

CHƯƠNG I : LÀM QUEN

Bài 1: Tạo ứng dụng đầu tiên

1.1.Android Studio và Hello World

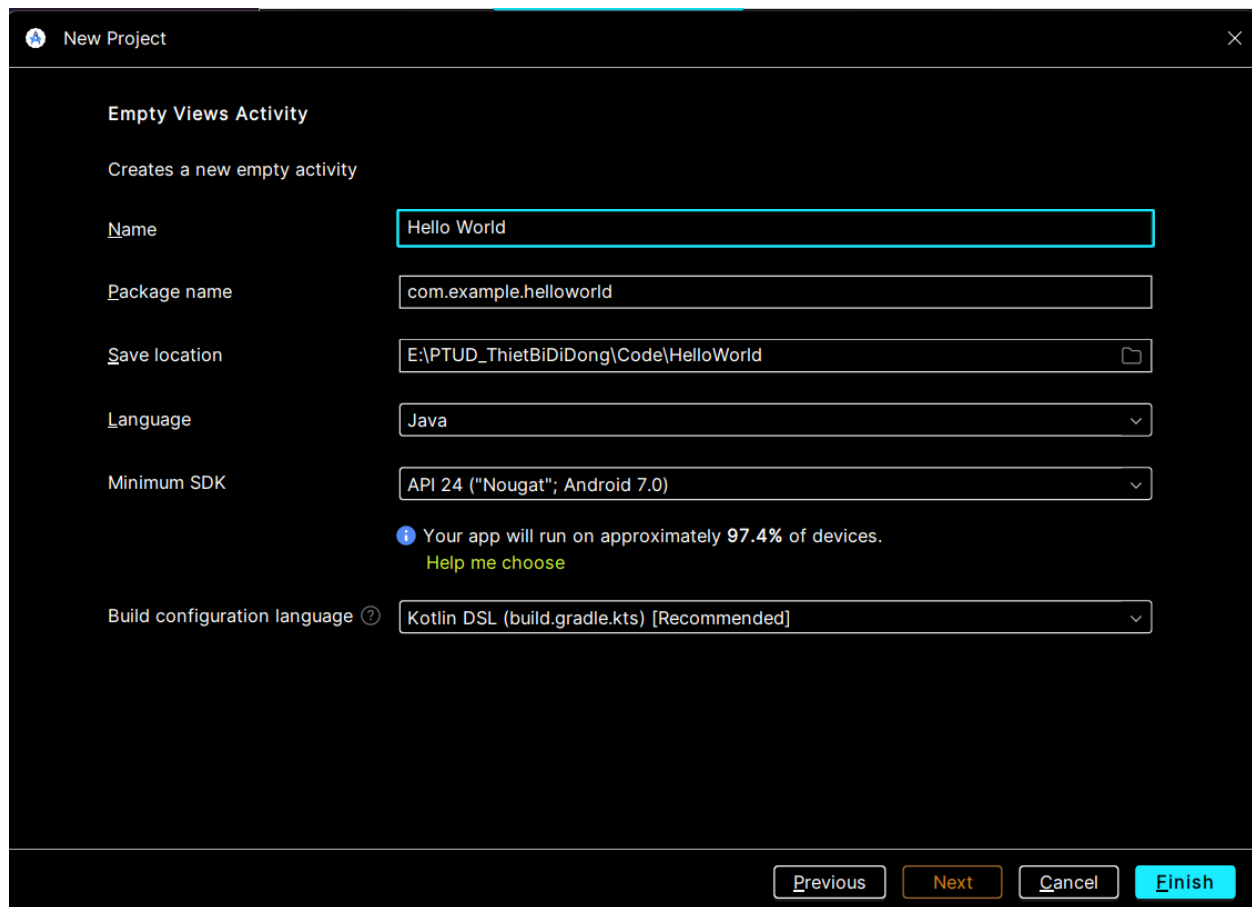
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

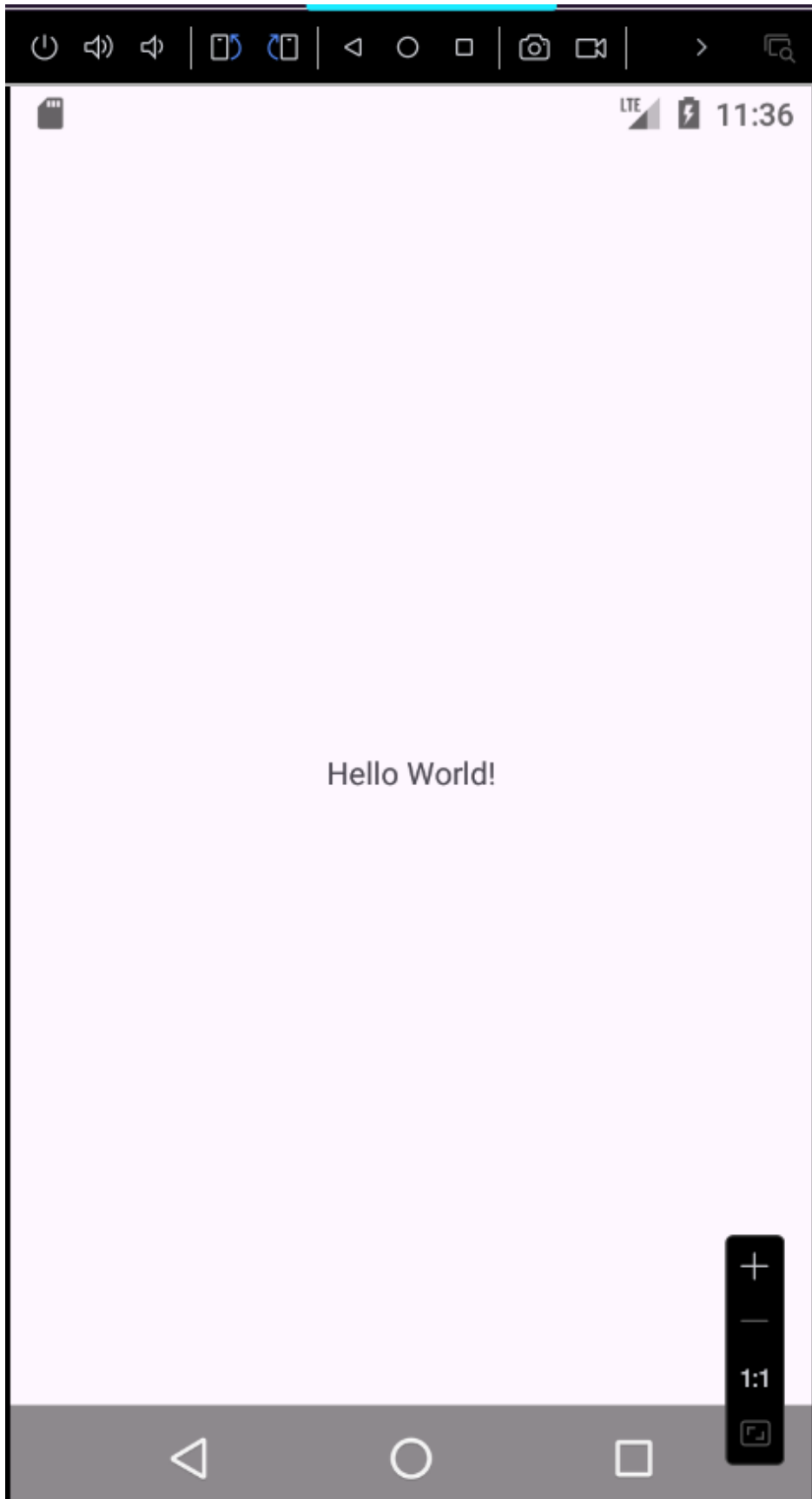
Những gì bạn sẽ làm

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

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 hàng 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 Windows, 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 biệt nào sẽ được ghi chú riêng.

Các bước cài đặt:

1. Truy cập trang web chính thức của Android Developers và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
2. Chấp nhận các thiết lập mặc 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.
3. 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.
4. Quá trình 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:

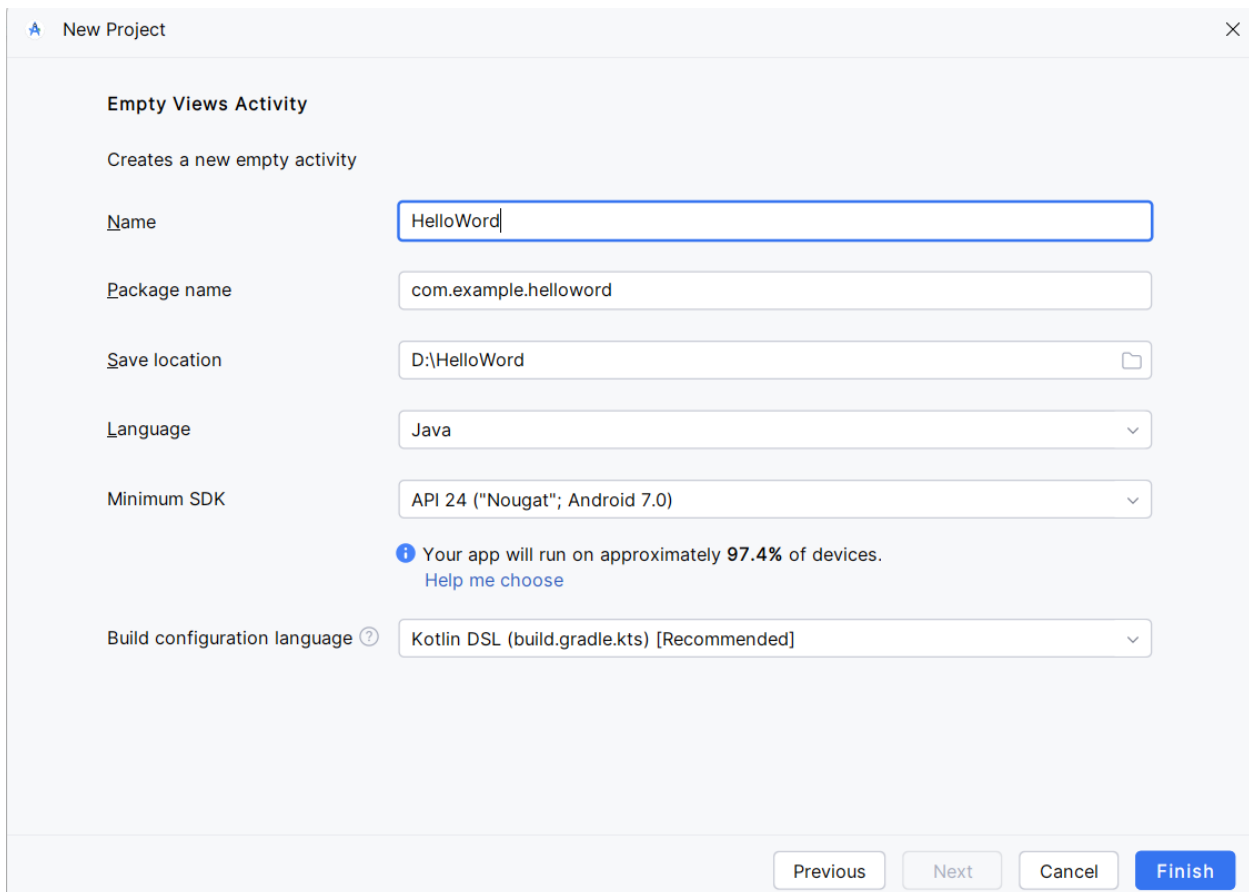
Nếu gặp sự cố khi cài đặt, hãy kiểm tra ghi chú phát hành của Android Studio hoặc tìm sự trợ giúp từ giảng viên của bạn.

