoàn tất bước tạo dự án Hello World

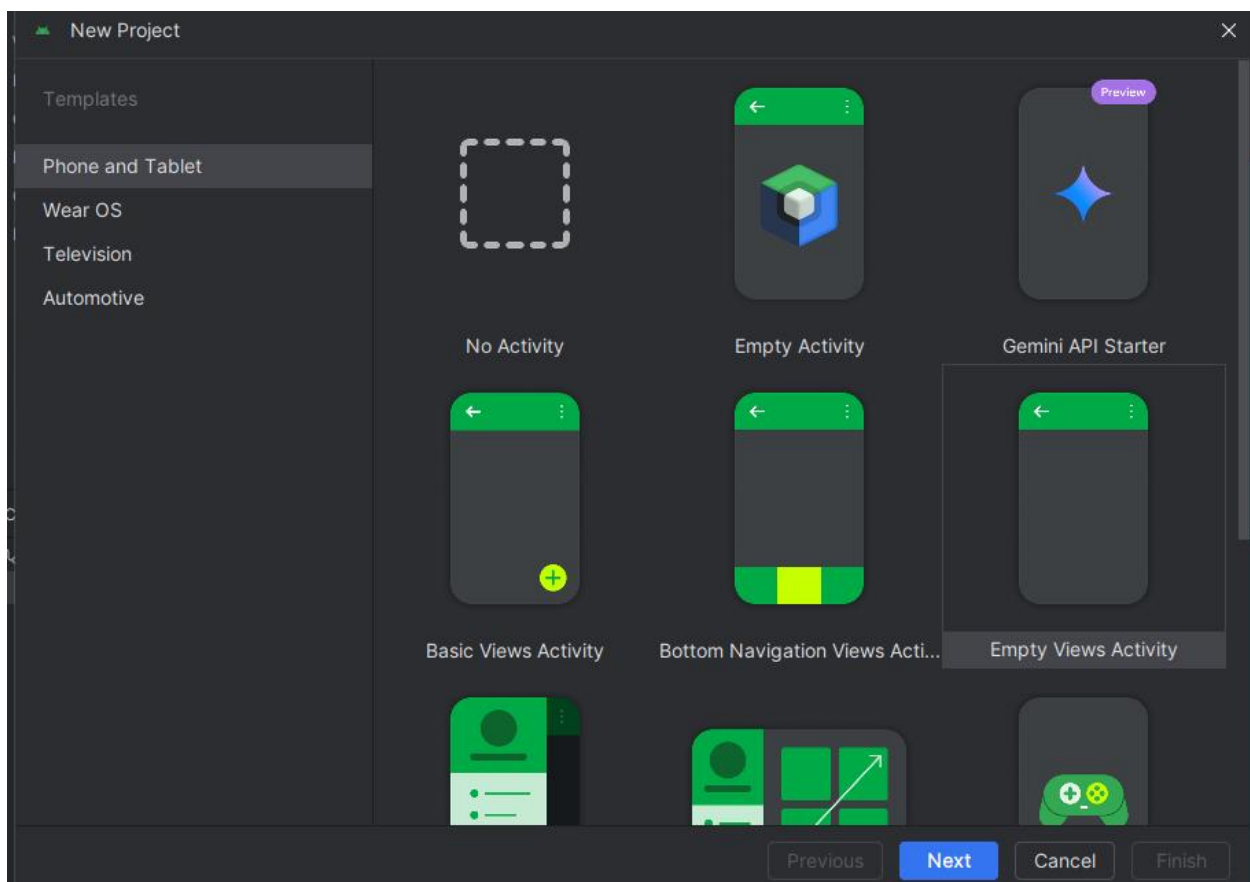
✓ **Bước 8:** Bỏ chọn **Include Instant App support** và tắt cả các tùy chọn khác, sau đó nhấn **Next**.

- Nếu dự án của bạn yêu cầu thêm thành phần cho **SDK đã chọn, Android Studio** sẽ tự động cài đặt.

✓ **Bước 9:** Màn hình **Add an Activity** sẽ xuất hiện.

- Một **Activity** là một phần quan trọng trong bất kỳ ứng dụng Android nào. Nó đại diện cho một tác vụ cụ thể mà người dùng có thể thực hiện.
- **Activity** thường có một **layout** đi kèm để xác định cách hiển thị các thành phần giao diện (**UI elements**) trên màn hình.
- Android Studio cung cấp nhiều mẫu **Activity** để giúp bạn bắt đầu nhanh chóng.

✓ Đối với dự án **Hello World**, hãy chọn **Empty Activity** như trong hình minh họa, sau đó nhấn **Next**.





✓ **Bước 10:** Khi màn hình **Configure Activity** xuất hiện, bạn sẽ thấy các tùy chọn cấu hình khác nhau (tùy thuộc vào mẫu **Activity** bạn đã chọn trước đó).

- Theo mặc định, Activity trống (**Empty Activity**) sẽ được đặt tên là **MainActivity**.
- Bạn có thể đổi tên nếu muốn, nhưng bài học này sử dụng **MainActivity**, vì vậy hãy giữ nguyên mặc định.

✓ **Bước 11:** Đảm bảo rằng tùy chọn **Generate Layout file** được chọn.

- **Tên mặc định** của tệp layout là **activity\_main.xml**.
- Bạn có thể đổi tên, nhưng bài học này sử dụng **activity\_main.xml**, nên hãy giữ nguyên.

✓ **Bước 12:** Đảm bảo rằng tùy chọn **Backwards Compatibility (App Compat)** được chọn.

- Tùy chọn này giúp ứng dụng **tương thích ngược** với các phiên bản Android cũ hơn.

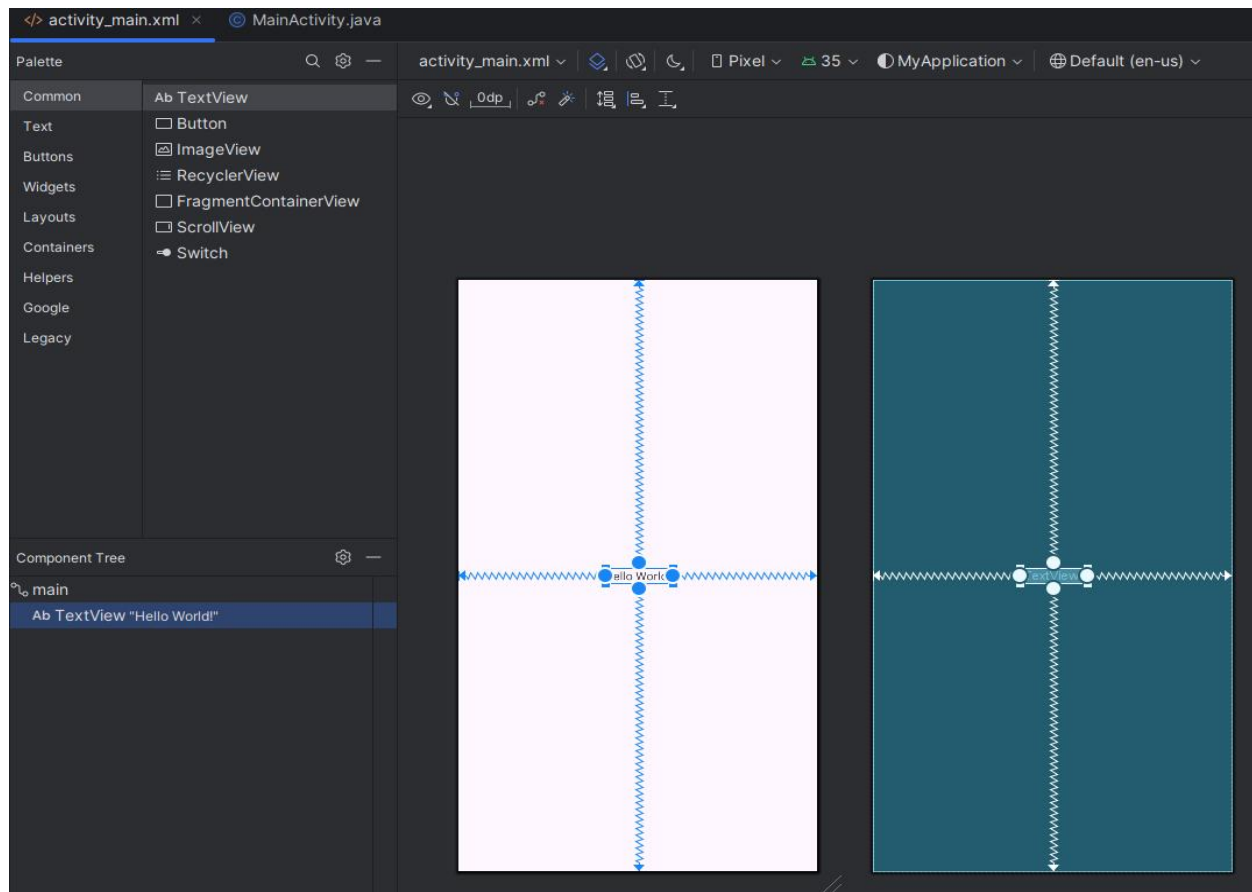
✓ **Bước 13:** Nhấp **Finish** để hoàn tất quá trình tạo dự án.

- **Android Studio** sẽ tạo thư mục dự án và **biên dịch** dự án bằng **Gradle** (quá trình này có thể mất vài phút).
- Nếu xuất hiện hộp thoại "**Tip of the day**", bạn có thể đọc hoặc nhấp **Close** để tắt.

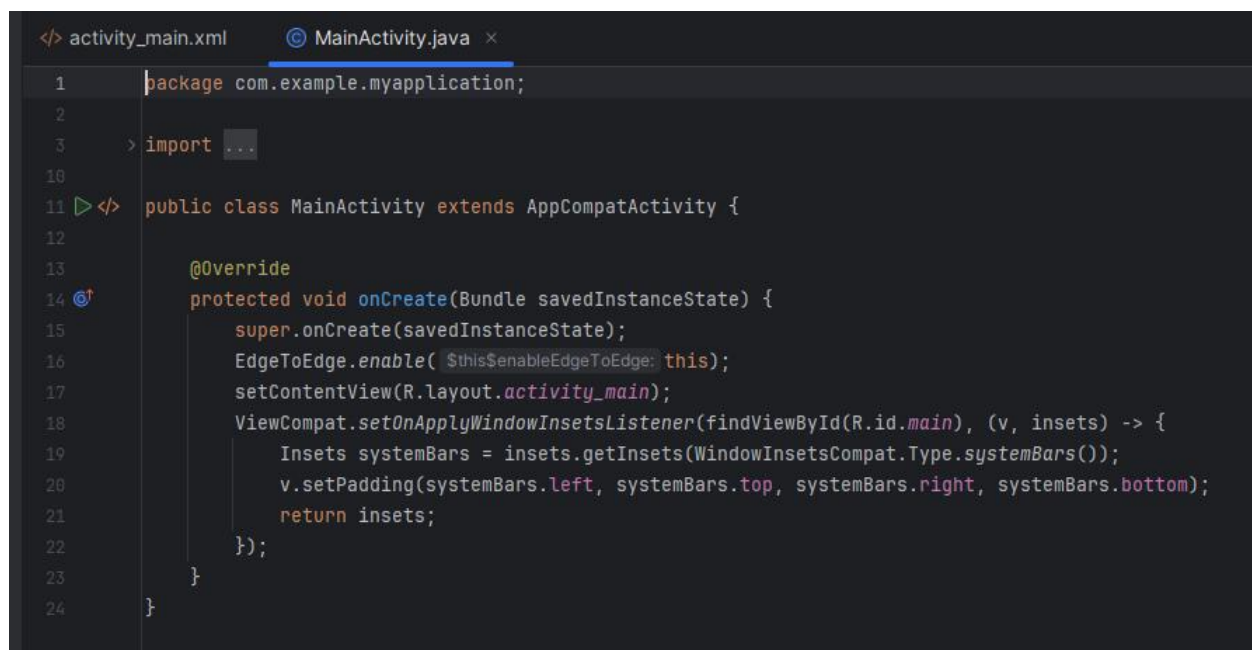
Biên tập viên Android Studio xuất hiện. Thực hiện theo các bước sau:

✓ **Bước 1:** Nhấp vào tab **activity\_main.xml** để mở trình chỉnh sửa layout.

✓ **Bước 2:** Nhấp vào tab **Design** (nếu chưa được chọn) để xem **bản xem trước giao diện đồ họa**.



✓ **Bước 3:** Nhấp vào tab **MainActivity.java** để mở trình chỉnh sửa mã nguồn.



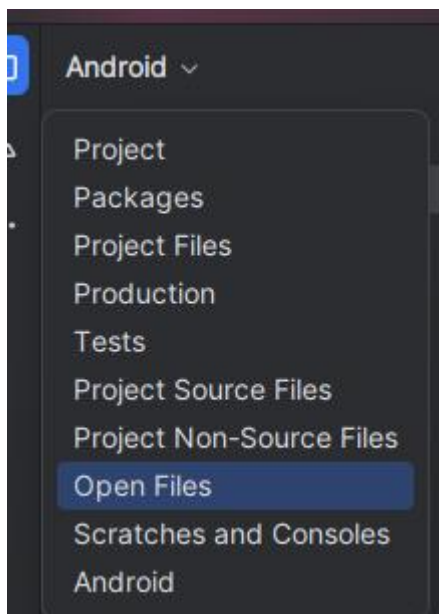
## 2.2 Khám phá mục Project > Android trong Android Studio

Trong phần này, bạn sẽ tìm hiểu cách **tổ chức dự án** trong **Android Studio**.

✓ **Bước 1:** Nếu chưa được chọn, nhấp vào tab **Project** ở **cột tab dọc bên trái** của cửa sổ **Android Studio**.

- Khi đó, khung **Project pane** sẽ xuất hiện, hiển thị cấu trúc của dự án.

✓ **Bước 2:** Ở **menu thả xuống** trên cùng của **Project pane**, chọn **Android** để hiển thị cấu trúc chuẩn của một dự án Android.

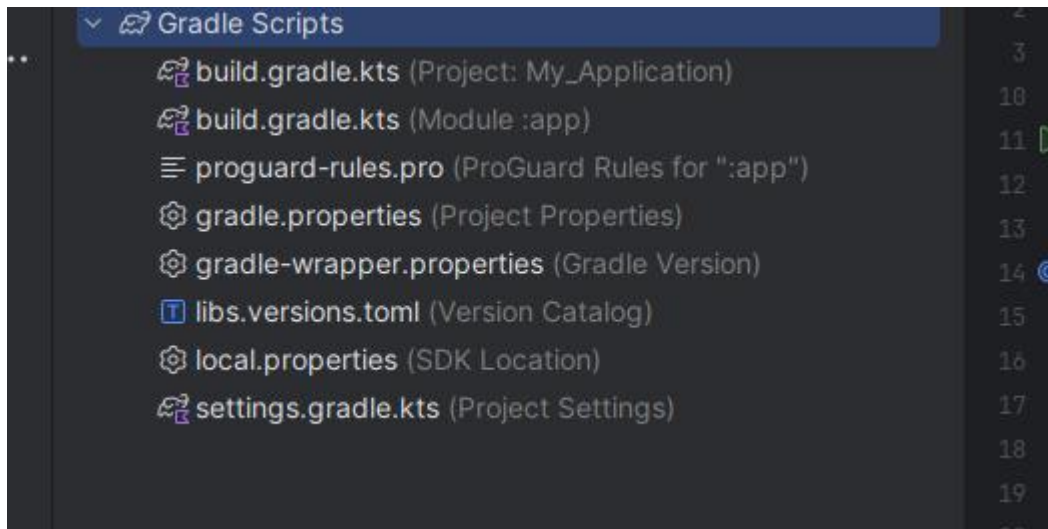


## 2.3 Khám phá thư mục Gradle Scripts trong Android Studio

**Gradle** là hệ thống build của **Android Studio**, giúp bạn dễ dàng thêm **thư viện bên ngoài** hoặc **các module khác** vào dự án của mình.

✓ **Lưu ý:** Khi **Project pane** được đặt ở chế độ **Android**, nó được gọi là **Project > Android pane** trong tài liệu hướng dẫn.

✓ Khi bạn tạo một dự án mới, **Gradle Scripts** sẽ được **mở rộng tự động** trong **Project > Android pane**.



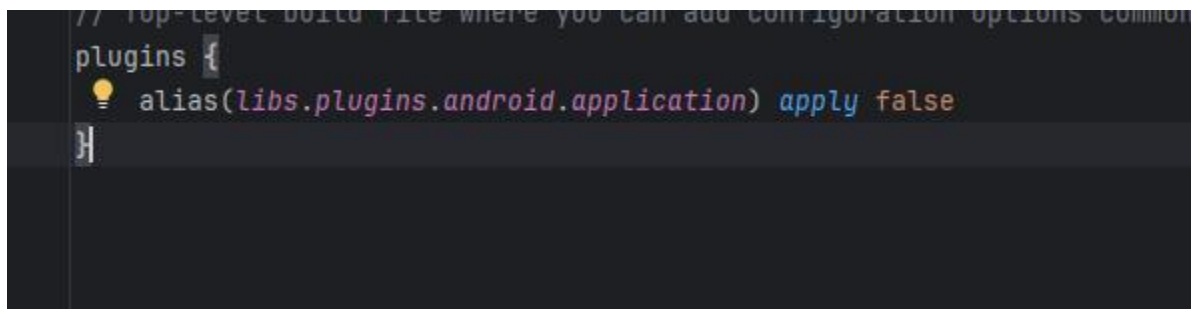
Làm theo các bước sau để tìm hiểu hệ thống **Gradle**:

✓ **Bước 1:** Nếu thư mục **Gradle Scripts** chưa được mở rộng, hãy **nhấp vào biểu tượng tam giác** để mở nó.

- **Thư mục này chứa tất cả các tệp cần thiết** cho hệ thống build.

✓ **Bước 2:** Tìm tệp **build.gradle (Project: HelloWorld)**.

- Đây là nơi chứa các **tùy chọn cấu hình chung** cho **tất cả các module** trong dự án.
- **Mỗi dự án Android Studio** chỉ có **một tệp build Gradle cấp cao nhất**.
- Thông thường, bạn **không cần chỉnh sửa tệp này**, nhưng việc hiểu nội dung của nó sẽ hữu ích.



✓ **Bước 3:** Tìm tệp **build.gradle (Module: app)** trong thư mục **Gradle Scripts**.

- **Tệp này dùng để cấu hình cài đặt build** cho từng module cụ thể (trong dự án HelloWorld, chỉ có một module duy nhất).

- Bạn có thể tùy chỉnh các **tùy chọn đóng gói**, như **build types** (loại build) hoặc **product flavors** (biến thể sản phẩm).
- Có thể **ghi đè** một số cài đặt trong **AndroidManifest.xml** hoặc **build.gradle** cấp dự án.

Sau đây là tệp build.gradle (Module:app) cho ứng dụng Helloworld:

```

1  plugins {
2      alias(libs.plugins.android.application)
3  }
4
5  android {
6      namespace = "com.example.myapplication"
7      compileSdk = 35
8
9      defaultConfig {
10         applicationId = "com.example.myapplication"
11         minSdk = 24
12         targetSdk = 35
13         versionCode = 1
14         versionName = "1.0"
15
16         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
17     }
18
19     buildTypes {
20         release {
21             isMinifyEnabled = false
22             proguardFiles(
23                 getDefaultProguardFile("proguard-android-optimize.txt"),
24                 "proguard-rules.pro"
25             )
26         }
27     }
28     compileOptions {
29         sourceCompatibility = JavaVersion.VERSION_11
30         targetCompatibility = JavaVersion.VERSION_11
31     }
32 }
33
34 dependencies {
35
36     implementation(libs.appcompat)
37     implementation(libs.material)
38     implementation(libs.activity)
39     implementation(libs.constraintlayout)
40     testImplementation(libs.junit)

```

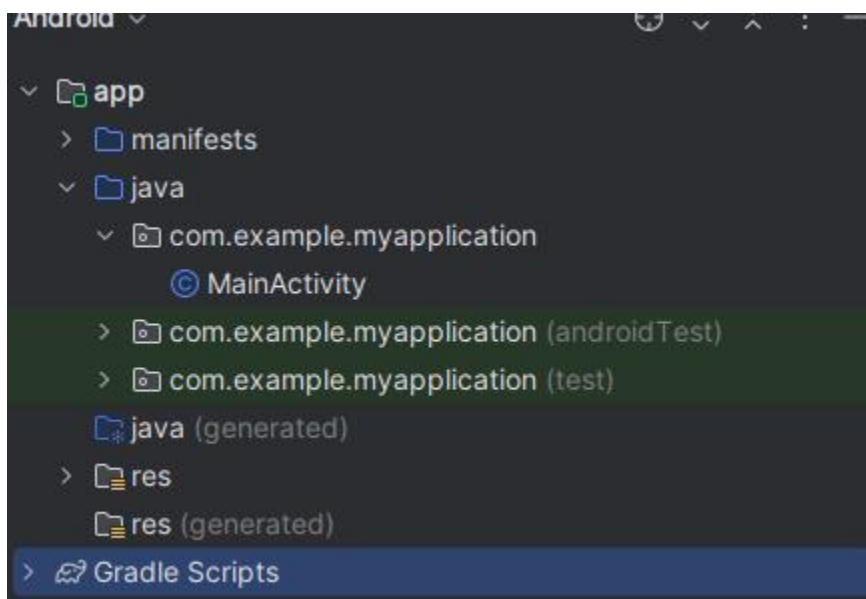
✓ Bước 4: Nhấp vào tam giác để đóng các tập lệnh Gradle

## 2.4 Khám phá thư mục `app` và `res` trong Android Studio

Tất cả **mã nguồn** và **tài nguyên** của ứng dụng đều nằm trong thư mục `app` và `res`.

