

**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	33
1.3) Trình chỉnh sửa bố cục	66
1.4) Văn bản và các chế độ cuộn	66
1.5) Tài nguyên có sẵn	66
Bài 2) Activities.....	66
2.1) Activity và Intent.....	66
2.2) Vòng đời của Activity và trạng thái	66
2.3) Intent ngầm định	66
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	66
3.1) Trình gỡ lỗi.....	66
3.2) Kiểm thử đơn vị	66
3.3) Thư viện hỗ trợ	66
CHƯƠNG 2. Trải nghiệm người dùng.....	67
Bài 1) Tương tác người dùng	67
1.1) Hình ảnh có thể chọn.....	67
1.2) Các điều khiển nhập liệu	67
1.3) Menu và bộ chọn	67
1.4) Điều hướng người dùng	67
1.5) RecyclerView	67
Bài 2) Trải nghiệm người dùng thú vị.....	67
2.1) Hình vẽ, định kiểu và chủ đề	67
2.2) Thẻ và màu sắc	67
2.3) Bố cục thích ứng.....	67
Bài 3) Kiểm thử giao diện người dùng	67

3.1) Espresso cho việc kiểm tra UI	67
CHƯƠNG 3. Làm việc trong nền	67
Bài 1) Các tác vụ nền.....	67
1.1) AsyncTask.....	67
1.2) AsyncTask và AsyncTaskLoader	67
1.3) Broadcast receivers	67
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	67
2.1) Thông báo.....	67
2.2) Trình quản lý cảnh báo	67
2.3) JobScheduler	67
CHƯƠNG 4. Lưu dữ liệu người dùng	68
Bài 1) Tùy chọn và cài đặt.....	68
1.1) Shared preferences	68
1.2) Cài đặt ứng dụng	68
Bài 2) Lưu trữ dữ liệu với Room	68
2.1) Room, LiveData và ViewModel.....	68
2.2) Room, LiveData và ViewModel.....	68
3.1) Trình gowx lỗi	

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

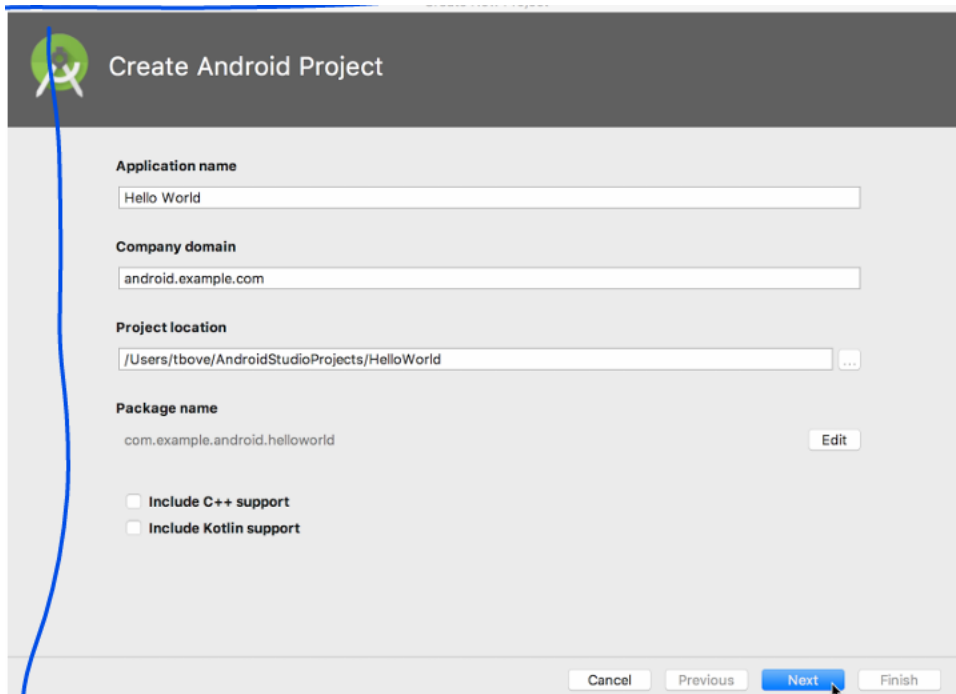
Giới thiệu

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 ứng dụng

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

Đây là ứng dụng sau khi hoàn thiện:

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 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, kiểm tra và hiệu suất giúp phát triển ứng dụng nhanh hơn và dễ dàng hơn. Bạn có thể kiểm tra ứng dụng của bạn với nhiều trình mô phỏng được cấu hình sẵn hoặc trên thiết bị di động của riêng bạn, hãy xây dựng ứng dụng và xuất bản 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. Bộ công cụ mới nhất OpenJDK(Java Development Kit) đi kèm với Android Studio.

Để thiết lập và chạy với Android Studio, đầu tiên hãy kiểm tra [system requirements](#) để đảm bảo rằng hệ thống của bạn đáp ứng chúng. Việc cài đặt tương tự cho tất cả các nền tảng. Bất kỳ sự khác biệt được ghi chú dưới đây.

1. Điều hướng đến [Android developers site](#) 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 quá trình cài đặt hoàn tất, Setup Wizard sẽ tải xuống và cài đặt thêm một số các thành phần bao gồm Android SDK. Hãy kiên nhẫn, việc này có thể mất một chút thời gian tùy thuộc vào tốc độ mạng của bạn và một số bước có thể có vẻ dư thừa.
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.

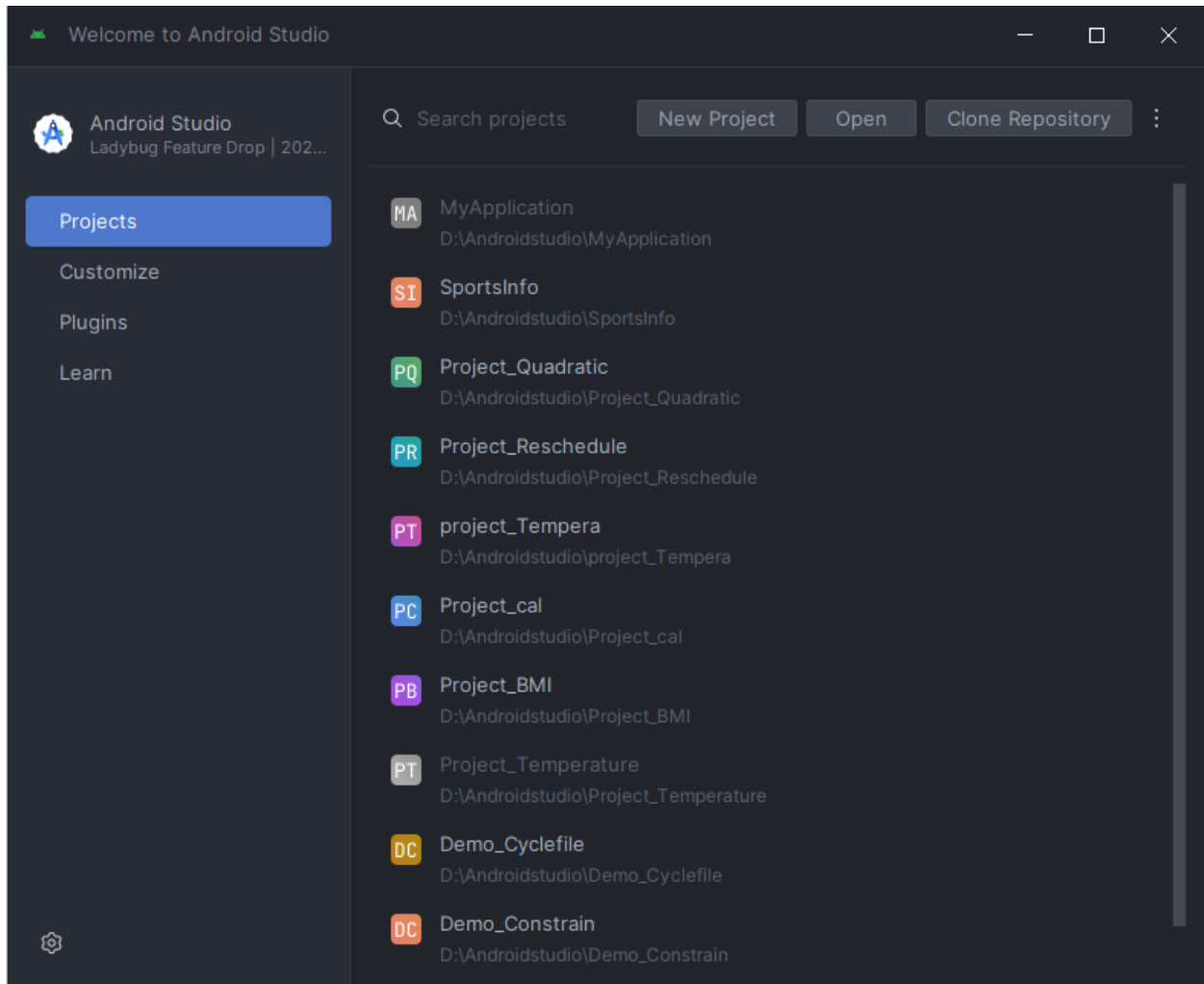
Xử lý sự cố: Nếu bạn gặp sự cố khi cài đặt, hãy kiểm tra [Android Studio release notes](#), hoặc nhờ người hướng dẫn trợ giúp.

Nhiệm vụ 2: Tạo ứng dụng Hello World

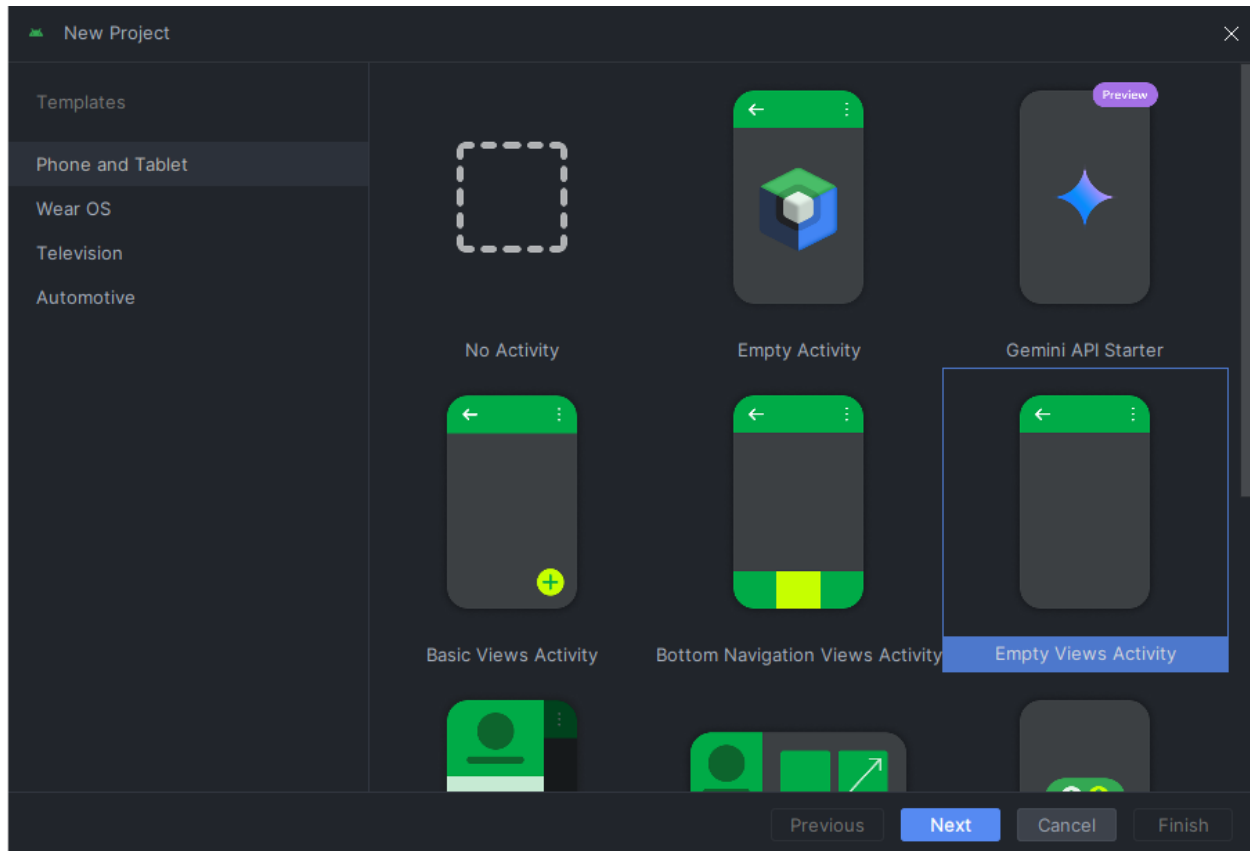
Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị "Hello World" để xác minh rằng Android Studio đã được cài đặt đúng cách và để tìm hiểu những điều cơ bản về phát triển bằng Android Studio.

2.1 Tạo dự án ứng dụng

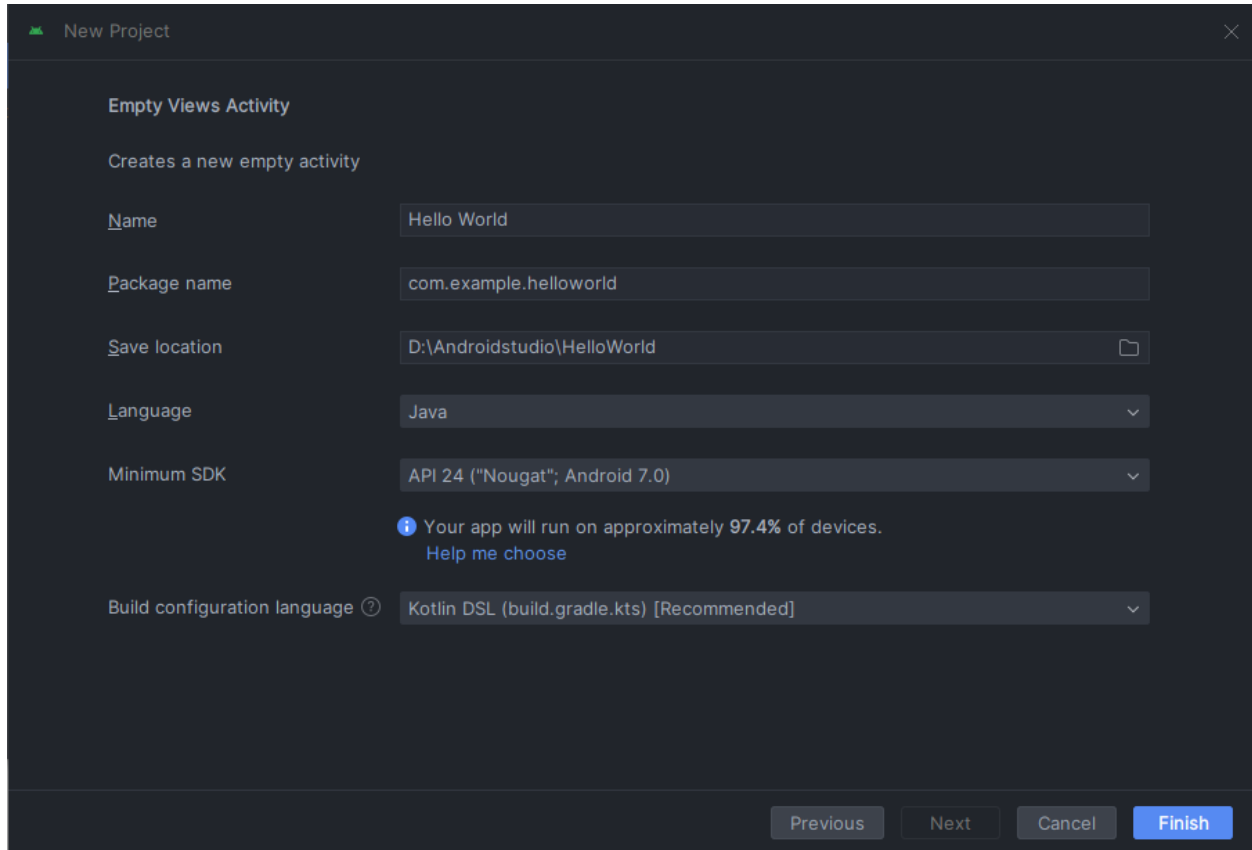
1. Mở Android Studio nếu nó chưa được mở.
2. Trong cửa sổ chính **Welcome to Android Studio**, hãy nhấp vào **New Project**.



3. Trong cửa sổ **New Project**. Đây là thành phần quan trọng của bất kỳ ứng dụng Android nào. Một Activity thường có một bố cục liên quan đến nó để xác định cách các thành phần UI xuất hiện trên màn hình. Android Studio cung cấp mẫu Activity để giúp bạn bắt đầu. Đối với dự án Hello World, hãy chọn **Empty Views Activity** như hiển thị bên dưới và nhấp vào **Next**.