Bài 2 : Tạo ứng dụng Hello Word

Trong bài này, bạn sẽ tạo một ứng dụng hiển thị ra dòng chữ “Hello Word” để xác minh rằng Android studio đã được cài đặt chính xác, và tìm hiểu những điều cơ bản của việc phát triển với Android Studio.

2.1. Tạo dự án ứng dụng

1. Mở Android Studio nếu nó chưa được mở
2. Trong cửa sổ chính Welcome to Android Studio, hãy nhấp vào Start a new Android Studio project.
3. Trong cửa sổ Create Android Project, hãy nhập Hello World cho tên ứng dụng



New Project

Empty Views Activity

Creates a new empty activity

Name: HelloWord

Package name: com.example.helloworld

Save location: D:\HelloWord

Language: Java

Minimum SDK: API 24 ("Nougat"; Android 7.0)

! Your app will run on approximately 97.4% of devices.
[Help me choose](#)

Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

Previous Next Cancel Finish

4. Xác minh rằng vị trí Dự án mặc định là nơi bạn muốn lưu trữ ứng dụng Hello World và các dự án Android Studio khác hoặc thay đổi thành thư mục bạn muốn.

5. Chấp nhận android.example.com mặc định cho Tên miền công ty hoặc tạo một tên miền công ty duy nhất. Nếu bạn không có kế hoạch phát hành ứng dụng của mình, bạn có thể chấp nhận mặc định. Lưu ý rằng việc thay đổi tên gói ứng dụng của bạn sau này sẽ tốn thêm công sức.

6. Bỏ chọn các tùy chọn Bao gồm hỗ trợ C++ và Bao gồm hỗ trợ Kotlin, rồi nhấp vào Tiếp theo.

7. Trên màn hình **Target Android Devices, Phone and Tablet** phải được chọn. Đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm SDK tối thiểu; nếu không, hãy sử dụng menu bật lên để đặt.

-> Đây là các thiết lập được sử dụng bởi các ví dụ trong các bài học cho khóa học này. Tính đến thời điểm viết bài này, các thiết lập này giúp ứng dụng Hello World của bạn tương thích với 97% thiết bị Android đang hoạt động trên Cửa hàng Google Play

8. Bỏ chọn mục **Include Instant App support** và tất cả các tùy chọn khác. Sau đó, nhấp vào **Next**. Nếu dự án của bạn yêu cầu các thành phần bổ sung cho SDK mục tiêu đã chọn, Android Studio sẽ tự động cài đặt chúng.

9. Cửa sổ **Add an Activity** xuất hiện. Hoạt động là một điều duy nhất, tập trung mà người dùng có thể thực hiện. Đây là thành phần quan trọng của bất kỳ ứng dụng Android nào. Hoạt động thường có bố cục liên quan đến nó để xác định cách các thành phần UI xuất hiện trên màn hình. Android Studio cung cấp các mẫu Hoạt động để giúp bạn bắt đầu. Đối với dự án Hello World, hãy chọn **Empty Activity** như được hiển thị bên dưới và nhấp vào **Next**.

10. Màn hình **Configure Activity** xuất hiện (khác nhau tùy thuộc vào mẫu bạn đã chọn ở bước trước). Theo mặc định, Empty Activity do mẫu cung cấp được đặt tên là MainActivity. Bạn có thể thay đổi tên này nếu muốn, nhưng bài học này sử dụng MainActivity.

11. Đảm bảo rằng tùy chọn **Generate Layout file** được chọn. Tên bố cục theo mặc định là activity_main. Bạn có thể thay đổi tùy chọn này nếu muốn, nhưng bài học này sử dụng activity_main.

12. Đảm bảo rằng tùy chọn **Backwards Compatibility (App Compat)** được chọn. Điều này đảm bảo rằng ứng dụng của bạn sẽ tương thích ngược với các phiên bản Android trước đó.

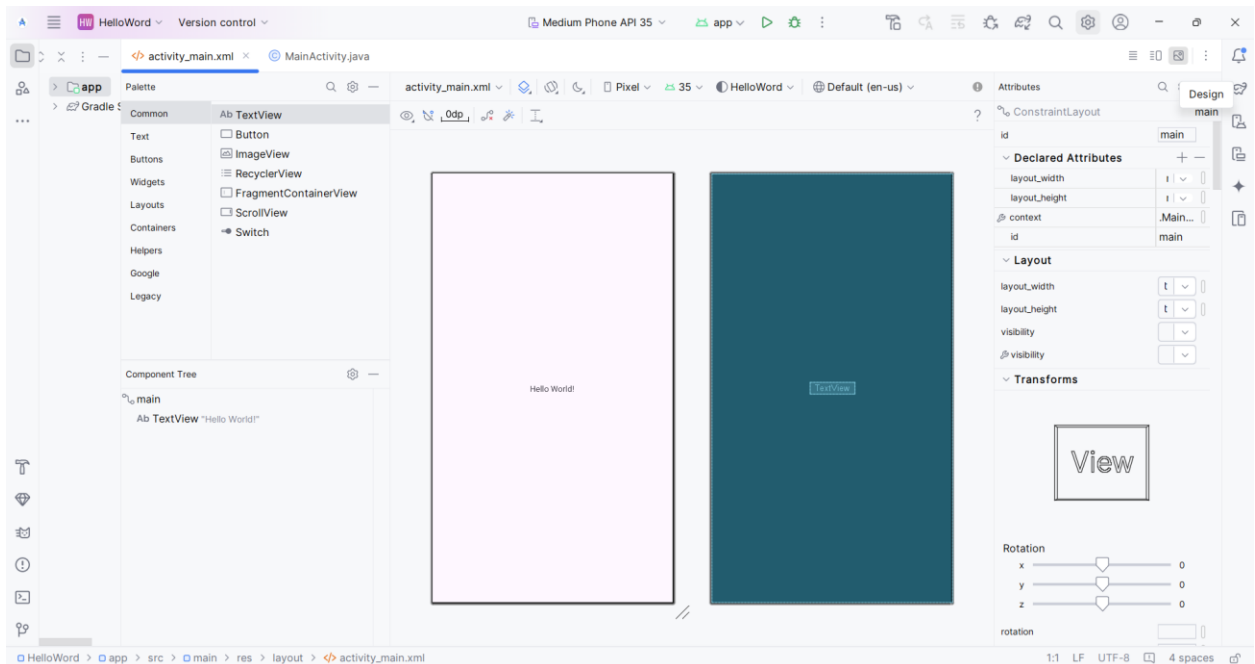
13. Nhấp vào Finish.

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

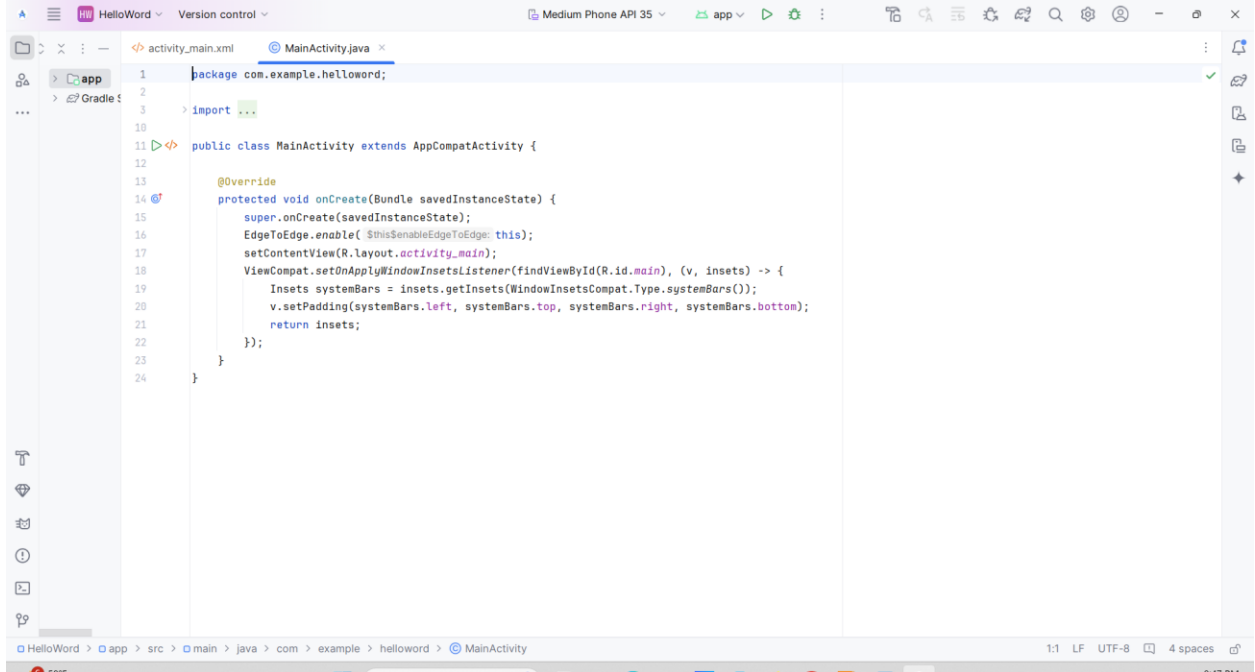
Mẹo: Xem trang "Cấu hình bản dựng" dành cho nhà phát triển để biết thông tin chi tiết. Bạn cũng có thể thấy thông báo "Mẹo trong ngày" với các phím tắt và các mẹo hữu ích khác. Nhấp vào **Close** để đóng thông báo.

Trình chỉnh sửa Android Studio xuất hiện. Thực hiện theo các bước sau:

1. Nhấp vào tab `activity_main.xml` để xem trình chỉnh sửa bố cục.
2. Nhấp vào tab **Design** của trình chỉnh sửa bố cục, nếu chưa được chọn, để hiển thị bản trình bày đồ họa của bố cục như được hiển thị bên dưới.