✓ **Bước 1:** · Mở rộng thư mục **app**, thư mục **java**, và thư mục **com.example.android.helloworld** để xem tệp **MainActivity.java**.

· Nhấp đúp vào tệp để mở nó trong **trình chỉnh sửa mã (code editor)**.



Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như hình minh họa ở trên. Thư mục **com.example.hello.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 để 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**.

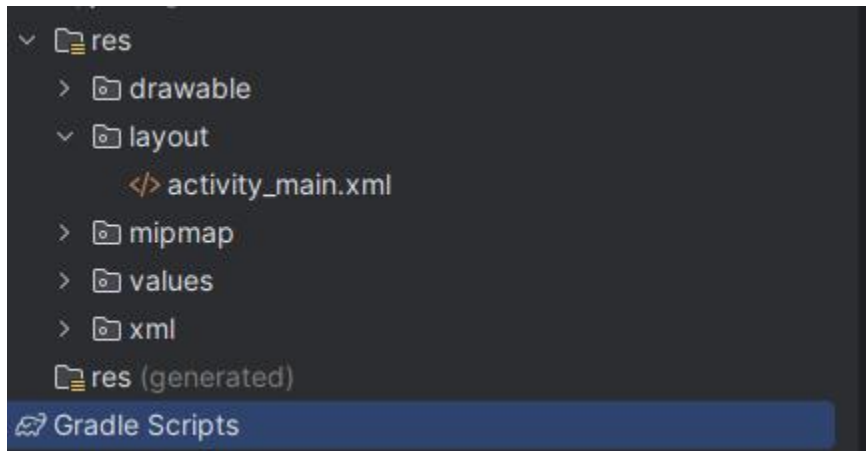
Tên của **Activity đầu tiên** (màn hình đầu tiên mà người dùng nhìn thấy), đồng thời khởi tạo các tài nguyên dùng chung trong ứng dụng, theo thông lệ sẽ được đặt là **MainActivity** (phần mở rộng tệp bị ẩn trong **Project > Android pane**).

✓ **Bước 2:** Mở rộng thư mục **res**, sau đó mở rộng thư mục **layout**, và nhấp đúp vào tệp **activity\_main.xml** để mở nó trong **trình chỉnh sửa giao diện (layout editor)**.

Thư mục **res** chứa các tài nguyên như **bố cục (layouts)**, **chuỗi văn bản (strings)**, và **hình ảnh (images)**.

Một **Activity** thường được liên kết với một **bố cục giao diện người dùng (UI views)**, được định nghĩa trong một tệp **XML**.

Tệp này thường được đặt tên theo **tên của Activity** mà nó liên kết.



Thư mục **res** chứa các tài nguyên như **bố cục (layouts)**, **chuỗi văn bản (strings)**, và **hình ảnh (images)**. Một **Activity** thường được liên kết với một **bố cục giao diện người dùng (UI views)**, được định nghĩa trong một tệp **XML**. Tệp này thường được đặt tên theo **tên của Activity** mà nó liên kết.

## 2.5 Khám phá thư mục manifests

Thư mục **manifests** chứa các tệp cung cấp thông tin quan trọng về ứng dụng cho **hệ thống Android**. Hệ thống cần thông tin này trước khi có thể chạy bất kỳ mã nào của ứng dụng.

### *Các bước thực hiện:*

1 Mở rộng thư mục manifests.

2 Nhấp đúp vào tệp **AndroidManifest.xml** để mở nó.

### *Giới thiệu về **AndroidManifest.xml***

- Đây là tệp mô tả **tất cả các thành phần** của ứng dụng Android.
- Mọi **thành phần** của ứng dụng (chẳng hạn như mỗi **Activity**) **đều phải được khai báo** trong tệp này.
- Trong các bài học tiếp theo, bạn sẽ sửa đổi tệp này để **thêm tính năng và cấp quyền** cho ứng dụng. **Tham khảo thêm:** [App Manifest Overview](#)

### Task 3: Sử dụng thiết bị ảo (emulator)

Trong nhiệm vụ này, bạn sẽ sử dụng Trình quản lý Thiết bị Ảo Android (**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**.

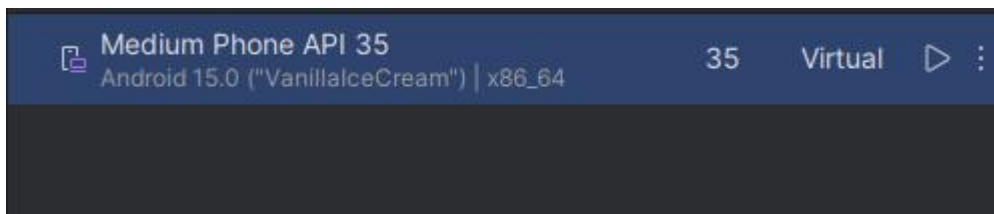
Sử dụng **AVD Manager**, bạn có thể xác định các đặc điểm phần cứng của một thiết bị, cấp độ API, bộ nhớ, giao diện và các thuộc tính khác, sau đó lưu lại dưới dạng một thiết bị ảo. Với các thiết bị ảo, bạn có thể kiểm tra ứng dụng trên các cấu hình thiết bị khác nhau (chẳng hạn như máy tính bảng và điện thoại) với các cấp độ API khác nhau mà không cần sử dụng thiết bị vật lý.

#### 3.1 Tạo một thiết bị ảo Android (AVD)

Để chạy trình giả lập (**emulator**) trên máy tính của bạn, trước tiên bạn cần tạo một cấu hình mô tả thiết bị ảo.

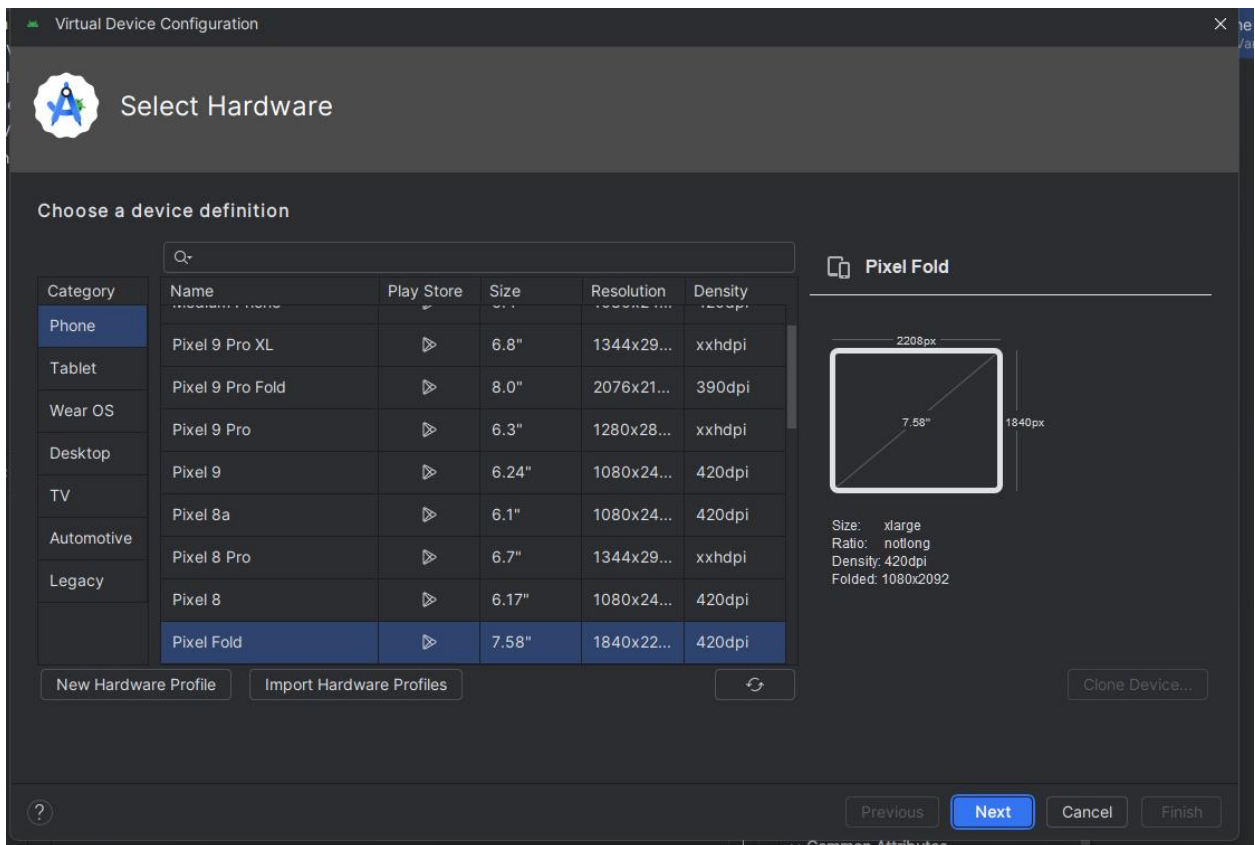
*a) Thực hiện các bước sau:*

1. Trong **Android Studio**, chọn **Tools > Android > AVD Manager**, hoặc nhấp vào biểu tượng **AVD Manager** trên thanh công cụ. Màn hình **Your Virtual Devices** sẽ xuất hiện. Nếu bạn đã tạo thiết bị ảo trước đó, danh sách các thiết bị sẽ hiển thị. Nếu chưa có thiết bị nào, danh sách sẽ trống.

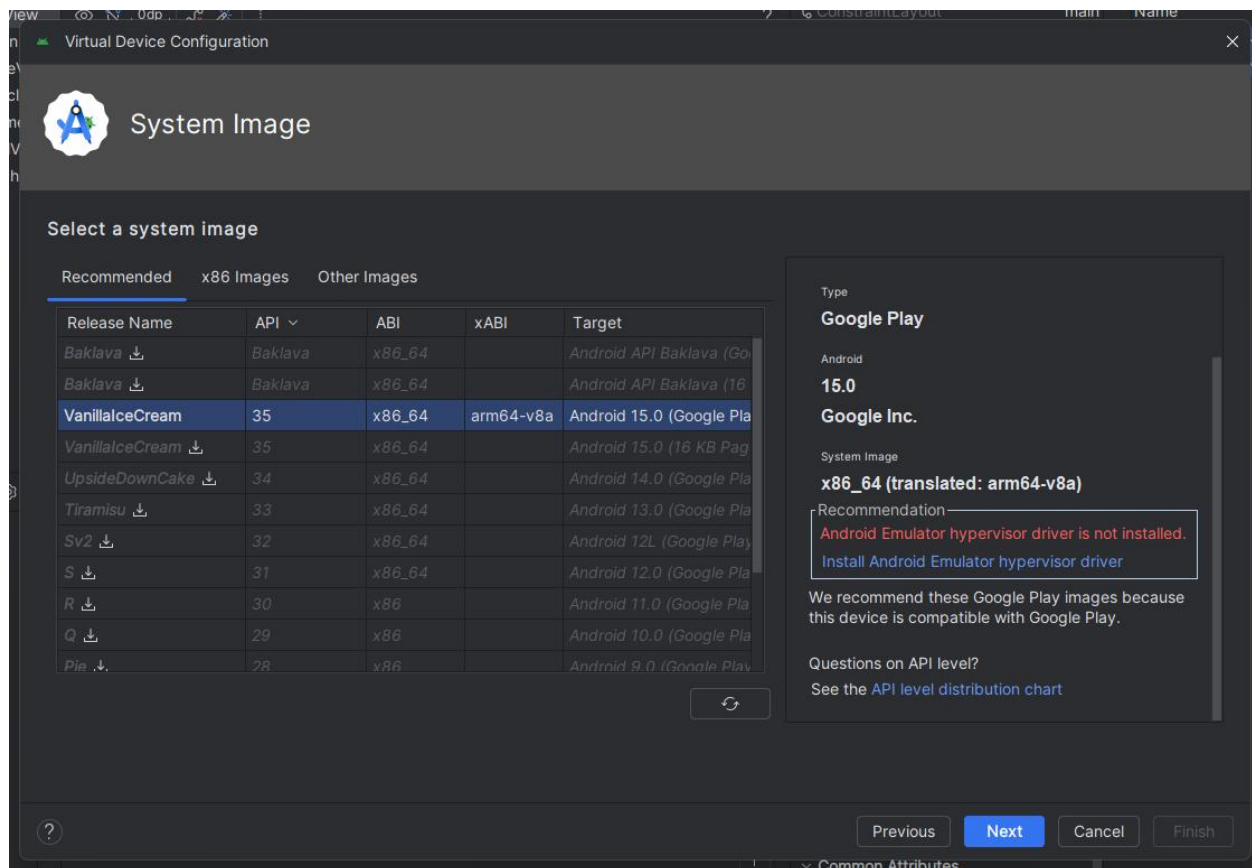


2. Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng cung cấp một cột cho kích thước đường chéo của màn hình (**Size**), độ phân giải màn hình tính bằng pixel (**Resolution**) và mật độ điểm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





3. Chọn một thiết bị như **Nexus 5X** hoặc **Pixel XL**, sau đó nhấp vào **Next**. Màn hình **System Image** sẽ xuất hiện.
4. Nhấp vào tab **Recommended** nếu nó chưa được chọn, sau đó chọn phiên bản hệ điều hành Android để chạy trên thiết bị ảo (chẳng hạn như **Oreo**).



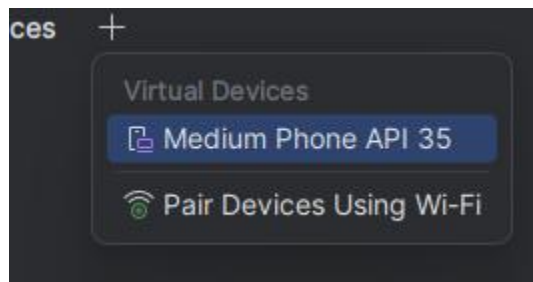
Nếu bạn muốn xem các tùy chọn khác, hãy nhấp vào các tab **Images** và **Other Images**. Nếu có liên kết **Download** bên cạnh một hệ điều hành mà bạn muốn sử dụng, điều đó có nghĩa là nó chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống, sau đó nhấp **Finish** khi quá trình tải xuống hoàn tất.

5. Sau khi chọn hệ điều hành, nhấp vào **Next**. Cửa sổ **Android Virtual Device (AVD)** sẽ xuất hiện. Bạn cũng có thể thay đổi tên của AVD. Kiểm tra lại cấu hình của bạn và nhấp vào **Finish**.

## 3.2 Chạy ứng dụng trên thiết bị ảo

Trong nhiệm vụ này, bạn sẽ chạy ứng dụng **Hello World** của mình trên thiết bị ảo.

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



Trình giả lập khởi động và hoạt động giống như một thiết bị vật lý. Tùy thuộc vào tốc độ của máy tính, quá trình này có thể mất một khoảng thời gian.

Ứng dụng của bạn sẽ được biên dịch, và khi trình giả lập sẵn sàng, **Android Studio** sẽ tải ứng dụng lên trình giả lập và chạy nó.

Bạn sẽ thấy ứng dụng **Hello World** hiển thị như trong hình minh họa sau.



Mẹo: Khi kiểm tra trên một thiết bị ảo, việc khởi động nó một lần ngay từ đầu phiên làm việc là một thực hành tốt. Bạn không nên đóng nó cho đến khi hoàn tất việc kiểm tra ứng dụng của

mình, để ứng dụng của bạn không phải trải qua quá trình khởi động thiết bị lại. Để đóng thiết bị ảo, hãy nhấp vào nút X ở phía trên của trình giả lập, chọn Quit từ menu hoặc nhấn Control-Q trên Windows hoặc Command-Q trên macOS.

## 4.1 Bật gỡ lỗi USB

Để Android Studio có thể giao tiếp với thiết bị của bạn, bạn cần bật **USB Debugging** trên thiết bị Android. Tùy chọn này có thể được kích hoạt trong phần **Developer options** của thiết bị.

Trên **Android 4.2** trở lên, màn hình **Developer options** bị ẩn theo mặc định. Để hiển thị **Developer options** và bật **USB Debugging**, hãy làm theo các bước sau:

1. Trên thiết bị của bạn, mở **Cài đặt (Settings)**, tìm **Giới thiệu về điện thoại (About phone)**, nhấn vào **Giới thiệu về điện thoại**, sau đó chạm vào **Số bản dựng (Build number)** bảy lần.
2. Quay lại màn hình trước (**Cài đặt / Hệ thống**). **Developer options** sẽ xuất hiện trong danh sách. Nhấn vào **Developer options**.
3. Bật **USB Debugging**.

## 4.2 Chạy ứng dụng trên thiết bị thực

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

1. Kết nối thiết bị của bạn với máy tính thông qua cáp USB.
2. Nhấn vào **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**.
4. Android Studio sẽ cài đặt và chạy ứng dụng trên thiết bị của bạn.

---

## Xử lý sự cố (Troubleshooting)

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

- **Rút ra và cắm lại** cáp USB.
- **Khởi động lại** Android Studio.

Nếu máy tính vẫn không nhận diện được thiết bị hoặc hiển thị trạng thái **"unauthorized"**, hãy làm theo các bước sau:

1. **Rút cáp** kết nối thiết bị.
2. Trên điện thoại, mở **Developer Options** trong **Cài đặt**.
3. Nhấn vào **Revoke USB Debugging authorizations** (Thu hồi quyền gỡ lỗi USB).
4. **Kết nối lại** thiết bị với máy tính.
5. Khi có thông báo trên màn hình thiết bị, hãy **cấp quyền** gỡ lỗi USB.

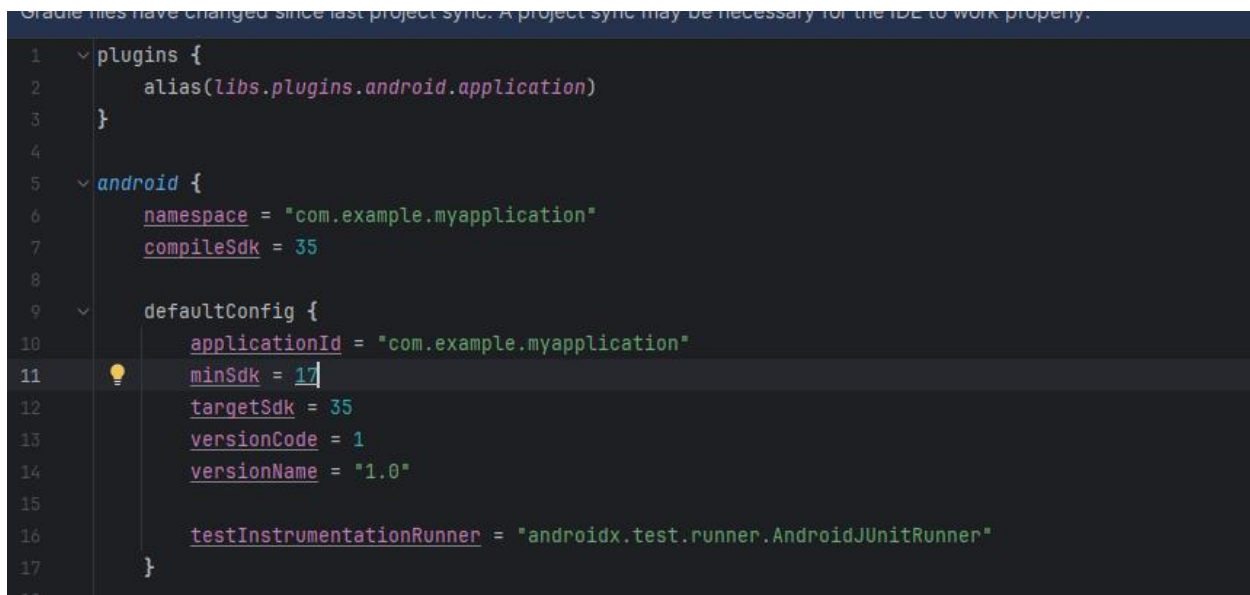
Bạn cũng có thể cần cài đặt **driver USB** phù hợp cho thiết bị. Xem tài liệu **Using Hardware Devices** để biết thêm chi tiết.

