

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



GIÁO TRÌNH

THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1. Làm quen	3
Bài 1) Tạo ứng dụng đầu tiên	3
1.1) Android Studio và Hello World.....	3
1.2) Giao diện người dùng tương tác đầu tiên	25
1.3) Trình chỉnh sửa bố cục	81
1.4) Văn bản và các chế độ cuộn	82
1.5) Tài nguyên có sẵn	82
Bài 2) Activities.....	82
2.1) Activity và Intent.....	82
2.2) Vòng đời của Activity và trạng thái	82
2.3) Intent ngầm định	82
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	82
3.1) Trình gỡ lỗi.....	82
3.2) Kiểm thử đơn vị	82
3.3) Thư viện hỗ trợ	82
CHƯƠNG 2. Trải nghiệm người dùng.....	83
Bài 1) Tương tác người dùng	83
1.1) Hình ảnh có thể chọn.....	83
1.2) Các điều khiển nhập liệu	83
1.3) Menu và bộ chọn	83
1.4) Điều hướng người dùng	83
1.5) RecyclerView	83
Bài 2) Trải nghiệm người dùng thú vị.....	83
2.1) Hình vẽ, định kiểu và chủ đề	83
2.2) Thẻ và màu sắc	83
2.3) Bố cục thích ứng.....	83
Bài 3) Kiểm thử giao diện người dùng	83

3.1) Espresso cho việc kiểm tra UI	83
CHƯƠNG 3. Làm việc trong nền	83
Bài 1) Các tác vụ nền.....	83
1.1) AsyncTask.....	83
1.2) AsyncTask và AsyncTaskLoader	83
1.3) Broadcast receivers	83
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	83
2.1) Thông báo.....	83
2.2) Trình quản lý cảnh báo	83
2.3) JobScheduler	83
CHƯƠNG 4. Lưu dữ liệu người dùng	84
Bài 1) Tùy chọn và cài đặt.....	84
1.1) Shared preferences	84
1.2) Cài đặt ứng dụng	84
Bài 2) Lưu trữ dữ liệu với Room	84
2.1) Room, LiveData và ViewModel.....	84
2.2) Room, LiveData và ViewModel.....	84
3.1) Trình gowx lỗi	

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

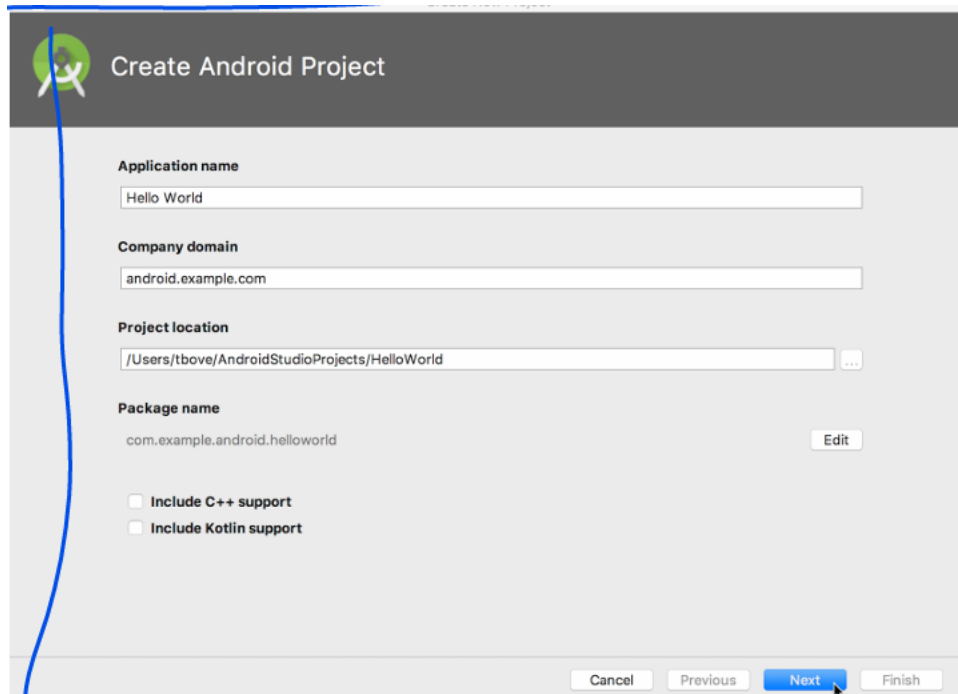
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

Sau khi cài đặt Android Studio thành công, bạn sẽ tạo, từ một mẫu, một dự án mới cho ứng dụng Hello World. Ứng dụng đơn giản này hiển thị chuỗi “Hello World” trên màn hình của thiết bị Android ảo hoặc vật lý.

Sau đây là giao diện của ứng dụng đã hoàn thành:

7:40



Hello World!

Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp (IDE) hoàn chỉnh bao gồm một trình soạn thảo mã nâng cao và một bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ để phát triển, gỡ lỗi, thử nghiệm và hiệu suất giúp phát triển ứng dụng nhanh hơn và dễ dàng hơn. Bạn có thể thử nghiệm ứng dụng của mình bằng nhiều trình giả lập được cấu hình sẵn hoặc trên thiết bị di động của riêng bạn, xây dựng các ứng dụng sản xuất và phát hành trên 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 Android Studio.

Android Studio có sẵn cho máy tính chạy Windows hoặc Linux và cho máy Mac chạy macOS. OpenJDK (Java Development Kit) mới nhất được đóng gói cùng với Android Studio.

Để bắt đầu và chạy Android Studio, trước tiên hãy kiểm tra các yêu cầu hệ thống để đảm bảo rằng hệ thống của bạn đáp ứng các yêu cầu đó. Quá trình cài đặt tương tự nhau cho tất cả các nền tảng. Mọi khác biệt đều được ghi chú bên dưới.

1. Điều hướng đến trang web dành cho nhà phát triển Android và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.

2. Chấp nhận cấu hình mặc định cho tất cả các bước và đảm bảo rằng tất cả các thành phần đều được chọn để cài đặt.

3. Sau khi hoàn tất quá trình cài đặt, Trình hướng dẫn thiết lập sẽ tải xuống và cài đặt một số thành phần bổ sung bao gồm Android SDK. Hãy kiên nhẫn, quá trình này có thể mất một thời gian tùy thuộc vào tốc độ Internet của bạn và một số bước có vẻ thừa thãi.

4. Khi quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng tạo dự án đầu tiên của mình.

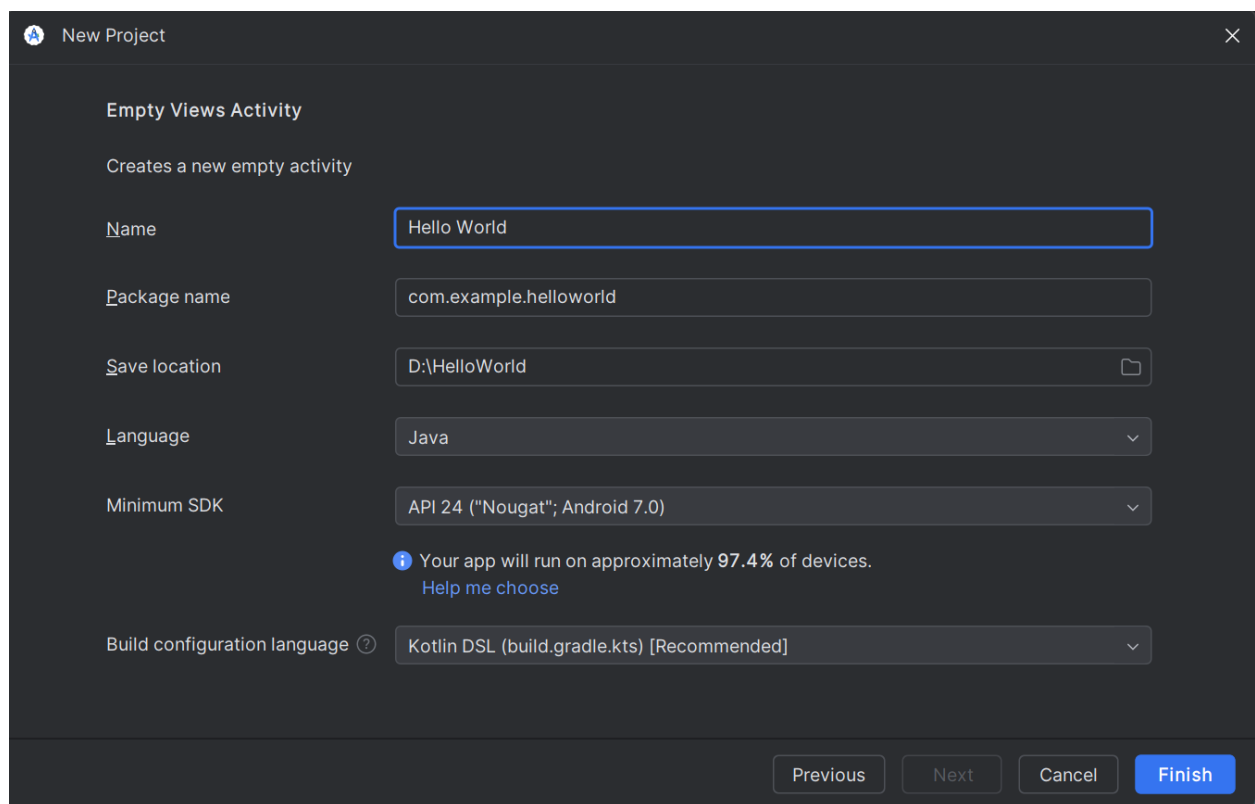
Khắc phục sự cố: Nếu bạn gặp sự cố với quá trình cài đặt, hãy kiểm tra ghi chú phát hành Android Studio hoặc nhờ người hướng dẫn trợ giúp.

Nhiệm vụ 2: Tạo ứng dụng Hello World

Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị "Hello World" để xác minh rằng Android Studio đã được cài đặt đúng cách và để tìm hiểu những điều cơ bản về phát triển với Android Studio.

2.1 Tạo dự án ứng dụng

1. Mở Android Studio nếu chưa 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



4. Xác minh rằng Project location 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 là công việc bổ sung.

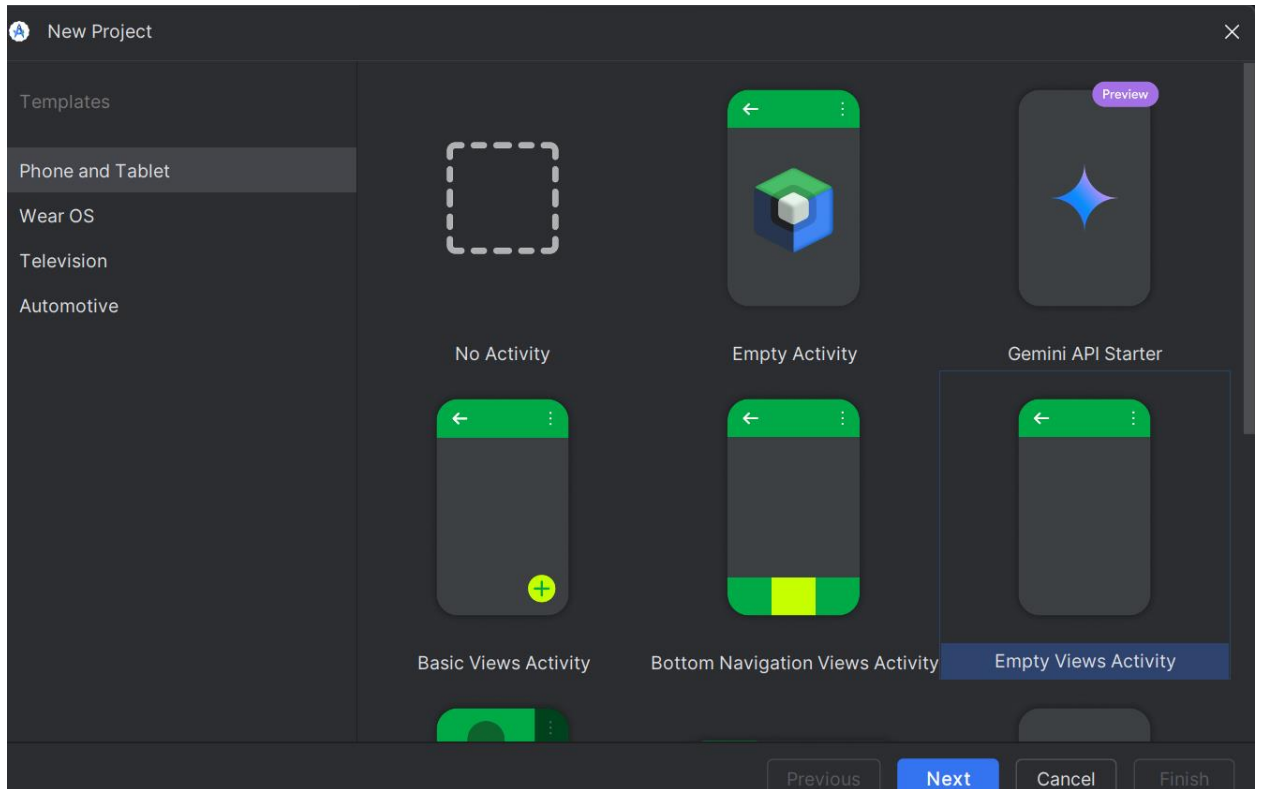
6. Bỏ chọn các tùy chọn Bao gồm hỗ trợ C++ và Bao gồm hỗ trợ Kotlin, rồi nhấp vào Next.

7. Trên màn hình Thiết bị Android đích, Điện thoại và Máy tính bảng 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 Bao gồm hỗ trợ ứng dụng tức thì và tất cả các tùy chọn khác. Sau đó, nhấp vào Tiếp theo. 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 Thêm hoạt động xuất hiện. Activity 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. Activity 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, Activity trống 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).

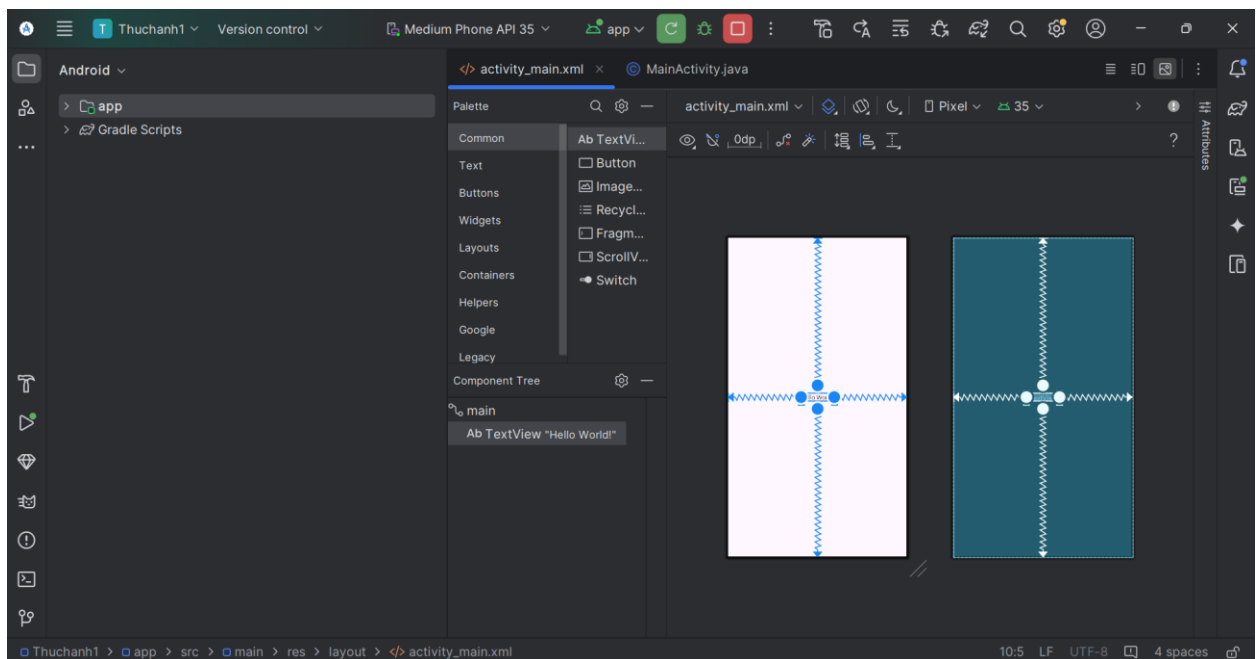
Mẹo: Xem trang Cấu hình nhà phát triển bản dựng của bạn để biết thông tin chi tiết.

Bạn cũng có thể thấy thông báo "Mẹo trong ngày" với các phím tắt và các mẹo hữu ích khác. Nhấp vào

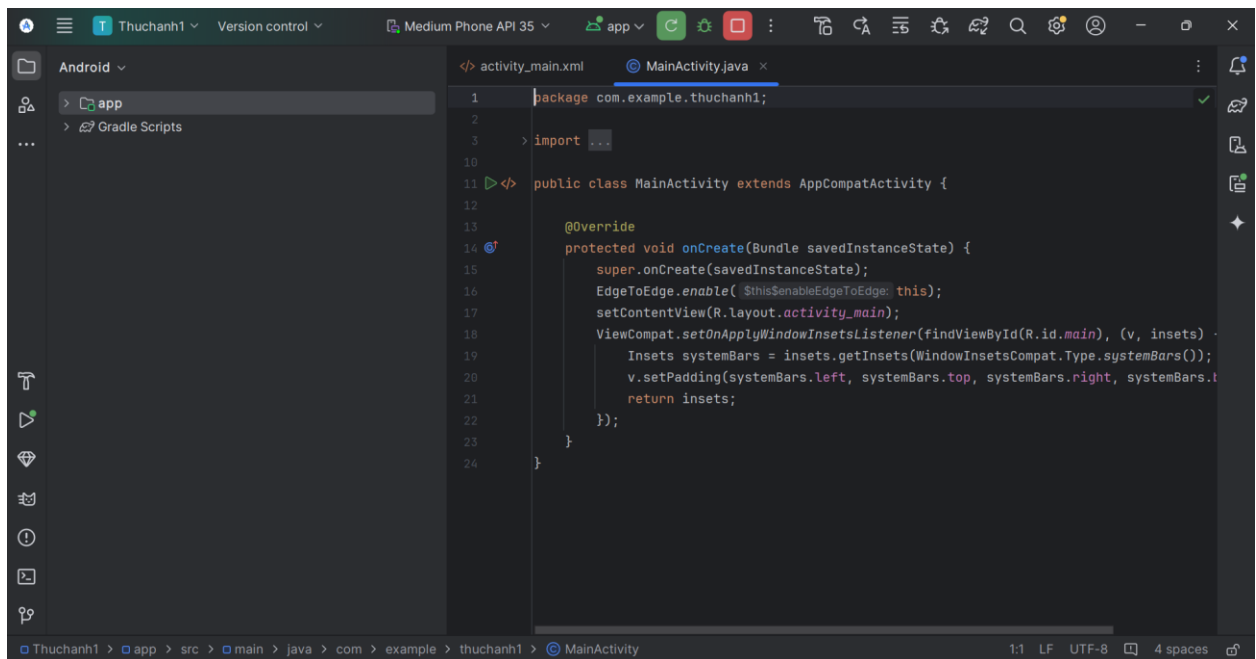
Đóng để đóng thông báo.

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 Thiết kế của trình chỉnh sửa bố cục, nếu chưa chọn, để hiển thị bản trình bày đồ họa của bố cục như được hiển thị bên dưới.