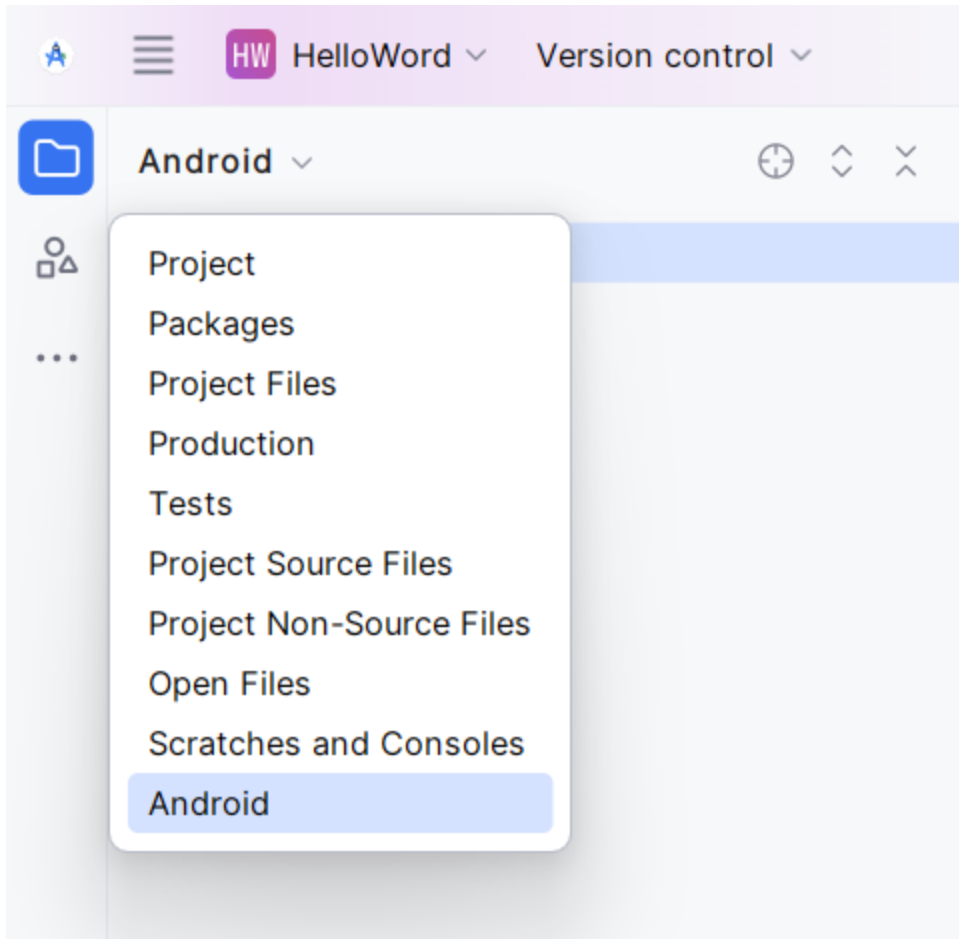
3. Nhấp vào tab `MainActivity.java` để mở trình chỉnh sửa mã (code editor), như được hiển thị bên dưới.



2.2. Khám phá **Project** > **Android** pane

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

1. Nếu chưa được chọn, nhấp vào tab **Project** trong cột tab dọc ở phía bên trái cửa sổ Android Studio. **Project** pane sẽ xuất hiện.
2. Để xem dự án theo cấu trúc thư mục chuẩn của Android, chọn **Android** từ menu bật lên ở đầu **Project** pane, như minh họa bên dưới.

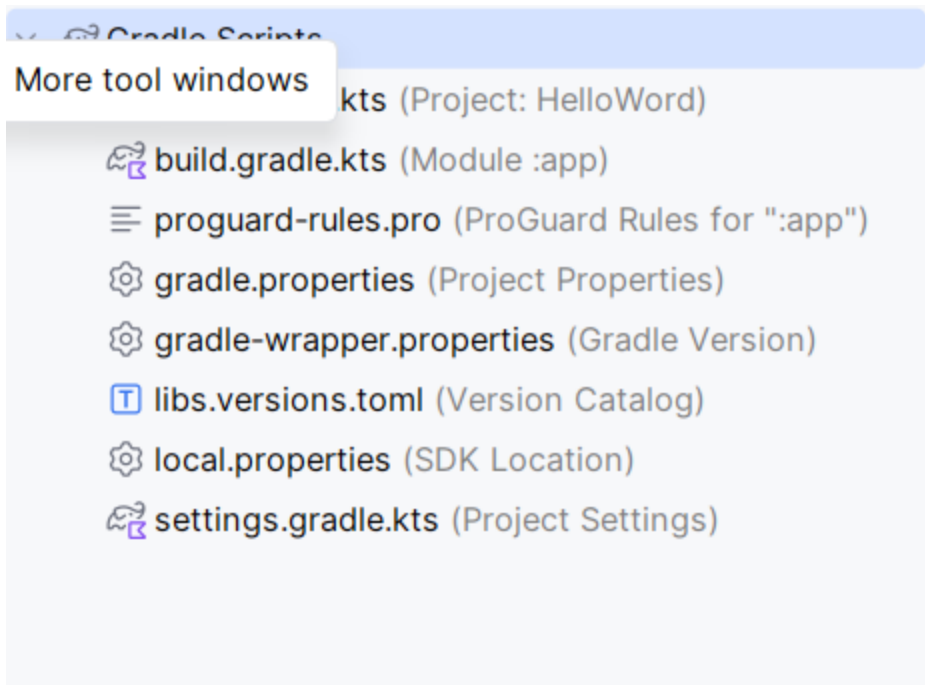


Lưu ý: Chương này và các chương khác đề cập đến **Project pane**, khi được đặt ở chế độ **Android**, sẽ được gọi là **Project > Android pane**.

2.3. Khám phá thư mục Gradle Scripts

Hệ thống build Gradle trong Android Studio giúp bạn dễ dàng thêm các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác làm dependencies vào quá trình build của bạn.

Khi bạn tạo một dự án ứng dụng lần đầu tiên, **Project > Android pane** sẽ xuất hiện với thư mục **Gradle Scripts** được mở rộng, như minh họa bên dưới.



Thực hiện theo các bước sau để khám phá hệ thống Gradle:

1. Mở rộng thư mục Gradle Scripts:

- Nếu thư mục **Gradle Scripts** chưa được mở rộng, nhấp vào biểu tượng tam giác để mở rộng nó.
- Thư mục này chứa tất cả các tệp cần thiết cho hệ thống build.

2. Tìm tệp build.gradle.kts (Project: HelloWorld):

- Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô-đun trong dự án của mình.
- Mỗi dự án Android Studio đều chứa một tệp Gradle build cấp cao nhất. Thông thường, bạn không cần thay đổi tệp này, nhưng vẫn nên hiểu nội dung của nó.

Cấu trúc mặc định của tệp build cấp cao nhất:

- Tệp này sử dụng khối buildscript để định nghĩa các kho lưu trữ (repositories) và dependencies chung cho tất cả các mô-đun trong dự án.

- Nếu dependency của bạn không phải là thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm kiếm các tệp trong các kho trực tuyến được chỉ định trong khối repositories.

Mặc định, các dự án mới trong Android Studio sẽ khai báo các kho lưu trữ sau:

- **JCenter**
- **Google** (bao gồm Google Maven repository).

```
1 // Top-level build file where you can add configuration options common to all sub-projects/modules
2 plugins {
3     id 'com.android.application' apply false
4 }
```

3. Tìm tệp build.gradle (Module: app)

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

Mục đích sử dụng

- Đây thường là tệp mà bạn sẽ chỉnh sửa nhiều nhất khi thay đổi các cấu hình ở cấp ứng dụng, chẳng hạn như khai báo dependencies trong phần dependencies.

Khai báo dependencies

- Bạn có thể khai báo một dependency thư viện bằng một trong nhiều cấu hình khác nhau. Mỗi cấu hình dependency cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện.
- Ví dụ:
 - Lệnh implementation fileTree(dir: 'libs', include: ['*.jar']) thêm dependency cho tất cả các tệp **".jar"** bên trong thư mục libs.

Ví dụ tệp build.gradle (Module: app)

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

```
plugins {  
    alias(libs.plugins.android.application)  
}  
  
android {  
    namespace = "com.example.helloworld"  
    compileSdk = 35  
  
    defaultConfig {  
        applicationId = "com.example.helloworld"  
        minSdk = 24  
        targetSdk = 35  
        versionCode = 1  
        versionName = "1.0"  
  
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
    }  
  
    buildTypes {  
        release {  
            isMinifyEnabled = false  
            proguardFiles(  
                getDefaultProguardFile("proguard-android-optimize.txt"),  
                "proguard-rules.pro"  
            )  
        }  
    }  
  
    compileOptions {  
        sourceCompatibility = JavaVersion.VERSION_11  
        targetCompatibility = JavaVersion.VERSION_11  
    }  
}
```

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

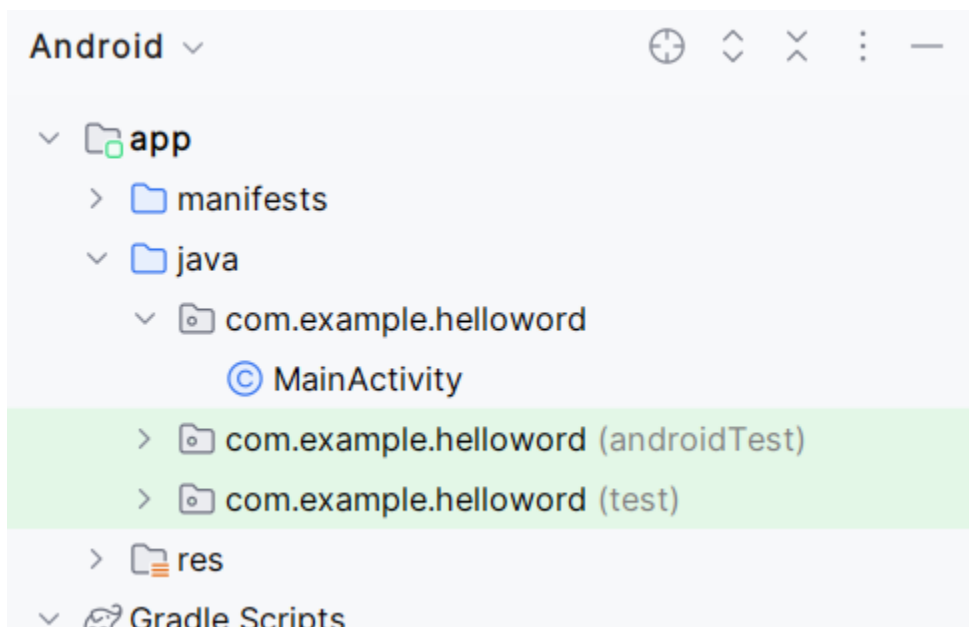
3. Nhấp vào biểu tượng tam giác để đóng thư mục Gradle Scripts.

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 tiếp tục mở rộng thư mục **com.example.android.helloworld** để xem tệp Java MainActivity.