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

1. Mở rộng thư mục **Gradle Scripts** (nếu chưa mở), sau đó **double-click** vào tệp `build.gradle` (Module: `app`).

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

2. Trong khối `defaultConfig`, thay đổi giá trị của `minSdkVersion` thành 17 như sau:



The screenshot shows a code editor with the following content:

```
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

1  plugins {
2      alias(libs.plugins.android.application)
3  }
4
5  android {
6      namespace = "com.example.myapplication"
7      compileSdk = 35
8
9      defaultConfig {
10         applicationId = "com.example.myapplication"
11         minSdk = 17
12         targetSdk = 35
13         versionCode = 1
14         versionName = "1.0"
15
16         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
17     }
18 }
```

Trình chỉnh sửa mã hiển thị một thanh thông báo ở đầu với liên kết **Sync Now**.

## 5.2 Đồng bộ cấu hình Gradle mới

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

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

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

Để có cái nhìn sâu hơn về Gradle, hãy xem tài liệu **Build System Overview** và **Configuring Gradle Builds**.

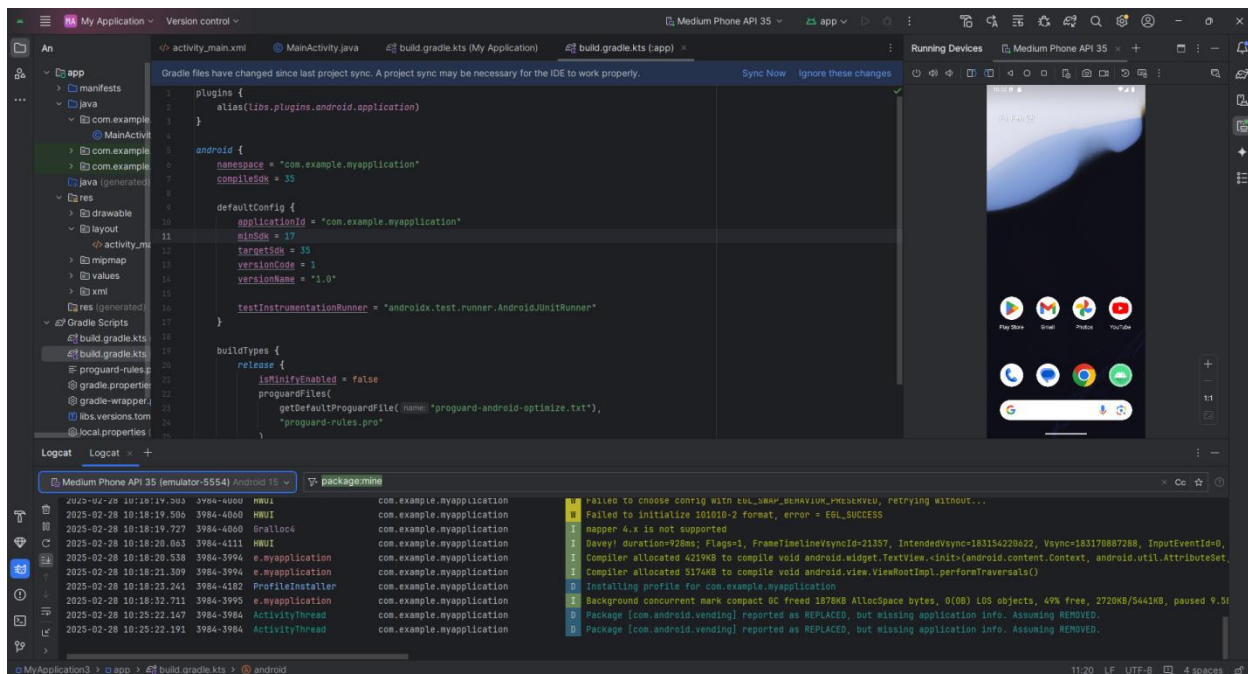
## Nhiệm vụ 6: Thêm câu lệnh log vào ứng dụng của bạn

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh **Log** vào ứng dụng của mình, các câu lệnh này sẽ hiển thị thông báo trong bảng **Logcat**.

Thông báo log là một công cụ gỡ lỗi mạnh mẽ mà bạn có thể sử dụng để kiểm tra giá trị, luồng thực thi và báo cáo ngoại lệ.

### 6.1 Xem bảng Logcat

Để xem bảng **Logcat**, hãy nhấp vào tab **Logcat** ở phía dưới cửa sổ **Android Studio**, như hiển thị trong hình bên dưới.



## Trong hình trên:

1. **Tab Logcat** để mở và đóng bảng **Logcat**, hiển thị thông tin về ứng dụng của bạn khi nó đang chạy. Nếu bạn thêm các câu lệnh **Log** vào ứng dụng, các thông báo **Log** sẽ xuất hiện ở đây.
2. **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 câu lệnh log vào ứng dụng của bạn

Các câu lệnh **log** trong mã ứng dụng của bạn sẽ hiển thị thông báo trong bảng **Logcat**. Ví dụ:

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

Các phần của thông điệp là:

- **Log:** Lớp Log để gửi thông điệp log đến bảng Logcat.
- **d:** Mức Debug Log để lọc hiển thị thông điệp log trong bảng Logcat. Các mức log khác bao gồm e cho Error, w cho Warn và i cho Info.
- **"MainActivity":** Đối số đầu tiên là một tag có thể được sử dụng để lọc thông điệp trong bảng Logcat. Đây thường là tên của Activity mà từ đó thông điệp bắt nguồn. Tuy nhiên, bạn có thể đặt bất kỳ tên nào hữu ích cho việc debug.

Theo quy ước, các tag log được định nghĩa là hằng số cho Activity.

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

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

Thực hiện các bước sau:

1. Mở ứng dụng **Hello World** của bạn trong **Android Studio**, sau đó mở **MainActivity**.
2. Để tự động thêm các import không mở hồ vào dự án của bạn (chẳng hạn như `android.util.Log` cần thiết để sử dụng Log), chọn **File > Settings** trong **Windows**, hoặc **Android Studio > Preferences** trong **macOS**.
3. Chọn **Editor > General > Auto Import**. Đánh dấu chọn tất cả các ô và đặt **Insert imports on paste** thành **All**.
4. Nhấp **Apply**, sau đó nhấp **OK**.
5. Trong phương thức `onCreate()` của **MainActivity**, thêm dòng lệnh sau:

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

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

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d("MainActivity", "Hello World");
}
```

6. Nếu ngăn Logcat chưa mở, nhấp vào tab **Logcat** ở dưới cùng của Android Studio để mở nó.
7. Kiểm tra xem tên của mục tiêu và tên gói của ứng dụng có chính xác không.



8. Thay đổi **Log level** trong ngăn Logcat thành **Debug** (hoặc giữ nguyên **Verbose** vì có rất ít thông báo log).

9 Chạy ứng dụng của bạn.

Thông báo sau sẽ xuất hiện trong ngăn **Logcat**:

