

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	61
1.3)	Trình chỉnh sửa bố cục	61
1.4)	Văn bản và các chế độ cuộn	61
1.5)	Tài nguyên có sẵn	61
Bài 2)	Activities	61
2.1)	Activity và Intent	61
2.2)	Vòng đời của Activity và trạng thái	61
2.3)	Intent ngầm định.....	61
Bài 3)	Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ.....	61
3.1)	Trình gỡ lỗi.....	61
3.2)	Kiểm thử đơn vị.....	61
3.3)	Thư viện hỗ trợ.....	61
CHƯƠNG 2.	Trải nghiệm người dùng.....	62
Bài 1)	Tương tác người dùng.....	62
1.1)	Hình ảnh có thể chọn	62
1.2)	Các điều khiển nhập liệu	62
1.3)	Menu và bộ chọn	62
1.4)	Điều hướng người dùng.....	62
1.5)	RecyclerView.....	62
Bài 2)	Trải nghiệm người dùng thú vị	62
2.1)	Hình vẽ, định kiểu và chủ đề	62
2.2)	Thẻ và màu sắc.....	62

2.3)	Bố cục thích ứng.....	62
Bài 3)	Kiểm thử giao diện người dùng	62
3.1)	Espresso cho việc kiểm tra UI.....	62
CHƯƠNG 3.	Làm việc trong nền	62
Bài 1)	Các tác vụ nền	62
1.1)	AsyncTask	62
1.2)	AsyncTask và AsyncTaskLoader	62
1.3)	Broadcast receivers	62
Bài 2)	Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	62
2.1)	Thông báo.....	62
2.2)	Trình quản lý cảnh báo	62
2.3)	JobScheduler	62
CHƯƠNG 4.	Lưu dữ liệu người dùng	63
Bài 1)	Tùy chọn và cài đặt.....	63
1.1)	Shared preferences	63
1.2)	Cài đặt ứng dụng	63
Bài 2)	Lưu trữ dữ liệu với Room	63
2.1)	Room, LiveData và ViewModel	63
2.2)	Room, LiveData và ViewModel	63
3.1)	Trình gowx loi	

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

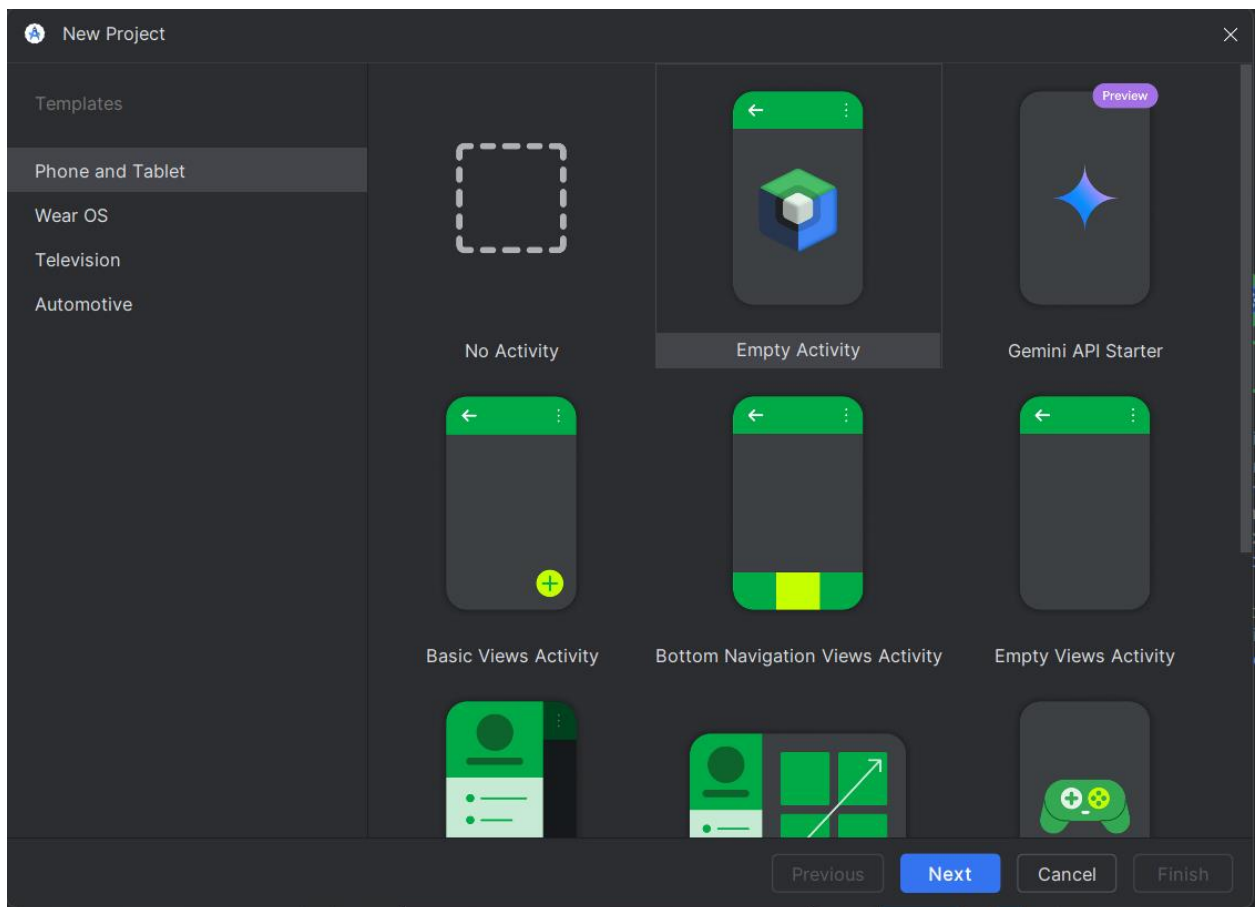
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