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



Thư mục java

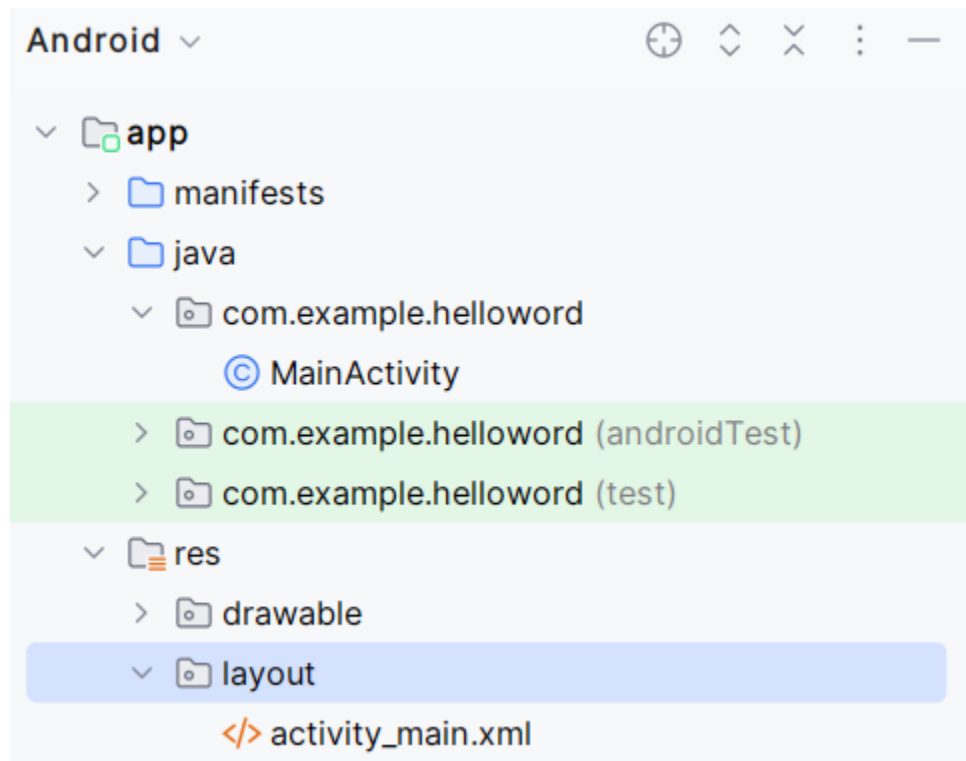
Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như minh họa trong hình trên:

- **Thư mục com.example.hello.helloworld** (hoặc tên miền mà bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng.
- **Hai thư mục khác** được sử dụng cho việc kiểm thử và sẽ được mô tả trong một bài học khác.

Trong ứng dụng Hello World, chỉ có một gói và nó chứa tệp MainActivity.java.

- Tệp này đại diện cho **Activity đầu tiên** (màn hình đầu tiên) mà người dùng nhìn thấy.
- Tệp MainActivity cũng khởi tạo các tài nguyên toàn cục của ứng dụng.
- Trong **Project > Android pane**, phần mở rộng tệp (.java) được bỏ qua để hiển thị ngắn gọn.

2.Mở rộng thư mục **res** và sau đó mở rộng thư mục **layout**. Nhấp đúp vào tệp activity_main.xml để mở nó trong trình chỉnh sửa giao diện (**layout editor**).



Thư mục **res** chứa các tài nguyên (resources) của ứng dụng, chẳng hạn như **Layouts**: Các bố cục giao diện người dùng (UI), **Strings**: Các chuỗi văn bản, **Images**: Các hình ảnh. Mỗi **Activity** thường được liên kết với một bố cục UI, được định nghĩa trong một tệp XML. Tệp bố cục này thường được đặt tên theo tên của **Activity** tương ứng.

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. Thông tin này cần thiết để hệ thống có thể chạy mã của ứng dụng.

Các bước thực hiện:

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

Thông tin về tệp AndroidManifest.xml

- Tệp AndroidManifest.xml mô tả tất cả các thành phần của ứng dụng Android, chẳng hạn như:
 - Các **Activity**.
 - Các **Service**, **BroadcastReceiver**, và **ContentProvider**.
- Mỗi thành phần của ứng dụng phải được khai báo trong tệp này.

Trong các bài học khác, bạn sẽ chỉnh sửa tệp này để:

- Thêm tính năng.
- Cấp quyền sử dụng tính năng cho ứng dụng.

Để tìm hiểu thêm, xem [App Manifest Overview](#).

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị Android bao gồm một hệ thống các đối tượng gọi là view — mọi thành phần trên màn hình đều là một View. Lớp View đại diện cho khối xây dựng cơ bản của tất cả các thành phần giao diện người dùng và là lớp cơ sở cho các lớp cung cấp các thành phần giao diện tương tác như nút bấm, hộp kiểm, và các trường nhập văn bản. Các lớp con View thường được sử dụng, được mô tả qua nhiều bài học, bao gồm:

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

Mã Java hiển thị và điều khiển giao diện người dùng được chứa trong một lớp mở rộng từ **Activity**. Một **Activity** thường được liên kết với một bố cục giao diện người dùng được định nghĩa dưới dạng tệp XML (eXtended Markup Language). Tệp XML này thường được đặt tên theo tên của Activity và định nghĩa 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 bố cục **activity_main.xml**, tệp này bao gồm một **TextView** với văn bản "Hello World".

Trong các ứng dụng phức tạp hơn, một **Activity** có thể thực hiện các hành động để phản hồi các 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 — một ứng dụng cho phép tương tác của người dùng. Bạn sẽ tạo một ứng dụng sử dụng mẫu **Empty Activity**. Bạn cũng học cách sử dụng trình chỉnh sửa bố cục để thiết

kế bố cục, và cách chỉnh sửa bố cục trong XML. Bạn cần phát triển các kỹ năng này để hoàn thành các bài thực hành khác trong khóa học này.

Những gì bạn cần biết trước

Bạn cần quen thuộc với:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử **Button** và một **TextView** vào bố cục.
- Thao tác từng phần tử trong **ConstraintLayout** để ràng buộc chúng với lề và các phần tử khác.
- Thay đổi thuộc tính của các phần tử giao diện người dùng (UI).
- Chỉnh sửa bố cục của ứng dụng trong XML.
- Trích xuất các chuỗi mã cứng thành tài nguyên chuỗi (string resources).
- Triển khai các phương thức xử lý sự kiện khi nhấn (click-handler) để hiển thị thông báo trên màn hình khi người dùng chạm vào mỗi nút bấm (Button).

Tổng quan về ứng dụng

Ứng dụng Hellotoast bao gồm hai phần tử nút và một TextView. Khi người dùng chạm vào đầu tiên

Nút, nó hiển thị một tin nhắn ngắn (bánh mì nướng) trên màn hình. Nhấn vào nút thứ hai tặng một

"Nhấp vào" Bộ đếm được hiển thị trong TextView, bắt đầu từ 0.

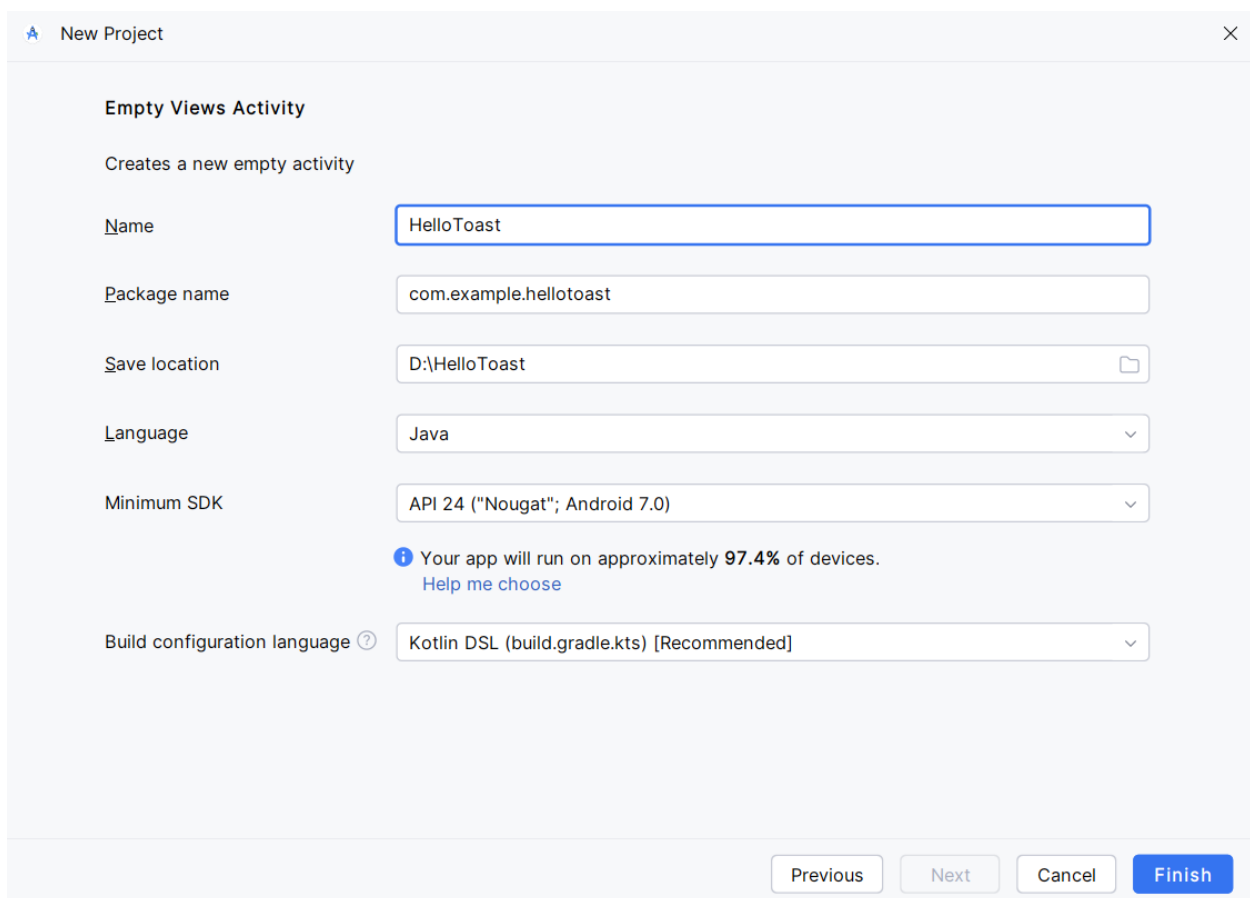
Đây là những gì ứng dụng đã hoàn thành trông như thế nào:

Nhiệm vụ 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 tới mã giải pháp sẽ được cung cấp ở phần cuối.