```
11- 24 14:06:59.001 4696- 4696/? D/MainActivity: Hello World
```

### Thử thách lập trình

Lưu ý: Tất cả các thử thách lập trình 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 xong 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. Thay 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 người mà bạn đã quên chúc mừng sinh nhật.
4. Một cách sử dụng phổ biến của lớp Log là ghi log các ngoại lệ Java khi chúng xảy ra trong chương trình của bạn. Có một số phương thức hữu ích, chẳng hạn như `Log.e()`, mà bạn có thể sử dụng cho mục đích này. Hãy khám phá các phương thức có thể sử dụng để bao gồm một ngoại lệ trong thông điệp Log. Sau đó, viết mã trong ứng dụng của bạn để kích hoạt và ghi log một ngoại lệ.

### Tóm tắt

- Để cài đặt Android Studio, truy cập **Android Studio** và làm theo hướng dẫn để tải xuống và cài đặt.

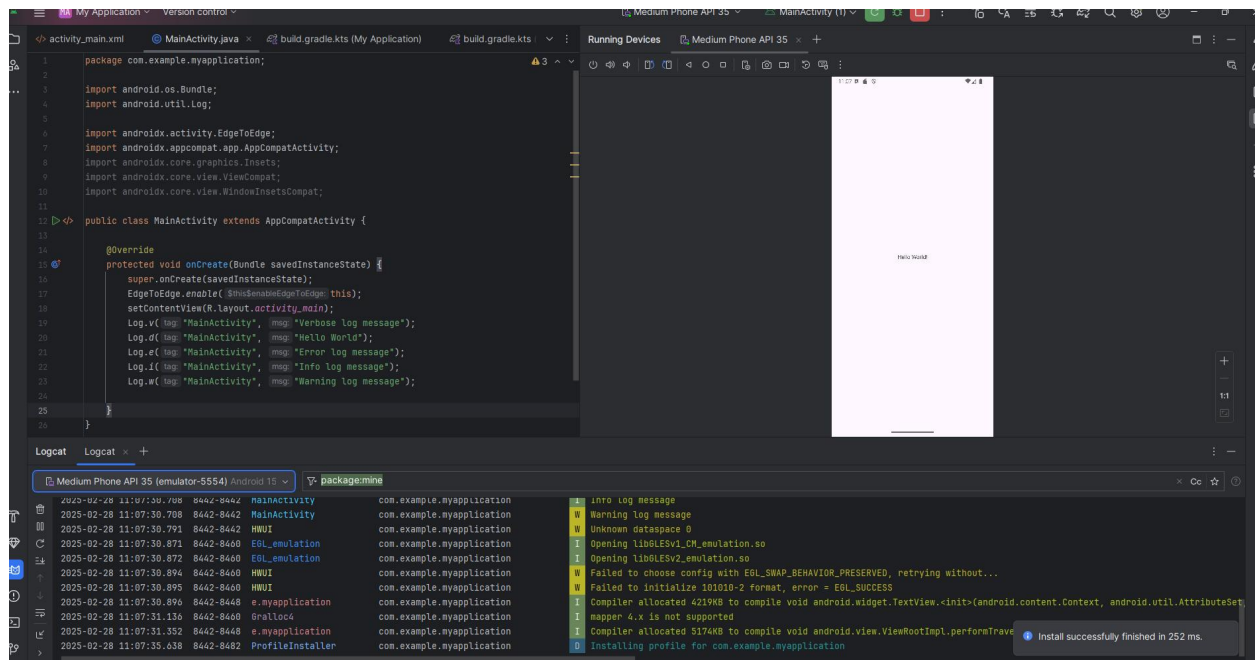
- Khi tạo một ứng dụng mới, đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm **Minimum SDK**.
- Để xem hệ thống phân cấp Android của ứng dụng trong **Project pane**, nhấp vào tab **Project** trong cột tab dọc, sau đó chọn **Android** trong menu bật lên ở trên cùng.
- Chỉnh sửa tệp **build.gradle(Module:app)** khi bạn cần thêm thư viện mới vào dự án hoặc thay đổi phiên bản thư viện.
- Tất cả mã và tài nguyên của ứng dụng được lưu trữ trong các thư mục **app** và **res**. Thư mục **java** chứa các **activity**, **test**, và các thành phần khác trong mã nguồn Java. Thư mục **res** lưu trữ các tài nguyên như **layouts**, **strings**, và **hình ảnh**.
- Chỉnh sửa tệp **AndroidManifest.xml** để thêm các **thành phần**, **tính năng**, và **quyền hạn (permissions)** vào ứng dụng Android của bạn. Mọi thành phần của ứng dụng, chẳng hạn như nhiều **activity**, phải được khai báo trong tệp XML này.
- Sử dụng **Android Virtual Device (AVD) manager** để tạo một thiết bị ảo (**emulator**) nhằm chạy ứng dụng của bạn.
- Thêm các **Log statements** vào ứng dụng để hiển thị thông báo trong **Logcat pane**, đây là công cụ cơ bản để gỡ lỗi (**debugging**).
- Để chạy ứng dụng trên **thiết bị Android vật lý** bằng **Android Studio**, hãy bật **USB Debugging** trên thiết bị.
  - Mở **Cài đặt (Settings)** > **Giới thiệu điện thoại (About phone)** và nhấn **Build number** bảy lần.
  - Quay lại màn hình trước (**Cài đặt**), nhấn vào **Tùy chọn nhà phát triển (Developer options)** và chọn **Gỡ lỗi USB (USB Debugging)**.

## Bài tập về nhà

### Xây dựng và chạy một ứng dụng

- Tạo một dự án Android mới từ mẫu **Empty Template**.
- Thêm các câu lệnh ghi log với các mức log khác nhau trong phương thức **onCreate()** của **MainActivity**.
- Tạo một trình giả lập (**emulator**) cho một thiết bị, chọn bất kỳ phiên bản Android nào bạn muốn, và chạy ứng dụng.

- Sử dụng bộ lọc trong **Logcat** để tìm các câu lệnh log của bạn và điều chỉnh mức hiển thị để chỉ hiển thị các log ở mức **debug** hoặc **error**.



## Trả lời các câu hỏi

### Câu hỏi 1

Tên của tệp layout cho **main activity** là gì?

- MainActivity.java**
- AndroidManifest.xml**
- activity\_main.xml**
- build.gradle**

### Câu hỏi 2

Tên của **resource string** xác định tên ứng dụng là gì?

- app\_name**
- xmlns:app**
- android:name**
- applicationId**

### Câu hỏi 3

Công cụ nào được sử dụng để tạo một 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 log sau đây:

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

- ✓ **Verbose**
- ✓ **Debug**
- ✓ **Info**

### Giải thích:

Hàm `Log.i()` tương ứng với mức **Info** trong hệ thống ghi log của Android. Các mức **Logcat** hoạt động như sau:

- **Verbose (V)** → Hiển thị tất cả các mức (Verbose, Debug, Info, Warn, Error, Assert).
- **Debug (D)** → Hiển thị Debug, Info, Warn, Error, Assert.
- **Info (I)** → Hiển thị Info, Warn, Error, Assert.
- **Warn (W)** → Chỉ hiển thị Warn, Error, Assert. (**Không hiển thị** `Log.i()`)
- **Error (E)** → Chỉ hiển thị Error, Assert. (**Không hiển thị** `Log.i()`)
- **Assert (A)** → Chỉ hiển thị Assert. (**Không hiển thị** `Log.i()`)

Vì `Log.i()` thuộc mức **Info**, nên nó chỉ xuất hiện khi **Logcat** được đặt ở **Verbose, Debug hoặc Info**, nhưng sẽ không hiển thị ở mức **Warn, Error hoặc Assert**.

## 1.2) Giao diện người dùng tương tác đầu tiên

### Giới thiệu

Giao diện người dùng (**UI**) xuất hiện trên màn hình của 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 giao diện người dùng (**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 (TextEntryField)**.

Một số **lớp con của View** thường được sử dụng sẽ được mô tả trong 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 phần tử có thể nhấp khác (chẳng hạn 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 phần tử **View** khác và định vị chúng.

Mã Java hiển thị và điều khiển giao diện người dùng (**UI**) được chứa trong một lớp kế thừa **Activity**.

Một **Activity** thường được liên kết với một bố cục UI của các **View**, được xác định trong một tệp **XML (eXtended Markup Language)**.

Tệp XML này thường được đặt tên theo **Activity** và định nghĩa bố cục của các phần tử **View** trên màn hình.

Ví dụ: Mã **MainActivity** trong ứng dụng **Hello World** hiển thị một bố cục được xác định trong tệp **activity\_main.xml**, trong đó có một **TextView** chứa văn bản "Hello World".

Trong các ứng dụng phức tạp hơn, một **Activity** có thể triển khai các hành động để phản hồi thao tác 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 **Activity** trong bài học khác.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên—một ứng dụng cho phép người dùng tương tác.

Bạn sẽ tạo một ứng dụng sử dụng mẫu **Empty Activity**. Bạn cũng sẽ học cách sử dụng **trình**

**chỉnh sửa bố cục** để thiết kế bố cục và chỉnh sửa bố cục trong **XML**.

Bạn cần phát triển những kỹ năng này để hoàn thành các bài thực hành khác trong khóa học này.

---

## Những gì bạn nên biết trước

Bạn nên quen thuộc với:

- Cách cài đặt và mở **Android Studio**.
- Cách tạo ứng dụng **Hello World**.
- Cách chạy ứng dụng **Hello World**.

## Những gì bạn sẽ học

- Cách tạo một ứng dụng với hành vi tương tác.
- Cách sử dụng **trình chỉnh sửa bố cục** để thiết kế giao diện.
- Cách chỉnh sửa bố cục trong **XML**.
- Nhiều thuật ngữ mới. Hãy xem phần **từ vựng và thuật ngữ** để có các định nghĩa dễ hiểu.

## Những gì bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử **Button** cùng một **TextView** vào bố cục.
- Điều chỉnh từng phần tử trong **ConstraintLayout** để căn chỉnh chúng theo lề và các phần tử khác.
- Thay đổi thuộc tính của các phần tử UI.
- Chỉnh sửa bố cục ứng dụng trong **XML**.
- Trích xuất các chuỗi mã cứng thành tài nguyên chuỗi (**string resources**).
- Triển khai phương thức **xử lý sự kiện nhấp** để hiển thị thông báo trên màn hình khi người dùng nhấn vào từng **Button**.

## Tổng quan về ứng dụng

Ứng dụng **HelloToast** bao gồm hai phần tử **Button** và một **TextView**.

- Khi người dùng nhấn vào nút đầu tiên, nó hiển thị một thông báo ngắn (**Toast**) trên màn hình.
- Khi người dùng nhấn vào nút thứ hai, một bộ đếm **số lần nhấn** sẽ tăng lên và được hiển thị trong **TextView**, bắt đầu từ số **0**.

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

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

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

#### 1.1 Tạo dự án Android Studio

14. Mở **Android Studio** và tạo một dự án mới với các tham số sau:

Attribute	Value
Application Name	<b>Hello Toast</b>
Company Name	<b>com.example.android</b> (or your own domain)
Phone and Tablet Minimum SDK	<b>API15: Android 4.0.3 IceCreamSandwich</b>
Template	<b>Empty Activity</b>

Generate Layout file box	Selected
Backwards Compatibility box	Selected

15. 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.

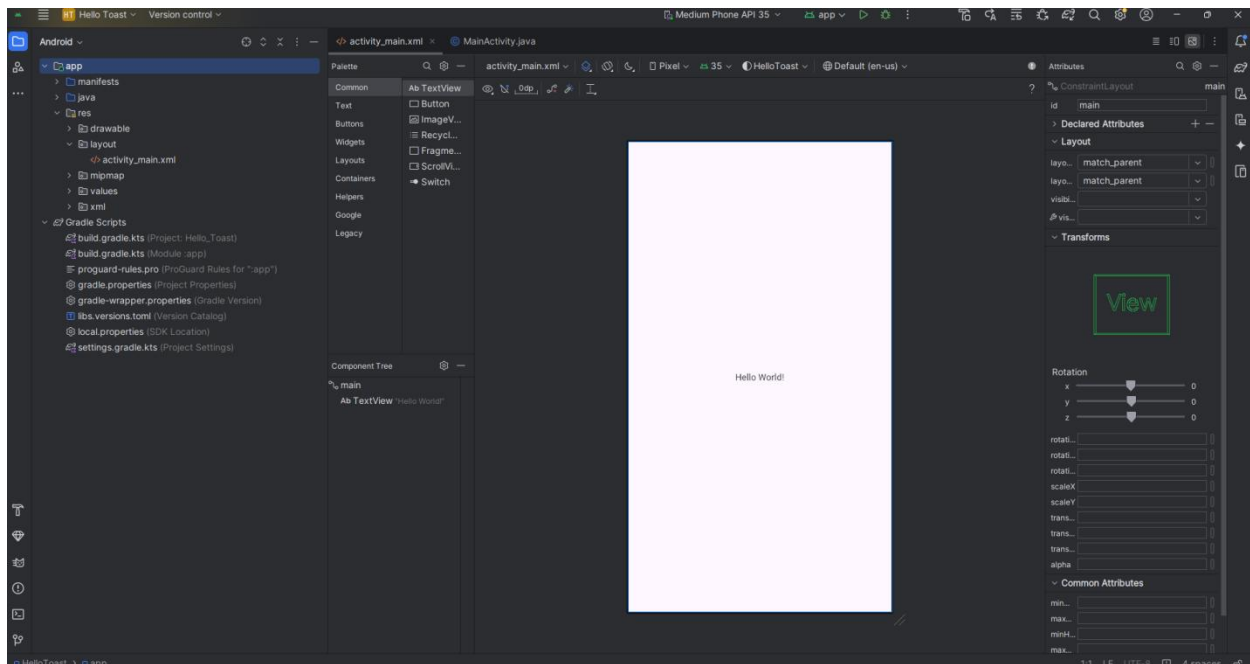
## 1.2 Khám phá trình chỉnh sửa bố cục

Android Studio cung cấp **trình chỉnh sửa bố cục** (layout editor) để nhanh chóng xây dựng bố cục giao diện người dùng (UI) của ứng dụng. Công cụ này cho phép bạn **kéo thả các phần tử** vào chế độ xem thiết kế trực quan hoặc bản thiết kế, **định vị chúng trong bố cục, thêm ràng buộc**, và **thiết lập thuộc tính**.

**Ràng buộc (Constraints)** xác định **vị trí của phần tử UI** trong bố cục. Một ràng buộc đại diện cho **kết nối hoặc căn chỉnh** với một **view khác, bố cục cha**, hoặc một **đường hướng dẫn vô hình**.

Hãy khám phá trình chỉnh sửa bố cục và tham khảo hình minh họa khi thực hiện các bước:





- Trong thư mục **app > res > layout** ở bảng **Project > Android**, nhấp đúp vào file **activity\_main.xml** để mở nếu nó chưa được mở.
- Nhấp vào tab **Design** nếu nó chưa được chọn. Bạn sử dụng tab **Design** để thao tác các phần tử và bố cục, và tab **Text** để chỉnh sửa mã XML cho bố cục.
- Bảng **Palettes** hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục ứng dụng của mình.
- Bảng **Component Tree** hiển thị **cấu trúc phân cấp của các phần tử UI**. Các phần tử giao diện người dùng được tổ chức thành một cây phân cấp với **cha và con**, trong đó phần tử con kế thừa thuộc tính của phần tử cha. Trong hình minh họa, **TextView** là phần tử con của **ConstraintLayout**.
- Các bảng thiết kế và bản thiết kế trong trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình minh họa, bố cục chỉ hiển thị một phần tử: một **TextView** hiển thị dòng chữ "Hello World".
- Tab **Attributes** hiển thị bảng **thuộc tính** để cài đặt các thuộc tính cho một phần tử UI.

## 2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện các bước sau:

1. Mở **activity\_main.xml** từ bảng **Project > Android** nếu nó chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào nó.

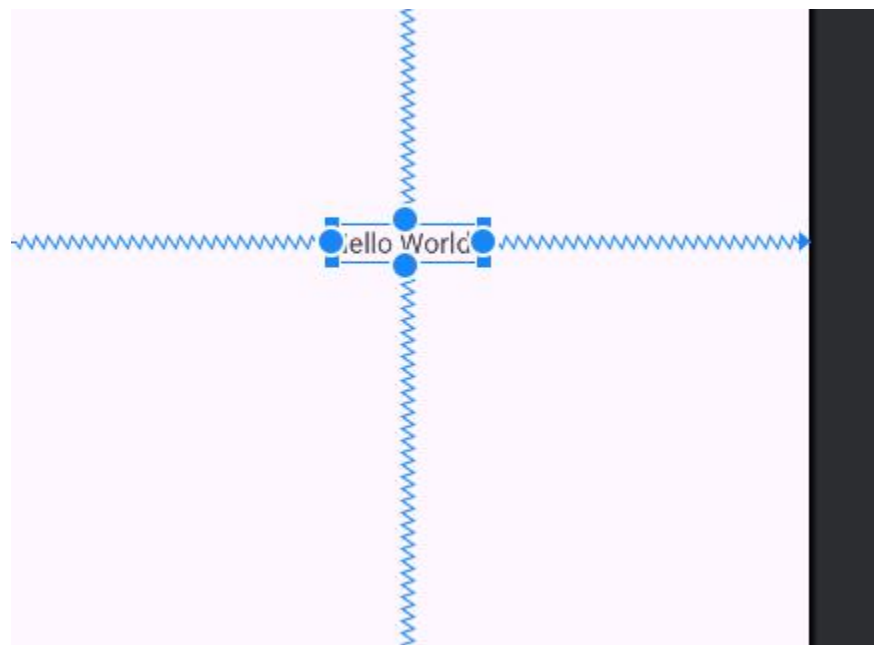
Nếu không có bản thiết kế (**Blueprint**), hãy nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**.

2. **Công cụ Autoconnect** cũng nằm trên thanh công cụ và được bật theo mặc định. Đối với bước này, hãy đảm bảo rằng công cụ không bị tắt.

3. Nhấp vào nút **zoom in** để phóng to bảng thiết kế và bản thiết kế để xem chi tiết hơn.

4. Trong bảng **Component Tree**, chọn **TextView.TextView "Hello World"** sẽ được tô sáng trong bảng thiết kế và bản thiết kế. Các ràng buộc (**constraints**) của phần tử này sẽ hiển thị.

5. Làm theo hướng dẫn trong hình động bên dưới cho bước này. Nhấp vào **chấm tròn** bên phải của **TextView** để xóa ràng buộc ngang đang liên kết nó với cạnh phải của bố cục. **TextView** sẽ nhảy về phía bên trái vì nó không còn bị ràng buộc với cạnh phải nữa. Để thêm lại ràng buộc ngang, nhấp vào cùng chấm tròn đó và kéo một đường đến cạnh phải của bố cục.



Trong bảng **Blueprint** hoặc **Design**, các tay cầm (**handles**) sau xuất hiện trên phần tử **TextView**:

**Constraint handle (Tay cầm ràng buộc):**

- Để tạo một ràng buộc như trong hình động trên, nhấp vào một **tay cầm ràng buộc** (hiển thị dưới dạng một vòng tròn ở cạnh phần tử).
- Sau đó, kéo tay cầm đó đến một **tay cầm ràng buộc khác** hoặc đến **đường viền của phần tử cha**.
