**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](#_Toc190855147)

[Bài 1) Tạo ứng dụng đầu tiên 3](#_Toc190855148)

[1.1) Android Studio và Hello World 3](#_Toc190855149)

[1.2) Giao diện người dùng tương tác đầu tiên 30](#_Toc190855150)

[1.3) Trình chỉnh sửa bố cục 38](#_Toc190855151)

[1.4) Văn bản và các chế độ cuộn 38](#_Toc190855152)

[1.5) Tài nguyên có sẵn 38](#_Toc190855153)

[Bài 2) Activities 38](#_Toc190855154)

[2.1) Activity và Intent 38](#_Toc190855155)

[2.2) Vòng đời của Activity và trạng thái 38](#_Toc190855156)

[2.3) Intent ngầm định 38](#_Toc190855157)

[Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ 38](#_Toc190855158)

[3.1) Trình gỡ lỗi 38](#_Toc190855159)

[3.2) Kiểm thử đơn vị 38](#_Toc190855160)

[3.3) Thư viện hỗ trợ 38](#_Toc190855161)

[CHƯƠNG 2. Trải nghiệm người dùng 39](#_Toc190855162)

[Bài 1) Tương tác người dùng 39](#_Toc190855163)

[1.1) Hình ảnh có thể chọn 39](#_Toc190855164)

[1.2) Các điều khiển nhập liệu 39](#_Toc190855165)

[1.3) Menu và bộ chọn 39](#_Toc190855166)

[1.4) Điều hướng người dùng 39](#_Toc190855167)

[1.5) RecycleView 39](#_Toc190855168)

[Bài 2) Trải nghiệm người dùng thú vị 39](#_Toc190855169)

[2.1) Hình vẽ, định kiểu và chủ đề 39](#_Toc190855170)

[2.2) Thẻ và màu sắc 39](#_Toc190855171)

[2.3) Bố cục thích ứng 39](#_Toc190855172)

[Bài 3) Kiểm thử giao diện người dùng 39](#_Toc190855173)

[3.1) Espresso cho việc kiểm tra UI 39](#_Toc190855174)

[CHƯƠNG 3. Làm việc trong nền 39](#_Toc190855175)

[Bài 1) Các tác vụ nền 39](#_Toc190855176)

[1.1) AsyncTask 39](#_Toc190855177)

[1.2) AsyncTask và AsyncTaskLoader 39](#_Toc190855178)

[1.3) Broadcast receivers 39](#_Toc190855179)

[Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền 39](#_Toc190855180)

[2.1) Thông báo 39](#_Toc190855181)

[2.2) Trình quản lý cảnh báo 39](#_Toc190855182)

[2.3) JobScheduler 39](#_Toc190855183)

[CHƯƠNG 4. Lưu dữ liệu người dùng 40](#_Toc190855184)

[Bài 1) Tùy chọn và cài đặt 40](#_Toc190855185)

[1.1) Shared preferences 40](#_Toc190855186)

[1.2) Cài đặt ứng dụng 40](#_Toc190855187)

[Bài 2) Lưu trữ dữ liệu với Room 40](#_Toc190855188)

[2.1) Room, LiveData và ViewModel 40](#_Toc190855189)

[2.2) Room, LiveData và ViewModel 40](#_Toc190855190)

3.1) Trinfh gowx loi ……………………………………………………………………...

# LÀM QUEN

## Tạo ứng dụng đầu tiên

### Android Studio và Hello World

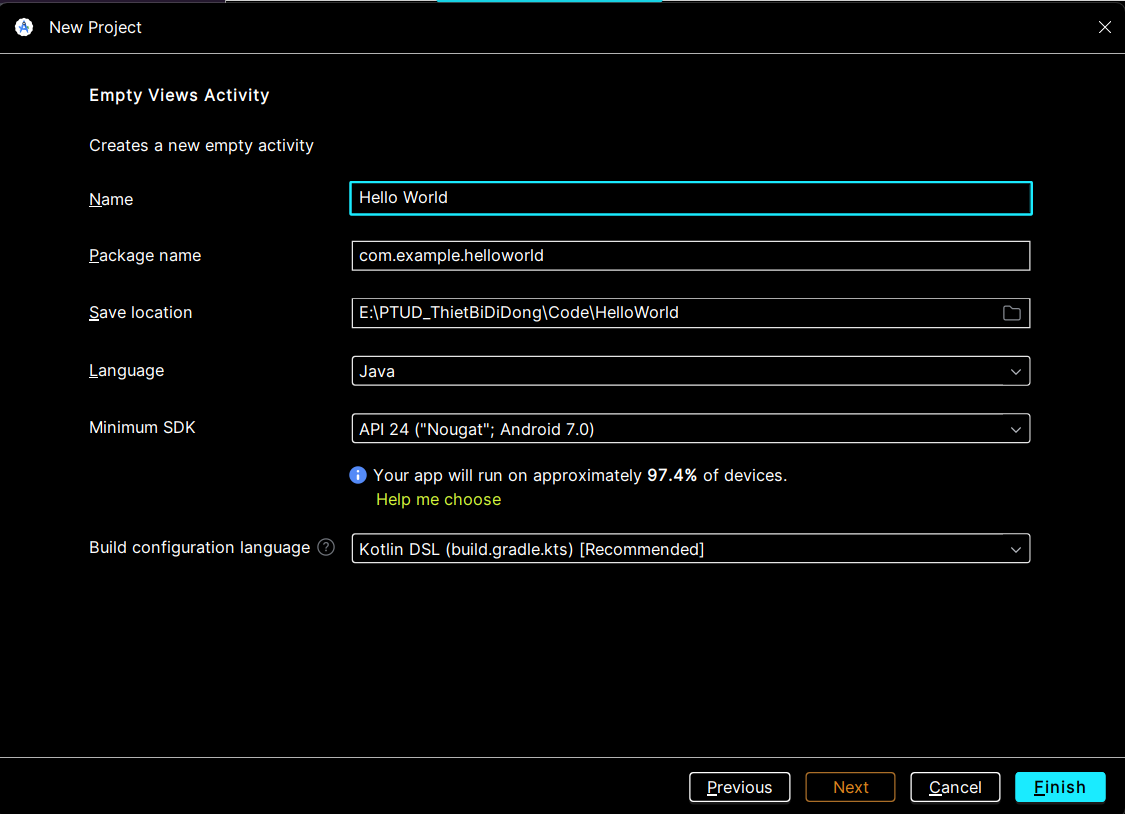
Giới thiệu

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.

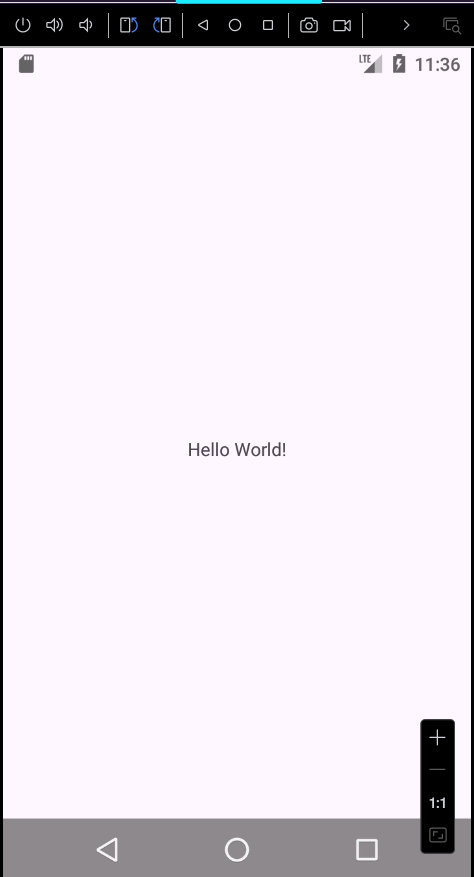
Những gì bạn sẽ làm

* 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**

**Giới thiệu về ứng dụng**

Sau khi cài đặt thành công Android Studio, bạn sẽ tạo một dự án mới từ mẫu có sẵn cho ứng dụng Hello World. Ứng dụng đơn giản này sẽ hiển thị dòng chữ “Hello World” trên màn hình của thiết bị Android ( trên máy ảo và máy thật).

Hình ảnh của ứng dụng hoàn chỉnh sẽ trông như sau:



**Bài 1: Cài đặt Android Studio**

Android Studio cung cấp một môi trường phát triển thích hợp (IDE) hoàn chỉnh, bao gồm một trình chỉnh sửa 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 hiệu suất, giúp việc phát triển ứng dụng trở nên nhanh chóng và dễ dàng hơn. Bạn có thể kiểm thử ứng dụng trên nhiều trình giả lập được cấu hình sẵn hoặc trên thiết bị di động của chính mình, xây dựng ứng dụng hoàn chỉnh và xuất bản lên cửa hang Google Play.

Lưu ý: Android Studio liên tục được cải tiến. Để biết thông tin mới nhất về yêu cầu hệ thống và hướng dẫn cài đặt, hãy xem tài liệu chính thức của Android Studio.

Android Studio có sẵn cho máy tính chạy Windowns, Linux và macOS. Phiên bản mới nhất của OpenJDK (Java Development Kit) đã được tích hợp sẵn trong Android Studio.

Để cài đặt và thiết lập Android Studio, trước tiên hãy kiểm tra yêu cầu hệ thống để đảm bảo thiết bị của bạn đáp ứng đủ điều kiện. Quy trình cài đặt tương tự trên tất cả các hệ điều hành, bất kỳ khác biết nào sẽ được ghi chú riêng.

Các bước cài đặt:

Truy cập trang web chính thức của Android Developers và làm theo hướng dẫn để tải xuống và cài đặt Androi Studio.

Chấp nhận các thiết lập mạc định trong suốt quá trình cài đặt, đồng thời đảm bảo rằng tất cả các thành phần cần thiết đều được chọn để cài đặt.

Sau khi quá trình cài đặt hoàn tất, Setup Wizard sẽ tiếp tục tải xuống và cài đặt một số thành phần bổ sung, bao gồm cả Android SDK. Hãy kiên nhẫn, vì quá trình này có thể mất thời gian tùy thuộc vào tốc độ Internet của bạn. Một số bước có thể trông giống nhau nhưng vẫn cần thiết.

Quá tringh tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng để tạo dự án đầu tiên của mình.

Xử lý sự cố:

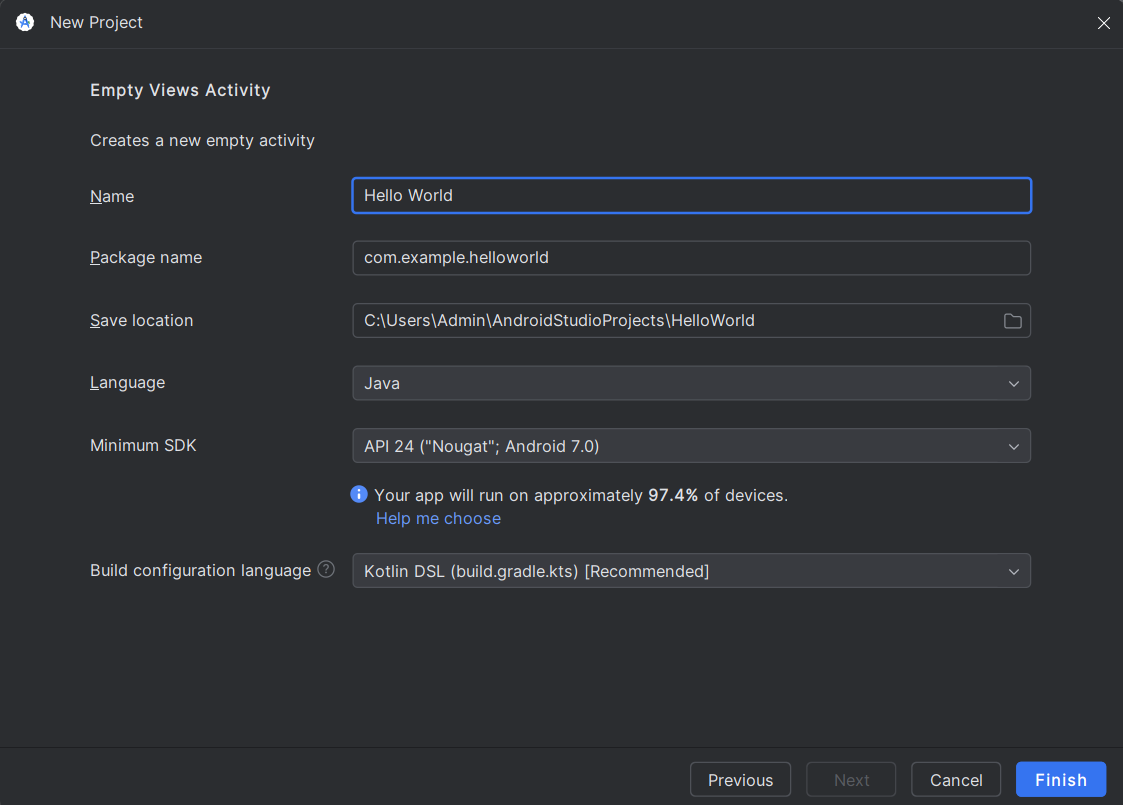
Nếu gặp sự cố khi cài đặt, hãy kiểm tra ghi chú phát hành của Android Studio hoặc tìm sự trợ giúp từ giảng viên của bạn.

**Bài 2: Tạo ứng dụng "Hello World"**

Trong bài tập này, bạn sẽ tạo một ứng dụng hiển thị dòng chữ "**Hello World**" để kiểm tra xem Android Studio đã được cài đặt đúng chưa, đồng thời làm quen với các bước phát triển ứng dụng trong Android Studio.

**2.1 Tạo dự án ứng dụng**

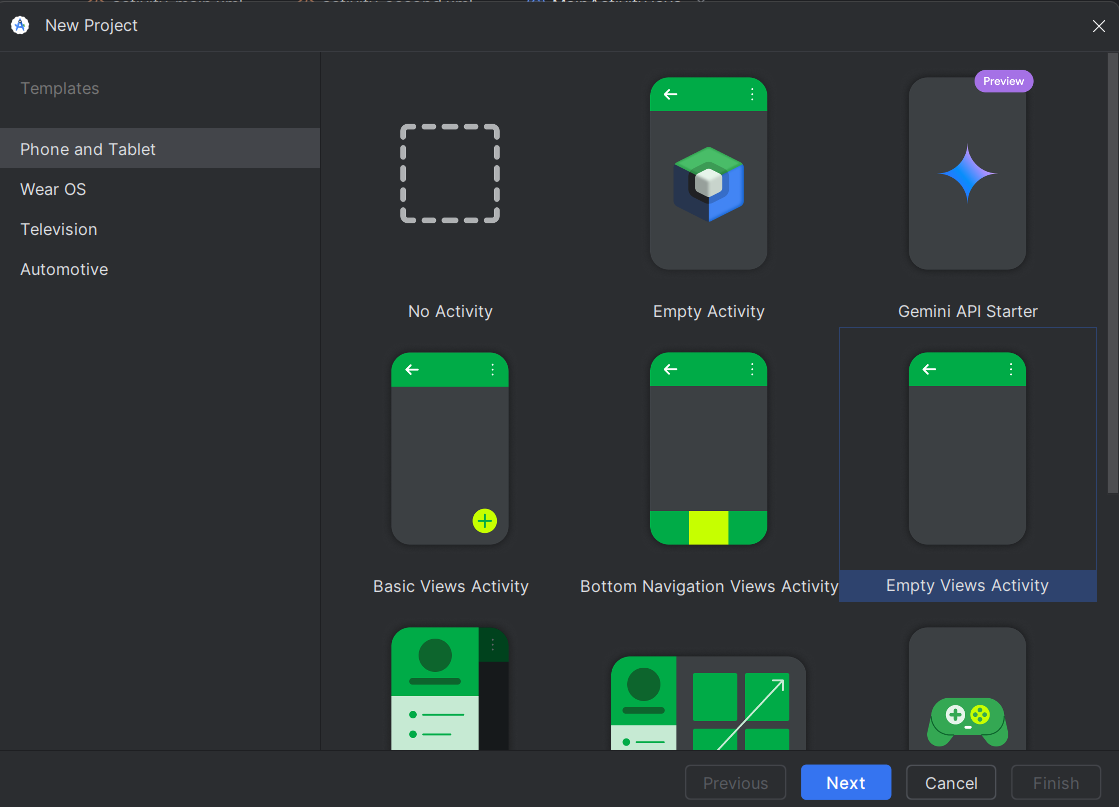
1. Mở **Android Studio** nếu nó chưa được mở.
2. Trong cửa sổ **Welcome to Android Studio**, nhấn vào **Start a new Android Studio project** (Bắt đầu một dự án Android Studio mới).
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** vào ô **Application name** (Tên ứng dụng).



1. Kiểm tra xem Project location (vị trí lưu dự án) mặc định có đúng nơi bạn muốn lưu ứng dụng Hello World và các dự án Android Studio khác không. Nếu cần, hãy thay đổi sang thư mục bạn muốn.
2. Giữ nguyên android.example.com làm Company Domain (tên miền công ty) hoặc tạo một tên miền riêng.

Nếu bạn không có ý định xuất bản ứng dụng, có thể giữ nguyên giá trị mặc định. Lưu ý rằng việc thay đổi package name (tên gói) sau này sẽ tốn thêm công sức.

1. Bỏ chọn các tùy chọn Include C++ support (Hỗ trợ C++) và Include Kotlin support (Hỗ trợ Kotlin), sau đó nhấn Next.
2. Ở màn hình Target Android Devices, đảm bảo rằng Phone and Tablet (Điện thoại và Máy tính bảng) được chọn. Kiểm tra xem API 15: Android 4.0.3 IceCreamSandwich có được đặt làm Minimum SDK (SDK tối thiểu) không; nếu không, hãy sử dụng menu thả xuống để đặt lại. Các bài học trong khóa học này sử dụng các cài đặt này, giúp ứng dụng Hello World tương thích với 97% thiết bị Android đang hoạt động trên Google Play Store.
3. Bỏ chọn Include Instant App support (Hỗ trợ Instant App) và tất cả các tùy chọn khác, sau đó nhấn Next. Nếu dự án của bạn yêu cầu cài đặt thêm thành phần cho SDK mục tiêu, Android Studio sẽ tự động cài đặt chúng.
4. Cửa sổ Add an Activity xuất hiện. Activity là một màn hình giao diện trong ứng dụng, nơi người dùng có thể thực hiện một hành động cụ thể. Mỗi Activity thường có một layout (bố cục giao diện) xác định cách hiển thị các thành phần UI trên màn hình. Android Studio cung cấp các mẫu Activity để giúp bạn bắt đầu nhanh hơn. Đối với dự án Hello World, hãy chọn Empty Activity (Hoạt động trống), sau đó nhấn Next.



1. Màn hình Configure Activity xuất hiện (có thể khác nhau tùy vào mẫu bạn chọn ở bước trước). Mặc định, Activity được đặt tên là MainActivity. Bạn có thể đổi nếu muốn, nhưng bài học này sử dụng MainActivity.
2. Đảm bảo rằng tùy chọn Generate Layout file (Tạo tệp bố cục) được chọn. Tên bố cục mặc định là activity\_main. Bạn có thể đổi nếu muốn, nhưng bài học này sử dụng activity\_main.
3. Đảm bảo rằng tùy chọn Backwards Compatibility (App Compat) (Hỗ trợ tương thích ngược) được chọn. Điều này giúp ứng dụng tương thích với các phiên bản Android cũ hơn.
4. Nhấn Finish để hoàn tất quá trình tạo dự án.

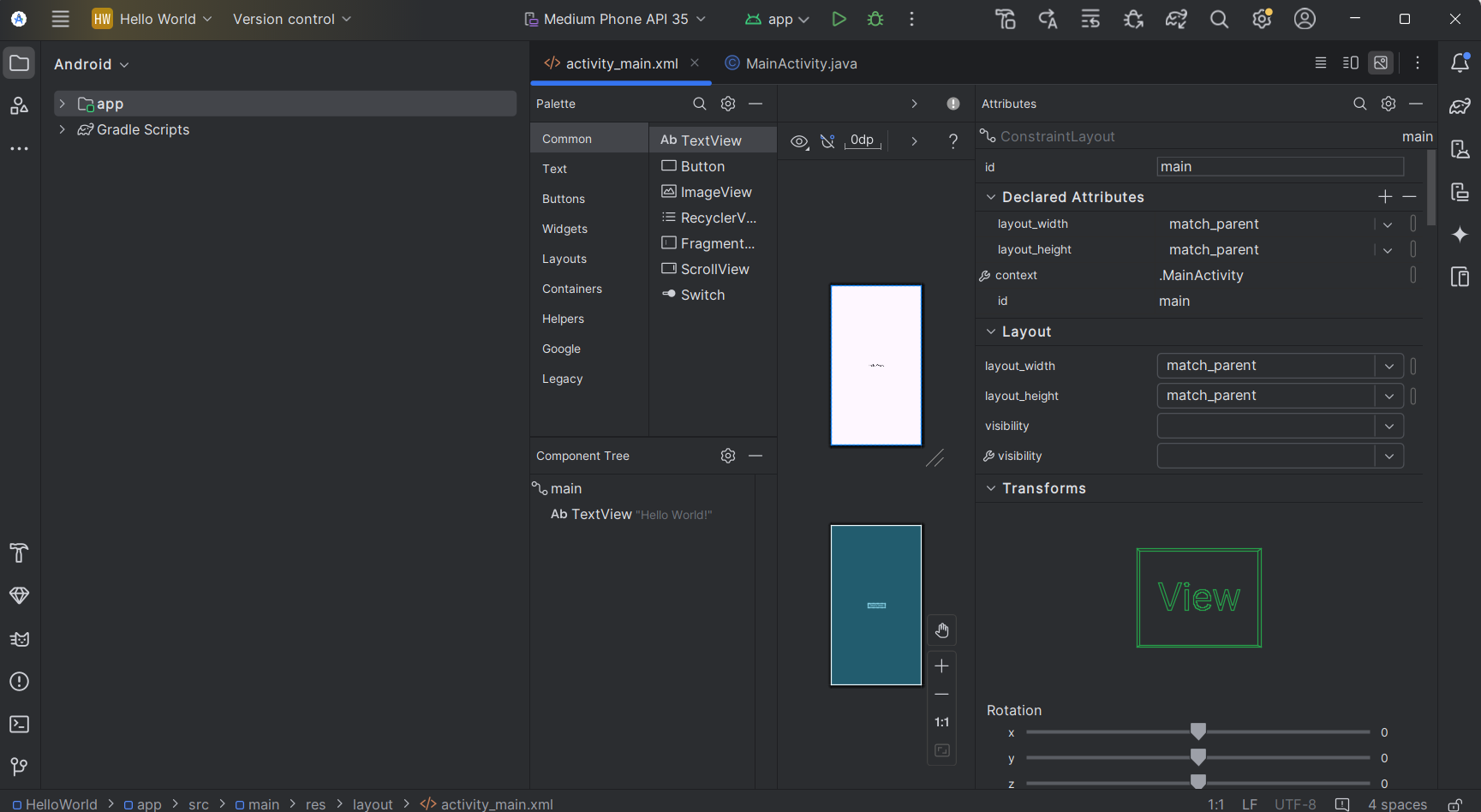
Android Studio tạo một thư mục cho các dự án của bạn và xây dựng dự án bằng Gradle (quá trình này có thể mất một vài phút).