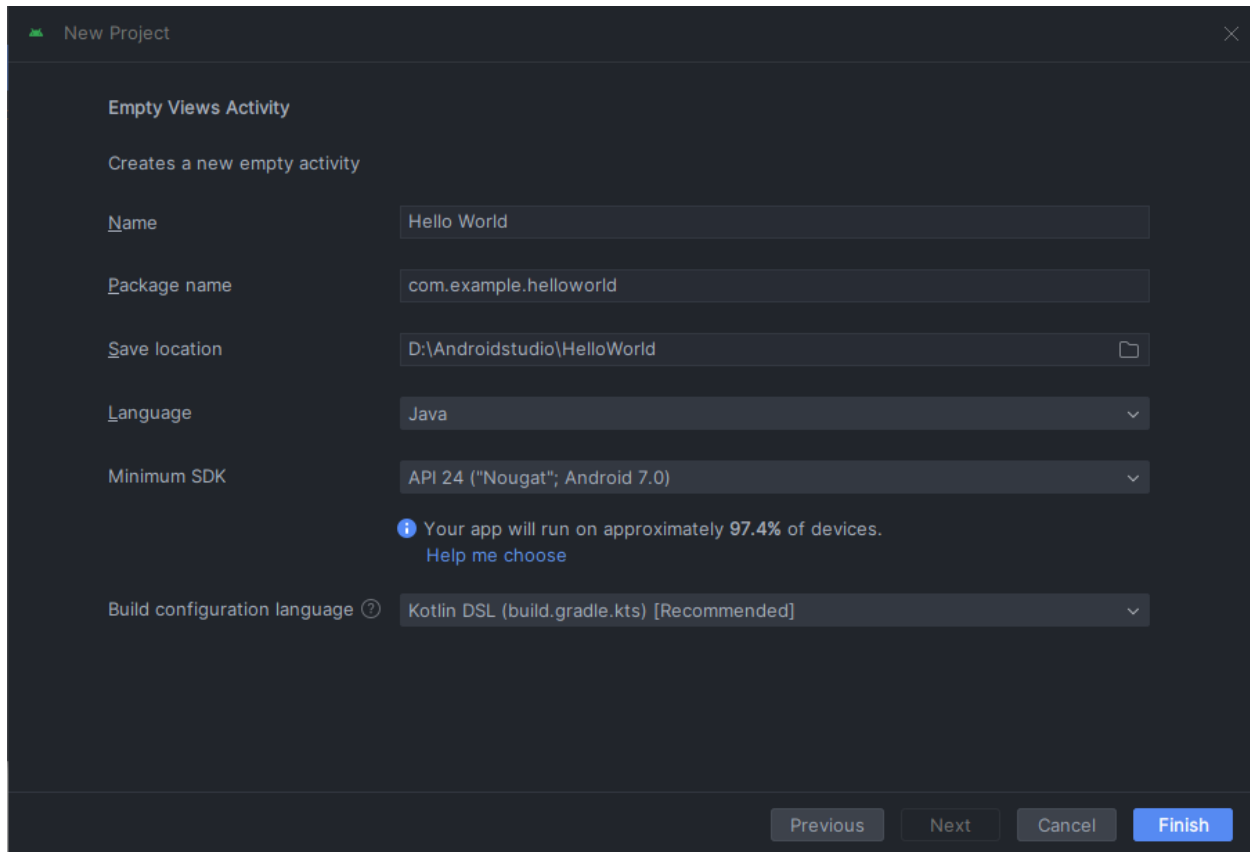


4. Trong cửa sổ **New Project**, nhập **Hello World** cho tên ứng dụng(**Name**).



5. Chấp nhận tên miền mặc định của Công ty **com.example.helloworld** hoặc tạo một tên miền Công ty độc quyền.
Nếu bạn không có kế hoạch xuất bản ứng dụng của mình, bạn có thể để mặc định. Lưu ý rằng việc thay đổi **Package name** của bạn sau này sẽ tốn thêm công sức
6. Xác định vị trí dự án mặc định **Save location** là nơi bạn muốn lưu trữ ứng dụng Hello Word và các dự án Android Studio khác hoặc thay đổi thành thư mục bạn muốn.

7. Tiếp theo hãy chọn **Language** là **Java**. Trong phần **Minimum SDK** được để mặc định là **API 24 (“Nougat”; Android 7.0)**, nếu bạn không muốn hãy bật danh sách Minimum SDK lên để chọn.



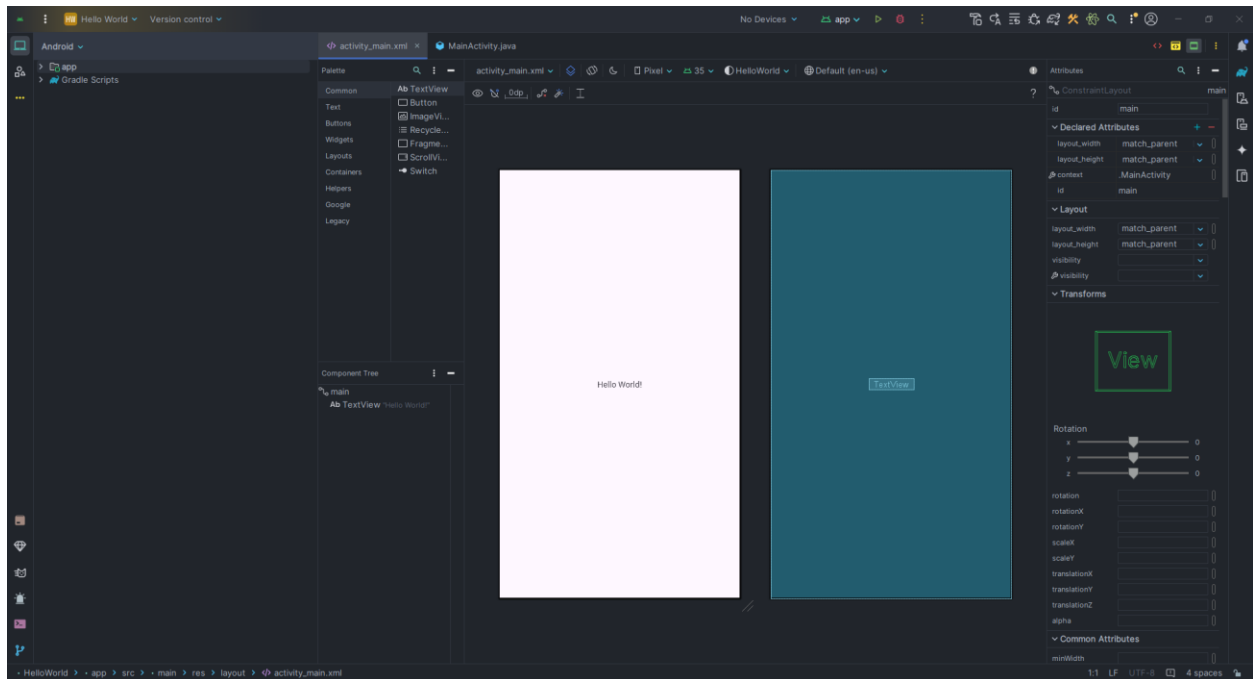
Các thiết lập này làm ứng dụng Hello Word của bạn tương thích với 97.4% thiết bị Android đang hoạt động trên cửa hàng Google Play tính tới thời điểm viết bài này.

8. Ở mục **Build configuration language** chọn **Kotlin DSL (build.gradle.kts)**. Sau đó nhấp vào **Finish**.

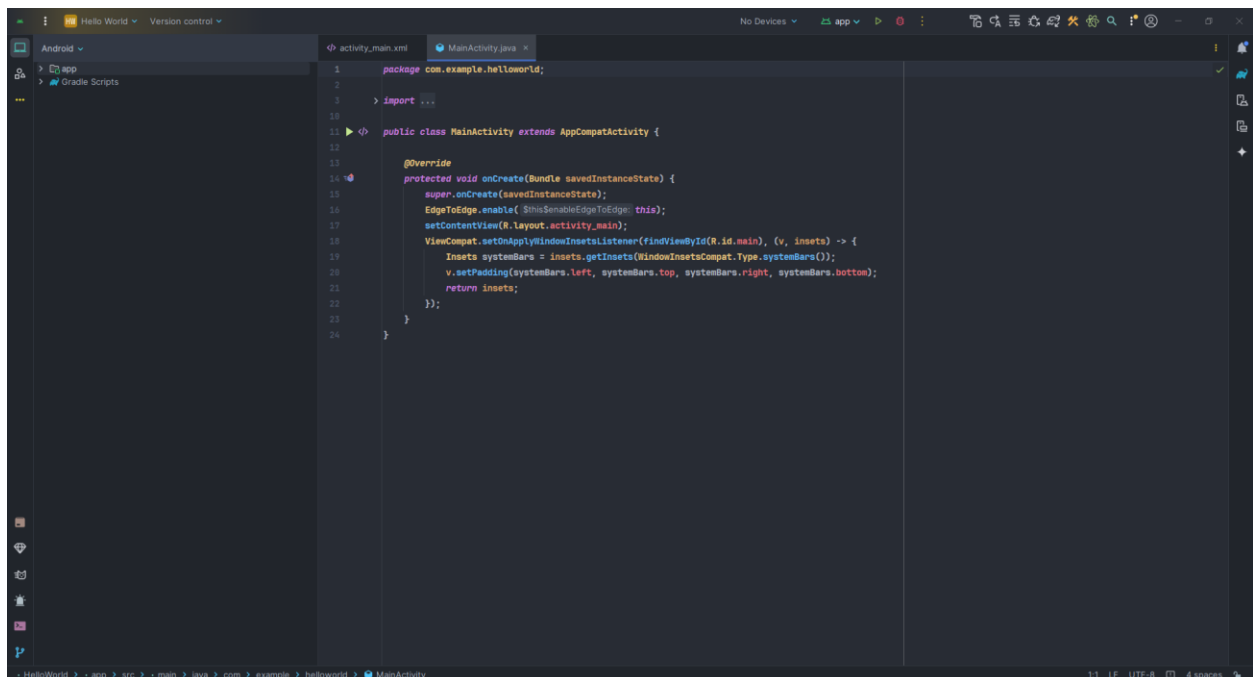
Android Studio tạo một thư mục cho các dự án của bạn và xây dựng dự án bằng **Gradle** (có thể mất vài phút).

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

1. Nhấp vào tab **activity_main.xml** để xem trình chỉnh sửa bố cục.
2. Nhấp vào tab **Design** của trình chỉnh sửa bố cục, nếu chưa chọn, để hiển thị bố cục của giao diện như minh họa bên dưới.



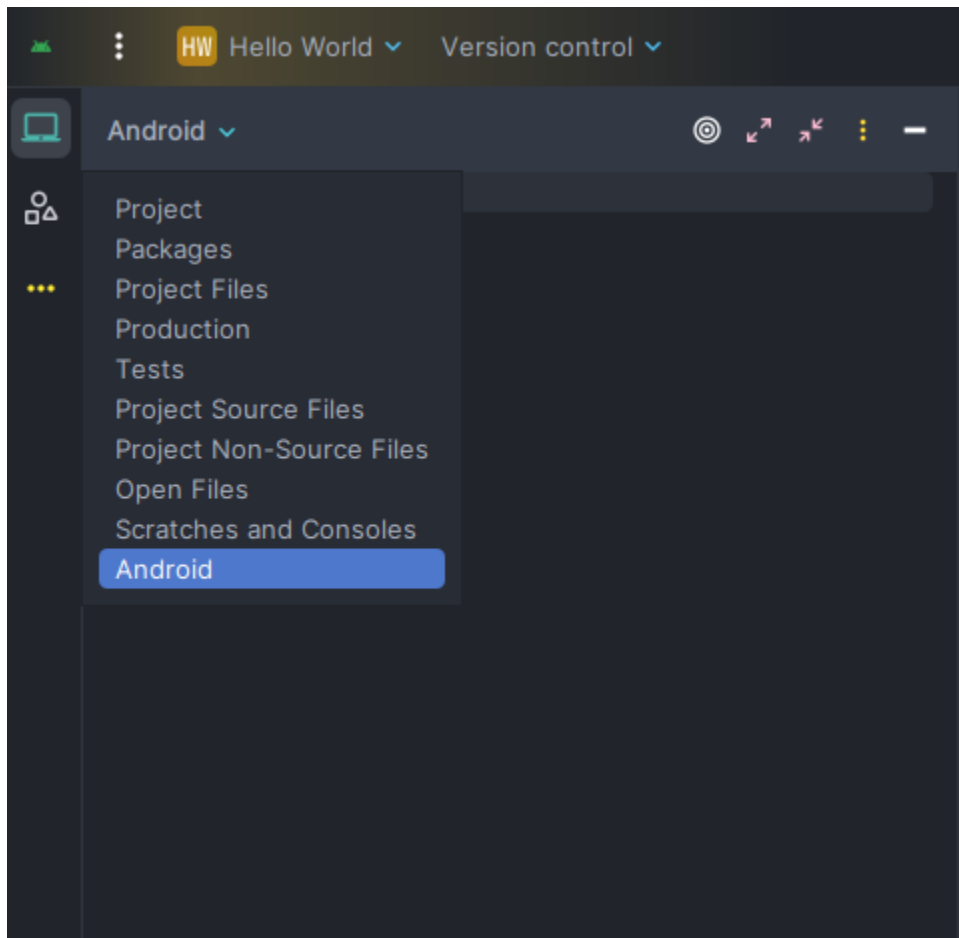
3. Nhấp vào tab **MainActivity.java** để xem trình soạn thảo mã như hiển thị bên dưới.



2.2 Khám phá dự án > Bảng điều khiển Android.

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

1. Nếu chưa được chọn, hãy nhấp vào tab **Project** trong cột tab dọc ở phía 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 trong ngăn Project, như hiển thị bên dưới.

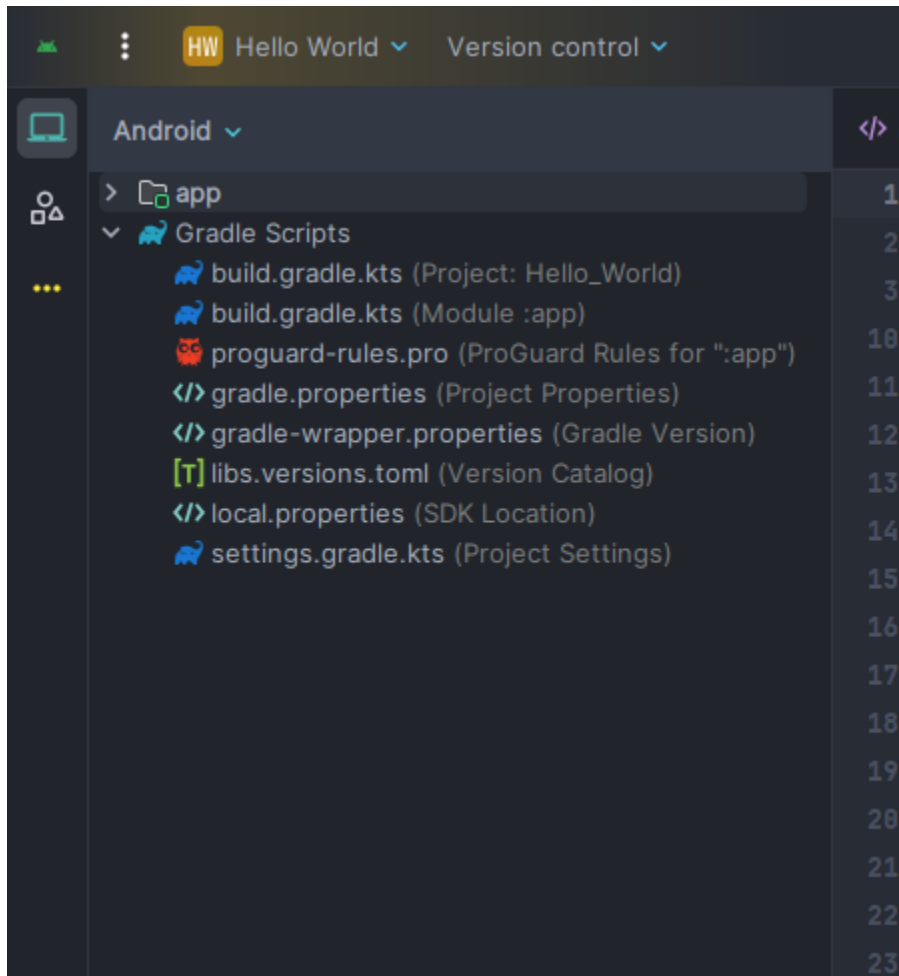


Lưu ý: Chương này và các chương khác đề cập đến ngăn Project, khi được đặt thành **Android**, là ngăn **Project > Android**.

2.3 Khám phá thư mục Gradle Scripts

Hệ thống xây 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 các phần phụ thuộc.

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



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

1. Nếu thư mục **Gradle Scripts** không được mở rộng, hãy nhấp vào hình tam giác để mở rộng thư mục. Thư mục này chứa tất cả các tệp cần thiết cho hệ thống xây dựng.
2. Hãy nhìn vào tập tin **build.gradle.kts (Project: Hello_World)**. 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

Gradle build 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 Google Maven repository) là các vị trí kho lưu trữ:

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

3. Hãy nhìn vào tệp tin **build.gradle.kts (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 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 phụ thuộc của tất cả các tệp `“.jar”` bên trong thư mục `libs`.