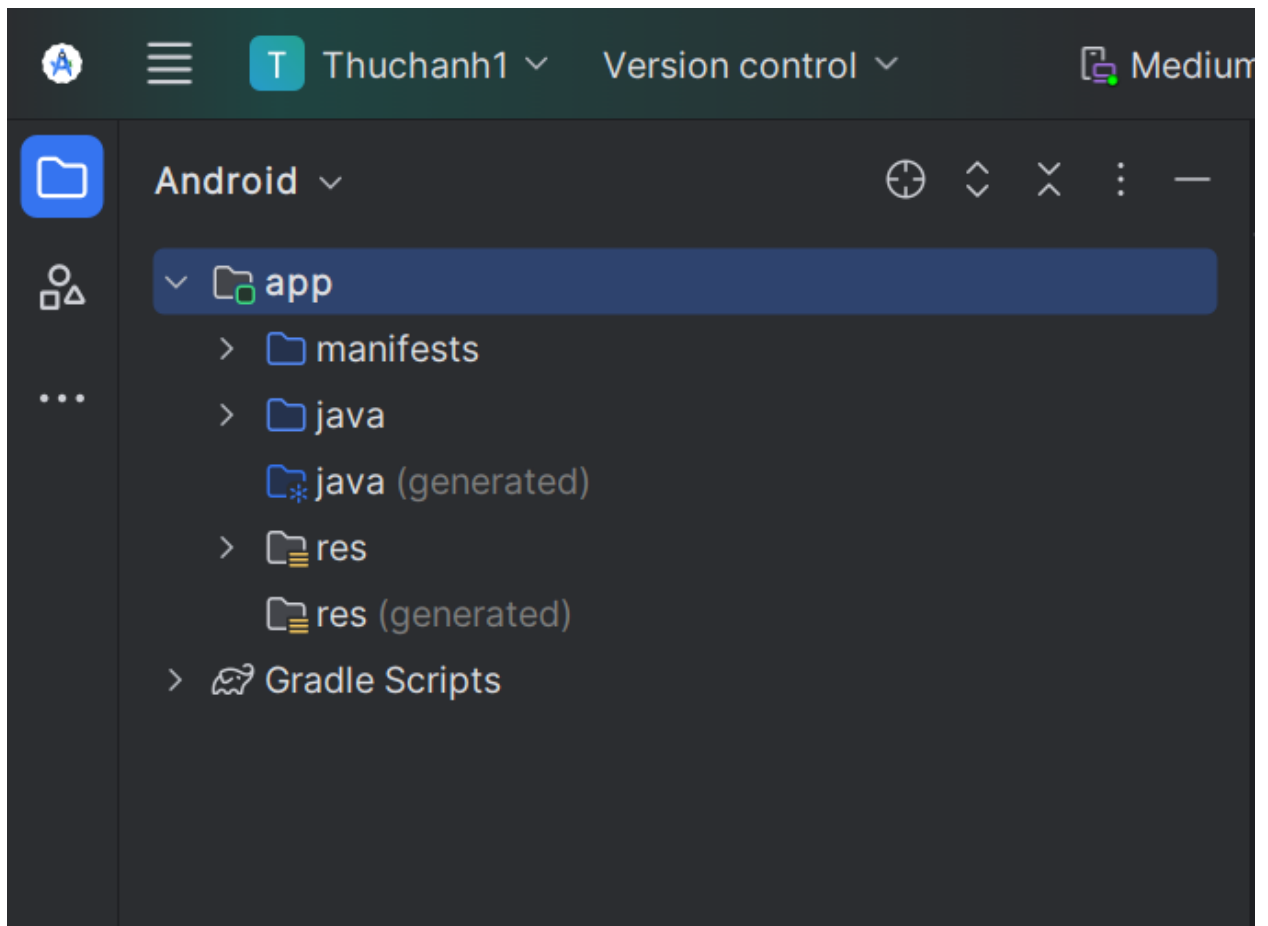
3. Nhấp vào tab `MainActivity.java` để xem trình soạn thảo mã như được hiển thị bên dưới



2.2 Khám phá ngăn Project > Android

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

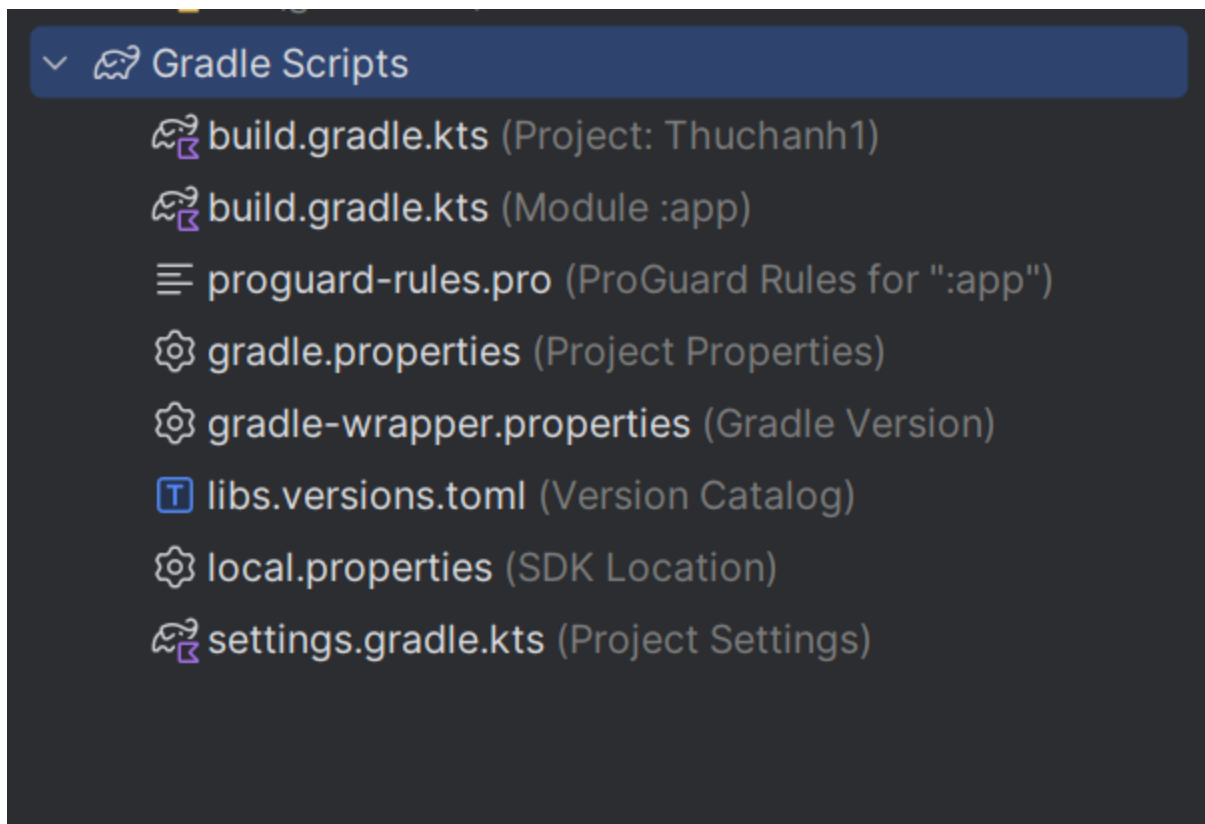
1. Nếu chưa chọn, hãy nhấp vào tab Project trong cột tab dọc ở bên trái của cửa sổ Android Studio. Ngăn Project sẽ xuất hiện.
2. Để xem dự án trong hệ thống phân cấp dự án Android chuẩn, hãy chọn Android từ menu bật lên ở đầu ngăn Project, như được hiển thị bên dưới.



2.3 Khám phá thư mục Gradle Scripts

Hệ thống dựng Gradle trong Android Studio giúp bạn dễ dàng đưa các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác vào bản dựng của mình dưới dạng phụ thuộc.

Khi bạn tạo dự án ứng dụng lần đầu tiên, ngăn Project > Android sẽ xuất hiện với thư mục Gradle Scripts được mở rộng như được hiển thị bên dưới.



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

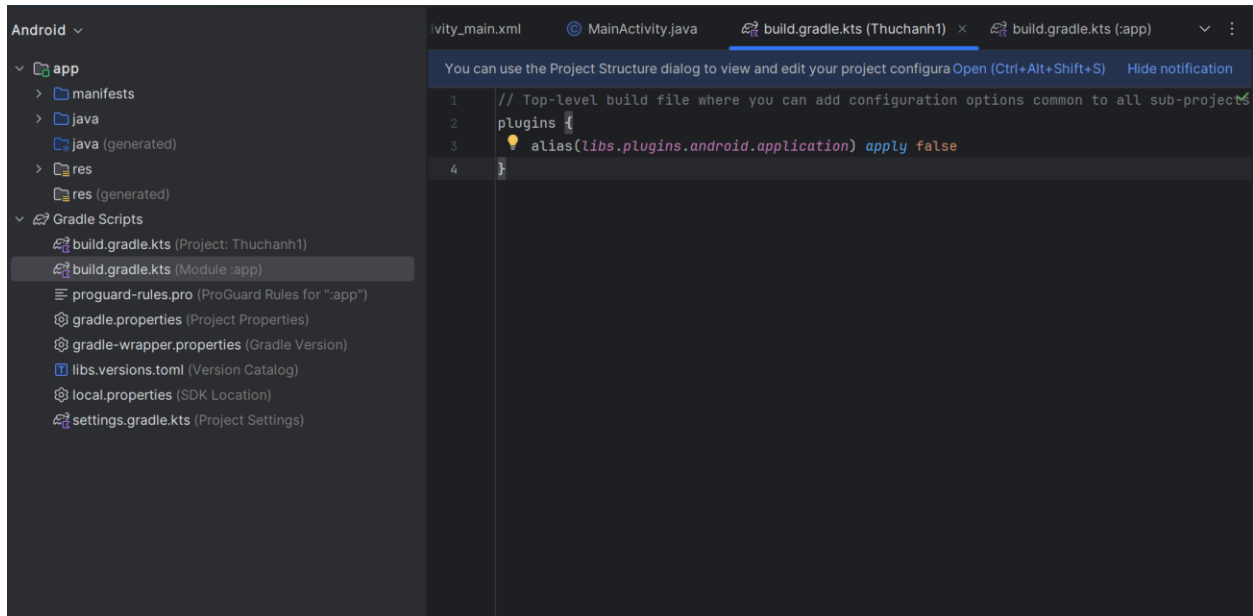
1. Nếu thư mục Gradle Scripts chưa được mở rộng, hãy nhấp vào hình tam giác để mở rộng thư mục.

Thư mục này chứa tất cả các tệp mà hệ thống dựng cần.

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 Android Studio đều chứa một tệp dựng Gradle cấp cao nhất. Hầu hết thời gian, bạn sẽ không cần thực hiện bất kỳ thay đổi nào đối với tệp này, nhưng vẫn hữu ích khi hiểu nội dung của tệp.

Theo mặc định, tệp dựng cấp cao nhất sử dụng khối buildscript để xác định các kho lưu trữ Gradle và các phụ thuộc chung cho tất cả các mô-đun trong dự án. Khi phụ thuộc của bạn là thứ gì đó khác với thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm kiếm các tệp trong bất kỳ kho lưu trữ trực tuyến nào được chỉ định trong khối kho lưu trữ của tệp này. Theo mặc định, các dự án Android Studio mới khai báo JCenter và Google (bao gồm kho lưu trữ Google Maven) là các vị trí kho lưu trữ:



3. Tìm tệp build.gradle(Module:app).

Ngoài tệp build.gradle cấp dự án, mỗi mô-đun đều có tệp build.gradle riêng, cho phép bạn định cấu hình cài đặt bản dựng cho từng mô-đun cụ thể (ứng dụng HelloWorld chỉ có một mô-đun). Định 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 hương vị sản phẩm. Bạn cũng có thể ghi đè các cài đặt trong tệp AndroidManifest.xml hoặc tệp build.gradle cấp cao nhất.

Tệp này thường là tệp cần chỉnh sửa khi thay đổi cấu hình cấp ứng dụng, chẳng hạn như khai báo các phụ thuộc trong phần phụ thuộc. Bạn có thể khai báo phụ thuộc thư viện bằng một trong một số cấu hình phụ thuộc khác nhau. Mỗi cấu hình phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện. Ví dụ: câu lệnh triển khai fileTree(dir: 'libs', include: ['*.jar']) thêm một phụ thuộc của tất cả các tệp ".jar" bên trong thư mục libs.

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

```

1  plugins {
2      ⚡ alias(libs.plugins.android.application)
3  }
4
5  android {
6      namespace = "com.example.thuchanh1"
7      compileSdk = 35
8
9      defaultConfig {
10         applicationId = "com.example.thuchanh1"
11         minSdk = 24
12         targetSdk = 35
13         versionCode = 1
14         versionName = "1.0"
15
16         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
17     }

```

```

19     buildTypes {
20         release {
21             isMinifyEnabled = false
22             proguardFiles(
23                 getDefaultProguardFile( name: "proguard-android-optimize.txt"),
24                 "proguard-rules.pro"
25             )
26         }
27     }
28     compileOptions {
29         sourceCompatibility = JavaVersion.VERSION_11
30         targetCompatibility = JavaVersion.VERSION_11
31     }
32 }

```

```

33  dependencies {
34
35      implementation(libs.appcompat)
36      implementation(libs.material)
37      implementation(libs.activity)
38      implementation(libs.constraintlayout)
39      testImplementation(libs.junit)
40      androidTestImplementation(libs.ext.junit)
41      androidTestImplementation(libs.espresso.core)
42  }
43

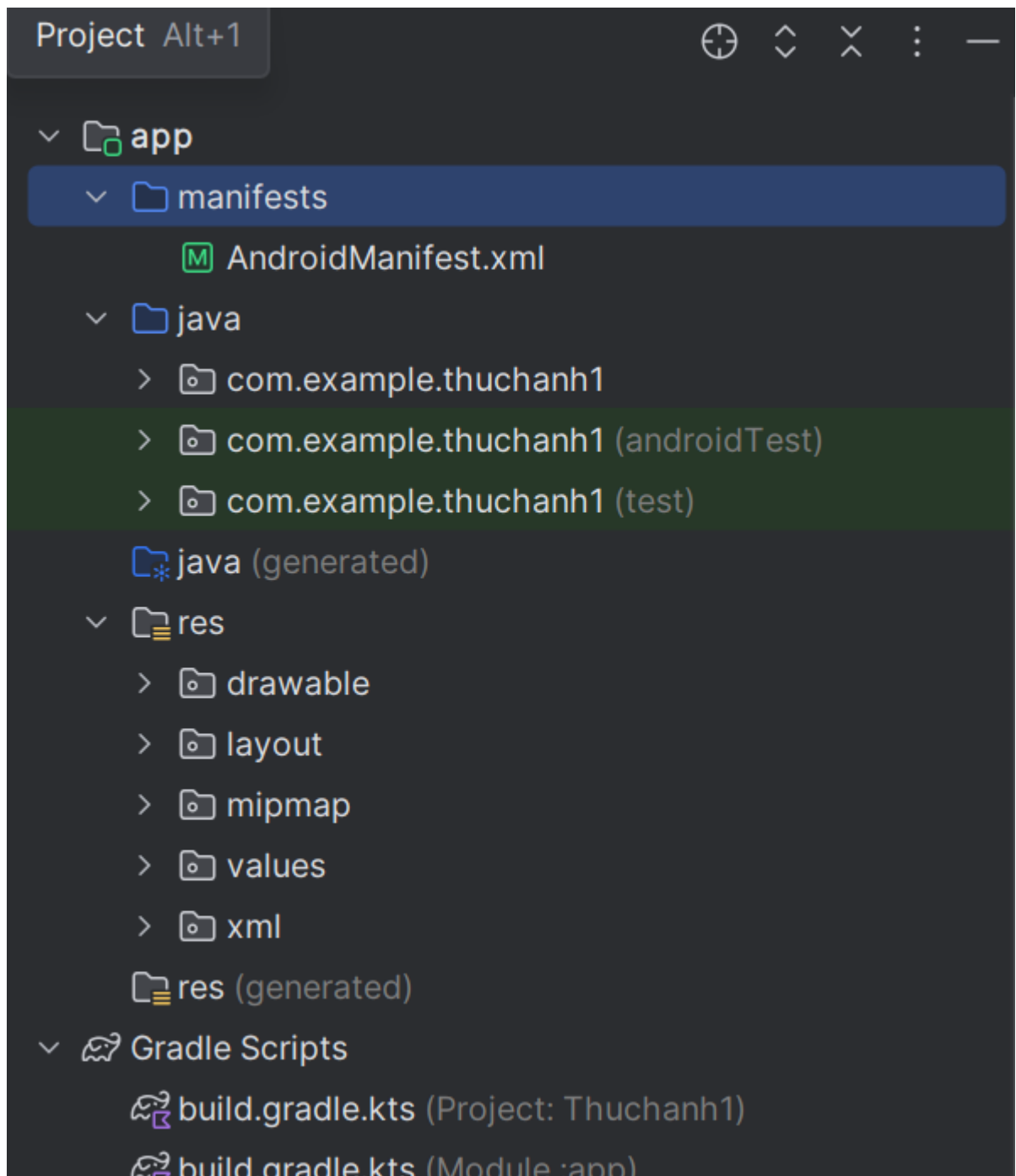
```

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

2.4 Khám phá các thư mục app và res

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

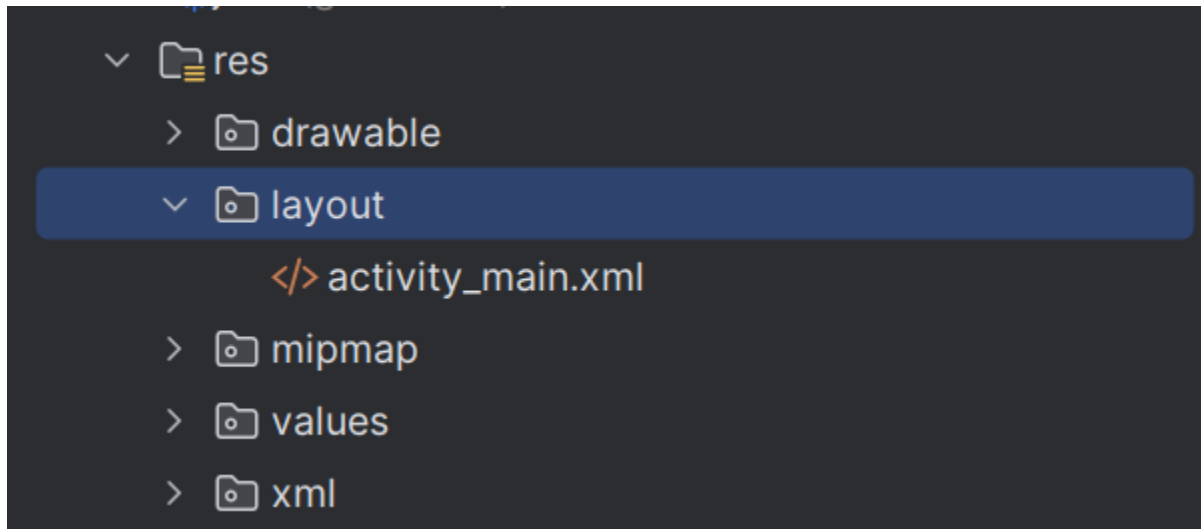
1. Mở rộng thư mục app, thư mục java và thư mục com.example.android.helloworld để xem tệp java MainActivity. Nhấp đúp vào tệp sẽ mở tệp đó trong trình chỉnh sửa mã.



Thư mục java bao gồm các tệp lớp Java trong ba thư mục con, như thể hiện trong hình trên. Thư mục com.example.hello.helloworld (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục còn lại được sử dụng để thử nghiệm và được mô tả trong một bài học khác. Đối với ứng dụng Hello World, chỉ có một gói và nó chứa MainActivity.java. Tên của Hoạt động đầu tiên (màn hình) mà người dùng nhìn thấy, cũng khởi tạo các tài nguyên trên toàn ứng dụng,

thường được gọi là MainActivity (phần mở rộng tệp bị bỏ qua trong ngăn Dự án > Android).

2. Mở rộng thư mục res và thư mục layout, rồi nhấp đúp vào tệp activity_main.xml để mở tệp đó trong trình chỉnh sửa layout.



Thư mục res chứa các tài nguyên, chẳng hạn như layout, chuỗi và hình ảnh. Activity thường được liên kết với layout của các chế độ xem UI được định nghĩa là tệp XML. Tệp này thường được đặt tên theo Activity của nó.

2.5 Khám phá thư mục manifests

Thư mục manifests chứa các tệp cung cấp thông tin cần thiết về ứng dụng của bạn cho hệ thống Android, hệ thống phải có 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. Tất cả các thành phần cho một 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 sửa đổi tệp này để thêm các tính năng và quyền tính năng. Để biết phần giới thiệu, hãy xem App Manifest tổng quan.

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

Trong nhiệm vụ này, bạn sẽ sử dụng trình quản lý Thiết bị ảo Android (AVD) để tạo một thiết bị ảo (còn được gọi là trình giả lập) mô phỏng cấu hình cho một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng Trình giả lập Android có các yêu cầu bổ sung ngoài các yêu cầu hệ thống cơ bản đối với Android Studio.

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

3.1 Tạo thiết bị ảo Android (AVD)

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

1. Trong Android Studio, chọn Tools > Android > AVD Manager hoặc nhấp vào biểu tượng Trình quản lý AVD trên thanh công cụ. Màn hình Thiết bị ảo của bạn sẽ xuất hiện. Nếu bạn đã tạo thiết bị ảo, màn hình sẽ hiển thị chúng (như trong hình bên dưới); nếu không, bạn sẽ thấy danh sách trống.

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

3. Chọn một thiết bị như Nexus 5x hoặc Pixel XL, rồi nhấp vào Next. Màn hình System Image xuất hiện

4. Nhấp vào tab Recommended nếu chưa chọn và chọn phiên bản hệ thống Android nào để chạy trên thiết bị ảo (như Oreo)

Có nhiều phiên bản hơn so với phiên bản hiển thị trong tab Recommended. Hãy xem các tab x86 Images và Other Images để xem chúng.

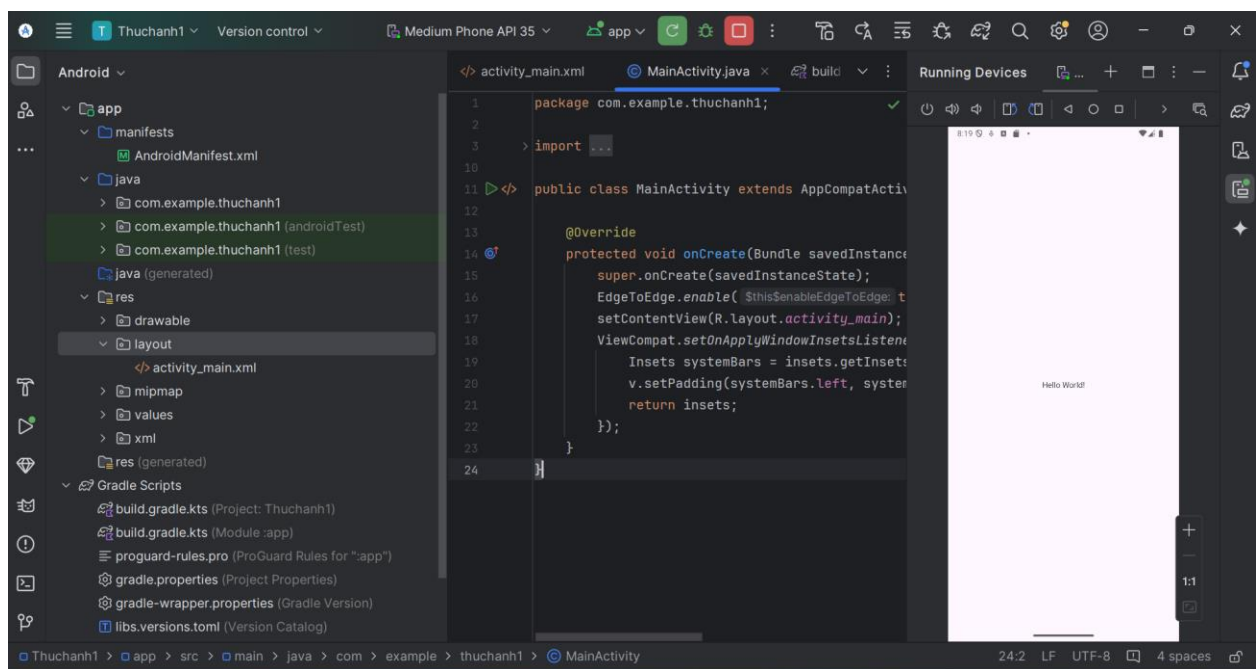
Nếu liên kết Download hiển thị bên cạnh ảnh hệ thống mà bạn muốn sử dụng, thì ảnh đó vẫn chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống và nhấp vào Finish khi hoàn tất.