**Mẹo:** Xem trang dành cho nhà phát triển **Cấu hình bản dựng (Configure your build)**để biết thông tin chi tiết.

Bạn cũng có thể thấy một thông báo "Mẹo trong ngày" (Tip of the day) với các phím tắt và mẹo hữu ích khác. Nhấn **Close** để đóng thông báo.

Sau đó, trình soạn thảo của Android Studio sẽ xuất hiện. Hãy làm theo các bước sau:

1. Nhấp vào tab **activity\_main.xml** để xem trình chỉnh sửa bố cục (layout editor).
2. Nhấp vào tab **Design** trong trình chỉnh sửa bố cục (nếu chưa được chọn) để hiển thị bản xem trước đồ họa của bố cục như hình bên dưới.

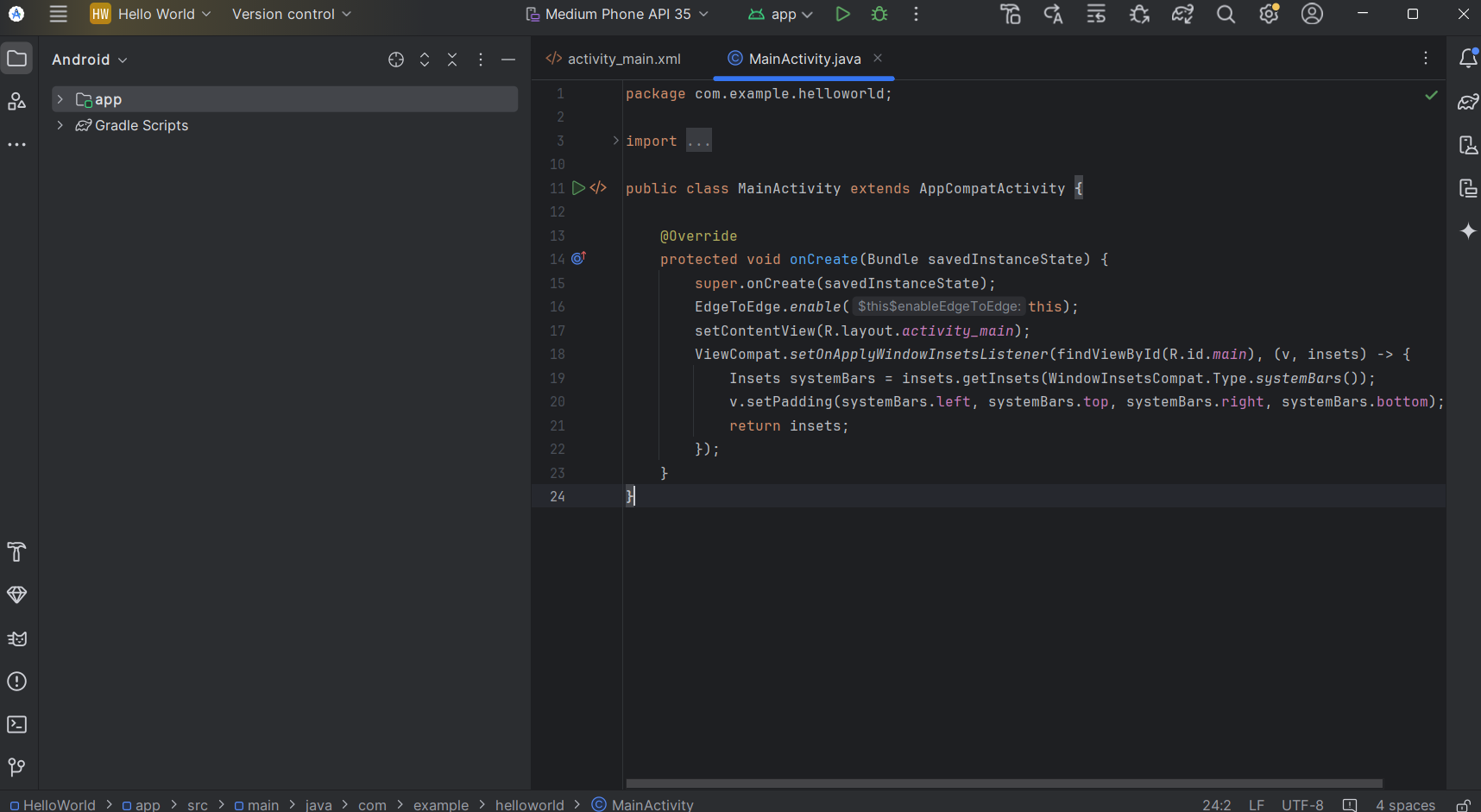


1. Nhấp vào tab **MainActivity.java** để xem trình chỉnh sửa mã nguồn như hình bên dưới.

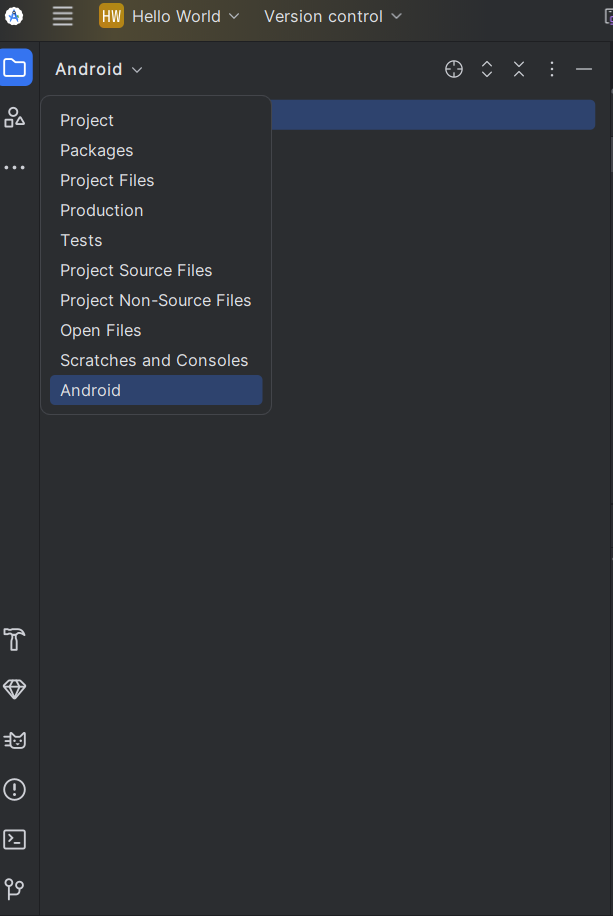
### 2.2 Khám phá ngăn ****Project > Android****

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

1. Nếu chưa được chọn, hãy nhấp vào tab **Project** trong cột tab dọc ở bên trái cửa sổ Android Studio. Ngăn **Project** sẽ xuất hiện.



1. Để xem dự án theo cấu trúc tiêu chuẩn của Android, chọn **Android** từ menu thả xuống ở đầu ngăn **Project**, như hình bên dưới.

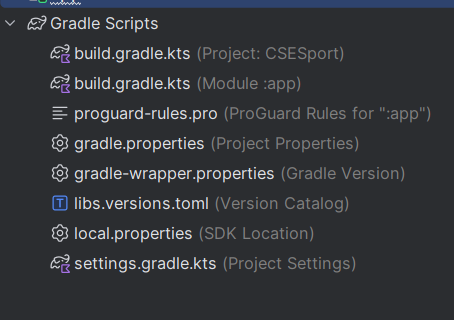


**Lưu ý:** Chương này và các chương khác sẽ gọi ngăn **Project** khi được đặt ở chế độ **Android** là **Project > Android**.

### 2.3 Khám phá thư mục ****Gradle Scripts****

Hệ thống xây dựng Gradle trong Android Studio giúp bạn dễ dàng bao gồm các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác vào bản dựng của bạn dưới dạng dependencies.

Khi bạn lần đầu tiên tạo một dự án ứng dụng, ngăn **Project > Android** xuất hiện với thư mục **Gradle Scripts** được mở rộng như hình dưới đây.



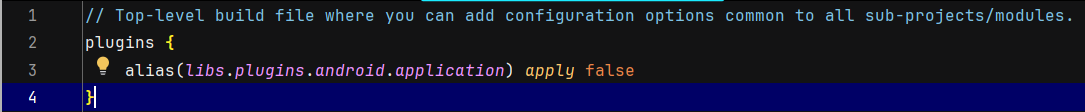
Hãy làm theo các bước sau để khám phá hệ thống Gradle:

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

Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô-đun tạo nên dự án của bạn.

Mỗi dự án trong Android Studio chứa một tệp Gradle build cấp cao nhất duy nhất. Hầu hết thời gian, bạn sẽ không cần thực hiện bất kỳ thay đổi nào đối với tệp này, nhưng việc hiểu nội dung của nó vẫn hữu ích.

Theo mặc định, tệp build cấp cao nhất sử dụng khối **buildscript** để xác định các kho lưu trữ Gradle và dependencies chung cho tất cả các mô-đun trong dự án. Khi dependency của bạn không phải là một thư viện cục bộ hoặc một thư mục tệp, Gradle sẽ tìm kiếm các tệp trong bất kỳ kho lưu trữ trực tuyến nào được chỉ định trong khối **repositories** của tệp này. Theo mặc định, các dự án Android Studio mới khai báo **JCenter** và **Google** (bao gồm **Google Maven repository**) làm vị trí kho lưu trữ.



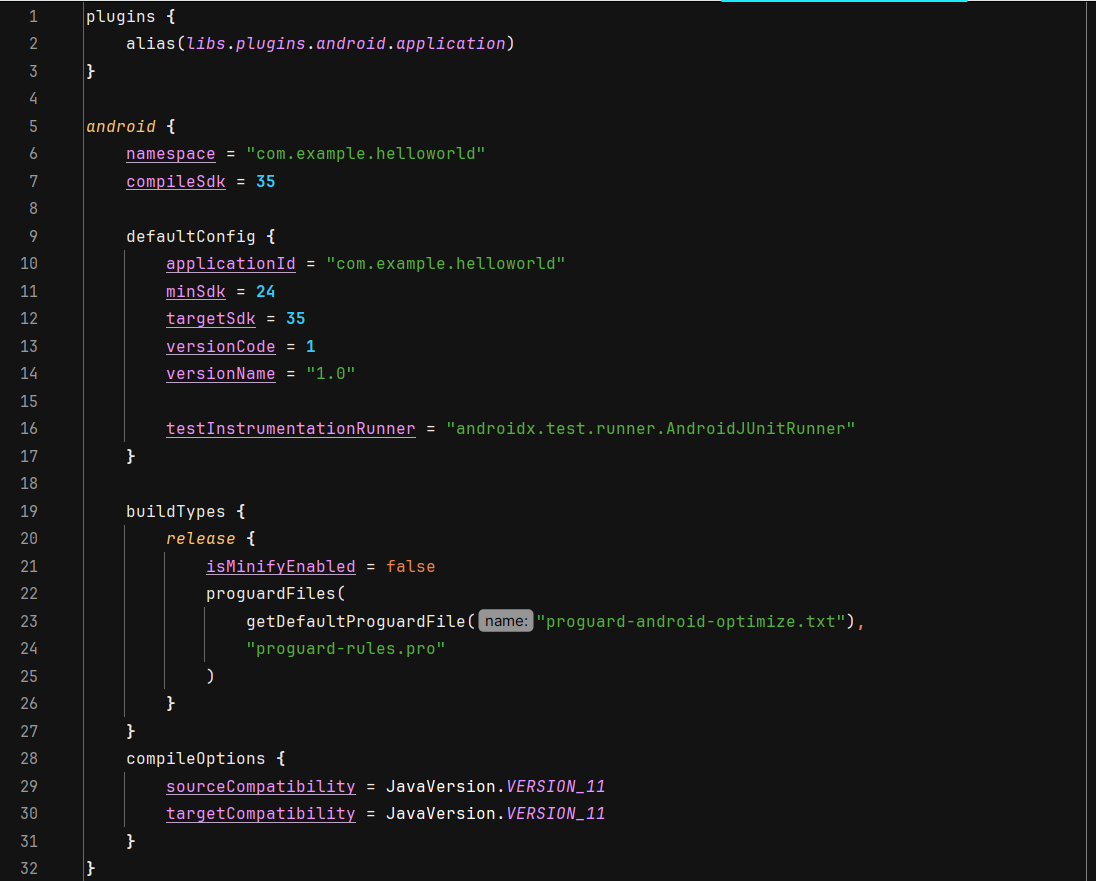
1. Tìm tệp **build.gradle (Module: app)**.

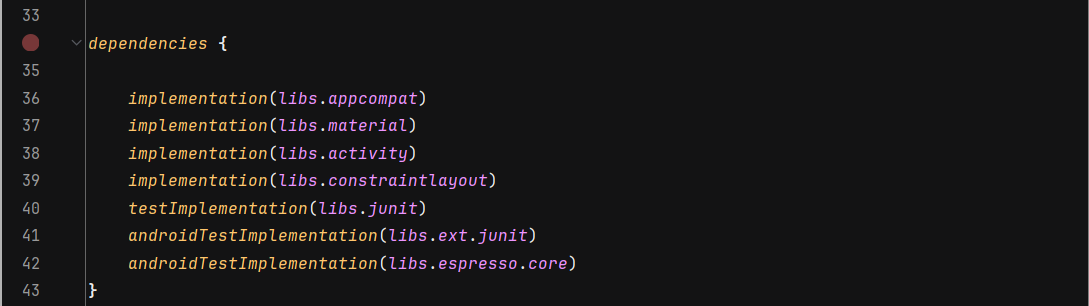
Ngoài tệp build.gradle cấp dự án, mỗi mô-đun có một tệp **build.gradle** riêng, cho phép bạn cấu hình các cài đặt bản dựng cho từng mô-đun cụ thể (ứng dụng HelloWorld chỉ có một mô-đun). Việc cấu hình các cài đặt bản dựng này cho phép bạn cung cấp các tùy chọn đóng gói tùy chỉnh, chẳng hạn như các loại bản dựng bổ sung và product flavors. Bạn cũng có thể ghi đè các cài đặt trong tệp **AndroidManifest.xml** hoặc tệp **build.gradle** cấp cao nhất.

Đây là tệp thường xuyên được chỉnh sửa nhất khi thay đổi các cấu hình cấp ứng dụng, chẳng hạn như khai báo dependencies trong phần **dependencies**. Bạn có thể khai báo một thư viện phụ thuộc bằng một trong số các cấu hình dependency khác nhau. Mỗi cấu hình dependency cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện. Ví dụ, câu lệnh: implementation fileTree(dir: 'libs', include: ['\*.jar'])

thêm tất cả các tệp “.jar” bên trong thư mục **libs** làm dependency.

**Sau đây là tệp build.gradle (Module: app) của ứng dụng HelloWorld:**



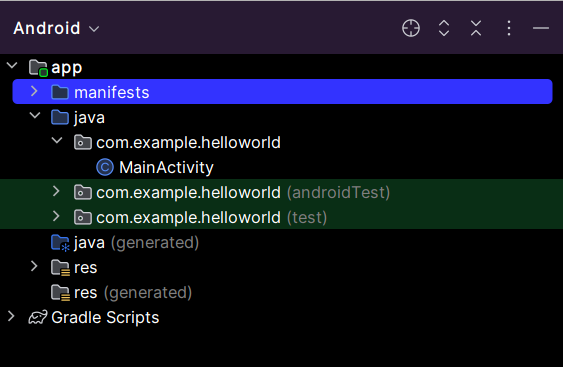


1. Nhấp vào tam giác để đóng **Gradle Scripts**.

### 2.4 Khám phá thư mục ****app**** và ****res****

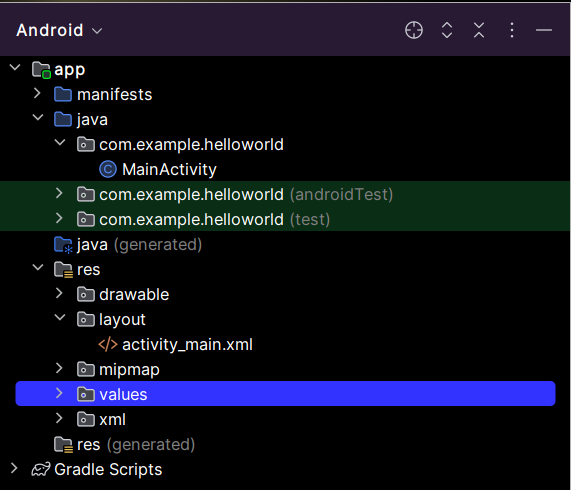
Tất cả mã nguồn và tài nguyên của ứng dụng đều nằm trong các thư mục **app** và **res**.

1. Mở rộng thư mục **app**, sau đó mở rộng thư mục **java** và thư mục **com.example.android.helloworld** để xem tệp **MainActivity.java**.
   * Nhấp đúp vào tệp này để mở nó trong trình chỉnh sửa mã nguồn.



* + Thư mục **java** chứa các tệp lớp Java trong ba thư mục con, như hình minh họa bên trên.
  + Thư mục **com.example.hello.helloworld** (hoặc tên miền 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** (màn hình) đầu tiên mà người dùng nhìn thấy, cũng là nơi khởi tạo các tài nguyên toàn cục của ứng dụng, thường được đặt là **MainActivity** (trong ngăn **Project > Android**, phần mở rộng tệp bị ẩn đi).

1. 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 bố cục (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 **Activity** của nó.

### 2.5 Khám phá thư mục ****manifests****

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

1. Mở rộng thư mục **manifests**.
2. Mở tệp **AndroidManifest.xml**.

Tệp **AndroidManifest.xml** mô tả tất cả các thành phần của ứng dụng Android của bạn.

* Mọi thành phần của ứng dụng, chẳng hạn như mỗi **Activity**, phải được khai báo trong tệp XML này.
* Trong các bài học khác của khóa học, bạn sẽ chỉnh sửa tệp này để thêm các tính năng và quyền (permissions) cho ứng dụng.
* Để tìm hiểu tổng quan, hãy xem **App Manifest Overview**.

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

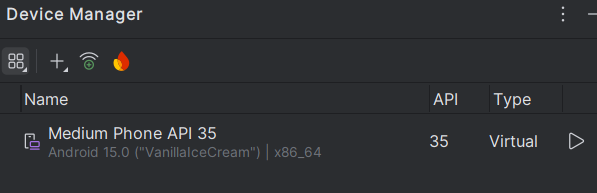
Trong nhiệm vụ này, bạn sẽ sử dụng **Android Virtual Device (AVD) Manager** để tạo một thiết bị ảo (còn gọi là trình giả lập) mô phỏng cấu hình của một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng **Android Emulator** có các yêu cầu bổ sung ngoài các yêu cầu hệ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

Sử dụng **AVD Manager**, bạn có thể xác định các đặc điểm phần cứng của thiết bị, cấp độ API, bộ nhớ, giao diện và các thuộc tính khác, sau đó lưu nó dưới dạng một thiết bị ảo. Với các thiết bị ảo, bạn có thể kiểm thử ứng dụng trên nhiều 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.