- Một đường **zigzag** sẽ xuất hiện để đại diện cho ràng buộc.

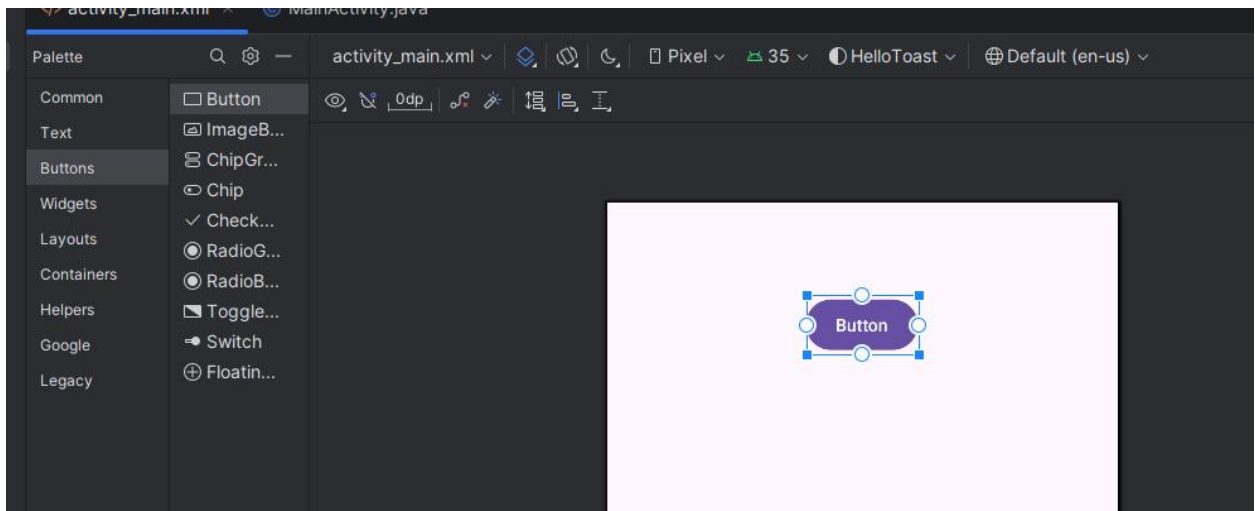
### **Resizing handle (Tay cầm thay đổi kích thước):**

- Để thay đổi kích thước phần tử, kéo **tay cầm thay đổi kích thước** (hiển thị dưới dạng hình vuông).
- Khi bạn kéo, tay cầm này sẽ chuyển thành một **góc vát** để dễ dàng nhận biết.

## **2.2 Thêm một Button vào bố cục**

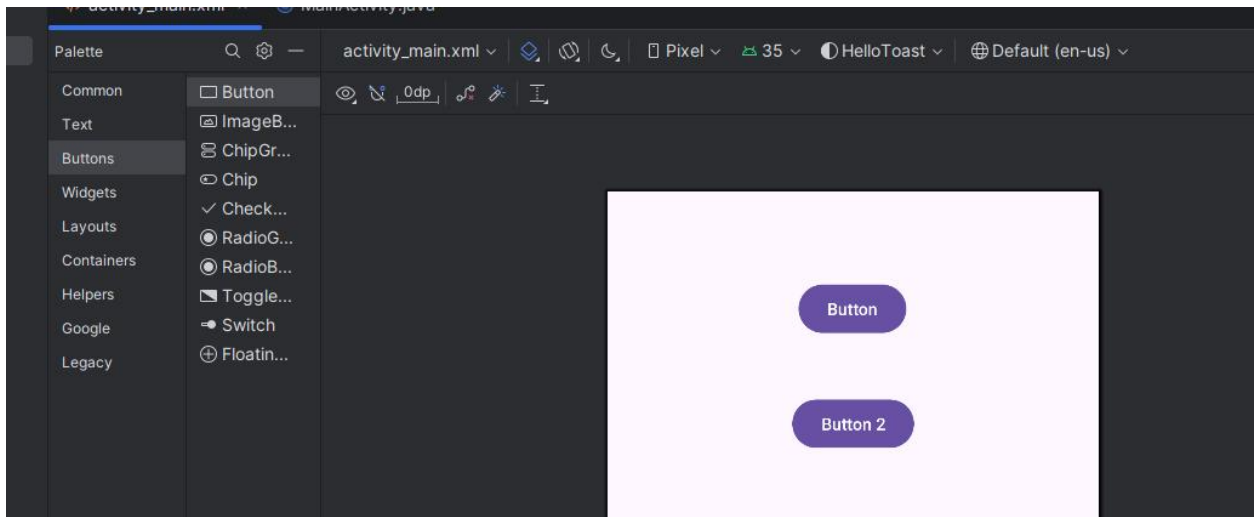
Khi được bật, công cụ Autoconnect sẽ tự động tạo hai hoặc nhiều ràng buộc cho một phần tử giao diện người dùng (UI element) đối với bố cục cha. Sau khi bạn kéo phần tử vào bố cục, nó sẽ tạo các ràng buộc dựa trên vị trí của phần tử đó. Thực hiện các bước sau để thêm một Button:

1. Bắt đầu với một bố cục trống. Phần tử TextView không còn cần thiết, vì vậy khi nó vẫn được chọn, nhấn phím Delete hoặc chọn **Edit > Delete**. Bạn bây giờ có một bố cục hoàn toàn trống.
2. Kéo một Button từ ngăn **Palette** vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả Button vào khu vực trung tâm phía trên của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo ràng buộc đến phía trên, bên trái và bên phải của bố cục như được hiển thị trong hình động bên dưới.



### 2.3 Thêm một Button thứ hai vào bố cục

1. Kéo một Button khác từ ngăn **Palette** vào giữa bố cục như được hiển thị trong hình động bên dưới. Công cụ **Autoconnect** có thể tự động cung cấp các ràng buộc ngang cho bạn (nếu không, bạn có thể kéo chúng theo cách thủ công).
2. Kéo một ràng buộc dọc đến cạnh dưới 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 chuyển con trỏ chuột qua để hiển thị nút **Clear Constraints**. Nhấp vào nút này để xóa tất cả ràng buộc của phần tử đã chọn. Để xóa một ràng buộc cụ thể, hãy nhấp vào tay cầm ràng buộc cụ thể.

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 của phần tử giao diện người dùng

Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng. Bạn có thể tìm thấy các thuộc tính (được gọi là **properties**) chung cho tất cả các phần tử trong tài liệu của lớp **View**.

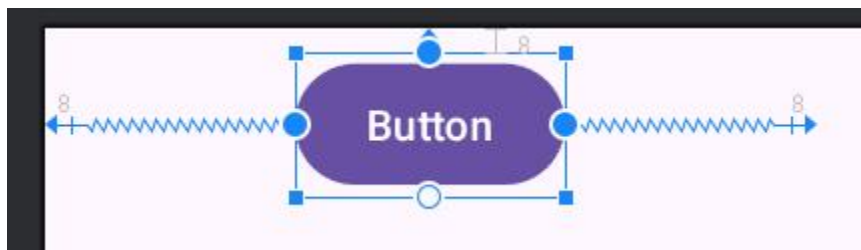
Trong nhiệm vụ này, bạn sẽ nhập giá trị mới và thay đổi giá trị của các thuộc tính quan trọng của **Button**, những thuộc tính này có thể áp dụng cho hầu hết các loại View.

#### 3.1 Thay đổi kích thước Button

Trình chỉnh sửa bố cục (**Layout Editor**) cung cấp các tay cầm thay đổi kích thước ở cả bốn góc của một **View**, giúp bạn có thể thay đổi kích thước nhanh chóng. Bạn có thể kéo các tay cầm ở mỗi góc của **View** để điều chỉnh kích thước, nhưng việc này sẽ gán kích thước cố định (**hardcoded**) cho chiều rộng và chiều cao.

Tránh đặt kích thước cố định cho hầu hết các phần tử **View**, vì kích thước cố định không thể thích ứng với nội dung và các kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn **Attributes** ở phía bên phải của trình chỉnh sửa bố cục để chọn một chế độ kích thước không sử dụng giá trị cố định. Ngăn **Attributes** bao gồm một bảng điều khiển kích thước hình vuông gọi là **view inspector** ở phía trên. Các ký hiệu bên trong hình vuông thể hiện cài đặt chiều cao và chiều rộng như sau:



*Trong hình minh họa:*

1. **Bộ điều khiển chiều cao:** Kiểm soát thuộc tính `layout_height`, xuất hiện dưới dạng hai đoạn ở phía trên và dưới của hình vuông. Các góc nghiêng chỉ ra rằng thuộc

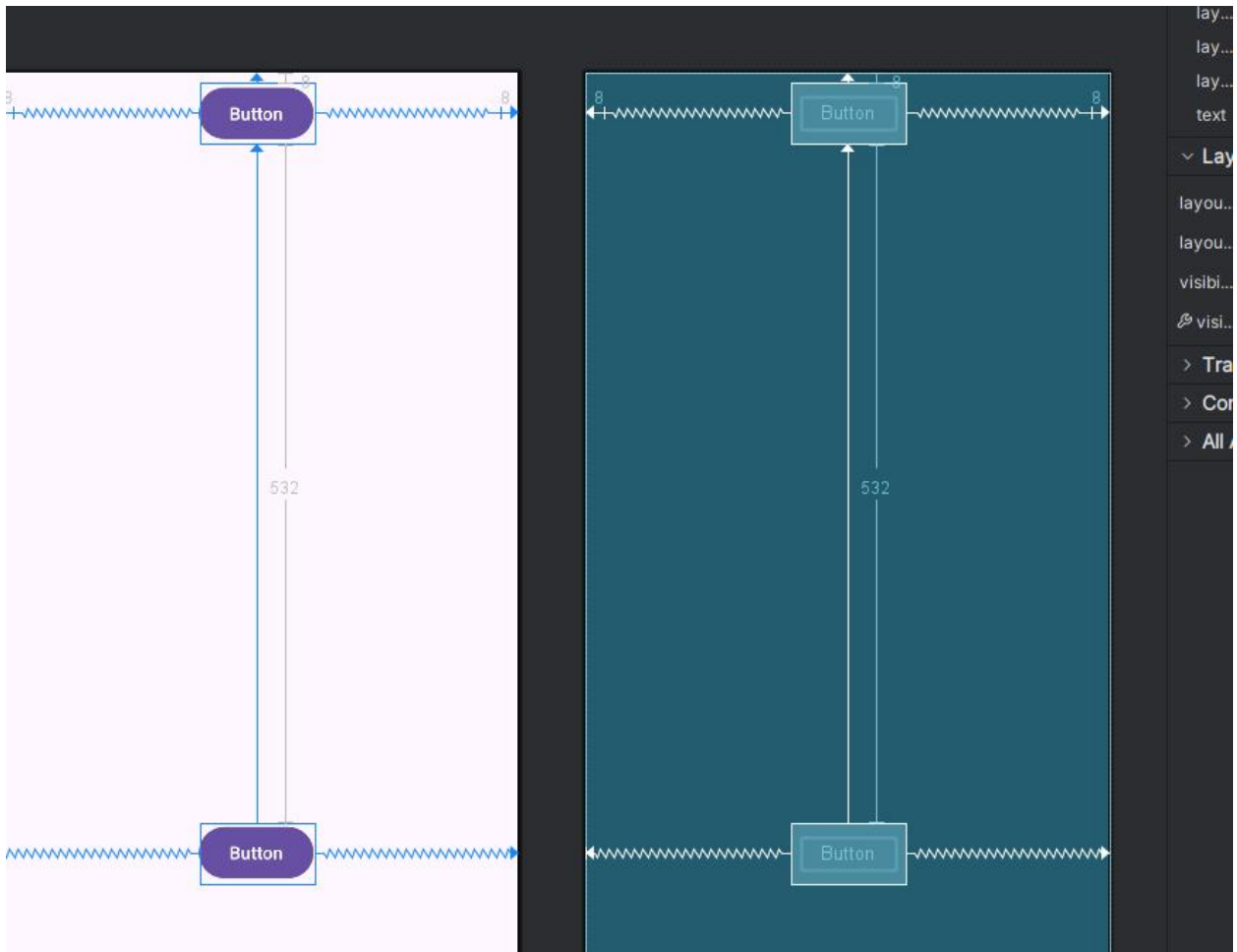
tính này đang được đặt thành `wrap_content`, có nghĩa là **View** sẽ mở rộng theo chiều dọc khi cần để phù hợp với nội dung. Số **8** chỉ ra rằng một lề tiêu chuẩn **8dp** đã được đặt.

2. **Bộ điều khiển chiều rộng:** Kiểm soát thuộc tính `layout_width`, xuất hiện dưới dạng hai đoạn ở phía trái và phải của hình vuông. Các góc nghiêng chỉ ra rằng thuộc tính này cũng được đặt thành `wrap_content`, có nghĩa là **View** sẽ mở rộng theo chiều ngang khi cần để phù hợp với nội dung, với lề **8dp**.
3. **Nút đóng ngăn Attributes:** Nhấp vào đây để đóng ngăn.

---

#### Thực hiện các bước sau:

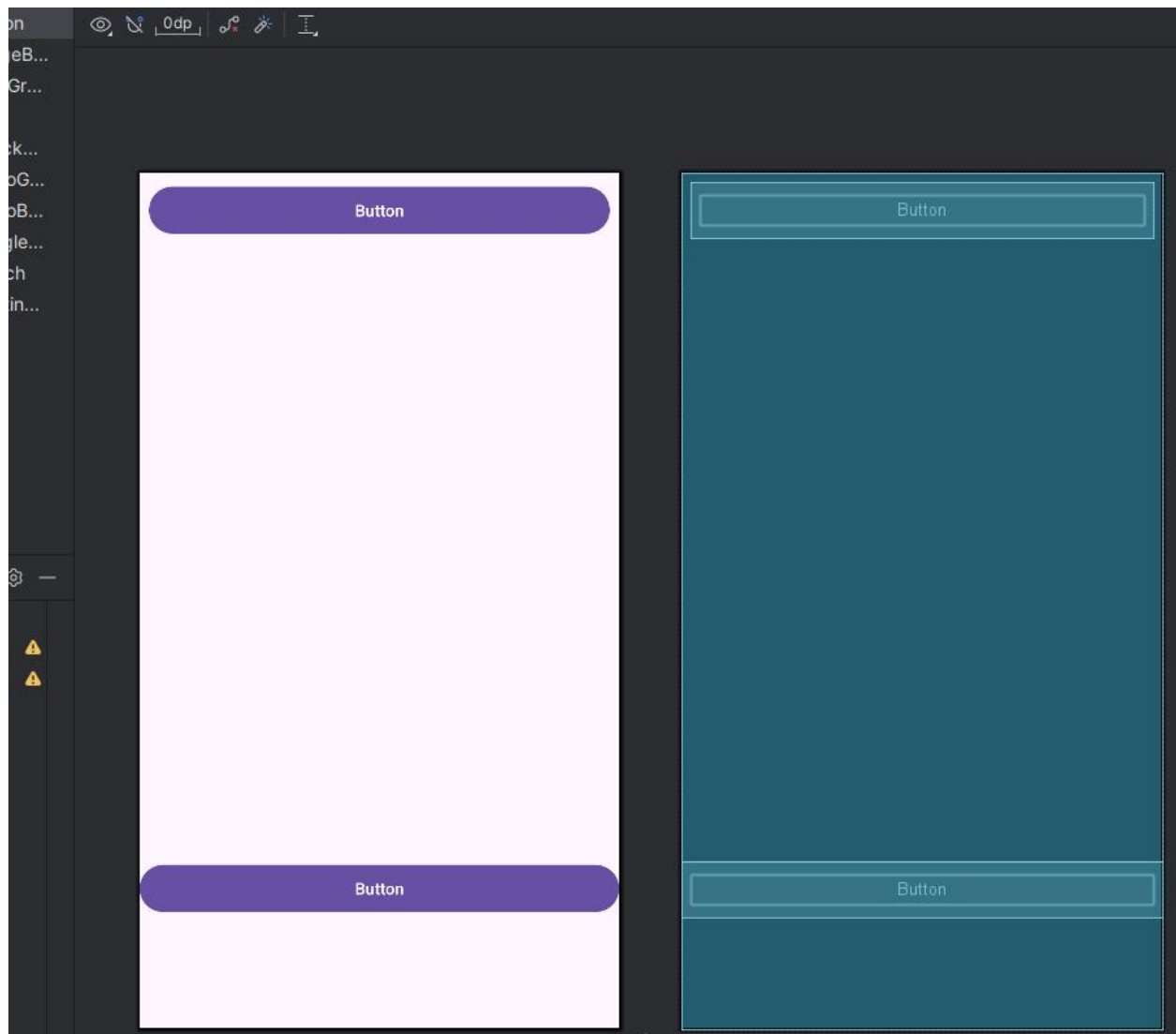
1. Chọn **Button** phía trên trong ngăn **Component Tree**.
2. Nhấp vào tab **Attributes** ở phía bên phải của cửa sổ trình chỉnh sửa bố cục.
3. Nhấp vào **bộ điều khiển chiều rộng** hai lần: Lần nhấp đầu tiên sẽ thay đổi chế độ thành **Fixed** với các đường thẳng. Lần nhấp thứ hai sẽ thay đổi chế độ thành **Match Constraints** với các lò xo co giãn, như trong hình động minh họa.



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

#### Thực hiện các bước tiếp theo:

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



Như đã thấy trong các bước trước, các thuộc tính `layout_width` và `layout_height` trong ngăn **Attributes** sẽ thay đổi khi bạn chỉnh sửa **bộ điều khiển chiều cao và chiều rộng** trong **view inspector**. Các thuộc tính này có thể nhận một trong ba giá trị trong **ConstraintLayout**:

- **match\_constraint**: Mở rộng phần tử **View** để lấp đầy **parent** theo chiều rộng hoặc chiều cao—tính đến cả **margin** nếu có. Trong trường hợp này, **parent** chính là **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.
- **wrap\_content**: Thu nhỏ kích thước của phần tử **View** sao cho nó vừa đủ để chứa nội dung bên trong. Nếu không có nội dung, phần tử **View** sẽ trở nên **vô hình**.



- **Giá trị cố định:** Để chỉ định một kích thước cố định nhưng vẫn thích ứng với kích thước màn hình của thiết bị, sử dụng **pixel độc lập với mật độ (dp)**. Ví dụ: 16dp có nghĩa là **16 pixel độc lập với mật độ**.

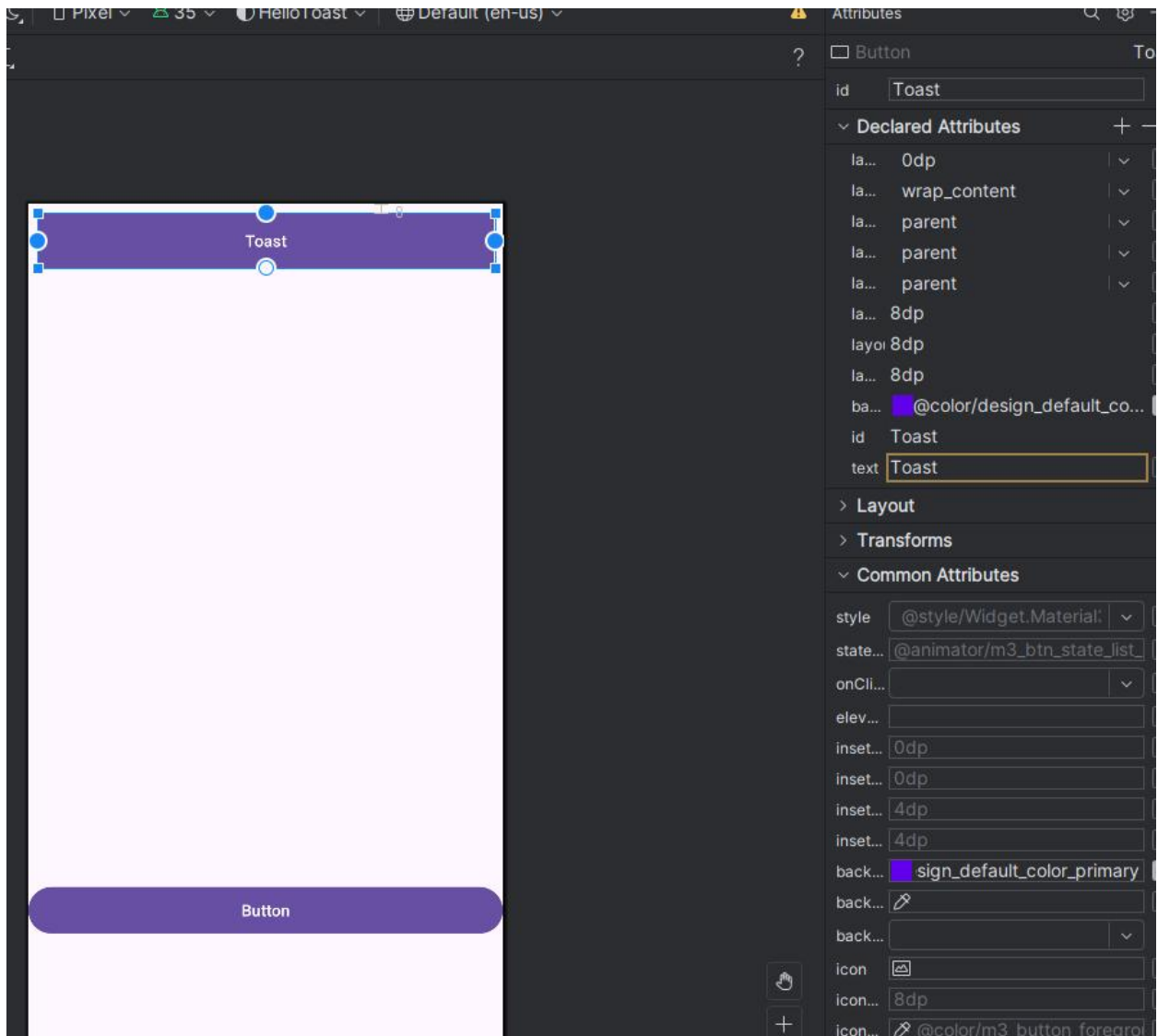
### 3.2 Thay đổi thuộc tính của Button

Để xác định từng View một cách duy nhất trong bố cục của Activity, mỗi View hoặc lớp con của View (chẳng hạn như Button) cần có 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 là màu hoặc hình ảnh.

Ngăn Thuộc tính (Attributes pane) 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 android:id, background, textColor, và text.

Hình động sau đây minh họa cách thực hiện các bước sau:

1. Sau khi chọn Button đầu tiên, 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/colorPrimary. (Khi nhập @c, các lựa chọn sẽ xuất hiện để dễ dàng chọn.)
3. Đặt thuộc tính **textColor** thành @android:color/white.
4. Chỉnh sửa thuộc tính **text** thành "Toast".

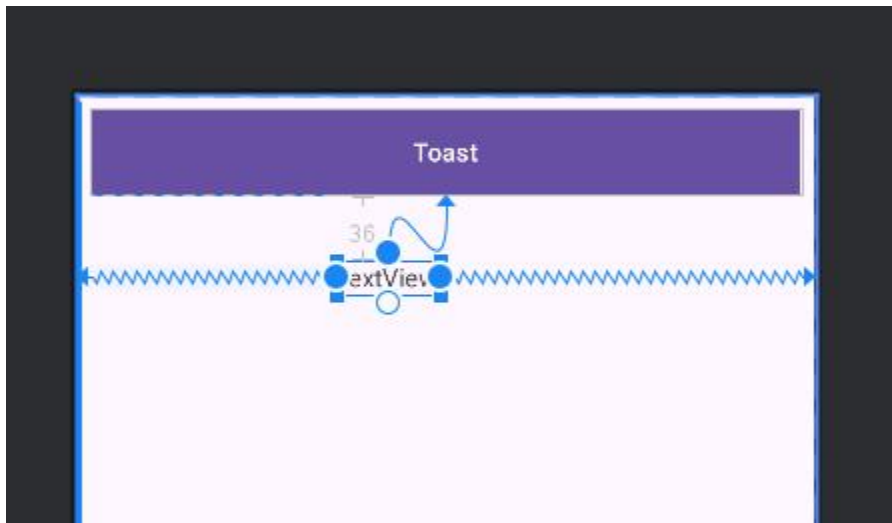


5. Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng `button_count` làm ID, "Count" cho thuộc tính **text**, và giữ nguyên màu nền cũng như màu chữ như các bước trước.

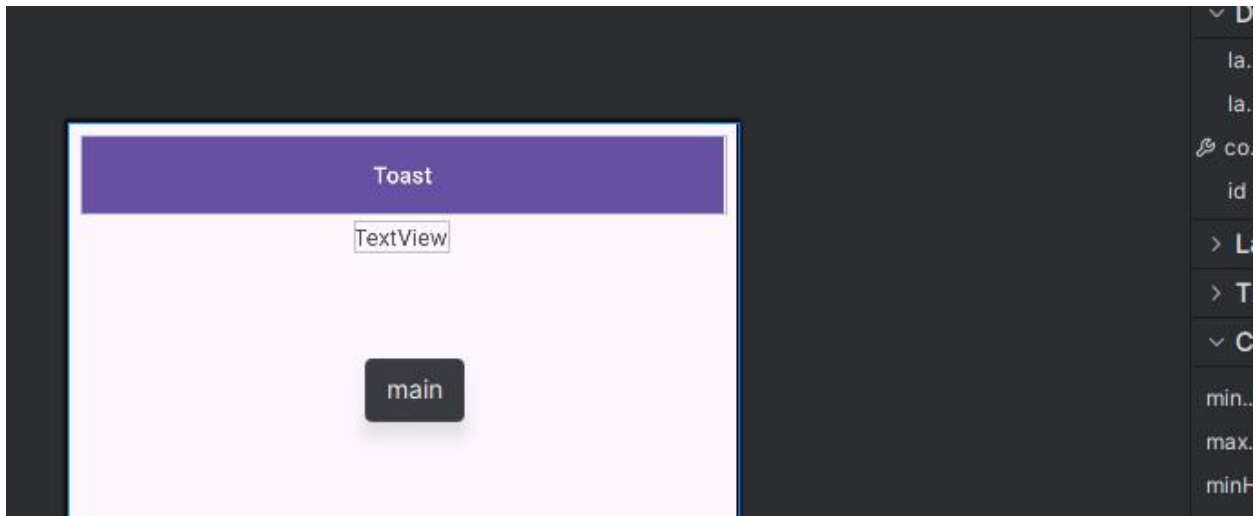
Màu `colorPrimary` là màu chính của chủ đề, một trong những màu cơ bản được xác định trước trong tệp tài nguyên `colors.xml`. Nó được sử dụng cho thanh ứng dụng (**app bar**). Việc sử dụng các màu cơ bản này cho các phần tử giao diện người dùng khác giúp tạo ra một giao diện đồng nhất. Bạn sẽ tìm hiểu thêm về chủ đề ứng dụng (**app themes**) và **Material Design** trong bài học khác.

#### 4.1 Thêm TextView và ràng buộc (Constraints)

1. Như minh họa trong hình động bên dưới, kéo một **TextView** từ ngăn **Palette** vào phần trên của bố cục, sau đó kéo một ràng buộc (**constraint**) từ đỉnh của **TextView** đến tay cầm (**handle**) ở đáy của Button **Toast**. Điều này sẽ ràng buộc **TextView** nằm bên dưới Button.



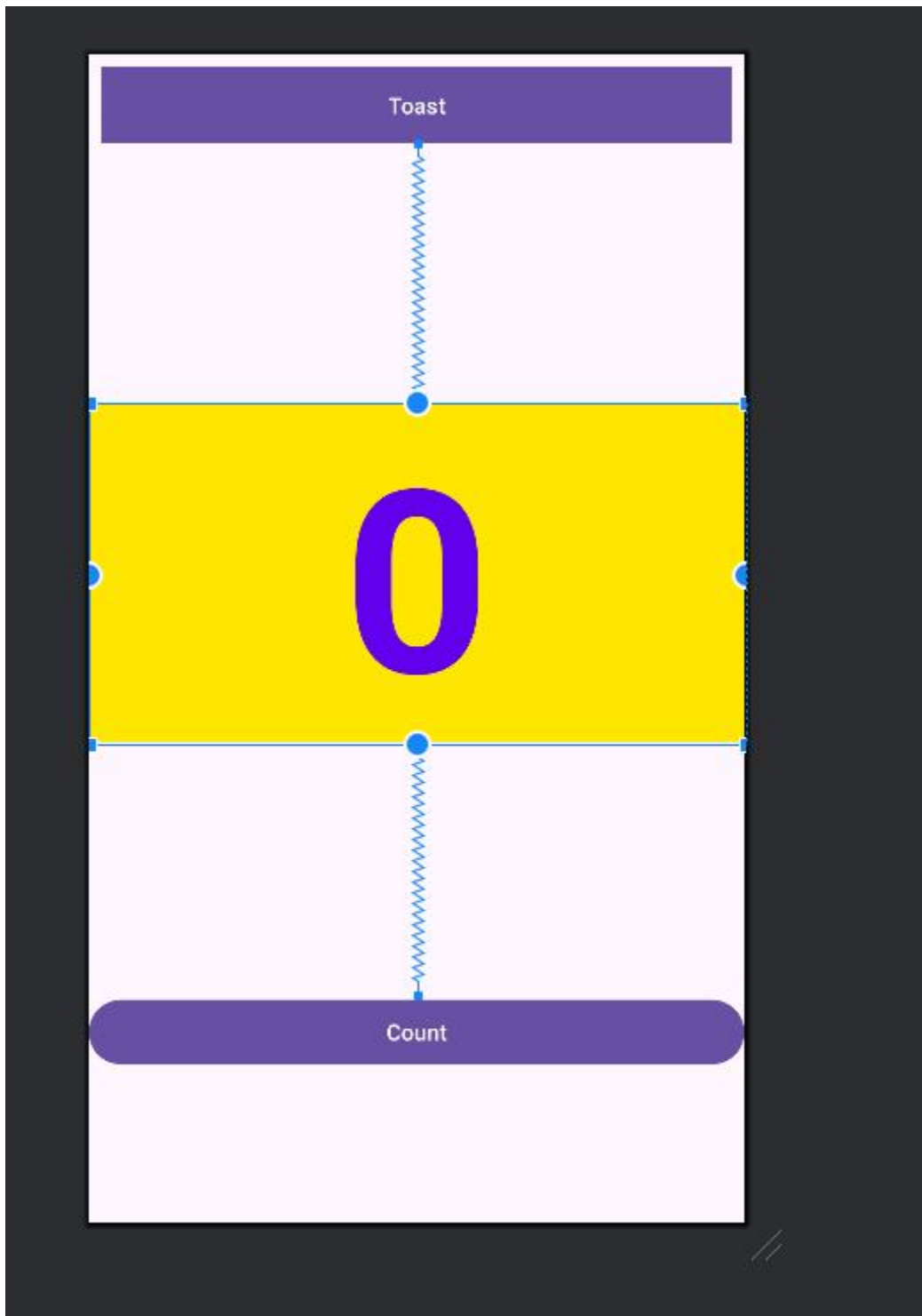
2. Tiếp theo, kéo một ràng buộc từ đáy của **TextView** đến tay cầm ở đỉnh của Button **Count**, và từ hai bên của **TextView** đến hai bên của bố cục. Điều này sẽ giữ **TextView** ở giữa bố cục giữa hai Button.



## 4.2 Thiết lập thuộc tính cho TextView

Với **TextView** được chọn, hãy mở ngăn **Attributes** (nếu chưa mở) và đặt các thuộc tính như sau:

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 `ALIGNCENTER` (căn giữa đoạn văn bản).
5. Thay đổi kích thước ngang (**layout\_width**) và dọc (**layout\_height**) thành `match_constraint`.
6. Đặt **textColor** thành `@color/colorPrimary`.
7. Cuộn xuống trong ngăn thuộc tính, nhấp vào **View all attributes**, tiếp tục cuộn xuống đến **background**, sau đó nhập `#FFF00` để chọn màu vàng.
8. Cuộn xuống **gravity**, mở rộng thuộc tính này và chọn `center_ver` (căn giữa theo chiều dọc).



- **textSize**: Kích thước chữ của **TextView**. Trong bài học này, kích thước được đặt là **160sp**. Đơn vị **sp** (scale-independent pixel) giống như **dp**, nhưng được dùng cho văn bản để điều chỉnh theo mật độ màn hình và cài đặt kích thước chữ của người dùng.

- **textStyle** và **textAlignment**:
  - **textStyle** được đặt thành **B (bold)** để làm đậm chữ.
  - **textAlignment** được đặt thành **ALIGNCENTER** để căn giữa văn bản.
- **gravity**: Xác định cách một **View** căn chỉnh trong **View** cha hoặc **ViewGroup**. Ở bước này, **TextView** sẽ được căn giữa theo chiều dọc trong **ConstraintLayout**.

Lưu ý: Thuộc tính **background** có thể xuất hiện trên trang đầu tiên của ngăn **Attributes** đối với **Button**, nhưng sẽ nằm trên trang thứ hai đối với **TextView**. Ngăn **Attributes** thay đổi tùy theo loại **View**, hiển thị các thuộc tính phổ biến nhất trên trang đầu tiên và các thuộc tính còn lại trên trang thứ hai. Bạn có thể quay lại trang đầu tiên bằng cách nhấp vào biểu tượng trên thanh công cụ ở đầu ngăn **Attributes**.

## Task 5: Chỉnh sửa bố cục trong XML

Ứng dụng **Hello Toast** gần như đã hoàn thành! Tuy nhiên, bạn có thể thấy dấu chấm than xuất hiện bên cạnh mỗi phần tử UI trong **Component Tree**.

- Khi di chuột qua dấu chấm than, bạn sẽ thấy thông báo cảnh báo: **Các chuỗi cứng (hardcoded strings) nên sử dụng tài nguyên chuỗi (string resources)**.
- Đây là một cảnh báo phổ biến trong Android Studio, khuyến khích bạn lưu trữ các chuỗi văn bản trong **tập tài nguyên (resources)** thay vì mã cứng trong bố cục.

### 5.1 Mở tập XML của bố cục

Để thực hiện tác vụ này:

1. Mở tập **activity\_main.xml** (nếu chưa mở).
2. Nhấp vào tab **Text** để chuyển sang trình chỉnh sửa XML.

Trình chỉnh sửa XML sẽ xuất hiện, thay thế các bảng thiết kế (**design**) và bản vẽ (**blueprint**).

Bạn sẽ thấy các cảnh báo được đánh dấu, cụ thể là các chuỗi cứng (hardcoded strings) như "Toast" và "Count" trong mã XML. (Chuỗi "0" cũng bị cảnh báo nhưng có thể không hiển thị trong hình minh họa.)

Di chuột qua chuỗi "Toast" để xem thông báo cảnh báo.

## 5.2 Trích xuất tài nguyên chuỗi

Thay vì mã cứng các chuỗi, một phương pháp hay nhất là sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Việc đặt các chuỗi trong một tệp riêng giúp dễ dàng quản lý chúng, đặc biệt nếu bạn sử dụng những chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và bản địa hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi riêng cho mỗi ngôn ngữ.

1. Nhấp một lần vào từ "Toast" (cảnh báo được đánh dấu đầu tiên).
2. Nhấn **Alt + Enter** trong Windows hoặc **Option + Enter** trong macOS và chọn **Extract string resource** từ menu bật lên.
3. Nhập `button_label_toast` làm **Resource name**.
4. Nhấp vào **OK**. Một tài nguyên chuỗi sẽ được tạo trong tệp `res/values/strings.xml`, và chuỗi trong mã của bạn sẽ được thay thế bằng một tham chiếu đến tài nguyên:

