

CHƯƠNG 1: Làm quen

Các bài giảng trong chương này

Bài 1: Tạo ứng dụng đầu tiên

- 1.1: Android Studio và Hello World
- 1.2: Giao diện người dùng tương tác đầu tiên
- 1.3: Trình chỉnh sửa bố cục
- 1.4: Văn bản và các chế độ cuộn
- 1.5: Tài nguyên có sẵn

Bài 2: Activities

- 2.1: Activity và Intent
- 2.2: Vòng đời của Activity và trạng thái
- 2.3: Intent ngầm định

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

- 3.1: Trình gỡ lỗi
- 3.2: Kiểm thử đơn vị
- 3.3: Thư viện hỗ trợ

Bài 1.1: Android Studio và Hello World

Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

- Hiểu quy trình phát triển phần mềm tổng quát cho các ứng dụng lập trình hướng đối tượng sử dụng một IDE (môi trường phát triển tích hợp) như Android Studio.

- Chứng minh rằng bạn có ít nhất 1-3 năm kinh nghiệm trong lập trình hướng đối tượng, với một phần trong số đó tập trung vào ngôn ngữ lập trình Java. (Các bài thực hành này sẽ không giải thích về lập trình hướng đối tượng hoặc ngôn ngữ Java).

Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

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

Đây là hình ảnh trực quan của một ứng dụng hoàn chỉnh:



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 nguồn tiên tiến và một bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ hỗ trợ phát triển, gỡ lỗi, kiểm thử và tối ưu hóa hiệu suất, giúp quá trình phát triển ứng dụng trở nên nhanh chóng và dễ dàng hơn. Bạn có thể kiểm thử ứng dụng trên nhiều trình giả lập được cấu hình sẵn hoặc trực tiếp trên thiết bị di động của mình, biên dịch ứng dụng cho môi trường sản xuất và phát hành trên Google Play Store.

Android Studio có sẵn cho máy tính chạy Windows, Linux và máy Mac chạy macOS. Bộ OpenJDK (Java Development Kit) mới nhất được tích hợp sẵn trong Android Studio.

Để bắt đầu với Android Studio, trước tiên hãy kiểm tra yêu cầu hệ thống để đảm bảo rằng hệ thống của bạn đáp ứng đủ điều kiện. Quá trình cài đặt tương tự trên tất cả các nền tảng, chỉ có một số điểm khác biệt sẽ được ghi chú cụ thể.

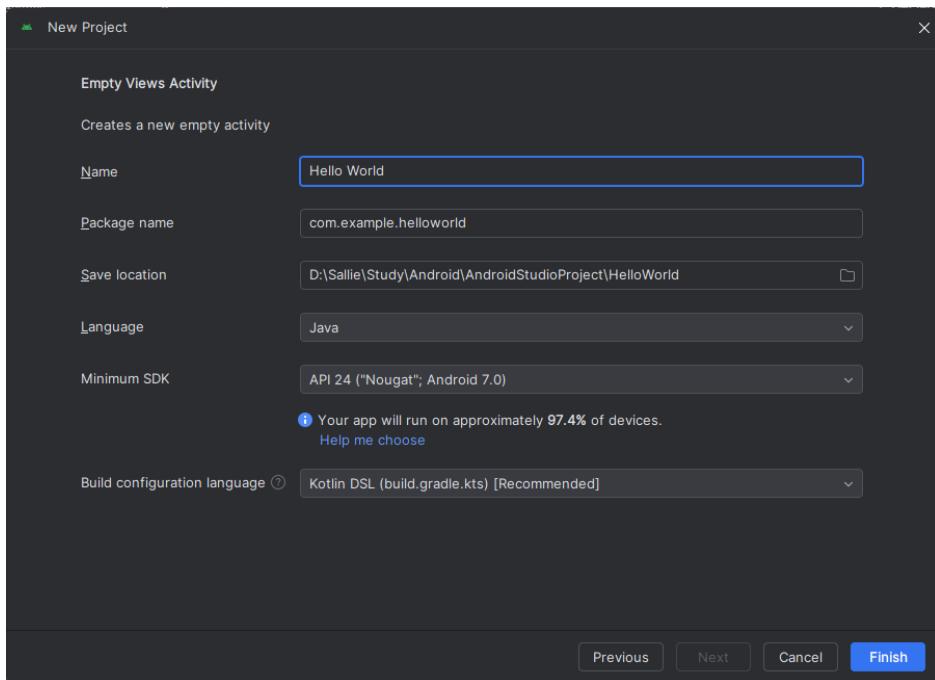
1. Truy cập trang web dành cho nhà phát triển Android và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
2. Chấp nhận các cấu hình mặc định trong tất cả các bước và đảm bảo rằng tất cả các thành phần cần thiết đều được chọn để cài đặt.
3. Sau khi hoàn tất quá trình cài đặt, Trình hướng dẫn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

Nhiệm vụ 2: Khởi tạo ứng dụng Hello world

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

2.1 Khởi tạo dự án ứng dụng

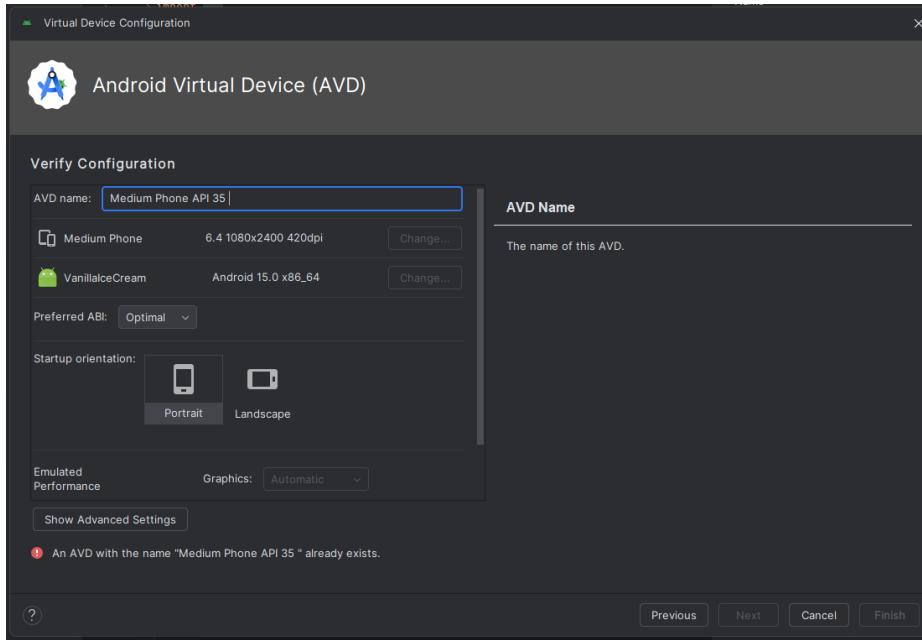
1. Mở Android Studio nếu phần mềm chưa được mở sẵn.
2. Trong cửa sổ chính **Welcome to Android Studio**, nhấn "**Start a new Android Studio project**".
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** vào **Application name**.



4. Xác nhận rằng **Project location** mặc định là nơi bạn muốn lưu trữ ứng dụng **Hello World** và các dự án khác trong **Android Studio**, hoặc thay đổi thành thư mục bạn mong muốn.
5. Chấp nhận **android.example.com** mặc định làm **Company Domain** hoặc tạo ra một tên miền tùy chỉnh.

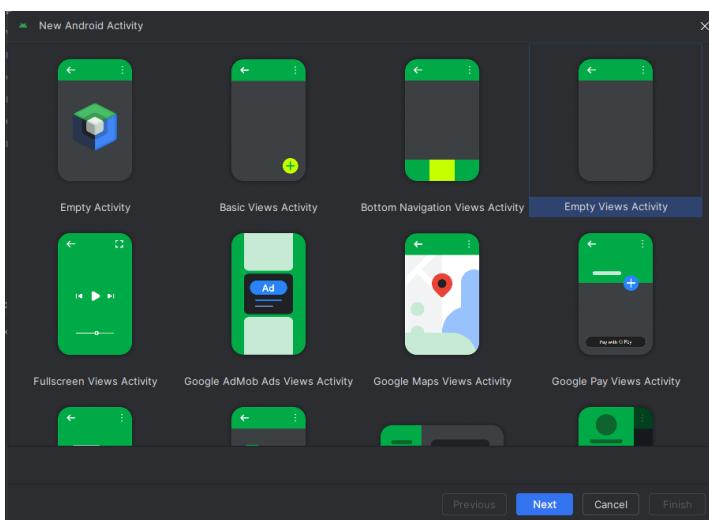
Nếu bạn không có kế hoạch phát hành ứng dụng, có thể giữ nguyên mặc định. Lưu ý rằng thay đổi package name sau này sẽ tốn công sức.

6. Bỏ chọn các tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấn **Next**.
7. Trong màn hình **Target Android Devices**, đảm bảo rằng **Phone and Tablet** đã được chọn. Đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** làm Minimum SDK. Nếu chưa đúng, sử dụng popup menu để chọn.

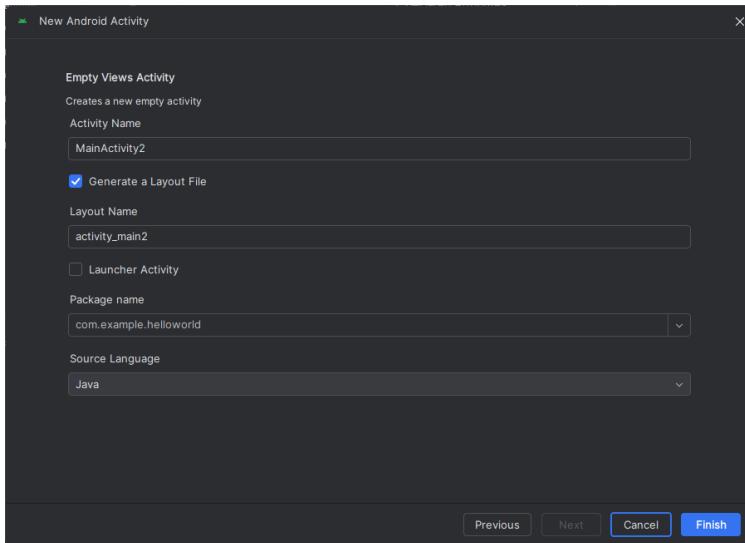


Đây là các thiết lập được sử dụng trong các ví dụ của khóa học này. Tại thời điểm hiện tại, các thiết lập này giúp ứng dụng Hello World của bạn tương thích với 97% thiết bị Android đang hoạt động trên Google Play Store.

8. Bỏ chọn **Include Instant App support** và tất cả các tùy chọn khác, sau đó nhấn **Next**. Nếu dự án của bạn yêu cầu các thành phần bổ sung cho target SDK đã chọn, Android Studio sẽ tự động cài đặt chúng.
9. Cửa sổ **Add an Activity** xuất hiện. **Activity** là một thành phần quan trọng trong kiến trúc ứng dụng Android, đại diện cho một tác vụ cụ thể mà người dùng có thể thực hiện. Mỗi **Activity** thường được liên kết với một layout XML, xác định cách các UI components hiển thị trên giao diện người dùng. **Android Studio** cung cấp các **Activity templates** để giúp bạn khởi tạo nhanh ứng dụng. Đối với dự án Hello World, chọn **Empty Activity**, sau đó nhấn **Next**.



10. Màn hình **Configure Activity** xuất hiện (giao diện có thể khác nhau tùy theo mẫu **Activity template** bạn đã chọn ở bước trước). Theo mặc định, **Empty Activity** được đặt tên là **MainActivity**. Bạn có thể thay đổi tên nếu muốn, nhưng trong bài học này, chúng ta sẽ sử dụng **MainActivity**.



11. Đảm bảo rằng tùy chọn **Generate Layout File** được chọn. Tên **layout** mặc định là **activity_main**. Bạn có thể thay đổi nếu muốn, nhưng trong bài học này, chúng ta sẽ sử dụng **activity_main**.

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

13. Nhấn **Finish**.

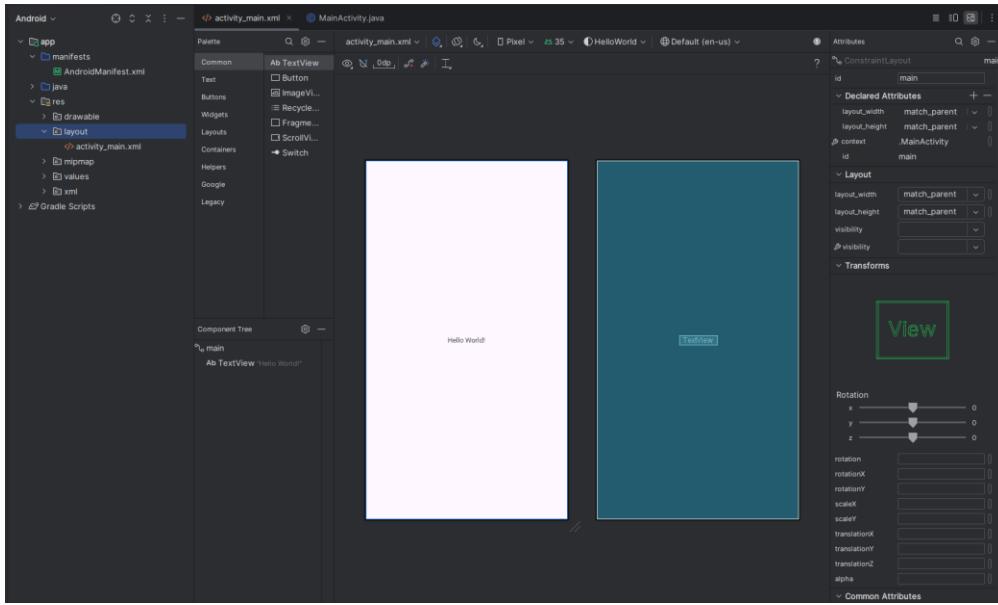
Android Studio sẽ tạo thư mục chứa dự án và tiến hành thực hiện dự án bằng **Gradle** (quá trình này có thể mất một vài phút).