5. Sau khi chọn ảnh hệ thống, hãy nhấp vào Next. Cửa sổ Android Virtual Device (AVD) xuất hiện. Bạn cũng có thể thay đổi tên của AVD. Kiểm tra cấu hình của bạn và nhấp vào Hoàn tất.

3.2 Chạy ứng dụng trên thiết bị ảo

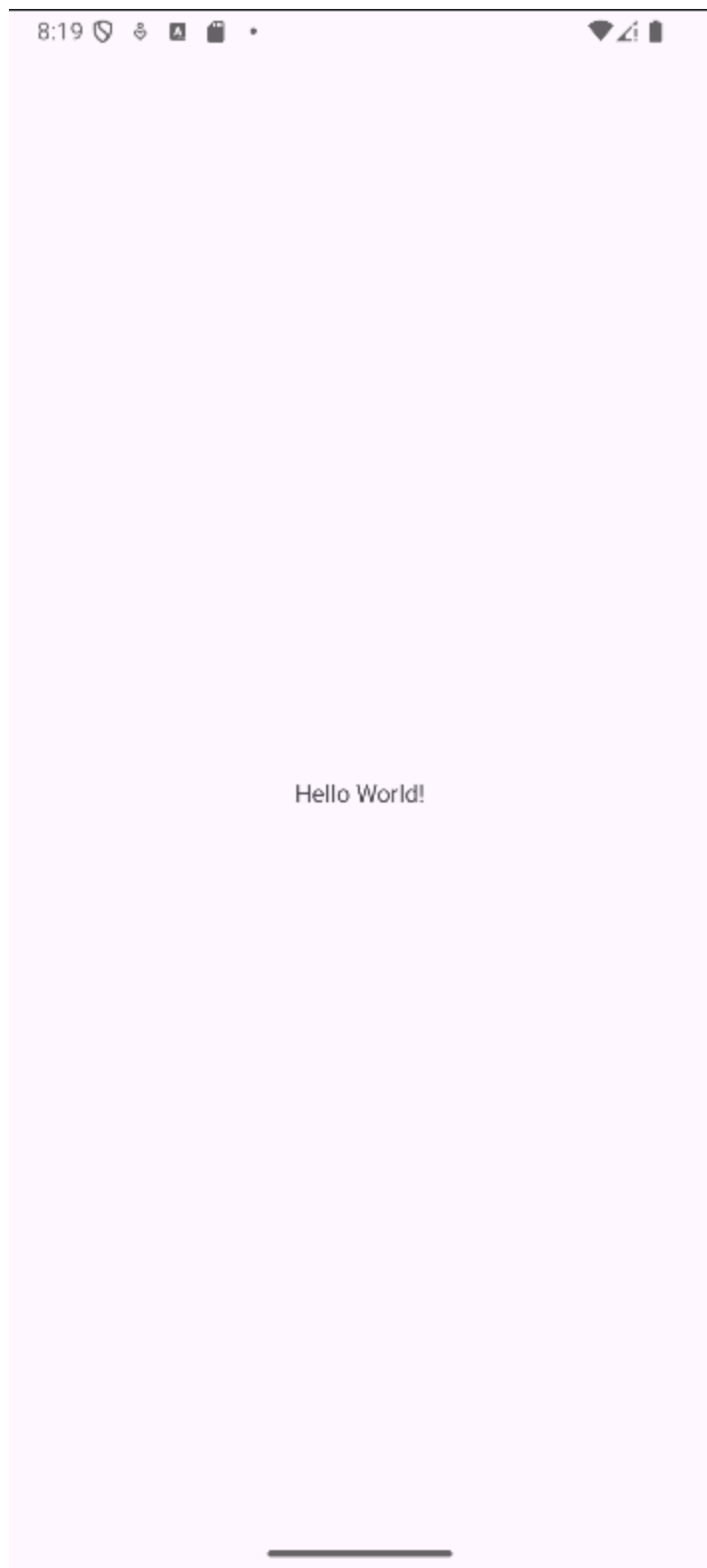
Trong tác vụ này, cuối cùng bạn sẽ chạy ứng dụng Hello World của mình.

1. Trong Android Studio, chọn Chạy > Chạy ứng dụng hoặc nhấp vào biểu tượng Chạy trên thanh công cụ.
2. Cửa sổ Chọn mục tiêu triển khai, bên dưới Thiết bị ảo khả dụng, chọn thiết bị ảo mà bạn vừa tạo và nhấp vào OK.



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

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



Mẹo: Khi thử nghiệm trên thiết bị ảo, bạn nên khởi động thiết bị một lần, ngay khi bắt đầu phiên của mình. Bạn không nên đóng thiết bị cho đến khi hoàn tất thử

nghiệm ứng dụng, để ứng dụng của bạn không phải trải qua quá trình khởi động thiết bị một lần nữa. Để đóng thiết bị ảo, hãy nhấp vào nút X ở đầu trình giả lập, chọn Thoát từ menu hoặc nhấn Control-Q trong Windows hoặc Command-Q trong macOS.

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

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

Những gì bạn cần:

- Thiết bị Android như điện thoại hoặc máy tính bảng.
- Cáp dữ liệu để kết nối thiết bị Android của bạn với máy tính qua cổng USB.
- Nếu bạn đang sử dụng hệ thống Linux hoặc Windows, bạn có thể cần thực hiện các bước bổ sung để chạy trên thiết bị phần cứng. Kiểm tra tài liệu Sử dụng thiết bị phần cứng. Bạn cũng có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Đối với trình điều khiển USB chạy trên Windows, hãy xem Trình điều khiển USB OEM.

4.1 Bật gỡ lỗi USB

Để Android Studio giao tiếp với thiết bị của bạn, bạn phải bật Gỡ lỗi USB trên thiết bị Android của mình. Tính năng này được bật trong cài đặt Developer options của thiết bị.

Trên Android 4.2 trở lên, màn hình Developer options bị ẩn theo mặc định. Để hiển thị tùy chọn nhà phát triển và bật Gỡ lỗi USB:

1. Trên thiết bị của bạn, hãy mở Settings, tìm kiếm About phone, nhấp vào About phone và chạm vào Build bảy lần.

2. Quay lại màn hình trước đó (Cài đặt / Hệ thống). Developer options xuất hiện trong danh sách.

Chạm vào Developer options.

3. Chọn Gỡ lỗi USB.

4.2 Chạy ứng dụng của bạn trên thiết bị

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

1. Kết nối thiết bị của bạn với máy phát triển bằng cáp USB.
2. Nhấp vào nút Chạy trên thanh công cụ. Cửa sổ Chọn Mục tiêu Triển khai mở ra với danh sách các trình giả lập và thiết bị được kết nối khả dụng.
3. Chọn thiết bị của bạn và nhấp vào OK.

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

Xử lý sự cố

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

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

Nếu máy tính của bạn vẫn không tìm thấy thiết bị hoặc tuyên bố thiết bị "không được phép", hãy làm theo các bước sau:

1. Rút phích cắm thiết bị.
2. Trên thiết bị, hãy mở Tùy chọn nhà phát triển trong ứng dụng Cài đặt.
3. Nhấn vào Thu hồi quyền gỡ lỗi USB.
4. Kết nối lại thiết bị với máy tính của bạn.
5. Khi được nhắc, hãy cấp quyền.

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

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

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

5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Thực hiện theo các bước sau:

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

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

2. Trong khối defaultConfig, hãy thay đổi giá trị của minSdkVersion thành 17 như hiển thị bên dưới (ban đầu được đặt thành 15).

Trình chỉnh sửa mã sẽ hiển thị thanh thông báo ở trên cùng với liên kết Đồng bộ hóa ngay.

5.2 Đồng bộ hóa cấu hình Gradle mới

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

Để đồng bộ hóa các tệp dự án, hãy nhấp vào Đồng bộ hóa ngay trên thanh thông báo xuất hiện khi thực hiện thay đổi (như được hiển thị trong hình trước) hoặc nhấp vào biểu tượng Đồng bộ hóa dự án với tệp Gradle trên thanh công cụ.

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

Để tìm hiểu sâu hơn về Gradle, hãy xem tài liệu Tổng quan về hệ thống bản dựng và Cấu hình bản dựng Gradle

.

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện người dùng(UI) xuất hiện trên màn hình của 1 thiết bị Android bao gồm 1 hệ thống phân cấp các đối tượng được gọi là views --- mỗi thành phần trên màn hình đều là 1 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 có tính tương tác như là các nút bấm, các hộp kiểm, và các trường nhập văn bản. Các lớp con thường được sử dụng của View, 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 thành phần có thể nhấp khác (chẳng hạn như **RadioButton**, **CheckBox**, và **Spinner**) để cung cấp hành vi tương tác.
- ☐ **ScrollView** và **RecyclerView** để hiển thị danh sách có thể cuộn.
- ☐ **ImageView** để hiển thị hình ảnh.
- ☐ **ConstraintLayout** và **LinearLayout** để chứa các phần tử **View** khác và định vị chúng trên màn hình.

Mã Java điều khiển giao diện người dùng (UI) được nằm trong một lớp mở rộng từ Activity. Một Activity thường được liên kết với một bố cục UI được định nghĩa trong một tệp XML. Tệp XML này thường được đặt tên theo Activity tương ứng và xác định cách sắp xếp các phần tử View trên màn hình.

Ví dụ, trong ứng dụng Hello World, lớp MainActivity sẽ hiển thị một bố cục được định nghĩa trong tệp activity_main.xml, bao gồm một TextView có nội dung "Hello World".

Trong các ứng dụng phức tạp hơn, một Activity triển khai các hoạt động để phản hồi thao tác nhấn của người dùng, vẽ nội dung đồ họa, hoặc yêu cầu dữ liệu của 1 cơ sở dữ liệu hoặc internet. Bạn học thêm về lớp hoạt động trong 1 bài học khác.

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

Những gì bạn nên biết

Bạn nên làm quen với:

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

- Cách chạy ứng dụng HelloWorld

Những gì bạn sẽ học

- Tạo 1 ứng dụng với giao diện hành vi
- Cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục
- Cách chỉnh sửa bố cục trong XML
- Rất nhiều thuật ngữ mới. Kiểm tra bảng thuật ngữ từ vựng và khái niệm cho thân thiện định nghĩa. Cách tạo 1 ứng dụng với giao diện hành vi

Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

Ứng dụng HelloToast bao gồm hai phần tử Button và một TextView. Khi người dùng nhấn vào nút đầu tiên, ứng dụng sẽ hiển thị một thông báo ngắn (Toast) trên màn hình. Khi nhấn vào nút thứ hai, bộ đếm số lần nhấn sẽ tăng lên và hiển thị trong TextView, bắt đầu từ số 0.

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

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

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

1.1 Tạo dự án Android Studio

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

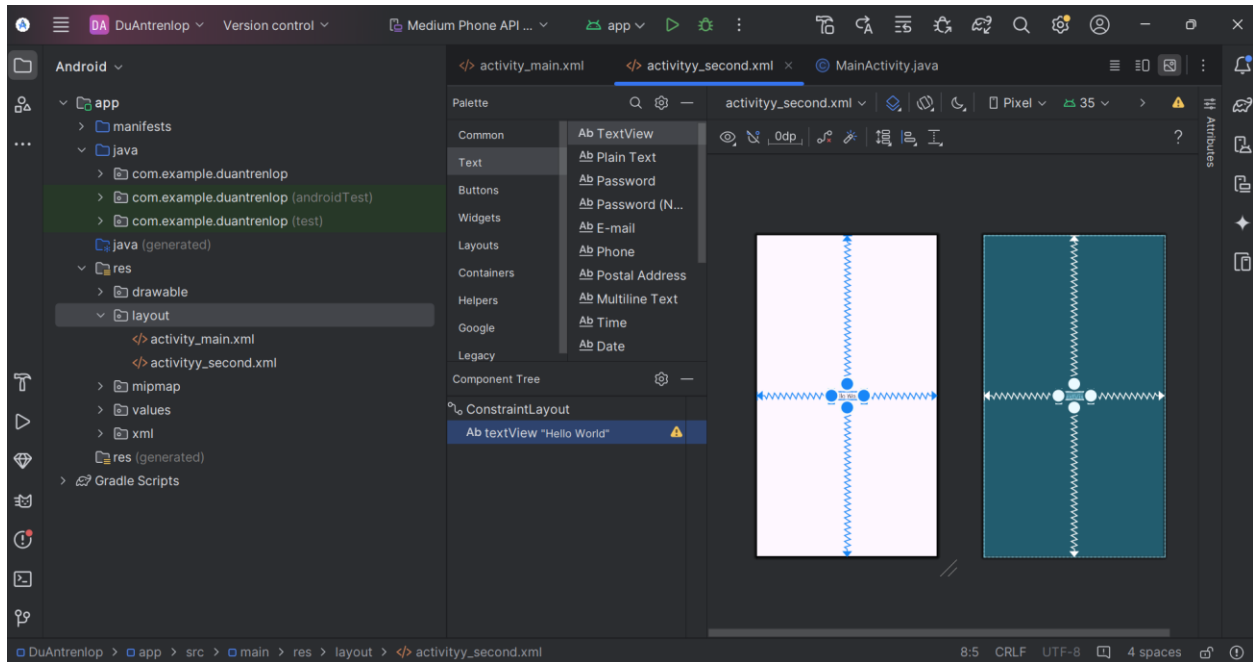
Thuộc tính	Giá trị
Tên ứng dụng	Hello Toast
Tên công ty	com.example.android (hoặc miền của bạn)
Thiết bị hỗ trợ	Điện thoại và máy tính bảng
SDK tối thiểu	API15: Android 4.0.3 IceCreamSanwich
Mẫu (Template)	Hoạt động trống
Chọn hộp "Generate Layout file"	Đã chọn
Chọn hộp "Backwards Compatibility"	Đã chọn

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

1.2 Khám phá trình chỉnh sửa bố cục

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

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




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


Mẹo: Xem tài liệu [Building a UI with Layout Editor](#) để biết thêm chi tiết về cách sử dụng trình chỉnh sửa bố cục, hoặc tham khảo [Meet Android Studio](#) để hiểu rõ hơn về Android Studio.

2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện theo các bước sau:

1. Mở `activity_main.xml` từ ngăn Project > Android nếu nó chưa mở. Nếu tab Design chưa được chọn, hãy nhấp vào tab đó.

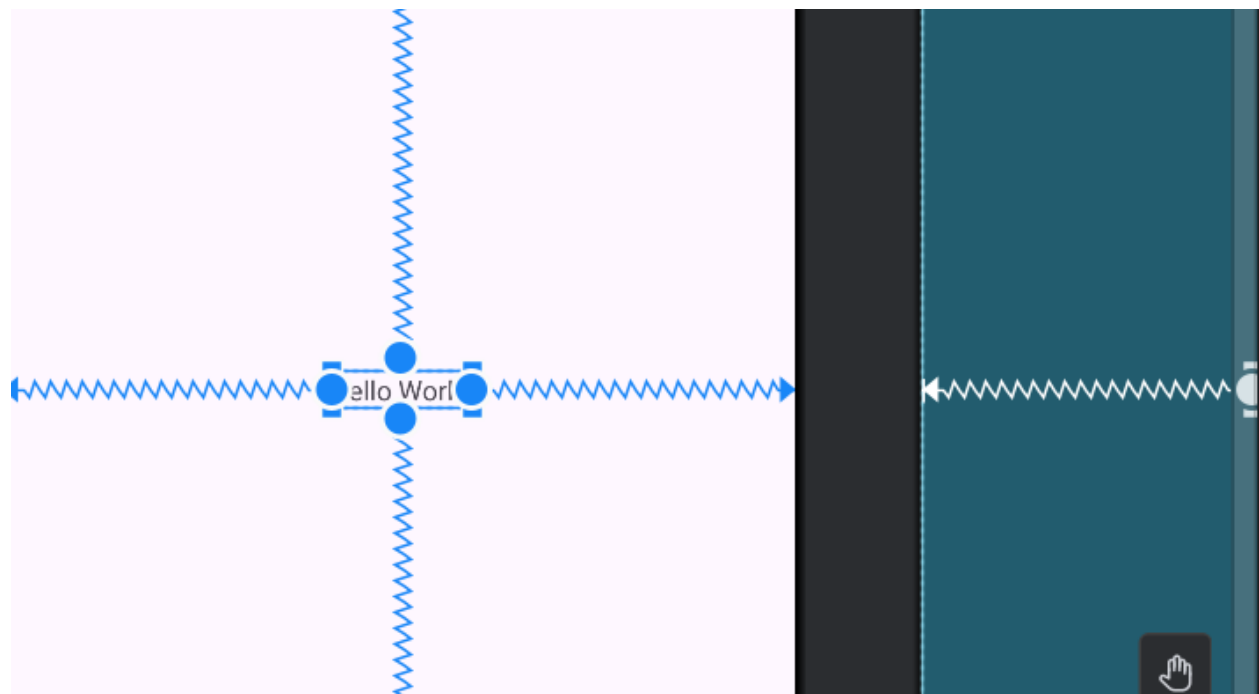
Nếu không có bản thiết kế, hãy nhấp vào nút Select Design  Surface trên thanh công cụ và chọn Design + Blueprint.

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

3. Nhấp vào nút phóng to  90%  để phóng to các ngăn thiết kế và bản thiết kế để xem cận cảnh.

4. Chọn TextView trong ngăn Component Tree. TextView "Hello World" được tô sáng trong các ngăn thiết kế và bản thiết kế và các ràng buộc cho phần tử sẽ hiển thị.

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



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

- **Constraint handle:** Để tạo ràng buộc như trong hình động ở trên, hãy nhấp vào một tay cầm ràng buộc, được hiển thị dưới dạng hình tròn ở bên cạnh một phần tử. Sau đó, kéo tay cầm đến một tay cầm ràng buộc khác hoặc đến ranh giới cha. Đường ngoằn ngoèo biểu thị ràng buộc.



- Tay cầm thay đổi kích thước: Để thay đổi kích thước phần tử, hãy kéo các tay cầm thay đổi kích thước hình vuông. Tay cầm sẽ chuyển thành góc nghiêng khi bạn kéo nó.



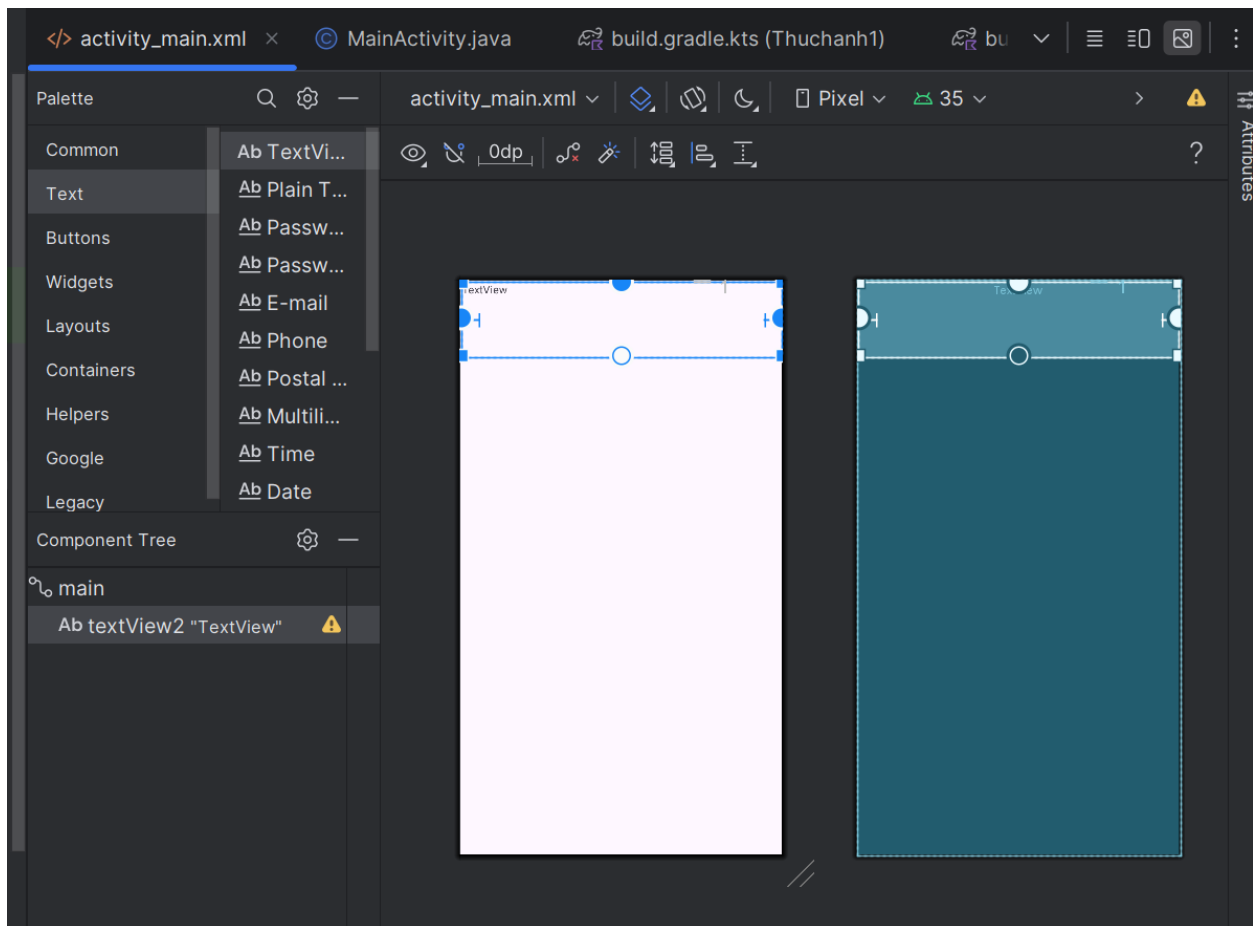
2.2 Thêm Button vào bố cục

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

Thực hiện theo các bước sau để thêm Nút:

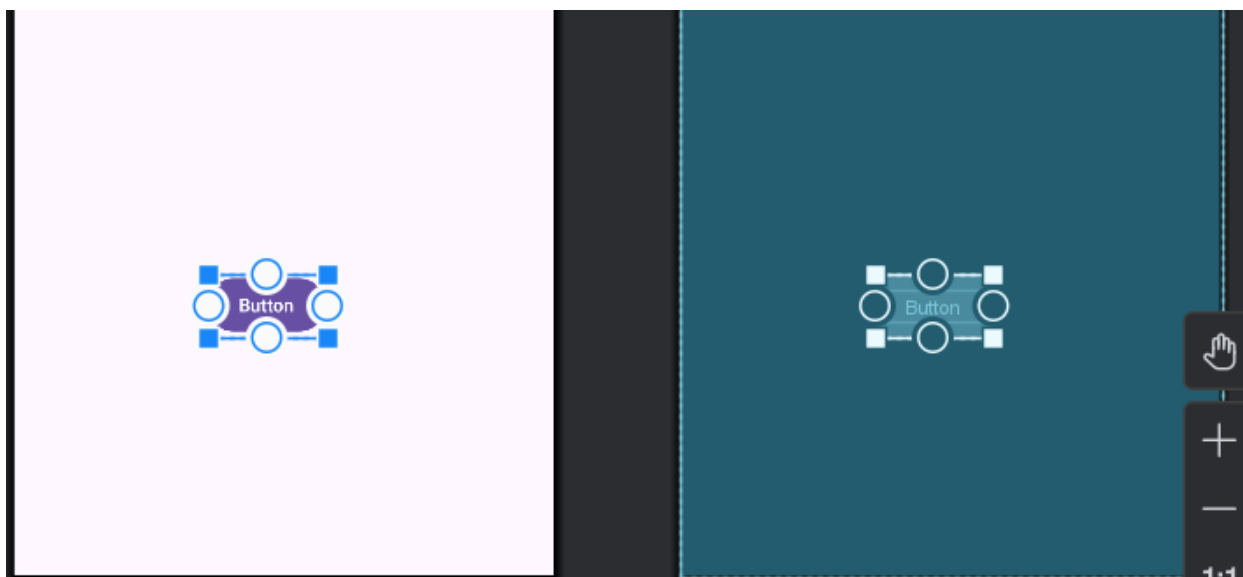
1. Bắt đầu với một bảng trắng. Thành phần TextView không cần thiết, vì vậy khi thành phần này vẫn được chọn, hãy nhấn phím Delete hoặc chọn Edit > Delete. Bây giờ bạn có một bố cục hoàn toàn trống.