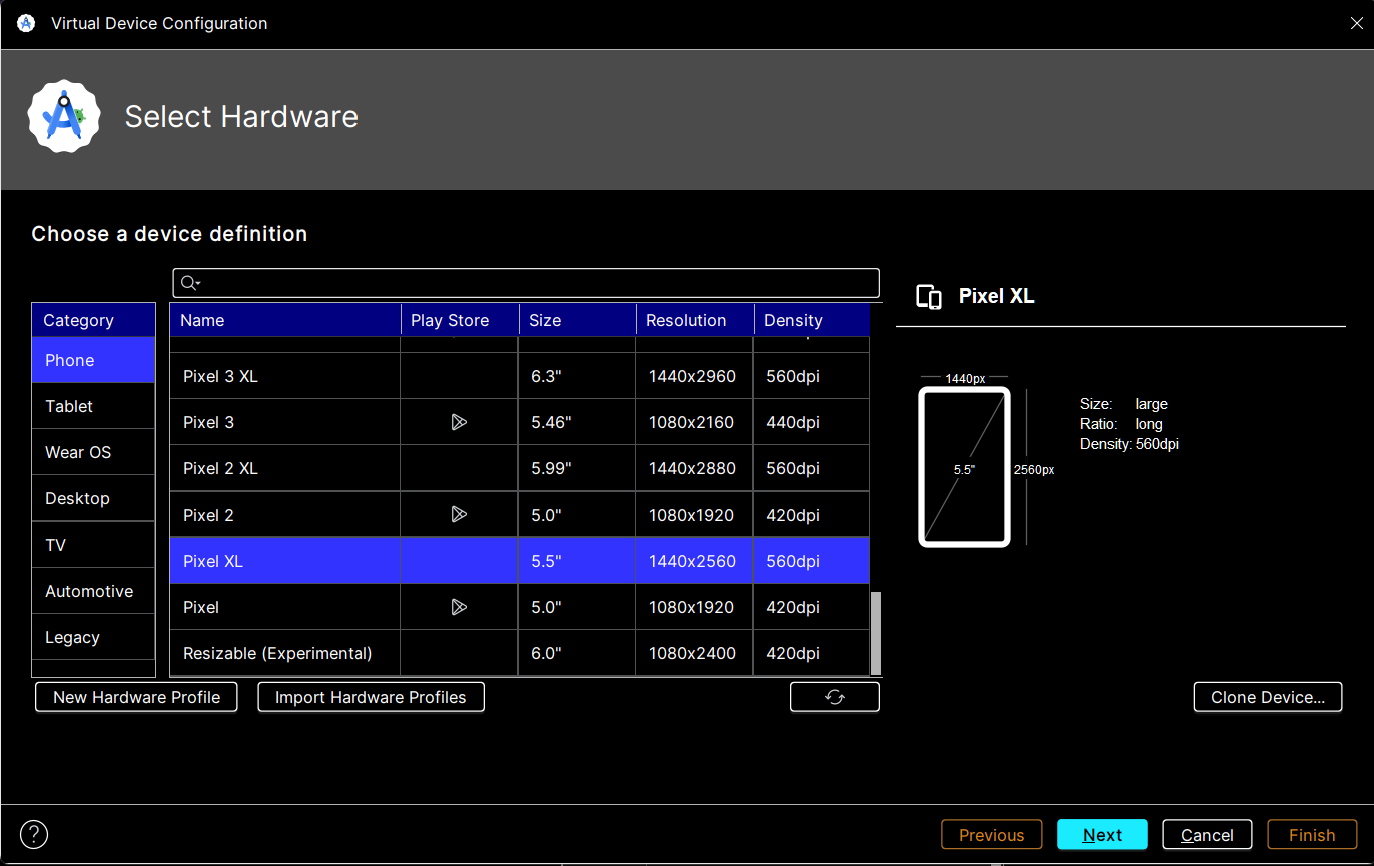
### ****3.1 Tạo một thiết bị ảo Android (AVD)****

Để chạy trình giả lập trên máy tính của bạn, trước tiên bạn phải tạo một cấu hình mô tả thiết bị ảo.

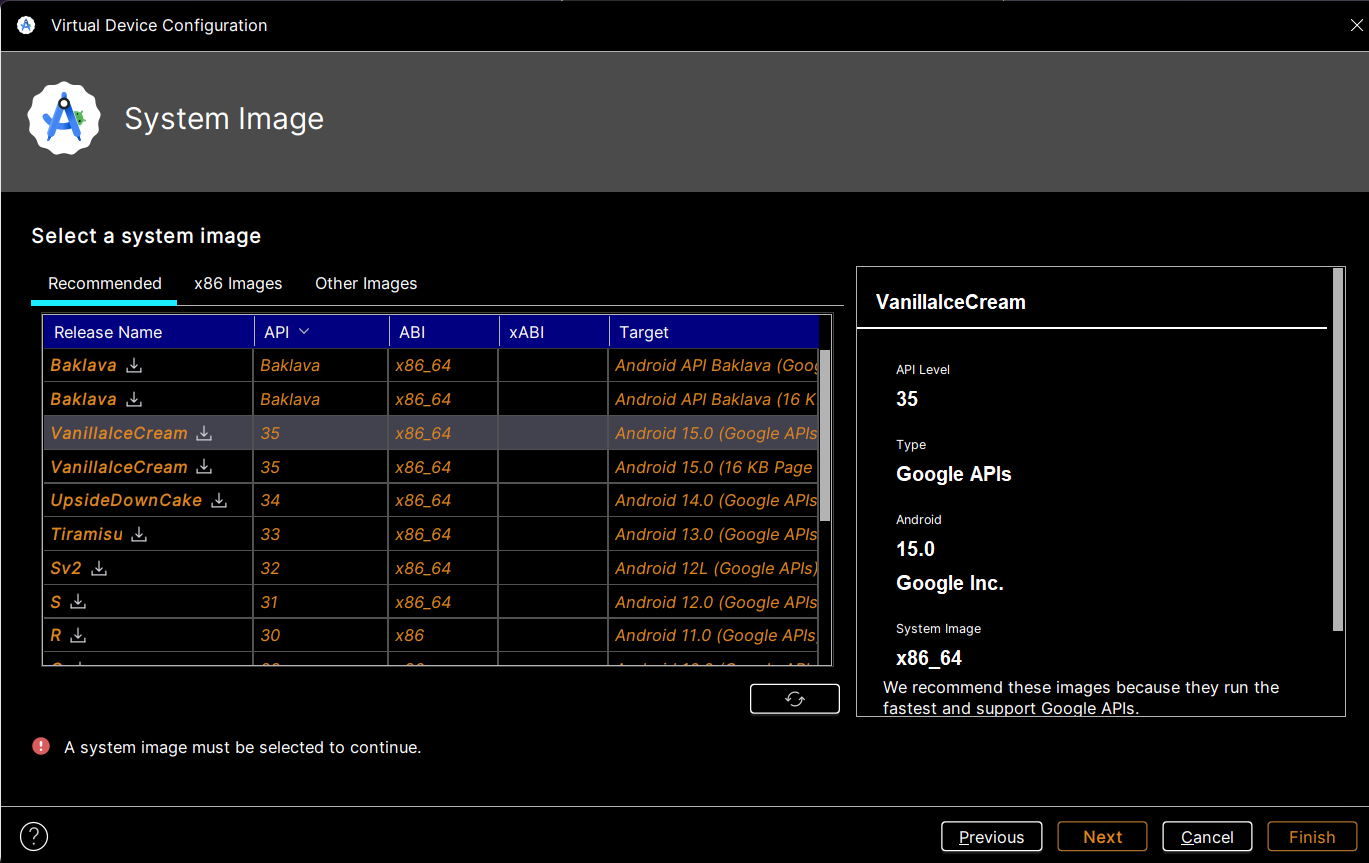
1. Trong **Android Studio**, chọn **Tools > Android > AVD Manager**, hoặc nhấp vào biểu tượng **AVD Manager** trên thanh công cụ.



1. Nhấp vào **+ Create Virtual Device**. Cửa sổ **Select Hardware** xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Mỗi thiết bị có các thông số như kích thước màn hình theo đường chéo (**Size**), độ phân giải màn hình tính theo pixel (**Resolution**), và mật độ điểm ảnh (**Density**).



1. Chọn một thiết bị, chẳng hạn như **Nexus 5X** hoặc **Pixel XL**, rồi nhấp vào **Next**. Màn hình **System Image** xuất hiện.
2. Nhấp vào tab **Recommended** (nếu chưa được chọn) và chọn phiên bản hệ điều hành Android mà bạn muốn chạy trên thiết bị ảo (chẳng hạn như **Oreo**).



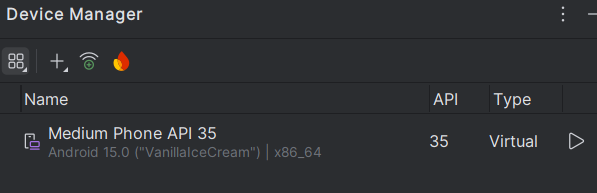
Có nhiều phiên bản hơn so với những gì được hiển thị trong tab **Recommended**. Bạn có thể xem thêm trong các tab **x86 Images** và **Other Images**. Nếu có liên kết **Download** bên cạnh hình ảnh hệ thống mà bạn muốn sử dụng, điều đó có nghĩa là nó chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống, sau đó nhấp **Finish** khi hoàn tất.

1. Sau khi chọn xong hệ thống, nhấp vào **Next**. Cửa sổ **Android Virtual Device (AVD)** xuất hiện. Bạn có thể thay đổi tên của thiết bị ảo nếu muốn. Kiểm tra lại cấu hình và nhấp vào **Finish**.

**3.2 Chạy ứng dụng trên thiết bị ảo**

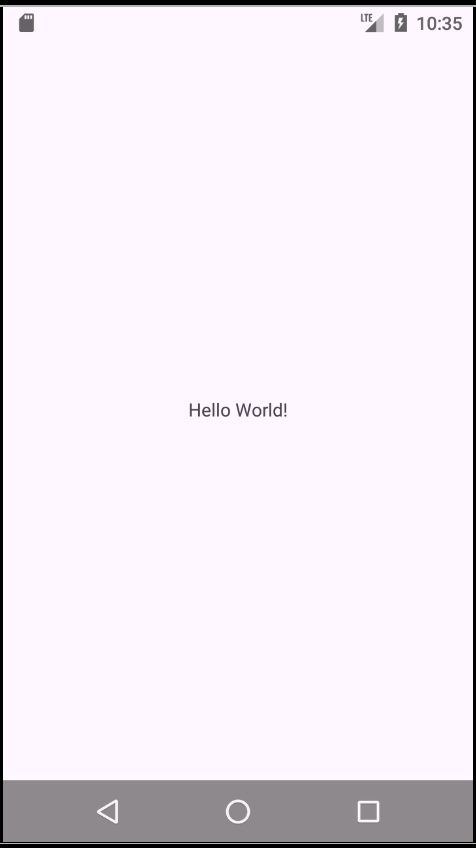
Trong nhiệm vụ này, bạn sẽ chạy ứng dụng **Hello World** của mình.

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



Trình giả lập sẽ khởi động và chạy giống như một thiết bị vật lý. Tùy thuộc vào tốc độ máy tính của bạn, quá trình này có thể mất một lúc. 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 bên dưới.



**Mẹo:** Khi kiểm thử trên thiết bị ảo, tốt nhất bạn nên khởi động nó một lần ngay từ đầu phiên làm việc và không đóng nó cho đến khi bạn hoàn thành việc kiểm thử. Điều này giúp tránh việc ứng dụng phải trải qua quá trình khởi động thiết bị nhiều lần.

Để đóng thiết bị ảo, bạn có thể:

* Nhấp vào nút **X** ở góc trên của trình giả lập.
* Chọn **Quit** từ menu.
* Nhấn **Control + Q** trên Windows hoặc **Command + Q** trên macOS.

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

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

#### **Bạn cần có:**

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

### ****4.1 Bật chế độ Gỡ lỗi USB (USB Debugging)****

Để **Android Studio** có thể giao tiếp với thiết bị của bạn, bạn phải bật **Gỡ lỗi USB** trên thiết bị Android. Tùy chọn này nằm trong 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**, thực hiện như sau:

1. Trên thiết bị, mở **Cài đặt (Settings)**, tìm **Giới thiệu về điện thoại (About phone)**, nhấn vào đó và chạm vào **Số bản dựng (Build number)** **7 lần** liên tiếp.
2. Quay lại màn hình trước đó (**Cài đặt / Hệ thống (Settings / System)**). Mục **Developer options** sẽ xuất hiện trong danh sách. Nhấn vào **Developer options**.
3. Bật tùy chọn **USB Debugging**.

### ****4.2 Chạy ứng dụng trên thiết bị****

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

1. **Kết nối** thiết bị với máy tính bằng cáp USB.
2. Nhấp vào **Run** trên thanh công cụ.
   * Cửa sổ **Select Deployment Target** sẽ mở ra, hiển thị danh sách trình giả lập và các thiết bị được kết nối.
3. **Chọn thiết bị** của bạn và nhấp **OK**.
   * **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, hãy thử các cách sau:

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

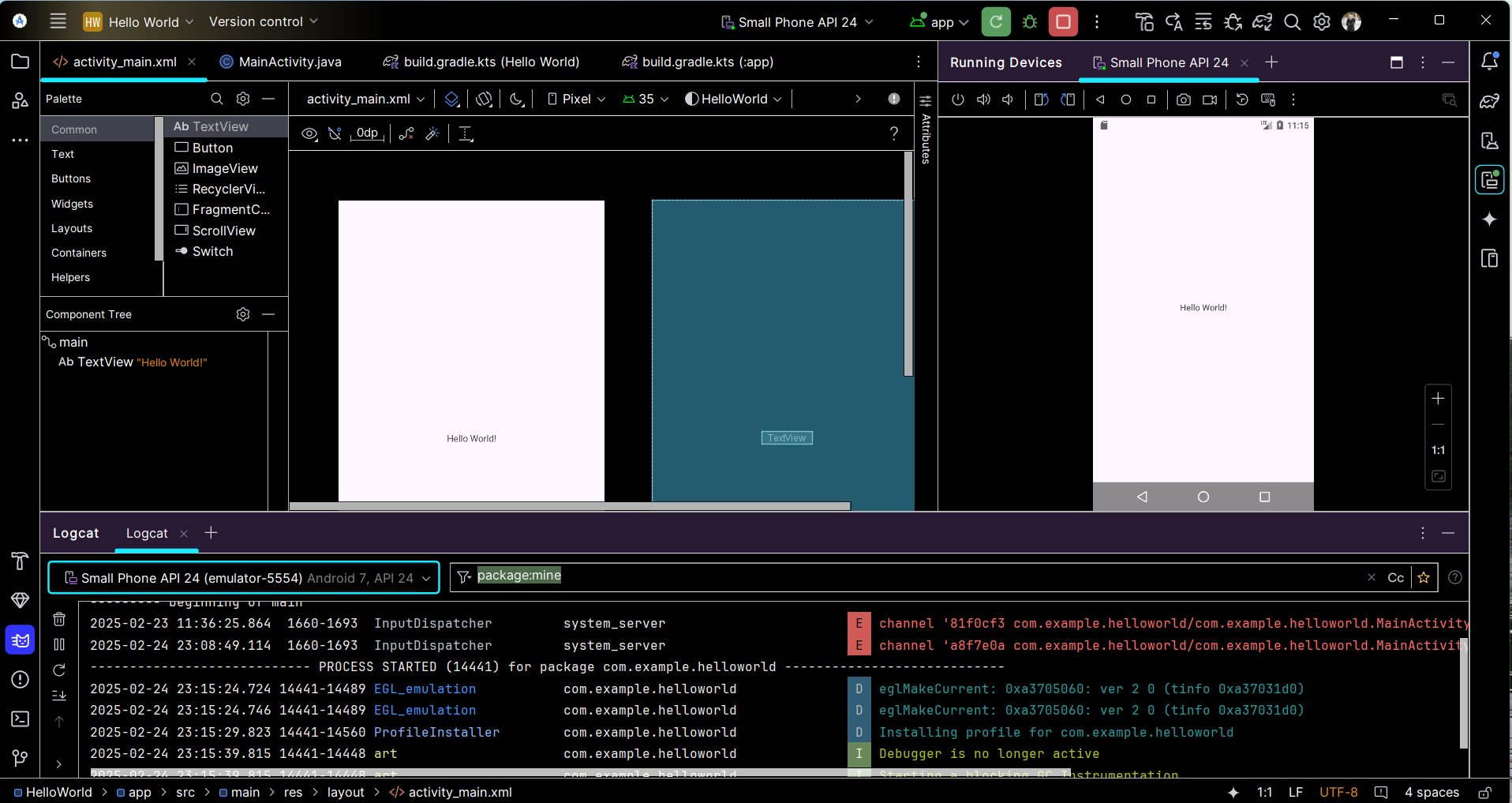
Nếu máy tính vẫn không tìm thấy thiết bị hoặc hiển thị trạng thái **"unauthorized"**, hãy làm như sau:

1. Rút kết nối thiết bị.
2. Trên thiết bị, mở **Developer Options** trong ứng dụng **Cài đặt (Settings)**.
3. Nhấn vào **Revoke USB Debugging authorizations**.
4. Kết nối lại thiết bị với máy tính.
5. Khi có thông báo yêu cầu cấp quyền, hãy chọn **Cho phép (Allow)**.

Bạn có thể cần cài đặt **trình điều khiển USB** phù hợp với thiết bị của mình. Xem tài liệu **Using Hardware Devices** để biết thêm chi tiết.

### ****6.1 Xem Logcat pane****

Để xem **Logcat pane**, hãy nhấp vào tab **Logcat** ở cuối cửa sổ **Android Studio**, như minh họa trong hình bên dưới.

Trong hình minh họa: 

1. **Tab Logcat** dùng để mở và đóng **Logcat pane**, hiển thị thông tin về ứng dụng khi nó đang chạy.
   * Nếu bạn thêm các câu lệnh **Log** vào ứng dụng, thông báo **Log** sẽ xuất hiện tại đây.
2. **Menu Log level** được đặt ở chế độ **Verbose** (**mặc định**), hiển thị tất cả các thông báo **Log**.
   * Các tùy chọn 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 nguồn của ứng dụng sẽ hiển thị thông báo trong **Logcat pane**. Ví dụ:

Log.d("MainActivity", "Hello World");

**Các thành phần của thông báo:**

* **Log**: Lớp **Log** dùng để gửi thông báo log đến **Logcat pane**.
* **d**: Cấp độ **Debug** của **Log**, dùng để lọc thông báo trong **Logcat pane**.
  + Các cấp độ log khác:
    - **e**: **Error** (Lỗi)
    - **w**: **Warn** (Cảnh báo)
    - **i**: **Info** (Thông tin)
* **"MainActivity"**: Đối số đầu tiên là **tag**, giúp lọc thông báo trong **Logcat pane**.
  + Thông thường, đây là tên của **Activity** chứa thông báo.
  + Tuy nhiên, bạn có thể đặt bất kỳ giá trị nào có ích cho việc gỡ lỗi.
  + Theo quy ước, **log tag** thường được định nghĩa là **hằng số** trong **Activity**.

private static final String LOG\_TAG = MainActivity.class.getSimpleName();

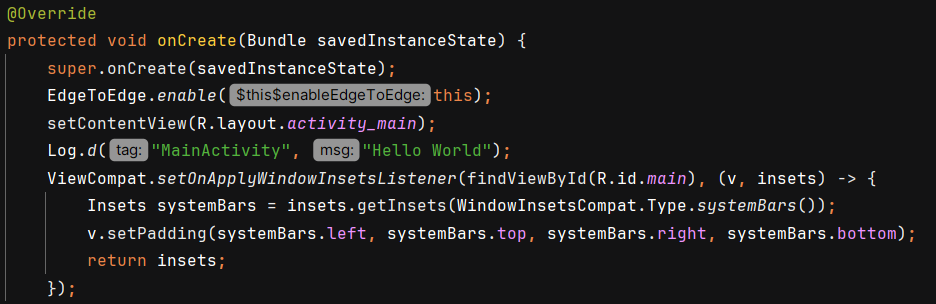
* **"Hello world"**: Đối số thứ hai là nội dung của thông báo.

### ****Các bước thực hiện:****

1. Mở ứng dụng **Hello World** trong **Android Studio** và mở tệp **MainActivity**.
2. Để tự động thêm các **import** cần thiết vào dự án (chẳng hạn như **android.util.Log** để sử dụng **Log**), thực hiện:
   * **Windows**: Chọn **File > Settings**
   * **macOS**: Chọn **Android Studio > Preferences**
3. Chọn **Editor > General > Auto Import**.
   * **Tích chọn tất cả các hộp kiểm** 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:



1. Nếu **Logcat pane** chưa mở, nhấp vào tab **Logcat** ở cuối cửa sổ **Android Studio** để mở.
2. Kiểm tra xem **tên ứng dụng** và **tên gói (package name)** có chính xác không.
3. Thay đổi **Log level** trong **Logcat pane** thành **Debug** (hoặc giữ nguyên **Verbose** nếu có ít thông báo log).
4. Chạy ứng dụng của bạn.

Sau khi chạy, bạn sẽ thấy thông báo xuất hiện trong **Logcat pane**.

2025-02-25 00:00:04.497 15139-15139 MainActivity com.example.helloworld D 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 môi trường và làm quen 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 dòng chữ "Hello World" thành "Happy Birthday to " kèm theo tên của một người vừa có sinh nhật gần đây.
3. (**Tùy chọn**) Chụp ảnh màn hình ứng dụng hoàn thành và gửi email cho ai đó mà bạn đã quên chúc mừng sinh nhật.
4. Một cách sử dụng phổ biến của lớp Log là ghi lại các ngoại lệ (**exceptions**) trong chương trình Java khi chúng xảy ra. Có một số phương thức hữu ích như Log.e() mà bạn có thể dùng cho mục đích này. Hãy khám phá các phương thức cho phép bạn bao gồm một ngoại lệ trong một thông báo Log. Sau đó, viết mã trong ứng dụng của bạn để tạo ra và ghi lại một ngoại lệ.