Mẹo: Xem trang **Configure your build** dành cho nhà phát triển để biết thêm thông tin chi tiết.

Bạn có thể thấy cửa sổ "**Tip of the day**" hiển thị các phím tắt và mẹo hữu ích. Nhấn **Close** để đóng cửa sổ này.

Giao diện chỉnh sửa trong Android Studio xuất hiện. Làm theo các bước sau:

1. Nhấp vào tab **activity_main.xml** để mở trình chỉnh sửa **layout**.
2. Chọn tab **Design** trong trình chỉnh sửa **layout** (nếu chưa được chọn) để xem giao diện đồ họa của **layout**.



3. Nhấp vào tab MainActivity.java để xem trình chỉnh sửa code như hình dưới đây.

```

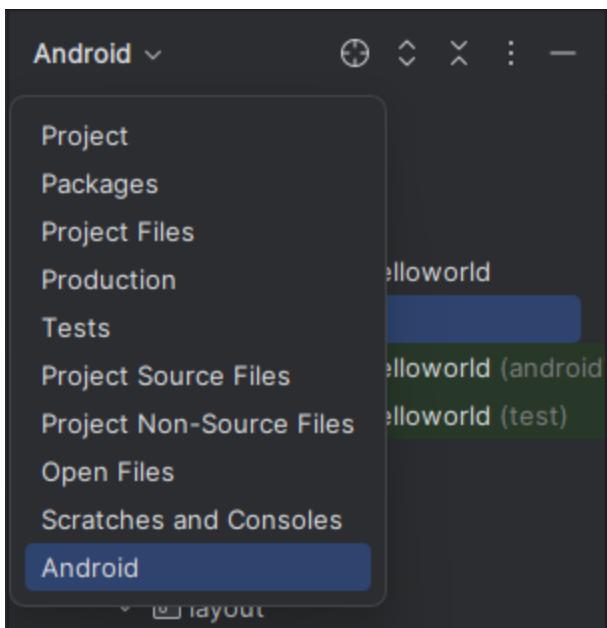
1 package com.example.helloworld;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        EdgeToEdge.enable( $this$enableEdgeToEdge, this );
11        setContentView(R.layout.activity_main);
12        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
13            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
14            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
15            return insets;
16        });
17    }
18}

```

2.2 Khám phá Project > Android Pane

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

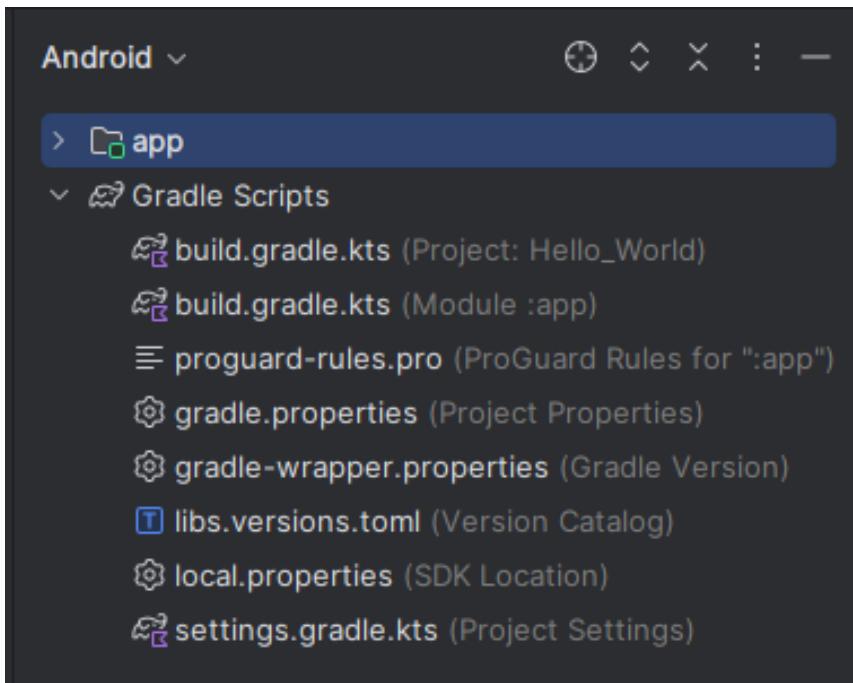
1. Nếu chưa được chọn, nhấp vào tab **Project** trong cột tab dọc bên trái của cửa sổ **Android Studio** để mở **Project pane**.
2. Để xem dự án theo cấu trúc thư mục tiêu chuẩn của **Android project hierarchy**, chọn **Android** từ menu thả xuống ở phía trên của **Project pane**, như hình dưới đây.



2.3 Khám phá thư mục Gradle Scripts

Hệ thống **Gradle build** trong **Android Studio** giúp dễ dàng tích hợp các **tệp nhị phân bên ngoài** hoặc các **module thư viện** vào quá trình **build** dưới dạng **dependencies**.

Khi bạn tạo một **app project** lần đầu tiên, **Project > Android pane** sẽ hiển thị với thư mục **Gradle Scripts** được mở rộng như hình dưới đây.



Thực hiện các bước sau để tìm hiểu về **Gradle system**:

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

Đây là nơi chứa các **tùy chọn cấu hình** chung cho tất cả các **module** trong dự án. Mỗi dự án trong **Android Studio** có một **top-level Gradle build file** duy nhất. Hầu hết thời gian, bạn không cần thay đổi tệp này, nhưng hiểu nội dung của nó sẽ hữu ích.

Mặc định, tệp **build.gradle** cấp cao nhất sử dụng **buildscript block** để xác định các **Gradle repositories** và **dependencies** dùng chung cho toàn bộ dự án. Nếu một **dependency** không phải là thư viện cục bộ hoặc **file tree**, **Gradle** sẽ tìm kiếm tệp trong các **online repositories** được khai báo trong **repositories block** của tệp này. Mặc định, các dự án mới trong **Android Studio** sử dụng **JCenter** và **Google** (bao gồm **Google Maven Repository**) làm vị trí lưu trữ.

```
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}
```

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

Ngoài tệp **build.gradle** cấp dự án, mỗi **module** cũng có một tệp **build.gradle** riêng, tệp này cho phép bạn cấu hình **cài đặt build** cho từng **module cụ thể** (trong dự án **HelloWorld**, chỉ có một module duy nhất). Việc cấu hình các **cài đặt build** giúp bạn tùy chỉnh cách đóng gói ứng dụng, chẳng hạn như thêm các **build types** hoặc **product flavors**. Bạn cũng có thể ghi đè các thiết lập trong **AndroidManifest.xml** hoặc **build.gradle** cấp dự án.

Tệp này thường là tệp chỉnh sửa khi bạn muốn thay đổi các cấu hình cấp ứng dụng, chẳng hạn như khai báo các **dependencies** trong phần **dependencies**. Bạn có thể thêm một **library dependency** bằng nhiều cách khác nhau, mỗi cách sẽ hướng dẫn **Gradle** cách sử dụng thư viện. Ví dụ, Lệnh implementation fileTree(dir: 'libs', include: ['*.jar']) sẽ thêm tất cả các tệp “.jar” trong thư mục **libs** làm dependency của ứng dụng.

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

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "com.example.android.helloworld"
        minSdkVersion 15
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
            "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles
                getDefaultProguardFile('proguard-android.txt'),
                'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation
        'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'

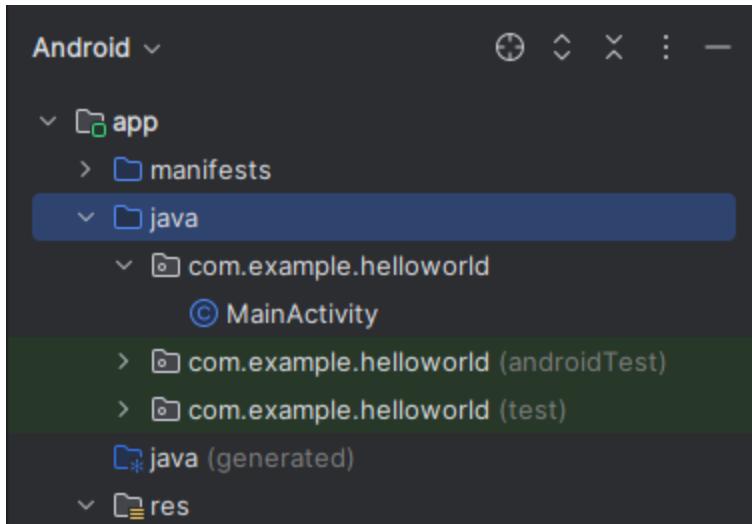
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation
        'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

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

2.4 Khám phá thư mục app và res

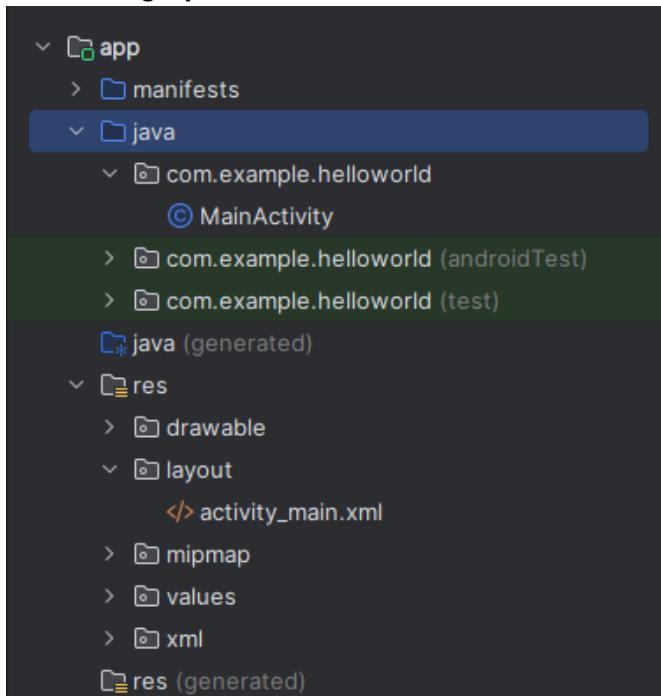
Tất cả mã nguồn (source code) và tài nguyên (resources) của ứng dụng được lưu trữ trong thư mục app và res.

1. Mở rộng thư mục **app**, thư mục **java**, và thư mục **com.example.android.helloworld** để tìm tệp **MainActivity.java**. Nhấp đúp vào tệp để mở nó trong **code editor**.



Thư mục **java** chứa các tệp **Java class** trong ba thư mục con, như hình trên. Thư mục **com.example.android.helloworld** (hoặc domain name bạn đã cấu hình) chứa tất cả các tệp của **app package**. Hai thư mục còn lại được sử dụng để kiểm thử (**testing**) và sẽ được đề cập trong bài học khác. Trong ứng dụng **Hello World**, chỉ có một package duy nhất và nó chứa tệp **MainActivity.java**. Tên của Activity đầu tiên (màn hình đầu tiên) mà người dùng nhìn thấy, đồng thời khởi tạo các tài nguyên toàn cục của ứng dụng, theo thông lệ thường được đặt là **MainActivity** (phần mở rộng tệp có thể bị ẩn trong **Project > Android pane**).

2. Mở rộng thư mục **res**, sau đó mở rộng thư mục **layout**, nhấp đúp vào tệp **activity_main.xml** để mở nó trong **layout editor**.



Thư mục **res** chứa các **tài nguyên (resources)** như **layouts**, **strings**, và **images**. Một **Activity** thường được liên kết với một **giao diện người dùng (UI views)** được định nghĩa trong một tệp **XML**. Tệp này thường được đặt tên theo **Activity** tương ứng.

2.5 Khám phá thư mục manifests

Thư mục **manifests** chứa các tệp cung cấp **thông tin quan trọng** về ứng dụng cho **hệ thống Android**. Hệ thống cần những thông tin này trước khi có thể thực thi bất kỳ đoạn 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 (components)** của ứng dụng Android. Mỗi **thành phần** trong ứng dụng, chẳng hạn như **Activity**, phải được khai báo trong tệp **XML** này. Trong các bài học khác, bạn sẽ **chỉnh sửa tệp manifest** để thêm **tính năng (features)** và **quyền (permissions)** cho ứng dụng. Để tìm hiểu thêm, hãy xem tài liệu **App Manifest Overview**.