```
@string/button_label_toast
```

5. Trích xuất các chuỗi còn lại:
  1. Sử dụng `button_label_count` cho "Count".
  2. Sử dụng `count_initial_value` cho "0".
6. Trong **Project > Android**, mở rộng **values** trong **res**, sau đó nhấp đúp vào `strings.xml` để xem tài nguyên chuỗi của bạn trong tệp `strings.xml`.

```
<resources>

<string name="app_name">Hello Toast</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 một chuỗi khác để sử dụng trong một nhiệm vụ tiếp theo nhằm hiển thị một thông báo.*

Thêm vào tệp **strings.xml** một tài nguyên chuỗi mới có tên **toast\_message** với nội dung "Hello Toast!".

```
<resources>

<string name="app_name">Hello Toast</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>
```

## Nhiệm vụ 6: Thêm trình xử lý onClick cho các Button

Trong nhiệm vụ này, bạn sẽ thêm một phương thức Java cho mỗi Button trong **MainActivity**, phương thức này sẽ được thực thi khi người dùng nhấn vào Button.

### 6.1 Thêm thuộc tính onClick và trình xử lý cho từng Button

Trình xử lý sự kiện nhấp chuột (**click handler**) là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI có thể nhấp.

Trong **Android Studio**, bạn có thể chỉ định tên của phương thức trong trường **onClick** trong **tab Design** của bảng **Attributes**. Bạn cũng có thể chỉ định tên của phương thức trong **trình chỉnh sửa XML** bằng cách thêm thuộc tính **android:onClick** vào Button. Bạn sẽ sử dụng phương pháp thứ hai vì bạn chưa tạo các phương thức xử lý sự kiện, và trình chỉnh sửa XML cung cấp cách tự động để tạo các phương thức này.

**1. Mở trình chỉnh sửa XML (tab Text). Tìm Button có android:id được đặt thành button\_toast.**

```
<Button

android:id="@+id/button_toast"

android:layout_width="0dp"

...

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent" />
```



2. **Thêm thuộc tính** `android:onClick` vào phần tử `button_toast`, sau thuộc tính cuối cùng và trước ký hiệu đóng `</>`.

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

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

Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy nhấp vào tên phương thức ("`showToast`"), nhấn **Alt + Enter** trên Windows hoặc **Option + Enter** trên Mac, chọn **Create 'showToast(view)' in MainActivity**, rồi nhấn **OK**.

**Hành động này sẽ tạo một phương thức mẫu (stub method) cho `showToast()` trong MainActivity.**

4. **Lặp lại các bước trên cho Button có `android:id` là `button_count`:** Thêm thuộc tính `android:onClick`. Thêm trình xử lý sự kiện nhấp chuột.

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

Sau khi thực hiện xong, mã XML của các phần tử UI trong **ConstraintLayout** sẽ trông như sau:

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="@color/colorPrimary"
    android:text="@string/button_label_toast"
    android:textColor="@android:color/white"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="showToast"/>

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

```
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_marginBottom="8dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:background="@color/colorPrimary"
android:text="@string/button_label_count"
android:textColor="@android:color/white"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
android:onClick="countUp" />
<TextView
android:id="@+id/show_count"
android:layout_width="0dp"
android:layout_height="0dp"
android:layout_marginBottom="8dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:layout_marginTop="8dp"
android:background="#FFFF00"
android:gravity="center_vertical"
android:text="@string/count_initial_value"
android:textAlignment="center"
android:textColor="@color/colorPrimary"
android:textSize="160sp"
android:textStyle="bold"
app:layout_constraintBottom_toTopOf="@+id/button_count"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/button_toast" />
```

5. Nếu **MainActivity.java** chưa mở, hãy mở **Project > Android**, mở rộng thư mục **java**, mở rộng **com.example.android.hellotoast**, rồi nhấp đúp vào **MainActivity.java** để xem mã nguồn.

```
package com.example.android.hellotoast;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

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

    public void showToast(View view) {
    }

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

## 6.2 Chỉnh sửa trình xử lý nút Toast

Bây giờ bạn sẽ chỉnh sửa phương thức `showToast()`—trình xử lý khi nhấn nút Toast trong `MainActivity`—để hiển thị một thông báo.

Một **Toast** cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ chiếm không gian cần thiết cho thông báo. Hoạt động hiện tại (**Activity**) vẫn hiển thị và có thể tương tác được.

**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 khi nhấn một **Button** hoặc thực hiện một hành động.

Hãy làm theo các bước sau để chỉnh sửa trình xử lý khi nhấn nút **Toast**:

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

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

2. Để tạo một thể hiện của **Toast**, hãy gọi phương thức makeText() của 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** ứng dụng. Vì **Toast** hiển thị trên giao diện **Activity**, hệ thống cần thông tin về **Activity** hiện tại. Khi đang ở trong **Activity** cần sử dụng **context**, bạn có thể dùng this như một cách viết tắt.

```
Toast toast = Toast.makeText(this,
```

4. Cung cấp thông báo để hiển thị, chẳng hạn như một **chuỗi tài nguyên** (toast\_message mà bạn đã tạo ở bước trước). **Chuỗi tài nguyên** toast\_message được xác định bởi R.string.toast\_message.

```
Toast toast = Toast.makeText(this, R.string.toast_message,
```

5. Cung cấp thời gian hiển thị. Ví dụ: Toast.LENGTH\_SHORT hiển thị **Toast** trong một khoảng thời gian ngắn.

```
Toast toast = Toast.makeText(this, R.string.toast_message,  
    Toast.LENGTH_SHORT);
```

Thời gian hiển thị của **Toast** có thể là `Toast.LENGTH_LONG` hoặc `Toast.LENGTH_SHORT`.

Thời gian thực tế: khoảng **3.5 giây** đối với `Toast.LENGTH_LONG` và **2 giây** đối với `Toast.LENGTH_SHORT`.

6. Hiển thị **Toast** bằng cách gọi `show()`.

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

Chạy ứng dụng và kiểm tra xem thông báo **Toast** có xuất hiện khi nhấn nút **Toast** hay không.

10:09



Toast

0



Hello Toast!

Count

### 6.3 Chỉnh sửa trình xử lý nút Count

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

Mã xử lý sự kiện phải:

- Theo dõi số đếm khi nó thay đổi.
- Gửi số đếm đã cập nhật đến TextView để hiển thị.

Làm theo các bước sau để chỉnh sửa trình xử lý nút Count:

1. Xác định vị trí phương thức `countUp()` vừa tạo.

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

2. Để theo dõi số đếm, bạn cần một **biến thành viên riêng**. Mỗi lần nhấn nút Count sẽ làm tăng giá trị của biến này. Nhập đoạn mã sau, mã này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn đỏ

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

Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy chọn biểu thức `mCount++`. Biểu tượng bóng đèn sẽ xuất hiện sau một lúc.

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'mCount'** từ menu bật lên. Thao tác này sẽ tạo một **biến thành viên riêng** ở đầu MainActivity, và Android Studio sẽ giả định rằng bạn muốn nó có kiểu số nguyên (`int`).

```
public class MainActivity extends AppCompatActivity {
```

```
private int mCount;
```

4. Thay đổi dòng khai báo biến thành viên để khởi tạo biến về 0.

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

5. Cùng với biến trên, bạn cũng cần một biến thành viên riêng để tham chiếu đến TextView show\_count, mà bạn sẽ thêm vào trình xử lý khi nhấn nút. Gọi biến này là mShowCount.

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

6. Khi đã có mShowCount, bạn có thể lấy tham chiếu đến TextView bằng cách sử dụng **ID** mà bạn đã đặt trong tệp bố cục (layout). Để chỉ lấy tham chiếu **một lần**, hãy khai báo nó trong phương thức onCreate(). Như bạn sẽ học trong một bài học khác, phương thức onCreate() được sử dụng để inflate layout, có nghĩa là thiết lập giao diện màn hình bằng tệp XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện người dùng khác trong bố cục, chẳng hạn như TextView. Xác định vị trí phương thức onCreate() trong MainActivity.

```
@Override
```

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



7. Thêm câu lệnh findViewById vào cuối phương thức:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    mShowCount = (TextView) findViewById(R.id.show_count);  
}
```

Một View, giống như một chuỗi, là một tài nguyên có thể có một ID. Lệnh findViewById nhận ID của một View làm tham số và trả về View. Vì phương thức này trả về một View, bạn phải ép kiểu kết quả về loại View mà bạn mong đợi, trong trường hợp này là TextView.

8. Sau khi gán mShowCount với TextView, bạn có thể sử dụng biến này để đặt văn bản trong TextView thành giá trị của biến mCount. Thêm đoạn mã sau vào phương thức countUp().

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

9. Chạy ứng dụng để kiểm tra xem số đếm có tăng lên khi nhấn nút Count hay không.

Toast

1

Count



1:1



## Thử thách lập trình

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

Ứng dụng **HelloToast** trông ổn khi thiết bị hoặc trình giả lập được đặt theo hướng dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang **hướng ngang**, nút **Count** có thể chồng lên TextView ở phía dưới, như được hiển thị trong hình bên dưới.

**Thử thách:** Thay đổi bố cục để nó trông đẹp trong cả hai hướng **ngang và dọc**:

1. Trên máy tính của bạn, **tạo một bản sao** của thư mục dự án **HelloToast** và đổi tên nó thành **HelloToastChallenge**.
2. Mở **HelloToastChallenge** trong **Android Studio** và **tái cấu trúc nó**. (Xem **Phụ lục: Tiện ích** để biết hướng dẫn về cách sao chép và tái cấu trúc một dự án.)
3. Thay đổi bố cục sao cho nút **Toast** và nút **Count** xuất hiện **ở bên trái**, như trong hình dưới đây. TextView xuất hiện bên cạnh chúng, nhưng chỉ rộng **vừa đủ để hiển thị nội dung**. (Gợi ý: Sử dụng `wrap_content`.)
4. Chạy ứng dụng trong cả **hướng ngang và hướng dọc** để kiểm tra.

## Tóm tắt

### *View, ViewGroup và bố cục (layouts):*

- Tất cả các phần tử giao diện người dùng (UI elements) đều là **lớp con của lớp View** và do đó kế thừa nhiều thuộc tính từ lớp cha **View**.
- Các phần tử **View** có thể được nhóm bên trong một **ViewGroup**, đóng vai trò là một **container**. Mối quan hệ này là **cha - con**, trong đó **cha** là **ViewGroup**, còn **con** có thể là **View** hoặc **một ViewGroup khác**.
- Phương thức `onCreate()` được sử dụng để **inflate layout**, nghĩa là thiết lập giao diện của màn hình với tệp XML bố cục. Bạn cũng có thể sử dụng phương thức này để lấy tham chiếu đến các phần tử giao diện khác trong bố cục.
- Một View, giống như một chuỗi (string), là một tài nguyên có thể có **id**. Lệnh `findViewById` nhận **ID của một View** làm tham số và trả về chính View đó.

---

### *Sử dụng trình chỉnh sửa bố cục (Layout Editor):*

- Nhấn vào **tab Design** để thao tác với các phần tử và bố cục, hoặc nhấn vào **tab Text** để chỉnh sửa mã XML của bố cục.
- Trong **tab Design**, **bảng Palette** hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục của ứng dụng, còn **bảng Component Tree** hiển thị **cây phân cấp của các phần tử UI**.
- **Các bảng thiết kế (Design) và bản vẽ bố cục (Blueprint)** hiển thị các phần tử UI trong bố cục.
- **Tab Attributes** hiển thị **bảng thuộc tính**, cho phép bạn đặt các thuộc tính cho một phần tử UI.

### Các công cụ chỉnh sửa:

- **Tay cầm ràng buộc (Constraint handle):** Nhấn vào một tay cầm ràng buộc (hình tròn ở mỗi cạnh của phần tử), sau đó kéo đến **tay cầm khác** hoặc **ranh giới của phần tử cha** để tạo ràng buộc. Ràng buộc sẽ được biểu diễn bằng **đường zigzag**.
- **Tay cầm thay đổi kích thước (Resizing handle):** Kéo các **tay cầm vuông** để thay đổi kích thước phần tử. Khi kéo, tay cầm sẽ chuyển sang **góc xiên**.
- **Công cụ Autoconnect (Tự động kết nối):** Khi được bật, công cụ này sẽ tự động **tạo hai hoặc nhiều ràng buộc** giữa phần tử UI và bố cục cha dựa trên vị trí của phần tử đó.
- **Xóa ràng buộc:** Chọn phần tử, di chuột qua nó để hiển thị **nút Clear Constraints** và nhấn để xóa tất cả ràng buộc. Để xóa một ràng buộc cụ thể, nhấn vào tay cầm ràng buộc tương ứng.
- **Bảng thuộc tính (Attributes pane):** Cho phép truy cập tất cả **thuộc tính XML** mà bạn có thể gán cho một phần tử UI. Nó cũng có **bảng kiểm tra giao diện (View Inspector)** ở trên cùng để chỉnh sửa kích thước.
- **Thuộc tính layout\_width và layout\_height:** Có thể nhận một trong ba giá trị trong **ConstraintLayout**:
  - **match\_constraint:** Mở rộng phần tử để **lấp đầy phần tử cha** theo chiều rộng hoặc chiều cao (đến lề nếu có).
  - **wrap\_content:** Thu nhỏ phần tử sao cho nó vừa đủ chứa nội dung bên trong. Nếu không có nội dung, phần tử sẽ **biến mất**.
  - **Số dp cố định:** Đặt kích thước cố định **theo đơn vị dp** (mật độ điểm ảnh độc lập), giúp điều chỉnh kích thước theo màn hình thiết bị.

---

### Trích xuất tài nguyên chuỗi (String resources):

Thay vì **mã hóa cứng (hard-code)** chuỗi trong mã, cách tốt nhất là sử dụng **tài nguyên chuỗi (string resources)**, giúp quản lý và dịch thuật dễ dàng hơn. Làm theo các bước sau:

1. Nhấn vào chuỗi cứng (hardcoded string) cần trích xuất, nhấn **Alt + Enter (Mac: Option + Enter)**, sau đó chọn **Extract string resources** từ menu xuất hiện.
2. Đặt **tên tài nguyên (Resource name)**.
3. Nhấn **OK**. Hệ thống sẽ tạo một tài nguyên chuỗi trong tệp `res/values/strings.xml`, đồng thời thay thế chuỗi trong mã bằng một **tham chiếu** đến tài nguyên đó: `@string/button_label_toast`.

---

### Xử lý sự kiện nhấn (Handling clicks):

- **Trình xử lý nhấn (Click handler)** là một phương thức được gọi khi người dùng nhấn hoặc chạm vào một phần tử giao diện.
- Để đặt **trình xử lý nhấn** cho một phần tử (ví dụ: Button):
  - Nhập tên phương thức vào **trường onClick** trong bảng thuộc tính (tab Design).
  - Hoặc thêm thuộc tính `android:onClick` vào mã XML của Button.

Trong MainActivity, tạo trình xử lý nhấn với tham số View. Ví dụ:

Bạn có thể tìm thấy tất cả **thuộc tính của Button** trong tài liệu lớp Button, và tất cả **thuộc tính của TextView** trong tài liệu lớp TextView.

---

### Hiển thị thông báo bằng Toast:

Toast là một cách để hiển thị **thông báo ngắn** trong một **cửa sổ bật lên nhỏ**. Nó chỉ chiếm **không gian đủ để hiển thị thông báo**, và màn hình ứng dụng vẫn hiển thị cũng như tương tác được.

### Các bước tạo một Toast:

1. Gọi phương thức `makeText()` của lớp `Toast`.
2. Cung cấp **context** của `Activity` hiện tại và thông điệp cần hiển thị (ví dụ: một chuỗi tài nguyên).
3. Cung cấp **thời gian hiển thị**:
  - `Toast.LENGTH_SHORT`: Hiển thị trong khoảng **2 giây**.
  - `Toast.LENGTH_LONG`: Hiển thị trong khoảng **3.5 giây**.
4. Gọi phương thức `show()` để hiển thị `Toast`.

## 1.3) Trình chỉnh sửa bố cục

### Giới thiệu

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

**ConstraintLayout** là một **ViewGroup**, một **View đặc biệt** có thể chứa các **đối tượng View khác** (gọi là **child views**). Trong bài thực hành này, bạn sẽ tìm hiểu thêm về các tính năng của **ConstraintLayout** và **trình chỉnh sửa bố cục**.

Bài thực hành này cũng giới thiệu hai lớp con khác của **ViewGroup**:

- **LinearLayout**: Một nhóm sắp xếp các phần tử View con **theo chiều ngang hoặc chiều dọc**.
- **RelativeLayout**: Một nhóm trong đó mỗi phần tử View con **được căn chỉnh dựa trên các phần tử View khác** bên trong cùng một ViewGroup. Các vị trí của phần tử con được xác định **so với các phần tử khác hoặc so với ViewGroup cha**.

---

### Những kiến thức cần có trước

Bạn nên biết cách:

- ✓ **Tạo ứng dụng "Hello World"** với Android Studio.
- ✓ **Chạy ứng dụng** trên trình giả lập hoặc thiết bị thực.
- ✓ **Tạo bố cục đơn giản** cho ứng dụng bằng **ConstraintLayout**.
- ✓ **Trích xuất và sử dụng tài nguyên chuỗi (String resources)**.

---

### Những gì bạn sẽ học

- Cách tạo **bố cục khác nhau** cho màn hình ngang (**landscape orientation**).
- Cách tạo **bố cục dành cho máy tính bảng và màn hình lớn**.
- Cách sử dụng **baseline constraint** để **căn chỉnh văn bản với các phần tử UI**.
- Cách sử dụng **các nút Pack và Align** để **căn chỉnh các phần tử trong bố cục**.



Cách **định vị phần tử** trong **LinearLayout**.  
Cách **định vị phần tử** trong **RelativeLayout**.

---

## Những gì bạn sẽ làm

Tạo bố cục cho màn hình ngang (landscape orientation).  
Tạo bố cục cho máy tính bảng và màn hình lớn.  
Chỉnh sửa bố cục để thêm ràng buộc cho các phần tử UI.  
Sử dụng baseline constraint để căn chỉnh các phần tử với văn bản.  
Sử dụng nút Pack và Align để căn chỉnh phần tử trong ConstraintLayout.  
Chuyển bố cục sang LinearLayout.  
Định vị các phần tử trong LinearLayout.  
Chuyển bố cục sang RelativeLayout.  
Sắp xếp lại các phần tử trong bố cục chính để chúng có vị trí tương đối với nhau.

---

## Tổng quan về ứng dụng

Ứng dụng **Hello Toast** trong bài học trước sử dụng **ConstraintLayout** để sắp xếp các phần tử UI trong bố cục của Activity, như minh họa trong hình dưới đây.

Để thực hành thêm với **ConstraintLayout**, bạn sẽ tạo một **biến thể của bố cục này dành cho chế độ ngang (horizontal orientation)**.

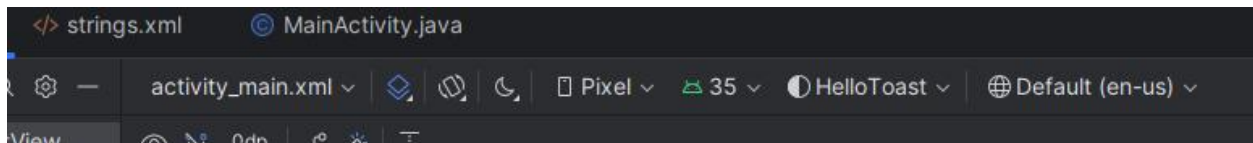
Bạn cũng sẽ học cách sử dụng **ràng buộc đường cơ sở (baseline constraints)** và một số **tính năng căn chỉnh của ConstraintLayout** bằng cách tạo một biến thể khác của bố cục dành cho **màn hình máy tính bảng**.

Ngoài ra, bạn sẽ tìm hiểu về các **lớp con khác của ViewGroup**, chẳng hạn như **LinearLayout** và **RelativeLayout**, đồng thời thay đổi bố cục của ứng dụng **Hello Toast** để sử dụng chúng.

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

Trong bài học trước, thử thách lập trình yêu cầu thay đổi bố cục của ứng dụng **Hello Toast** để nó hiển thị đúng trong cả **chế độ ngang (horizontal)** và **chế độ dọc (vertical)**. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo **các biến thể bố cục cho điện thoại ở cả chế độ ngang (landscape) và chế độ dọc (portrait)**, cũng như cho các 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 (layout editor)**. Thanh công cụ trên cùng cho phép bạn **cấu hình giao diện của bản xem trước bố cục trong trình chỉnh sửa bố cục**:



Trong hình minh họa bên trên:

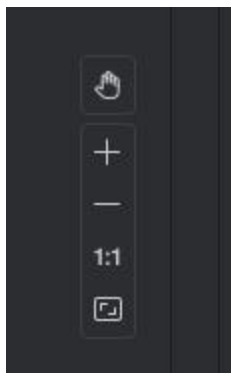
1. **Chọn bề mặt thiết kế (Select Design Surface):** Chọn **Design** để hiển thị bản xem trước có màu của bố cục. Chọn **Blueprint** để chỉ hiển thị đường viền của từng phần tử UI. Để xem cả hai chế độ **cạnh nhau**, chọn **Design + Blueprint**.
2. **Chọn hướng trong trình chỉnh sửa (Orientation in Editor):** Chọn **Portrait** hoặc **Landscape** để xem trước bố cục ở chế độ dọc hoặc ngang. Điều này hữu ích để xem trước bố cục mà không cần chạy ứng dụng trên trình giả lập hoặc thiết bị. Để tạo **bố cục thay thế**, chọn **Create Landscape Variation** hoặc các biến thể khác.
3. **Chọn thiết bị trong trình chỉnh sửa (Device in Editor):** Chọn loại thiết bị: **Điện thoại/Máy tính bảng (Phone/Tablet)**, **Android TV** hoặc **Android Wear**.
4. **Chọn phiên bản API trong trình chỉnh sửa (API Version in Editor):** Chọn phiên bản Android để hiển thị bản xem trước.
5. **Chọn chủ đề trong trình chỉnh sửa (Theme in Editor):** Chọn một chủ đề (chẳng hạn **AppTheme**) để áp dụng cho bản xem trước.
6. **Chọn ngôn ngữ trong trình chỉnh sửa (Locale in Editor):** Chọn ngôn ngữ và vùng miền cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có trong tài nguyên chuỗi (string resources). Bạn cũng có thể chọn **Preview as Right To Left** để xem bố cục như thể một ngôn ngữ RTL (phải sang trái) đã được chọn.