Hãy chọn vào tệp **build.gradle(Module:app)** cho ứng dụng HelloWorld:

```
plugins {  
    alias(libs.plugins.android.application)  
}  
  
android {  
    namespace = "com.example.helloworld"  
    compileSdk = 35  
}
```

```

defaultConfig {
    applicationId = "com.example.helloworld"
    minSdk = 24
    targetSdk = 35
    versionCode = 1
    versionName = "1.0"

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

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

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

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

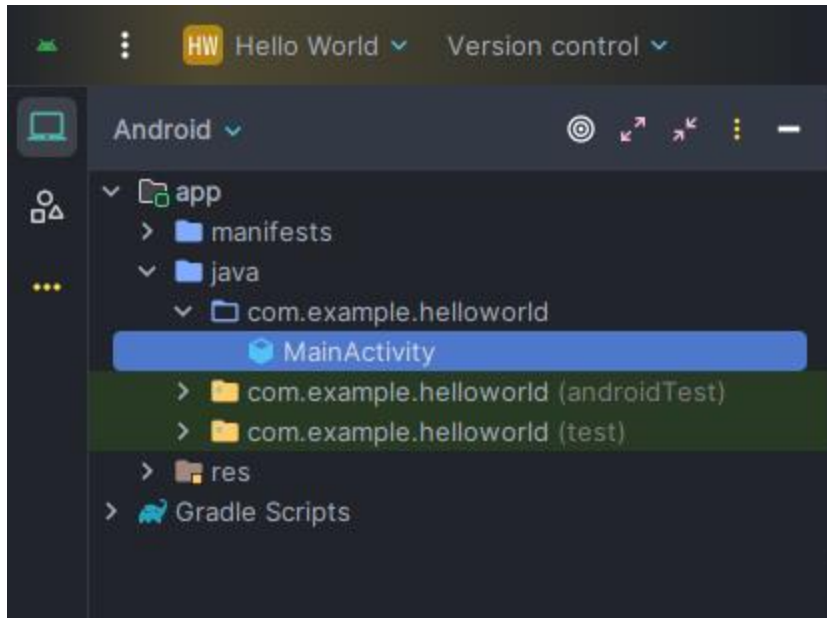
```

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

2.4 Khám phá ứng dụng và thư mục res

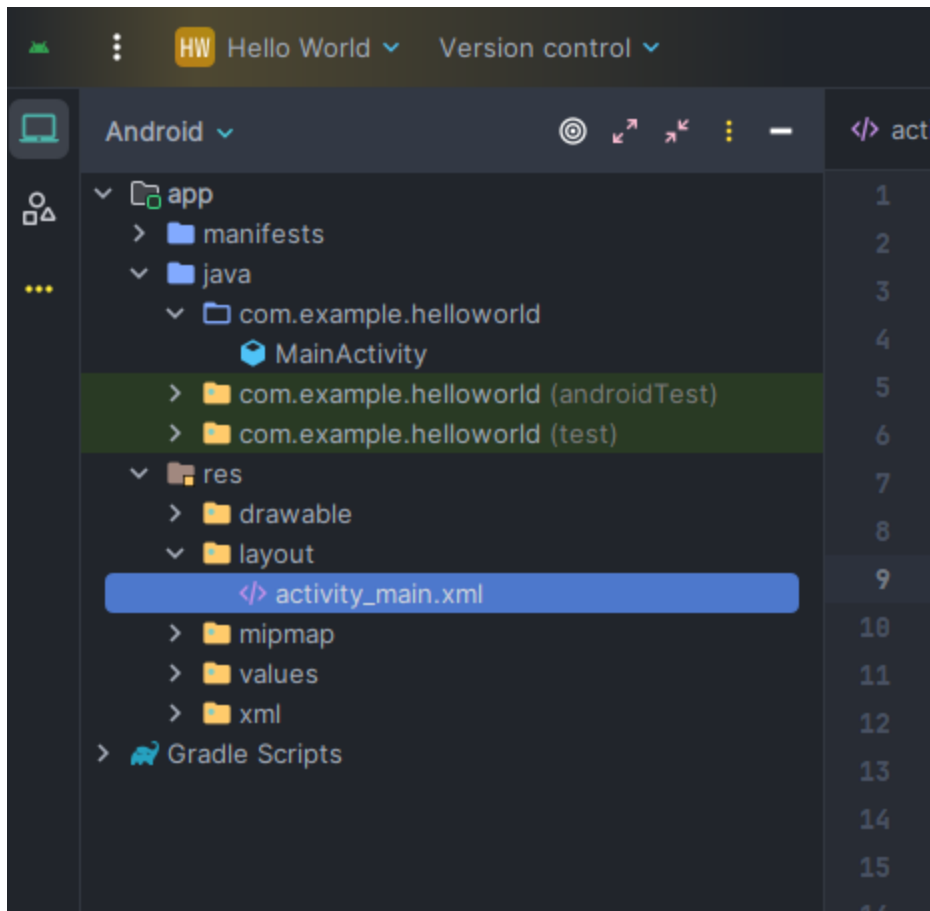
Tất cả mã và tài nguyên cho ứng dụng đều nằm trong thư mục app và res.

1. Mở rộng thư mục **app**, thư mục **java**, và thư mục **com.example.android.helloworld** để xem tệp java **MainActivity**. Đúp chuột vào tệp để mở tệp trong trình 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 **Project > 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 bố cục, chuỗi và hình ảnh. Một Hoạt động thường được liên kết với bố cục của chế độ xem UI được định nghĩa là tệp XML. Tệp này thường được đặt tên theo hoạt động 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 Hoạt động, 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 Overview.


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

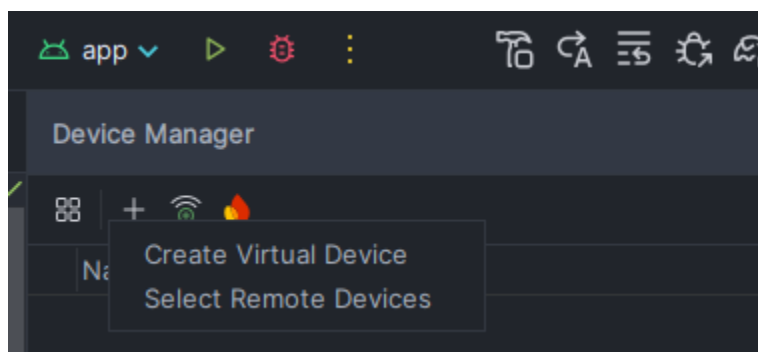
Trong nhiệm vụ này, bạn sẽ sử dụng **Android Virtual Device (AVD) manager** để tạo một thiết bị ảo (còn được gọi là trình giả lập) mô phỏng cấu hình cho một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng trình giả lập Android có các yêu cầu bổ sung ngoài các yêu cầu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

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

3.1 Create an Android virtual device (AVD)

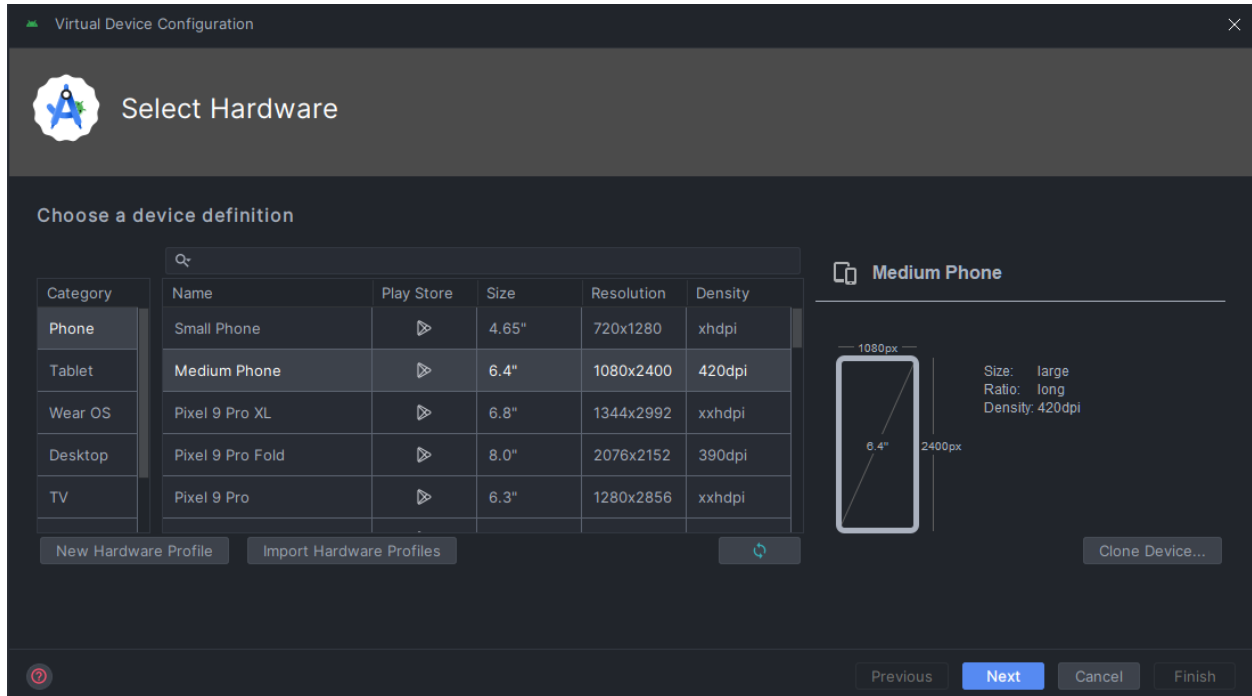
Để chạy trình giả lập trên máy tính, bạn phải tạo cấu hình mô tả thiết bị ảo.

1. Trong Android Studio, chọn **Tools > Android > AVD Manager**, hoặc nhấp chuột vào biểu tượng AVD Manager  trong thanh công cụ. Nếu chưa tạo máy ảo màn hình sẽ hiển thị như sau.



2. Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** sẽ xuất hiện, hiển thị danh sách các thiết bị 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**).

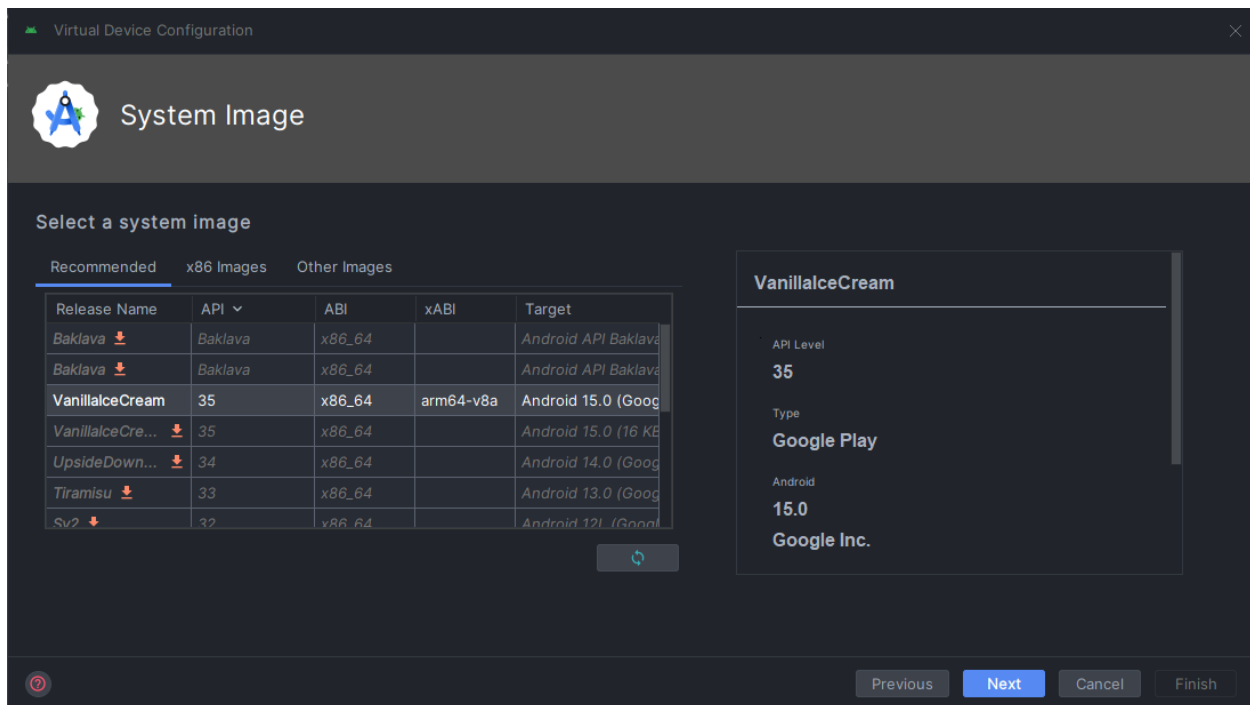


3. Chọn một thiết bị như **Medium Phone** và nhấp vào **Next**. Màn hình **System Image** sẽ xuất hiện.
4. Nhấp vào tab **Recommended** nếu chưa chọn và chọn phiên bản hệ thống Android để chạy trên thiết bị ảo (**VanillaIceCream**).

Có nhiều phiên bản khả dụng hơn những phiên bản được hiển thị trong tab **Recommended**. Hãy xem tab **x86 Images** và **Other Images** để xem chúng.

Nếu liên kết **Download** hiển thị bên cạnh ảnh hệ thống bạn muốn sử dụng, thì nó 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 một system image, 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 **Finish**.



3.2 Chạy ứng dụng trên thiết bị ảo

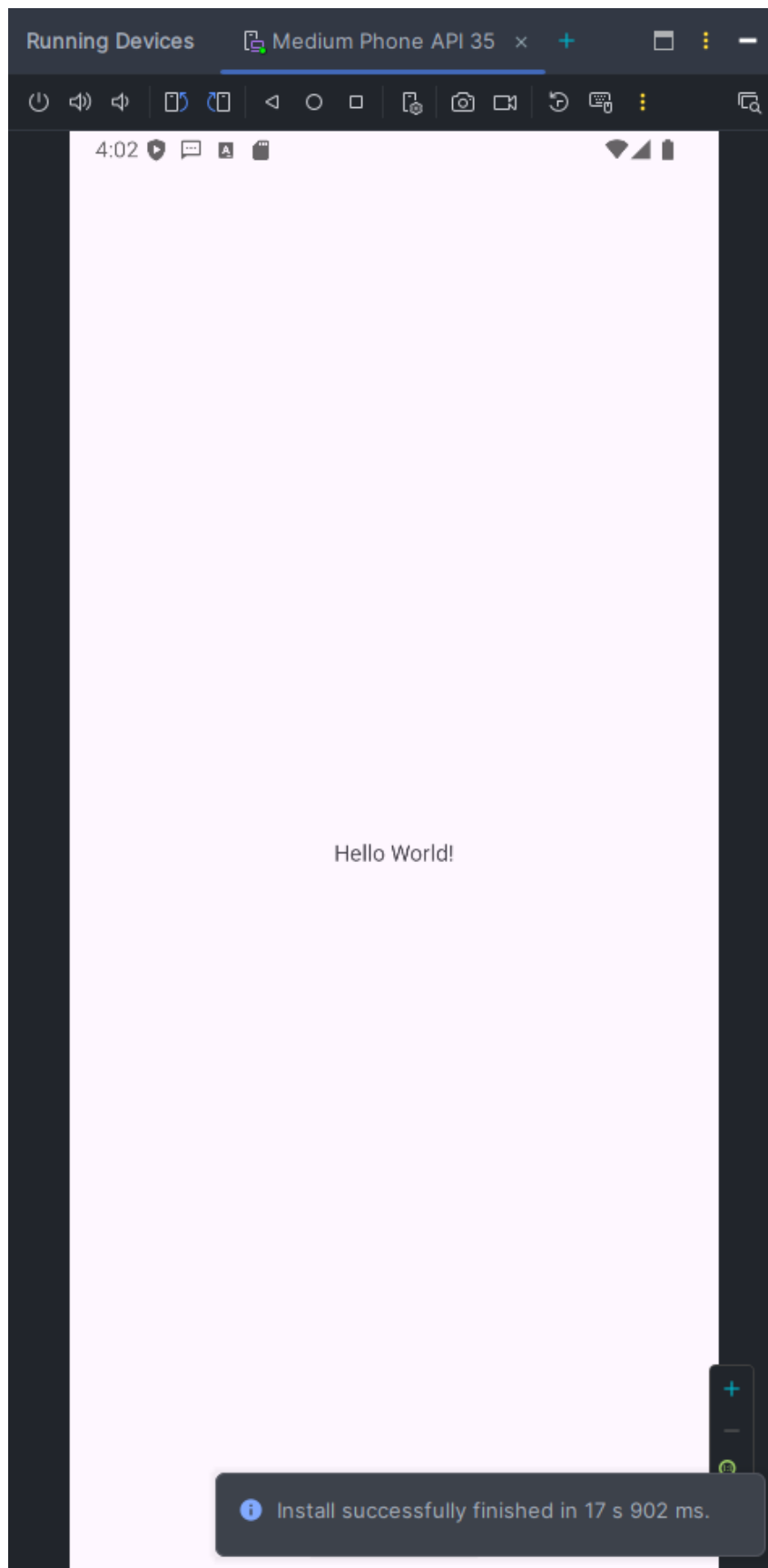
Trong nhiệm vụ này, cuối cùng bạn sẽ chạy ứng dụng Hello World của mình.