1.1 Tạo dự án Android Studio

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



New Project

Empty Views Activity

Creates a new empty activity

Name: HelloToast

Package name: com.example.hellotoast

Save location: D:\HelloToast

Language: Java

Minimum SDK: API 24 ("Nougat"; Android 7.0)

Information: Your app will run on approximately 97.4% of devices. [Help me choose](#)

Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

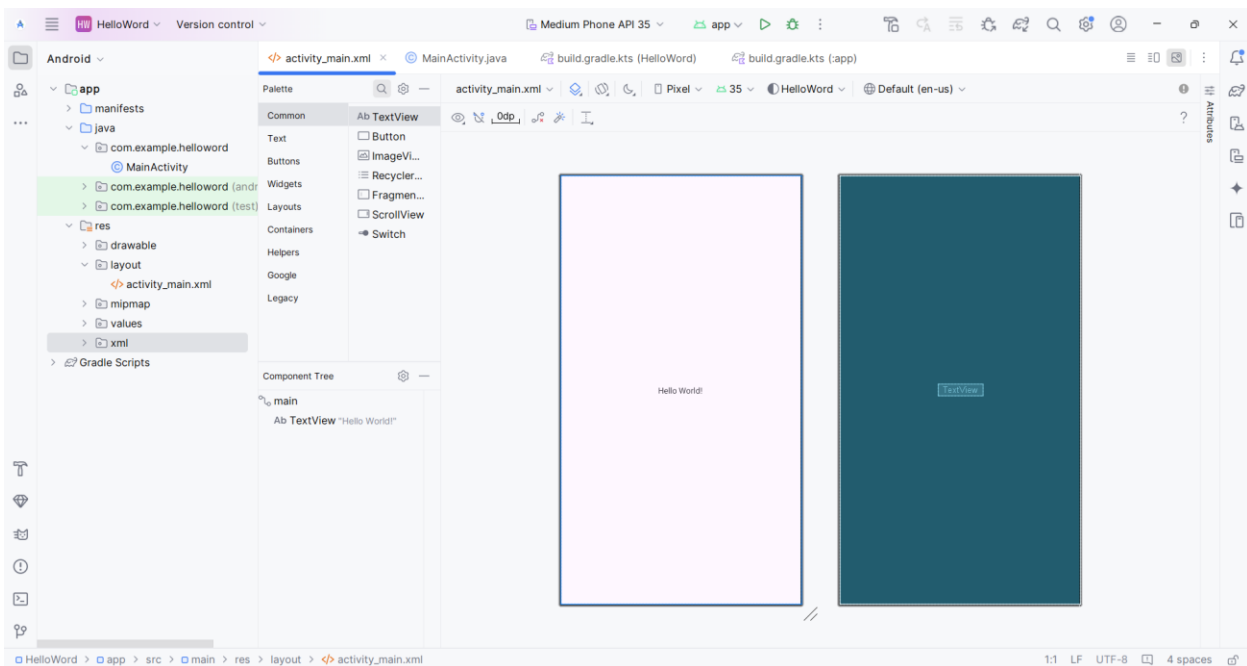
Previous Next Cancel Finish

15. Chọn Run > Run app hoặc nhấp vào biểu tượng Run trên thanh công cụ để xây dựng và chạy ứng dụng trên trình giả lập hoặc thiết bị của bạn.

1.2 Khám phá trình chỉnh sửa bố cục (layout editor)

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

Khám phá trình chỉnh sửa bố cục, và tham khảo hình minh họa bên dưới khi bạn thực hiện theo các bước được đánh số:



1. Trong thư mục `app > res > layout` trong ngăn Project > Android, nhấp đúp vào tệp `activity_main.xml` để mở, nếu tệp chưa được mở.
2. Nhấp vào tab Design nếu tab này chưa được chọn. Bạn sử dụng tab Design để thao tác các phần tử và bố cục, và tab Text để chỉnh sửa mã XML của bố cục.
3. Ngăn Palettes hiển thị các phần tử giao diện người dùng (UI) mà bạn có thể sử dụng trong bố cục ứng dụng của mình.
4. Ngăn Component tree hiển thị cấu trúc cây phân cấp của các phần tử UI. Các phần tử giao diện được tổ chức thành cấu trúc cây bao gồm cha và con,

trong đó một phần tử con kế thừa các thuộc tính từ phần tử cha. Trong hình minh họa, TextView là phần tử con của ConstraintLayout. Bạn sẽ tìm hiểu thêm về các phần tử này trong bài học sau.

5. Các ngăn thiết kế và bản thiết kế trong trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình minh họa, bố cục chỉ hiển thị một phần tử: một TextView hiển thị dòng chữ "Hello World".
6. Tab Attributes hiển thị ngăn Attributes dùng để thiết lập các thuộc tính cho một phần tử UI.

Mẹo: Tham khảo [Xây dựng giao diện người dùng với Layout Editor](#) để biết chi tiết về cách sử dụng trình chỉnh sửa bố cục, và [Giới thiệu về Android Studio](#) để xem toàn bộ tài liệu hướng dẫn về Android Studio.

Nhiệm vụ 2: Thêm các phần tử View trong trình chỉnh sửa bố cục

Trong nhiệm vụ này, bạn 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 các ràng buộc (constraints) theo cách thủ công, như được mô tả bên dưới, hoặc tự động bằng cách sử dụng công cụ Autoconnect.

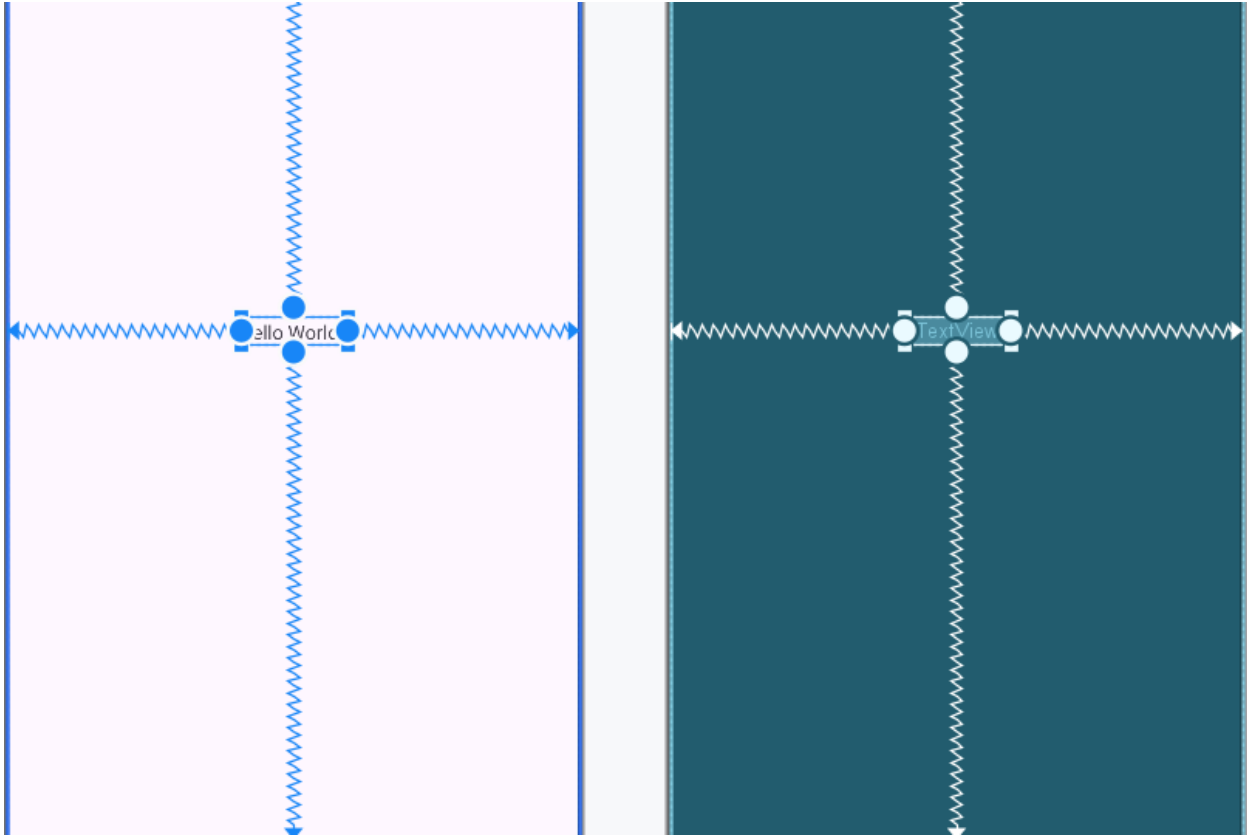
2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện theo các bước sau:

1. Mở tệp activity_main.xml từ ngăn Project > Android nếu nó chưa được mở. Nếu tab Design chưa được chọn, nhấp vào tab này. Nếu không có bản thiết kế (blueprint), nhấp vào nút Select Design Surface trên thanh công cụ và chọn tùy chọn Design + Blueprint.
2. Công cụ Autoconnect cũng nằm trên thanh công cụ và được bật theo mặc định. Đảm bảo rằng công cụ này không bị tắt ở bước này.
3. Nhấp vào nút phóng to (zoom in) để phóng to các ngăn thiết kế và bản thiết kế, giúp bạn xem kỹ hơn.
4. Chọn TextView trong ngăn Component Tree. TextView với dòng chữ "Hello World" sẽ được đánh dấu trong các ngăn thiết kế và bản thiết kế, đồng thời các ràng buộc của phần tử sẽ hiển thị.
5. Tham khảo hình minh họa động dưới đây cho bước này. Nhấp vào chốt tròn (circular handle) ở bên phải của TextView để xóa ràng buộc ngang (horizontal constraint) liên kết phần tử này với cạnh phải của bố cục.

Khi xóa ràng buộc này, TextView sẽ nhảy sang phía bên trái vì nó không còn bị ràng buộc với cạnh phải.

Để thêm lại ràng buộc ngang, nhấp vào cùng chốt tròn và kéo một đường đến cạnh phải của bố cục.



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

- **Chốt ràng buộc (Constraint handle):** Để tạo một ràng buộc như minh họa trong hình động ở trên, nhấp vào chốt ràng buộc (hiển thị dưới dạng hình tròn ở bên cạnh của một phần tử). Sau đó, kéo chốt này đến một chốt ràng buộc khác hoặc đến ranh giới của bố cục cha. Một đường zigzag biểu thị ràng buộc được tạo.



- **Chốt thay đổi kích thước (Resizing handle):** Để thay đổi kích thước của phần tử, kéo các chốt thay đổi kích thước hình vuông. Trong khi bạn kéo, chốt này sẽ chuyển thành góc nghiêng.



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 giao diện (UI) với bố cục cha. Sau khi bạn kéo phần tử vào bố cục, nó sẽ tạo các ràng buộc dựa trên vị trí của phần tử.