---

## Thanh công cụ thứ hai

Thanh công cụ thứ hai cho phép bạn **cấu hình giao diện của các phần tử UI trong ConstraintLayout**, cũng như **thu phóng và di chuyển bản xem trước**:





Trong hình minh họa bên trên:

1. **Hiển thị (Show)**: Chọn **Show Constraints** và **Show Margins** để hiển thị hoặc ẩn chúng trong bản xem trước.
2. **Tự động kết nối (Autoconnect)**: Bật hoặc tắt **Autoconnect**. Khi **Autoconnect** được bật, bạn có thể kéo bất kỳ phần tử nào (như **Button**) đến bất kỳ vị trí nào trong bố cục để tạo ràng buộc (constraints) tự động với bố cục cha.
3. **Xóa tất cả ràng buộc (Clear All Constraints)**: Xóa toàn bộ ràng buộc trong bố cục.
4. **Suy luận ràng buộc (Infer Constraints)**: Tự động tạo ràng buộc bằng cách suy luận vị trí của các phần tử.
5. **Lề mặc định (Default Margins)**: Đặt khoảng cách lề mặc định giữa các phần tử.
6. **Đóng gói (Pack)**: Thu gọn hoặc mở rộng các phần tử đã chọn.
7. **Căn chỉnh (Align)**: Căn chỉnh các phần tử đã chọn.
8. **Hướng dẫn (Guidelines)**: Thêm đường hướng dẫn dọc hoặc ngang để sắp xếp các phần tử.
9. **Điều khiển thu phóng/dịch chuyển (Zoom/pan controls)**: Phóng to hoặc thu nhỏ bản xem trước.

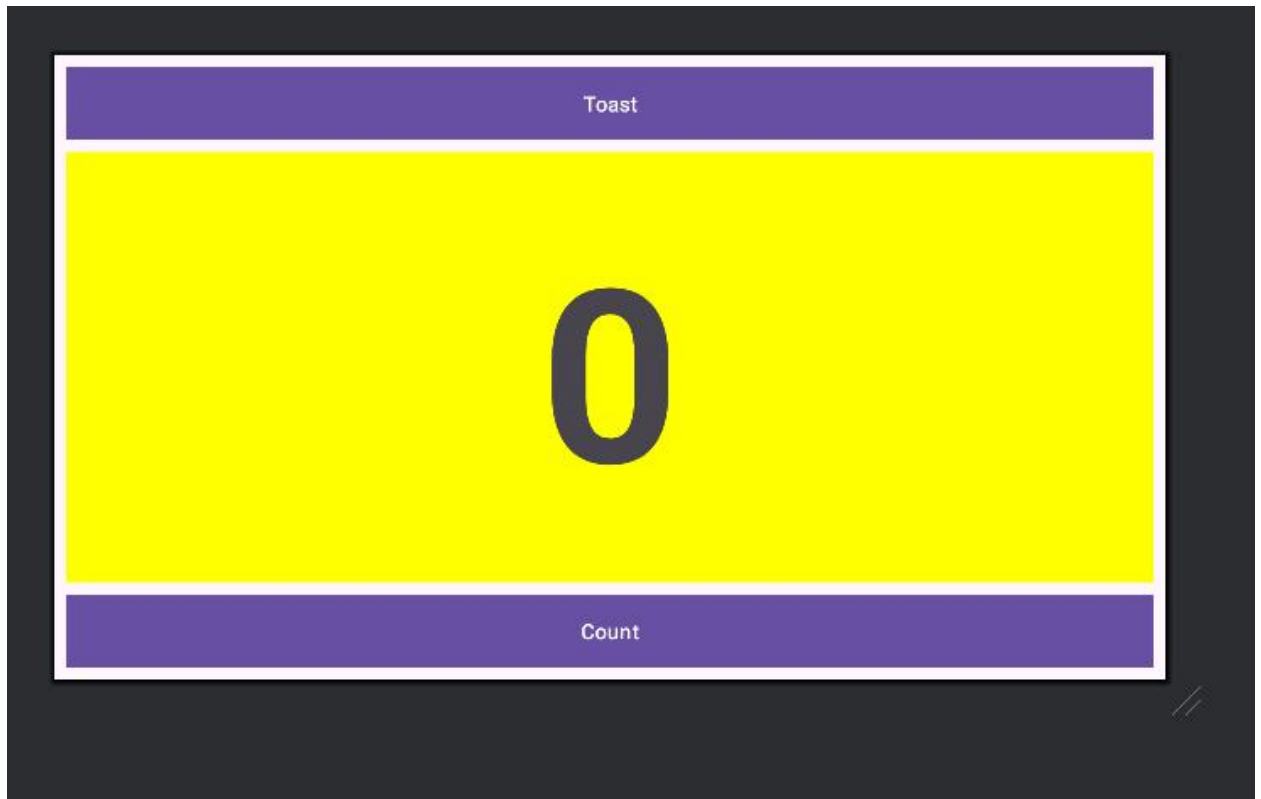
**Mẹo**: Để tìm hiểu thêm về cách sử dụng **trình chỉnh sửa bố cục (layout editor)**, hãy xem **Build a UI with Layout Editor**. Để tìm hiểu thêm về cách xây dựng bố cục với **ConstraintLayout**, hãy xem **Build a Responsive UI with ConstraintLayout**.

### 1.1 Xem trước bố cục ở chế độ ngang

Để xem trước bố cục ứng dụng **Hello Toast** ở chế độ ngang (horizontal orientation), hãy làm theo các bước sau:

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

**Lưu ý:** Nếu bạn đã tải xuống mã nguồn của **HelloToast**, bạn cần xóa các bố cục **chế độ ngang (landscape)** và **màn hình cực lớn (extra-large)** đã hoàn thành mà bạn sẽ tạo trong nhiệm vụ này. Chọn cả thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**. Sau đó, chuyển bảng **Project** trở lại **Project > Android**.



2. Mở tệp bố cục **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. Trong menu thả xuống, chọn **Switch to Landscape**. Bố cục sẽ hiển thị ở chế độ **ngang (horizontal orientation)**. Để quay lại chế độ **dọc (vertical orientation)**, chọn **Switch to Portrait**.

---

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

Sự khác biệt trực quan giữa **chế độ dọc (vertical)** và **chế độ ngang (horizontal)** của bố cục này là: **Chữ số (0) trong TextView show\_count quá thấp trong chế độ ngang**—quá gần với **nút Count**. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, **TextView** có thể hiển thị quá lớn hoặc không căn giữa do kích thước chữ cố định là **160sp**.

### Cách khắc phục:

Bạn có thể **tạo một biến thể** của bố cục ứng dụng **Hello Toast** dành riêng cho **chế độ ngang** mà không ảnh hưởng đến **chế độ dọc**.

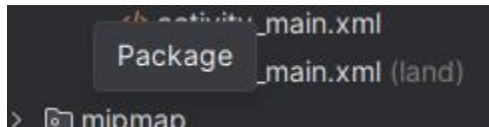
### Các bước thực hiện:

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

Một cửa sổ trình chỉnh sửa mới sẽ mở với tab **land/activity\_main.xml**, hiển thị bố cục dành cho **chế độ ngang (horizontal orientation)**.

Bạn có thể thay đổi bố cục này **mà không ảnh hưởng đến bố cục gốc ở chế độ dọc**.

3. Trong bảng **Project > Android**, mở thư mục **res > layout**. Lúc này, **Android Studio** sẽ tự động tạo một biến thể bố cục mới có tên **activity\_main.xml (land)**.



---

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

Bạn có thể **xem trước bố cục trên các thiết bị khác nhau** mà không cần chạy ứng dụng trên thiết bị hoặc trình giả lập.

### Các bước thực hiện:

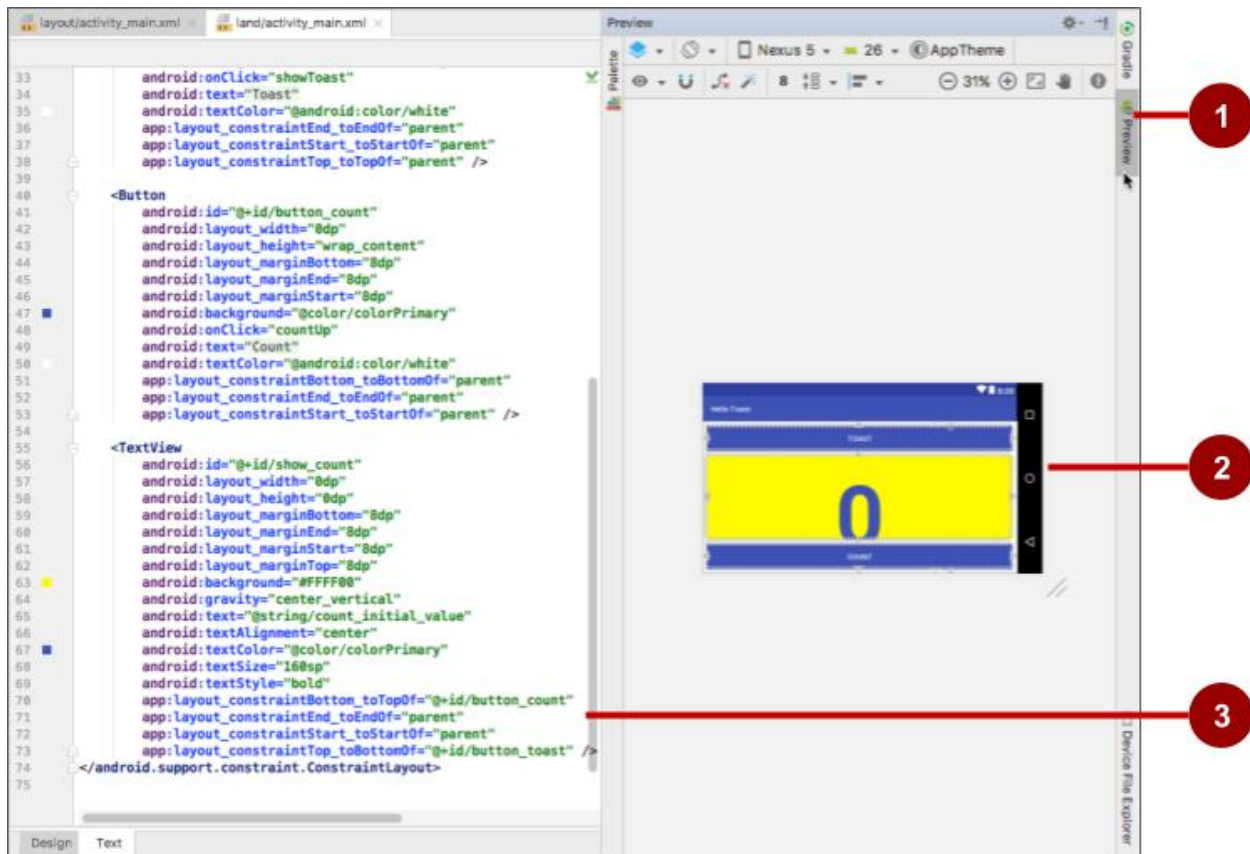
1. Tab **land/activity\_main.xml** vẫn đang mở trong trình chỉnh sửa bố cục. Nếu không, nhấp đúp vào tệp **activity\_main.xml (land)** trong thư mục **layout**.
2. Nhấp vào nút **Device in Editor** trên thanh công cụ trên cùng.
3. Chọn một thiết bị khác từ danh sách thả xuống. Ví dụ: chọn **Nexus 4**, **Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong bản xem trước. Những khác biệt này xảy ra do **kích thước văn bản cố định** của **TextView**.

---

## 1.4 Thay đổi bố cục cho chế độ ngang

Bạn có thể sử dụng **bảng Attributes trong tab Design** để thiết lập hoặc thay đổi thuộc tính, nhưng đôi khi **chỉnh sửa trực tiếp mã XML** sẽ nhanh hơn.

**Tab Text** hiển thị **mã XML**, đồng thời cung cấp **tab Preview** ở phía bên phải cửa sổ để xem trước bố cục.



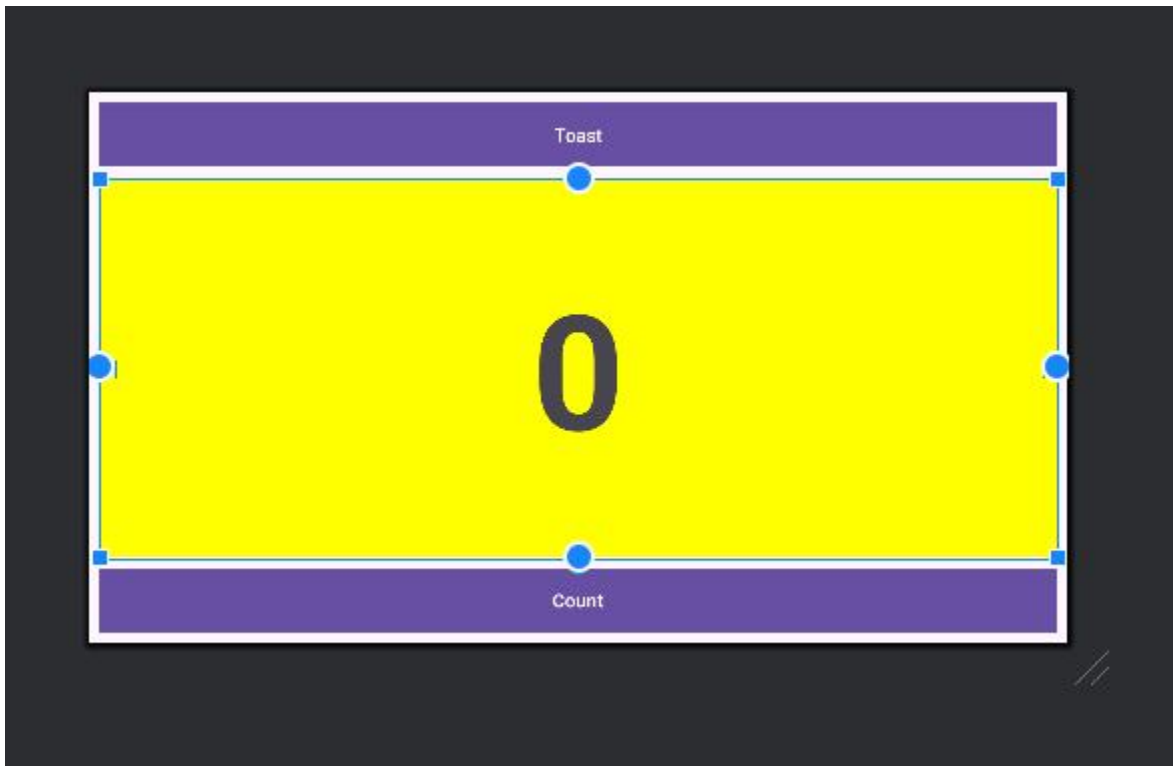
**Mô tả trong hình minh họa:**

1. **Tab Preview**, dùng để hiển thị ngăn xem trước.
2. **Ngăn xem trước**, nơi hiển thị giao diện bố cục.
3. **Mã XML**, chứa cấu trúc của bố cục.

**Các bước thay đổi bố cục:**

1. Tab **land/activity\_main.xml** vẫn đang mở trong trình chỉnh sửa bố cục. Nếu không, nhấp đúp vào tệp **activity\_main.xml (land)** trong thư mục **layout**.
2. Nhấp vào **tab Text** và **tab Preview** (nếu chưa được chọn).
3. Tìm phần tử **TextView** trong mã XML.

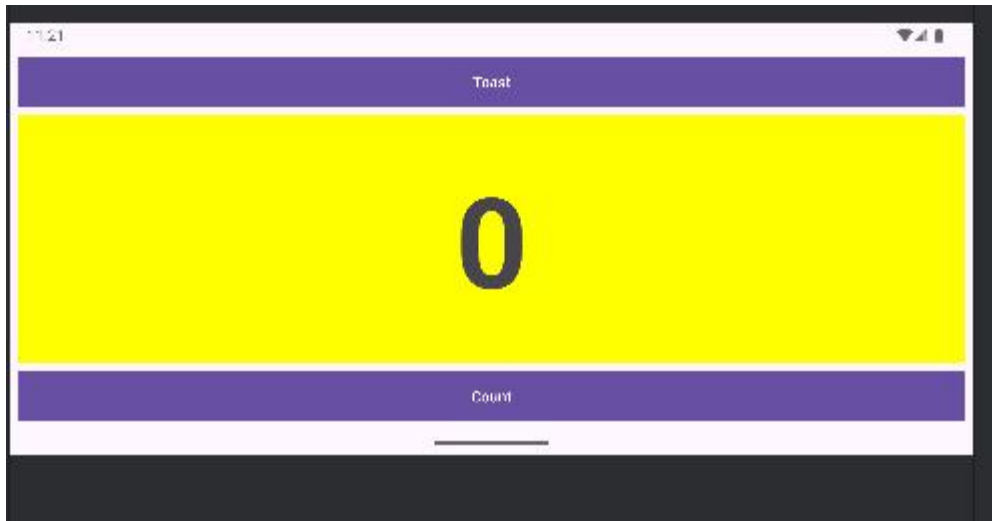
4. Thay đổi thuộc tính: `android:textsize = "160sp"` -> `"120sp"` **Bố cục xem trước sẽ cập nhật thay đổi ngay lập tức.**



5. Chọn các thiết bị khác nhau trong menu **Device in Editor** để kiểm tra giao diện trên nhiều màn hình khác nhau.

Trong trình chỉnh sửa, **tab land/activity\_main.xml** hiển thị bố cục cho **chế độ ngang**, còn **tab activity\_main.xml** hiển thị bố cục cho **chế độ dọc**. Bạn có thể chuyển đổi giữa hai chế độ bằng cách nhấp vào các tab.

6. Chạy ứng dụng trên trình giả lập hoặc thiết bị thật, sau đó xoay màn hình từ **dọc sang ngang** để xem hai bố cục hoạt động.



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

Như bạn đã học trước đó, bạn có thể **xem trước bố cục trên các thiết bị khác nhau** bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ trên cùng.

Nếu bạn chọn một thiết bị như **Nexus 10 (máy tính bảng)** từ menu, bạn sẽ thấy rằng **bố cục hiện tại không phù hợp** với màn hình máy tính bảng:

- Văn bản trên mỗi **Button** quá nhỏ.
- Cách sắp xếp các **Button** ở trên và dưới **không lý tưởng** cho màn hình lớn của máy tính bảng.

### Cách khắc phục:

Bạn có thể **tạo một biến thể bố cục dành riêng cho máy tính bảng**, mà **không ảnh hưởng** đến các bố cục dành cho **điện thoại ở chế độ ngang và dọc**.

---

### Các bước thực hiện:

Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị bảng thiết kế và sơ đồ bố cục (**blueprint**).

Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.

Chọn **Create layout x-large Variation**. Một cửa sổ trình chỉnh sửa mới sẽ mở với tab **xlarge/activity\_main.xml**, hiển thị bố cục dành cho **thiết bị có màn hình lớn**



(máy tính bảng). Trình chỉnh sửa cũng sẽ tự động chọn một thiết bị máy tính bảng để xem trước, chẳng hạn như **Nexus 9 hoặc Nexus 10**.

Bạn có thể **chỉnh sửa bố cục này** dành riêng cho máy tính bảng **mà không làm thay đổi các bố cục khác**.

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

Bạn có thể sử dụng **bảng thuộc tính (Attributes pane)** trong tab **Design** để thay đổi các thuộc tính của bố cục này.

---

Các bước thực hiện:

### 1. Tắt công cụ Autoconnect

- Trên thanh công cụ, đảm bảo rằng **công cụ Autoconnect đã bị tắt**.