1. Trong Android Studio, chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run**  trên thanh công cụ.

Trình giả lập khởi động và khởi động giống như một thiết bị vật lý. Tùy thuộc vào tốc độ máy tính của bạn, quá trình này có thể mất một lúc. Ứng dụng của bạn 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 làm việc. Bạn không nên đóng thiết bị cho đến khi hoàn tất thử nghiệm ứng dụng, để ứng dụng không phải trải qua quá trình khởi động thiết bị một lần nữa. Để đóng thiết bị ảo, hãy nhấp vào nút **X** ở đầu trình giả lập, chọn **Quit** 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 nên luôn kiểm tra ứng dụng của mình trên cả thiết bị ảo và vật lý.

Những gì bạn cần:

- Một thiết bị Android như điện thoại hoặc máy tính bảng.
- Một 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 [Using Hardware Devices](#). Bạn cũng có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Đối với trình điều khiển USB dựa trên Windows, hãy xem [OEM USB Drivers](#).

4.1 Bật gỡ lỗi USB


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

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

1. Trên thiết bị của bạn, hãy mở **Settings**, tìm kiếm **About phone**, nhấp vào **About phone** và chạm vào **Build number** bảy lần.
2. Quay lại màn hình trước đó (**Settings / System**). **Developer options** xuất hiện trong danh sách. Nhấn vào **Developer options**.
3. Chọn **USB Debugging**.

4.2 Chạy ứng dụng của bạn trên thiết bị

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

1. Kết nối thiết bị của bạn với máy phát triển bằng cáp USB
2. Nhấp vào nút Run  trên thanh công cụ. Cửa sổ **Select Deployment Target** sẽ mở ra với danh sách các trình giả lập khả dụng và các thiết bị được kết nối.
3. Chọn thiết bị của bạn, và nhấp **OK**.

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

Xử lý sự cố

Nếu Android Studio không nhận ra thiết bị của bạn, có thể thử cách sau:

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

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

1. Rút phích cắm thiết bị
2. Trên thiết bị, mở **Developer Options in Settings app**.
3. Nhấn vào thu hồi quyền **USB Debugging**.
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. Tham khảo [Using Hardware Devices documentation](#).

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.

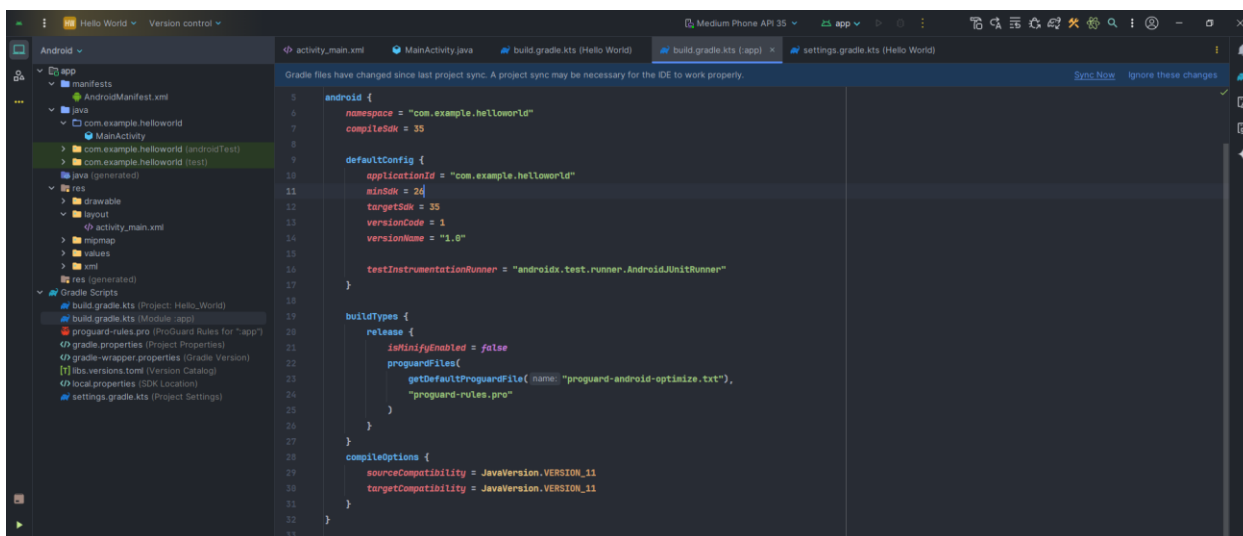
5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Làm theo các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu nó chưa được mở, và nhấp chuột vào tệp **build.gradle(Module:app)**.

Nội dung của tệp tin xuất hiện trong trình soạn thảo mã.

2. Trong khối `defaultConfig`, hãy thay đổi giá trị của `minSdk` thành 26 như hiển thị bên dưới (ban đầu giá trị này được đặt thành 24).



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

5.2 Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi đối với các tệp cấu hình xây 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 các tệp dự án, hãy nhấp vào **Sync Now** trên thanh thông báo xuất hiện khi thực hiện thay đổi (như thể hiện trong hình trước) hoặc nhấn vào biểu tượng **Sync Project with Gradle Files**  trong 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.

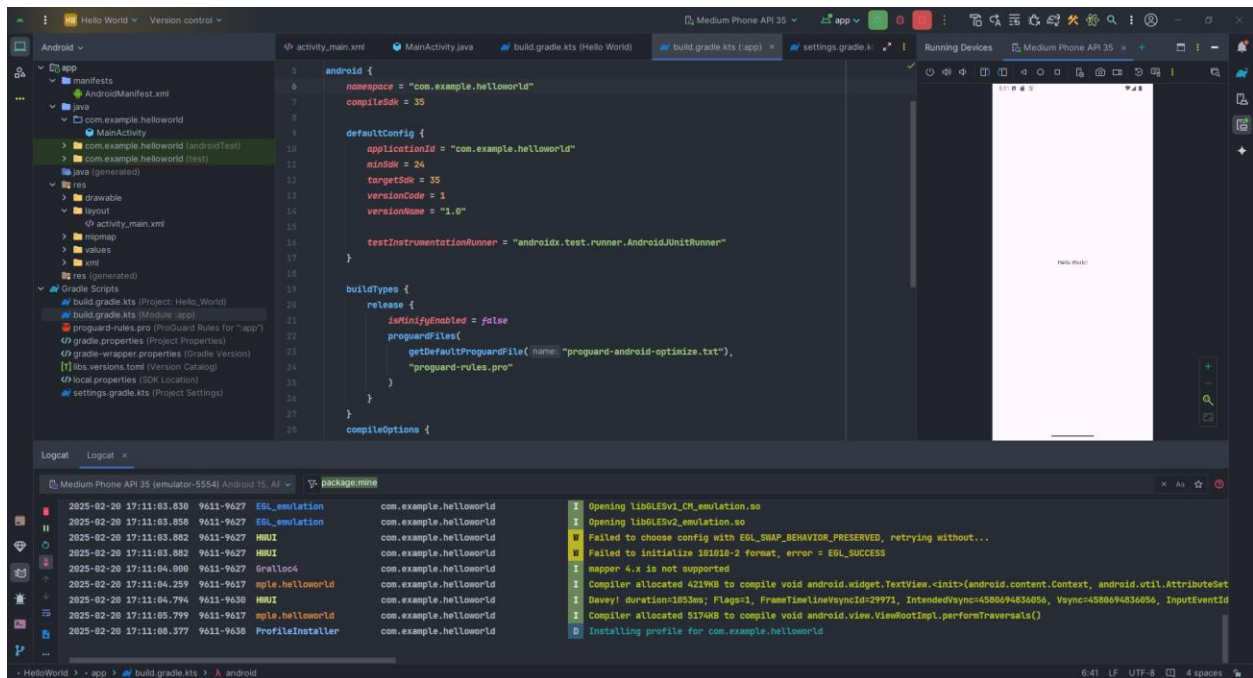
Để hiểu sâu hơn về Gradle, hãy tham khảo tài liệu [Build System Overview](#) và [Configuring Gradle Builds](#).

Nhiệm vụ 6: Thêm log statement vào ứng dụng của bạn

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

6.1 Xem ngăn Logcat

Để xem ngăn **Logcat**, hãy nhấp vào biểu tượng **Logcat**  ở thanh công cụ bên trái màn hình Android Studio như minh họa trong hình bên dưới.



Trong hình trên:

1. Tab **Logcat** để mở và đóng ngăn **Logcat**, hiển thị thông tin về ứng dụng của bạn khi ứng dụng đang chạy. Nếu bạn thêm câu lệnh Log vào ứng dụng, thông báo Log sẽ xuất hiện ở đây.
2. Menu cấp độ của Log được để mặc định, hiển thị tất cả các thông báo Log. Các thiết lập bao gồm **Debug**, **Error**, **Info** và **Warn**.

6.2 Thêm câu lệnh Log vào ứng dụng của bạn

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

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

Các phần của tin nhắn bao gồm:

- Log: Lớp **Log** để gửi tin nhắn log đến ngăn Logcat.

- d: Cài đặt mức **Debug** Log để lọc hiển thị thông báo log trong ngăn Logcat. Các mức log khác là e cho **Error**, w cho **Warn** và i cho **Info**.
- "MainActivity": Đối số đầu tiên là một thẻ có thể được sử dụng để lọc tin nhắn trong ngăn Logcat. Đây thường là tên của Activity mà tin nhắn bắt nguồn. Tuy nhiên, bạn có thể biến nó thành bất kỳ thứ gì hữu ích cho bạn để gỡ lỗi.

Theo quy ước, thẻ log được định nghĩa là hằng số cho Activity:

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

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

Làm theo các bước sau:

1. Mở ứng dụng Hello World của bạn trong Android studio và mở MainActivity.
2. Để tự động thêm các lệnh nhập rõ ràng vào dự án của bạn (chẳng hạn như android.util.Log cần thiết để sử dụng Log), hãy chọn **File > Settings** trong Windows hoặc **Android Studio > Preferences** trong macOS.
3. Chọn **Editor > General > Auto Import**. Chọn tất cả các hộp kiểm và thiết lập **Insert imports on paste to All**.
4. Chọn **Apply** và sau đó nhấn **OK**.
5. Trong phương thức onCreate() của MainActivity, thêm câu lệnh sau:

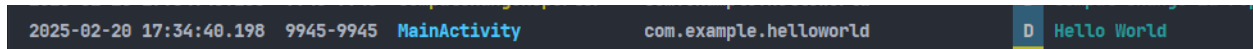
```
Log.d("MainActivity", "Hello World");
```

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

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

6. Nếu ngăn Logcat chưa mở, hãy nhấp vào biểu tượng Logcat ở thanh công cụ bên trái Android Studio để mở.
7. Kiểm tra xem tên mục tiêu và tên gói của ứng dụng có đúng không.
8. Thay đổi mức Log trong ngăn **Logcat** thành **Debug** (hoặc giữ nguyên vì có rất ít thông báo log).
9. Chạy ứng dụng của bạn.

Thông báo sau sẽ xuất hiện trong ngăn Logcat:



Thử thách viết code

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

1. Tạo một dự án mới trong Android Studio.
2. Đổi lời chào "Hello World" thành "Happy Birthday to " và tên của một người có ngày sinh nhật gần đây.
3. (Tùy chọn) Chụp ảnh màn hình ứng dụng đã hoàn thành của bạn và gửi email cho một người có ngày sinh nhật mà bạn đã quên.
4. Một cách sử dụng phổ biến của lớp **Log** là ghi log **Java exceptions** khi chúng xảy ra trong chương trình của bạn. Có một số phương thức hữu ích, chẳng hạn như **Log.e()**, mà bạn có thể sử dụng cho mục đích này. Khám phá các phương thức bạn có thể sử dụng để bao gồm một ngoại lệ với thông báo Log. Sau đó, viết mã trong ứng dụng của bạn để kích hoạt và ghi nhật ký một ngoại lệ.

```
package com.example.happybirthday;

import static android.content.ContentValues.TAG;

import android.os.Bundle;
import android.util.Log;
```

```

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

public class MainActivity extends AppCompatActivity {

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

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        try{
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            Log.e(TAG, "An error occurred: Division by zero", e);
        }
    }
}

```

Logcat:

```

2025-02-28 17:49:00.525 10364-10364 ContentValues com.example.helloworld E Ma hien thi loi!!!
java.lang.ArithmeticException: divide by zero

```

Bài tập về nhà

Xây dựng và chạy ứng dụng

- Tạo một dự án Android mới từ Empty Views Template.
- Thêm các câu lệnh ghi nhật ký cho nhiều cấp độ log khác nhau trong onCreate() trong MainActivity.
- Tạo trình giả lập cho thiết bị, nhắm mục tiêu đến bất kỳ phiên bản Android nào bạn thích và chạy ứng dụng.

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

```
package com.example.helloworld;

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

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

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "LogcatExampleApp";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        Log.v(TAG, "This is a VERBOSE log message.");
        Log.d(TAG, "This is a DEBUG log message.");
        Log.i(TAG, "This is an INFO log message.");
        Log.w(TAG, "This is a WARNING log message.");
        Log.e(TAG, "This is an ERROR log message.");
    }
}
```

Logcat:

2025-02-20 18:00:15.385	10474-10474	LogcatExampleApp	com.example.helloworld	V	This is a VERBOSE log message.
2025-02-20 18:00:15.385	10474-10474	LogcatExampleApp	com.example.helloworld	D	This is a DEBUG log message.
2025-02-20 18:00:15.385	10474-10474	LogcatExampleApp	com.example.helloworld	I	This is an INFO log message.
2025-02-20 18:00:15.385	10474-10474	LogcatExampleApp	com.example.helloworld	W	This is a WARNING log message.
2025-02-20 18:00:15.385	10474-10474	LogcatExampleApp	com.example.helloworld	E	This is an ERROR log message.

Trả lời những câu hỏi này

Câu hỏi 1

Tên của tệp bố cục cho main activity là gì?

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

Câu hỏi 2

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

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

Câu hỏi 3

Bạn sử dụng công cụ nào để tạo trình giả lập mới?

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

Câu hỏi 4

Giả sử ứng dụng của bạn bao gồm câu lệnh log này:

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

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

- Verbose

- Debug

- Info

- Warn

- Error

- Assert

Gửi ứng dụng của bạn để chấm điểm

Kiểm tra để đảm bảo ứng dụng có những điều sau:

- Một Activity hiển thị " Hello World " trên màn hình.
- Ghi nhật ký các câu lệnh vào onCreate() trong hoạt động chính.
- Cấp độ nhật ký trong ngăn **Logcat** chỉ hiển thị các câu lệnh log debug hoặc error.

Link code: <https://github.com/TamHocDev/Hello-Word.git>

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là chế độ xem — mọi thành phần của màn hình là một chế độ Xem. Lớp View biểu thị khối xây dựng cơ bản cho tất cả các thành phần UI và là lớp cơ sở cho các lớp cung cấp các thành phần UI tương tác như buttons, checkboxes và text entry fields. Các lớp con View thường được sử dụng 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 (như **RadioButton**, **CheckBox** và **Spinner**) để cung cấp hành vi tương tác.
- **ScrollView** và **RecyclerView** để hiển thị các mục có thể cuộn.

- **ImageView** để hiển thị hình ảnh.
- **ConstraintLayout** và **LinearLayout** để chứa các thành phần View khác và định vị chúng.

Đoạn mã Java hiển thị và điều khiển giao diện người dùng (UI) được chứa trong một lớp mở rộng từ **Activity**. Một **Activity** thường được liên kết với một bố cục (layout) của các thành phần giao diện người dùng (UI views) được định nghĩa trong một tệp XML (eXtended Markup Language). Tệp XML này thường được đặt tên theo tên của **Activity** và định nghĩa bố cục của các thành phần **View** trên màn hình.

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

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

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

Những điều bạn nên biết

Bạn cần 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

- Cách tạo một ứng dụng với hành vi tương tác.
- Cách sử dụng layout editor để thiết kế bố cục.
- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới. Hãy tham khảo **Vocabulary words and concepts glossary** để có các định nghĩa dễ hiểu.

Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

Ứng dụng **HelloToast** bao gồm hai phần tử **Button** và một **TextView**. Khi người dùng nhấn vào **Button** đầu tiên, một thông báo ngắn (**Toast**) sẽ hiển thị trên màn hình. Nhấn vào **Button** thứ hai sẽ tăng giá trị bộ đếm "click" được hiển thị trong **TextView**, bắt đầu từ số không.