Nhiệm vụ 3: sử dụng thiết bị ảo

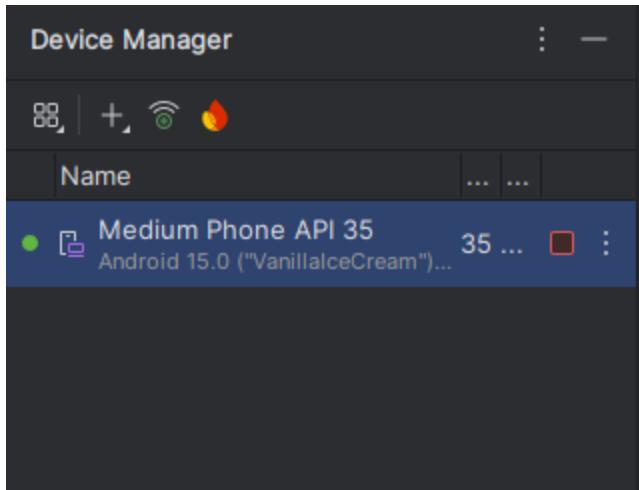
Trong nhiệm vụ này, bạn sẽ sử dụng Android Virtual Device (AVD) manager để khởi tạo một thiết bị ảo (còn được gọi là trình giả lập) mô phỏng cấu hình của một thiết bị Android cụ thể. Sau đó, bạn sẽ sử dụng thiết bị ảo này để chạy ứng dụng. Lưu ý rằng **Android Emulator** có yêu cầu hệ thống bổ sung so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

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

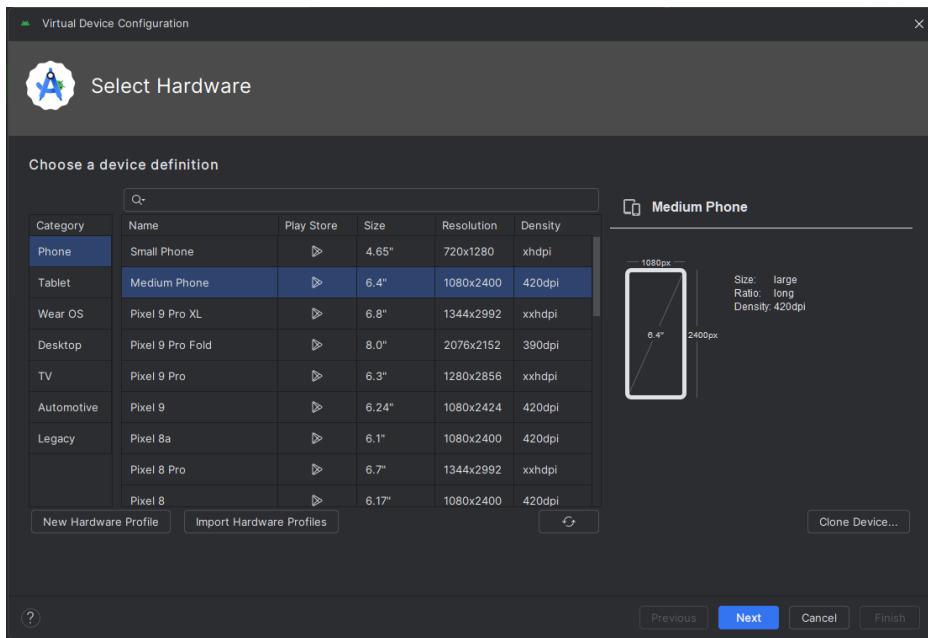
3.1 khởi tạo Thiết bị Android ảo(AVD)

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

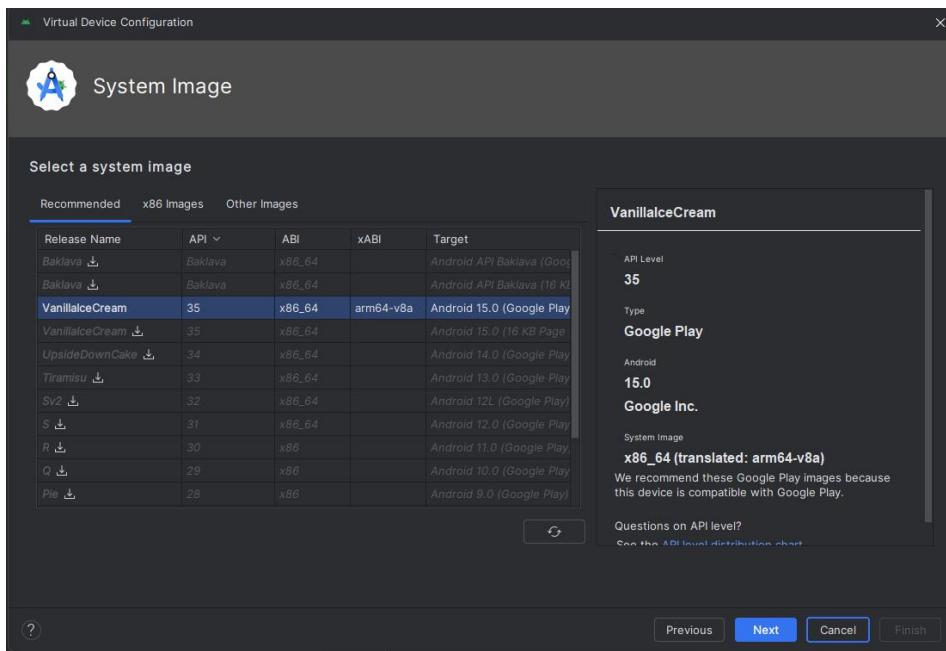
1. Trong **Android Studio**, chọn **Tools > Android > AVD Manager** hoặc nhấp vào biểu tượng **AVD Manager** trên thanh công cụ. Màn hình **Your Virtual Devices** sẽ xuất hiện. Nếu bạn đã tạo **thiết bị ảo (virtual devices)** trước đó, danh sách các thiết bị sẽ hiển thị. Ngược lại, nếu chưa có thiết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



- Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** sẽ xuất hiện, hiển thị danh sách các **thiết bị phần cứng được cấu hình sẵn**. Đối với mỗi thiết bị, bảng thông tin sẽ cung cấp các thông số sau: Kích thước màn hình (**Size**), Độ phân giải (**Resolution**), Mật độ điểm ảnh (**Density**).



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



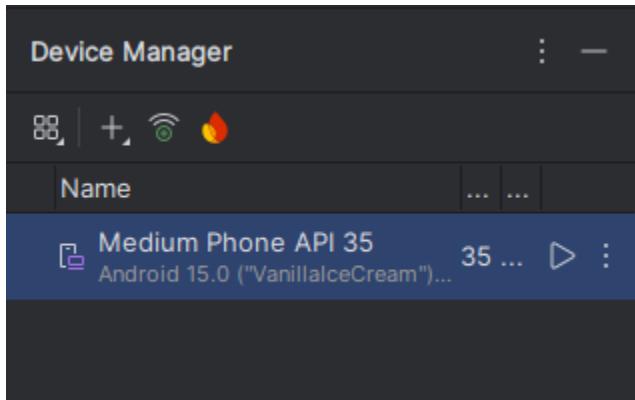
Có nhiều phiên bản hệ điều hành hơn so với những phiên bản hiển thị trong tab **Recommended**. Bạn có thể xem thêm trong các tab **x86 Images** và **Other Images**.

Nếu có liên kết **Download** bên cạnh phiên bản hệ điều hành bạn muốn sử dụng, nghĩa là phiên bản đó chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống và chọn **Finish** khi quá trình tải hoàn tất. Sau khi chọn hệ thống ảnh, nhấp vào **Next**. Màn hình **Android Virtual Device (AVD)** sẽ xuất hiện. Bạn có thể thay đổi **tên của thiết bị ảo (AVD)** nếu muốn. Kiểm tra lại cấu hình của bạn, sau đó nhấp vào **Finish** để hoàn tất.

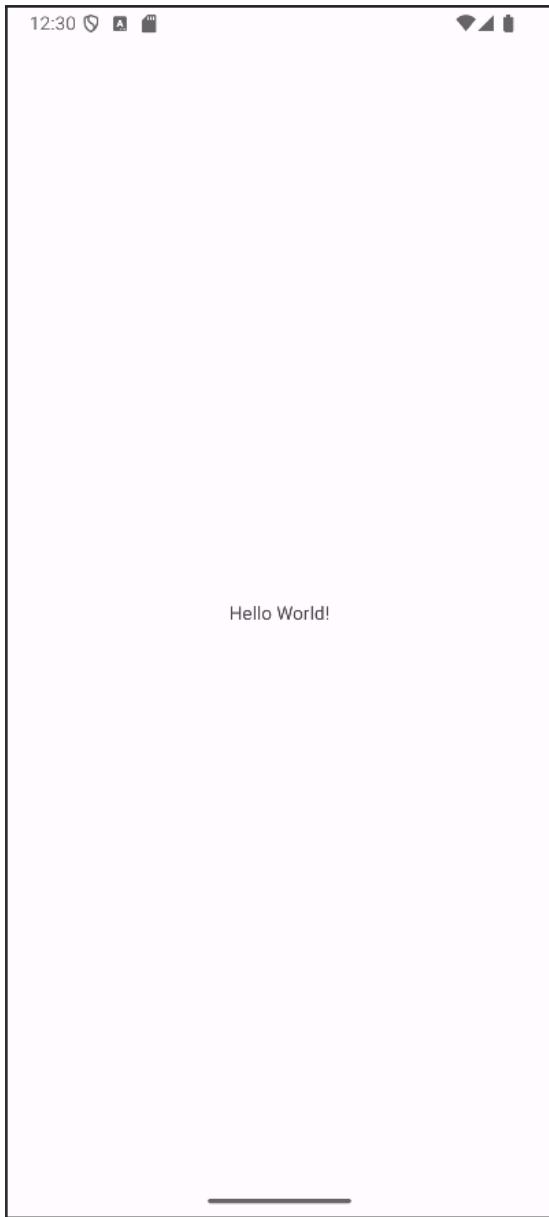
3.2 Khởi chạy ứng dụng trên thiết bị ảo

Ở nhiệm vụ này, bạn sẽ được khởi chạy ứng dụng Hello World.

- Trong Android Studio, chọn Run > Run App hoặc nhấp chuột vào biểu tượng Run trên thanh công cụ.
- Trong cửa sổ Select Deployment Target, dưới Available Virtual Devices, Chọn thiết bị ảo mà bạn vừa tạo và nhấn OK.



Trình giả lập sẽ khởi động giống như một thiết bị thực. Tùy thuộc vào tốc độ của máy tính, quá trình này có thể mất một khoảng thời gian. Sau khi ứng dụng được **build**, Android Studio sẽ tải ứng dụng lên trình giả lập và chạy nó ngay khi trình giả lập sẵn sàng.
Bạn sẽ thấy ứng dụng **Hello World** hiển thị như trong hình minh họa bên dưới.



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

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

Những gì bạn cần:

- Một thiết bị Android, chẳng hạn như điện thoại hoặc máy tính bảng.
- Cáp dữ liệu để kết nối thiết bị Android của bạn với máy tính thông qua cổng USB.
- Nếu bạn đang sử dụng hệ điều hành **Linux** hoặc **Windows**, có thể cần thực hiện thêm một số bước để chạy ứng dụng trên thiết bị phần cứng. Hãy kiểm tra tài liệu **Using Hardware Devices**. Bạn cũng có thể cần cài đặt trình điều khiển USB (USB driver) phù

hợp cho thiết bị của mình. Đối với trình điều khiển USB trên Windows, hãy tham khảo **OEM USB Drivers**.

4.1 Bật chế độ gỡ lỗi USB

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

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

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

4.2 Chạy ứng dụng trên thiết bị thực

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

Các bước thực hiện:

1. **Kết nối thiết bị** với máy tính bằng cáp USB.
2. **Nhấn vào nút Run** trên thanh công cụ. Cửa sổ **Select Deployment Target** sẽ xuất hiện, hiển thị danh sách các trình giả lập và thiết bị được kết nối.
3. **Chọn thiết bị của bạn**, sau đó nhấn **OK**.

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

Xử lý sự cố

Nếu Android Studio không nhận diện thiết bị của bạn, hãy thử:

1. Rút và cắm lại cáp USB.
2. Khởi động lại Android Studio.

Nếu máy tính không tìm thấy thiết bị hoặc báo "unauthorized", hãy làm như sau:

1. Rút cáp USB khỏi thiết bị.
2. Trên thiết bị, mở Developer Options trong ứng dụng Cài đặt.
3. Nhấn vào Revoke USB Debugging authorizations.
4. Kết nối lại thiết bị với máy tính.
5. Khi có thông báo yêu cầu cấp quyền, nhấn Cho phép.

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**.

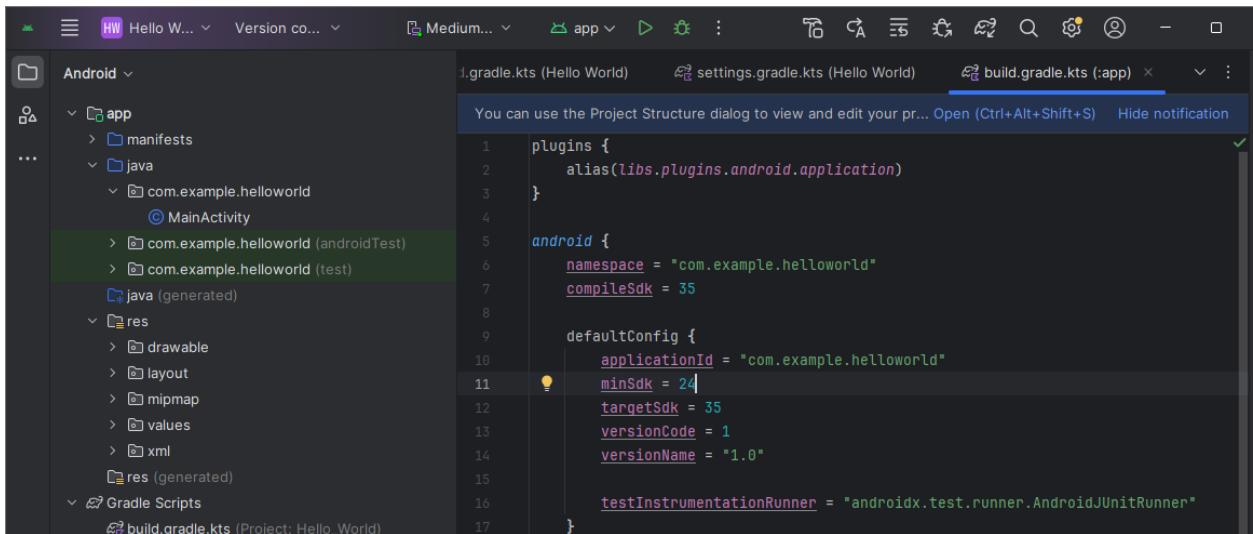
Nhiệm vụ 5: Thay đổi cấu hình Gradle của ứng dụng

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

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

Thực hiện các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu nó chưa được mở và nhấp đúp vào tệp **build.gradle(Module:app)**.
Nội dung của tệp sẽ hiển thị trong **code editor**.
2. Trong khối **defaultConfig**, thay đổi giá trị của **minSdkVersion** thành **17** như ví dụ dưới đây (giá trị ban đầu là **15**).



```
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"
    }
}
```

Code editor sẽ hiển thị một thanh thông báo ở phía trên cùng với liên kết **Sync Now**.