### 2. Xóa tất cả các ràng buộc trong bố cục

- 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. Thay đổi kích thước TextView

- Trong **Component Tree**, chọn **TextView** có tên **show\_count**.
- Để dễ dàng di chuyển các **Button**, hãy kéo một góc của **TextView** để **thay đổi kích thước**.

**Lưu ý:** Việc thay đổi kích thước sẽ gán **kích thước cố định (hardcoded)** cho phần tử. Tuy nhiên, **tránh cố định kích thước** cho hầu hết các phần tử vì điều này có thể ảnh hưởng đến giao diện trên các màn hình có kích thước và mật độ khác nhau. Ở bước này, bạn **tạm thời thay đổi kích thước để di chuyển phần tử**, và sẽ chỉnh lại sau.

### 4. Thay đổi thuộc tính của Button "button\_toast"

- Trong **Component Tree**, chọn **button\_toast**.
- Mở **tab Attributes**, thực hiện thay đổi:

1. **textSize** → **60sp**
2. **layout\_width** → **wrap\_content**

**Mẹo:** Bạn có thể đặt **layout\_width** thành **wrap\_content** bằng hai cách:

- Cách 1: Nhấp vào **biểu tượng điều khiển chiều rộng** trên phần tử để chuyển đổi sang **Wrap Content**.
- Cách 2: Chọn trực tiếp **wrap\_content** từ menu trong **layout\_width**.

**Lợi ích của wrap\_content:**

- Nếu văn bản của **Button** được dịch sang một ngôn ngữ khác, kích thước **Button** sẽ **tự động điều chỉnh** để phù hợp với văn bản mới.

### **5. Thay đổi thuộc tính của Button "button\_count"**

- Chọn **button\_count** trong **Component Tree**.
- Thực hiện các thay đổi:
  - **textSize** → **60sp**
  - **layout\_width** → **wrap\_content**
- **Di chuyển Button** đến một vị trí trống phía trên **TextView** trong bố cục.

## **1.7 Sử dụng ràng buộc đường cơ sở (Baseline Constraint)**

Bạn có thể căn chỉnh một phần tử giao diện người dùng (**UI element**) chứa văn bản, chẳng hạn như **TextView** hoặc **Button**, với một phần tử khác cũng chứa văn bản.

**Ràng buộc đường cơ sở (Baseline Constraint)** cho phép bạn **căn chỉnh dòng cơ sở của văn bản** giữa các phần tử này.

---

**Các bước thực hiện:**

### 1. Thiết lập ràng buộc cho các Button

- Ràng buộc Button "button\_toast" vào **cạnh trên và cạnh trái** của bố cục.
- Di chuyển Button "button\_count" đến vị trí gần **button\_toast**.
- Ràng buộc Button "button\_count" vào **cạnh trái** của **button\_toast**, như trong hình minh họa động.

### 2. Sử dụng ràng buộc đường cơ sở

- Chọn **button\_count**.
- Di chuột qua **button\_count** cho đến khi **biểu tượng ràng buộc đường cơ sở** xuất hiện bên dưới phần tử.

### 3. Tạo ràng buộc đường cơ sở

- Nhấp vào **biểu tượng ràng buộc đường cơ sở**.
- **Tay cầm ràng buộc đường cơ sở (baseline handle)** sẽ xuất hiện và nhấp nháy màu xanh lá.
- Nhấp và kéo **đường ràng buộc đường cơ sở** đến **đường cơ sở** của **button\_toast** để căn chỉnh văn bản giữa hai Button.

## 1.8 Mở rộng các nút theo chiều ngang

Nút **Pack** trên thanh công cụ cung cấp các tùy chọn để **gom nhóm hoặc mở rộng** các phần tử UI đã chọn. Bạn có thể sử dụng nó để **căn chỉnh đều các nút Button theo chiều ngang** trên bố cục.

---

### Các bước thực hiện:

#### 1. Chọn các nút Button

- Trong **Component Tree**, chọn **button\_count**.
- Nhấn **Shift** và chọn **button\_toast** để chọn cả hai nút.

#### 2. Mở rộng các nút theo chiều ngang

- Nhấp vào nút *Pack* trên thanh công cụ.
- Chọn *Expand Horizontally* như trong hình minh họa.

### *3. Ràng buộc TextView show\_count*

- Ràng buộc *TextView show\_count* vào cạnh dưới của *button\_toast*, hai cạnh bên, và cạnh dưới của bố cục như trong hình động minh họa.

### *4. Chỉnh sửa thuộc tính show\_count*

- Đặt *layout\_width* và *layout\_height* của *show\_count* thành *Match Constraints*.
- Đặt *textSize* thành *200sp*.

### *5. Xem trước giao diện ngang (Landscape)*

- Nhấp vào nút *Orientation in Editor* trên thanh công cụ.
- Chọn *Switch to Landscape* để xem trước bố cục ở chế độ ngang.
- (Bạn có thể chọn *Switch to Portrait* để quay lại chế độ dọc.)

### *6. Chạy ứng dụng trên các trình giả lập khác nhau*

- Chạy ứng dụng trên các trình giả lập khác nhau.
- Thay đổi hướng màn hình sau khi chạy ứng dụng để xem giao diện trên các thiết bị có kích thước màn hình và mật độ khác nhau.

✓ Bạn đã tạo thành công một ứng dụng có giao diện phù hợp trên cả điện thoại và máy tính bảng với nhiều kích thước màn hình khác nhau!

## Task 2: Thay đổi layout thành *LinearLayout*

**LinearLayout** là một **ViewGroup** sắp xếp tập hợp các **view** của nó theo hàng **ngang** hoặc **dọc**. **LinearLayout** là một trong những layout phổ biến nhất vì nó **đơn giản và nhanh chóng**. Nó thường được sử dụng trong một **view group** khác để sắp xếp các phần tử giao diện người dùng theo **chiều ngang hoặc chiều dọc**.

---

Một *LinearLayout* yêu cầu các thuộc tính sau:

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

Giá trị của `layout_width` và `layout_height` có thể là:

- *match\_parent*: Mở rộng view để lấp đầy *parent* theo chiều rộng hoặc chiều cao. Khi *LinearLayout* là *root view*, nó sẽ mở rộng theo kích thước màn hình (*parent view*).
- *wrap\_content*: Thu nhỏ kích thước view sao cho vừa đủ chứa nội dung của nó. Nếu không có nội dung, view sẽ trở nên *không hiển thị*.
- *Số dp cố định*: Chỉ định kích thước cố định (tính theo *density-independent pixels*). Ví dụ: `16dp` có nghĩa là *16 density-independent pixels*, được điều chỉnh theo mật độ màn hình của thiết bị.

`orientation` có thể là:

- *horizontal*: Các view được sắp xếp từ trái sang phải.
- *vertical*: Các view được sắp xếp từ trên xuống dưới.

---

## 2.1 Thay đổi nhóm view gốc thành *LinearLayout*

1. Mxt ở cuối khung chỉnh sửa để xem mã XML. Ở dòng đầu tiên của ứng dụng *Hello Toast* từ nhiệm vụ trước.
2. Mở tệp *activity\_main.xml* (nếu chưa mở), sau đó nhấn vào tab Tea mã XML có thể sau:

```
<android.support.constraint.ConstraintLayout xmlns:android="http:...
```

3. Thay đổi thẻ `<android.support.constraint.ConstraintLayout>` thành `<LinearLayout>` để mã trông như sau:

```
<LinearLayout xmlns:android="http:...
```

4. Đảm bảo 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 không, hãy thay đổi thủ công.)
5. Dưới dòng `<LinearLayout>`, thêm thuộc tính sau 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 sẽ được **gạch chân màu đỏ**, vì chúng được sử dụng với **ConstraintLayout** và không phù hợp với **LinearLayout**.

## 2.2 Thay đổi thuộc tính của các phần tử cho *LinearLayout*

Thực hiện các bước sau để thay đổi thuộc tính của các phần tử UI sao cho chúng hoạt động với **LinearLayout**:

1. Mở ứng dụng *Hello Toast* từ nhiệm vụ trước.
2. Mở tệp *activity\_main.xml* (nếu chưa mở), và nhấp vào tab *Text*.
3. Tìm phần tử *button\_toast* (`Button`), và thay đổi thuộc tính sau:

Original	Change
<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"
```

5. Tìm phần tử *button\_count* (`Button`), và thay đổi thuộc tính sau:

Original	Change
----------	--------

android:layout_width="0dp"	android:layout_width="match_parent"

#### 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"

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

<i>Original</i>	<i>Change</i>
android:layout_width="0dp"	android:layout_width="match_parent"

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

app:layout\_constraintBottom\_toTopOf="@+id/button\_count"

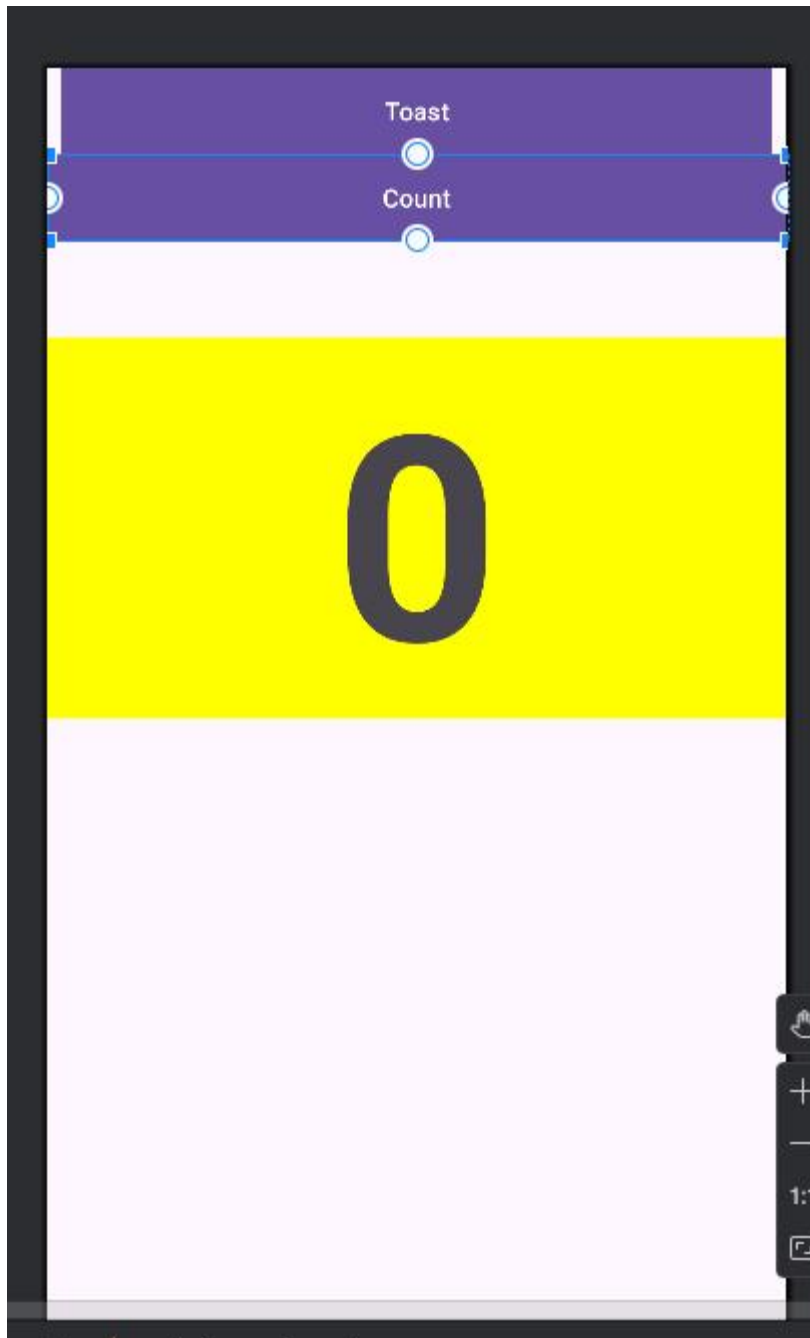
app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/button\_toast"



9. Nhấp vào tab *Preview* ở bên phải cửa sổ *Android Studio* (nếu chưa được chọn) để xem bản xem trước của *layout* cho đến thời điểm này.



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

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

Để thay đổi vị trí của chúng sao cho nút **Count** nằm ở dưới cùng, hãy thực hiện các bước sau:

1. Mở ứng dụng *Hello Toast* từ nhiệm vụ trước.
2. Mở tệp *activity\_main.xml* (nếu chưa mở), và nhấp vào tab *Text*.
3. Chọn phần tử *button\_count* (`Button`) và tắt cả các thuộc tính của nó, từ thẻ mở `<Button` đến thẻ đóng `>`, sau đó chọn *Edit > Cut*.
4. Nhấp sau thẻ đóng `>` của phần tử *TextView* nhưng trước thẻ đóng `</LinearLayout>`, sau đó chọn *Edit > Paste*.
5. (Tùy chọn) Để sửa lỗi thụt lề hoặc khoảng cách cho mục đích thẩm mỹ, chọn *Code > Reformat Code* để định dạng lại mã XML với khoảng cách và thụt lề đúng.

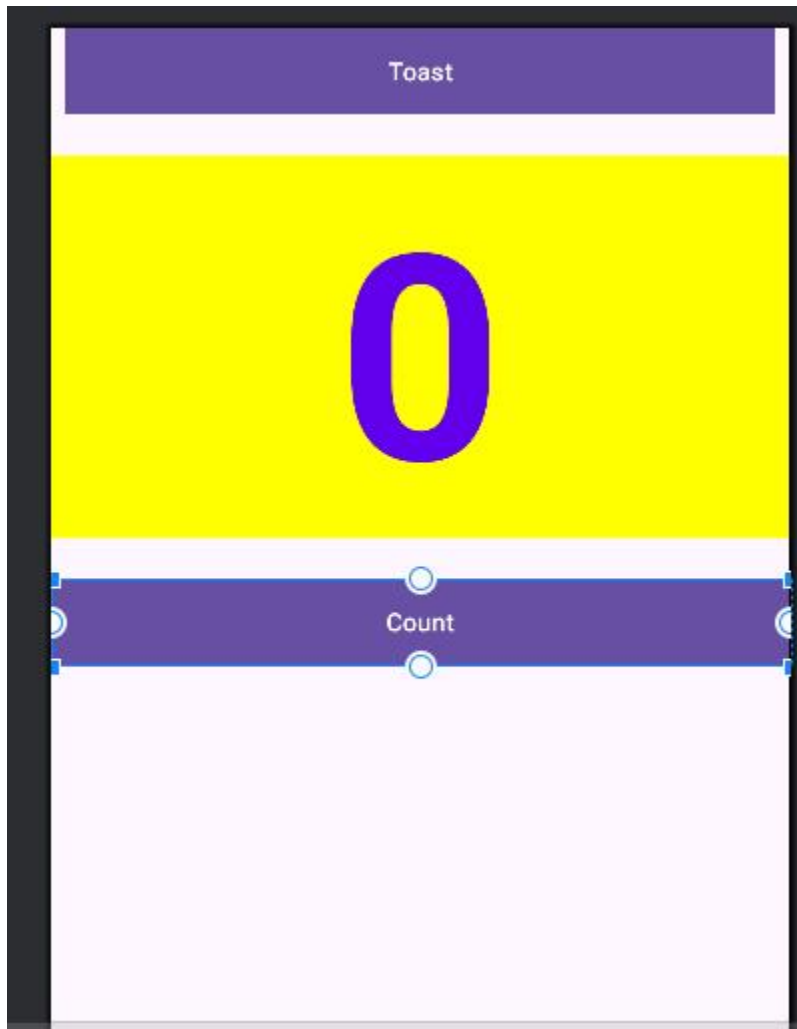
Mã XML cho các phần tử giao diện người dùng bây giờ sẽ giống như sau:

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="@color/colorPrimary"
    android:onClick="showToast"
    android:text="@string/button_label_toast"
    android:textColor="@android:color/white" />
```

```
<TextView
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="#FFFF00"
    android:gravity="center_vertical"
    android:text="@string/count_initial_value"
    android:textAlignment="center"
    android:textColor="@color/colorPrimary"
    android:textSize="160sp"
    android:textStyle="bold" />

<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:onClick="countUp"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white" />
```

Bằng cách di chuyển nút **button\_count** xuống dưới **TextView**, bố cục hiện tại đã gần giống như trước đây, với nút **Count** nằm ở dưới cùng. Bản xem trước của bố cục bây giờ sẽ trông như sau:



## 2.4 Thêm thuộc tính trọng số (*weight*) 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 đặt thuộc tính này cho **TextView** `show_count` để căn giữa nội dung (số 0) trong **TextView**.

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

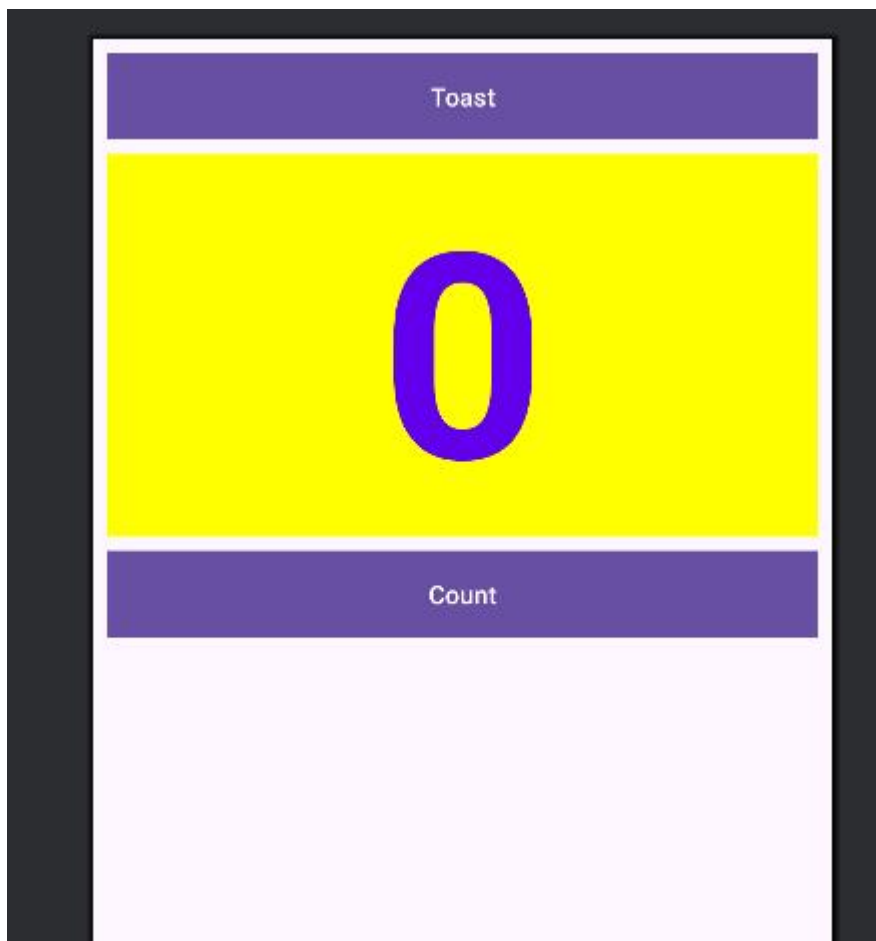
Thuộc tính `android:layout_weight` xác định mức độ không gian bổ sung trong **LinearLayout** sẽ được phân bổ cho **View**. Nếu chỉ một **View** có thuộc tính này, nó sẽ nhận toàn bộ không gian dư thừa. Nếu có nhiều **View**, không gian sẽ được chia theo tỷ lệ. Ví dụ, nếu mỗi **Button** có **weight** là 1 và **TextView** là 2, tổng là 4, thì mỗi **Button** nhận  $1/4$  không gian, còn **TextView** chiếm **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 hai nút **Toast** và **Count**. Để mở rộng **TextView** sao cho nó luôn lấp đầy không gian có sẵn bất kể thiết bị nào, hãy chỉ định thuộc tính `android:gravity` cho **TextView**. Thực hiện các bước sau:

1. Mở ứng dụng *Hello Toast* từ nhiệm vụ trước.
2. Mở tệp `activity_main.xml` (nếu chưa mở), và nhấp vào tab *Text*.
3. Tìm phần tử **TextView** `show_count`, và thêm thuộc tính sau:

```
android:layout_weight="1"
```

Sau khi thêm thuộc tính, bản xem trước sẽ giống như hình sau.



Phần tử **TextView** `show_count` giờ đây chiếm toàn bộ không gian giữa các nút. Bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ của bảng xem trước và chọn một thiết bị khác. Dù bạn chọn thiết bị nào, **TextView** `show_count` vẫn sẽ chiếm toàn bộ không gian giữa các nút.

#### 1.4) Văn bản và các chế độ cuộn

#### 1.5) Tài nguyên có sẵn

### 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**