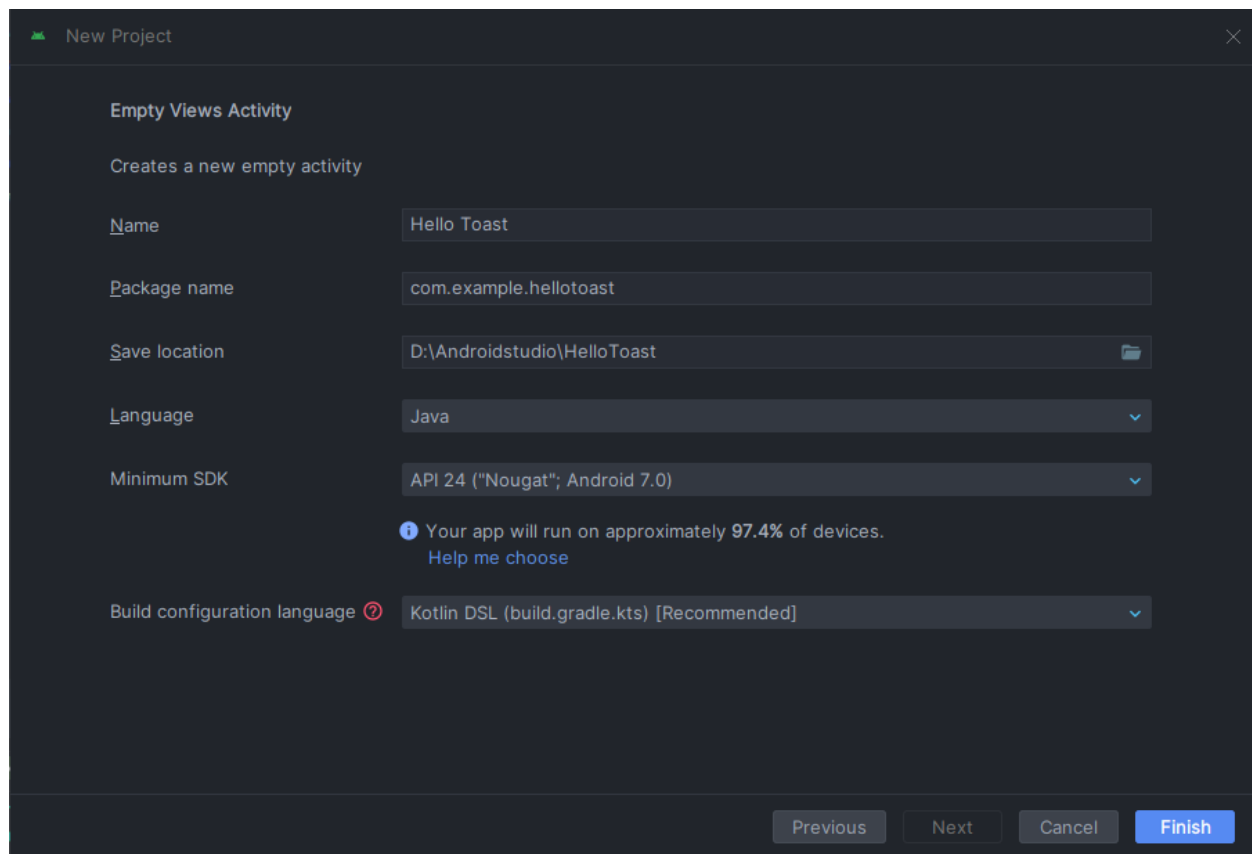
Sau đây là hình ảnh ứng dụng đã hoàn thành:

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

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

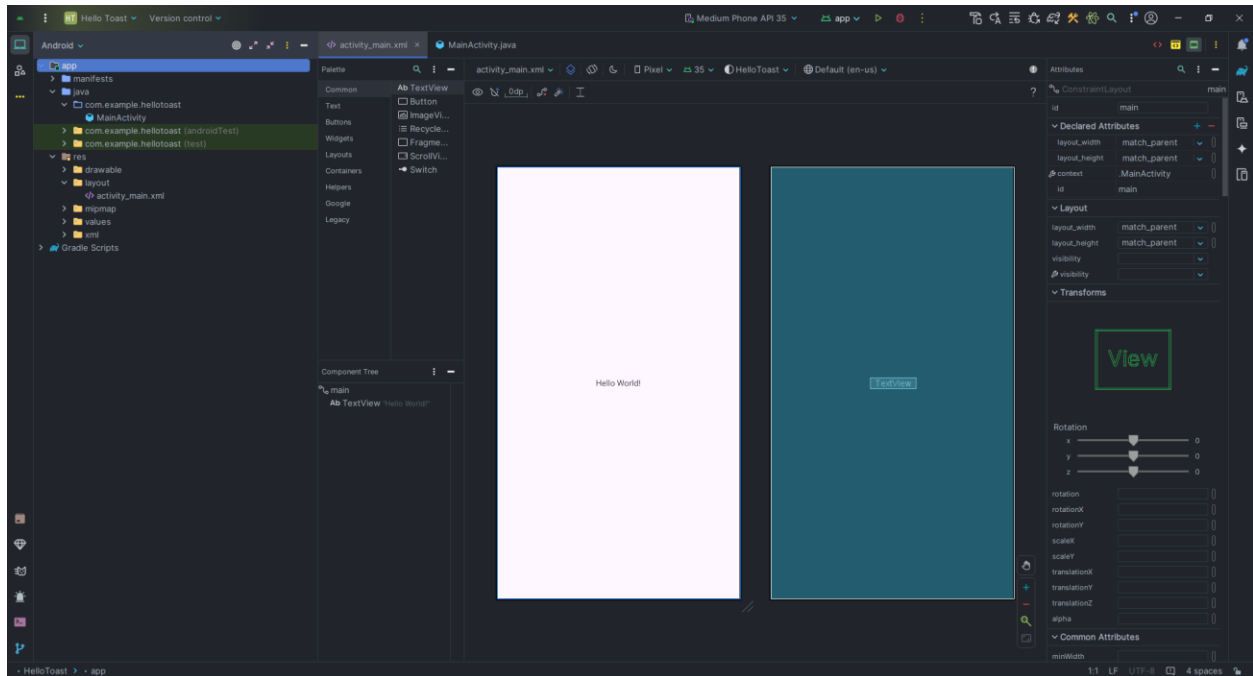
1.1 Tạo dự án Android Studio

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



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

1.2 Khám phá trình chỉnh sửa bố cục



1. Trong thư mục **app > res > layout** của **Project > Android** pane, nhấp đúp vào tệp **activity_main.xml** để mở nó nếu tệp chưa được mở.
2. Nhấp vào tab **Design** nếu tab này chưa được chọn. Bạn sử dụng tab **Design** để thao tác với các phần tử và bố cục, và sử dụng tab **Text** để chỉnh sửa mã XML của bố cục.
3. Ngăn **Palettes** hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục của ứng dụng.
4. Ngăn **Component tree** hiển thị cấu trúc phân cấp của các phần tử UI. Các phần tử View được tổ chức thành một cây phân cấp gồm các phần tử cha và con, trong đó phần tử con kế thừa các thuộc tính từ phần tử cha. Trong hình minh họa, TextView là một phần tử con của **ConstraintLayout**. Bạn sẽ học thêm về các phần tử này trong bài học sau.
5. Các ngăn design và blueprint của layout editor hiển thị các phần tử UI trong bố cục. Trong hình minh họa, bố cục chỉ hiển thị một phần tử: một TextView hiển thị dòng chữ "Hello World".
6. Tab **Attributes** hiển thị ngăn **Attributes**, nơi bạn có thể thiết lập các thuộc tính cho một phần tử UI.

Mẹo: Tham khảo **Building a UI with Layout Editor** để biết thêm chi tiết về cách sử dụng **layout editor**, và xem **Meet Android Studio** để đọc toàn bộ tài liệu hướng dẫn về **Android Studio**.

Nhiệm vụ 2: Thêm các thành phần View vào trình chỉnh sửa bố cục


Trong nhiệm vụ này, bạn sẽ tạo bố cục giao diện người dùng (**UI layout**) cho ứng dụng **HelloToast** trong **layout editor** bằng cách sử dụng các tính năng của **ConstraintLayout**. Bạn có thể tạo các ràng buộc (**constraints**) thủ công, như được minh họa sau đó, hoặc tự động bằng công cụ **Autoconnect**.


2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện các bước sau:

1. Mở tệp **activity_main.xml** từ **Project > Android** pane nếu tệp này chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào đó.

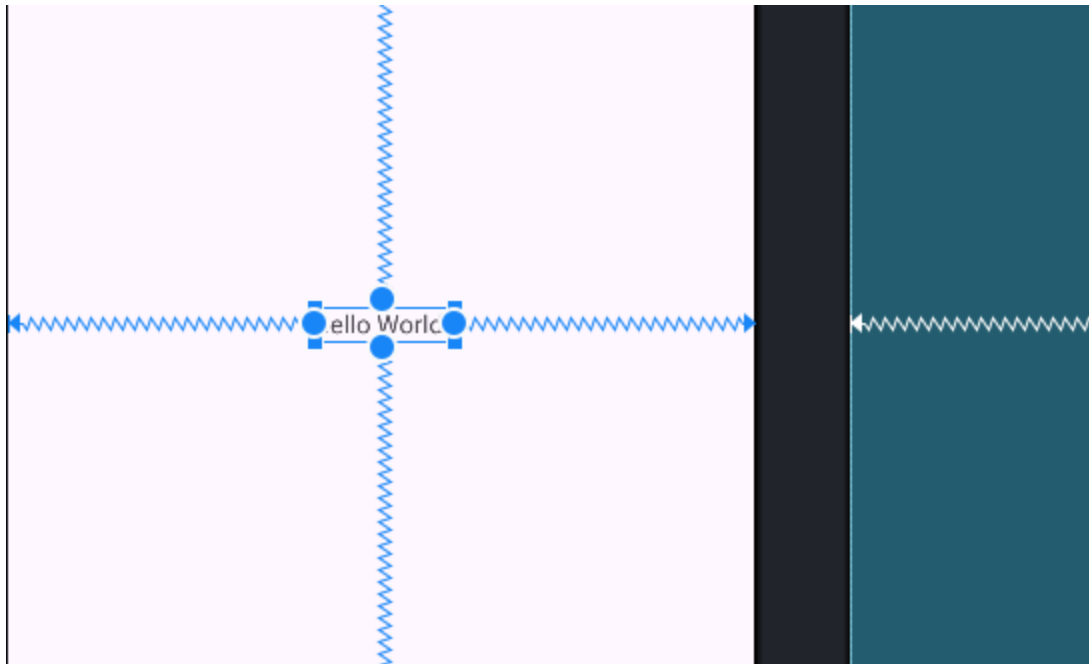
Nếu không có bản thiết kế (**blueprint**), nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**.

2. Công cụ **Autoconnect**  cũng nằm trên thanh công cụ và được bật theo mặc định. Đảm bảo rằng công cụ này không bị vô hiệu hóa.

3. Nhấp vào nút Zoom  in để phóng to các ngăn thiết kế (design) và bản thiết kế (blueprint) để quan sát kỹ hơn.

4. Chọn **TextView** trong ngăn Component Tree. Phần tử TextView với nội dung "Hello World" sẽ được làm nổi bật trong cả hai ngăn design và blueprint, và các ràng buộc của phần tử sẽ hiển thị.

5. Làm theo hình minh họa động dưới đây: Nhấp vào biểu tượng hình tròn ở phía bên phải của TextView để xóa ràng buộc ngang kết nối phần tử này với cạnh phải của bố cục. TextView sẽ chuyển sang phía bên trái vì nó không còn bị ràng buộc vào cạnh phải. Để thêm lại ràng buộc ngang, nhấp vào cùng biểu tượng hình tròn và kéo một đường tới cạnh phải của bố cục.



Trong các ngăn **blueprint** hoặc **design**, các tay nắm (**handles**) sau sẽ xuất hiện trên phần tử **TextView**:

- **Constraint handle**: Để tạo một ràng buộc như minh họa trong hình động ở trên, nhấp vào tay nắm ràng buộc, được hiển thị dưới dạng một hình tròn ở cạnh của một phần tử. Sau đó, kéo tay nắm đó đến một tay nắm ràng buộc khác hoặc đến đường biên của phần tử cha. Một đường gấp khúc sẽ biểu thị ràng buộc được tạo ra.



- **Resizing handle**: Để thay đổi kích thước phần tử, kéo các tay nắm chỉnh kích thước hình vuông. Khi bạn kéo, tay nắm sẽ chuyển thành một góc xiên.

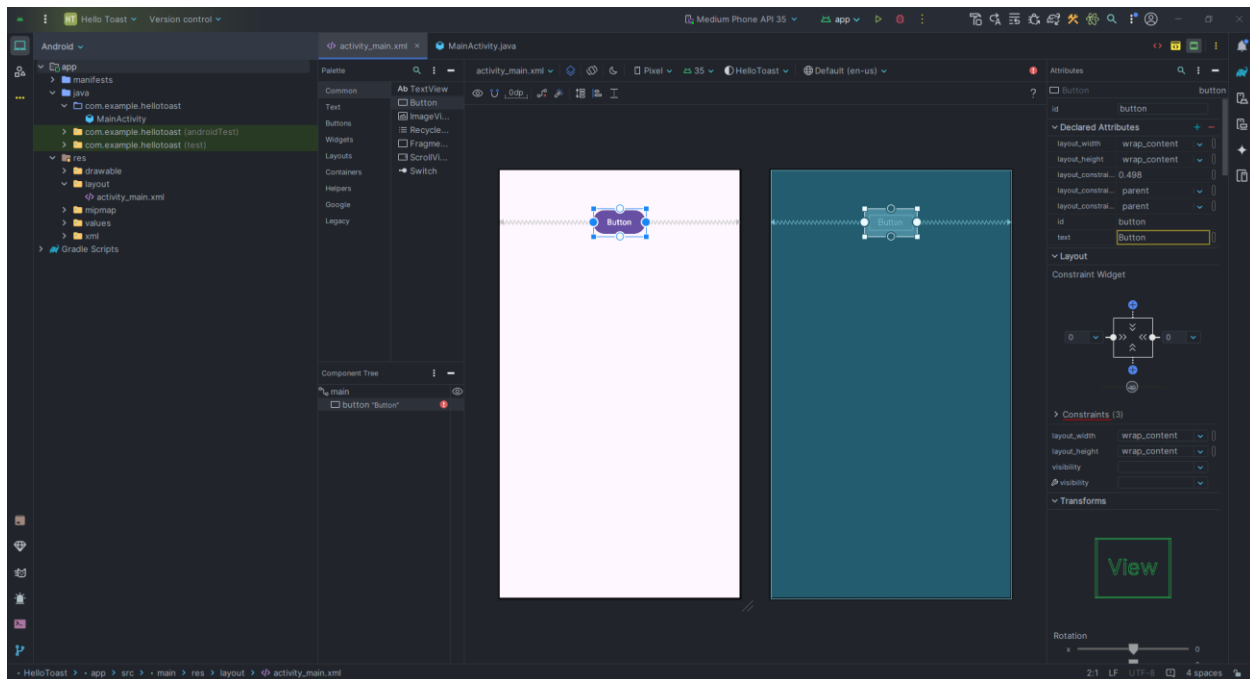


2.2 Thêm một Button vào bố cục

Khi được bật, công cụ **Autoconnect** tự động tạo hai hoặc nhiều ràng buộc (**constraints**) cho một phần tử giao diện người dùng (**UI element**) với bố cục cha (**parent layout**). Sau khi bạn kéo phần tử vào bố cục, công cụ này sẽ tạo ràng buộc dựa trên vị trí của phần tử.