5.2 Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi trong các tệp **cấu hình build** của dự án, **Android Studio** yêu cầu đồng bộ hóa (sync) các tệp dự án để nhập các thay đổi cấu hình build và thực hiện kiểm tra nhằm đảm bảo cấu hình mới không gây lỗi biên dịch (**build errors**).

Để đồng bộ hóa tệp dự án, nhấn **Sync Now** trong thanh thông báo xuất hiện sau khi thực hiện thay đổi (**như trong hình trước đó**), hoặc nhấp vào biểu tượng **Sync Project with Gradle Files** trên **toolbar**.

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

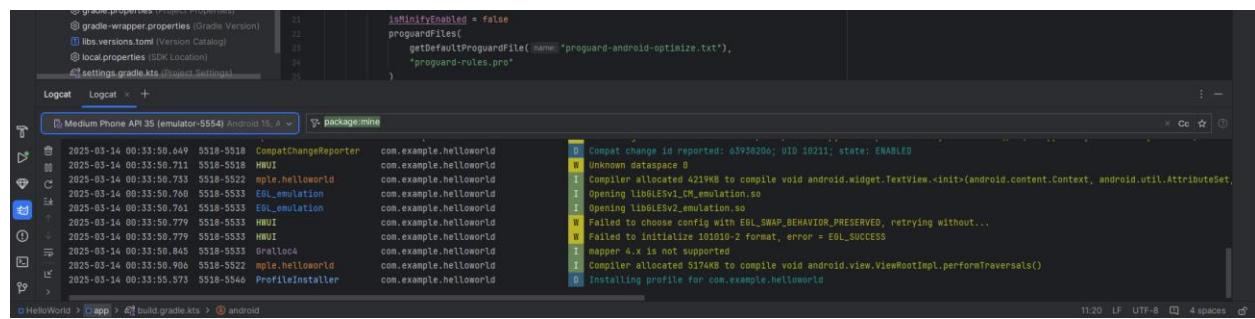
Để tìm hiểu sâu hơn về **Gradle**, hãy tham khảo tài liệu **Build System Overview** và **Configuring Gradle Builds**.

Nhiệm vụ 6: Thêm câu lệnh Log vào ứng dụng

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh **Log** vào ứng dụng, giúp hiển thị thông điệp trong **Logcat pane**. Các thông báo log là một công cụ **debugging** mạnh mẽ, giúp kiểm tra giá trị biến, luồng thực thi (**execution paths**) và báo cáo ngoại lệ (**exceptions**).

6.1 Hiển thị Logcat pane

Để xem **Logcat pane**, nhấn vào tab **Logcat** ở thanh công cụ dưới cùng của cửa sổ **Android Studio**, như minh họa trong hình dưới đây.



Trong hình minh họa trên:

1. **Tab Logcat** dùng để mở và đóng **Logcat pane**, nơi hiển thị thông tin về ứng dụng trong quá trình chạy. Nếu bạn thêm các câu lệnh **Log** vào ứng dụng, các thông báo **Log messages** sẽ xuất hiện tại đây.
2. **Menu Log level** được đặt ở chế độ **Verbose** (mặc định), hiển thị tất cả **Log messages**. Các tùy chọn khác bao gồm **Debug**, **Error**, **Info**, và **Warn**.

6.2 Thêm câu lệnh Log vào ứng dụng của bạn

Các câu lệnh **Log** trong mã nguồn của ứng dụng sẽ hiển thị thông báo trong **Logcat pane**. Ví dụ:

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

Các thành phần của thông điệp log:

- **Log:** Lớp **Log** dùng để gửi thông điệp log đến **Logcat pane**.

- **d**: Mức log **Debug**, dùng để lọc và hiển thị log trong **Logcat pane**. Các mức log khác bao gồm **e** (Error) – lỗi nghiêm trọng, **w** (Warn) – cảnh báo, **i** (Info) – thông tin.
- "**MainActivity**": Đối số đầu tiên là **tag**, giúp lọc thông điệp trong **Logcat pane**. Thông thường, tag là tên của **Activity** chứa log, nhưng có thể đặt tên bất kỳ giúp dễ dàng gỡ lỗi.

Theo quy ước, các **log tags** thường được định nghĩa dưới dạng **hằng số** trong **Activity**.

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

- "**Hello world**": Đối số thứ hai là thông điệp thực tế được hiển thị trong **Logcat pane**.

Thực hiện các bước sau:

1. Mở ứng dụng **Hello World** trong **Android Studio**, sau đó mở tệp **MainActivity.java**.
2. Để tự động thêm **import statements** cần thiết (ví dụ: `android.util.Log` để sử dụng **Log**), Chọn **File > Settings** trên Windows, hoặc **Android Studio > Preferences** trên MacOS.
3. Chọn **Editor > General > Auto Import**. Tích chọn tất cả các hộp kiểm và Thiết lập **Insert imports on paste** thành **All**.
4. Nhấn **Apply**, sau đó chọn **OK** để lưu thiết lập.
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);
    setContentView(R.layout.activity_main);
    Log.d("MainActivity", "Hello World");
}
```

1. Nếu khung Logcat chưa được mở, nhập vào tab Logcat ở dưới cùng của Android Studio để mở nó.
2. Kiểm tra rằng tên mục tiêu và tên gói của ứng dụng là chính xác.
3. Thay đổi mức Log trong khung Logcat thành Debug (hoặc để là Verbose vì có rất ít thông báo log).
4. Chạy ứng dụng của bạn.

Thông báo sau đây sẽ xuất hiện trong khung Logcat:

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

Tóm tắt

- Để cài đặt Android Studio, truy cập Android Studio và làm theo các hướng dẫn để tải xuống và cài đặt.
- Khi tạo ứng dụng mới, đảm bảo rằng API 15: Android 4.0.3 IceCreamSandwich được đặt làm SDK tối thiểu.
- Để xem hệ thống phân cấp Android của ứng dụng trong khung Dự án, nhấp vào tab Dự án trong cột tab dọc, sau đó chọn Android trong menu bật lên ở trên cùng.
- Chỉnh sửa tệp build.gradle (Mô-đun: Ứng dụng) khi bạn cần thêm thư viện mới vào dự án hoặc thay đổi phiên bản thư viện.
- Tất cả mã và tài nguyên cho ứng dụng nằm trong thư mục app và res. Thư mục java bao gồm các hoạt động, kiểm tra và các thành phần khác trong mã nguồn Java. Thư mục res chứa các tài nguyên, chẳng hạn như bộ cục, chuỗi và hình ảnh.
- Chỉnh sửa tệp AndroidManifest.xml để thêm các thành phần và quyền tính năng vào ứng dụng Android của bạn. Tất cả các thành phần cho một ứng dụng, chẳng hạn như nhiều hoạt động, phải được khai báo trong tệp XML này.
- Sử dụng trình quản lý Thiết bị ảo Android (AVD) để tạo một thiết bị ảo (còn được gọi là trình giả lập) để chạy ứng dụng của bạn.
- Thêm các câu lệnh Log vào ứng dụng của bạn, hiển thị các thông báo trong khung Logcat như một công cụ cơ bản để gỡ lỗi.
- Để chạy ứng dụng của bạn trên một thiết bị Android vật lý bằng Android Studio, bật Gỡ lỗi USB trên thiết bị. Mở Cài đặt > Giới thiệu về điện thoại và chạm vào Số bản dựng bảy lần. Quay lại màn hình trước (Cài đặt) và nhấn vào Tùy chọn nhà phát triển. Chọn Gỡ lỗi USB.

1.2 Phân A: Giao diện người dùng tương tác đầu tiên của bạn

Giới thiệu Giao diện người dùng (UI) xuất hiện trên màn hình của một thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là views — mỗi yếu tố của màn hình là một View. Lớp View đại diện cho khối xây dựng cơ bản cho tất cả các thành phần UI và là lớp cơ bản cho các lớp cung cấp các thành phần UI tương tác như nút, hộp kiểm và trường nhập văn bản. 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 yếu tố có thể nhấp khác (chẳng hạn như **RadioButton**, **CheckBox** và **Spinner**) để cung cấp hành vi tương tác.

- **ScrollView** và **RecyclerView** để hiển thị các mục có thể cuộn.
- **ImageView** để hiển thị hình ảnh.
- **ConstraintLayout** và **LinearLayout** để chứa các yếu tố View khác và định vị chúng.

Mã Java hiển thị và điều khiển giao diện người dùng được chứa trong một lớp mở rộng Activity. Một Activity thường được liên kết với một bộ cục các chế độ xem giao diện người dùng được định nghĩa là một tệp XML (eXtended Markup Language). Tệp XML này thường được đặt tên theo Activity của nó và định nghĩa bộ cục của các yếu tố View trên màn hình.

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị một bộ cục được định nghĩa trong tệp bộ cục activity_main.xml, bao gồm một TextView với văn bản "Hello World".

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

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

Những gì bạn nên biết trước

- 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 có hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bộ cục để thiết kế bộ cục.
- Cách chỉnh sửa bộ cục trong XML.
- Nhiều thuật ngữ mới. Kiểm tra từ điển thuật ngữ và khái niệm cho các định nghĩa thân thiện.

Những gì bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử Button và một TextView vào bộ cục.
- Thao tác từng phần tử trong ConstraintLayout để hạn chế chúng vào các lề và các phần tử khác.
- Thay đổi thuộc tính phần tử giao diện người dùng.
- Chỉnh sửa bộ cục của ứng dụng trong XML.
- Trích xuất các chuỗi cứng vào tài nguyên chuỗi.

- Thực hiện các phương pháp xử lý click 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, nó sẽ hiển thị một thông báo ngắn (một Toast) trên màn hình. Nhấn vào nút thứ hai sẽ tăng bộ đếm "click" hiển thị trong TextView, bắt đầu từ số không.

Đây là hình ảnh của ứng dụng đã 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 sẽ 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:

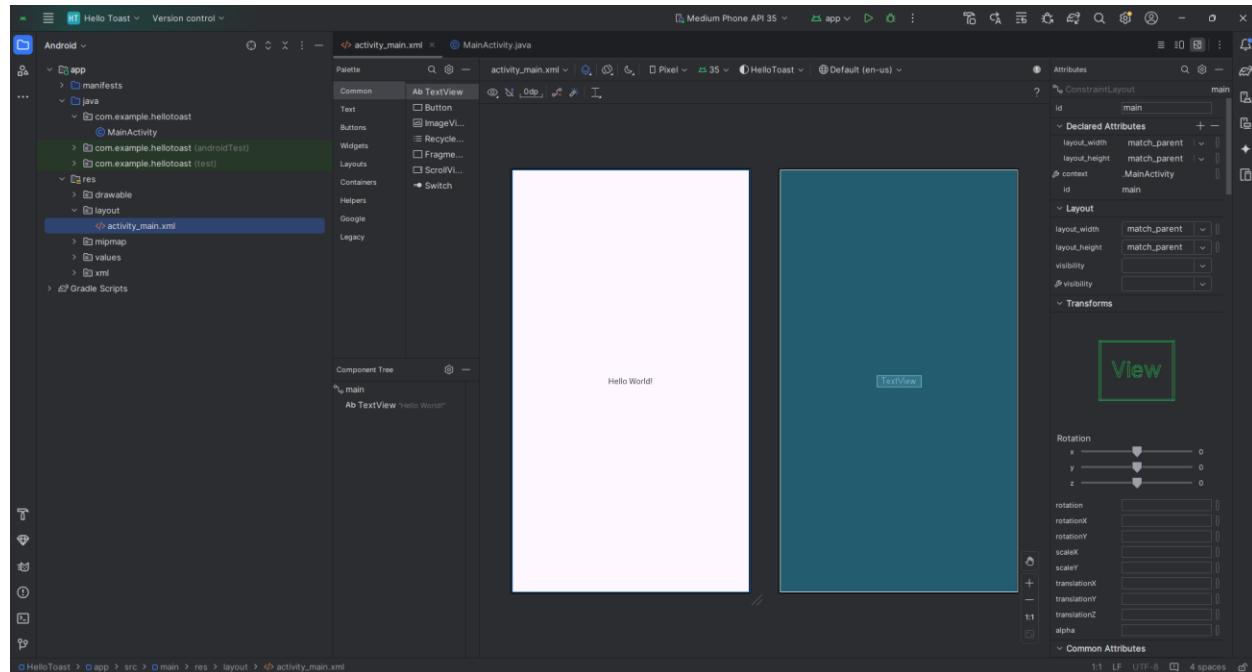
Thuộc tính	Giá trị
Tên ứng dụng	Hello Toast
Tên công ty	com.example.android (hoặc miền của riêng bạn)
SDK tối thiểu cho điện thoại và máy tính bảng	API15: Android 4.0.3 IceCreamSandwich
Mẫu	Empty Activity
Hộp tệp bô cục tạo	Đã chọn
Hộp Tương thích Ngược	Đã chọn

- Chọn Chạy > Chạy ứng dụng 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 giao diện người dùng (UI) của một ứng dụng. Nó cho phép bạn kéo các yếu tố vào thiết kế trực quan và chế độ xem bản vẽ, định vị chúng trong bố cục, thêm các ràng buộc và đặt thuộc tính. Các ràng buộc xác định vị trí của một yếu tố UI trong bố cục. Một ràng buộc đại diện cho một kết nối hoặc cẩn chỉnh đến một view khác, bố cục cha, hoặc một hướng dẫn vô hình.

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



- Trong thư mục app > res > layout trong khung Dự án > Android, nhấp đúp vào tệp activity_main.xml để mở nó, nếu nó chưa được mở.
- Nhấp vào tab Design nếu nó chưa được chọn. Bạn sử dụng tab Design để thao tác các yếu tố và bố cục, và tab Text để chỉnh sửa mã XML cho bố cục.
- Khung Palettes hiển thị các yếu tố UI mà bạn có thể sử dụng trong bố cục của ứng dụng.
- Khung Component tree hiển thị hệ thống phân cấp các yếu tố UI. Các yếu tố View được tổ chức thành một hệ thống phân cấp cây của các phần tử cha và con, trong đó một phần tử con kế thừa các thuộc tính của phần tử cha của nó. Trong hình trên, TextView là phần tử con của ConstraintLayout. Bạn sẽ tìm hiểu về những yếu tố này sau trong bài học này.
- Các khung thiết kế và bản vẽ của trình chỉnh sửa bố cục hiển thị các yếu tố UI trong bố cục. Trong hình trên, bố cục chỉ hiển thị một yếu tố: một TextView hiển thị "Hello World".
- Tab Attributes hiển thị khung Attributes để đặt các thuộc tính cho một yếu tố UI.

Mẹo: Xem Building a UI with Layout Editor để biết chi tiết về cách sử dụng trình chỉnh sửa bố cục, và Meet Android Studio để xem đầy đủ tài liệu về Android Studio.

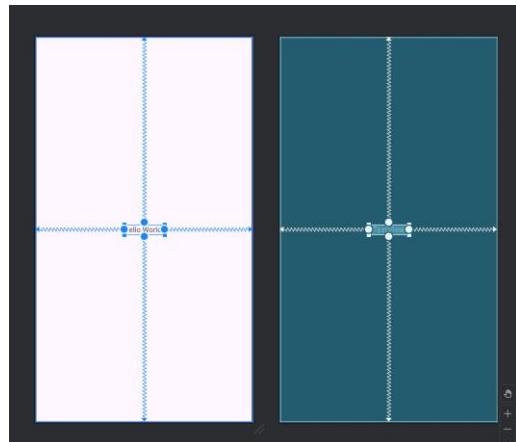
Nhiệm vụ 2: Thêm các yếu tố View trong trình chỉnh sửa bố cục

Trong nhiệm vụ này, bạn sẽ tạo bố cục giao diện người dùng cho ứng dụng HelloToast trong trình chỉnh sửa bố cục bằng cách sử dụng các tính năng ConstraintLayout. Bạn có thể tạo các ràng buộc theo cách thủ công, như được chỉ ra sau, hoặc tự động sử dụng công cụ Autoconnect.

2.1 Kiểm tra các ràng buộc của yếu tố

Làm theo các bước sau:

- Mở tệp activity_main.xml từ khung Dự án > Android nếu nó chưa được mở. Nếu tab Design chưa được chọn, hãy nhấp vào nó. Nếu không có bản vẽ, nhấp vào nút Chọn Bề mặt Thiết kế trên thanh công cụ và chọn Thiết kế + Bản vẽ.
- Công cụ Autoconnect cũng nằm trên thanh công cụ. Nó được bật theo mặc định. Đối với bước này, đảm bảo rằng công cụ không bị tắt.
- Nhấp vào nút phóng to để phóng to các khung thiết kế và bản vẽ để xem cận cảnh.
- Chọn TextView trong khung Component Tree. TextView "Hello World" được tô sáng trong các khung thiết kế và bản vẽ và các ràng buộc cho yếu tố là có thể nhìn thấy.
- Tham khảo hình động dưới đây cho bước này. Nhấp vào tay cầm tròn ở bên phải của TextView để xóa ràng buộc ngang ràng buộc view vào phía bên phải của bố cục. TextView nhảy sang bên trái vì nó không còn bị ràng buộc vào phía bên phải. Để thêm lại ràng buộc ngang, nhấp vào cùng một tay cầm và kéo một đường sang phía bên phải của bố cục.



Trong các khung thiết kế hoặc bản vẽ, các tay cầm sau xuất hiện trên yếu tố TextView:

- Constraint handle:** Để tạo một ràng buộc như được chỉ ra trong hình động trên, nhấp vào một tay cầm ràng buộc, được hiển thị dưới dạng vòng tròn ở bên của một yếu tố. Sau đó kéo tay cầm đến một tay cầm ràng buộc khác hoặc đến một giới hạn cha. Một đường gấp khúc đại diện cho ràng buộc.



- **Resizing handle:** Để thay đổi kích thước của yếu tố, kéo các tay cầm thay đổi kích thước hình vuông. Tay cầm chuyển đổi thành một góc nghiêng khi bạn đang kéo nó.

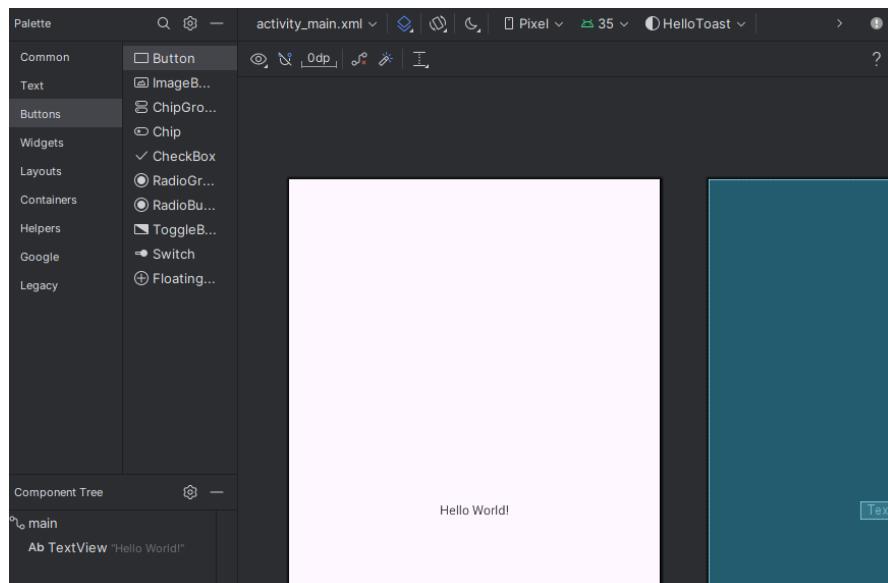


2.2 Thêm nút vào bố cục

Khi được bật, công cụ Autoconnect tự động tạo hai hoặc nhiều ràng buộc cho một yếu tố giao diện người dùng với bố cục cha. Sau khi bạn kéo yếu tố vào bố cục, nó tạo ra các ràng buộc dựa trên vị trí của yếu tố.

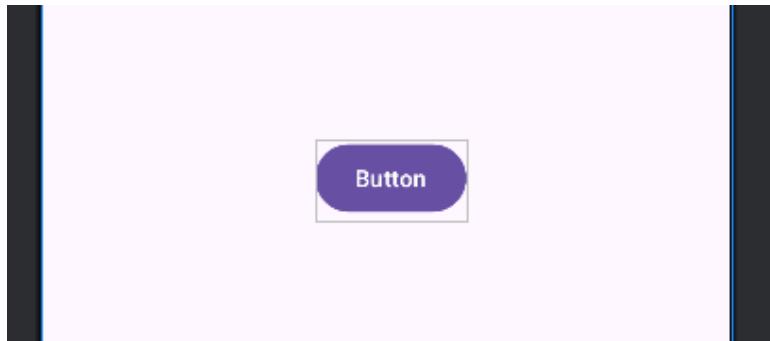
Làm theo các bước sau để thêm một nút:

1. Bắt đầu với một trang trống. Yếu tố TextView không cần thiết, vì vậy trong khi nó vẫn được chọn, nhấn phím Delete hoặc chọn Chỉnh sửa > Xóa. Jetzt haben Sie einen leeren Layout-Bereich.
2. Kéo một nút từ khung Palette đến bất kỳ vị trí nào trong bố cục. Nếu bạn thả nút 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, bên trái và bên phải của bố cục như trong hình động dưới đây.