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



2.3 Thêm Button thứ hai vào bố cục

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



Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di con trỏ lên trên để hiển thị nút **Clear Constraints**. Nhấp vào button này để xóa tất cả các ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc duy nhất, hãy nhấp vào trình xử lý cụ thể đặt ràng buộc đó.

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

Nhiệm vụ 3: Thay đổi các thuộc tính của phần tử 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 attributes (được gọi là properties) chung cho tất cả các chế độ xem trong View class documentation

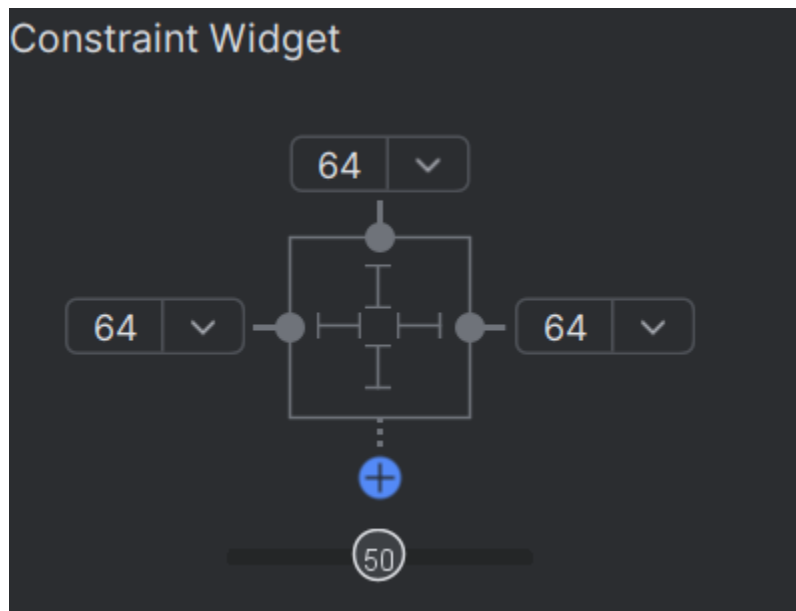
Trong nhiệm vụ này, bạn nhập các giá trị mới và thay đổi các giá trị cho các thuộc tính Nút quan trọng, có thể áp dụng cho hầu hết các loại Chế độ xem.

3.1 Thay đổi kích thước Nút

Trình chỉnh sửa bố cục cung cấp các nút điều chỉnh kích thước ở cả bốn góc của Chế độ xem để bạn có thể thay đổi kích thước Chế độ xem một cách nhanh chóng. Bạn có thể kéo các nút điều chỉnh ở mỗi góc của Chế độ xem để thay đổi kích thước, nhưng làm như vậy sẽ mã hóa cứng các kích thước

chiều rộng và chiều cao. Tránh mã hóa cứng các kích thước cho hầu hết các thành phần Chế độ xem, vì các kích thước được mã hóa cứng không thể thích ứng với các kích thước nội dung và màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn Attributes ở bên phải của trình chỉnh sửa bố cục để chọn chế độ định cỡ không sử dụng các kích thước được mã hóa cứng. Ngăn Attributes bao gồm một bảng điều khiển định cỡ hình vuông được gọi là trình kiểm tra chế độ xem ở trên cùng. Các ký hiệu bên trong hình vuông biểu thị các thiết lập chiều cao và chiều rộng như sau:



Trong hình trên:

1.Height control . Kiểm soát này chỉ định thuộc tính `layout_height` và xuất hiện trong hai phân đoạn ở phía trên và phía dưới của hình vuông. Các góc cho biết rằng kiểm soát này được đặt thành `wrap_content`, nghĩa là View sẽ mở rộng theo chiều dọc khi cần để vừa với nội dung của nó. "8" cho biết lề chuẩn được đặt thành 8dp.

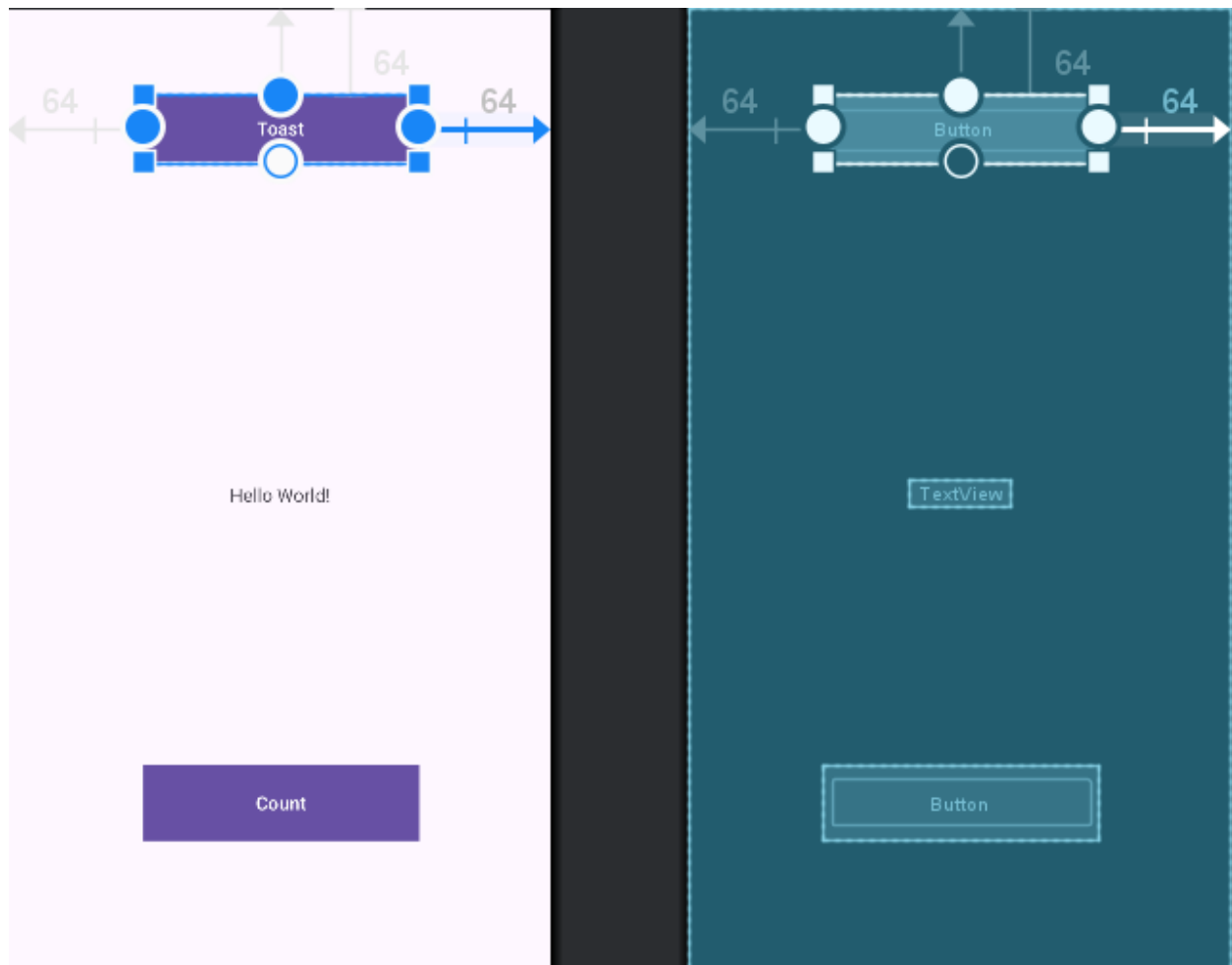
2.Width . Kiểm soát này chỉ định `layout_width` và xuất hiện trong hai phân đoạn ở phía

trái và phải của hình vuông. Các góc cho biết rằng kiểm soát này được đặt thành `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần để vừa với nội dung của nó, lên đến biên độ 8dp.

3. Nút đóng ngăn Attributes. Nhấp để đóng ngăn.

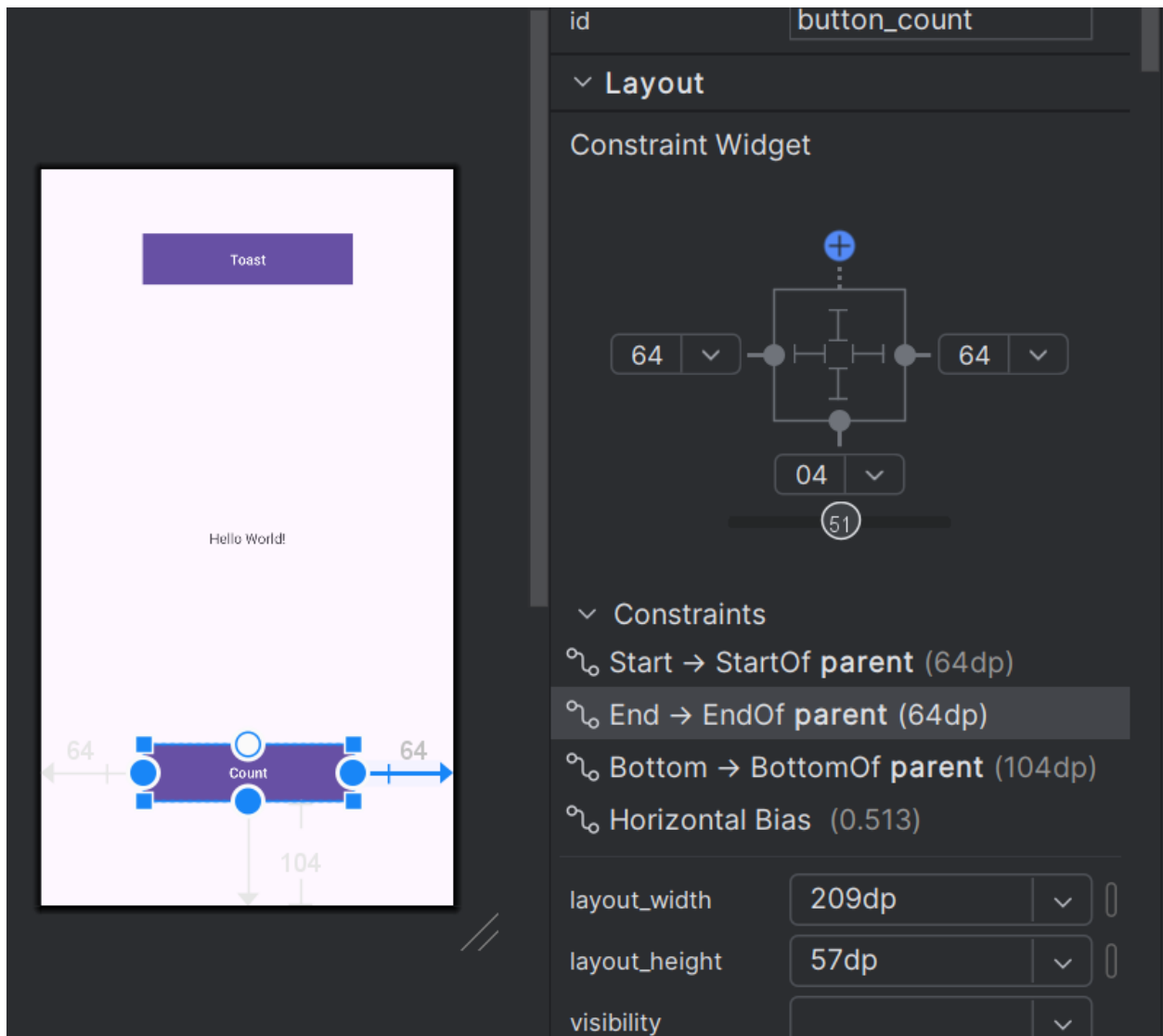
Thực hiện theo các bước sau:

1. Chọn Nút trên cùng trong ngăn Cây thành phần.
2. Nhấp vào tab Thuộc tính ở bên phải cửa sổ trình chỉnh sửa bố cục.
3. Nhấp vào nút điều khiển chiều rộng hai lần—lần nhấp đầu tiên sẽ thay đổi thành Fixed với các đường thẳng và lần nhấp thứ hai sẽ thay đổi thành Phù hợp với các ràng buộc với cuộn lò xo, như thể hiện trong hình động bên dưới.



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

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



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

- Thiết lập `match_constraint` mở rộng phần tử View để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao—lên đến một lề, nếu có. Phần tử cha trong trường hợp này là `ConstraintLayout`. Bạn tìm hiểu thêm về `ConstraintLayout` trong tác vụ tiếp theo.
- Thiết lập `wrap_content` thu nhỏ kích thước của phần tử View để nó chỉ đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, phần tử View sẽ trở nên vô hình.

- Để chỉ định kích thước cố định điều chỉnh theo kích thước màn hình của thiết bị, hãy sử dụng số lượng cố định pixel không phụ thuộc vào mật độ (đơn vị dp). Ví dụ: 16dp nghĩa là 16 pixel không phụ thuộc vào mật độ.

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

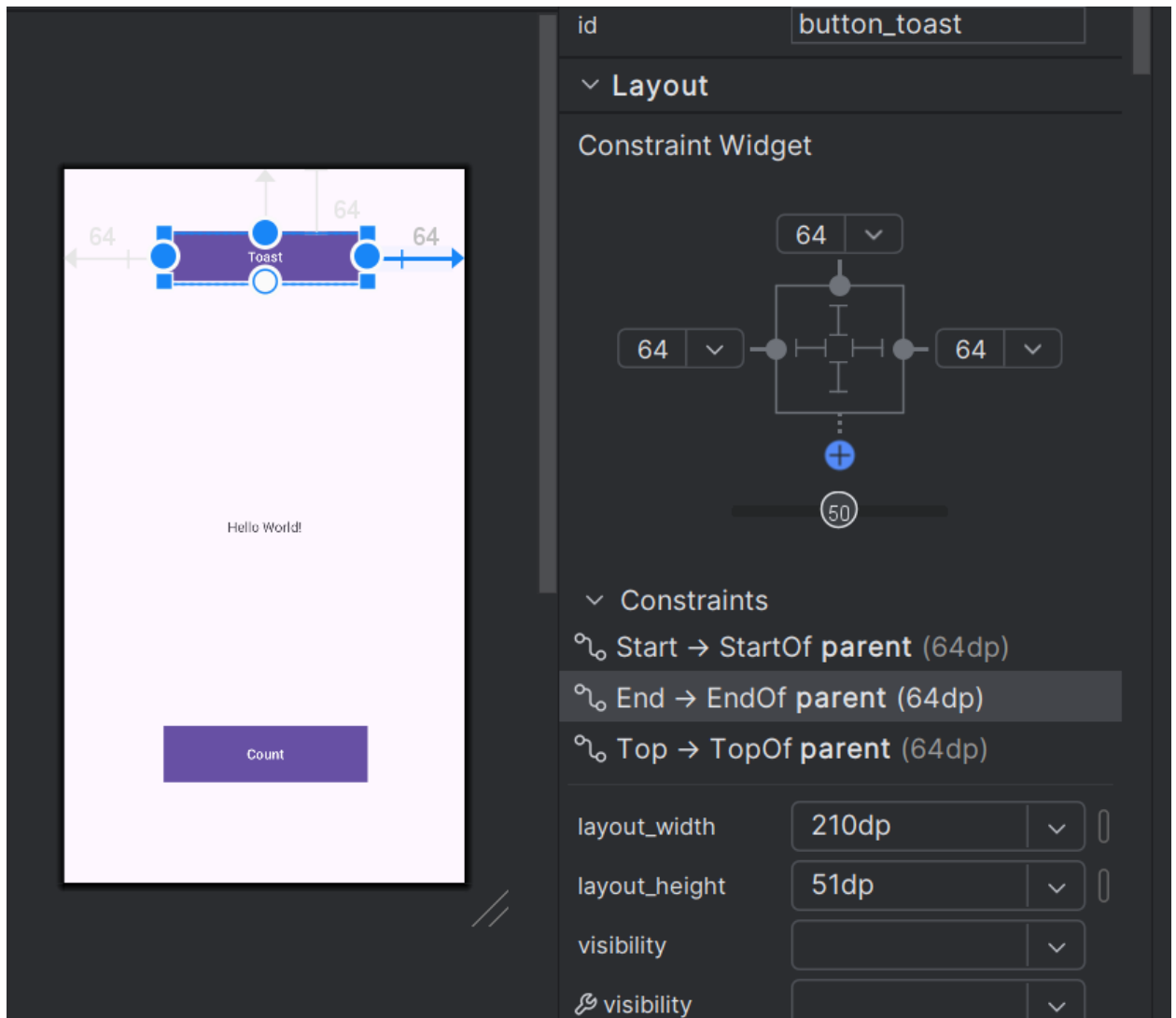
3.2 Thay đổi các thuộc tính của Button

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

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

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

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



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

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

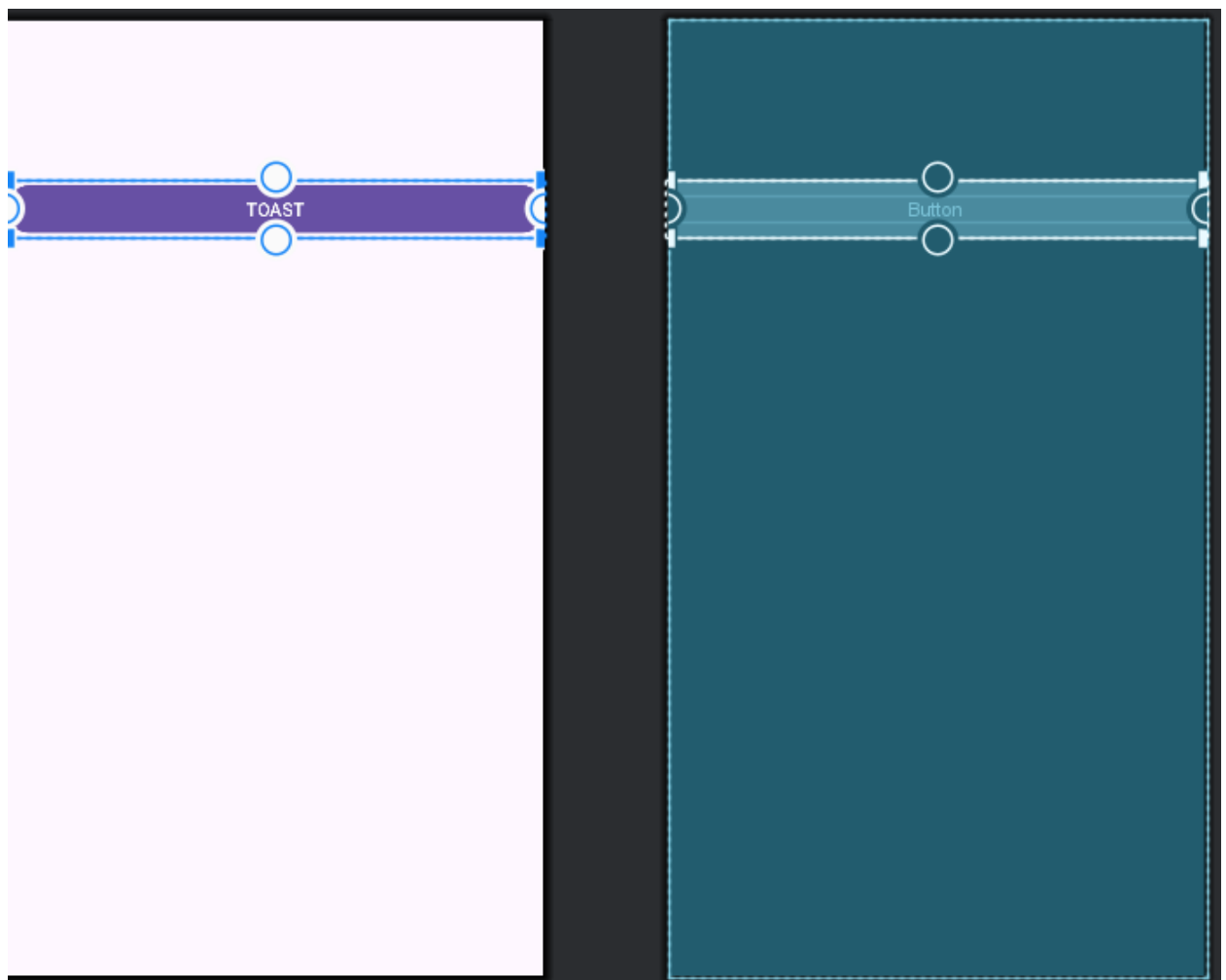
Nhiệm vụ 4: Thêm `TextEdit` và đặt thuộc tính của nó

Một trong những lợi ích của `ConstraintLayout` là khả năng căn chỉnh hoặc hạn chế các thành phần so với các thành phần khác. Trong tác vụ này, bạn sẽ thêm một

TextView vào giữa bố cục và giới hạn theo chiều ngang với lề và theo chiều dọc với hai phần tử Button. Sau đó, bạn sẽ thay đổi các thuộc tính cho TextView trong ngăn Thuộc tính.