### ****Tóm tắt****

* Để cài đặt Android Studio, truy cập [Android Studio](https://developer.android.com/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 cấu trúc Android của ứng dụng trong **Project pane**, nhấp vào tab **Project** ở cột dọc và chọn **Android** trong menu thả xuống ở trên cùng.
* Chỉnh sửa tệp **build.gradle(Module:app)** khi bạn cần thêm thư viện mới hoặc thay đổi phiên bản thư viện trong dự án.
* Tất cả mã nguồn và tài nguyên của ứng dụng đều nằm trong thư mục **app** và **res**. Thư mục **java** chứa các activity, test, và các thành phần khác trong mã nguồn Java. Thư mục **res** chứa các tài nguyên như layout, chuỗi ký tự (**strings**) và hình ảnh (**images**).
* Chỉnh sửa tệp **AndroidManifest.xml** để thêm các thành phần (**components**) và quyền truy cập (**permissions**) cho 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**) để chạy ứng dụng.
* Thêm các câu lệnh **Log** vào ứng dụng để hiển thị thông báo trong **Logcat pane**, giúp bạn gỡ lỗi (**debugging**) dễ dàng hơn.
* Để chạy ứng dụng trên thiết bị Android thực tế bằng Android Studio, bật **USB Debugging** trên thiết bị.
  + Mở **Settings > About phone** và nhấn vào **Build number** bảy lần.
  + Quay lại màn hình trước (**Settings**) và chọn **Developer options**.
  + Bật **USB Debugging**.

### ****Các khái niệm liên quan****

Tài liệu về các khái niệm liên quan có trong:

* **1.0: Giới thiệu về Android**
* **1.1: Ứng dụng Android đầu tiên của bạn**

### ****Tìm hiểu thêm****

#### **Tài liệu về Android Studio:**

* [Trang tải xuống Android Studio](https://developer.android.com/studio)
* [Ghi chú phát hành Android Studio](https://developer.android.com/studio/releases)
* [Giới thiệu về Android Studio](https://developer.android.com/studio/intro)
* [Công cụ dòng lệnh Logcat](https://developer.android.com/studio/command-line/logcat)
* [Trình quản lý thiết bị ảo Android (AVD Manager)](https://developer.android.com/studio/run/managing-avds)
* [Tổng quan về Android Manifest](https://developer.android.com/guide/topics/manifest/manifest-intro)
* [Cấu hình Gradle](https://developer.android.com/studio/build)
* [Lớp Log trong Android](https://developer.android.com/reference/android/util/Log)
* [Tạo và quản lý thiết bị ảo](https://developer.android.com/studio/run/managing-avds)

#### **Khác:**

* Làm thế nào để cài đặt Java?
* [Cài đặt phần mềm JDK và thiết lập biến JAVA\_HOME](https://www.oracle.com/java/technologies/javase-jdk11-downloads.html)
* [Trang chủ Gradle](https://gradle.org/)
* Cú pháp Apache Groovy
* [Trang Wikipedia về Gradle](https://en.wikipedia.org/wiki/Gradle)

### ****Bài tập về nhà****

#### **Xây dựng và chạy một ứng dụng**

1. Tạo một dự án Android mới từ **Empty Template**.
2. Thêm các câu lệnh Log cho nhiều cấp độ nhật ký khác nhau (**log levels**) trong phương thức onCreate() của activity chính.
3. Tạo một trình giả lập (**emulator**) cho một thiết bị, chọn phiên bản Android tùy ý, và chạy ứng dụng trên đó.
4. Sử dụng bộ lọc trong **Logcat** để tìm các câu lệnh Log của bạn và điều chỉnh mức hiển thị chỉ để hiển thị **debug** hoặc **error**.

### ****Trả lời các câu hỏi****

#### **Câu hỏi 1**

Tên của tệp layout cho activity chính là gì?

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

#### **Câu hỏi 2**

Tên của tài nguyên chuỗi xác định tên của ứng dụng là gì?

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

#### **Câu hỏi 3**

Công cụ nào được sử dụng để tạo trình giả lập mới?

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

#### **Câu hỏi 4**

Giả sử ứng dụng của bạn có câu lệnh log sau:

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

Bạn sẽ thấy câu lệnh "MainActivity layout is complete" trong **Logcat pane** nếu menu **Log level** được đặt ở mức nào? (Có thể chọn nhiều câu trả lời)

* Verbose
* Debug
* Info
* Warn
* Error
* Assert

### ****Nộp bài kiểm tra ứng dụng****

Kiểm tra để đảm bảo rằng ứng dụng có các yếu tố sau:

* Một **Activity** hiển thị "Hello World" trên màn hình.
* Các câu lệnh **Log** trong **onCreate()** của **MainActivity**.
* Mức **Log** trong **Logcat** chỉ hiển thị các thông báo **debug** hoặc **error**.

### Giao diện người dùng tương tác đầu tiên

### ****Giới thiệu****

Giao diện người dùng (UI) hiển thị trên màn hình của một thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là **view** — mỗi phần tử trên màn hình đều là một **View**. Lớp **View** đại diện cho khối xây dựng cơ bản của tất cả các thành phần UI và là lớp cơ sở cho các lớp cung cấp các thành phần UI tương tác như nút bấm, hộp kiểm và trường nhập văn bản.

Các lớp con phổ biến của **View** 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 UI được chứa trong một lớp mở rộng từ **Activity**. Một **Activity** thường được liên kết với một bố cục UI được định nghĩa trong tệp XML (**eXtended Markup Language**).

Tệp XML này thường được đặt tên theo **Activity** của nó và xác định cách bố trí các phần tử **View** trên màn hình.

Ví dụ, mã **MainActivity** trong ứng dụng **Hello World** hiển thị một bố cục được định nghĩa trong tệp **activity\_main.xml**, trong đó bao gồm một **TextView** có nội dung "Hello World".

Trong các ứng dụng phức tạp hơn, một **Activity** có thể thực hiện 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 một bài học khác.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên của mình—một ứng dụng cho phép tương tác của người dùng. Bạn sẽ tạo ứng dụng bằng mẫu **Empty Activity**. Bạn cũng sẽ học cách sử dụng trình chỉnh sửa bố cục (**layout editor**) để thiết kế bố cục và chỉnh sửa bố cục bằng XML. Bạn cần phát triển các kỹ năng này để hoàn thành các bài thực hành khác trong khóa học này.

### ****Những gì bạn cần 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 có hành vi tương tác.
* Cách sử dụng trình chỉnh sửa bố cục để thiết kế giao diện.
* Cách chỉnh sửa bố cục bằng XML.
* Nhiều thuật ngữ mới. Hãy tham khảo **bảng thuật ngữ và khái niệm** để có các định nghĩa dễ hiểu.

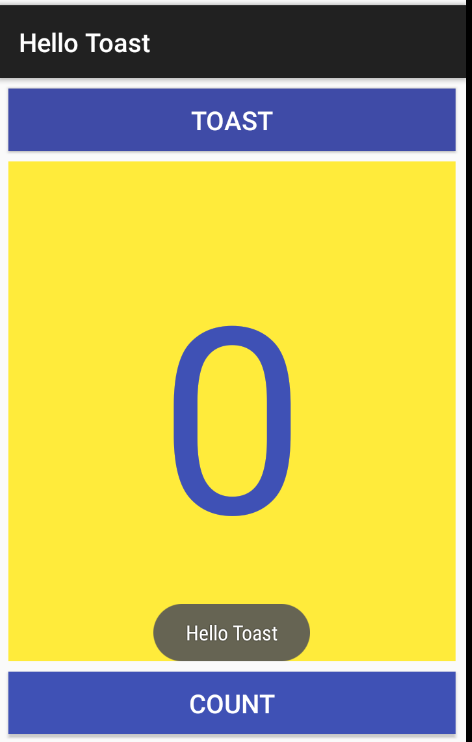
### ****Những gì bạn sẽ làm****

* Tạo một ứng dụng và thêm hai **Button** cùng một **TextView** vào bố cục.
* Điều chỉnh từng phần tử trong **ConstraintLayout** để ràng buộc chúng với lề và các phần tử khác.
* Thay đổi thuộc tính của các phần tử UI.
* Chỉnh sửa bố cục của ứng dụng trong XML.
* Trích xuất chuỗi văn bản cố định thành tài nguyên chuỗi (**string resources**).
* Triển khai phương thức xử lý sự kiện nhấp chuột (**click-handler methods**) để hiển thị thông báo trên màn hình khi người dùng chạm vào từng **Button**.

### ****Tổng quan về ứng dụng****

Ứng dụng **HelloToast** bao gồm hai phần tử **Button** và một **TextView**. Khi người dùng chạm vào **Button** thứ nhất, một thông báo ngắn (**Toast**) sẽ xuất hiện trên màn hình. Khi chạm vào **Button** thứ hai, bộ đếm số lần nhấp (**click counter**) được hiển thị trong **TextView** sẽ tăng lên, bắt đầu từ số 0.

Đây là giao diện cuối cùng của ứng dụng:



### Bài 1: Tạo và khám phá một dự án mới

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

1.1 Tạo dự án Android Studio

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

| **Thuộc tính** | **Giá trị** |
| --- | --- |
| **Tên ứng dụng** | Hello Toast |
| **Tên công ty** | com.example.android (hoặc miền của bạn) |
| **Phone and Tablet Minimum SDK** | API 15: Android 4.0.3 IceCreamSandwich |
| **Mẫu (Template)** | Empty Activity |
| **Tùy chọn "Generate Layout file"** | Được chọn |
| **Tùy chọn "Backwards Compatibility"** | Được chọn |

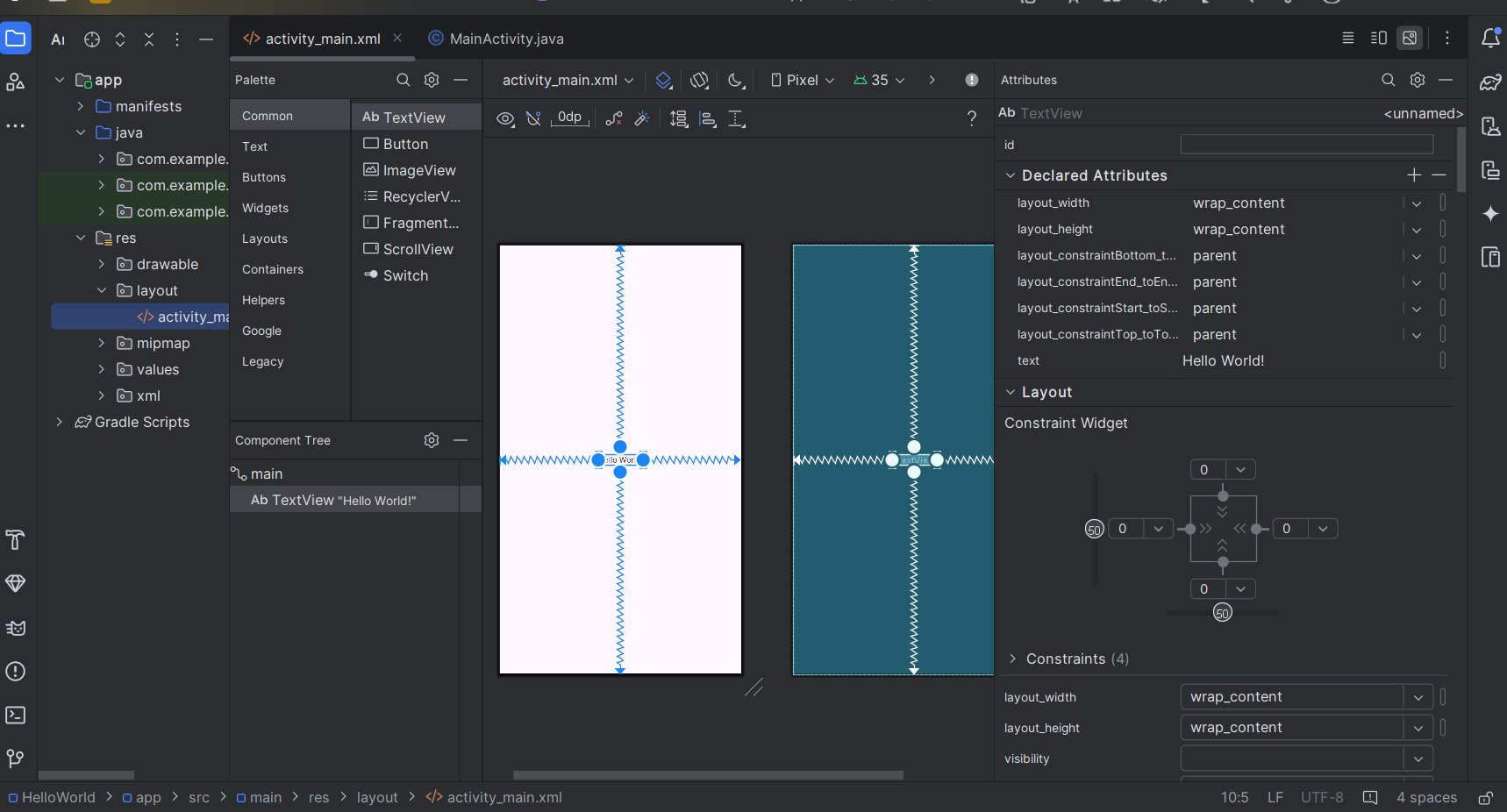
1. Chạy ứng dụng:
   * Chọn **Run > Run app** hoặc nhấn vào **biểu tượng Run** trên thanh công cụ để **biên dịch và chạy ứng dụng** trên trình giả lập hoặc thiết bị thực tế.

### ****Khám phá trình chỉnh sửa bố cục (Layout Editor)****

Android Studio cung cấp **trình chỉnh sửa bố cục (layout editor)** để giúp bạn xây dựng giao diện người dùng (UI) một cách nhanh chóng. Trình chỉnh sửa này cho phép bạn kéo thả các phần tử vào **chế độ thiết kế trực quan (design view)** và **chế độ bản thiết kế (blueprint view)**, căn chỉnh vị trí, thêm **ràng buộc (constraints)** và thiết lập thuộc tính.

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

### ****Khám phá trình chỉnh sửa bố cục****



Hãy làm theo các bước sau:

1. Trong **khung Project > Android**, điều hướng đến thư mục **app > res > layout**, sau đó **nhấp đúp vào tệp** activity\_main.xml để mở nó (nếu chưa mở).
2. Chuyển sang **tab Design** nếu nó chưa được chọn.
   * **Tab Design**: Dùng để thao tác với các phần tử UI và bố cục.
   * **Tab Text**: Dùng để chỉnh sửa mã XML của bố cục.
3. **Pane Palettes** hiển thị danh sách các phần tử UI mà bạn có thể sử dụng trong giao diện ứng dụng.
4. **Pane Component Tree** hiển thị **cây phân cấp** của các phần tử UI.
   * Các phần tử UI được sắp xếp theo thứ bậc **cha – con**, trong đó phần tử con sẽ kế thừa thuộc tính từ phần tử cha.
   * Ví dụ: Trong hình minh họa, **TextView** là phần tử con của **ConstraintLayout**.
5. **Pane thiết kế và bản thiết kế** hiển thị các phần tử UI trong bố cục.
   * Trong hình minh họa, bố cục chỉ có một phần tử duy nhất là **TextView** hiển thị nội dung "Hello World".
6. **Tab Attributes** hiển thị **pane thuộc tính**, nơi bạn có thể thiết lập các thuộc tính cho một phần tử UI.

**Mẹo**:

* Xem tài liệu [**Xây dựng giao diện UI với Layout Editor**](https://developer.android.com/studio/write/layout-editor?hl=vi) để biết thêm chi tiết.
* Tham khảo tài liệu [**Android Studio**](https://developer.android.com/studio?hl=vi) để tìm hiểu đầy đủ về Android Studio.

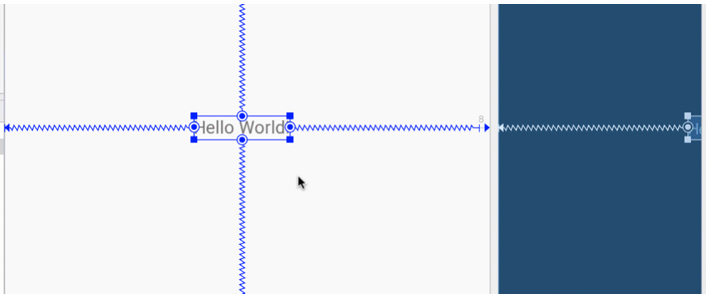
### ****Bài 2: Thêm các phần tử giao diện vào trình chỉnh sửa bố cục****

Trong bài này, bạn sẽ tạo giao diện người dùng (UI) cho ứng dụng **HelloToast** trong trình chỉnh sửa bố cục bằng cách sử dụng các tính năng của **ConstraintLayout**.  
Bạn có thể tạo ràng buộc (constraints) **thủ công** (hướng dẫn sau) hoặc **tự động** bằng công cụ **Autoconnect**.

### ****2.1 Kiểm tra ràng buộc của phần tử****

Hãy làm theo các bước sau:

1. **Mở tệp activity\_main.xml** trong khung **Project > Android**, nếu nó chưa mở.
   * Nếu **tab Design** chưa được chọn, hãy nhấp vào nó.
   * Nếu không thấy bản thiết kế (**blueprint**), nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**.
2. **Kiểm tra công cụ Autoconnect **
   * Công cụ **Autoconnect** nằm trên thanh công cụ và **được bật theo mặc định**.
   * Hãy kiểm tra và đảm bảo nó **không bị vô hiệu hóa**.
3. **Phóng to giao diện thiết kế**
   * Nhấp vào **nút zoom in **để phóng to khu vực thiết kế và bản thiết kế.
4. **Chọn phần tử TextView**
   * Trong **Component Tree**, chọn **TextView** (có nội dung "Hello World").
   * Khi chọn, **TextView** sẽ được làm nổi bật trong **cửa sổ thiết kế** và **các ràng buộc (constraints)** của nó sẽ hiển thị.
5. **Xóa và thêm lại ràng buộc ngang**
   * Nhấp vào **chấm tròn** (**circular handle**) bên phải **TextView** để **xóa ràng buộc ngang**, khiến **TextView** nhảy sang trái.
   * Để thêm lại ràng buộc này, nhấp vào **cùng chấm tròn**, kéo một đường đến cạnh phải của bố cục.



### ****Các nút điều khiển trong blueprint hoặc design pane:****

**Constraint handle** (tay nắm ràng buộc)

* Để **tạo một ràng buộc**, nhấp vào **chấm tròn** ở cạnh của phần tử.
* Kéo tay nắm này đến một **tay nắm khác** hoặc **đường biên của phần tử cha**.
* **Đường zic-zac** hiển thị ràng buộc vừa tạo.



**Resizing handle** (tay nắm thay đổi kích thước)

* Để **thay đổi kích thước** phần tử, kéo **tay nắm vuông** ở góc.
* Khi kéo, tay nắm sẽ chuyển thành **góc xiên**.

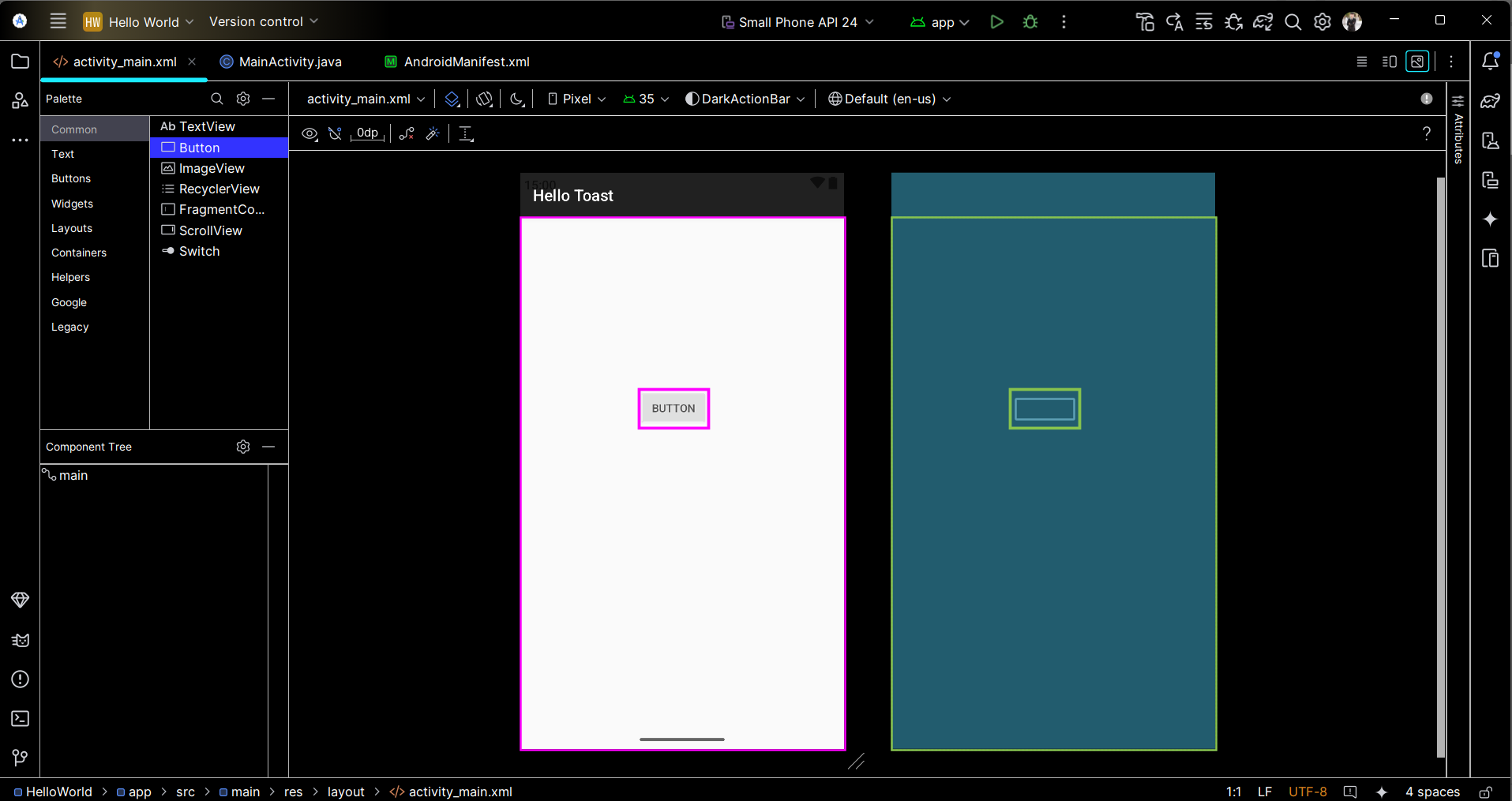


### ****2.2 Thêm một nút Button vào bố cục****

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

### ****Các bước thêm một Button:****

1. **Xóa mọi phần tử cũ**
   * Vì không cần **TextView**, hãy chọn nó và nhấn **phím Delete** hoặc chọn **Edit > Delete**.
   * Bây giờ, bố cục của bạn hoàn toàn trống.
2. **Thêm một Button vào bố cục**
   * Trong **khung Palette**, kéo một **Button** vào **bất kỳ vị trí nào** trong bố cục.
   * Nếu bạn thả **Button** vào **khu vực giữa phía trên**, có thể **các ràng buộc sẽ tự động xuất hiện**.
   * Nếu không, bạn có thể **kéo tay nắm ràng buộc** để kết nối **Button** với **cạnh trên, cạnh trái và cạnh phải của bố cục**, như minh họa trong hình động bên dưới.