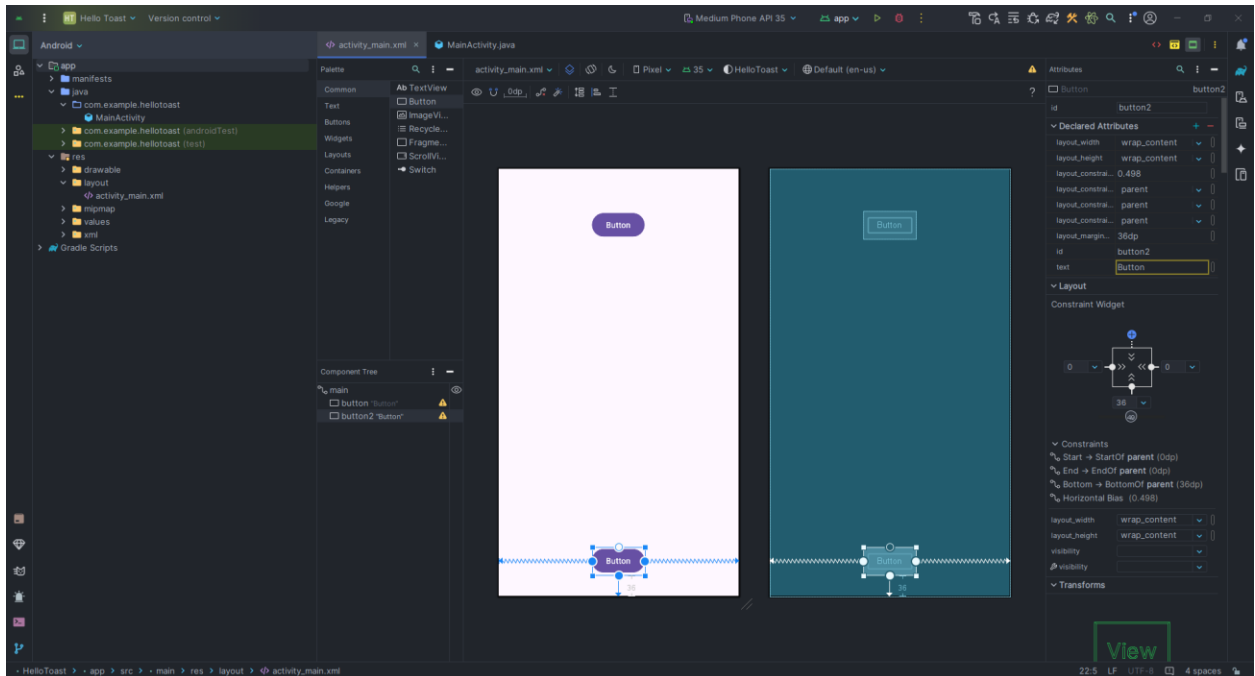
Thực hiện các bước sau để thêm một Button:

1. Bắt đầu với một bố cục trống. Phần tử **TextView** không cần thiết, vì vậy khi nó đang được chọn, hãy nhấn phím **Delete** hoặc chọn **Edit > Delete**. Lúc này, bạn sẽ có một bố cục hoàn toàn trống.
2. Kéo một **Button** từ ngăn **Palette** vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả **Button** vào khu vực chính giữa phía trên của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc để kết nối **Button** với cạnh trên, cạnh trái, và cạnh phải của bố cục như được minh họa trong hình động bên dưới.



2.3 Thêm Nút thứ hai vào bố cục

1. Kéo một **Button** khác từ ngăn **Palette** vào giữa bố cục, như được minh họa trong hình động bên dưới. Công cụ **Autoconnect** có thể tự động tạo các ràng buộc ngang cho bạn (nếu không, bạn có thể tự kéo các ràng buộc này).
2. Kéo một ràng buộc dọc (**vertical constraint**) từ **Button** xuống đáy của bố cục (**bottom of the layout**) như minh họa trong hình bên dưới. Điều này sẽ gắn kết **Button** với cạnh dưới của **ConstraintLayout**.



Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di chuột qua nó để hiển thị nút Clear Constraints. Nhấp vào nút này để xóa tất cả các ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc cụ thể, hãy nhấp vào tay cầm (handle) đặt ràng buộc đó. Để xóa tất cả các ràng buộc trong toàn bộ bố cục, nhấp vào công cụ **Clear All Constraints** trên thanh công cụ. Công cụ này rất hữu ích nếu bạn muốn thiết lập lại tất cả các ràng buộc trong bố cục của mình.

Nhiệm vụ 3: Thay đổi thuộc tính của phần tử UI

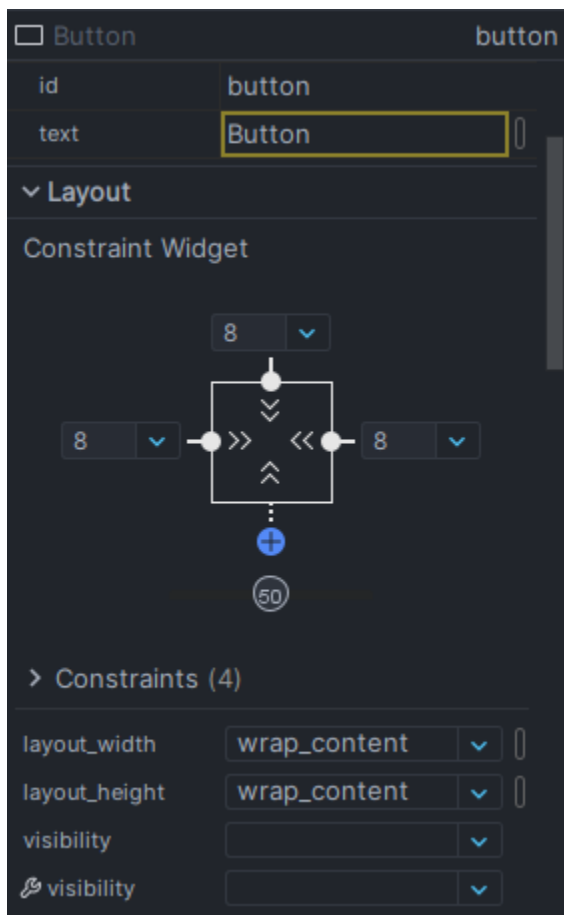
Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng (UI). Bạn có thể tìm các thuộc tính (được gọi là thuộc tính **properties**) chung cho tất cả các **View** trong View class documentation.

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

3.1 Thay đổi kích thước nút

Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước ở cả bốn góc của một **View**, giúp bạn có thể nhanh chóng thay đổi kích thước **View**. Bạn có thể kéo các tay cầm ở mỗi góc của **View** để thay đổi kích thước, nhưng việc này sẽ mã hóa cứng các kích thước chiều rộng và chiều cao. Hạn chế mã hóa cứng kích thước cho hầu hết các phần tử **View**, vì các kích thước được mã hóa cứng không thể thích ứng với nội dung và kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng bảng **Attributes** ở phía bên phải của trình chỉnh sửa bố cục để chọn chế độ kích thước không sử dụng các kích thước được mã hóa cứng. Bảng **Attributes** bao gồm một bảng kích thước hình vuông được gọi là **view inspector** ở phía trên. Các biểu tượng bên trong hình vuông đại diện cho các cài đặt chiều cao và chiều rộng như sau:

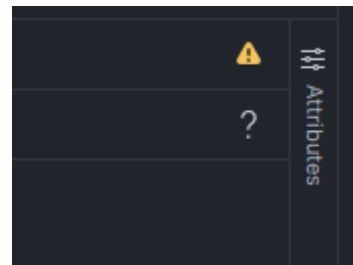
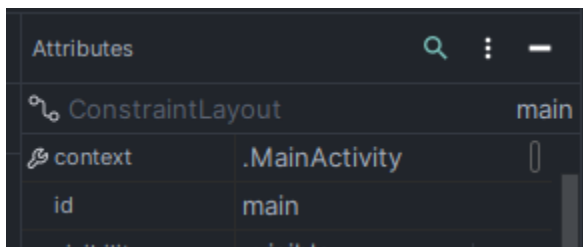


Trong hình trên:

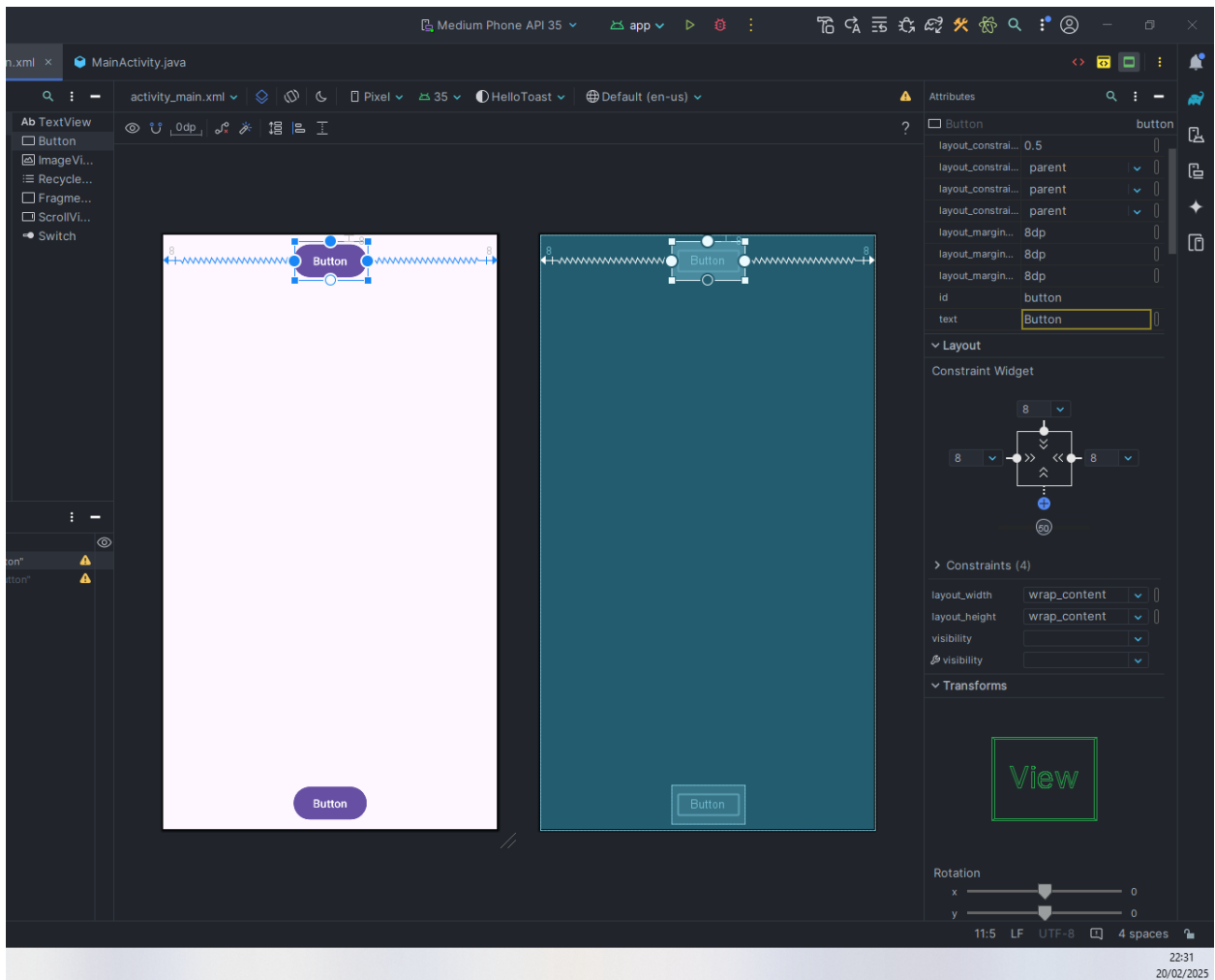
1. **Height control.** Điều khiển này xác định thuộc tính **layout_height** và xuất hiện ở hai đoạn trên và dưới của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành **wrap_content**, nghĩa là View sẽ mở rộng theo chiều dọc khi cần để phù hợp với nội dung của nó. Số "8" chỉ ra một lề chuẩn được đặt là 8dp.
2. **Width control.** Điều khiển này xác định thuộc tính **layout_width** và xuất hiện ở hai đoạn bên trái và phải của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành **wrap_content**, nghĩa là View sẽ mở rộng theo chiều ngang khi cần để phù hợp với nội dung của nó, tối đa đến một lề là 8dp.
3. Nút đóng **Attributes pane.** Nhấp để đóng bảng điều khiển.

Làm theo các bước sau:

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

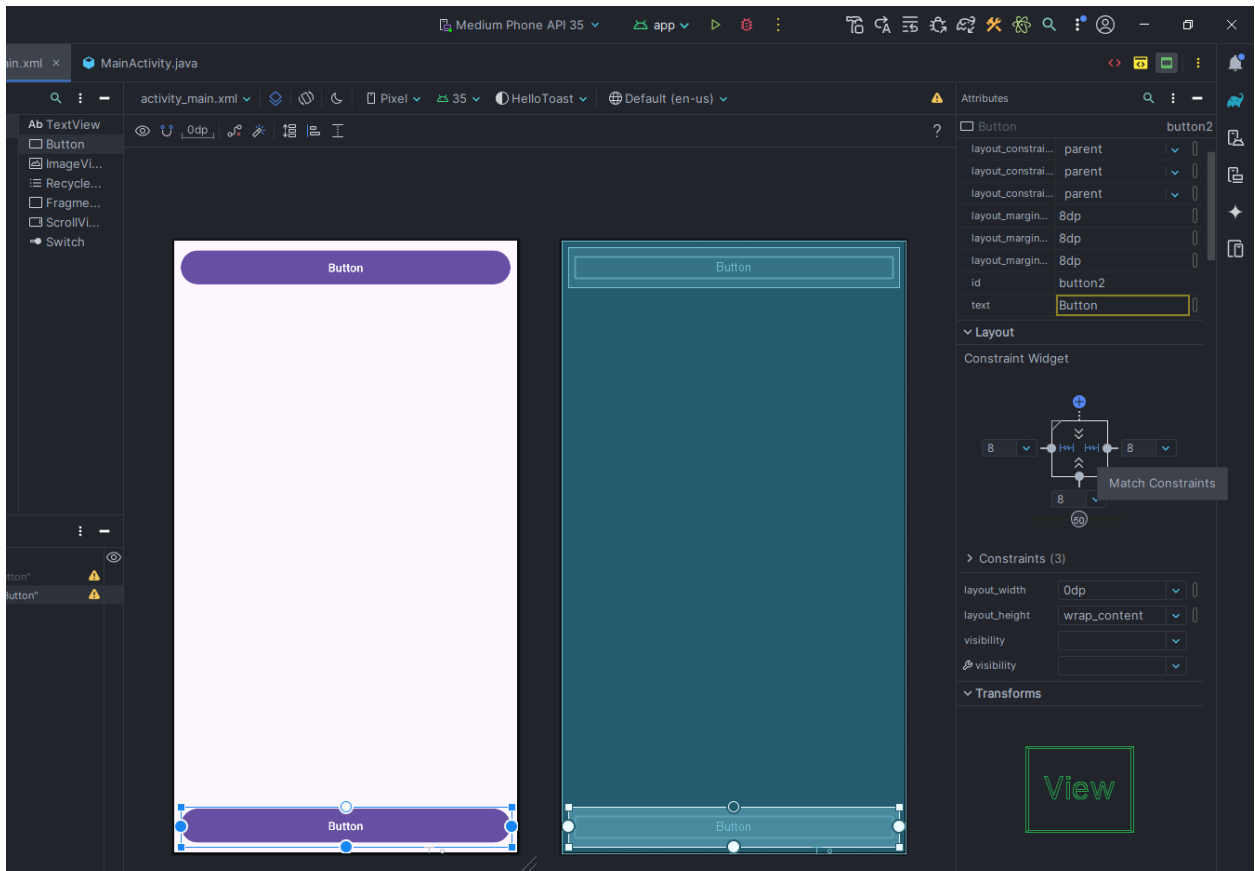


3. Nhấp vào điều khiển độ rộng hai lần - lần nhấp đầu tiên thay đổi thành **Fixed** với các đường thẳng, và lần nhấp thứ hai thay đổi thành **Match Constraints** với các cuộn lò xo, như minh họa trong hình động dưới đây.



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

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



Như đã trình bày trong các bước trước, các thuộc tính `layout_width` và `layout_height` trong khung `Attributes` thay đổi khi bạn thay đổi các điều khiển chiều cao và chiều rộng trong trình kiểm tra. Các thuộc tính này có thể nhận một trong ba giá trị đối với bố cục, đây là `ConstraintLayout`:

- Cài đặt `match_constraint` mở rộng phần tử `View` để lấp đầy bố mẹ theo chiều rộng hoặc chiều cao—tính đến phần margin, nếu có. Bố mẹ trong trường hợp này là `ConstraintLayout`. Bạn sẽ tìm hiểu thêm về `ConstraintLayout` trong nhiệm vụ tiếp theo.
- Cài đặt `wrap_content` thu nhỏ kích thước của phần tử `View` sao cho vừa đủ 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 có thể điều chỉnh theo kích thước màn hình của thiết bị, hãy sử dụng một số cố định theo đơn vị pixel không phụ thuộc vào mật độ (dp). Ví dụ, 16dp có nghĩa là 16 pixel không phụ thuộc vào mật độ.

Mẹo: Nếu bạn thay đổi thuộc tính **layout_width** bằng menu bật lên của nó, thuộc tính **layout_width** sẽ được đặt thành **zero** vì không có kích thước được chỉ định. Cài đặt này tương đương với **match_constraint**—phần tử có thể mở rộng tối đa để đáp ứng các ràng buộc và cài đặt margin.