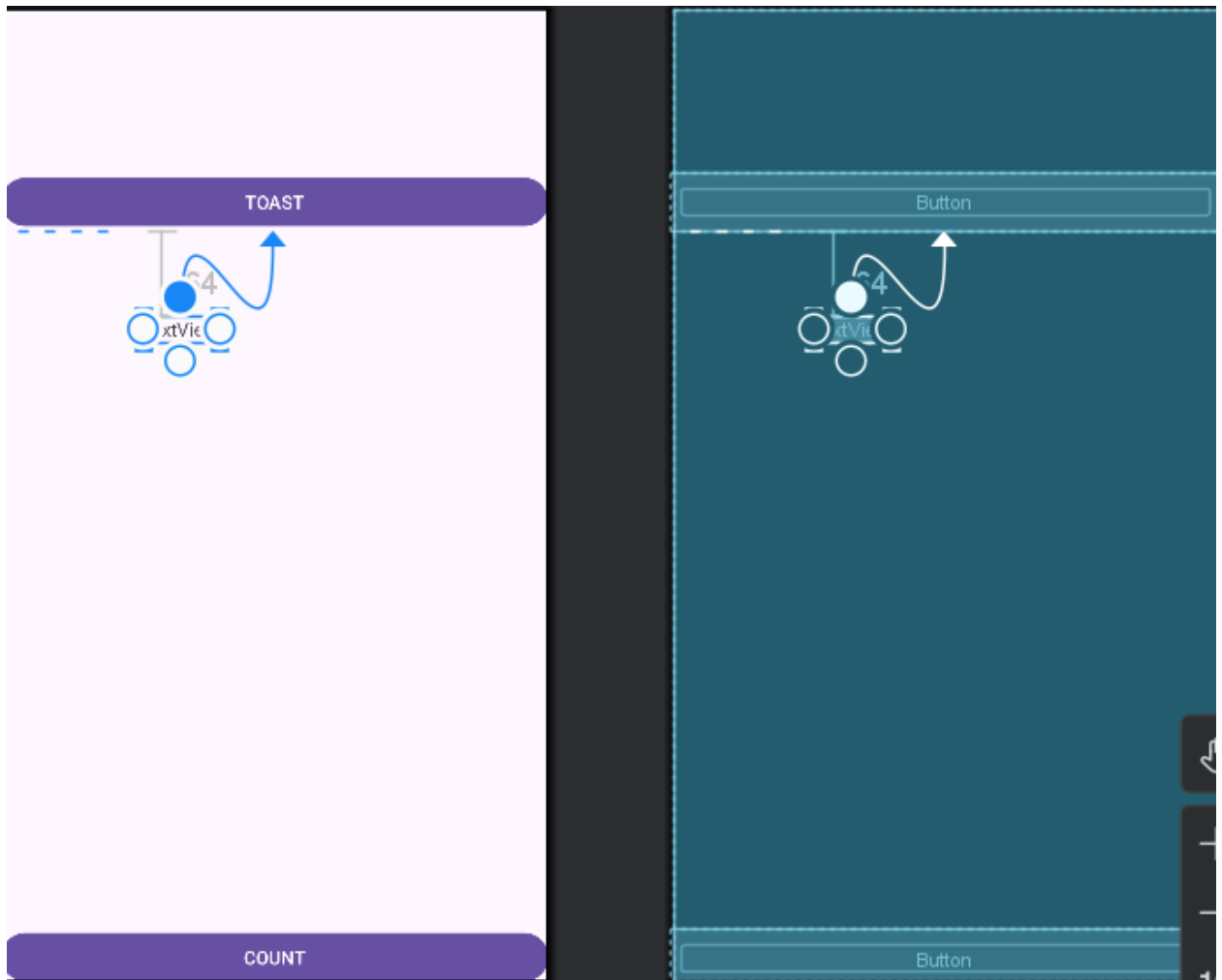
4.1 Thêm TextView và các ràng buộc

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



2. Như được hiển thị trong hình động bên dưới, hãy kéo một ràng buộc từ dưới cùng của TextView đến tay cầm ở trên cùng của Nút đếm và từ các cạnh của

TextView đến các cạnh của bố cục. Thao tác này ràng buộc TextView ở giữa bố cục giữa hai phần tử Nút.



4.2 Đặt các thuộc tính của TextView

Với TextView được chọn, hãy mở ngăn Thuộc tính, nếu ngăn này chưa mở. Đặt các thuộc tính cho TextView như được hiển thị trong hình động bên dưới. Các thuộc tính bạn chưa gặp phải được giải thích sau hình:

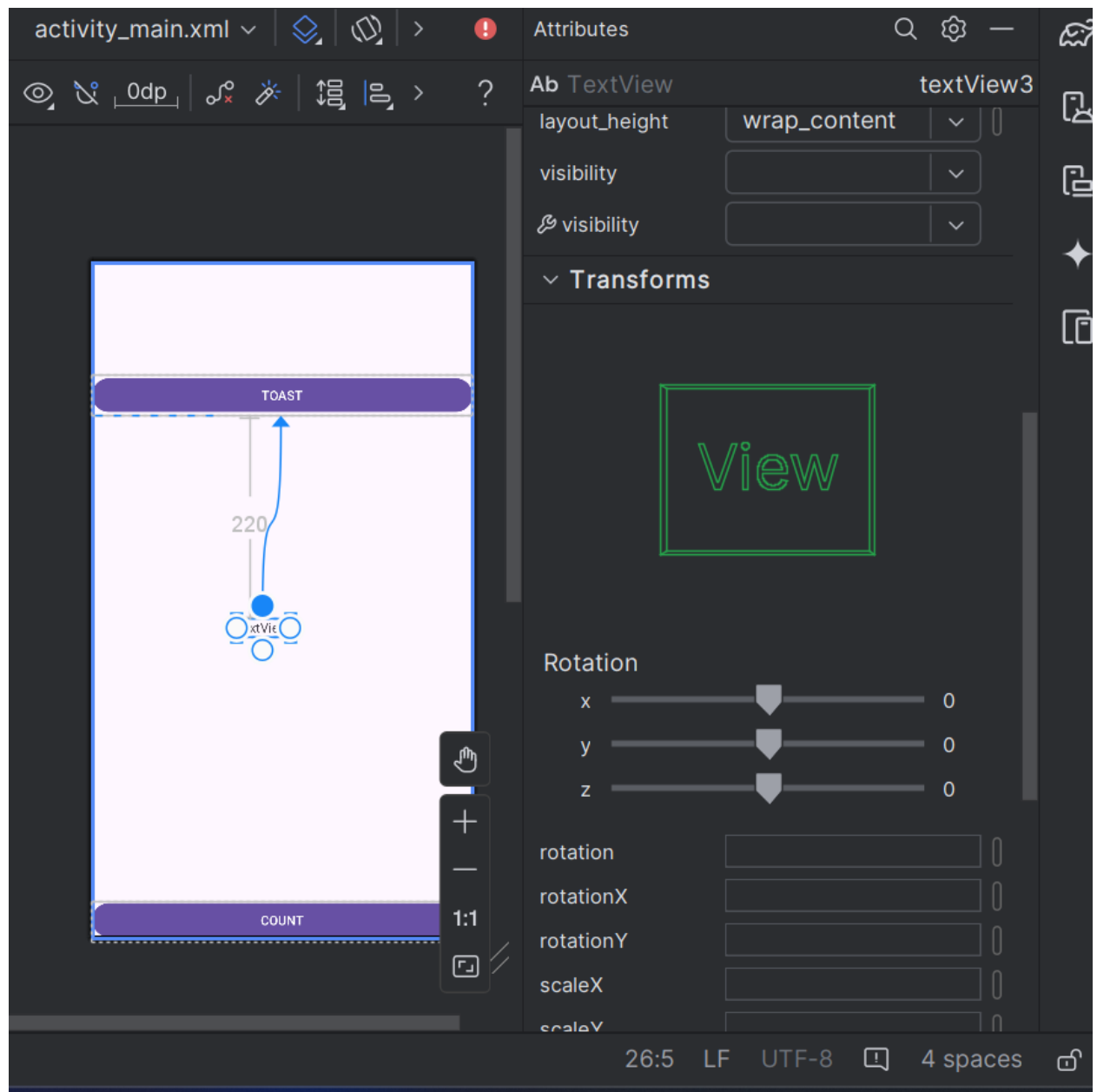
1. Đặt ID thành `show_count`.
2. Đặt văn bản thành 0.
3. Đặt `textSize` thành 160sp.
4. Đặt `textStyle` thành B (in đậm) và `textAlignment` thành `ALIGNCENTER` (căn giữa đoạn văn).

5. Thay đổi các điều khiển kích thước chế độ xem theo chiều ngang và chiều dọc (layout_width và layout_height) thành match_constraint.

6. Đặt textColor thành @color/colorPrimary

7. Cuộn xuống ngăn và nhấp vào Xem tất cả các thuộc tính, cuộn xuống trang thứ hai của thuộc tính để làm nền, sau đó nhập #FFF00 để có màu vàng.

8. Cuộn xuống trọng lực, mở rộng trọng lực và chọn center_ver (để căn giữa theo chiều dọc).



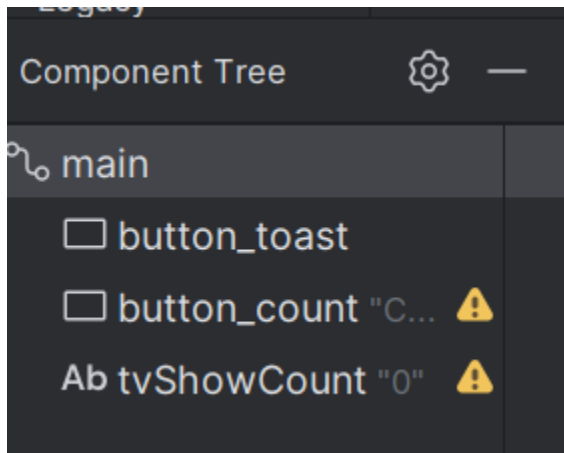
- **textSize:** Kích thước văn bản của TextView. Đối với bài học này, kích thước được đặt thành 160sp. Sp là viết tắt của pixel không phụ thuộc tỷ lệ và giống như dp, là đơn vị tỷ lệ với mật độ màn hình và tùy chọn kích thước phông chữ của người dùng. Sử dụng đơn vị dp khi bạn chỉ định kích thước phông chữ để kích thước được điều chỉnh cho cả mật độ màn hình và tùy chọn của người dùng.
- **textStyle** và **textAlignment:** Kiểu văn bản, được đặt thành B (in đậm) trong bài học này và căn chỉnh văn bản, được đặt thành ALIGNCENTER (căn giữa đoạn văn).
- **gravity:** Thuộc tính trọng lực chỉ định cách căn chỉnh View trong View hoặc ViewGroup cha của nó. Trong bước này, bạn căn giữa TextView để căn giữa theo chiều dọc trong ConstraintLayout cha.

Bạn có thể nhận thấy rằng thuộc tính nền nằm trên trang đầu tiên của ngăn Thuộc tính cho một Nút, nhưng lại nằm trên trang thứ hai của ngăn Thuộc tính cho một TextView. Ngăn Thuộc tính thay đổi cho từng loại Chế độ xem: Các thuộc tính phổ biến nhất cho loại Chế độ xem xuất hiện trên trang đầu tiên, và các thuộc tính còn lại được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của ngăn Thuộc tính, hãy nhấp vào biểu tượng trên thanh công cụ ở đầu ngăn.

e)Nhiệm vụ 5 : Chỉnh sửa layout trong XML.

5.1 Mở tệp bố cục XML.

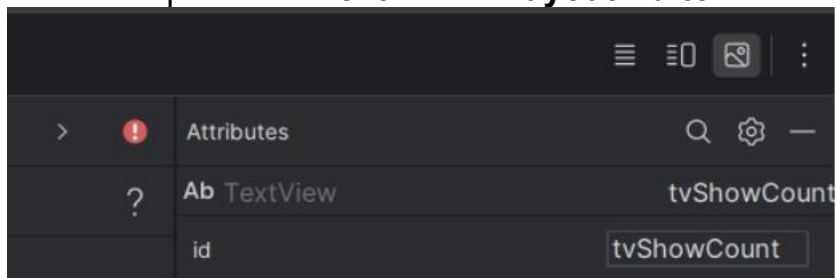
Giao diện của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử UI trong Component Tree. Di chuột qua các dấu chấm than này để xem thông báo cảnh báo, như minh họa bên dưới. Tất cả ba phần tử đều có chung một cảnh báo: Các chuỗi được mã hóa cứng (hardcoded) nên sử dụng tài nguyên.



Cách dễ nhất để khắc phục sự cố bố cục là chỉnh sửa trực tiếp trong XML. Mặc dù **Layout Editor** là một công cụ mạnh mẽ, nhưng một số thay đổi sẽ dễ thực hiện hơn trong mã XML.

Thực hiện các bước sau:

1. Mở tệp **activity_main.xml** (nếu chưa mở).
2. Nhấp vào tab **Text** ở cuối **Layout Editor**.



Trình chỉnh sửa XML sẽ xuất hiện, thay thế các bảng thiết kế và bản vẽ bố cục (**Design** và **Blueprint**). Như trong hình bên dưới, các cảnh báo được đánh dấu—bao gồm các chuỗi được mã hóa cứng **"Toast"** và **"Count"**. (Chuỗi **"0"** cũng bị đánh dấu nhưng không hiển thị trong hình minh họa.)

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

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

    <Button
        android:id="@+id/button_toast"
        android:layout_width="210dp"
        android:layout_height="51dp"
        android:layout_marginStart="64dp"
        android:layout_marginTop="64dp"
        android:layout_marginEnd="64dp"
        android:background="@color/design_default_color_on_primary"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="Toast"
        android:textColor="@color/design_default_color_primary"/>

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

```

<Button
    android:id="@+id/button_count"
    android:layout_width="209dp"
    android:layout_height="57dp"
    android:layout_marginStart="64dp"
    android:layout_marginEnd="64dp"
    android:layout_marginBottom="104dp"
    android:background="@color/design_default_color_on_primary"
    android:text="Count"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.513"
    app:layout_constraintStart_toStartOf="parent"
    android:textColor="@color/design_default_color_primary"
/>

<TextView
    android:id="@+id/tvShowCount"
    android:layout_width="403dp"
    android:layout_height="230dp"
    android:layout_marginStart="173dp"
    android:layout_marginTop="239dp"

```

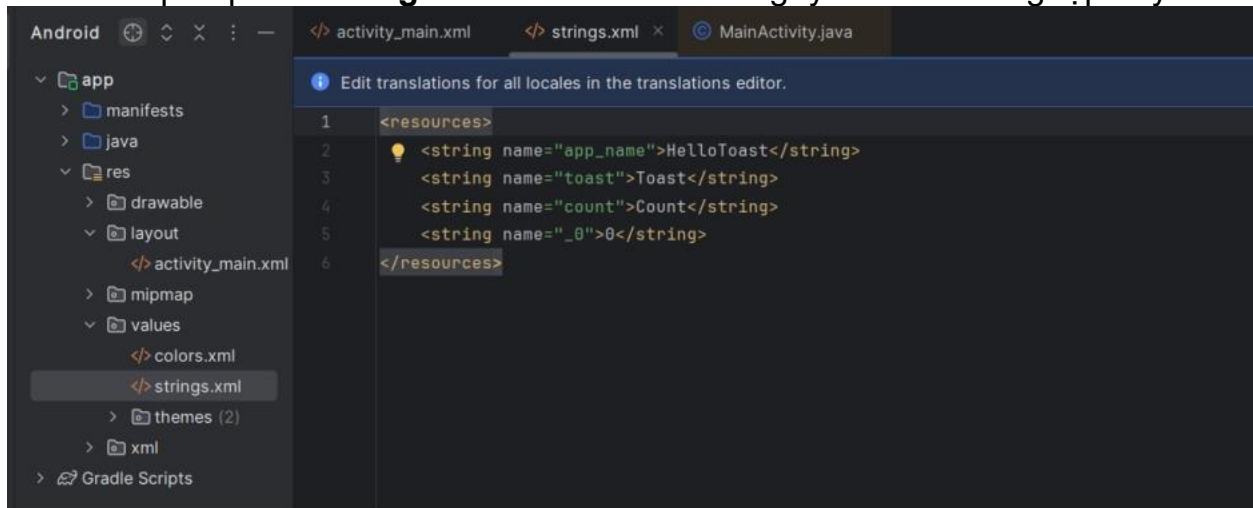
5.2 Trích xuất tài nguyên chuỗi.

Thay vì mã hóa cứng (hard-code) chuỗi văn bản, **thực hành tốt nhất** là sử dụng tài nguyên chuỗi (**string resources**) để đại diện cho các chuỗi. Việc lưu trữ chuỗi trong một tệp riêng giúp dễ dàng quản lý, đặc biệt nếu bạn sử dụng chúng nhiều lần. Ngoài ra, tài nguyên chuỗi là **bắt buộc** để dịch và bản địa hóa ứng dụng, vì bạn cần tạo một tệp tài nguyên chuỗi riêng cho từng ngôn ngữ.

Thực hiện các bước sau:

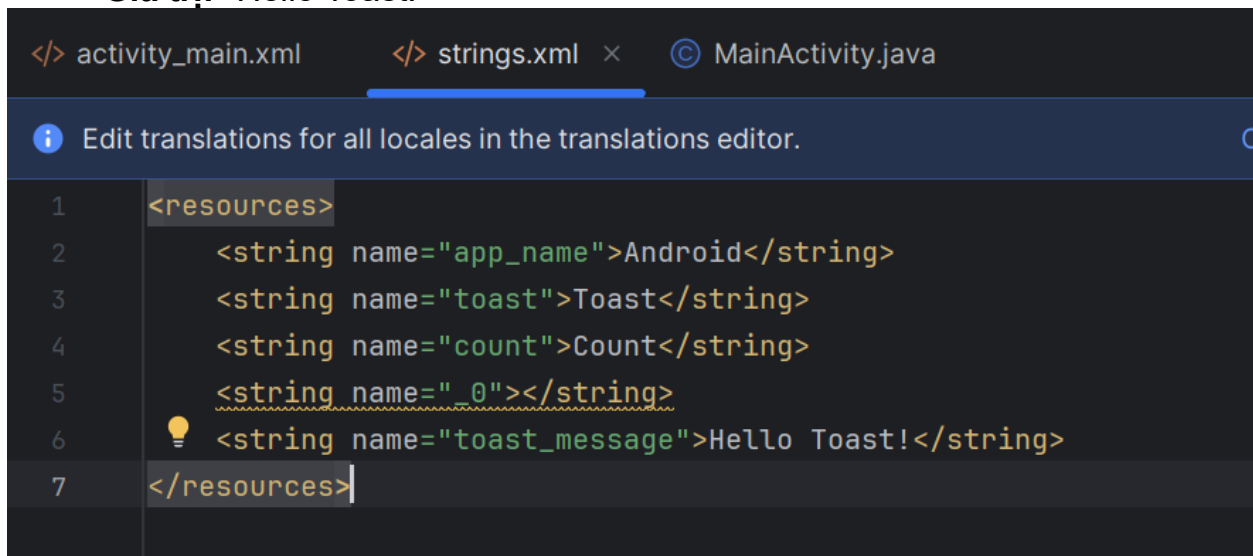
1. Nhấp vào từ **"Toast"** (cảnh báo đầu tiên được đánh dấu).
2. Nhấn **Alt + Enter** trên Windows hoặc **Option + Enter** trên macOS, sau đó chọn **Extract string resource** từ menu bật lên.
3. Nhập **button_label_toast** cho trường **Resource name**.
4. Nhấn **OK**. Một tài nguyên chuỗi sẽ được tạo trong tệp **values/res/strings.xml**, và chuỗi trong mã sẽ được thay thế bằng tham chiếu đến tài nguyên:
5. Trích xuất các chuỗi còn lại:
 - **button_label_count** cho **"Count"**
 - **count_initial_value** cho **"0"**

6. Trong **Project > Android**, mở rộng thư mục **values** trong **res**, sau đó nhấp đúp vào **strings.xml** để xem các tài nguyên chuỗi trong tệp này.