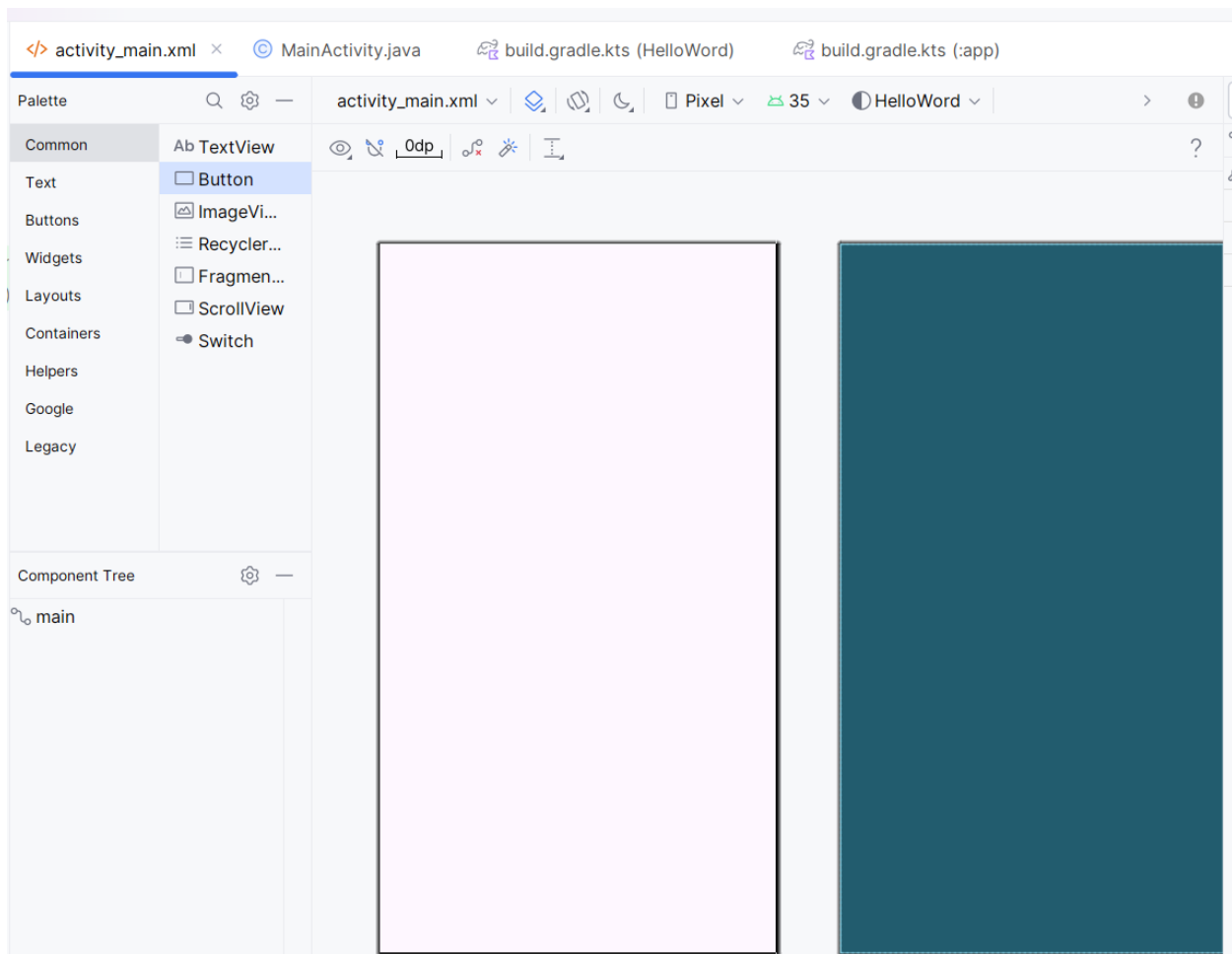
Thực hiện các bước sau để thêm một nút:

1. Bắt đầu với một bố cục trống:

Phần tử TextView không cần thiết, vì vậy khi nó vẫn đang được chọn, nhấn phím Delete hoặc chọn Edit > Delete. Lúc này, bạn sẽ có một bố cục hoàn toàn trống.

2. Kéo một nút từ ngăn Palette vào bố cục:

Kéo nút Button từ ngăn Palette vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả nút ở khu vực giữa trên cùng của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc đến cạnh trên, cạnh trái và cạnh phải của bố cục như minh họa trong hình động bên dưới.



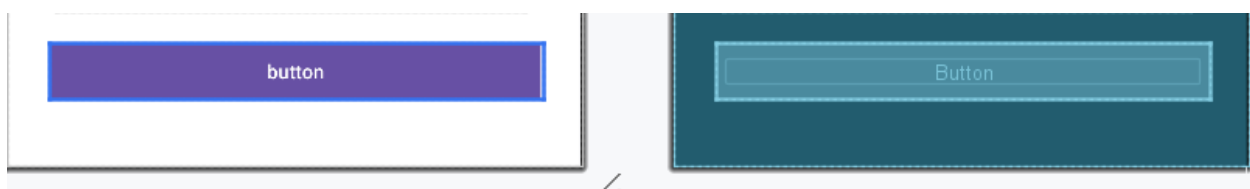
2.3 Thêm một nút thứ hai vào bố cục

1. Kéo một nút khác từ ngăn Palette:

Kéo một nút Button khác từ ngăn Palette vào giữa bố cục như minh họa trong hình động bên dưới. Công cụ Autoconnect có thể tự động tạo các ràng buộc ngang (horizontal constraints) cho bạn. Nếu không, bạn có thể tự kéo các ràng buộc này.

2. Thêm ràng buộc dọc:

Kéo một ràng buộc dọc (vertical constraint) từ nút này đến cạnh dưới của bố cục (tham khảo hình minh họa bên dưới).



Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách:

- Chọn phần tử và di chuyển con trỏ chuột qua nó để hiển thị nút **Clear Constraints**. Nhấp vào nút này để xóa tất cả ràng buộc của phần tử được chọn.
- Để xóa một ràng buộc cụ thể, hãy nhấp vào chốt cụ thể đã đặt ràng buộc đó.

Xóa tất cả các ràng buộc trong toàn bộ bố cục:

- Nhấp vào công cụ **Clear All Constraints** trong thanh công cụ. Công cụ này rất hữu ích nếu bạn muốn làm lại tất cả các ràng buộc trong bố cục của mình.

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.

Attributes

button_count

background	#1565C0
backgroundTint	#1565C0
id	button_count
text	COUNT

Layout

Constraint Widget

Constraints

- Start → StartOf **parent** (0dp)
- End → EndOf **parent** (0dp)
- Bottom → BottomOf **parent** (48dp)
- Horizontal Bias (0.534)

layout_width	353dp
layout_height	40dp
visibility	
visibility	

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):

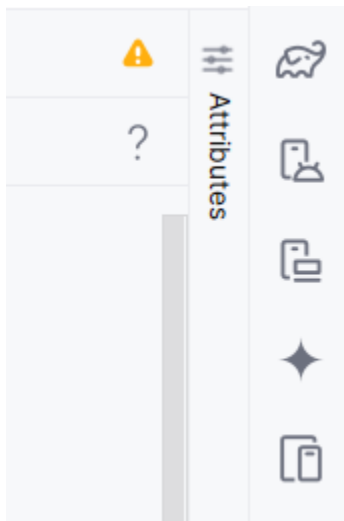
- Điều khiển này xác định thuộc tính `layout_width` và xuất hiện ở hai đoạn bên trái và bên phải của hình vuông.
- Các góc xiên cho biết điều khiển này được đặt thành `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần để vừa với nội dung, đến giới hạn margin 8dp.

3. Nút đóng ngăn Attributes (Attributes pane close button):

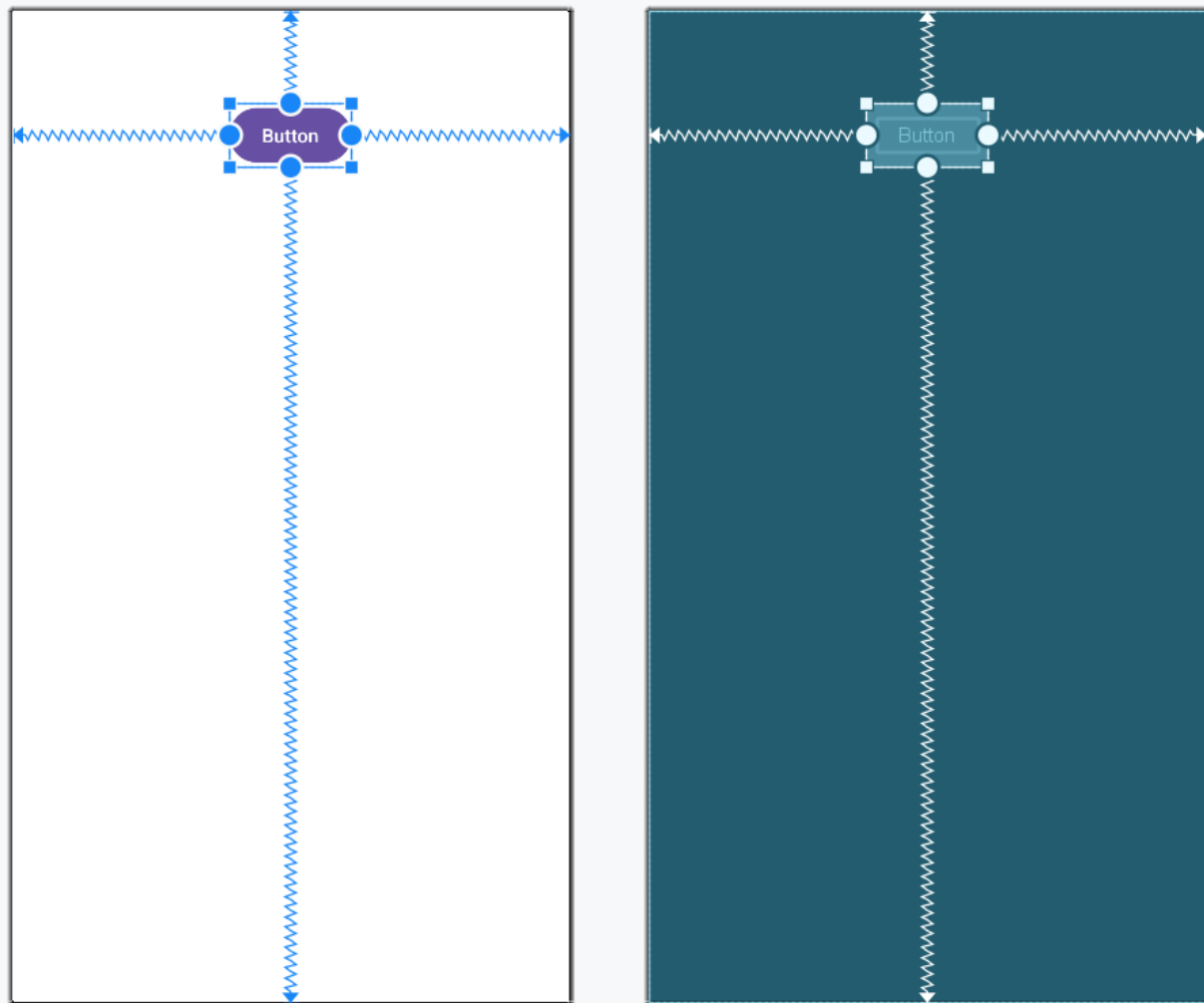
- Nhấp vào nút này để đóng ngăn Attributes.