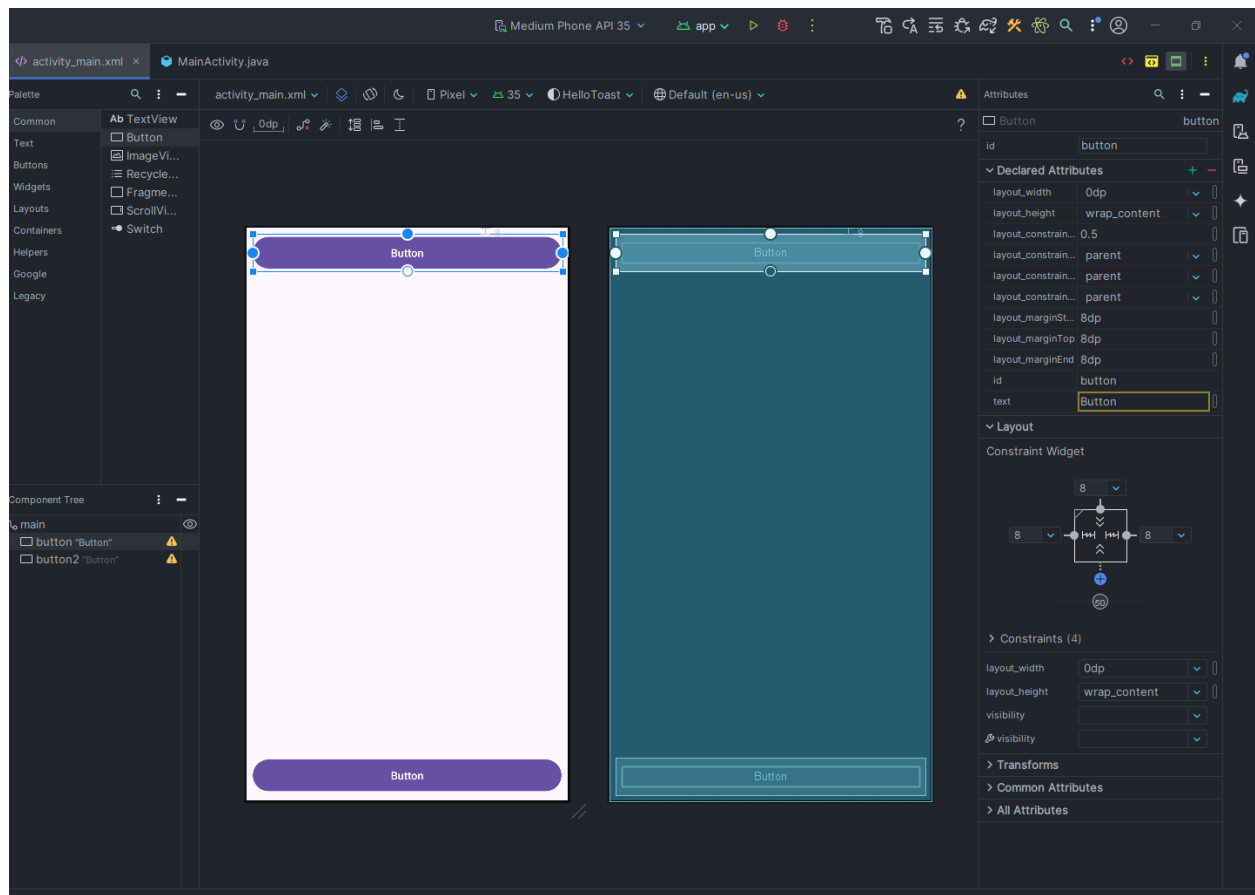
3.2 Thay đổi thuộc tính Nút

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

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

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

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



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

ColorPrimary là màu chính của chủ đề, một trong các màu cơ bản được định nghĩa sẵn trong tệp tài nguyên **colors.xml**. Màu này được sử dụng cho thanh ứng dụng (app bar). Sử dụng các màu cơ bản cho các thành phần giao diện người dùng khác sẽ tạo ra một giao diện thống nhất. Bạn sẽ tìm hiểu thêm về chủ đề ứng dụng và Material Design trong bài học khác.

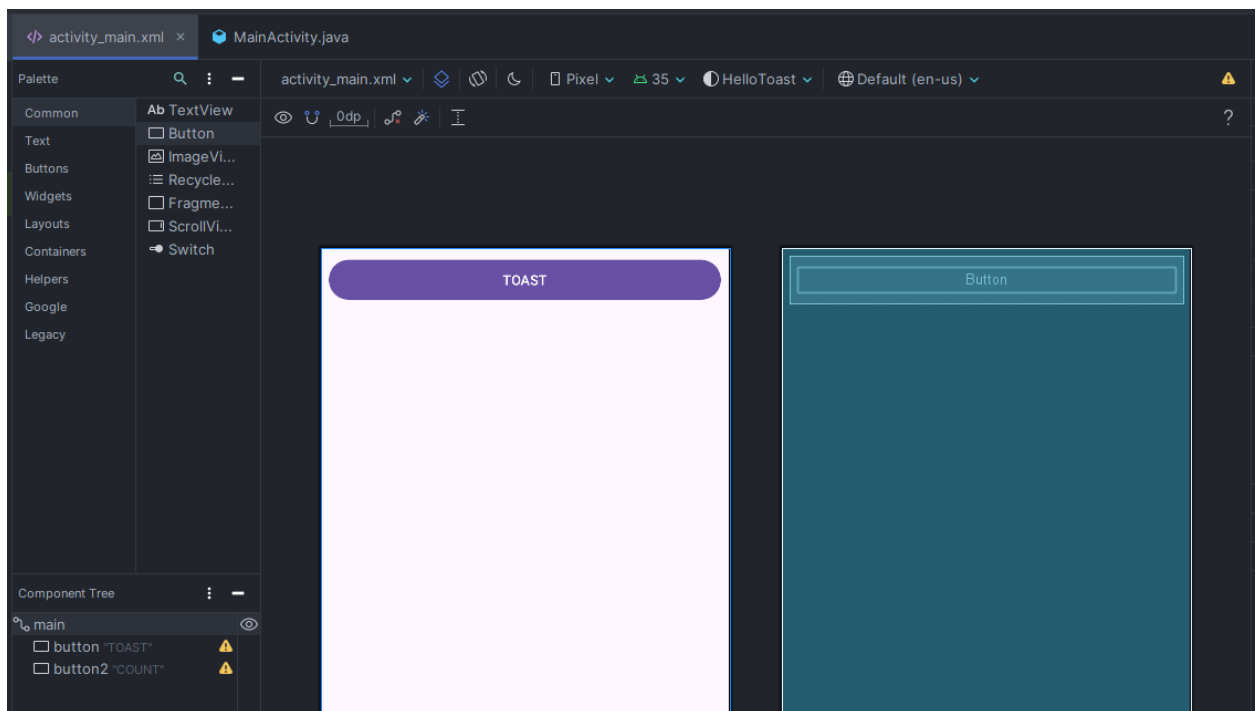
Nhiệm vụ 4: Thêm **TextView** và đặt thuộc tính của nó

Một trong những lợi ích của **ConstraintLayout** là khả năng căn chỉnh hoặc ràng buộc các phần tử so với các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một **TextView** vào giữa bố cục, và ràng buộc nó theo chiều ngang với các lề và theo

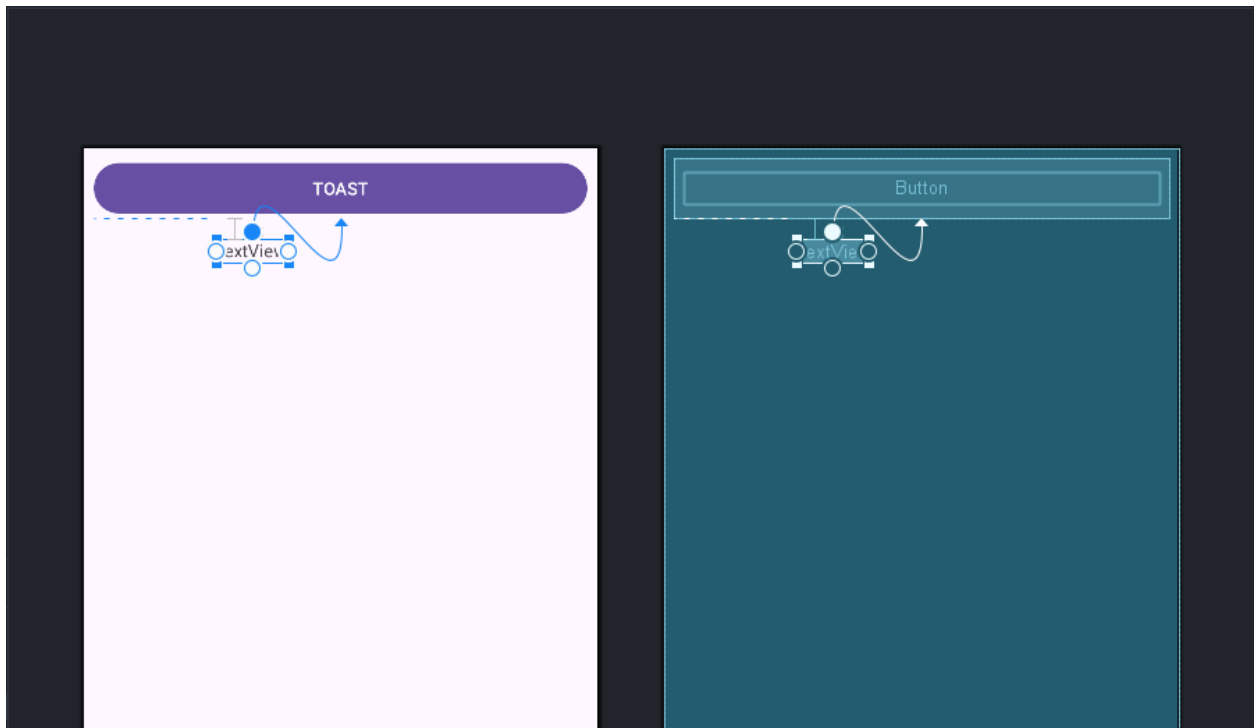
chiều dọc với hai phần tử **Button**. Sau đó, bạn sẽ thay đổi các thuộc tính cho **TextView** trong bảng **Attributes**.

4.1 Thêm TextView và các ràng buộc

1. Như được minh họa trong hình động dưới đây, kéo một **TextView** từ bảng **Palette** vào phần trên của bố cục, và kéo một ràng buộc từ đỉnh của **TextView** đến tay cầm ở phía dưới của nút **Toast Button**. Điều này ràng buộc **TextView** nằm bên dưới nút **Button**.



2. Như được minh họa trong hình động dưới đây, kéo một ràng buộc từ phía dưới của **TextView** đến tay cầm ở phía trên của nút **Count Button**, và từ các cạnh của **TextView** đến các cạnh của bố cục. Điều này ràng buộc **TextView** nằm ở giữa bố cục giữa hai nút **Button**.




4.2 Đặt thuộc tính TextView

Với **TextView** đã được chọn, mở bảng **Attributes**, nếu bảng chưa được mở. Thiết lập các thuộc tính cho **TextView** như được minh họa trong hình động dưới đây. Các thuộc tính mà bạn chưa gặp trước đó sẽ được giải thích sau hình minh họa:

1. Đặt **ID** là **show_count**.
2. Đặt **text** thành **0**.
3. Đặt **textSize** là **160sp**.
4. Đặt **textStyle** thành **B** (in đậm) và **textAlignment** thành **ALIGNCENTER** (căn giữa đoạn văn).
5. Thay đổi các tùy chọn kích thước ngang và dọc (**layout_width** và **layout_height**) thành **match_constraint**.
6. Đặt **textColor** là **@color/colorPrimary**.
7. Cuộn xuống trong khung **Attributes** và nhấp vào **View all attributes**, sau đó cuộn xuống trang thứ hai của các thuộc tính đến **background**, và nhập **#FFF00** để chọn màu vàng.
8. Cuộn xuống **gravity**, mở rộng mục **gravity**, và chọn **center_ver** (để căn giữa theo chiều dọc).

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

Bạn có thể nhận thấy rằng thuộc tính **background** nằm ở trang đầu tiên của bảng **Attributes** đối với một Button, nhưng lại nằm ở trang thứ hai đối với một TextView. Bảng **Attributes** thay đổi theo từng loại View: Các thuộc tính phổ biến nhất của loại View đó xuất hiện trên trang đầu tiên, và các thuộc tính còn lại được liệt kê ở trang thứ hai. Để quay lại trang đầu tiên của bảng **Attributes**, hãy nhấp vào biểu tượng  trên thanh công cụ ở đầu bảng.


Nhiệm vụ 5: Chỉnh sửa bố cục trong XML

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



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

5.1 Mở mã XML cho bố cục

Đối với tác vụ này, hãy mở tệp `activity_main.xml` nếu nó chưa được mở, và chuyển sang dạng Text trong tab  ở thanh công cụ.

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

```
<Button
    android:id="@+id/btn_toast"
    android:layout_width="8dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="TOAST"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/btn_count"
    android:layout_width="8dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:text="COUNT"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
```

5.2 Trích xuất tài nguyên chuỗi

Thay vì mã hóa cứng các chuỗi, một thực hành tốt nhất là sử dụng tài nguyên chuỗi (string resources), đại diện cho các chuỗi đó. Việc đặt các chuỗi trong một tệp riêng biệt giúp bạn dễ dàng quản lý hơn, đặc biệt khi bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc khi dịch và bản địa hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho mỗi ngôn ngữ.

1. Nhấp một lần vào từ "Toast" (cảnh báo được đánh dấu đầu tiên).
2. Nhấn **Alt-Enter** trên Windows hoặc **Option-Enter** trên macOS và chọn **Extract string resource** từ menu bật lên.
3. Nhập **button_label_toast** làm Tên tài nguyên (**Resource name**).
4. Nhấp **OK**. Một tài nguyên chuỗi được tạo trong tệp values/res/strings.xml, và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: **@string/button_label_toast**.
5. Trích xuất các chuỗi còn lại: **button_label_count** cho "Count" và **count_initial_value** cho "0".
6. Trong **Project > Android** pane, mở rộng **values** trong **res**, sau đó nhấp đúp vào **strings.xml** để xem các tài nguyên chuỗi của bạn trong tệp strings.xml:

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

7. Bạn cần thêm một chuỗi khác để sử dụng trong một nhiệm vụ tiếp theo hiển thị thông báo. Thêm vào tệp **strings.xml** một tài nguyên chuỗi khác có tên **toast_message** cho cụm từ "Hello Toast!":

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

Mẹo: Các tài nguyên chuỗi bao gồm tên ứng dụng, xuất hiện trong thanh ứng dụng ở đầu màn hình nếu bạn bắt đầu dự án ứng dụng của mình bằng Mẫu Trống

(Empty Template). Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa tài nguyên **app_name**.

Nhiệm vụ 6: Thêm trình xử lý onClick cho các nút

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

6.1 Thêm thuộc tính và trình xử lý onClick vào mỗi Nút

Một **trình xử lý nhấp chuột (click handler)** là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử giao diện người dùng có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường **onClick** ở tab **Design** trong **Attributes pane**. Bạn cũng có thể chỉ định tên của trình xử lý bằng cách chỉnh sửa trong **XML editor**, thêm thuộc tính **android:onClick** vào Button.

Bạn sẽ sử dụng phương pháp thứ hai vì bạn chưa tạo các phương thức xử lý, và **XML editor** cung cấp cách tự động để tạo các phương thức này.

1. Với **XML editor** đang mở (tab **Text**), tìm **Button** có thuộc tính **android:id** được đặt là **button_toast**:

```
<Button
    android:id="@+id/btn_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="TOAST"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

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

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

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

Nếu biểu tượng bóng đèn đỏ không xuất hiện, nhấp vào tên phương thức ("**showToast**"). Nhấn **Alt-Enter** (hoặc **Option-Enter** trên Mac), chọn **Create 'showToast(view)' in MainActivity**, và nhấp **OK**.

Thao tác này sẽ tạo một đoạn mã khung (placeholder method stub) cho phương thức **showToast()** trong **MainActivity**, như được hiển thị ở cuối các bước này:

4. Lặp lại hai bước cuối cùng với Button_count Button: Thêm thuộc tính `android:onClick` vào cuối và thêm trình xử lý nhấp chuột:

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

Mã XML cho các thành phần giao diện người dùng trong ConstraintLayout bây giờ trông như thế này:

```
<Button
    android:id="@+id/btn_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="TOAST"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="showToast" />

<Button
    android:id="@+id/btn_count"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:text="COUNT"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
```

```

        android:onClick="countUp" />

<TextView
    android:id="@+id/show_count"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="#FFFF00"
    android:text="0"
    android:gravity="center_vertical"
    android:textAlignment="center"
    android:textColor="@color/design_default_color_primary"
    android:textSize="160sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/btn_count"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/btn_toast" />

```

5. Nếu tệp **MainActivity.java** chưa được mở, hãy mở rộng mục **java** trong **Project > Android view**, mở rộng **com.example.android.hellotoast**, sau đó nhấp đúp vào **MainActivity**. Trình chỉnh sửa mã sẽ hiển thị với mã trong **MainActivity**.

```

package com.example.hellotoast;

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

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

public class MainActivity extends AppCompatActivity {

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

```



```

}

public void showToast(View view) {
}

public void countUp(View view) {
}
}

```

6.2 Chỉnh sửa trình xử lý nút Toast

Bây giờ bạn sẽ chỉnh sửa phương thức **showToast()**—trình xử lý nhấp chuột của nút **Toast Button** trong **MainActivity**—để hiển thị một thông báo. **Toast** cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ popup nhỏ. Nó chỉ chiếm không gian cần thiết cho thông báo. Hoạt động hiện tại vẫn hiển thị và tương tác được. **Toast** có thể hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn—thêm một thông báo **Toast** để hiển thị kết quả của việc nhấn vào một nút hoặc thực hiện một hành động.

Hãy làm theo các bước sau để chỉnh sửa trình xử lý nhấp chuột vào Nút **Toast**:

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

```

public void showToast(View view) {
}

```

2. Để tạo một thể hiện của **Toast**, hãy gọi phương thức nhà máy **makeText()** trên lớp **Toast**.