7. Bạn cần thêm một chuỗi mới để sử dụng trong bước tiếp theo. **Thêm vào tệp strings.xml một tài nguyên chuỗi mới với:**

- **Tên:** toast_message
- **Giá trị:** "Hello Toast!"



1.3) Lưu ý:

Tài nguyên chuỗi bao gồm **tên ứng dụng**, xuất hiện trên thanh ứng dụng (App Bar) nếu bạn tạo dự án với mẫu **Empty Template**. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa giá trị **app_name** trong **strings.xml**.

a) Nhiệm vụ 6 : Tạo sự kiện onClick cho các Button

1.1 Thêm thuộc tính onClick và trình xử lý cho từng Button.

Trình xử lý sự kiện nhấp chuột (click handler) là phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử giao diện có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường **onClick** ở bảng **Attributes** của 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 **Button**. Chúng ta sẽ sử dụng phương pháp thứ hai vì hiện tại các phương thức xử lý chưa được tạo, và trình chỉnh sửa XML cung cấp cách tự động tạo các phương thức này.

1. Khi trình chỉnh sửa XML đang mở (tab **Text**), tìm **Button** có **android:id** được đặt thành **button_toast**.

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="210dp"
    android:layout_height="51dp"
    android:layout_marginStart="64dp"
    android:layout_marginTop="64dp"
    android:layout_marginEnd="64dp"
    android:background="@color/design_default_color_on_primary"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:text="Toast"
    android:textColor="@color/design_default_color_primary"/>
```

2. Thêm thuộc tính **android:onClick** vào cuối phần tử **button_toast**, sau thuộc tính cuối cùng và trước ký hiệu **/>**.
3. Nhấp vào biểu tượng **bóng đèn đỏ** xuất hiện bên cạnh thuộc tính. Chọn **Create click handler**, chọn **MainActivity**, rồi nhấp **OK**.
 - Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy nhấp vào tên phương thức ("**showToast**"), nhấn **Alt + Enter** (hoặc **Option + Enter** trên Mac), chọn **Create 'showToast(view)' in MainActivity**, rồi nhấn **OK**.
 - Hành động này sẽ tạo một phương thức **showToast()** trong **MainActivity**.

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="210dp"
    android:layout_height="51dp"
    android:layout_marginStart="64dp"
    android:layout_marginTop="64dp"
    android:layout_marginEnd="64dp"
    android:background="@color/design_default_color_on_primary"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:text="Toast"
    android:textColor="@color/design_default_color_primary"
    android:onClick="showToast"
    tools:ignore="OnClick" />
```

4. Lặp lại các bước trên với **button_count**: Thêm thuộc tính **android:onClick** và tạo trình xử lý sự kiện.


```
<Button
    android:id="@+id/button_count"
    android:layout_width="209dp"
    android:layout_height="57dp"
    android:layout_marginStart="64dp"
    android:layout_marginEnd="64dp"
    android:layout_marginBottom="104dp"
    android:background="@color/design_default_color_on_primary"
    android:text="Count"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.513"
    app:layout_constraintStart_toStartOf="parent"
    android:textColor="@color/design_default_color_primary"
    android:onClick="countUp"
    tools:ignore="OnClick"
/>
```

5. Nếu MainActivity.java chưa được mở, hãy mở rộng thư mục java trong chế độ xem Project > Android, mở rộng com.example.android.hellotoast, sau đó nhấp đúp vào MainActivity.

Trình chỉnh sửa mã sẽ hiển thị nội dung của MainActivity.

```
</> activity_main.xml × </> strings.xml © MainActivity.java ×
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         EdgeToEdge.enable($this$enableEdgeToEdge: this);
18         setContentView(R.layout.activity_main);
19         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) {
20             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
21             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars
22             return insets;
23         });
24     }
25
26     1 usage
27     public void showToast(View view){
28
29     }
30
31     1 usage
32     public void countUp(View view){
33
34     }
35 }
```

1.2 Chỉnh sửa trình xử lý sự kiện của Button Toast

Bạn sẽ chỉnh sửa phương thức `showToast()`—trình xử lý nhấp vào nút Toast trong `MainActivity`—để hiển thị một thông báo.

Toast cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ chiếm không gian cần thiết cho thông báo. Hoạt động hiện tại vẫn hiển thị và có thể tương tác. Toast có thể hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn—thêm một thông báo Toast để hiển thị kết quả của việc nhấn một nút hoặc thực hiện một hành động.

Thực hiện các bước sau để chỉnh sửa trình xử lý nhấp vào nút Toast:

1. Xác định vị trí phương thức `showToast()` vừa tạo.
2. Để tạo một thể hiện của Toast, hãy gọi phương thức `makeText()` của lớp `Toast`. Câu lệnh này chưa hoàn chỉnh cho đến khi bạn hoàn thành tất cả các bước.

3. Cung cấp ngữ cảnh của Activity ứng dụng. Vì Toast hiển thị trên giao diện Activity, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong Activity cần ngữ cảnh, hãy sử dụng `this` làm lỗi tắt.
4. Cung cấp thông báo để hiển thị, chẳng hạn như một tài nguyên chuỗi (chuỗi `toast_message` bạn đã tạo ở bước trước). Tài nguyên chuỗi `toast_message` được xác định bởi `R.string`.
5. Cung cấp thời gian hiển thị cho Toast. Ví dụ, `Toast.LENGTH_SHORT` sẽ hiển thị thông báo trong một khoảng thời gian ngắn. Ví dụ, `Toast.LENGTH_SHORT` sẽ hiển thị thông báo trong một khoảng thời gian ngắn. Thời gian hiển thị của **Toast** có thể là `Toast.LENGTH_LONG` hoặc `Toast.LENGTH_SHORT`. Thời gian thực tế khoảng **3.5 giây** đối với `Toast.LENGTH_LONG` và **2 giây** đối với `Toast.LENGTH_SHORT`.

```
1 usage
public void showToast(View view){
    Toast.makeText( context: this, R.string.toast_message, Toast.LENGTH_LONG).show();
}
```

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

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

5.3 Chỉnh sửa trình xử lý sự kiện của Button Toast

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

Mã xử lý phải:

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

Thực hiện các bước sau để chỉnh sửa trình xử lý khi nhấn nút **Count**:

1. Xác định vị trí phương thức `countUp()` vừa tạo.
2. Để theo dõi số đếm, bạn cần một biến thành viên riêng. Mỗi lần nhấn nút **Count** sẽ tăng giá trị của biến này. Nhập đoạn sau, phần này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn đỏ:

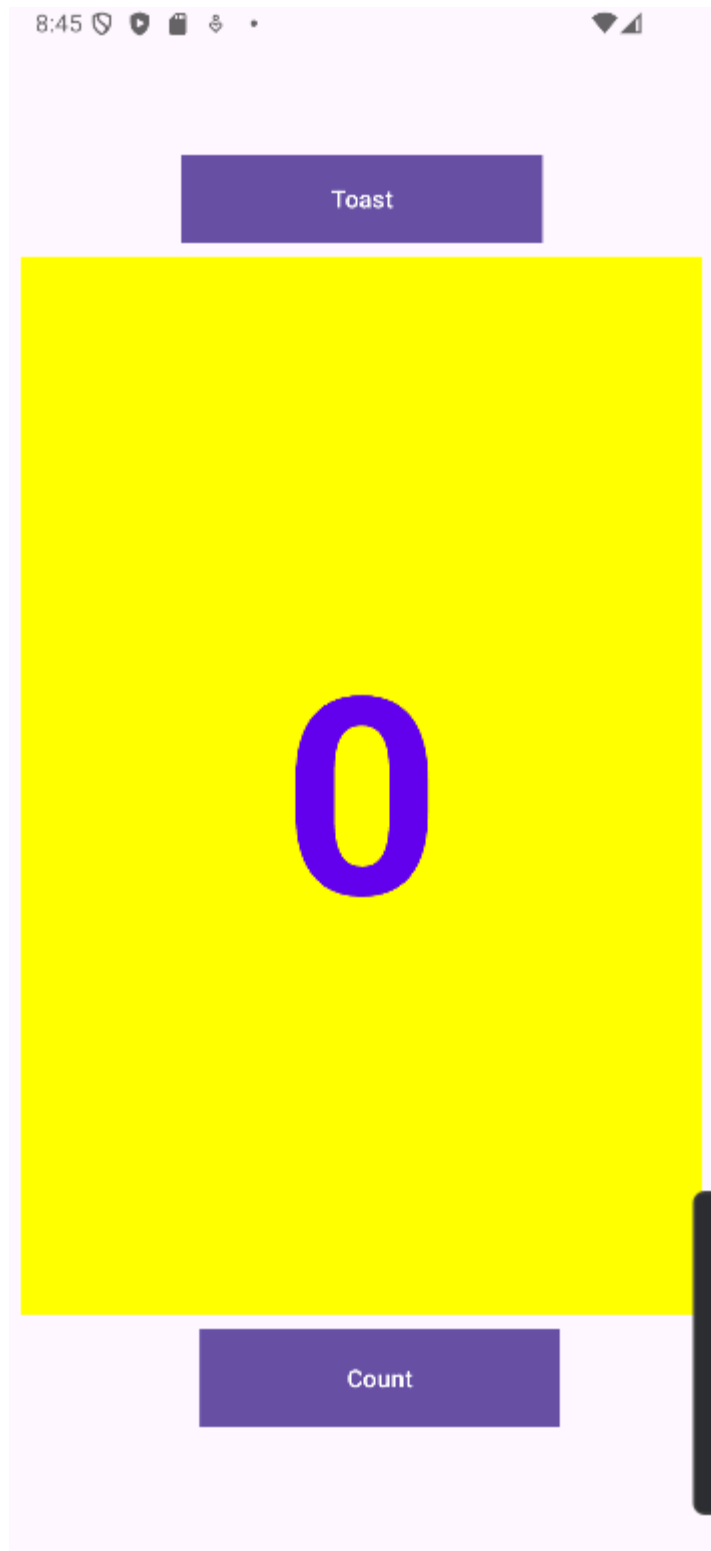
```
1 usage  
public void showToast(View view){  
  
}
```

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

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'mCount'** từ menu bật lên. Thao tác này sẽ tạo một biến thành viên riêng ở đầu `MainActivity`, và Android Studio sẽ tự động đặt kiểu dữ liệu là `int`.
4. Thay đổi câu lệnh khai báo biến thành viên riêng để khởi tạo giá trị ban đầu là 0.
5. Bên cạnh biến trên, bạn cũng cần một biến thành viên riêng để tham chiếu đến `TextView` **show_count**, biến này sẽ được thêm vào trình xử lý khi nhấn nút. Đặt tên biến là `mShowCount`.

```
<img alt="Run icon" data-bbox="121 755 141 775"/> </> public class MainActivity extends AppCompatActivity {  
  
2 usages  
private int mCount = 0;  
2 usages
```

6. Khi đã có `mShowCount`, bạn có thể lấy tham chiếu đến `TextView` bằng ID đã đặt trong tệp giao diện. Để tránh gọi lại nhiều lần, hãy khai báo tham chiếu này trong phương thức `onCreate()`. Như bạn sẽ học trong bài học khác, phương thức `onCreate()` được dùng để **inflate layout**, nghĩa là đặt nội dung hiển thị của màn hình theo tệp XML. Nó cũng có thể dùng để lấy tham chiếu đến các thành phần UI khác trong giao diện, như `TextView`. Xác định vị trí phương thức `onCreate()` trong `MainActivity`.



7. Thêm câu lệnh `findViewById` vào cuối phương thức:
Một `View`, giống như một chuỗi, là một tài nguyên có thể có `id`. Lệnh `findViewById` nhận ID của một `View` làm tham số và trả về `View`. Vì phương

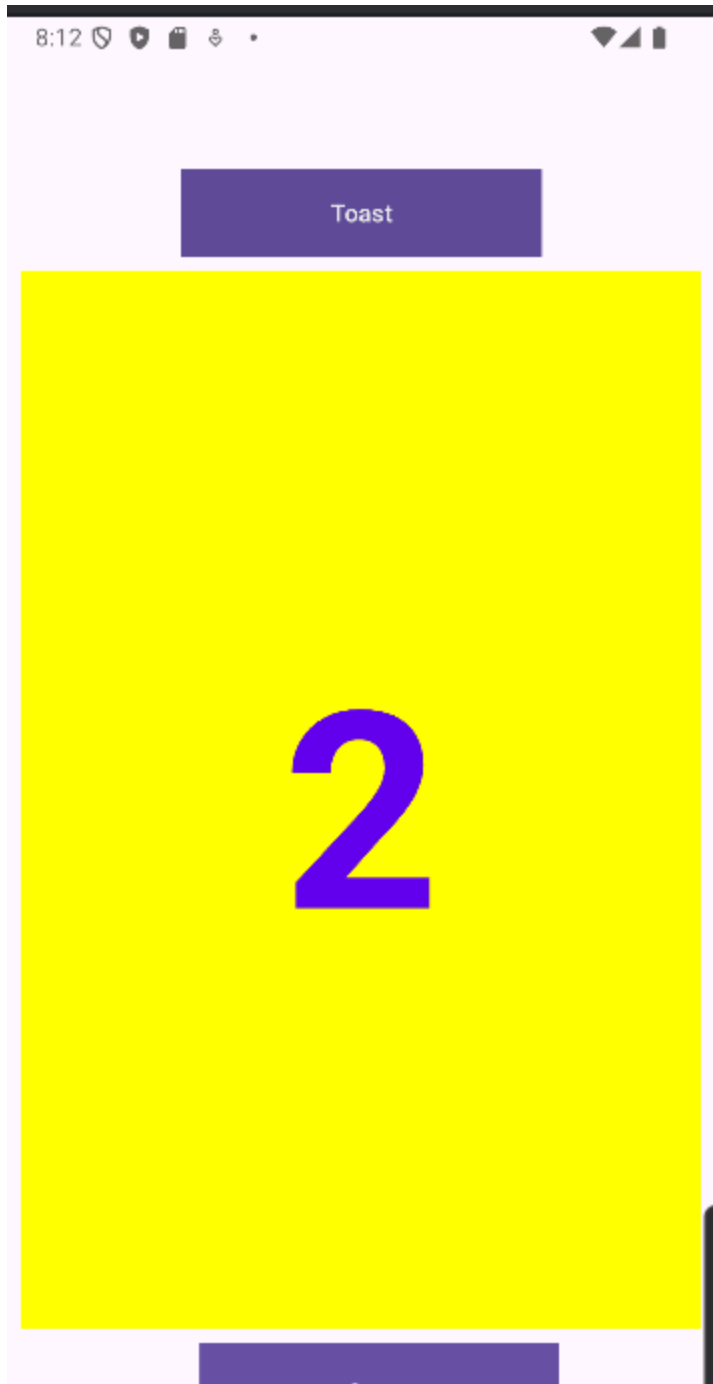
thức này trả về một View, bạn cần ép kiểu kết quả về kiểu mong đợi, trong trường hợp này là (TextView).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_main);
    mShowCount = findViewById(R.id.show_count);
    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;
    });
}
```

8. Khi đã gán TextView vào mShowCount, bạn có thể sử dụng biến này để đặt văn bản trong TextView thành giá trị của biến mCount. Thêm dòng lệnh này vào phương thức countUp().