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.

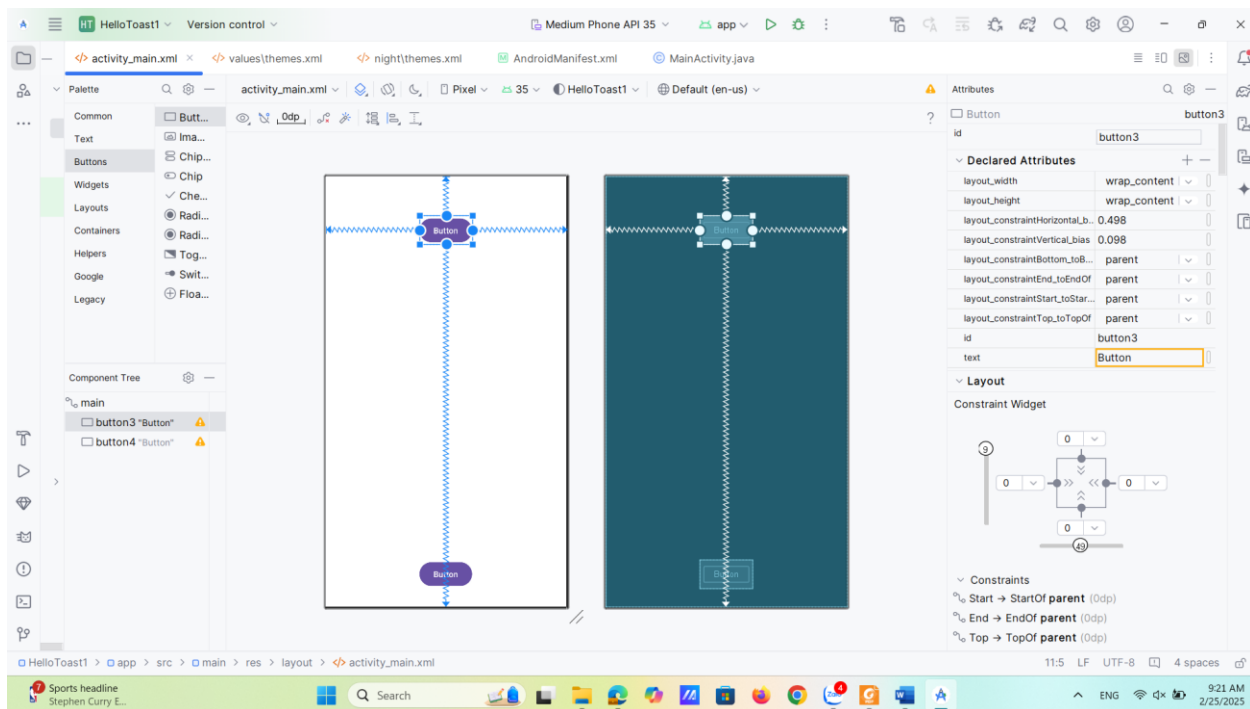


3. Nhấp vào **bộ điều khiển chiều rộng** hai lần — lần nhấp đầu tiên thay đổi thành **Fixed** với các đường thẳng, và lần nhấp thứ hai thay đổi thành **Match Constraints** với các cuộn lò xo, như được minh họa trong hình động bên dưới.



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.

4. 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.

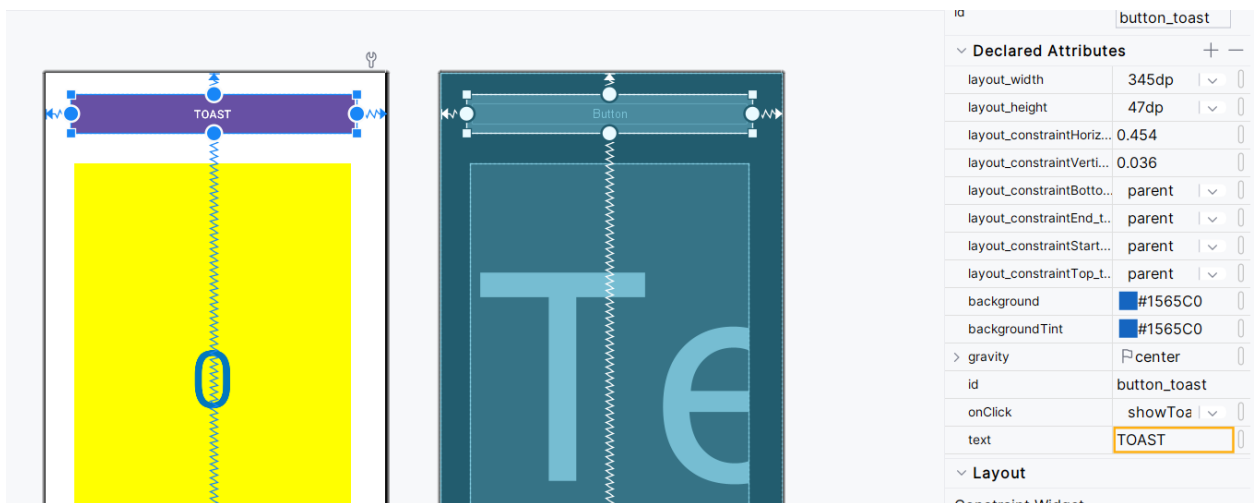
3.2 Thay đổi các thuộc tính của Button

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**.



5. 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

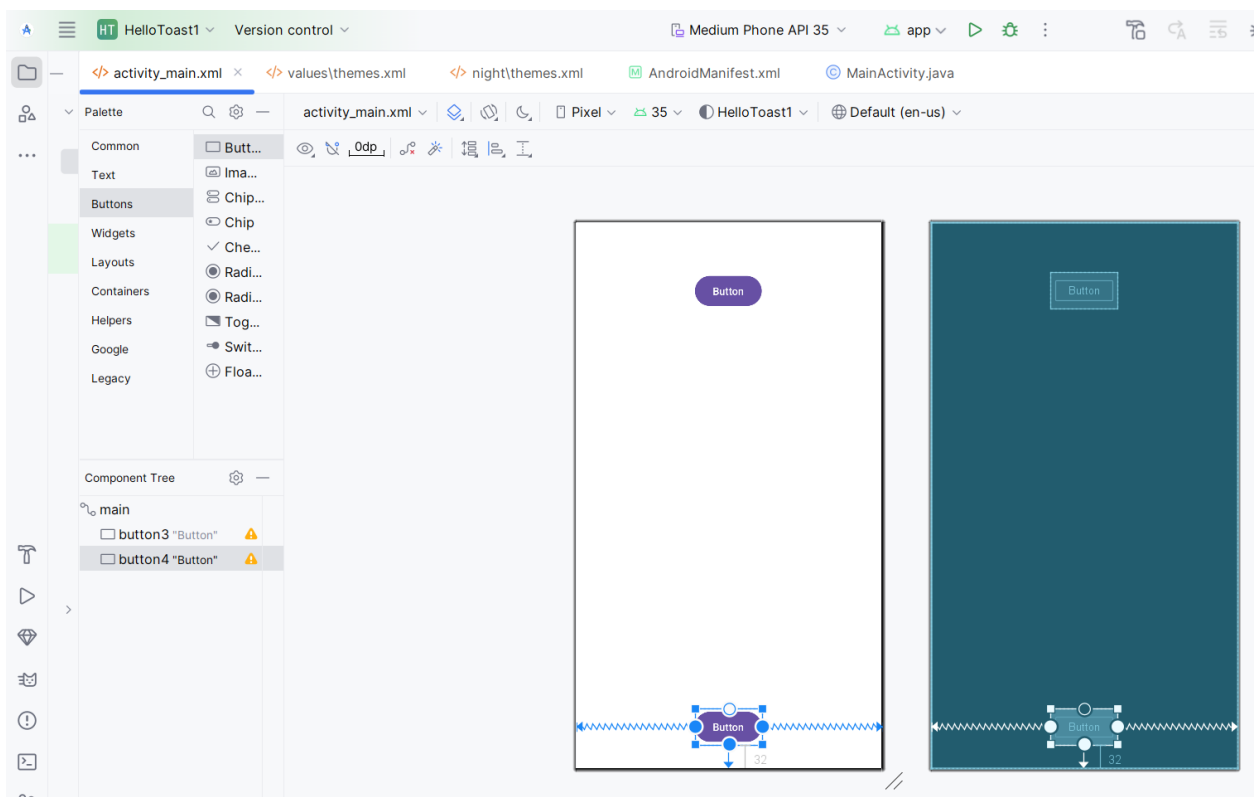
dụng (app bar). Việc sử dụng các màu cơ sở cho các phần tử UI khác sẽ tạo ra giao diện người dùng (UI) thống nhất. Bạn sẽ tìm hiểu thêm về theme ứng dụng và Material Design trong một bài học khác.

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**.