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

Hãy làm theo các bước sau để thêm một **nút Button thứ hai** vào giao diện:

1. **Thêm Button mới**
   * Trong **khung Palette**, kéo một **Button** khác vào **giữa bố cục** như minh họa.
   * **Autoconnect** có thể tự động tạo **ràng buộc ngang** cho bạn.
   * Nếu không, bạn có thể **kéo tay nắm ràng buộc** để tạo ràng buộc ngang **thủ công**.
2. **Thêm ràng buộc dọc**
   * Kéo một **ràng buộc dọc** từ **Button thứ hai** đến **cạnh dưới của bố cục** để cố định vị trí.



### ****Xóa ràng buộc của phần tử UI****

**Xóa tất cả ràng buộc của một phần tử**

* Chọn phần tử và di chuyển chuột qua nó.
* **Nút "Clear Constraints"** sẽ xuất hiện. Nhấp vào để **xóa tất cả ràng buộc** của phần tử đó.

**Xóa một ràng buộc cụ thể**

* Nhấp vào **tay nắm ràng buộc** đã đặt để xóa **chỉ ràng buộc đó**.

**Xóa tất cả ràng buộc trong bố cục**

* Nhấp vào công cụ **"Clear All Constraints"** trên **thanh công cụ**.
* Công cụ này rất hữu ích khi bạn muốn **thiết lập lại toàn bộ ràng buộc** trong giao diện.

**Nhiệm vụ 3: Thay đổi thuộc tính của phần tử giao diện người dùng (UI)**

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

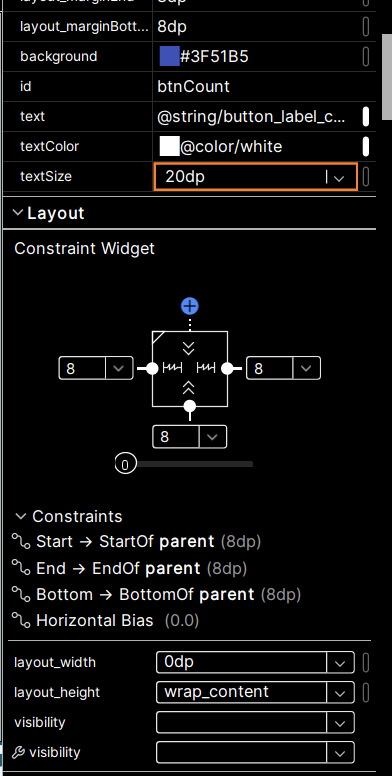
Trong nhiệm vụ này, bạn sẽ nhập giá trị mới và thay đổi giá trị của các thuộc tính quan trọng cho nút Button, các thuộc tính này cũng áp dụng cho hầu hết các loại phần tử View.

**3.1 Thay đổi kích thước nút**

Trình chỉnh sửa bố cục cung cấp các bộ điều khiển thay đổi kích thước ở cả bốn góc của View để bạn có thể nhanh chóng thay đổi kích thước View.

Bạn có thể kéo các bộ điều khiển ở mỗi góc của View để thay đổi kích thước, nhưng làm như vậy sẽ cố định kích thước chiều rộng và chiều cao (hardcoded). Tránh mã hóa kích thước cố định cho hầu hết các phần tử View vì nó không thể thích nghi với nội dung và kích thước màn hình khác nhau.

Thay vào đó, sử dụng ngăn Attributes ở phía bên phải trình chỉnh sửa bố cục để chọn chế độ kích thước không dùng kích thước cố định.Ngăn Attributes bao gồm một bảng điều chỉnh kích thước hình vuông được gọi là View Inspector ở trên cùng.

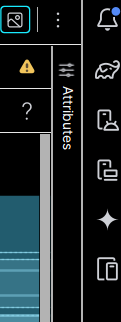


**Ý nghĩa của các ký hiệu trong hình vuông:**

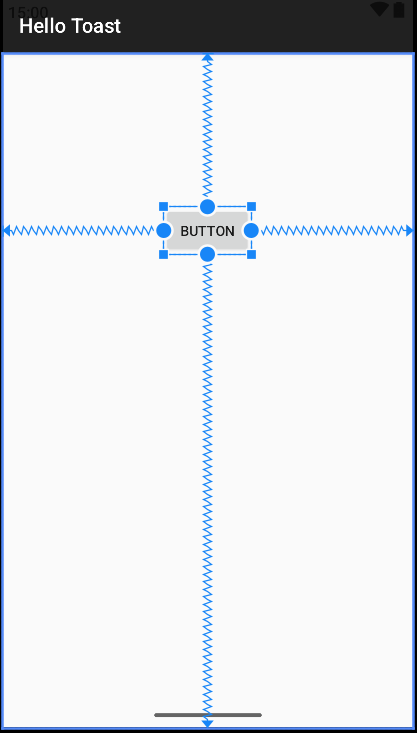
1. **Điều khiển chiều cao (Height control):**
   * Điều khiển này xác định thuộc tính layout\_height và xuất hiện ở hai đoạn trên và dưới của hình vuông.
   * Các góc xiên cho biết điều khiển này được đặt thành wrap\_content, có nghĩa là View sẽ mở rộng theo chiều dọc khi cần để vừa với nội dung.
   * Số "8" biểu thị khoảng cách margin tiêu chuẩn được đặt là 8dp.
2. **Điều khiển chiều rộng (Width control):**
   * Điều khiển này xác định thuộc tính layout\_width và xuất hiện ở hai đoạn bên trái và bên phải của hình vuông.
   * Các góc xiên cho biết điều khiển này được đặt thành wrap\_content, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần để vừa với nội dung, đến giới hạn margin 8dp.
3. **Nút đóng ngăn Attributes (Attributes pane close button):**
   * Nhấp vào nút này để đóng ngăn Attributes.

Làm theo các bước sau:

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

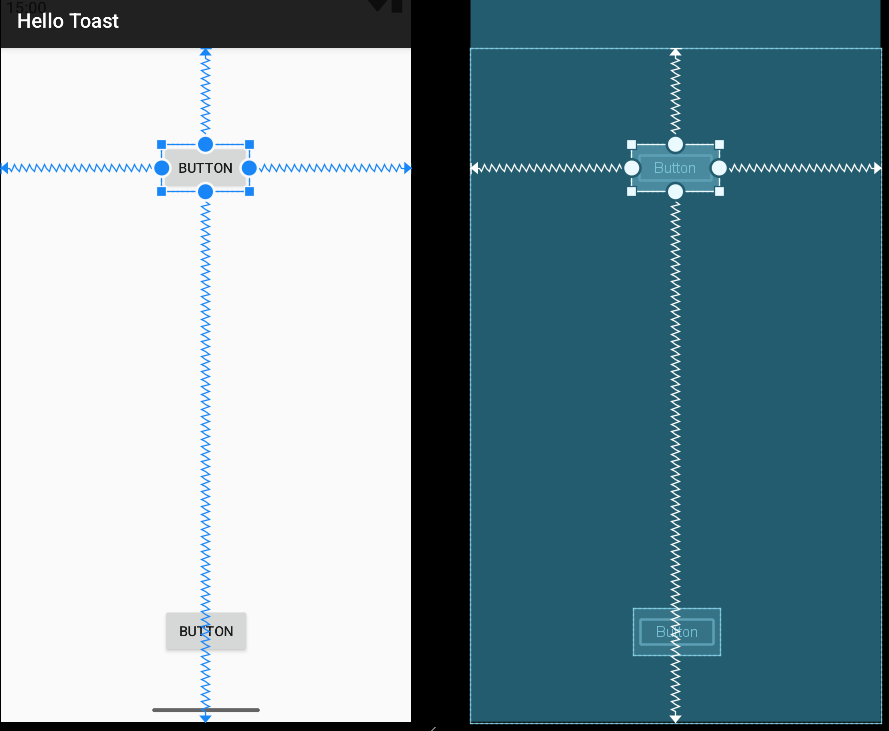


1. Nhấp vào **bộ điều khiển chiều rộng** hai lần — lần nhấp đầu tiên thay đổi thành **Fixed** với các đường thẳng, và lần nhấp thứ hai thay đổi thành **Match Constraints** với các cuộn lò xo, như được minh họa trong hình động bên dưới.



Kết quả của việc thay đổi **bộ điều khiển chiều rộng**, thuộc tính **layout\_width** trong ngăn **Attributes** hiển thị giá trị **match\_constraint**, và phần tử **Button** kéo dài theo chiều ngang để lấp đầy không gian giữa hai bên trái và phải của bố cục.

1. Chọn **Button** thứ hai và thực hiện các thay đổi tương tự đối với **layout\_width** như ở bước trước, như được minh họa trong hình dưới đây.



Như đã trình bày trong các bước trước, các thuộc tính **layout\_width** và **layout\_height** trong ngăn **Attributes** thay đổi khi bạn thay đổi các bộ điều khiển chiều cao và chiều rộng trong **view inspector**. Các thuộc tính này có thể nhận một trong ba giá trị đối với bố cục **ConstraintLayout**:

* **match\_constraint**: Thiết lập này mở rộng phần tử **View** để lấp đầy không gian của phần tử cha theo chiều rộng hoặc chiều cao—đến mức biên, nếu được đặt. Trong trường hợp này, phần tử cha là **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.
* **wrap\_content**: Thiết lập này thu nhỏ kích thước của phần tử **View** sao cho chỉ vừa đủ để bao bọc nội dung của nó. Nếu không có nội dung, phần tử **View** sẽ trở nên không hiển thị.
* Để chỉ định một kích thước cố định phù hợp với kích thước màn hình của thiết bị, hãy sử dụng một số cố định của đơn vị điểm ảnh độc lập với mật độ (**dp units**). Ví dụ, **16dp** nghĩa là 16 điểm ảnh độc lập với mật độ.

**Mẹo:** Nếu bạn thay đổi thuộc tính **layout\_width** bằng cách sử dụng menu bật lên của nó, thuộc tính **layout\_width** sẽ được đặt thành **zero** (0) vì không có kích thước nào được đặt cụ thể. Cài đặt này giống với **match\_constraint**—phần tử **View** có thể mở rộng hết mức có thể để đáp ứng các ràng buộc và cài đặt biên.

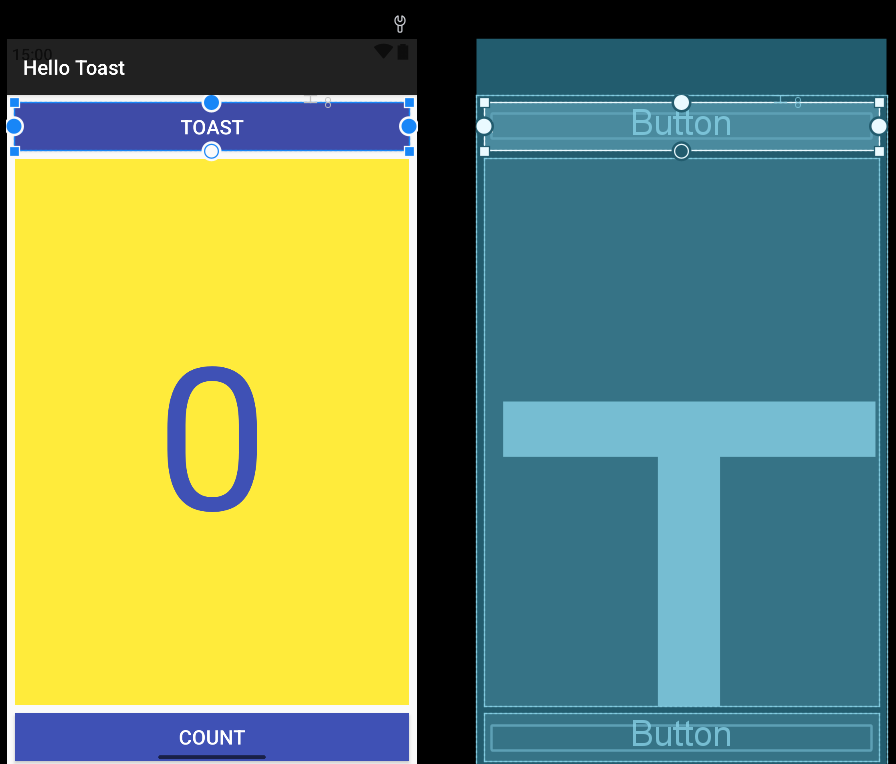
**3.2 Thay đổi các thuộc tính của Button**

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

Ngăn Attributes cung cấp quyền truy cập vào tất cả các thuộc tính 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 minh họa sau đây hướng dẫn 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 dùng để xác định phần tử trong bố cục.
2. Đặt thuộc tính background thành @color/colorPrimary. (Khi bạn nhập @c, các lựa chọn sẽ xuất hiện để dễ dàng chọn lựa.)
3. Đặt thuộc tính textColor thành @android:color/white.
4. Chỉnh sửa thuộc tính text thành Toast.



1. Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng **button\_count** làm ID, đặt **Count** cho thuộc tính text, và sử dụng cùng các màu cho background và text như các bước trước.

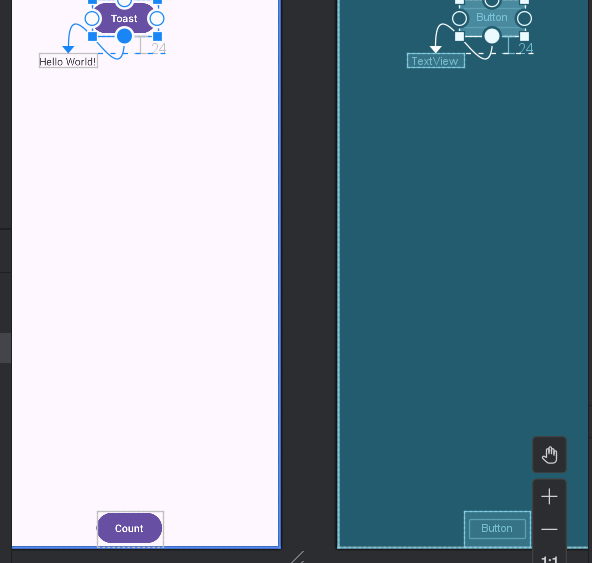
**Lưu ý:**

Màu colorPrimary là màu chủ đạo của theme, một trong những màu cơ sở được định nghĩa sẵn trong tệp tài nguyên colors.xml. Nó được sử dụng cho thanh ứng dụng (app bar). Việc sử dụng các màu cơ sở cho các phần tử UI khác sẽ tạo ra giao diện người dùng (UI) thống nhất. Bạn sẽ tìm hiểu thêm về theme ứng dụng và Material Design trong một bài học khác.

**Nhiệm vụ 4: Thêm một TextView và thiết lập các thuộc tính**

Một trong những lợi ích của ConstraintLayout là khả năng căn chỉnh hoặc ràng buộc các phần tử tương đối với các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một TextView vào giữa bố cục và ràng buộc nó theo chiều ngang với các lề và theo chiều dọc giữa hai nút Button. Sau đó, bạn sẽ thay đổi các thuộc tính của TextView trong ngăn Attributes.

* 1. **Thêm một TextView và thiết lập ràng buộc**
  2. Như minh họa trong hình động dưới đây, kéo một TextView từ ngăn Palette vào phần trên của bố cục.Kéo một ràng buộc từ phía trên của TextView đến chốt ở phía dưới của nút Toast. Việc này sẽ ràng buộc TextView nằm ngay bên dưới nút Toast.
  3. Như minh họa trong hình động dưới đây, kéo một ràng buộc từ phía dưới của **TextView** đến chốt ở phía trên của nút **Count**. Sau đó, kéo các ràng buộc từ hai bên của **TextView** đến hai cạnh của bố cục. Việc này ràng buộc **TextView** nằm giữa bố cục, ở giữa hai nút **Button**



**4.2 Đặt các thuộc tính cho TextView**

Với **TextView** được chọn, hãy mở bảng thuộc tính (**Attributes pane**) nếu nó chưa mở. Đặt các thuộc tính cho **TextView** như hướng dẫn dưới đây. Những thuộc tính mới sẽ được giải thích sau:

1. Đặt **ID** thành show\_count.
2. Đặt **text** thành 0.
3. Đặt **textSize** thành 160sp.
4. Đặt **textStyle** thành **B** (in đậm) và **textAlignment** thành **ALIGNCENTER** (căn giữa đoạn văn bản).
5. Thay đổi chế độ kích thước ngang và dọc (**layout\_width** và **layout\_height**) thành **match\_constraint**.
6. Đặt **textColor** thành @color/colorPrimary.
7. Cuộn xuống bảng thuộc tính và nhấp vào "View all attributes", cuộn đến trang thứ hai của các thuộc tính, tìm thuộc tính "background" và nhập giá trị #FFF00 để chọn một sắc thái màu vàng.
8. Cuộn xuống thuộc tính "gravity", mở rộng mục "gravity" và chọn "center\_ver" (để căn giữa theo chiều dọc).

* **textSize**: Kích thước văn bản của TextView. Trong bài học này, kích thước được đặt là **160sp**. **sp** là viết tắt của **scale-independent pixel** (pixel không phụ thuộc tỷ lệ) và giống như **dp**, là đơn vị điều chỉnh theo mật độ màn hình và tùy chọn kích thước phông chữ của người dùng. Hãy sử dụng đơn vị **dp** khi bạn xác định kích thước phông chữ để đảm bảo kích thước được điều chỉnh cho cả mật độ màn hình và sở thích của người dùng.
* **textStyle** và **textAlignment**: Kiểu văn bản, được đặt là **B (bold)** (in đậm) trong bài học này. Căn chỉnh văn bản, được đặt là **ALIGNCENTER** (căn giữa đoạn văn bản).
* **gravity**: Thuộc tính *gravity* xác định cách một *View* được căn chỉnh trong *View* hoặc *ViewGroup* cha của nó. Trong bước này, bạn căn chỉnh *TextView* theo chiều dọc ở giữa *ConstraintLayout* cha.

Bạn có thể nhận thấy rằng thuộc tính *background* nằm ở trang đầu tiên của bảng *Attributes* (Thuộc tính) đối với một *Button*, nhưng lại nằm ở trang thứ hai đối với một *TextView*. Bảng *Attributes* thay đổi tùy thuộc vào từng loại *View*: các thuộc tính phổ biến nhất của loại *View* sẽ xuất hiện trên trang đầu tiên, và các thuộc tính khác được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của bảng *Attributes*, nhấn vào biểu tượng trên thanh công cụ ở phía trên cùng của bảng.

  
Bố cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, bên cạnh mỗi phần tử giao diện người dùng trong bảng Component Tree lại xuất hiện một dấu chấm than. Di chuyển con trỏ chuột qua các dấu chấm than này để xem thông báo cảnh báo, như hình minh họa bên dưới. Cùng một cảnh báo xuất hiện cho cả ba phần tử: chuỗi được mã hóa cứng nên được thay thế bằng tài nguyên.

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

**5.1 Mở mã XML của bố cục**  
Trong nhiệm vụ này, hãy mở tệp activity\_main.xml nếu nó chưa được mở, và nhấp vào tab Text ở dưới cùng của trình chỉnh sửa bố cục.