2.3 Thêm nút thứ hai vào bố cục

1. Kéo một nút khác từ khung Palette vào giữa bố cục như hiển thị trong hình động dưới đây. Autoconnect có thể cung cấp các ràng buộc ngang cho bạn (nếu không, bạn có thể kéo chúng tự mình).
2. Kéo một ràng buộc dọc xuống dưới bố cục (tham khảo hình dưới đây).



Bạn có thể xóa các ràng buộc từ một yếu tố bằng cách chọn yếu tố đó và di chuột qua nó để hiển thị nút Xóa các Ràng buộc. Nhấp vào nút này để xóa tất cả các ràng buộc trên yếu tố đã chọn. Để xóa một ràng buộc duy nhất, nhấp vào tay cầm cụ thể đặt ràng buộc.

Để xóa tất cả các ràng buộc trong toàn bộ bố cục, nhấp vào công cụ Xóa Tất cả các Ràng buộc 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 giao diện người dùng

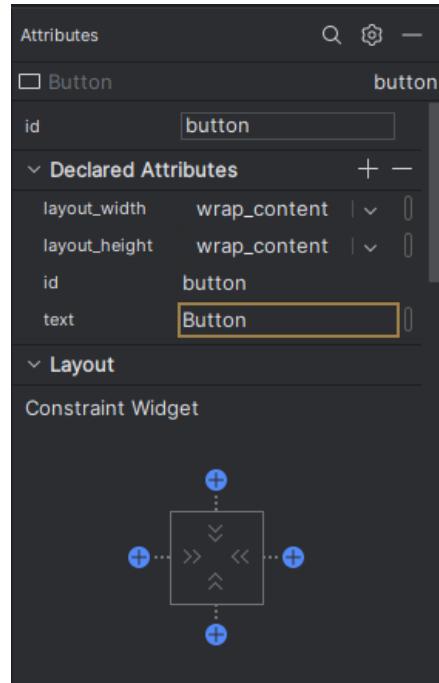
Khung Attributes cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng. Bạn có thể tìm thấy các thuộc tính (được gọi là properties) chung cho tất cả các views trong tài liệu của lớp View.

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, mà 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 trên bốn góc của một View để bạn có thể nhanh chóng thay đổi kích thước của View. Bạn có thể kéo các tay cầm trên mỗi góc của View để thay đổi kích thước của nó, nhưng làm như vậy sẽ cố định các kích thước chiều rộng và chiều cao. Tránh cố định kích thước cho hầu hết các yếu tố View, vì các kích thước cố định không thích ứng với các nội dung và kích thước màn hình khác nhau.

Thay vào đó, sử dụng khung Attributes ở bên phải của trình chỉnh sửa bố cục để chọn một chế độ kích thước không sử dụng các kích thước cố định. Khung Attributes bao gồm một bảng điều khiển kích thước vuông được gọi là trình kiểm tra view ở trên cùng. Các biểu tượng bên trong hình vuông đại diện cho các cài đặt chiều cao và chiều rộng như sau:



Trong hình trên:

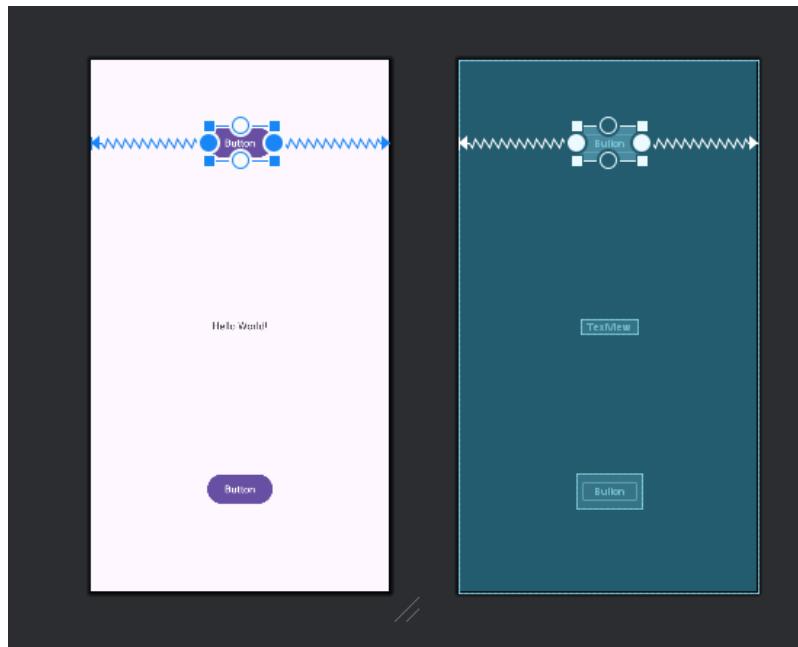
1. **Điều khiển chiều cao:** Điều khiển này xác định thuộc tính layout_height và xuất hiện ở hai đoạn trên và dưới của hình vuông. Các góc chỉ ra rằng điều khiển này được đặt thành wrap_content, có nghĩa là View sẽ mở rộng theo chiều dọc khi cần để phù hợp với nội dung của nó. "8" chỉ ra một lề tiêu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

Làm theo các bước sau:

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

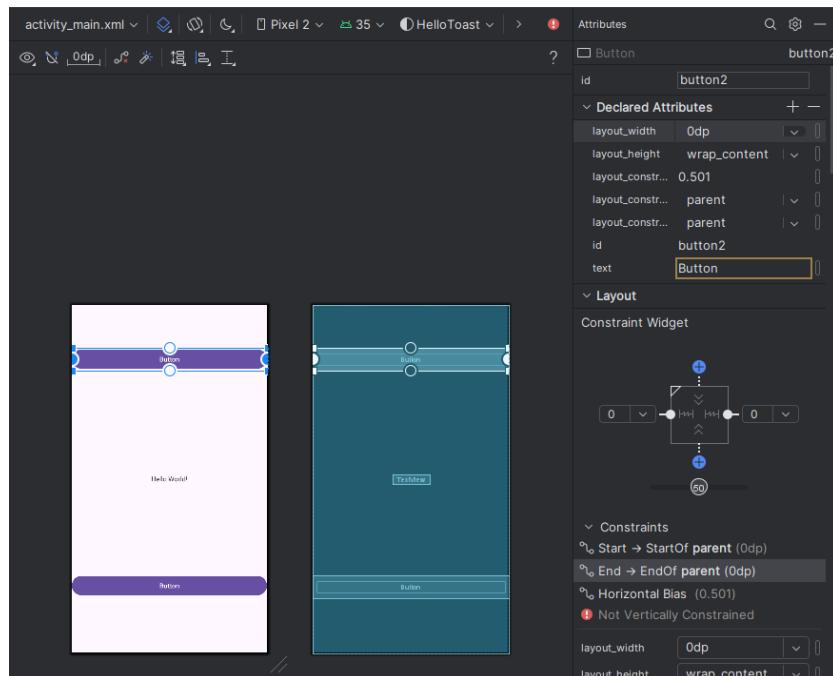


3. Nhấp vào điều khiển chiều rộng hai lần—lần nhấp đầu tiên thay đổi thành Cố định với các đường thẳng, và lần nhấp thứ hai thay đổi thành Khớp với Ràng buộc với các cuộn dây lò xo, như trong hình động dưới đây.



Do kết quả của việc thay đổi điều khiển chiều rộng, thuộc tính layout_width trong khung Attributes hiển thị giá trị match_constraint và phần tử Button kéo dài theo chiều ngang để lấp đầy không gian giữa bên trái và bên phải của bố cục.

4. Chọn nút thứ hai và thực hiện các thay đổi tương tự với layout_width như trong bước trước, như hiển thị trong hình dưới đây.



Như đã chỉ ra 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 cho bố cục, đó là một ConstraintLayout:

- Cài đặt match_constraint mở rộng yếu tố View để lấp đầy bố mẹ của nó theo chiều rộng hoặc chiều cao—tới lè 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 hẹp kích thước của yếu tố View sao cho nó vừa đủ để bao bọc nội dung của nó. Nếu không có nội dung, yếu tố View sẽ trở nên vô hình.
- Để chỉ định kích thước cố định điều chỉnh theo kích thước màn hình của thiết bị, sử dụng một số cố định của các điểm ảnh độc lập với mật độ (đơn vị dp). Ví dụ, 16dp có nghĩa là 16 điểm ảnh độc lập với mật độ.

Mẹo: Nếu bạn thay đổi thuộc tính layout_width bằng cách sử dụng menu bật lên của nó, thuộc tính layout_width sẽ được đặt thành zero vì không có kích thước được đặt. Cài đặt này tương tự như match_constraint — view có thể mở rộng hết mức có thể để đáp ứng các ràng buộc và cài đặt lề.

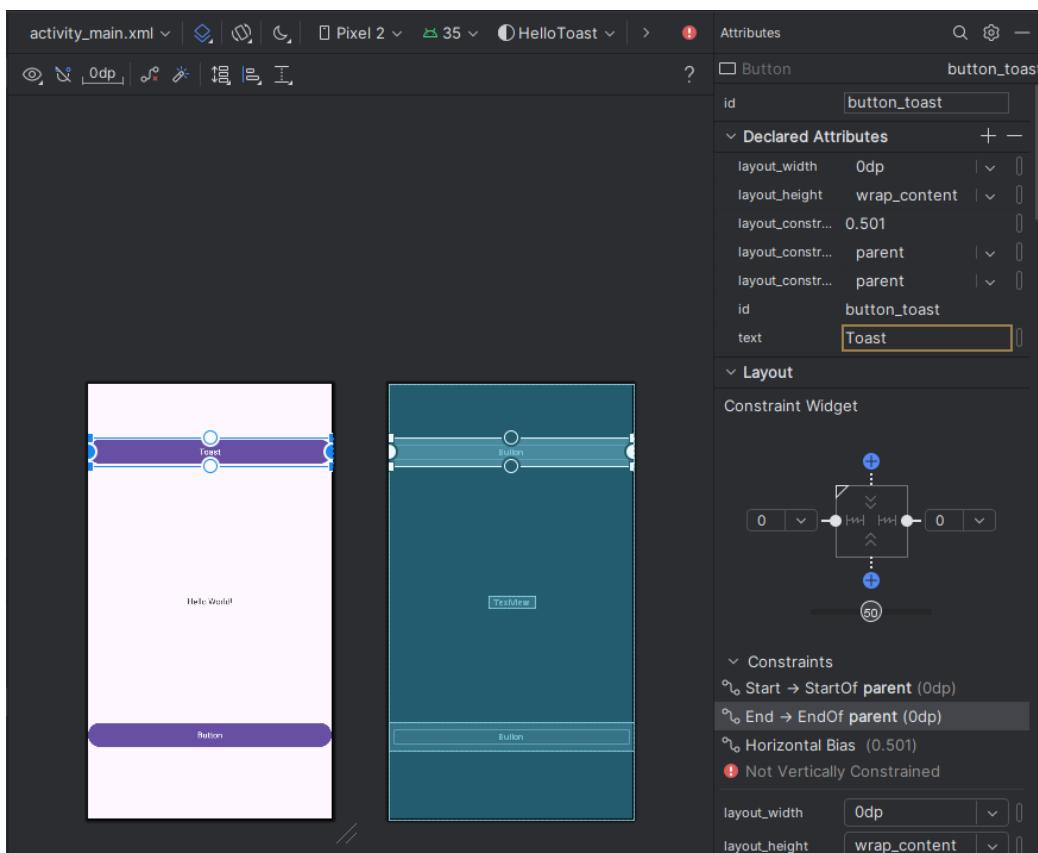
3.2 Thay đổi các thuộc tính của nút

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

Khung Attributes cung cấp quyền truy cập vào tất cả các thuộc tính mà bạn có thể gán cho một yếu tố View. Bạn có thể nhập các giá trị cho mỗi thuộc tính, chẳng hạn như android:id, background, textColor, và text.

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

1. Sau khi chọn nút đầu tiên, chỉnh sửa trường ID ở trên cùng của khung Attributes thành button_toast cho thuộc tính android:id, được sử dụng để xác định yếu tố 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.



- Thực hiện các thay đổi thuộc tính tương tự cho nút thứ hai, sử dụng button_count làm ID, Count cho thuộc tính văn bản, và cùng màu cho nền và văn bản như các bước trước.

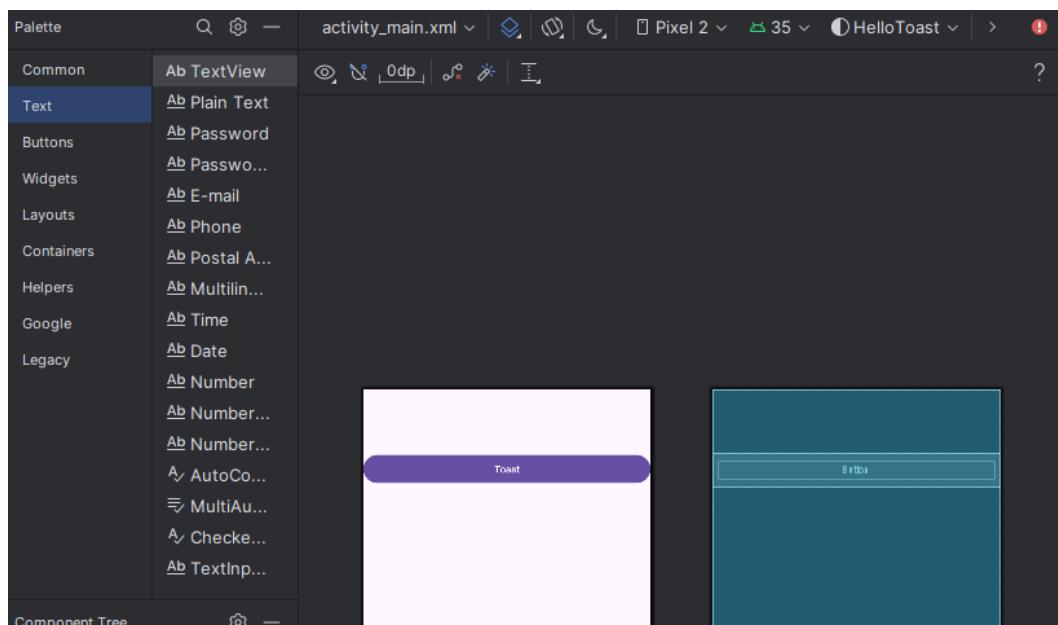
colorPrimary là màu chính của chủ đề, một trong những màu cơ bản được xác định trước trong tệp tài nguyên colors.xml. Nó được sử dụng cho thanh ứng dụng. Sử dụng các màu cơ bản cho các yếu tố giao diện người dùng khác tạo ra một giao diện người dùng đồng nhất. Bạn sẽ học thêm về chủ đề ứng dụng và Material Design trong bài học khác.

Nhiệm vụ 4: Thêm TextView và đặt các thuộc tính

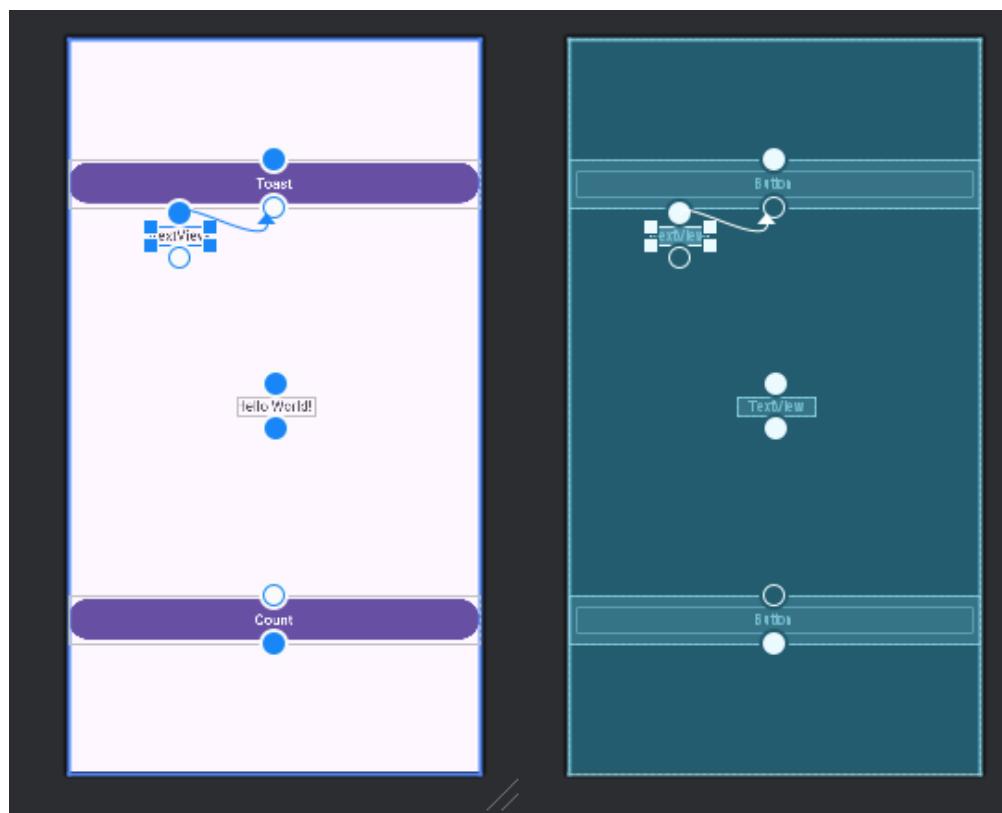
Một trong những lợi ích của ConstraintLayout là khả năng căn chỉnh hoặc ràng buộc các yếu tố tương đối với các yếu tố khác. Trong nhiệm vụ này, bạn sẽ thêm một TextView ở giữa bố cục, và ràng buộc nó theo chiều ngang vào các lề và theo chiều dọc vào hai yếu tố Button. Sau đó, bạn sẽ thay đổi các thuộc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

4.1 Thêm TextView và các ràng buộc

- Như được hiển thị trong hình động dưới đây, kéo một TextView từ khung Palette lên phần trên của bố cục, và kéo một ràng buộc từ đầu của TextView đến tay cầm ở dưới cùng của nút Toast. Điều này ràng buộc TextView nằm dưới nút.



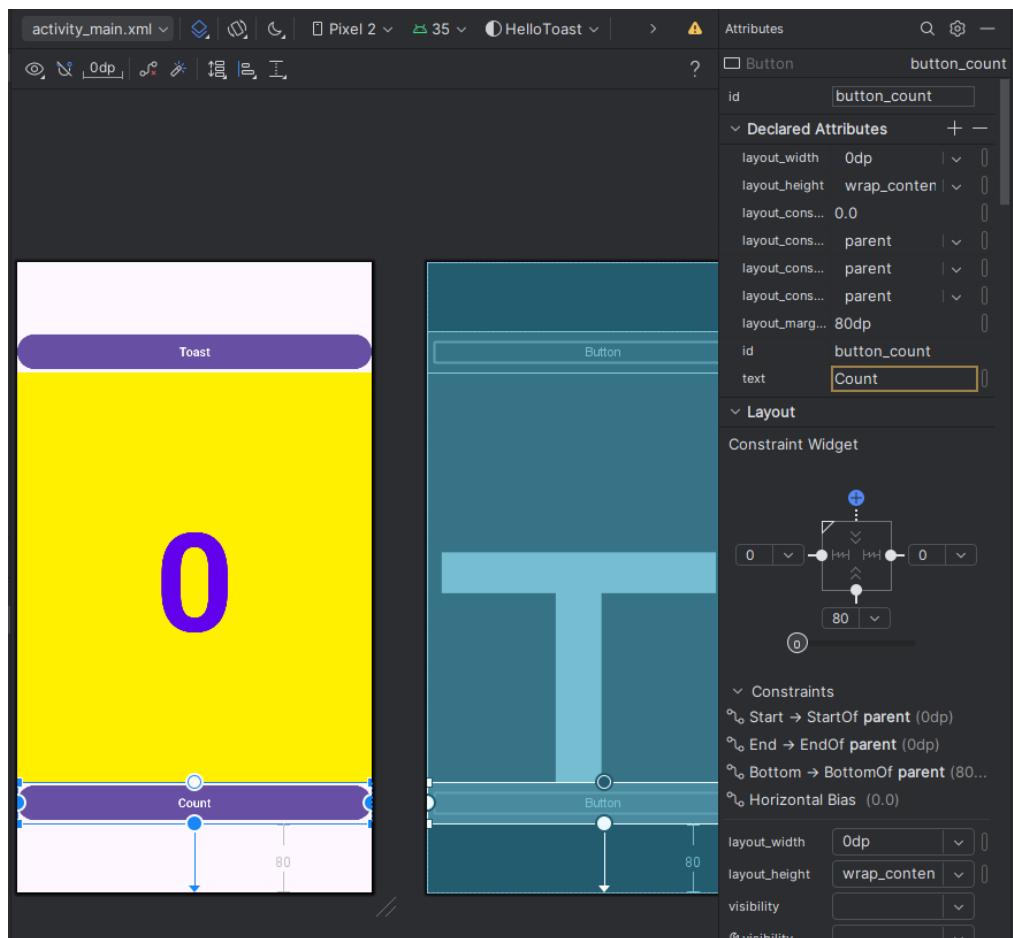
2. Như hiển thị trong hình động bên dưới, kéo một ràng buộc từ đáy của TextView đến tay cầm trên đỉnh của Count Button và từ các cạnh của TextView đến các cạnh của bối cảnh. Điều này ràng buộc TextView ở giữa bối cảnh giữa hai phần tử Button.



4.2 Đặt các thuộc tính cho TextView

Với TextView được chọn, mở khung Attributes nếu nó chưa được mở. Đặt các thuộc tính cho TextView như hiển thị trong hình động dưới đây. Các thuộc tính bạn chưa gắp phải được giải thích sau hình:

1. Đặt ID thành show_count.
2. Đặt văn bản thành 0.
3. Đặt kích thước văn bản thành 160sp.
4. Đặt kiểu văn bản thành B (in đậm) và căn chỉnh văn bản thành ALIGNCENTER (căn giữa đoạn văn).
5. Thay đổi các điều khiển kích thước view ngang và dọc (layout_width và layout_height) thành match_constraint.
6. Đặt màu văn bản thành @color/colorPrimary
7. Cuộn xuống khung và nhấp vào Xem tất cả các thuộc tính, cuộn xuống trang thứ hai của các thuộc tính đến background và sau đó nhập #FFF000 cho một sắc màu vàng.
8. Cuộn xuống đến gravity, mở rộng gravity và chọn center_ver (để căn giữa theo chiều dọc).

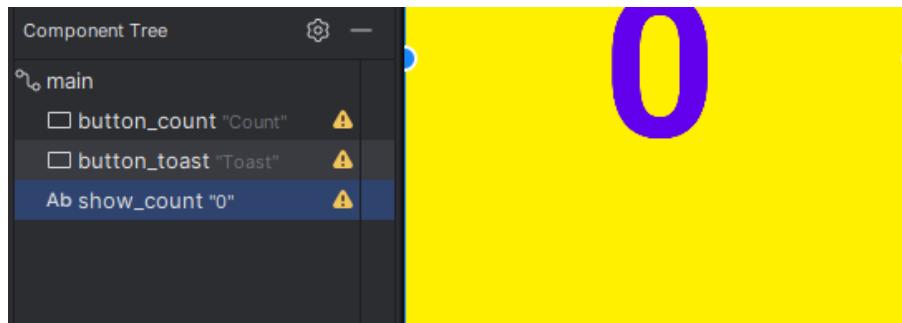


- **textSize**: Kích thước văn bản của TextView. Trong bài học này, kích thước được đặt là 160sp. sp là viết tắt của scale-independent pixel, và giống như dp, là một đơn vị thay đổi theo mật độ màn hình và sở thích kích thước văn bản của người dùng. Sử dụng đơn vị dp khi bạn chỉ định kích thước văn bản để các kích thước được điều chỉnh theo cả mật độ màn hình và sở thích của người dùng.
- **textStyle** và **textAlignment**: Kiểu văn bản được đặt là B (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 giữa TextView theo chiều dọc trong ConstraintLayout cha.

Bạn có thể nhận thấy rằng thuộc tính background nằm trên trang đầu tiên của khung Attributes cho một Button, nhưng trên trang thứ hai của khung Attributes cho một TextView. Khung Attributes thay đổi cho mỗi loại View: Các thuộc tính phổ biến nhất cho loại View sẽ xuất hiện trên trang đầu tiên và các thuộc tính còn lại sẽ được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của khung Attributes, nhấp vào biểu tượng ở thanh công cụ ở đầu khung.

Nhiệm vụ 5: *Chỉnh sửa bố cục trong XML*

Bố cục ứng dụng Hello Toast gần hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi yếu tố UI trong Component Tree. Di chuột qua các dấu chấm than này để xem các thông báo cảnh báo, như hiển thị dưới đây. Cùng một cảnh báo xuất hiện cho cả ba yếu tố: các chuỗi cố định nên sử dụng tài nguyên.



Cách dễ nhất để khắc phục các vấn đề 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 dễ thực hiện trực tiếp hơn trong mã nguồn XML.

5.1 Mở mã XML cho bố cục

Đối với nhiệm vụ này, mở tệp activity_main.xml nếu nó chưa được mở, và nhấp vào tab Text

Design **Text**

ở dưới cùng của trình chỉnh sửa bố cục.

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

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/button_count"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="80dp"
        android:text="Count"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent" />
    <Button
        android:id="@+id/button_toast"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="80dp"
        android:text="Toast"
        app:layout_constraintBottom_toTopOf="@+id/button_count"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent" />
    <TextView
        android:id="@+id/show_count"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:background="#FFFF00"
        android:gravity="center"
        android:text="0"
        android:textAlignment="center"
        android:textColor="@color/design_default_color_primary" />