```

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

```

Câu lệnh này chưa đầy đủ cho đến khi bạn hoàn thành tất cả các bước

3. Cung cấp ngữ cảnh của Activity trong ứng dụng. Vì một **Toast** hiển thị trên giao diện người dùng của Activity, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong ngữ cảnh của Activity mà bạn cần, sử dụng **this** như một cách viết tắt.

```

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

```

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

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

5. Cung cấp thời lượng để hiển thị. Ví dụ: **Toast.LENGTH_SHORT** hiển thị toast trong một khoảng thời gian tương đối ngắn.

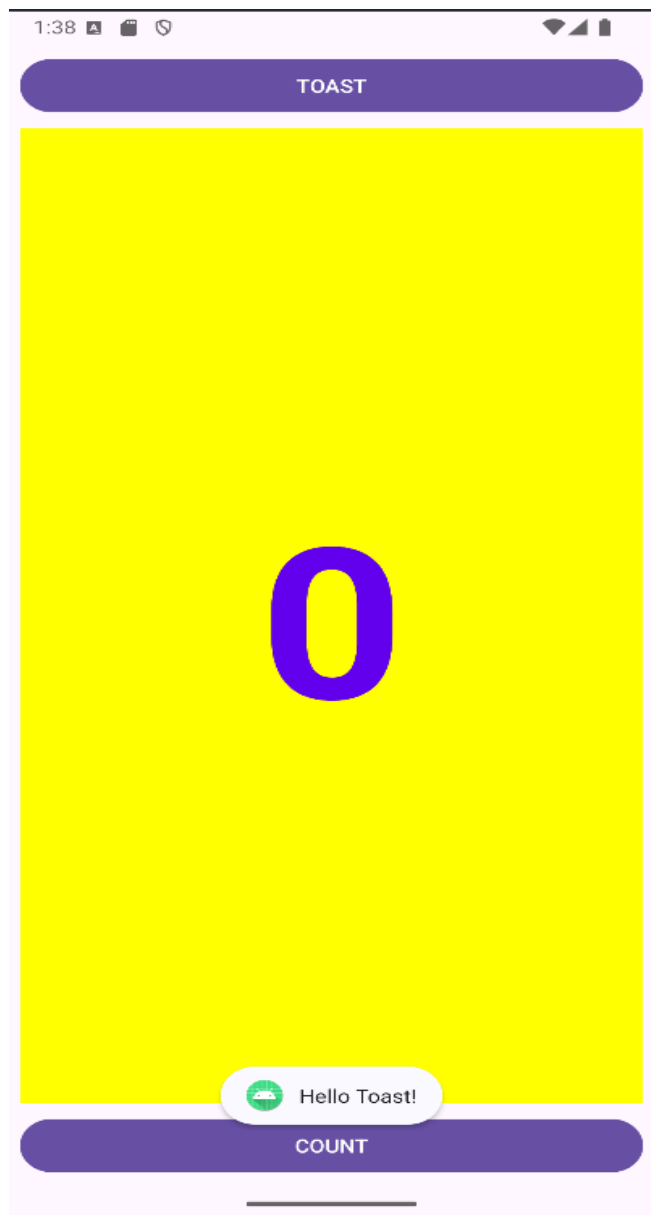
```
public void showToast(View view) {  
    Toast toast = Toast.makeText(MainActivity.this, R.string.toast_message, Toast.LENGTH_LONG);  
}
```

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

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

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

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



6.3 Chỉnh sửa trình xử lý Nút đếm

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

Mã cho trình xử lý phải:

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

Thực hiện theo các bước sau để chỉnh sửa trình xử lý nhấp vào Nút đếm:

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

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

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

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

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

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

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

4. Thay đổi câu lệnh khai báo biến thành viên riêng để khởi tạo biến với giá trị bằng không:

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

5. Cùng với biến ở 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 sự kiện nhấp. Gọi biến này là **mShowCount**:

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

6. Bây giờ, khi bạn đã có **mShowCount**, bạn có thể lấy tham chiếu đến **TextView** bằng cách sử dụng ID mà bạn đã đặt trong tệp bố cục. Để chỉ lấy tham chiếu này một lần, hãy xác định nó trong phương thức **onCreate()**.

Như bạn đã học trong một bài học khác, phương thức **onCreate()** được sử dụng để "inflate" bố cục, nghĩa là đặt nội dung hiển thị của màn hình theo tệp bố cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện khác trong bố cục, chẳng hạn như **TextView**. Xác định vị trí phương thức **onCreate()** trong **MainActivity**:

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

7. Thêm câu lệnh [findViewById](#) vào cuối phương thức:

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

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

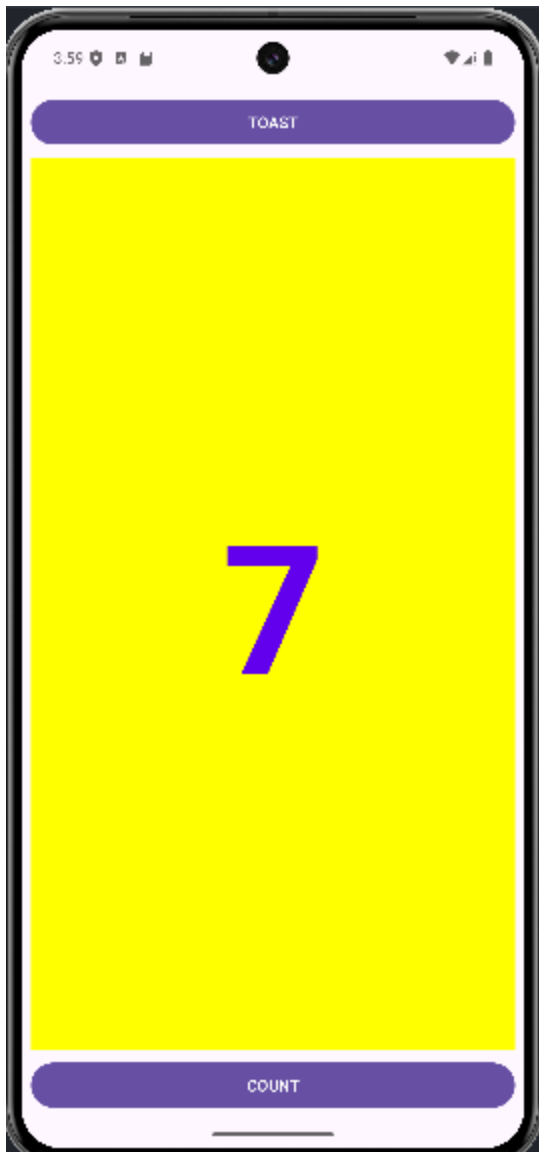
8. Bây giờ, khi bạn đã gán **TextView** cho **mShowCount**, bạn có thể sử dụng biến này để thiết lập văn bản trong **TextView** thành giá trị của biến **mCount**. Thêm đoạn sau vào phương thức **countUp()**:

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

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

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

9. Chạy ứng dụng để kiểm tra rằng số đếm tăng lên khi bạn nhấn vào nút **Count**.



Mã giải pháp

Dự án Android Studio:

```

package com.example.hellotoast;

import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

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

public class MainActivity extends AppCompatActivity {
    private int mCount = 0;
    private TextView mShowCount;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        mShowCount = (TextView) findViewById(R.id.show_count);
    }

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

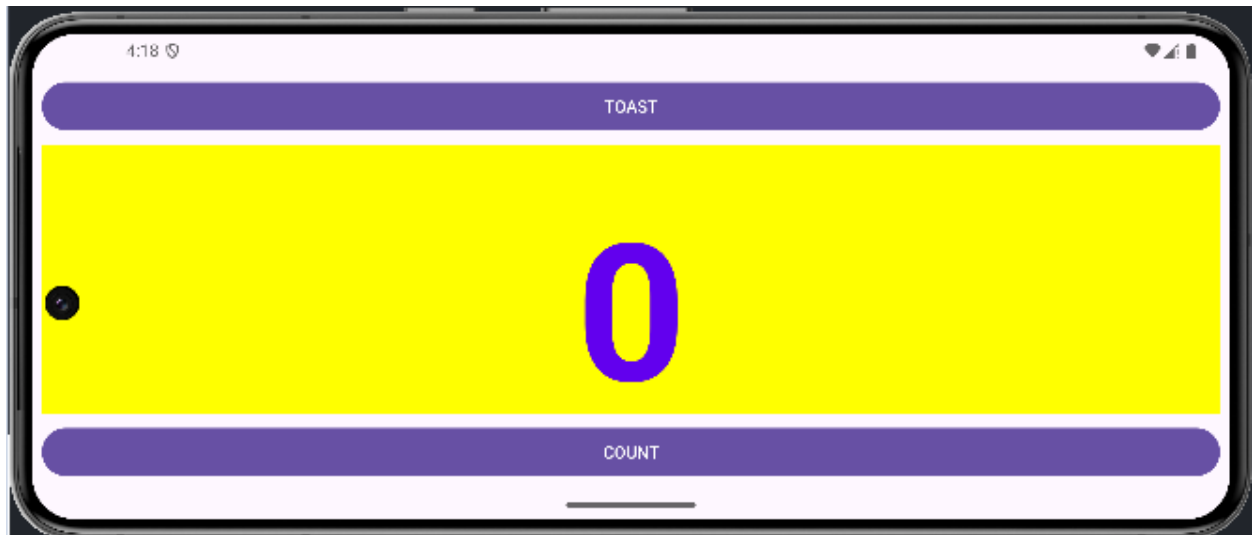
    public void countUp(View view) {
        mCount++;
        if(mShowCount != null)
            mShowCount.setText(Integer.toString(mCount));
    }
}

```

Thử thách mã hóa

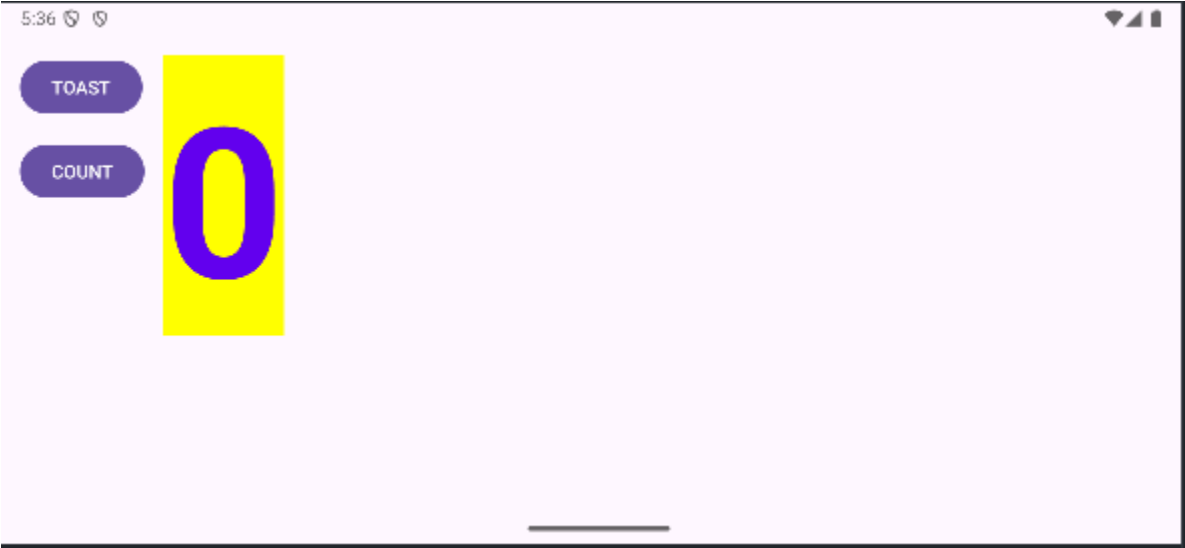
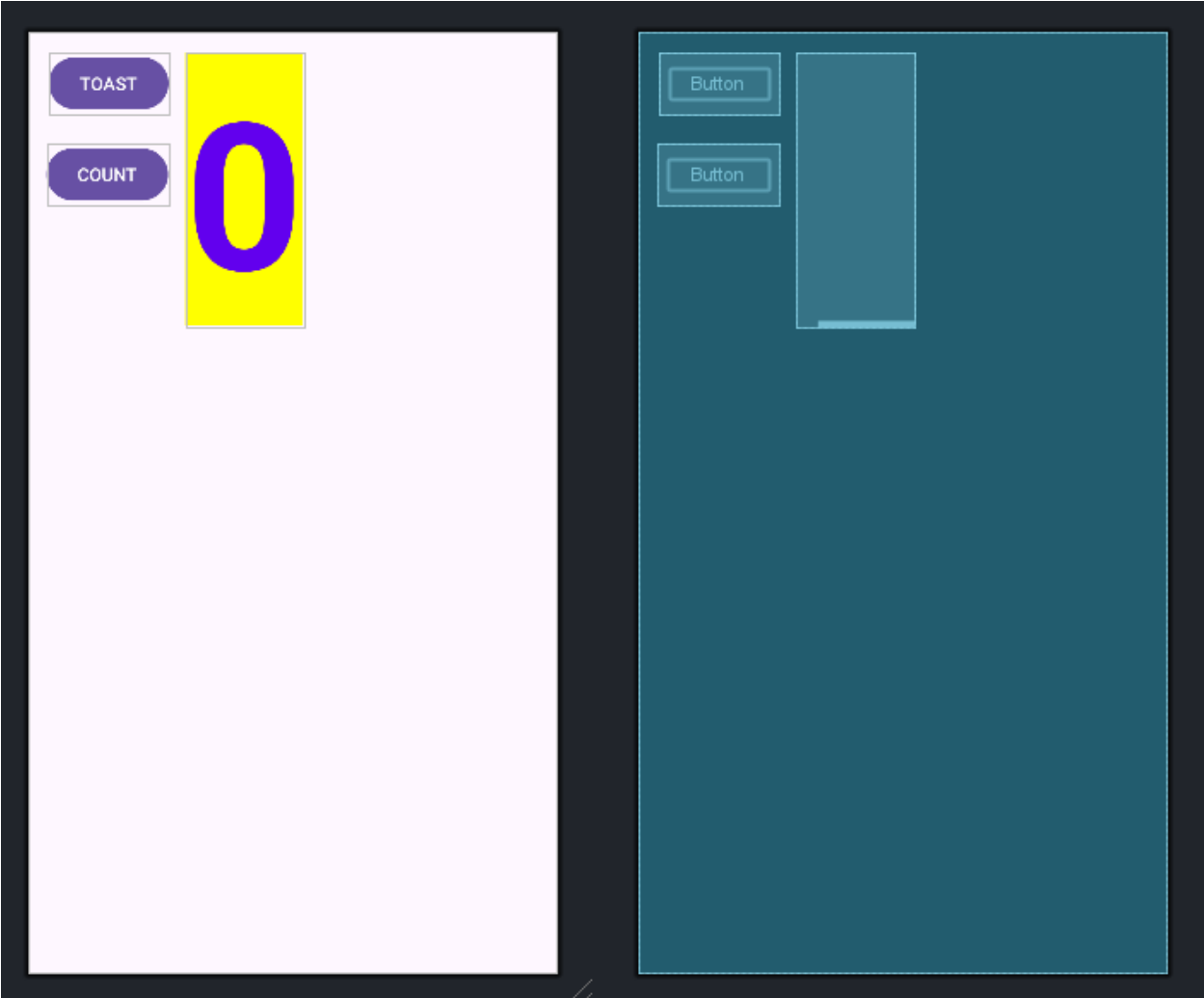
Ứng dụng HelloToast trông ổn khi thiết bị hoặc trình giả lập được định hướng theo chiều dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang định hướng

theo chiều ngang, Nút Đếm có thể chồng lên TextView dọc theo phía dưới như minh họa trong hình bên dưới.



Thách thức: Thay đổi bố cục sao cho đẹp theo cả hướng ngang và hướng dọc:

1. Trên máy tính của bạn, tạo một bản sao của thư mục dự án **HelloToast** và đổi tên thành **HelloToastChallenge**.
2. Mở **HelloToastChallenge** trong Android Studio và sắp xếp lại. (Xem [Appendix: Utilities](#) để biết hướng dẫn về cách sao chép và sắp xếp lại dự án.)
3. Thay đổi bố cục sao cho Nút **Toast** và Nút **Count** xuất hiện ở bên trái, như hiển thị trong hình bên dưới. TextView xuất hiện bên cạnh chúng, nhưng chỉ đủ rộng để hiển thị nội dung của nó. (Gợi ý: Sử dụng `wrap_content`.)
4. Chạy ứng dụng theo cả hướng ngang và hướng dọc.



Mã giải pháp thách thức

Dự án Android Studio: <https://github.com/TamHocDev/HelloToast.git>

1.3)

1.4) Trình chỉnh sửa bố cục

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