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

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".MainActivity" >  
  
 <Button  
 android:id="@+id/btnToast"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="8dp"  
 android:background="#3F4BA7"  
 android:text="@string/button\_label\_toast"  
 android:textColor="@color/white"  
 android:textSize="20dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/textToast"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginBottom="8dp"  
 android:background="#FFEB3B"  
 android:gravity="center"  
 android:text="@string/count\_initial\_value"  
 android:textAllCaps="true"  
 android:textColor="#3F51B5"  
 android:textSize="200dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnCount"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/btnToast"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 <Button  
 android:id="@+id/btnCount"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginBottom="8dp"  
 android:background="#3F51B5"  
 android:text="@string/button\_label\_count"  
 android:textColor="@color/white"  
 android:textSize="20dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent" />  
</androidx.constraintlayout.widget.ConstraintLayout>

  
Thay vì mã hóa cứng các chuỗi, việc sử dụng tài nguyên chuỗi là một thực hành tốt, vì các chuỗi được lưu trữ trong một tệp riêng giúp quản lý dễ dàng hơn, đặc biệt khi bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và địa phương hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho mỗi ngôn ngữ.

Các bước thực hiện:

1. Nhấp một lần vào từ "Toast" (chuỗi cảnh báo đầu tiên được đánh dấu).
2. Nhấn Alt-Enter trên Windows hoặc Option-Enter trên macOS và chọn Extract string resource từ menu bật lên.
3. Nhập "button\_label\_toast" cho tên tài nguyên.
4. Nhấp OK. Một tài nguyên chuỗi sẽ được tạo trong tệp values/res/strings.xml, và chuỗi trong mã của bạn sẽ được thay thế bằng tham chiếu tới tài nguyên:  
   @string/button\_label\_toast
5. Tách các chuỗi còn lại: sử dụng "button\_label\_count" cho "Count" và "count\_initial\_value" cho "0".
6. Trong ngăn Project > Android, mở rộng mục values trong thư mục res, sau đó nhấp đúp vào tệp strings.xml để xem các 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>  
 <string name="toast\_message">Hello Toast!</string>  
</resources>

1. Bạn cần một chuỗi khác để sử dụng trong một nhiệm vụ sau hiển thị thông báo. Thêm vào tệp strings.xml một tài nguyên chuỗi mới có tên là toast\_message cho cụm từ "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>

Mẹo: Các tài nguyên chuỗi bao gồm tên ứng dụng, xuất hiện trên thanh ứng dụng ở đầu màn hình nếu bạn khởi tạo dự án bằng Mẫu Trống. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa tài nguyên app\_name.

  
Trong nhiệm vụ này, bạn sẽ thêm một phương thức Java cho mỗi nút trong MainActivity, phương thức này sẽ được gọi khi người dùng nhấn vào nút.

6.1 Thêm thuộc tính onClick và phương thức xử lý cho mỗi nút  
Một click handler là phương thức được gọi khi người dùng nhấn hoặc chạm vào một phần tử UI có thể nhấn được. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick ở ngăn Attributes của tab Design. Bạn cũng có thể chỉ định tên của phương thức xử lý trong trình chỉnh sửa XML bằng cách thêm thuộc tính android:onClick vào nút. Ở đây, bạn sẽ sử dụng phương pháp thứ hai vì bạn chưa tạo các phương thức xử lý, và trình chỉnh sửa XML cung cấp cách tự động tạo các phương thức đó.

1. Với trình chỉnh sửa XML (tab Text) đang mở, tìm nút có android:id được đặt là button\_toast.

<Button  
 android:id="@+id/btnToast"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="8dp"  
 android:background="#3F4BA7"  
 android:text="@string/button\_label\_toast"  
 android:textColor="@color/white"  
 android:textSize="20dp"  
 />

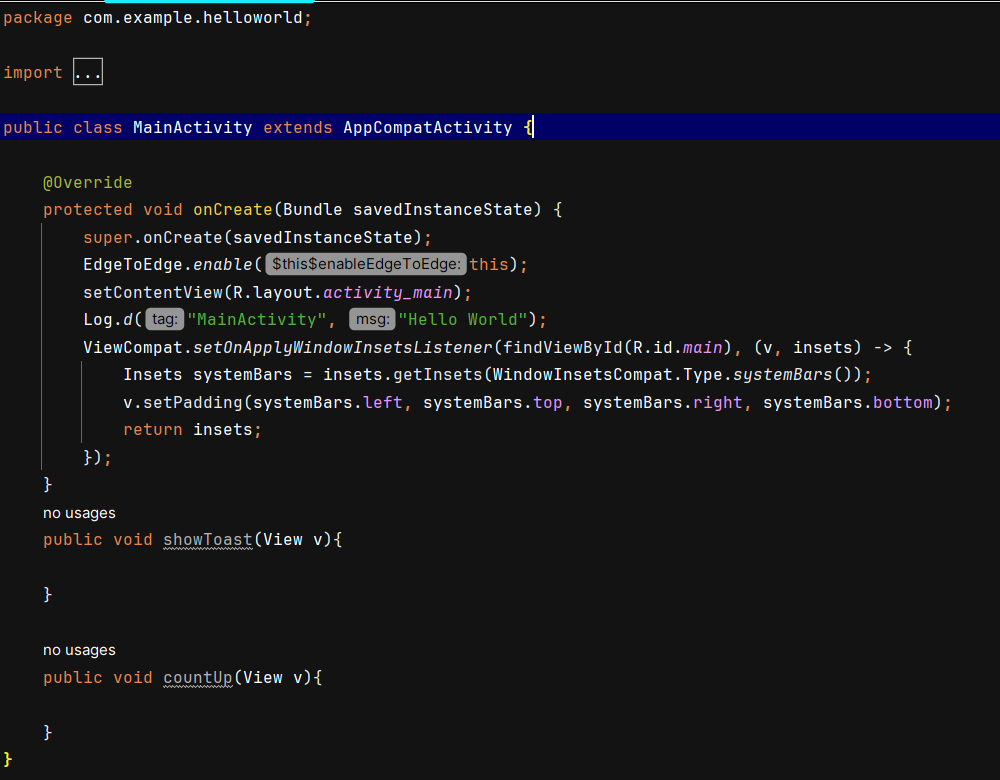
1. Thêm thuộc tính android:onClick vào cuối phần tử button\_toast, sau thuộc tính cuối cùng và trước dấu kết thúc "/>"
2. Nhấp vào biểu tượng bóng đèn đỏ xuất hiện bên cạnh thuộc tính. Chọn Create click handler, chọn MainActivity, và nhấp OK. Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy nhấp vào tên phương thức ("showToast"). Nhấn Alt-Enter (trên Windows/Linux) hoặc Option-Enter (trên Mac), chọn Create 'showToast(view)' in MainActivity, và nhấp OK. Thao tác này sẽ tạo ra một phương thức mẫu (placeholder method stub) cho phương thức showToast() trong MainActivity, như được hiển thị ở cuối các bước này.
3. Lặp lại hai bước cuối với nút button\_count: Thêm thuộc tính android:onClick vào cuối phần tử, và tạo click handler

android:onClick="countUp"/>

Mã XML của các phần tử giao diện người dùng bên trong ConstraintLayout bây giờ trông như sau:

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <Button  
 android:id="@+id/btnToast"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="8dp"  
 android:background="#3F4BA7"  
 android:text="@string/button\_label\_toast"  
 android:textColor="@color/white"  
 android:textSize="20dp"  
 android:onClick="countUp"/>  
 />  
  
 <TextView  
 android:id="@+id/show\_count"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginBottom="8dp"  
 android:background="#FFEB3B"  
 android:gravity="center"  
 android:text="@string/count\_initial\_value"  
 android:textAllCaps="true"  
 android:textColor="#3F51B5"  
 android:textSize="120dp"  
 android:layout\_below="@id/btnToast"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentStart="true"/>  
  
 <Button  
 android:id="@+id/btnCount"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginBottom="8dp"  
 android:background="#3F51B5"  
 android:text="@string/button\_label\_count"  
 android:textColor="@color/white"  
 android:textSize="20dp"  
 android:layout\_below="@id/show\_count"  
 android:layout\_centerHorizontal="true"  
 />  
  
</RelativeLayout>

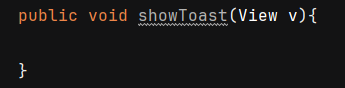
1. Nếu MainActivity.java chưa được mở, mở rộng mục java trong ngăn Project > Android, mở rộng com.example.android.hellotoast, sau đó nhấp đúp vào MainActivity. Trình soạn thảo mã sẽ xuất hiện với mã của MainActivity.



  
Bây giờ, bạn sẽ chỉnh sửa phương thức showToast() – xử lý sự kiện click của nút Toast trong MainActivity – để hiển thị một thông báo. Một Toast cung cấp cách hiển thị thông báo đơn giản trong một cửa sổ popup nhỏ. Toast chỉ chiếm không gian cần thiết để hiển thị thông báo, và Activity hiện tại vẫn hiển thị và có thể tương tác. Toast có thể hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn – hãy thêm một thông báo Toast để hiển thị kết quả của thao tác nhấn nút hoặc thực hiện một hành động.

Thực hiện theo các bước sau để chỉnh sửa xử lý sự kiện nút Toast:

1. Tìm vị trí của phương thức showToast() mới được tạo trong MainActivity.



1. Để tạo một đối tượng Toast, hãy gọi phương thức makeText (factory method) trên lớp Toast.

public void showToast(View v){  
 Toast toast = Toast.makeText()**;**}

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

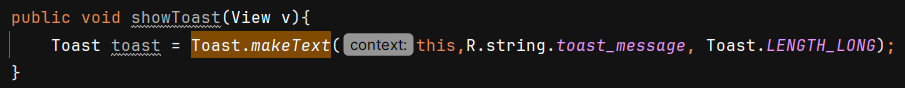
1. Cung cấp context của Activity của ứng dụng. Vì một Toast được hiển thị phía trên giao diện người dùng của Activity, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong context của Activity mà bạn cần, hãy sử dụng từ khóa "this" như một cách rút gọn.

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

1. Cung cấp thông báo để hiển thị, chẳng hạn như một tài nguyên chuỗi (ví dụ: toast\_message mà bạn đã tạo ở bước trước). Tài nguyên chuỗi toast\_message được nhận diện bằng R.string.toast\_message.

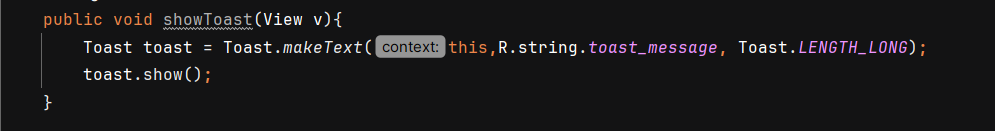
public void showToast(View v){  
 Toast toast = Toast.makeText(this**,**R.string.toast\_message**,**)**;**}

1. Cung cấp thời lượng hiển thị. Ví dụ, Toast.LENGTH\_SHORT hiển thị Toast trong một khoảng thời gian tương đối ngắn.

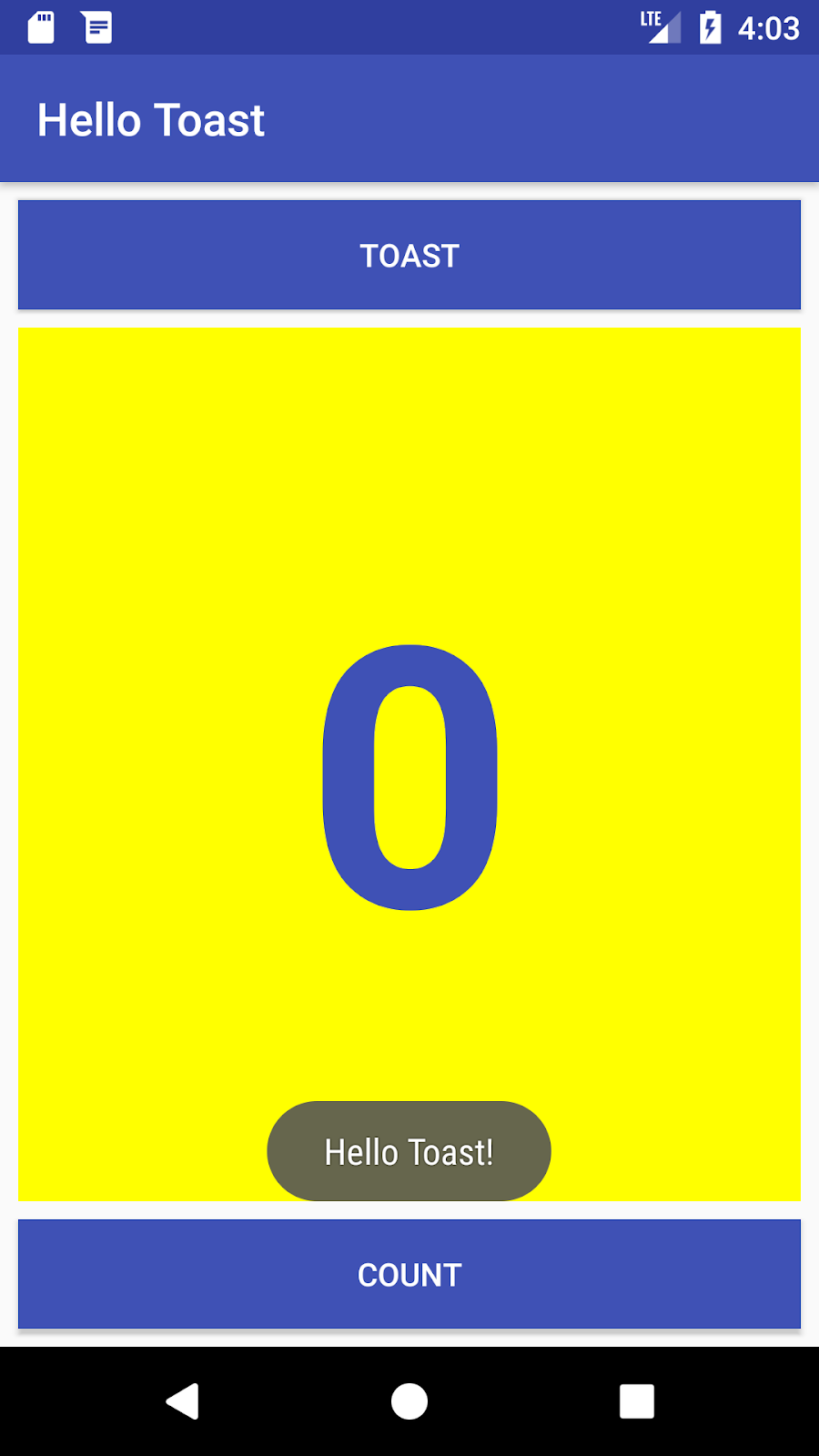


Thời lượng hiển thị của Toast có thể là Toast.LENGTH\_LONG hoặc Toast.LENGTH\_SHORT. Thời gian thực tế là khoảng 3,5 giây cho Toast dài và 2 giây cho Toast ngắn.

1. Hiển thị Toast bằng cách gọi phương thức show(). Dưới đây là toàn bộ phương thức showToast():



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

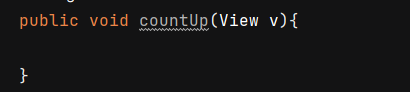


  
Bây giờ, bạn sẽ chỉnh sửa phương thức countUp() – xử lý sự kiện click của nút Count trong MainActivity – sao cho nó hiển thị giá trị đếm hiện tại sau mỗi lần nhấn nút Count. Mỗi lần nhấn nút sẽ tăng giá trị đếm lên một đơn vị.  
Mã xử lý sự kiện cần thực hiện các điều sau:

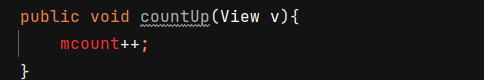
* Theo dõi giá trị đếm khi nó thay đổi.
* Gửi giá trị đếm được cập nhật đến TextView để hiển thị.

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

1. Tìm vị trí của phương thức countUp() mới được tạo trong MainActivity, có dạng:

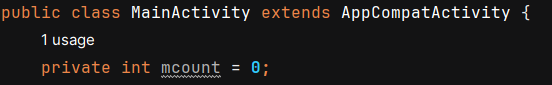


1. Để theo dõi giá trị đếm, bạn cần một biến thành viên riêng (private member variable). Mỗi lần nhấn nút Count sẽ tăng giá trị của biến này. Nhập đoạn mã sau (đoạn mã này sẽ được đánh dấu màu đỏ và xuất hiện biểu tượng bóng đèn đỏ):

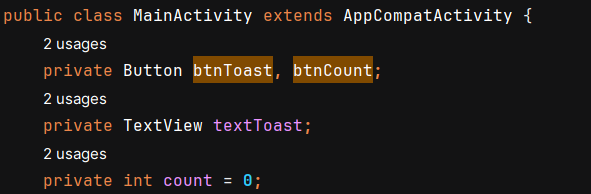


Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy chọn biểu thức mCount++; sau một lúc, biểu tượng bóng đèn sẽ xuất hiện.

1. Nhấp vào biểu tượng bóng đèn đỏ và chọn Create field 'mCount' từ menu bật lên. Thao tác này tạo ra một biến thành viên riêng tại đầu lớp MainActivity, và Android Studio mặc định kiểu của biến này là integer (int):
2. Thay đổi câu lệnh khai báo biến thành viên thành khởi tạo giá trị bằng 0:



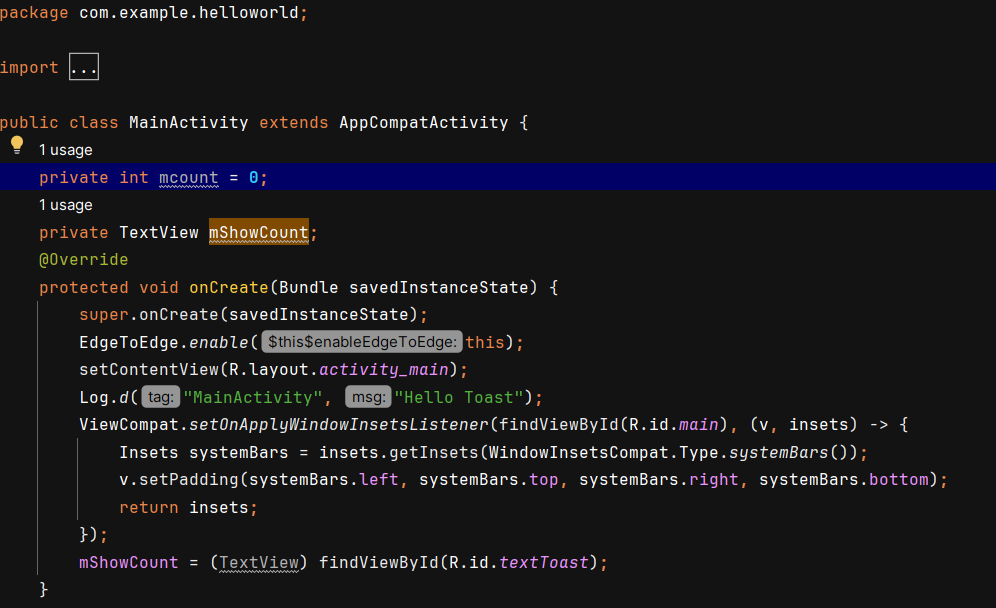
1. Ngoài biến mCount, bạn cũng cần một biến thành viên riêng để lưu tham chiếu đến TextView hiển thị số đếm, gọi biến này là mShowCount:



1. Bây giờ, đã có mShowCount, bạn có thể lấy tham chiếu đến TextView thông qua ID đã đặt trong tệp bố cục. Để lấy tham chiếu này chỉ một lần, hãy thực hiện trong phương thức onCreate(). Như bạn đã học trong một bài học khác, phương thức onCreate() được dùng để inflate bố cục, tức là đặt content view của màn hình bằng tệp XML bố cục. Bạn cũng có thể sử dụng onCreate() để lấy tham chiếu đến các phần tử UI khác trong bố cục, chẳng hạn như TextView. Tìm phương thức onCreate() trong MainActivity:

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState)**;** EdgeToEdge.*enable*(this)**;** setContentView(R.layout.*activity\_main*)**;** Log.*d*("MainActivity"**,** "Hello Toast")**;** ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*)**,** (v**,** insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*())**;** v.setPadding(systemBars.left**,** systemBars.top**,** systemBars.right**,** systemBars.bottom)**;** return insets**;** })**;**}

1. Thêm câu lệnh findViewById vào cuối phương thức onCreate() để gán giá trị cho mShowCount:



Một View, giống như một chuỗi, là một tài nguyên có thể có một ID. Gọi findViewById với ID của một view sẽ trả về đối tượng View. Vì phương thức này trả về một View, bạn cần ép kiểu kết quả về loại view mà bạn mong đợi, trong trường hợp này là (TextView).