```

5.2 Trích xuất tài nguyên chuỗi

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

1. Nhập một lần vào từ "Toast" (cảnh báo được đánh dấu đầu tiên).
2. Nhấn Alt-Enter trong Windows hoặc Option-Enter trong macOS và chọn Trích xuất tài nguyên chuỗi từ menu bật lên.
3. Nhập button_label_toast cho Tên tài nguyên.
4. Nhập vào OK. Một tài nguyên chuỗi được tạo trong tệp values/res/string.xml, và chuỗi trong mã của bạn được thay thế bằng một 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 khung Dự án > Android, mở rộng values trong res, và 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 một chuỗi khác để sử dụng trong nhiệm vụ tiếp theo hiển thị một 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. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa tài nguyên app_name.

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

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

6.1. Thêm thuộc tính onClick và trình xử lý cho mỗi nút

Một trình xử lý click là một phương thức được gọi khi người dùng nhấp hoặc nhấn vào một yếu tố UI có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick trong khung Attributes của tab Design. Bạn cũng có thể chỉ định tên của phương thức xử lý trong trình chỉnh sửa XML bằng cách thêm thuộc tính android:onClick vào nút. Bạn sẽ sử dụng phương pháp sau vì bạn chưa tạo các phương thức xử lý và trình chỉnh sửa XML cung cấp một cách tự động để tạo các phương thức đó.

1. chỉnh sửa XML mở (tab Text), tìm nút với android:id được đặt là button_toast.

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="80dp"
    android:text="@string/button_label_toast"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

- Thêm thuộc tính android:onClick vào cuối phần tử button_toast sau thuộc tính cuối cùng và trước chỉ số kết thúc />

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

- Nhấp vào biểu tượng bóng đèn đỏ xuất hiện bên cạnh thuộc tính. Chọn Create click handler, chọn MainActivity, và nhấp vào OK. Nếu biểu tượng bóng đèn đỏ không xuất hiện, nhấp vào tên phương thức ("showToast"). Nhấn Alt-Enter (Option-Enter trên Mac), chọn Create 'showToast(view)' in MainActivity, và nhấp vào OK. Hành động này tạo một phương thức trình giữ chỗ cho phương thức showToast() trong MainActivity, như hiển thị ở cuối các bước này.
- Lặp lại hai bước cuối với nút button_count: Thêm thuộc tính android:onClick vào cuối, và thêm trình xử lý click:

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

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

```
<Button  
    android:id="@+id/button_count"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="80dp"  
    android:text="@string/button_label_count"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.0"  
    app:layout_constraintStart_toStartOf="parent"  
    android:onClick="countUp"/>
```

```
<Button  
    android:id="@+id/button_toast"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="80dp"  
    android:text="@string/button_label_toast"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.0"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent"
```

```

        android:onClick="showToast" />

<TextView
    android:id="@+id/show_count"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="#FFF000"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAlignment="center"
    android:textColor="@color/design_default_color_primary"
    android:textSize="160sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/button_count"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button_toast"
    app:layout_constraintVertical_bias="0.5" />