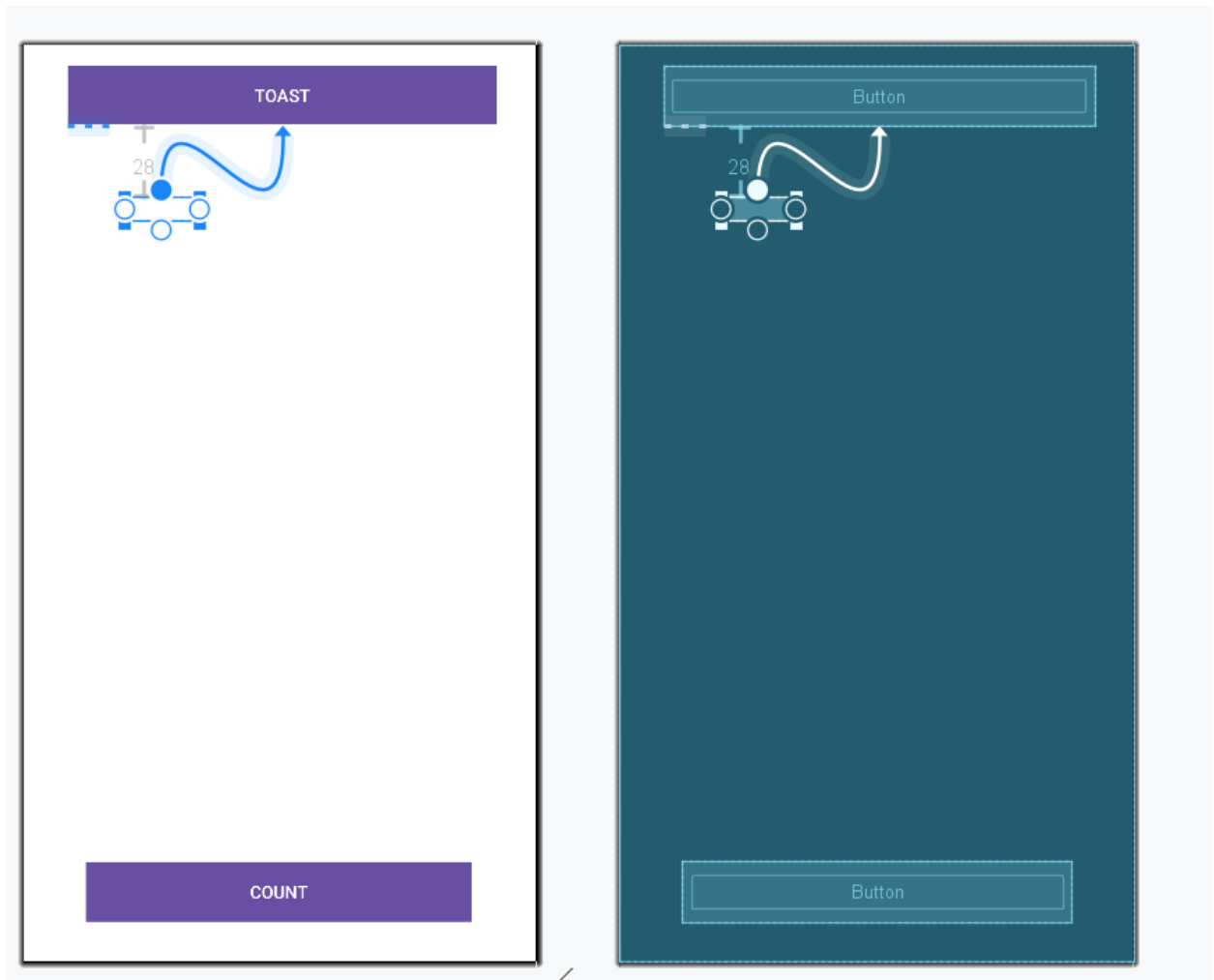
4.1 Thêm một **TextView** và thiết lập ràng buộc

1. 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**.



2. 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.

Task 5: Chỉnh sửa bố cục trong XML

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.

Android Studio interface showing the XML layout file `activity_main.xml` for the `HelloToast1` application. The file is located at `app > src > main > res > layout > activity_main.xml`.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="#FFFFFF"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button_toast"
        android:layout_width="345dp"
        android:layout_height="47dp"
        android:background="#1565C0"
        android:backgroundTint="#1565C0"
        android:gravity="center"
        android:text="TOAST"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.534"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.023" />

    <Button
        android:id="@+id/button_count"
        android:layout_width="311dp"
        android:layout_height="8dp"
        android:layout_marginBottom="32dp"
        android:background="#1565C0"
        android:backgroundTint="#1565C0"
        android:text="COUNT"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent" />

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

The status bar at the bottom indicates the file is named `activity_main.xml`, the encoding is UTF-8, and there are 4 spaces used for indentation.

```

38
39     <TextView
40         android:id="@+id/show_count"
41         android:layout_width="332dp"
42         android:layout_height="523dp"
43         android:layout_marginStart="8dp"
44         android:layout_marginTop="8dp"
45         android:layout_marginEnd="8dp"
46         android:background="#FFFF00"
47         android:gravity="center"
48         android:text="0"
49         android:textColor="#0277BD"
50         android:textSize="96sp"
51         app:layout_constraintBottom_toBottomOf="parent"
52         app:layout_constraintEnd_toEndOf="parent"
53         app:layout_constraintHorizontal_bias="0.492"
54         app:layout_constraintStart_toStartOf="parent"
55         app:layout_constraintTop_toTopOf="parent"
56         app:layout_constraintVertical_bias="0.385" />
57
58 </androidx.constraintlayout.widget.ConstraintLayout>
59

```

5.2 Tách các chuỗi thành tài nguyên

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">HelloToast1</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
</resources>
```

7. Bạn cần một chuỗi khác để sử dụng trong một nhiệm vụ 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">HelloToast1</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.

Nhiệm vụ 6: Thêm xử lý sự kiện onClick cho các nút

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/button_toast"
    android:layout_width="345dp"
    android:layout_height="47dp"
    android:background="#1565C0"
    android:backgroundTint="#1565C0"
    android:gravity="center"
    android:text="TOAST"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.454"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.036"
/>

```

2. Thêm thuộc tính `android:onClick` vào cuối phần tử `button_toast`, sau thuộc tính cuối cùng và trước dấu kết thúc `</>`

```

    android:onClick="showToast"/>

```

3. 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.
4. 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"?>
<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:background="#FFFFFF"
    tools:context=".MainActivity">

```

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="#1565C0"
    android:backgroundTint="#1565C0"
    android:gravity="center"
    android:text="TOAST"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.454"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.036"
    android:onClick="showToast"/>
```

```
<Button
    android:id="@+id/button_count"
    android:layout_width="358dp"
    android:layout_height="50dp"
    android:layout_marginBottom="48dp"
    android:background="#1565C0"
    android:backgroundTint="#1565C0"
    android:onClick="countUp"
    android:text="COUNT"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.534"
    app:layout_constraintStart_toStartOf="parent" />
```

```
<TextView
    android:id="@+id/show_count"
    android:layout_width="336dp"
    android:layout_height="513dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
```

```
android:background="#FFFF00"
android:gravity="center"
android:text="0"
android:textColor="#0277BD"
android:textSize="96sp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.45"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.475" />
```

</androidx.constraintlayout.widget.ConstraintLayout>

5. 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.


```

package com.example.hellotoast1;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            actionBar.setBackgroundDrawable(new ColorDrawable(Color.BLUE));
        }
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }

    1 usage
    public void showToast(View view) {
    }

    1 usage
    public void countUp(View view) {
    }
}

```

6.2 Chỉnh sửa xử lý sự kiện của nút Toast

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`.

```

public void showToast(View view) {
}

```

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

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

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

3. Cung cấp context của Activity của ứng dụng. Vì 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.

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

4. 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`.

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

5. Cung cấp thời lượng hiển thị. Ví dụ, `Toast.LENGTH_SHORT` hiển thị Toast trong một khoảng thời gian tương đối ngắn.

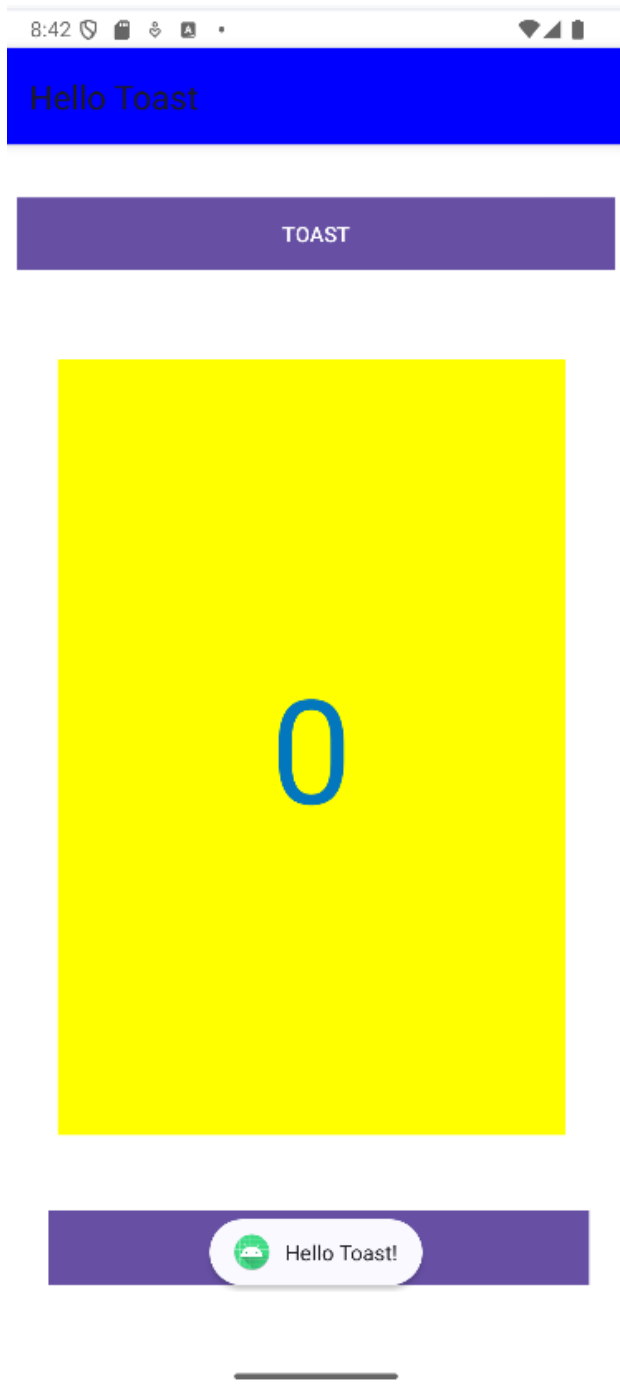
```
    Toast toast = Toast.makeText(this,  
    R.string.toast_message,Toast.LENGTH_SHORT);
```

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.

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

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

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



6.3 Chỉnh sửa xử lý sự kiện của nút Count

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:

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

2. 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):

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

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

3. 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):
4. Thay đổi câu lệnh khai báo biến thành viên thành khởi tạo giá trị bằng 0:

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

5. 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`:

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

6. 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`:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable(this);  
    ActionBar actionBar = getSupportActionBar();  
    if (actionBar != null) {  
        actionBar.setBackgroundDrawable(new ColorDrawable(Color.BLUE));  
    }  
    setContentView(R.layout.activity_main);  
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),  
(v, insets) -> {  
        Insets systemBars =  
insets.getInsets(WindowInsetsCompat.Type.systemBars());  
        v.setPadding(systemBars.left, systemBars.top, systemBars.right,  
systemBars.bottom);  
        return insets;  
    }));  
}
```

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

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable(this);  
    ActionBar actionBar = getSupportActionBar();  
    if (actionBar != null) {
```

```

        actionBar.setBackgroundDrawable(new ColorDrawable(Color.BLUE));
    }
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
        Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
        return insets;
    });
    mShowCount = (TextView) findViewById(R.id.show_count);
}

```

Một View, giống như một chuỗi, là một tài nguyên có thể có một ID. 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).

- 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 view) {
    mCount++;
    if (mShowCount != null)
        mShowCount.setText(Integer.toString(mCount));
}

```

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

8:37



Hello Toast

TOAST

1

COUNT

Mẹo: Để tìm hiểu chi tiết về cách sử dụng `ConstraintLayout`, hãy xem Codelab "[Using ConstraintLayout to design your views](#)".