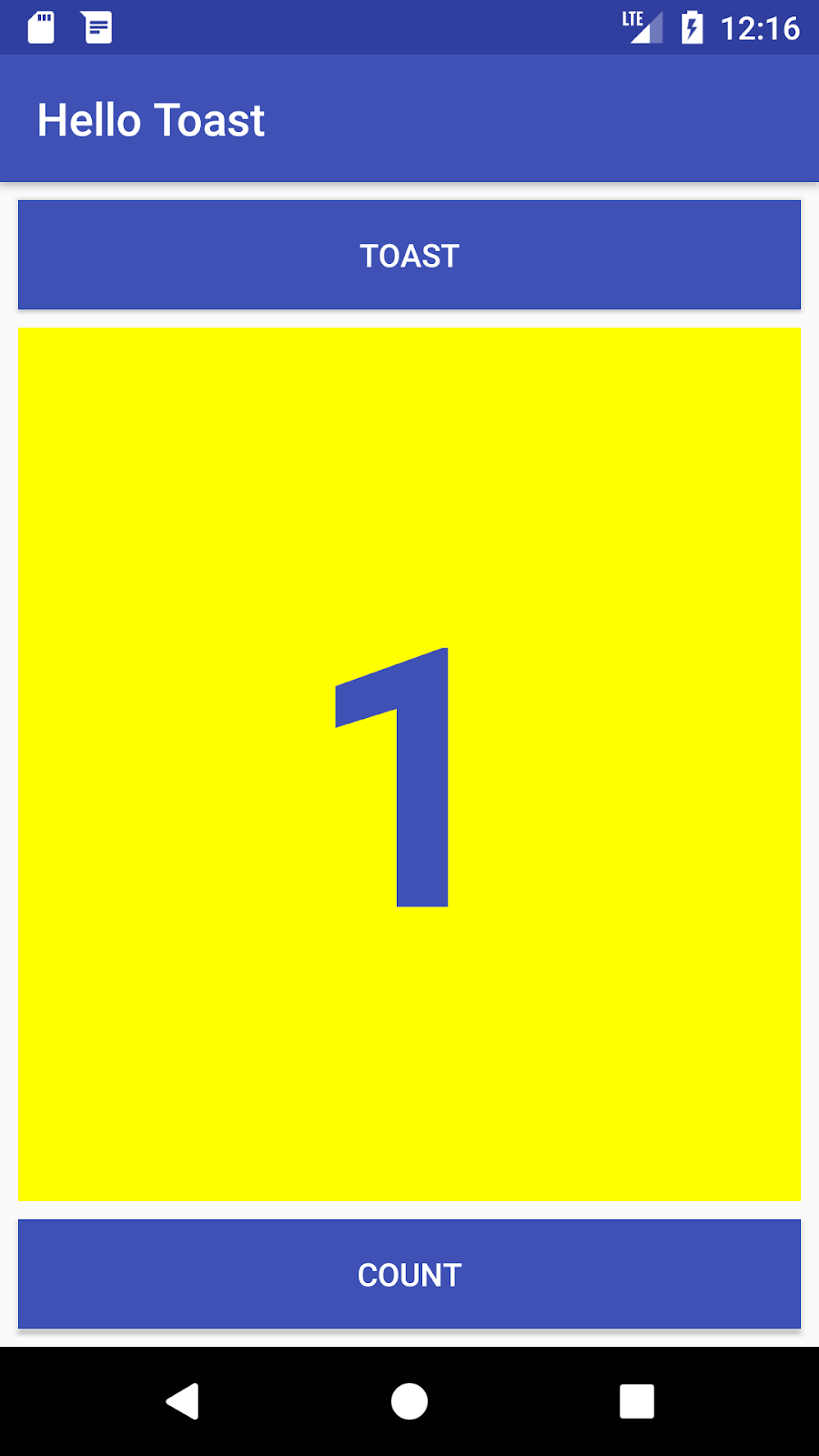
1. Sau khi đã gán tham chiếu cho mShowCount, bạn có thể sử dụng biến này để cập nhật văn bản của TextView với giá trị của biến mCount. Thêm đoạn mã sau vào phương thức countUp():

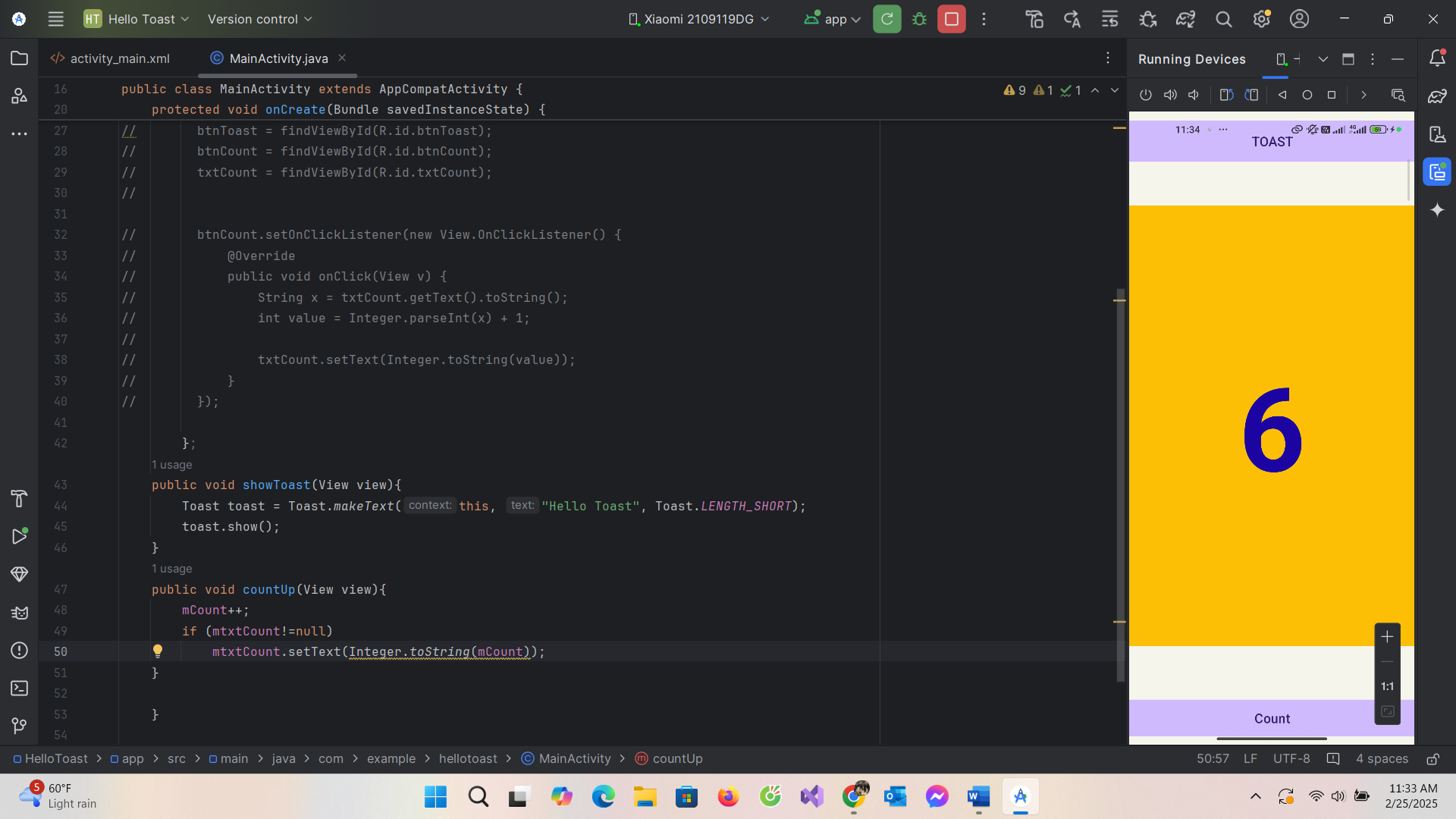
if(mShowCount != null){  
 mShowCount.setText(Integer.*toString*(mcount))**;**}

Toàn bộ phương thức countUp() bây giờ trông như sau:

public void countUp(View v){  
 mcount++**;** if(mShowCount != null){  
 mShowCount.setText(Integer.*toString*(mcount))**;** }  
}

1. Chạy ứng dụng để xác minh rằng giá trị đếm tăng lên mỗi khi bạn nhấn nút Count.





Mẹo: Để tìm hiểu chi tiết về cách sử dụng ConstraintLayout, hãy xem Codelab "Using ConstraintLayout to design your views".

Thách thức lập trình  
Lưu ý: Tất cả các thách thức lập trình đều không bắt buộc 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 chiều dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang chiều ngang, nút **Count** có thể chồng lên **TextView** ở phía dưới như trong hình minh họa bên dưới.

**Thử thách**: Thay đổi bố cục để nó trông đẹp cả khi ở chế độ ngang và dọc:

1. Trên máy tính của bạn, sao chép thư mục dự án **HelloToast** và đổi tên nó thành **HelloToastChallenge**.
2. Mở **HelloToastChallenge** trong Android Studio và chỉnh sửa nó. (Xem Phụ lục: Tiện ích để biết hướng dẫn sao chép và chỉnh sửa một dự án.)
3. Thay đổi bố cục sao cho nút **Toast** và nút **Count** xuất hiện ở phía bên trái, như trong hình minh họa bên dưới. **TextView** xuất hiện bên cạnh chúng, nhưng chỉ rộng đủ để hiển thị nội dung của nó. (Gợi ý: Sử dụng wrap\_content.)
4. Chạy ứng dụng ở cả chế độ ngang và dọc.

Tóm tắt:  
**View, ViewGroup và layouts:**

* Tất cả các phần tử giao diện người dùng (UI) đều là các lớp con của lớp **View**, do đó kế thừa nhiều thuộc tính từ lớp **View**.
* Các phần tử **View** có thể được nhóm lại bên trong một **ViewGroup**, hoạt động như một container. Mối quan hệ này là mối quan hệ cha-con, trong đó **parent** là một **ViewGroup**, còn **child** là một **View** hoặc một **ViewGroup** khác.
* Phương thức **onCreate()** được sử dụng để **inflate layout**, nghĩa là thiết lập nội dung hiển thị của màn hình từ tệp XML layout. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện người dùng (UI) khác trong layout.
* Một **View**, giống như một chuỗi ký tự (string), là một tài nguyên có thể có ID. Phương thức findViewById nhận ID của một View làm tham số và trả về View đó.

**Sử dụng trình chỉnh sửa bố cục:**

* Nhấp vào tab **Design** để thao tác các phần tử và bố cục, hoặc tab **Text** để chỉnh sửa mã XML của bố cục.
* Trong tab **Design**, bảng **Palettes** hiển thị các phần tử giao diện người dùng (UI) có thể sử dụng trong bố cục ứng dụng, và bảng **Component tree** hiển thị cây phân cấp của các phần tử UI.
* Các phần tử UI trong bố cục được hiển thị trong các bảng **design** và **blueprint**. Tab **Attributes** hiển thị bảng **Attributes** để cài đặt các thuộc tính cho phần tử UI.
* Constraint handle: Nhấp vào một constraint handle (vòng tròn ở mỗi cạnh của phần tử), kéo đến một constraint khác hoặc đường viền cha để tạo ràng buộc. Ràng buộc được hiển thị bằng một đường gấp khúc.
* Resizing handle: Kéo tay nắm (hình vuông) để thay đổi kích thước phần tử. Trong khi kéo, tay nắm sẽ chuyển thành góc xiên.
* Autoconnect tool: Khi được bật, công cụ này tự động tạo hai hoặc nhiều ràng buộc cho phần tử UI với bố cục cha, dựa trên vị trí của phần tử.
* Bạn có thể xóa ràng buộc của một phần tử bằng cách chọn phần tử, di chuột qua để hiển thị nút **Clear Constraints**, và nhấp để xóa tất cả các ràng buộc. Để xóa một ràng buộc cụ thể, nhấp vào constraint handle tương ứng.
* Bảng **Attributes** cung cấp truy cập vào tất cả các thuộc tính XML có thể gán cho một phần tử UI. Nó cũng bao gồm một bảng định cỡ hình vuông gọi là **view inspector** ở trên cùng. Các biểu tượng trong hình vuông đại diện cho cài đặt chiều cao và chiều rộng.

Thuộc tính **layout\_width** và **layout\_height** thay đổi khi bạn thay đổi chiều cao và chiều rộng trong bảng điều khiển. Các giá trị có thể là:

1. **match\_constraint:** Phần tử mở rộng để lấp đầy không gian bố cục cha (có tính đến lề nếu được đặt).
2. **wrap\_content:** Phần tử co lại vừa với nội dung của nó. Nếu không có nội dung, phần tử sẽ trở nên vô hình.
3. **dp cố định:** Chỉ định kích thước cố định, phù hợp với kích thước màn hình của thiết bị.

**Trích xuất tài nguyên chuỗi (String Resources):**

Thay vì mã hóa cứng chuỗi, nên sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Thực hiện theo các bước sau:

1. Nhấp vào chuỗi mã hóa cứng để trích xuất, nhấn **Alt-Enter** (hoặc **Option-Enter** trên Mac) và chọn **Extract string resources** từ menu bật lên.
2. Đặt tên cho **Resource name**.
3. Nhấn **OK**. Điều này sẽ tạo một tài nguyên chuỗi trong tệp values/res/string.xml, và chuỗi trong mã sẽ được thay thế bằng tham chiếu tới tài nguyên: @string/button\_label\_toast.

**Xử lý sự kiện nhấp (Handling Clicks):**

* **Click handler** là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI.
* Để chỉ định một click handler cho một phần tử UI như **Button**, nhập tên phương thức trong trường **onClick** của bảng **Attributes** ở tab **Design**, hoặc trong trình chỉnh sửa XML bằng cách thêm thuộc tính android:onClick vào phần tử **Button**.
* Tạo click handler trong Activity chính bằng cách sử dụng tham số **View**. Ví dụ:

public void showToast(View view) {

// Xử lý khi nhấp

}

* Có thể tìm thấy thông tin về tất cả các thuộc tính của **Button** trong tài liệu lớp **Button**, và tất cả các thuộc tính của **TextView** trong tài liệu lớp **TextView**.

Toast cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ popup nhỏ. Nó chỉ chiếm không gian đủ để chứa thông báo. Để tạo một instance của Toast, thực hiện các bước sau:

1. Gọi phương thức **makeText()** từ lớp **Toast**.
2. Cung cấp **context của Activity ứng dụng** và thông báo cần hiển thị (chẳng hạn như một tài nguyên chuỗi).
3. Cung cấp thời lượng hiển thị, ví dụ: **Toast.LENGTH\_SHORT** cho thời gian ngắn. Thời lượng có thể là **Toast.LENGTH\_LONG** hoặc **Toast.LENGTH\_SHORT**.
4. Hiển thị Toast bằng cách gọi phương thức **show()**.

Toast.makeText(context, R.string.message\_text, Toast.LENGTH\_SHORT).show();

### Trình chỉnh sửa bố cục

**Giới thiệu**

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

**ConstraintLayout** là một **ViewGroup**, một loại **View** đặc biệt có thể chứa các đối tượng **View** khác (được gọi là **children** hoặc **child views**). Phần thực hành này giới thiệu nhiều tính năng hơn của **ConstraintLayout** và trình chỉnh sửa bố cục.

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

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

**Những gì bạn cần biết trước:**

Bạn cần có khả năng:

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

**Những gì bạn sẽ học:**

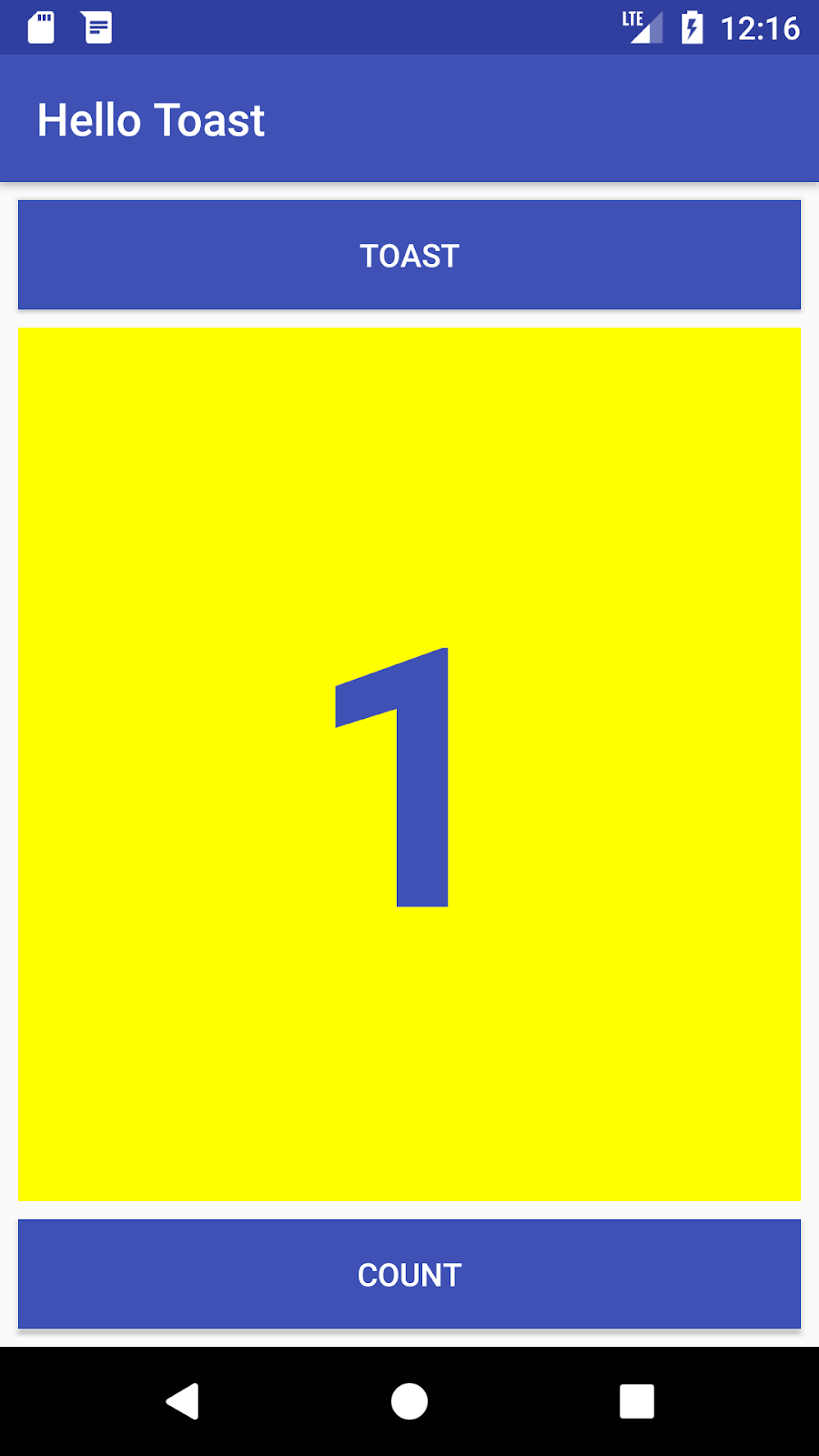
* Cách tạo một biến thể bố cục cho màn hình ngang (landscape).
* Cách tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
* Cách sử dụng ràng buộc đường cơ sở (baseline constraint) để căn chỉnh các phần tử UI với văn bản.
* Cách sử dụng các nút gói (pack) và căn chỉnh (align) để căn chỉnh các phần tử trong bố cục.
* Cách định vị các view trong **LinearLayout**.
* Cách định vị các view trong **RelativeLayout**.

**Những gì bạn sẽ làm:**

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

**Tổng quan về ứng dụng:**

Ứng dụng **Hello Toast** từ 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 màn hình ngang (horizontal orientation), như minh họa trong hình dưới đây.

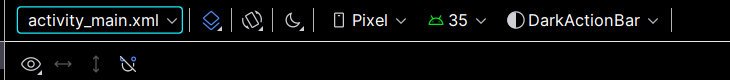
Bạn cũng sẽ học cách sử dụng ràng buộc đường cơ sở (baseline constraints) và một số tính năng căn chỉnh của **ConstraintLayout** bằng cách tạo một biến thể bố cục khác dành cho màn hình máy tính bảng.

Bạn cũng sẽ tìm hiểu về các lớp con khác của **ViewGroup** 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 bạn thay đổi bố cục của ứng dụng **Hello Toast** để phù hợp với cả hai chế độ **dọc** và **ngang**. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo các biến thể bố cục cho các thiết bị di động ở chế độ **dọc** hoặc **ngang**, cũng như cho các thiết bị màn hình lớn như máy tính bảng.

Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trên **hai thanh công cụ trên cùng** trong trình chỉnh sửa bố cục (**Layout Editor**).

Thanh công cụ trên cùng giúp bạn cấu hình cách hiển thị bản xem trước của bố cục trong trình chỉnh sửa bố cục: 

**Các bước chính:**

1. **Chọn bề mặt thiết kế (Design Surface):**Chọn **Design** để hiển thị bản xem trước màu sắc của bố cục, hoặc **Blueprint** để chỉ hiển thị các đường viền của từng phần tử giao diện người dùng (UI).Để xem cả hai chế độ cạnh nhau, chọn **Design + Blueprint**.
2. Thay đổi hướng hiển thị trong trình chỉnh sửa (Orientation in Editor):Chọn **Portrait** (chế độ dọc) hoặc **Landscape** (chế độ ngang) để xem trước bố cục theo chiều dọc hoặc ngang. Điều này rất 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 cho từng hướng, chọn **Create Landscape Variation** hoặc các biến thể khác.
3. **Chọn thiết bị trong trình chỉnh sửa (Device in Editor):**Chọn loại thiết bị (ví dụ: điện thoại/máy tính bảng, 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 (ví dụ: **AppTheme**) để áp dụng cho bản xem trước.
6. **Chọn ngôn ngữ và khu vực (Locale in Editor):**Chọn ngôn ngữ và khu vực cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có trong tài nguyên chuỗi văn bản (**string resources**).Bạn cũng có thể chọn **Preview as Right To Left** để xem bố cục như khi chọn một ngôn ngữ đọc từ phải sang trái (**RTL**).

Thanh công cụ thứ hai giúp bạn cấu hình cách hiển thị các phần tử giao diện người dùng trong **ConstraintLayout**, cũng như phóng to hoặc cuộn bản xem trước.