```

5. Nếu MainActivity.java chưa được mở, hãy mở rộng java trong khung Dự án > Chế độ xem Android, mở rộng com.example.android.hellotoast, và sau đó nhấp đúp vào MainActivity. Trình chỉnh sửa mã xuất hiện với mã trong MainActivity:

```

package com.example.hellotoast;
import ...

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

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

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

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

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

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

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

3. Cung cấp ngữ cảnh của Activity ứng dụng. Vì một Toast hiển thị trên đầu giao diện người dùng 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 làm phím tắt.

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

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

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

5. Cung cấp thời lượng cho việc hiển thị. Ví dụ, Toast.LENGTH_SHORT hiển thị toast trong thời gian tương đối ngắn.

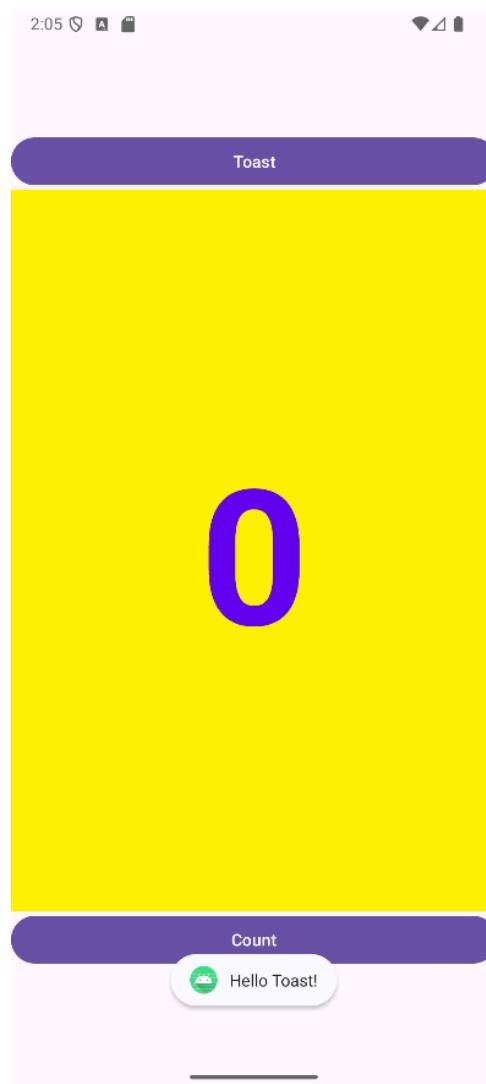
```
Toast toast = Toast.makeText(this, R.string.toast_message, Toast.LENGTH_SHORT);
```

Thời lượng hiển thị của một Toast có thể là Toast.LENGTH_LONG hoặc Toast.LENGTH_SHORT. Thời gian thực tế khoảng 3.5 giây cho Toast dài và 2 giây cho Toast ngắn.

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

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

Chạy ứng dụng và kiểm tra rằng thông báo Toast xuất hiện khi bạn nhấn vào nút Toast.



6.3. Chỉnh sửa trình xử lý nút Count

Bạn sẽ chỉnh sửa phương thức countUp() — trình xử lý click nút Count trong MainActivity — để hiển thị số đếm hiện tại sau khi nhấn nút Count. 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 cập nhật đến TextView để hiển thị.

Thực hiện các bước sau để chỉnh sửa trình xử lý click nút Count:

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

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

2. Để theo dõi số đếm, bạn cần một biến thành viên riêng. Mỗi lần nhấn nút Count sẽ tăng giá trị của biến này. Nhập mã sau, 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, chọn biểu thức mCount++. Biểu tượng bóng đèn đỏ cuối cùng sẽ xuất hiện.

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn Create field 'mCount' từ menu bật lên. Điều này tạo ra một biến thành viên riêng ở đầu MainActivity, và Android Studio giả định rằng bạn muốn nó là một số nguyên (int):

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

4. Thay đổi câu lệnh biến thành viên riêng để khởi tạo biến thành không:

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

5. Cùng với biến trên, bạn cũng cần một biến thành viên riêng cho tham chiếu của show_count TextView, mà bạn sẽ thêm vào trình xử lý click. Gọi biến này là mShowCount.

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

6. Bây giờ bạn đã có mShowCount, bạn có thể lấy một tham chiếu đến TextView bằng cách sử dụng ID mà bạn đã đặt trong tệp bố cục. Để lấy tham chiếu này chỉ một lần, hãy chỉ định nó trong phương thức onCreate(). Như bạn đã học trong một bài học khác, phương thức onCreate() được sử dụng để khởi tạo bố cục, có nghĩa là đặt nội dung của màn hình vào bố cục XML. Bạn cũng có thể sử dụng nó để lấy các tham chiếu đến các yếu tố UI khác trong bố cục, chẳng hạn như TextView. Tìm phương thức onCreate() trong MainActivity:

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

7. Thêm câu lệnh findViewById vào cuối phương thức:

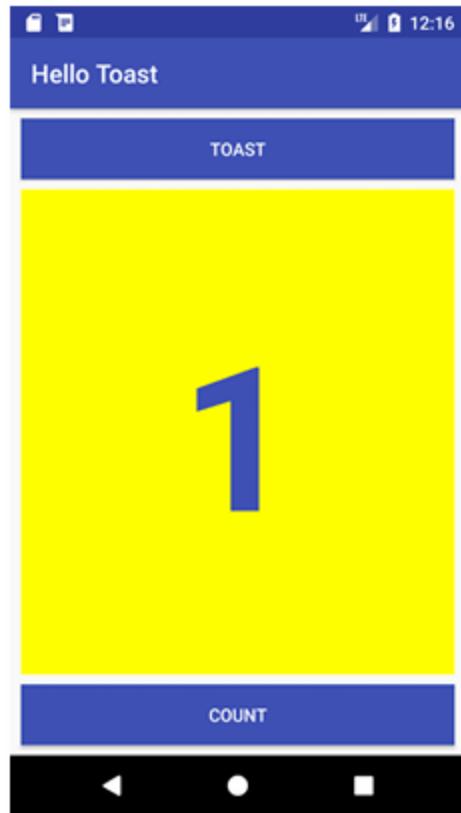
```
@Override protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);  
mShowCount = (TextView) findViewById(R.id.show_count); }
```

Một View, giống như một chuỗi, là một tài nguyên có thể có ID. Cuộc gọi findViewById lấy ID của một view làm tham số của nó và trả về View. Vì phương thức này trả về một View, bạn phải chuyển đổi kết quả thành kiểu view mà bạn mong đợi, trong trường hợp này là (TextView).

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

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

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



Mẹo: Để có hướng dẫn chuyên sâu về việc sử dụng ConstraintLayout, hãy xem Codelab Sử dụng ConstraintLayout để thiết kế các view của bạn.

Thử thách mã hóa

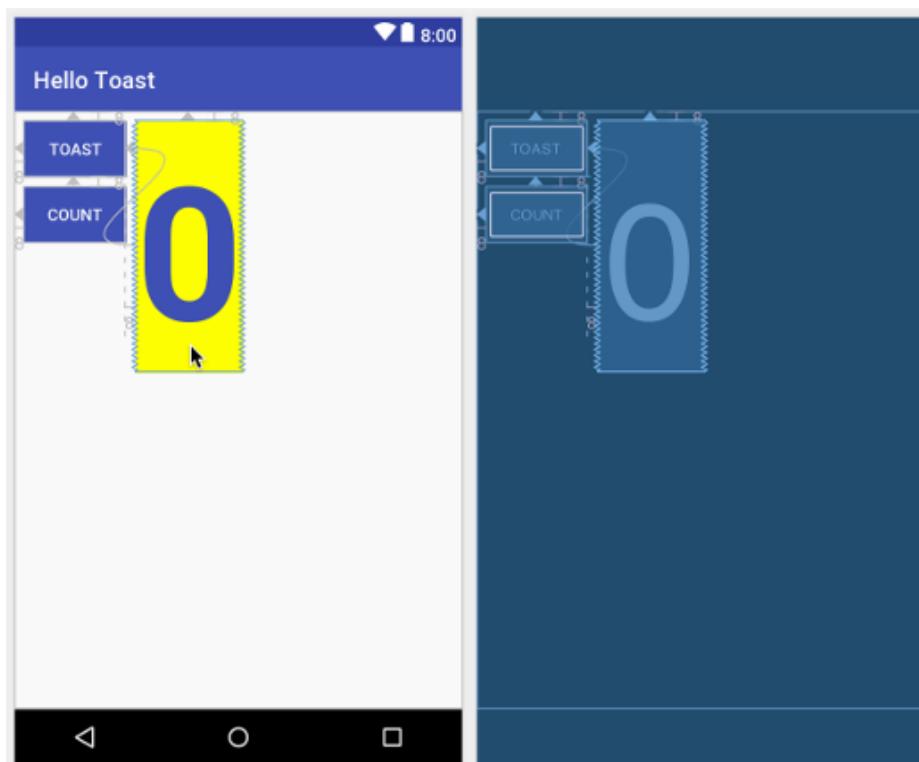
Lưu ý: Tất cả các thử thách mã hóa đều là tùy chọn và không phải là điều kiện tiên quyết cho các bài học sau.

Ứng dụng HelloToast trông ổn khi thiết bị hoặc trình giả lập được định hướng theo chiều dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang định hướng ngang, nút Count có thể chồng lên TextView ở dưới cùng như trong hình dưới đây.



Thử thách: Thay đổi bố cục sao cho trông đẹp ở cả hai hướng ngang và dọc:

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





Tóm tắt

View, ViewGroup và bối cảnh:

- Tất cả các yếu tố UI đều là các lớp con của lớp View và do đó thừa kế nhiều thuộc tính của lớp View.
- Các yếu tố View có thể được nhóm lại bên trong một ViewGroup, hoạt động như một hộp chứa. Mỗi quan hệ này là mối quan hệ cha-con, trong đó cha là ViewGroup và con là View hoặc một ViewGroup khác.
- Phương thức `onCreate()` được sử dụng để khởi tạo bối cảnh, có nghĩa là đặt nội dung màn hình vào bối cảnh XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các yếu tố UI khác trong bối cảnh.
- Một View, giống như một chuỗi, là một tài nguyên có thể có ID. Lệnh gọi `findViewById` lấy ID của một view làm tham số của nó và trả về View.

Sử dụng trình chỉnh sửa bối cảnh:

- Nhập vào tab Design để thao tác các yếu tố và bối cảnh, và tab Text để chỉnh sửa mã XML cho bối cảnh.
- Trong tab Design, khung Palettes hiển thị các yếu tố UI mà bạn có thể sử dụng trong bối cảnh ứng dụng của bạn, và khung Component tree hiển thị cấu trúc phân cấp của các yếu tố UI.
- Các khung thiết kế và bản vẽ của trình chỉnh sửa bối cảnh hiển thị các yếu tố UI trong bối cảnh.
- Tab Attributes hiển thị khung Attributes để thiết lập thuộc tính cho một yếu tố UI.

- Tay cầm ràng buộc: Nhấp vào một tay cầm ràng buộc, hiển thị dưới dạng một vòng tròn ở mỗi bên của một yếu tố, sau đó kéo đến một tay cầm ràng buộc khác hoặc đến ranh giới cha để tạo ràng buộc. Ràng buộc được biểu diễn bằng đường zigzag.
- Tay cầm thay đổi kích thước: Bạn có thể kéo tay cầm thay đổi kích thước hình vuông để thay đổi kích thước của yếu tố. Khi kéo, tay cầm thay đổi thành một góc xiên.
- Khi được bật, công cụ Autoconnect tự động tạo hai hoặc nhiều ràng buộc cho một yếu tố UI vào bố cục cha. Sau khi bạn kéo yếu tố vào bố cục, nó tạo các ràng buộc dựa trên vị trí của yếu tố.
- Bạn có thể xóa các ràng buộc từ một yếu tố bằng cách chọn yếu tố đó và di con trỏ qua nó để hiển thị nút Clear Constraints. Nhấp vào nút này để xóa tất cả các ràng buộc trên yếu tố đã chọn. Để xóa một ràng buộc đơn, nhấp vào tay cầm cụ thể đặt ràng buộc đó.
- Khung Attributes cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một yếu tố UI. Nó cũng bao gồm một bảng Kích thước hình vuông gọi là view inspector ở trên cùng. Các biểu tượng bên trong hình vuông đại diện cho các cài đặt chiều cao và chiều rộng.

Thiết lập chiều rộng và chiều cao bố cục:

- Các thuộc tính layout_width và layout_height thay đổi khi bạn thay đổi các điều khiển kích thước chiều cao và chiều rộng trong view inspector. Các thuộc tính này có thể nhận một trong ba giá trị cho một ConstraintLayout:
 - Cài đặt match_constraint mở rộng view để lấp đầy cha của nó bằng chiều rộng hoặc chiều cao – đến một biên nếu được đặt.
 - Cài đặt wrap_content thu nhỏ Kích thước view để view vừa đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, view sẽ trở nên vô hình.
 - Sử dụng số cố định dp (density-independent pixels) để chỉ định kích thước cố định, điều chỉnh cho kích thước màn hình của thiết bị.

Trích xuất tài nguyên chuỗi:

- Thay vì mã hóa chuỗi, tốt nhất là sử dụng tài nguyên chuỗi, là đại diện cho các chuỗi đó. Thực hiện theo các bước sau:
 - Nhấp một lần vào chuỗi được mã hóa cứng để trích xuất, nhấn Alt-Enter (Option-Enter trên Mac), và chọn Extract string resources từ menu bật lên.
 - Đặt tên tài nguyên.
 - Nhấp OK. Điều này tạo một tài nguyên chuỗi trong tệp values/res/string.xml, và chuỗi trong mã của bạn được thay thế bằng một tham chiếu đến tài nguyên: @string/button_label_toast

Xử lý các lượt nhấp:

- Một phương thức xử lý click được gọi khi người dùng nhấp hoặc nhấn vào một yếu tố UI.

- Chỉ định một phương thức xử lý click cho một yếu tố UI như Button bằng cách nhập tên của nó trong trường onClick ở khung Attributes của tab Design, hoặc trong trình chỉnh sửa XML bằng cách thêm thuộc tính android:onClick vào một yếu tố UI như Button.
- Tạo phương thức xử lý click trong Activity chính bằng cách sử dụng tham số View. Ví dụ: public void showToast(View view) {...}.
- Bạn có thể tìm thông tin về tất cả các thuộc tính Button trong tài liệu lớp Button, và tất cả các thuộc tính TextView trong tài liệu lớp TextView.

Hiển thị thông báo Toast:

- Một Toast cung cấp một cách để hiển thị một thông báo đơn giản trong một cửa sổ pop-up nhỏ. Nó chỉ chiếm không gian cần thiết cho thông điệp.
- Để tạo một thẻ hiện của Toast, thực hiện theo các bước sau:
- Gọi phương thức makeText() trên lớp Toast.
- Cung cấp ngữ cảnh của Activity ứng dụng và thông báo để hiển thị (chẳng hạn như tài nguyên chuỗi).
- Cung cấp thời lượng hiển thị, ví dụ, Toast.LENGTH_SHORT cho một khoảng thời gian ngắn. Thời lượng có thể là Toast.LENGTH_LONG hoặc Toast.LENGTH_SHORT.
- Hiển thị Toast bằng cách gọi show().

1.2 Phần B: Trình chỉnh sửa bố cục

Giới thiệu Như bạn đã học trong Bài 1.2 Phần A: Giao diện người dùng tương tác đầu tiên của bạn, bạn có thể tạo giao diện người dùng (UI) sử dụng ConstraintLayout trong trình chỉnh sửa bố cục, đặt các yếu tố UI vào một bố cục bằng cách sử dụng các kết nối ràng buộc đến các yếu tố khác và cạnh của bố cục. ConstraintLayout được thiết kế để dễ dàng kéo các yếu tố UI vào trình chỉnh sửa bố cục.

ConstraintLayout là một ViewGroup, là một View đặc biệt có thể chứa các đối tượng View khác (được gọi là children hoặc child views). Thực hành này cho thấy nhiều tính năng hơn của ConstraintLayout và trình chỉnh sửa bố cục. Thực hành này cũng giới thiệu hai lớp con khác của ViewGroup:

- **LinearLayout:** Một nhóm căn chỉnh các yếu tố View con bên trong nó theo chiều ngang hoặc dọc.
- **RelativeLayout:** Một nhóm các yếu tố View con trong đó mỗi yếu tố View được đặt và căn chỉnh tương đối với các yếu tố View khác trong ViewGroup. Các vị trí của các yếu tố View con được mô tả liên quan đến nhau hoặc đến ViewGroup cha.

Những gì bạn nên biết trước:

- Tạo một ứng dụng Hello World với Android Studio.
- Chạy một ứng dụng trên trình giả lập hoặc thiết bị.

- Tạo một bố cục đơn giản cho một ứng dụng với ConstraintLayout.
- Trích xuất và sử dụng tài nguyên chuỗi.

Những gì bạn sẽ học:

- Cách tạo một biến thể bố cục cho hướng ngang (landscape).
- Cách tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Cách sử dụng ràng buộc cơ sở để căn chỉnh các yếu tố UI với văn bản.
- Cách sử dụng các nút pack và align để căn chỉnh các yếu tố trong bố cục.
- Cách đặt các view trong một LinearLayout.
- Cách đặt các view trong một RelativeLayout.

Những gì bạn sẽ làm:

- Tạo một biến thể bố cục cho hướng hiển thị ngang.
- Tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Sửa đổi bố cục để thêm các ràng buộc cho các yếu tố UI.
- Sử dụng các ràng buộc cơ sở của ConstraintLayout để căn chỉnh các yếu tố với văn bản.
- Sử dụng các nút pack và align của ConstraintLayout để căn chỉnh các yếu tố.
- Thay đổi bố cục để sử dụng LinearLayout.
- Đặt các yếu tố trong một LinearLayout.
- Thay đổi bố cục để sử dụng RelativeLayout.
- Sắp xếp lại các view trong bố cục chính để tương đối với nhau.

Tổng quan về ứng dụng:

Ứng dụng Hello Toast trong bài học trước sử dụng ConstraintLayout để sắp xếp các yếu tố UI trong bố cục Activity, như được hiển thị trong hình dưới đây.