```
1 usage
public void countUp(View view){
    mCount++;
    if (mShowCount != null) {
        mShowCount.setText(String.valueOf(mCount));
    }
}
```

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



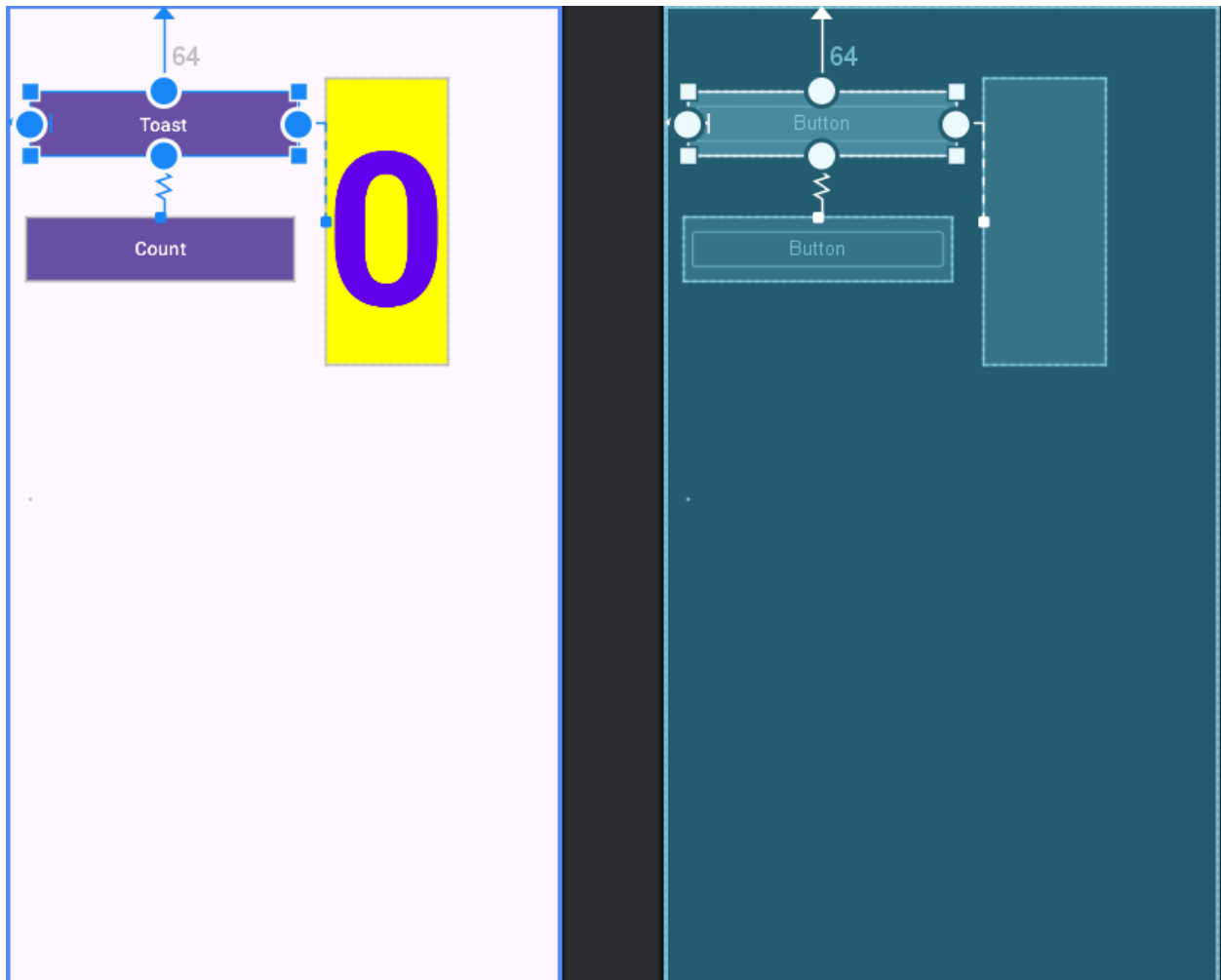
Coding challenge

Ứng dụng **HelloToast** hiển thị tốt khi thiết bị hoặc trình giả lập ở chế độ dọc. Tuy nhiên, khi chuyển sang chế độ ngang, nút **Count** có thể chồng lên **TextView** ở phía dưới, như hình minh họa.

Android Studio project:

Thử thách: Thay đổi bố cục để hiển thị tốt ở cả chế độ ngang và dọc

1. Trên máy tính, sao chép thư mục dự án **HelloToast** và đổi tên thành **HelloToastChallenge**.
2. Mở **HelloToastChallenge** trong Android Studio và thực hiện refactor. (Xem **Phụ lục: Tiện ích** để biết cách sao chép và refactor dự án.)
3. Thay đổi bố cục sao cho nút **Toast** và **Count** xuất hiện ở bên trái, như trong hình minh họa. **TextView** nằm bên cạnh nhưng chỉ đủ rộng để hiển thị nội dung. (Gợi ý: Dùng `wrap_content`.)
4. Chạy ứng dụng ở cả chế độ ngang và dọc để kiểm tra.



Challenge solution code

Android Studio project:

Tóm tắt

View, ViewGroup và Layouts:

- Tất cả các phần tử giao diện người dùng (UI) đều là lớp con của View, kế thừa nhiều thuộc tính từ lớp View cha.
- Các phần tử View có thể được nhóm bên trong ViewGroup, đóng vai trò như một container. Mỗi quan hệ giữa chúng là **parent-child**: **parent** là ViewGroup, còn **child** là View hoặc một ViewGroup khác.
- Phương thức onCreate() được dùng để **inflate layout**, tức là đặt nội dung màn hình theo tệp XML. Nó cũng có thể dùng để lấy tham chiếu đến các phần tử UI khác trong layout.
- Một View, giống như một chuỗi, là một tài nguyên có thể có id. Phương thức findViewById nhận ID của View làm tham số và trả về View tương ứng.

Sử dụng trình chỉnh sửa layout

- Nhấn vào tab **Design** để thao tác trực tiếp trên giao diện, hoặc tab **Text** để chỉnh sửa mã XML.
- **Palettes pane** hiển thị các phần tử UI có thể sử dụng, còn **Component tree pane** hiển thị cây phân cấp của giao diện.
- **Blueprint pane** và **Design pane** hiển thị các phần tử UI trong layout.
- Tab **Attributes** hiển thị các thuộc tính của phần tử UI.
- **Constraint handle**: Nhấp vào điểm tròn trên mỗi cạnh của phần tử, kéo đến một điểm khác hoặc cạnh của ViewGroup để tạo ràng buộc (constraint), hiển thị dưới dạng đường zigzag.
- **Resizing handle**: Kéo góc vuông để thay đổi kích thước phần tử.
- **Autoconnect** tự động tạo ràng buộc khi kéo phần tử vào layout.
- Xóa ràng buộc bằng nút **Clear Constraints** hoặc nhấp vào điểm cụ thể của constraint cần xóa.
- **View inspector** trong tab **Attributes** giúp thay đổi chiều rộng và chiều cao của phần tử.

Thiết lập chiều rộng và chiều cao layout

layout_width và layout_height thay đổi khi chỉnh sửa trong **view inspector**. Trong ConstraintLayout, có 3 giá trị:

- **match_constraint**: Mở rộng phần tử theo chiều rộng hoặc chiều cao để lấp đầy phần tử cha (trừ khi có margin).
- **wrap_content**: Thu nhỏ phần tử vừa đủ để chứa nội dung. Nếu không có nội dung, phần tử sẽ ẩn.
- **Sử dụng dp (density-independent pixels)** để đặt kích thước cố định, điều chỉnh phù hợp với màn hình thiết bị.

Trích xuất chuỗi tài nguyên

Không nên hard-code chuỗi, thay vào đó nên sử dụng **string resources**:

1. Nhấp vào chuỗi cần trích xuất, nhấn Alt + Enter (Option + Enter trên Mac) và chọn **Extract string resources**.
2. Đặt tên tài nguyên (Resource name).
3. Nhấn **OK**, chuỗi sẽ được lưu vào res/values/strings.xml và trong mã nguồn sẽ thay thế bằng @string/resource_name.

Xử lý sự kiện nhấn (Click Handler)

- **Click handler** là phương thức được gọi khi người dùng nhấn vào phần tử UI.
- Cách đặt click handler cho Button:
 - Nhập tên phương thức vào **onClick** trong tab **Attributes** của **Design**.
 - Hoặc thêm thuộc tính android:onClick vào XML của Button.
- Tạo phương thức xử lý trong MainActivity
- Xem tài liệu của lớp **Button** để biết các thuộc tính của nút, và tài liệu của **TextView** để biết các thuộc tính của văn bản.

Toast hiển thị một thông báo ngắn trong cửa sổ popup nhỏ, chỉ chiếm không gian vừa đủ cho nội dung.

Cách tạo Toast:

- Gọi phương thức makeText() từ lớp Toast.
- Truyền **context của Activity** và **chuỗi thông báo** (có thể dùng string resource).
- Truyền thời gian hiển thị:
 - Toast.LENGTH_SHORT (ngắn)
 - Toast.LENGTH_LONG (dài)
- Gọi show() để hiển thị **Toast**.

1.1) Trình chỉnh sửa bố cục

Giới thiệu

Như bạn đã học trong **Phần 1.3: Giao diện người dùng (UI) tương tác đầu tiên**, bạn có thể tạo UI bằng **ConstraintLayout** trong trình chỉnh sửa bố cục. **ConstraintLayout** cho phép sắp xếp các phần tử UI bằng cách thiết lập ràng buộc với các phần tử khác hoặc với mép của bố cục. Nó được thiết kế để giúp bạn dễ dàng kéo thả các phần tử vào giao diện.

ConstraintLayout là một **ViewGroup**, tức là một View đặc biệt có thể chứa các View khác (gọi là **child views**). Trong bài thực hành này, bạn sẽ tìm hiểu thêm về các tính năng của **ConstraintLayout** và trình chỉnh sửa bố cục. Ngoài ra, bài học cũng giới thiệu hai lớp con khác của **ViewGroup**:

- **LinearLayout**: Bố cục sắp xếp các phần tử con theo chiều ngang hoặc dọc.

- **RelativeLayout**: Bố cục cho phép sắp xếp các phần tử con dựa trên vị trí của các phần tử khác hoặc của chính **ViewGroup** cha.

Kiến thức cần có

Bạn nên biết cách:

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

Bạn sẽ học được

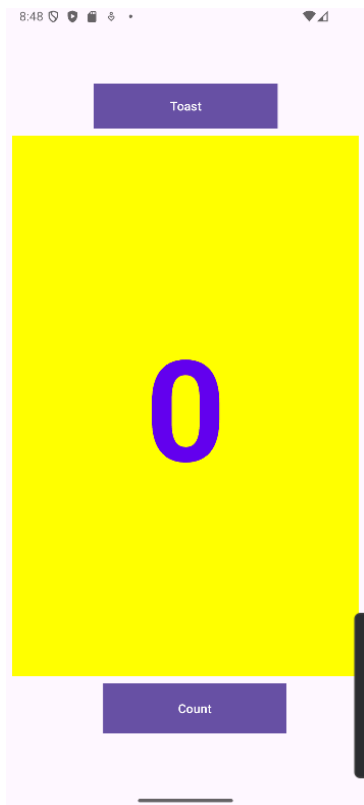
- Cách tạo bố cục riêng cho màn hình ngang (**landscape**).
- Cách tạo bố cục riêng cho máy tính bảng và màn hình lớn.
- Cách sử dụng **baseline constraint** để căn chỉnh phần tử UI với văn bản.
- Cách dùng nút **pack** và **align** để sắp xếp phần tử trong bố cục.
- Cách định vị phần tử trong **LinearLayout**.
- Cách định vị phần tử trong **RelativeLayout**.

Bạn sẽ thực hiện

- Tạo bố cục cho màn hình ngang.
- Tạo bố cục cho máy tính bảng và màn hình lớn.
- Điều chỉnh bố cục và thêm ràng buộc cho các phần tử UI.
- Dùng **baseline constraint** trong **ConstraintLayout** để căn chỉnh phần tử với văn bản.
- Sử dụng nút **pack** và **align** trong **ConstraintLayout** để sắp xếp phần tử.
- Chuyển bố cục sang **LinearLayout**.
- Định vị các phần tử trong **LinearLayout**.
- Chuyển bố cục sang **RelativeLayout**.
- Sắp xếp lại các phần tử trong bố cục chính để căn chỉnh tương đối với nhau.

Tổng quan về ứng dụng

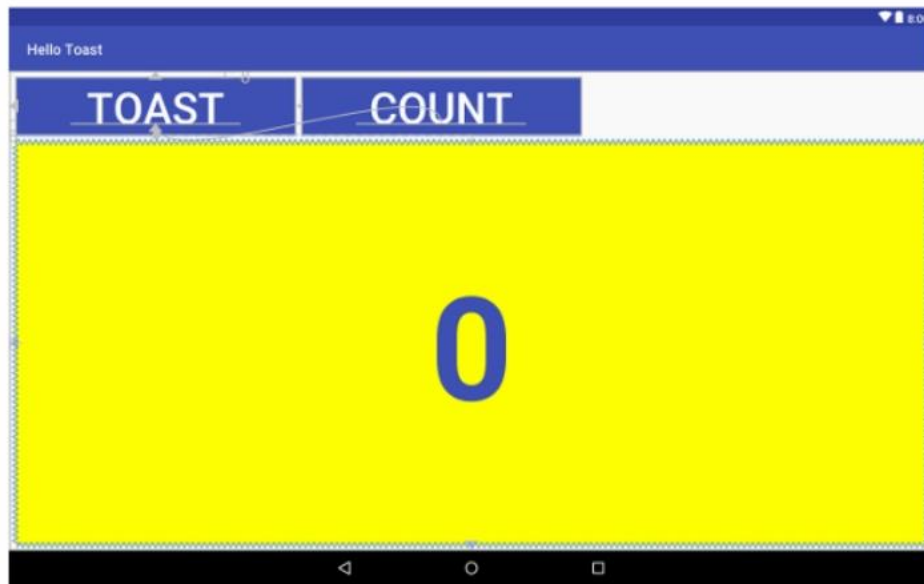
Ứng dụng Hello Toast trong bài học trước sử dụng **ConstraintLayout** để sắp xếp các thành phần giao diện trong Activity, như minh họa trong hình dưới đây.



Để thực hành thêm với `ConstraintLayout`, bạn sẽ tạo một phiên bản bố cục khác cho chế độ ngang, như hình minh họa.



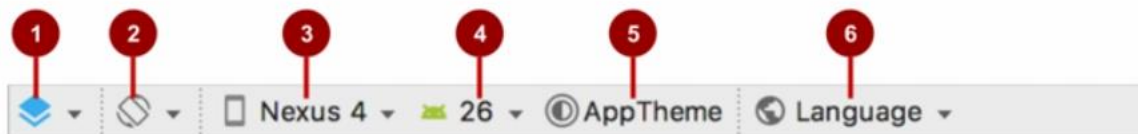
Bạn cũng sẽ học cách sử dụng ràng buộc đường cơ sở (baseline constraints) và một số tính năng căn chỉnh của ConstraintLayout bằng cách tạo một phiên bản bố cục khác dành cho máy tính bảng.



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

a) Nhiệm vụ 1 : Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu thay đổi bố cục của ứng dụng Hello Toast để hiển thị phù hợp ở chế độ ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng tạo các biến thể bố cục cho chế độ ngang (còn gọi là *landscape*) và chế độ dọc (*portrait*) trên điện thoại, cũng như trên các thiết bị màn hình lớn như máy tính bảng.



Bạn sẽ sử dụng một số nút trong hai thanh công cụ trên cùng của trình chỉnh sửa bố cục (Layout Editor):

Thanh này cho phép bạn tùy chỉnh giao diện xem trước bố cục trong trình chỉnh sửa:

1. **Chọn bề mặt thiết kế (Design Surface):**
 - Chọn **Design** để xem trước bố cục có màu sắc.
 - Chọn **Blueprint** để hiển thị các đường viền của các phần tử UI.
 - Chọn **Design + Blueprint** để hiển thị cả hai chế độ song song.
2. **Hướng trong trình chỉnh sửa (Orientation in Editor):**
 - Chọn **Portrait** hoặc **Landscape** để xem trước bố cục theo chiều dọc hoặc ngang.
 - Để tạo bố cục thay thế, chọn **Create Landscape Variation** hoặc các biến thể khác.
3. **Thiết bị trong trình chỉnh sửa (Device in Editor):**
 - Chọn loại thiết bị (điện thoại/máy tính bảng, Android TV, Android Wear).
4. **Phiên bản API trong trình chỉnh sửa (API Version in Editor):**
 - Chọn phiên bản Android để hiển thị xem trước.
5. **Chủ đề trong trình chỉnh sửa (Theme in Editor):**
 - Chọn chủ đề (ví dụ: **AppTheme**) để áp dụng cho bố cục xem trước.
6. **Ngôn ngữ trong trình chỉnh sửa (Locale in Editor):**
 - Chọn ngôn ngữ hiển thị. Danh sách này chỉ hiển thị các ngôn ngữ có trong tài nguyên chuỗi của ứng dụng.
 - Có thể chọn **Preview as Right To Left** để xem bố cục dưới dạng ngôn ngữ viết từ phải sang trái (RTL).

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



Trong hình trên:

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

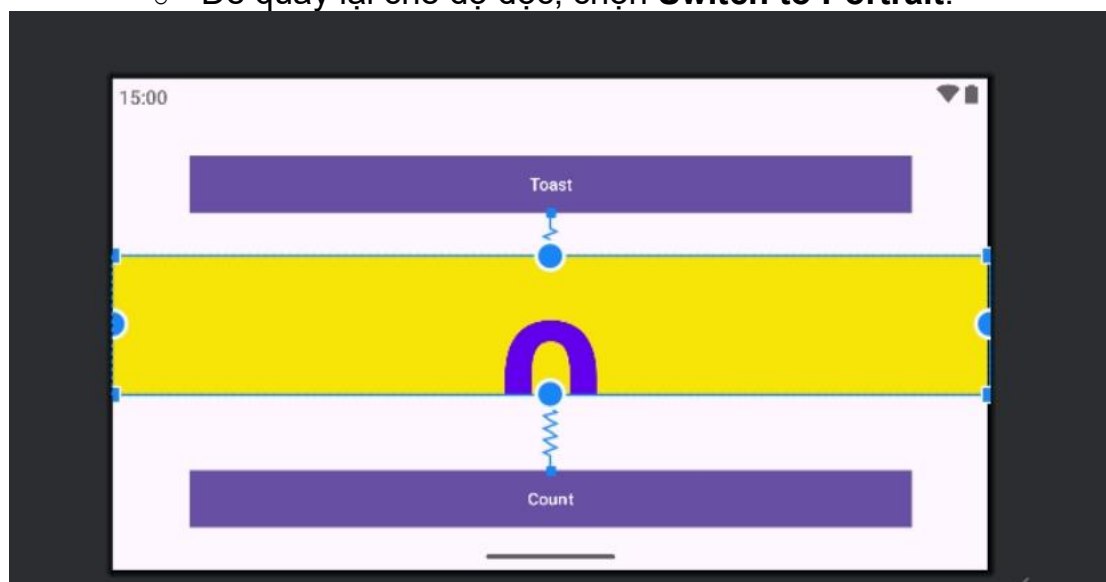
1.1 Xem trước bố cục ở chế độ ngang.

Để xem trước bố cục của ứng dụng **Hello Toast** ở chế độ ngang, thực hiện các bước sau:

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

Lưu ý: Nếu bạn đã tải xuống mã nguồn hoàn chỉnh của **HelloToast**, bạn cần xóa các bố cục **landscape** và **extra-large** đã hoàn thành trong nhiệm vụ này.

- Chuyển từ **Project > Android** sang **Project > Project Files** trong bảng **Project**.
 - Mở rộng thư mục: **app > app > src/main > res**.
 - Chọn cả hai thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**.
 - Chuyển bảng **Project** trở lại **Project > Android**.
2. Mở tệp **activity_main.xml**. Nhấp vào tab **Design** nếu nó chưa được chọn.
 3. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
 4. Chọn **Switch to Landscape** trong menu thả xuống. Bố cục sẽ hiển thị ở chế độ ngang như hình bên dưới.
 - Để quay lại chế độ dọc, chọn **Switch to Portrait**.



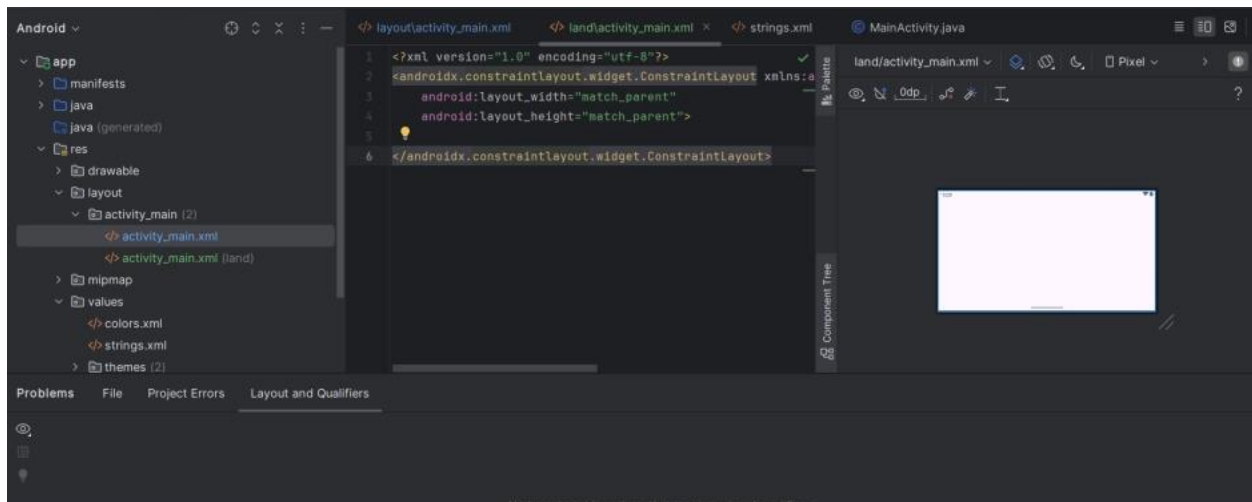
1.1 . Tạo biến thể bố cục cho chế độ ngang.

Sự khác biệt trực quan giữa chế độ dọc và ngang của bố cục này là chữ số **(0)** trong phần tử **TextView** của **show_count** quá thấp so với chế độ ngang—nó nằm

quá gần nút **Count**. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử **TextView** có thể hiển thị quá lớn hoặc không được căn giữa do kích thước văn bản cố định ở **160sp**.

Để khắc phục vấn đề này trong chế độ ngang mà không ảnh hưởng đến chế độ dọc, bạn có thể tạo một biến thể của bố cục ứng dụng **Hello Toast** dành riêng cho chế độ ngang. Thực hiện các bước sau:

1. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
2. Chọn **Create Landscape Variation**.
 - Một cửa sổ chỉnh sửa mới sẽ mở với tab **land/activity_main.xml**, hiển thị bố cục cho chế độ ngang.
 - Bạn có thể thay đổi bố cục này mà không làm thay đổi bố cục gốc ở chế độ dọc.
3. Trong bảng **Project > Android**, mở thư mục **res > layout**.
 - Bạn sẽ thấy **Android Studio** đã tự động tạo biến thể có tên **activity_main.xml (land)**.



1.2 Xem trước bố cục trên các thiết bị khác nhau.

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

1. Tab **land/activity_main.xml** **vẫn nên được 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 trên cùng.

3. Chọn một thiết bị khác trong menu thả xuống. Ví dụ:
 - Chọn **Nexus 4**, **Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong bản xem trước.
 - Những khác biệt này chủ yếu do kích thước văn bản cố định của **TextView**.

1.3 Thay đổi bố cục cho chế độ ngang.

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

Hình minh họa phía trên hiển thị:

1. **Tab Preview**, dùng để hiển thị khung xem trước.
2. **Khung xem trước**.
3. **Mã XML**.

Thực hiện thay đổi bố cục bằng các bước sau:

1. **Tab** `land/activity_main.xml` **vẫn nên được 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 tab **Text** và tab **Preview** (nếu chưa được chọn).
3. Tìm phần tử **TextView** trong mã XML.
4. Thay đổi thuộc tính:
5. Chọn các thiết bị khác nhau trong menu **Device in Editor** để kiểm tra bố cục trên nhiều thiết bị ở chế độ ngang.
6. **Chạy ứng dụng trên trình giả lập hoặc thiết bị thật** và **chuyển đổi giữa chế độ dọc và ngang** để quan sát cả hai bố cục.

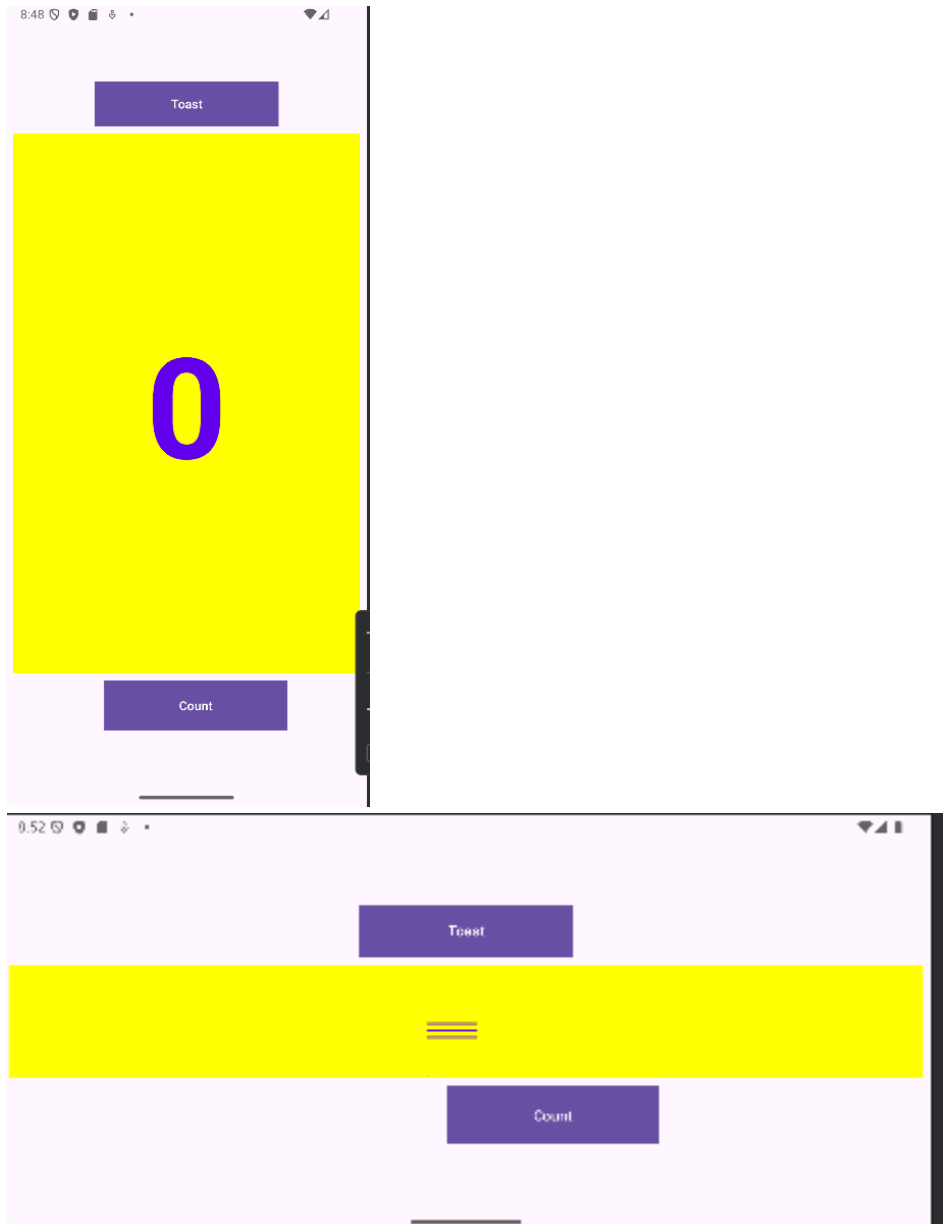
1.4 Tạo một biến thể bố cục cho máy tính bảng.

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

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

- Văn bản trên mỗi **Button** quá nhỏ.
- Việc sắp xếp các phần tử **Button** ở trên và dưới không tối ưu cho màn hình lớn.

Để khắc phục điều này mà không ảnh hưởng đến bố cục trên điện thoại ở chế độ dọc và ngang, bạn có thể tạo một biến thể bố cục riêng dành cho máy tính bảng.



- **Thực hiện các bước sau:**

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị khung thiết kế và bản vẽ bố cục.
2. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
3. Chọn **Create layout x-large Variation**.
 - Một cửa sổ chỉnh sửa mới sẽ mở với tab **xlarge/activity_main.xml**, hiển thị bố cục dành riêng cho thiết bị có màn hình lớn.

- Trình chỉnh sửa cũng tự động chọn một thiết bị máy tính bảng như **Nexus 9** hoặc **Nexus 10** để xem trước.
- Bạn có thể thay đổi bố cục này mà không ảnh hưởng đến các bố cục khác.

1.5 Thay đổi biến thể bố cục máy tính bảng

Bạn có thể sử dụng bảng **Attributes** trong tab **Design** để thay đổi thuộc tính cho bố cục này.

1. Thực hiện các bước sau:

2. **Tắt công cụ Autoconnect** trong thanh công cụ. Đảm bảo rằng công cụ này đã được vô hiệu hóa.
3. **Xóa tất cả các ràng buộc (constraints)** trong bố cục bằng cách nhấp vào nút **Clear All Constraints** trên thanh công cụ.
 - Khi đã xóa các ràng buộc, bạn có thể tự do di chuyển và thay đổi kích thước các phần tử trong bố cục.
4. **Di chuyển và thay đổi kích thước phần tử**
 - Trình chỉnh sửa bố cục cung cấp tay cầm thay đổi kích thước ở bốn góc của một phần tử.
 - Trong **Component Tree**, chọn **TextView** có ID là `show_count`.
 - Để dọn chỗ và dễ dàng kéo các phần tử **Button**, kéo một góc của **TextView** để thay đổi kích thước của nó.

Lưu ý: Việc thay đổi kích thước một phần tử sẽ gán cứng (hardcode) giá trị width và height. Tránh đặt kích thước cố định vì nó có thể không hiển thị đúng trên các màn hình có kích thước và mật độ khác nhau. Bạn chỉ đang tạm thời di chuyển phần tử này và sẽ chỉnh sửa lại kích thước sau.

5. Thay đổi thuộc tính của Button "Toast"

- Trong **Component Tree**, chọn **Button** có ID `button_toast`.
- Mở tab **Attributes**, thay đổi các giá trị sau:
 - `textSize` → **60sp**
 - `layout_width` → **wrap_content**
- Ở phía bên phải, bạn có thể sử dụng **view inspector** để thay đổi **layout_width** thành `wrap_content`.

Sử dụng `wrap_content` giúp Button có thể tự động điều chỉnh kích thước nếu văn bản bên trong thay đổi khi được dịch sang ngôn ngữ khác.

6. Thay đổi thuộc tính của Button "Count"

- Trong **Component Tree**, chọn **Button** có ID `button_count`.
- Thay đổi các giá trị sau:
 - `textSize` → **60sp**

- layout_width → **wrap_content**
- Kéo Button này lên trên **TextView** đến một vị trí trống trong bố cục.

1.7 Sử dụng ràng buộc đường cơ sở.

Bạn có thể căn chỉnh một phần tử giao diện người dùng chứa văn bản (chẳng hạn như **TextView** hoặc **Button**) với một phần tử văn bản khác bằng cách sử dụng **Baseline Constraint**. Điều này giúp văn bản trong các phần tử này được căn chỉnh chính xác theo cùng một đường cơ sở.

1. Thực hiện các bước sau:
2. Thêm ràng buộc vị trí cho các nút **Button**:
 - Ràng buộc **button_toast** vào mép trên và bên trái của bố cục.
 - Kéo **button_count** đến gần **button_toast**, sau đó ràng buộc **button_count** vào mép trái của **button_toast**.
3. Sử dụng **Baseline Constraint**:
 - Chọn phần tử **button_count**.
 - Di chuyển con trỏ chuột lên phần tử này cho đến khi biểu tượng **baseline constraint** xuất hiện bên dưới.
 - Nhấp vào nút **baseline constraint** để hiển thị tay cầm căn chỉnh màu xanh lá cây.
 - Kéo tay cầm căn chỉnh đến đường cơ sở của **button_toast**.

1.1 Mở rộng các Button theo chiều ngang.

Bạn có thể sử dụng nút **Pack** trên thanh công cụ để sắp xếp hoặc mở rộng các phần tử UI theo chiều ngang, giúp các **Button** được căn đều trong bố cục.

1. Thực hiện các bước sau:
2. Trong **Component Tree**, chọn **button_count**.
3. Giữ phím **Shift**, sau đó chọn **button_toast** để chọn cả hai Button.
4. Nhấp vào **Pack** trên thanh công cụ và chọn **Expand Horizontally**.
 - Lúc này, các Button sẽ mở rộng theo chiều ngang để lấp đầy bố cục.
5. Hoàn thiện bố cục:
 - Ràng buộc **show_count (TextView)** vào phía dưới của **button_toast**, cũng như vào hai cạnh bên và cạnh dưới của bố cục.
6. Chỉnh sửa thuộc tính của **show_count**:
 - Đặt layout_width và layout_height thành **Match Constraints**.
 - Thay đổi textSize thành **200sp**.
7. Xem trước ở chế độ ngang:
 - Nhấp vào **Orientation in Editor** trên thanh công cụ.
 - Chọn **Switch to Landscape** để xem bố cục máy tính bảng trong chế độ ngang.

- Chọn **Switch to Portrait** để quay lại chế độ dọc.
8. **Chạy ứng dụng trên nhiều thiết bị khác nhau:**
- Chạy ứng dụng trên các trình giả lập khác nhau.
 - Chuyển đổi giữa chế độ ngang và dọc để kiểm tra giao diện trên các thiết bị có kích thước màn hình và mật độ điểm ảnh khác nhau.

Coding challenge 1

Thử thách: Để phù hợp với chế độ ngang (landscape) của máy tính bảng, bạn có thể căn giữa các phần tử **Button** trong **activity_main.xml (xlarge)** để chúng hiển thị như trong hình dưới đây.

Gợi ý: Chọn các phần tử, nhấp vào nút **Align** trên thanh công cụ và chọn **Center Horizontally**.

*a) Nhiệm vụ 2 : thay đổi bố cục thành **LinearLayout***

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

LinearLayout yêu cầu các thuộc tính sau:

- **layout_width**
- **layout_height**
- **orientation**

Các giá trị có thể có cho **layout_width** và **layout_height**:

- **match_parent**: Mở rộng view để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao. Khi **LinearLayout** là root view, nó sẽ mở rộng theo kích thước của màn hình (phần tử cha).
- **wrap_content**: Thu nhỏ kích thước của view sao cho nó chỉ vừa đủ để chứa nội dung của nó. Nếu không có nội dung, view sẽ trở nên vô hình.
- **Giá trị cố định theo dp (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.

Các giá trị có thể có cho **orientation**:

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

- **Thay đổi nhóm bố cục gốc thành LinearLayout**

Trong nhiệm vụ này, bạn sẽ thay đổi nhóm bố cục gốc **ConstraintLayout** của ứng dụng **Hello Toast** thành **LinearLayout** để thực hành sử dụng **LinearLayout**.

2.1 Thay đổi nhóm bố cục gốc thành LinearLayout

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở) và nhấp vào tab **Text** ở cuối khung chỉnh sửa để xem mã XML. Ở đầu mã XML có dòng thẻ sau:
3. Thay đổi thẻ **<android.support.constraint.ConstraintLayout** thành **<LinearLayout**, để mã trông như sau:

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

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