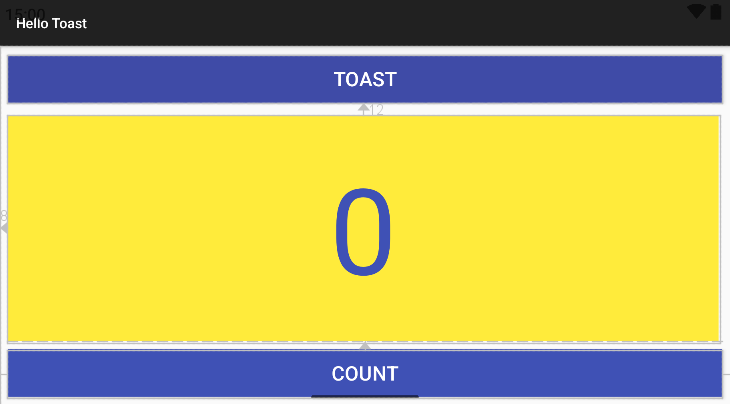
**Trong hình trên:**

1. **Hiển thị (Show):** Chọn **Show Constraints** và **Show Margins** để hiển thị hoặc ẩn các ràng buộc và khoảng cách trong bản xem trước.
2. **Autoconnect:** Bật hoặc tắt tính năng Autoconnect. Khi tính năng này được bật, bạn có thể kéo bất kỳ phần tử nào (như Button) đến bất kỳ phần nào trong bố cục để tự động tạo ràng buộc 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):** Tạo ràng buộc bằng cách suy luận.
5. **Khoảng cách mặc định (Default Margins):** Thiết lập khoảng cách mặc định.
6. **Gói (Pack):** Gom nhóm hoặc mở rộng các phần tử được chọn.
7. **Căn chỉnh (Align):** Căn chỉnh các phần tử được chọn.
8. **Đường dẫn hướng (Guidelines):** Thêm đường dẫn hướng dọc hoặc ngang.
9. **Điều khiển phóng to/thu nhỏ (Zoom/pan controls):** Phóng to hoặc thu nhỏ bản xem trước.

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

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

1. Mở ứng dụng **Hello Toast** từ bài học trước.
   * **Lưu ý:** Nếu bạn đã tải xuống mã hoàn chỉnh cho HelloToast, hãy 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.
   * Chuyển từ chế độ hiển thị **Project > Android** sang **Project > Project Files** trong bảng **Project**. Sau đó, mở rộng đường dẫn **app > src/main > res**, chọn cả hai thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**.
   * Sau khi hoàn tất, chuyển lại bảng **Project** về **Project > Android**.
2. Mở tệp **activity\_main.xml** trong thư mục bố cục (**layout**). Nhấp vào tab **Design** nếu nó chưa được chọn.
3. Nhấp vào nút **Orientation in Editor** trên thanh công cụ ở phía trên để thay đổi hướng hiển thị.
4. Trong menu thả xuống, chọn **Switch to Landscape**. Bố cục sẽ xuất hiện ở chế độ ngang, như minh họa bên dưới. Để quay lại chế độ dọc, chọn **Switch to Portrait**.



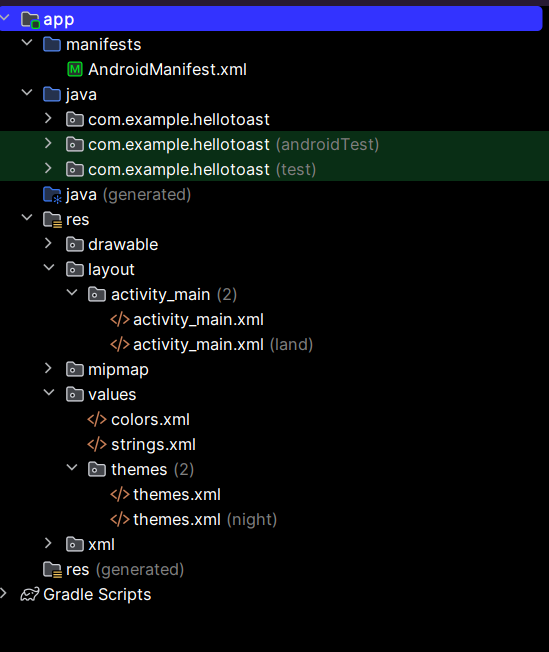
### 1.2 Tạo biến thể bố cục cho chế độ ngang

Sự khác biệt về mặt hình ảnh giữa chế độ dọc và ngang trong bố cục này là chữ số **0** trong phần tử **show\_count** (**TextView**) nằm quá thấp đối với chế độ ngang, quá gần với nút **Count** và kích thước cố định của **TextView** (160sp) có thể khiến bố cục trong chế độ ngang trông không cân đối, với chữ số **0** quá lớn hoặc không được căn giữa. Để giải quyết vấn đề này trong chế độ ngang mà vẫn giữ nguyên bố cục cho chế độ dọc, bạn có thể tạo một biến thể bố cục cho ứng dụng **Hello Toast**. Thực hiện theo các bước sau:

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

Khi đó, một cửa sổ chỉnh sửa mới sẽ mở ra với tab **land/activity\_main.xml**, hiển thị bố cục dành riêng cho chế độ ngang. Bạn có thể thay đổi bố cục này mà không ảnh hưởng đến bố cục gốc cho chế độ dọc.

1. Trong bảng **Project > Android**, mở thư mục **res > layout**, bạn sẽ thấy Android Studio đã tự động tạo biến thể mới có tên **activity\_main.xml (land)**.



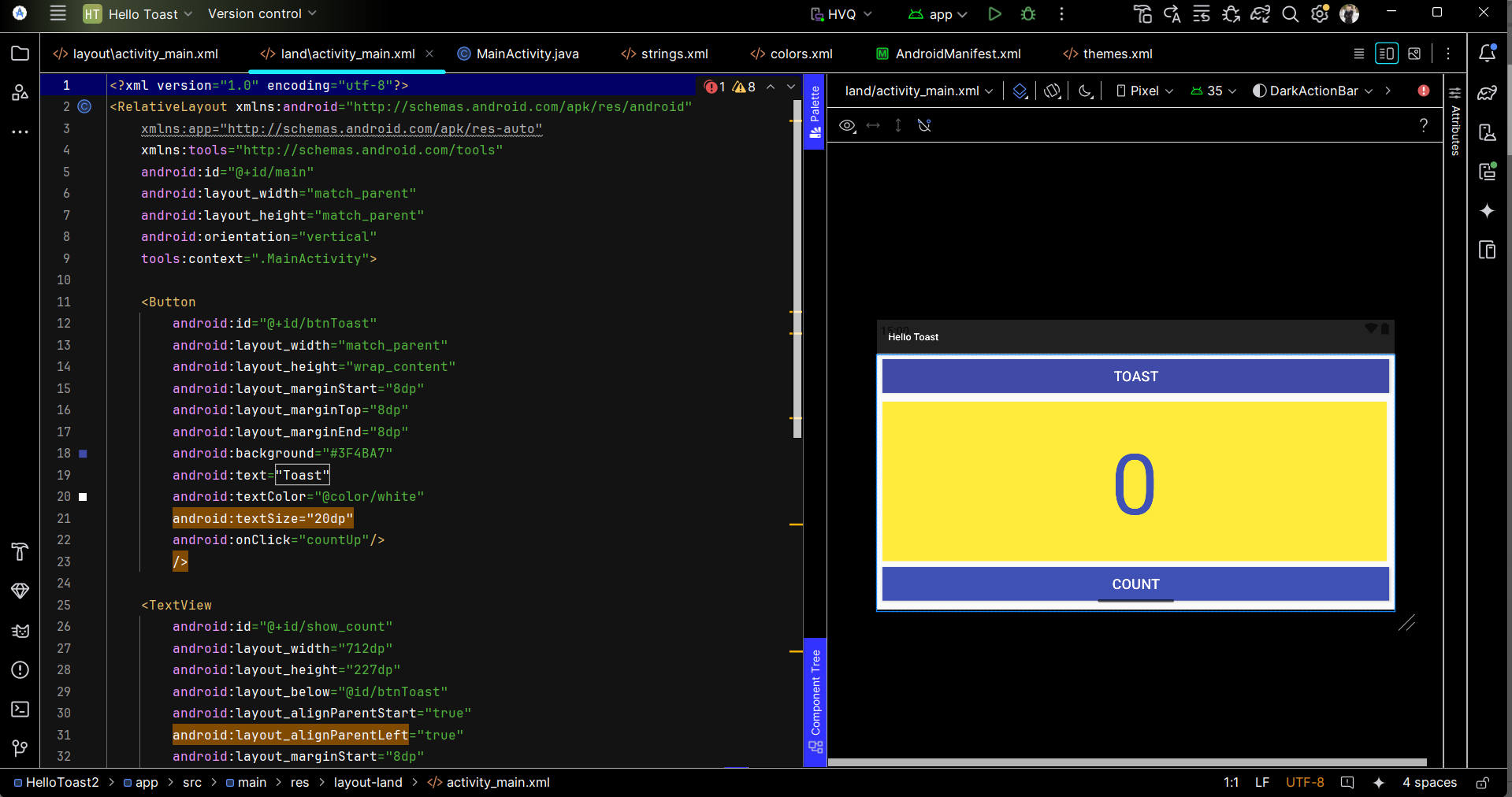
**1.3 Xem trước bố cục trên các thiết bị khác nhau**

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

1. Tab **land/activity\_main.xml** vẫn đang mở trong trình chỉnh sửa bố cục. Nếu không, hãy nhấp đúp vào tệp **activity\_main.xml (land)** trong thư mục **layout**.
2. Nhấp vào nút **Device in Editor** trên thanh công cụ phía trên.
3. Trong menu thả xuống, chọn một thiết bị khác. Ví dụ, chọn **Nexus 4**, **Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong các 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 các thuộc tính. Tuy nhiên, đôi khi việc chỉnh sửa trực tiếp mã XML trong tab **Text** sẽ nhanh hơn. Tab **Text** hiển thị mã XML và cung cấp một tab **Preview** ở phía bên phải cửa sổ để hiển thị bản xem trước bố cục, như minh họa trong hình bên dưới.



**Thay đổi bố cục với các bước chi tiết**

**Hình minh họa:**

1. **Tab Preview:** Sử dụng để hiển thị bảng xem trước bố cục.
2. **Preview Pane:** Hiển thị cách bố cục trông trên thiết bị.
3. **XML Code:** Phần mã nguồn của bố cục trong tab **Text**.

**Các bước thay đổi bố cục:**

1. Đảm bảo tab **land/activity\_main.xml** vẫn mở trong trình chỉnh sửa bố cục. Nếu không, nhấp đúp vào tệp **activity\_main.xml (land)** trong thư mục **layout**.
2. Chuyển sang tab **Text** và bật tab **Preview** (nếu chưa được chọn).
3. Trong mã XML, tìm phần tử **TextView**.
4. Thay đổi thuộc tính android:textSize="100sp" thành android:textSize="120sp". Kết quả thay đổi sẽ hiển thị ngay lập tức trong bản xem trước.

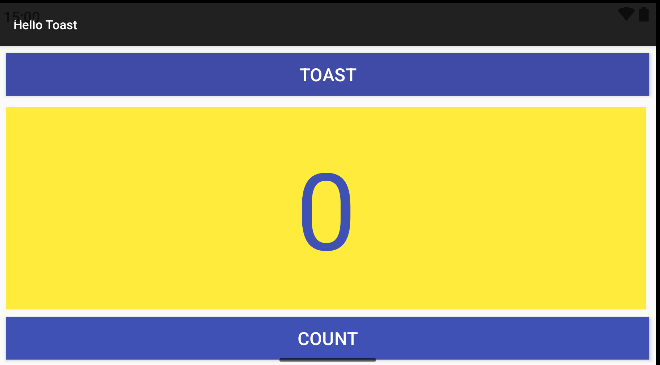
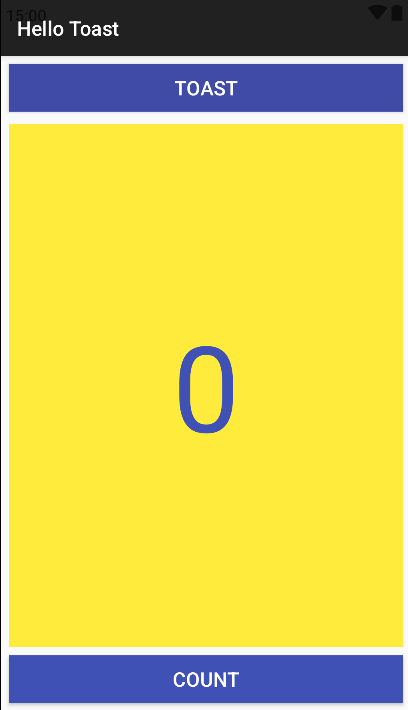


1. Sử dụng menu thả xuống **Device in Editor** để chọn các thiết bị khác nhau và xem cách bố cục hiển thị ở chế độ ngang trên các thiết bị đó.

**Lưu ý trong trình chỉnh sửa:**

* Tab **land/activity\_main.xml** hiển thị bố cục cho chế độ ngang.
* Tab **activity\_main.xml** hiển thị bố cục không thay đổi cho chế độ dọc.  
  Bạn có thể chuyển đổi qua lại giữa hai tab để so sánh.

1. Chạy ứng dụng trên trình giả lập hoặc thiết bị thực. Chuyển hướng hiển thị từ dọc sang ngang để xem cách cả hai bố cục hoạt động.



* 1. **Tạo một biến thể bố cục cho máy tính bảng**

Như bạn đã học trước đây, 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 phía trên.Nếu bạn chọn một thiết bị như **Nexus 10** (máy tính bảng) từ menu, bạn sẽ nhận thấy rằng bố cục không phù hợp cho màn hình máy tính bảng lớn - văn bản của mỗi nút (**Button**) quá nhỏ và cách sắp xếp các nút (**Button**) ở trên và dưới không lý tưởng cho màn hình lớn của máy tính bảng.

Để điều chỉnh bố cục phù hợp với máy tính bảng mà không ảnh hưởng đến bố cục cho các thiết bị điện thoại ở chế độ ngang hoặc dọc, hãy thực hiện các bước sau:

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

Sau khi thực hiện:

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

Bạn có thể thay đổi bố cục này, dành riêng cho máy tính bảng, mà không 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 cho bố cục này.

1. **Tắt công cụ Autoconnect** trên thanh công cụ. Ở bước này, hãy đảm bảo rằng công cụ đã bị vô hiệu hóa.
2. Xóa tất cả các ràng buộc (constraints) trong bố cục bằng cách nhấp vào nút **Clear All Constraints** trên thanh công cụ.
   * Khi các ràng buộc đã bị xóa, bạn có thể tự do di chuyển và thay đổi kích thước các phần tử trên bố cục.
3. Trình chỉnh sửa bố cục cung cấp các tay cầm (handles) ở bốn góc của mỗi phần tử để thay đổi kích thước chúng. Trong **Component Tree**, chọn **TextView** có tên **show\_count**. Để dời **TextView** ra ngoài đường đi, giúp bạn dễ dàng kéo các phần tử **Button**, hãy kéo một góc của nó để thay đổi kích thước, như được minh họa trong hình động dưới đây.

Khi thay đổi kích thước một phần tử, nó sẽ cố định các giá trị chiều rộng và chiều cao. Tránh cố định kích thước cho hầu hết các phần tử, vì điều này có thể dẫn đến giao diện không nhất quán trên các màn hình có kích thước và mật độ khác nhau.Trong bước này, việc thay đổi kích thước chỉ được thực hiện để di chuyển phần tử tạm thời, và các kích thước sẽ được điều chỉnh lại trong bước khác.

**Bước 4:**

Chọn nút **button\_toast** trong **Component Tree**, nhấp vào tab **Attributes** để mở bảng thuộc tính, và thay đổi **textSize** thành **60sp** (#1 trong hình bên dưới), **layout\_width** thành **wrap\_content** (#2 trong hình bên dưới

Như được hiển thị ở phía bên phải của hình trên (2), bạn có thể nhấp vào điều khiển chiều rộng của trình kiểm tra chế độ xem, xuất hiện dưới dạng hai đoạn ở hai bên trái và phải của hình vuông, cho đến khi nó hiển thị **Wrap Content**. Ngoài ra, bạn có thể chọn **wrap\_content** từ menu **layout\_width**.

Bạn sử dụng **wrap\_content** để nếu văn bản của **Button** được bản địa hóa sang một ngôn ngữ khác, nút **Button** sẽ hiển thị rộng hơn hoặc hẹp hơn để phù hợp với từ trong ngôn ngữ khác.

1. Chọn nút **button\_count** trong **Component Tree**, thay đổi **textSize** thành **60sp** và **layout\_width** thành **wrap\_content**, sau đó kéo nút **Button** lên trên **TextView** vào một không gian trống 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) chứa văn bản, như **TextView** hoặc **Button**, với một phần tử giao diện khác cũng chứa văn bản. Ràng buộc đường cơ sở (**baseline constraint**) cho phép bạn ràng buộc các phần tử sao cho các đường cơ sở của văn bản được căn thẳng hàng.

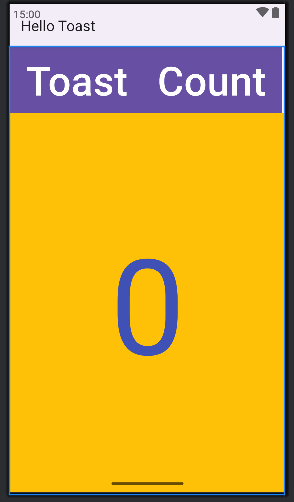
1. Ràng buộc nút **button\_toast** vào phía trên và bên trái của bố cục. Kéo nút **button\_count** đến một vị trí gần nút **button\_toast**. Sau đó, ràng buộc nút **button\_count** vào phía bên trái của nút **button\_toast**, như minh họa trong hình động.
2. **Sử dụng ràng buộc đường cơ sở (baseline constraint)**, bạn có thể ràng buộc nút **button\_count** sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của nút **button\_toast**. Chọn phần tử **button\_count**, sau đó di chuyển con trỏ chuột qua phần tử cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.
3. Nhấp vào nút ràng buộc đường cơ sở. Tay cầm ràng buộc đường cơ sở sẽ xuất hiện, nhấp nháy màu xanh lá. Nhấp và kéo một đường ràng buộc đường cơ sở đến đường cơ sở của phần tử **button\_toast** để hoàn tất.

**1.8 Mở rộng các nút theo chiều ngang**  
Nút **pack** trên thanh công cụ cung cấp các tùy chọn để sắp xếp hoặc mở rộng các phần tử giao diện người dùng đã chọn. Bạn có thể sử dụng nó để sắp xếp đều các nút **Button** theo chiều ngang trên giao diện.

1. Chọn nút **button\_count** trong **Component Tree**, sau đó nhấn giữ **Shift** và chọn thêm nút **button\_toast** để cả hai nút đều được chọn.
2. Nhấp vào nút **pack** trên thanh công cụ, sau đó chọn tùy chọn **Expand Horizontally** như minh họa trong hình.

Các phần tử **Button** sẽ mở rộng theo chiều ngang để lấp đầy toàn bộ giao diện như minh họa bên dưới.

1. Để hoàn thiện bố cục, hãy ràng buộc **show\_count TextView** vào phía dưới của nút **button\_toast** và ràng buộc nó với các cạnh bên cũng như cạnh dưới của bố cục, như được minh họa trong hình động bên dưới.
2. Các bước cuối cùng là thay đổi **layout\_width** và **layout\_height** của **show\_count TextView** thành **Match Constraints** và đặt **textSize** thành **200sp**. Bố cục cuối cùng sẽ trông như hình minh họa bên dưới.



1. Nhấn vào nút **Orientation in Editor** trên thanh công cụ ở phía trên và chọn **Switch to Landscape** (Chuyển sang chế độ ngang). Giao diện của bố cục trên máy tính bảng sẽ xuất hiện với hướng nằm ngang như hình dưới đây. (Bạn có thể chọn **Switch to Portrait** để quay lại chế độ dọc).
2. Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng màn hình sau khi chạy ứng dụng để xem giao diện trông như thế nào trên các loại thiết bị khác nhau. Bạn đã thành công trong việc tạo một ứng dụng có giao diện người dùng (UI) hoạt động đúng trên điện thoại và máy tính bảng với các kích thước và mật độ màn hình khác nhau.  
   Mẹo: Để có hướng dẫn chi tiết về cách sử dụng ConstraintLayout, hãy xem **Using ConstraintLayout to design your views**.

**Thử thách:** Để hỗ trợ giao diện ngang (landscape) cho máy tính bảng, bạn có thể căn giữa các nút (Button) trong tệp activity\_main.xml (xlarge) để chúng xuất hiện như hình minh họa bên dưới.  
**Gợi ý:** Chọn các phần tử, nhấp vào nút căn chỉnh trong thanh công cụ, và chọn **Căn giữa theo chiều ngang (Center Horizontally)**.

**Task 2: Thay đổi bố cục thành LinearLayout**

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

LinearLayout yêu cầu có các thuộc tính sau:

* **layout\_width**
* **layout\_height**
* **orientation**

**layout\_width** và **layout\_height** có thể nhận một trong các giá trị sau:

* **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 của màn hình (**parent view**).
* **wrap\_content**: Thu nhỏ kích thước view sao cho nó vừa đủ chứa nội dung. Nếu không có nội dung, view sẽ trở nên vô hình.
* **Số dp cố định (density-independent pixels)**: Xác định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ, **16dp** có nghĩa là 16 pixel độc lập với mật độ.

**orientation** có thể là:

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