```
</> activity_main.xml × MainActivity.java
1 <?xml version="1.0" encoding="utf-8"?>
2 © <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/main"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
```

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

2.2 Thay đổi thuộc tính phần tử cho LinearLayout

Làm theo các bước sau để thay đổi thuộc tính của phần tử giao diện người dùng sao cho phù hợp với **LinearLayout**:

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

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="210dp"
    android:layout_height="51dp"
```

4. Xóa các thuộc tính sau khỏi phần tử **button_toast**
5. Tìm phần tử **button_count** và thay đổi thuộc tính sau
6. Xóa các thuộc tính sau khỏi phần tử **button_count**
7. Tìm phần tử **show_count** (**TextView**) và thay đổi các thuộc tính sau
8. Xóa các thuộc tính sau khỏi phần tử **show_count**

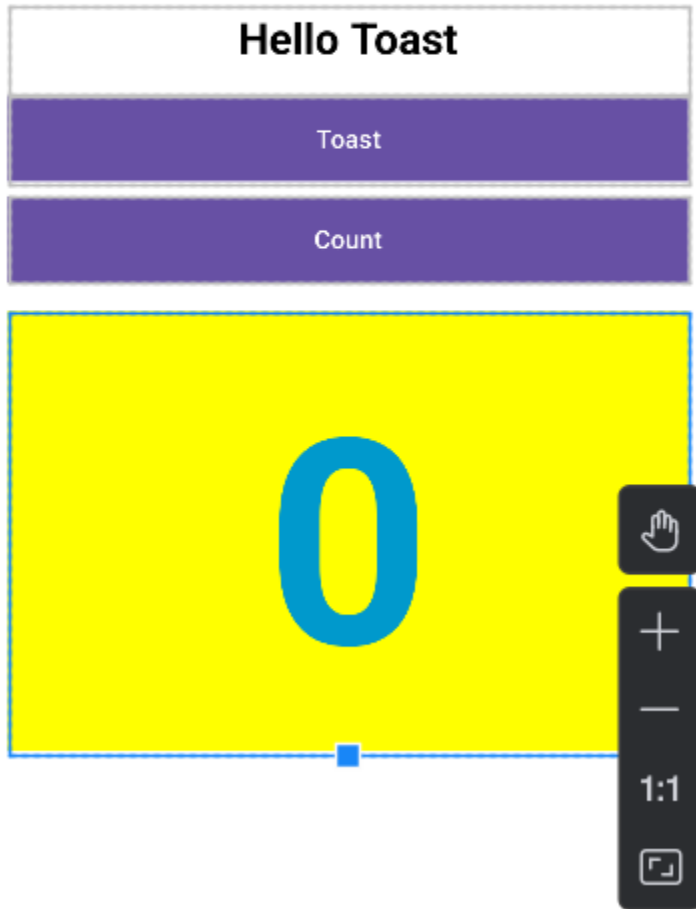
<Button

```
    android:id="@+id/button_count"
    android:layout_width="209dp"
    android:layout_height="57dp"
    android:layout_marginStart="64dp"
    android:layout_marginEnd="64dp"
    android:layout_marginBottom="64dp"
    android:background="@android:color/holo_blue_dark"
    android:onClick="countUp"
    android:text="Count"
    android:textColor="@android:color/white"
    tools:ignore="OnClick" />
```

<TextView

```
    android:id="@+id/show_count"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_margin="8dp"
    android:background="#FFFF00"
    android:gravity="center"
    android:text="0"
    android:textAlignment="center"
    android:textColor="@color/colorPrimary"
    android:textSize="160sp"
    android:textStyle="bold" />
```

9. Nhấp vào tab **Preview** ở bên phải cửa sổ **Android Studio** (nếu chưa được chọn) để xem trước bố cục cho đến thời điểm hiện tại.



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

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

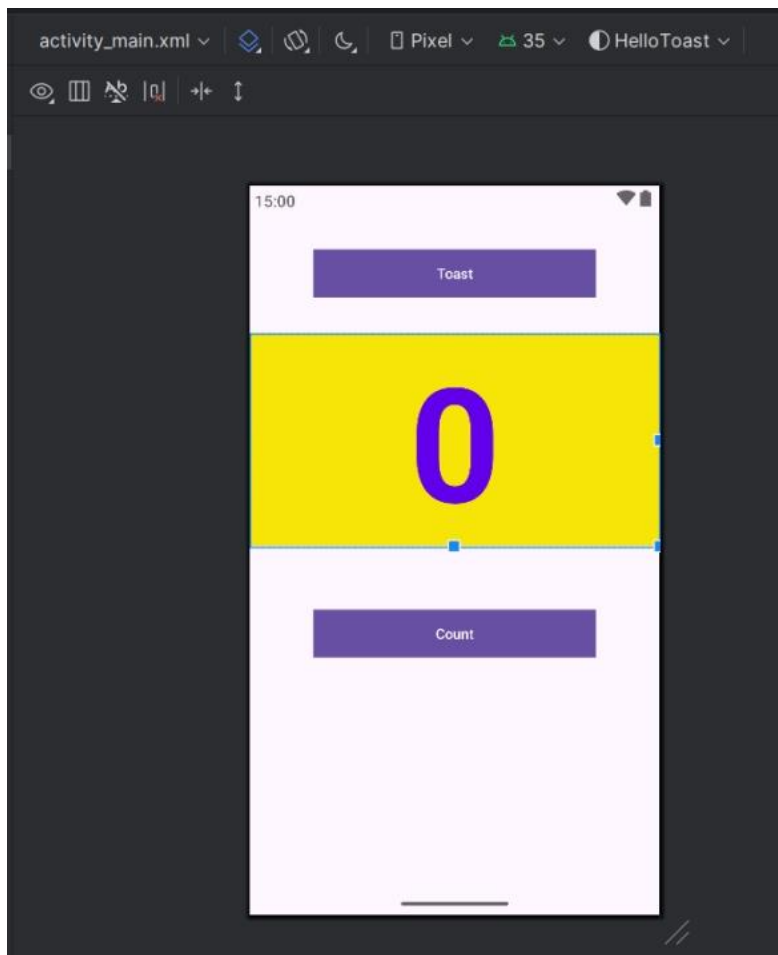
Để thay đổi vị trí sao cho nút **Count** nằm ở dưới cùng, hãy làm theo các bước sau:

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước.

2. Mở tệp **activity_main.xml** (nếu chưa mở) và nhấp vào tab **Text**.
3. Chọn phần tử **button_count (Button)** và tắt cả thuộc tính của nó, từ thẻ `<Button>` đến thẻ đóng `/>`, sau đó chọn **Edit > Cut**.
4. Nhấp sau thẻ đóng `/>` của phần tử **TextView** nhưng trước thẻ đóng `</LinearLayout>`, sau đó chọn **Edit > Paste**.
5. (Tùy chọn) Để sửa lỗi thụt lề hoặc khoảng cách cho mã XML, chọn **Code > Reformat Code** để định dạng lại với khoảng cách và thụt lề phù hợp.

Bằng cách di chuyển nút **button_count (Button)** xuống dưới **TextView**, bố cục hiện tại gần giống với bố cục ban đầu, với nút **Count** nằm ở phía dưới.

Bản xem trước của bố cục hiện tại trông như sau:



1.1 Thêm thuộc tính weight cho phần tử TextView

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

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

```
<TextView
    android:id="@+id/tvShowCount"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="36dp"
    android:layout_marginBottom="64dp"
    android:background="#F8E508"
    android:foregroundGravity="center"
    android:gravity="center|center_vertical"
    android:text="@string/_0"
    android:textColor="@color/design_default_color_primary"
    android:textSize="160sp"
    android:textStyle="bold" />
```

Thuộc tính **android:layout_weight** chỉ định lượng không gian bổ sung trong **LinearLayout** sẽ được phân bổ cho **View**. Nếu chỉ có một **View** có thuộc tính này, nó sẽ chiếm toàn bộ không gian còn lại. Nếu nhiều **View** có thuộc tính **weight**, không gian sẽ được chia theo tỷ lệ. Ví dụ: nếu hai **Button** có **weight = 1** và **TextView** có **weight = 2** (tổng cộng là 4), mỗi **Button** nhận $\frac{1}{4}$ không gian, trong khi **TextView** nhận một nửa.

Trên các thiết bị khác nhau, bố cục có thể hiển thị phần tử **show_count (TextView)** với kích thước thay đổi giữa hai nút **Toast** và **Count**. Để đảm bảo **TextView** mở rộng để lấp đầy không gian trống trên mọi thiết bị, bạn cần chỉ định thuộc tính **android:gravity** cho **TextView**.

- **Thực hiện:**

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), sau đó nhấp vào tab **Text**.

3. Tìm phần tử **show_count** (**TextView**) và thêm thuộc tính sau:

```
<TextView
    android:id="@+id/tvShowCount"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginTop="36dp"
    android:layout_marginBottom="64dp"
    android:background="#F8E508"
    android:foregroundGravity="center"
    android:gravity="center|center_vertical"
    android:text="@string/_0"
    android:textColor="@color/design_default_color_primary"
    android:textSize="160sp"
    android:textStyle="bold"
    android:layout_weight="1"
/>
```

Sau khi thực hiện, bản xem trước sẽ hiển thị như hình bên dưới.

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

*a) Nhiệm vụ 3: Thay đổi bố cục sang **RelativeLayout***

RelativeLayout là một nhóm giao diện trong đó mỗi thành phần được định vị và căn chỉnh tương đối với các thành phần khác trong cùng nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục bằng **RelativeLayout**.

3.1 Thay đổi **LinearLayout thành **RelativeLayout****

Một cách dễ dàng để thay đổi **LinearLayout** thành **RelativeLayout** là chỉnh sửa các thuộc tính XML trong tab **Text**.

1. Mở tệp **activity_main.xml** và nhấp vào tab **Text** ở cuối bảng chỉnh sửa để xem mã XML.
2. Thay đổi `<LinearLayout` ở đầu thành `<RelativeLayout>`.

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

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

3.2 Sắp xếp lại các thành phần trong RelativeLayout

Một cách dễ dàng để sắp xếp và định vị các thành phần trong **RelativeLayout** là chỉnh sửa các thuộc tính XML trong tab **Text**.

1. Nhấp vào tab **Preview** ở bên cạnh trình chỉnh sửa (nếu chưa được chọn) để xem trước bố cục.

Sau khi thay đổi sang **RelativeLayout**, trình chỉnh sửa bố cục cũng đã tự động thay đổi một số thuộc tính của thành phần giao diện, cụ thể:

- Nút **Count** (**button_count**) đang chồng lên nút **Toast** (**button_toast**), khiến nút **Toast** không hiển thị.
 - Phần trên của **TextView** (**show_count**) đang chồng lên các nút **Button**.
2. Thêm thuộc tính **android:layout_below** vào nút **button_count** để đặt nút này ngay bên dưới **show_count** (**TextView**)
 3. Thêm thuộc tính **android:layout_centerHorizontal** vào **button_count** để căn giữa nút theo chiều ngang trong **RelativeLayout**.

```

<Button
    android:id="@+id/btnCount"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="64dp"
    android:layout_marginEnd="64dp"
    android:layout_marginBottom="36dp"
    android:background="@color/design_default_color_on_primary"
    android:text="@string/count"
    android:onClick="countUp"
    android:layout_below="@+id/tvShowCount"
    android:layout_centerHorizontal="true"
/>

```

4. Thêm các thuộc tính sau vào **show_count (TextView)**:
- **android:layout_alignParentLeft** để căn trái theo **RelativeLayout**.
 - **android:layout_alignParentStart** để căn chỉnh theo hướng ngôn ngữ, giúp hiển thị đúng trên các thiết bị có ngôn ngữ từ phải sang trái.

```

<TextView
    android:id="@+id/tvShowCount"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#F8E508"
    android:gravity="center|center_vertical"
    android:text="@string/_0"
    android:textColor="@color/design_default_color_primary"
    android:textSize="160sp"
    android:textStyle="bold"
    android:layout_below="@+id/btn_toast"
    android:layout_alignParentStart="true"
/>

```

5. Xóa thuộc tính **android:layout_weight="1"** khỏi **show_count**, vì thuộc tính này không áp dụng cho **RelativeLayout**.

Sau khi thực hiện các thay đổi, bố cục sẽ hiển thị đúng trong phần xem trước.

Gợi ý: **RelativeLayout** giúp dễ dàng định vị các thành phần trong bố cục. Để tìm hiểu thêm về cách định vị các thành phần trong **RelativeLayout**, hãy tham khảo tài liệu "**Positioning Views**" trong hướng dẫn về **RelativeLayout** của API.

1.4) Trình chỉnh sửa bố cục

Bài 1.3: Văn bản và chế độ xem cuộn

Giới thiệu

Lớp **TextView** là lớp con của lớp **View** hiển thị văn bản trên màn hình. Bạn có thể kiểm soát cách văn bản xuất hiện bằng các thuộc tính **TextView** trong tệp bố cục XML. Bài thực hành này cho thấy cách làm việc với nhiều phần tử **TextView**, bao gồm một phần tử mà người dùng có thể cuộn nội dung của nó theo chiều dọc.

Nếu bạn có nhiều thông tin hơn mức hiển thị trên màn hình của thiết bị, bạn có thể tạo chế độ xem cuộn để người dùng có thể cuộn theo chiều dọc bằng cách vuốt lên hoặc xuống hoặc theo chiều ngang bằng cách vuốt sang phải hoặc sang trái.

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

1.5) Văn bản và các chế độ cuộn

1.6) Tài nguyên có sẵn

Bài 2) Activities

2.1) Activity và Intent

2.2) Vòng đời của Activity và trạng thái

2.3) Intent ngầm định

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

3.1) Trình gỡ lỗi

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

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

Bài 2) Trải nghiệm người dùng thú vị

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

Bài 3) Kiểm thử giao diện người dùng

- 3.1) Espresso cho việc kiểm tra UI**

CHƯƠNG 3. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

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

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

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

CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

1.1) Shared preferences

1.2) Cài đặt ứng dụng

Bài 2) Lưu trữ dữ liệu với Room

2.1) Room, LiveData và ViewModel

2.2) Room, LiveData và ViewModel