- Một máy tính chạy Windows hoặc Linux, hoặc một Mac chạy macOS. Xem trang tải xuống Android Studio để biết yêu cầu hệ thống cập nhật.

- Truy cập Internet hoặc một phương pháp thay thế để tải các cài đặt mới nhất của Android Studio và Java lên máy tính của bạn.

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

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

Đây là hình ảnh ứng dụng đã hoàn thành sẽ trông như:

Nhiệm vụ 1: Cài đặt Android Studio

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

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

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

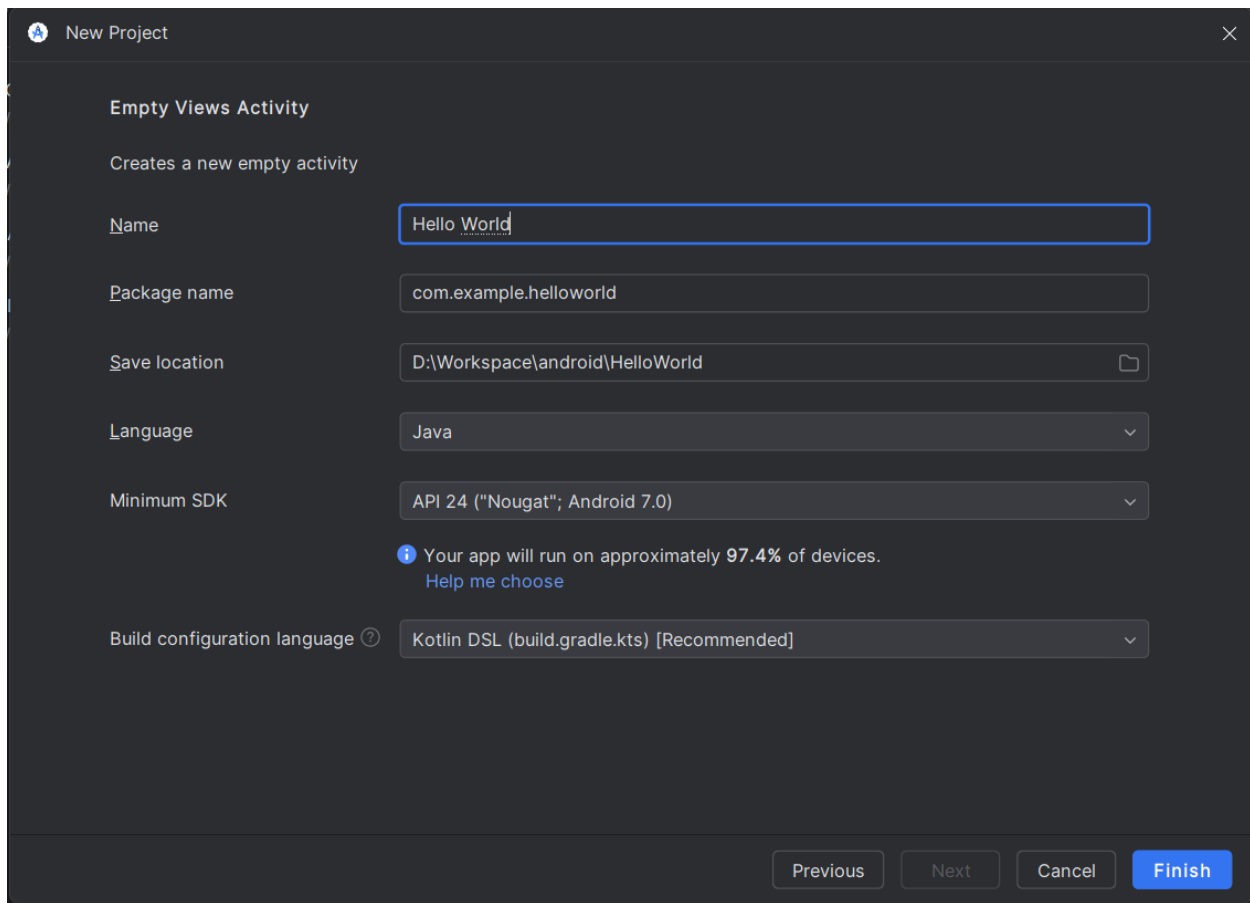
Xử lý sự cố: Nếu bạn gặp sự cố với cài đặt của mình, hãy kiểm tra ghi chú phát hành Android Studio hoặc nhận trợ giúp từ người hướng dẫn của bạn.

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 chưa mở.
2. Trong cửa sổ chính **Welcome to Android Studio**, nhấp chuột vào **Start a new Android Studio project**.
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** cho **Application name**.

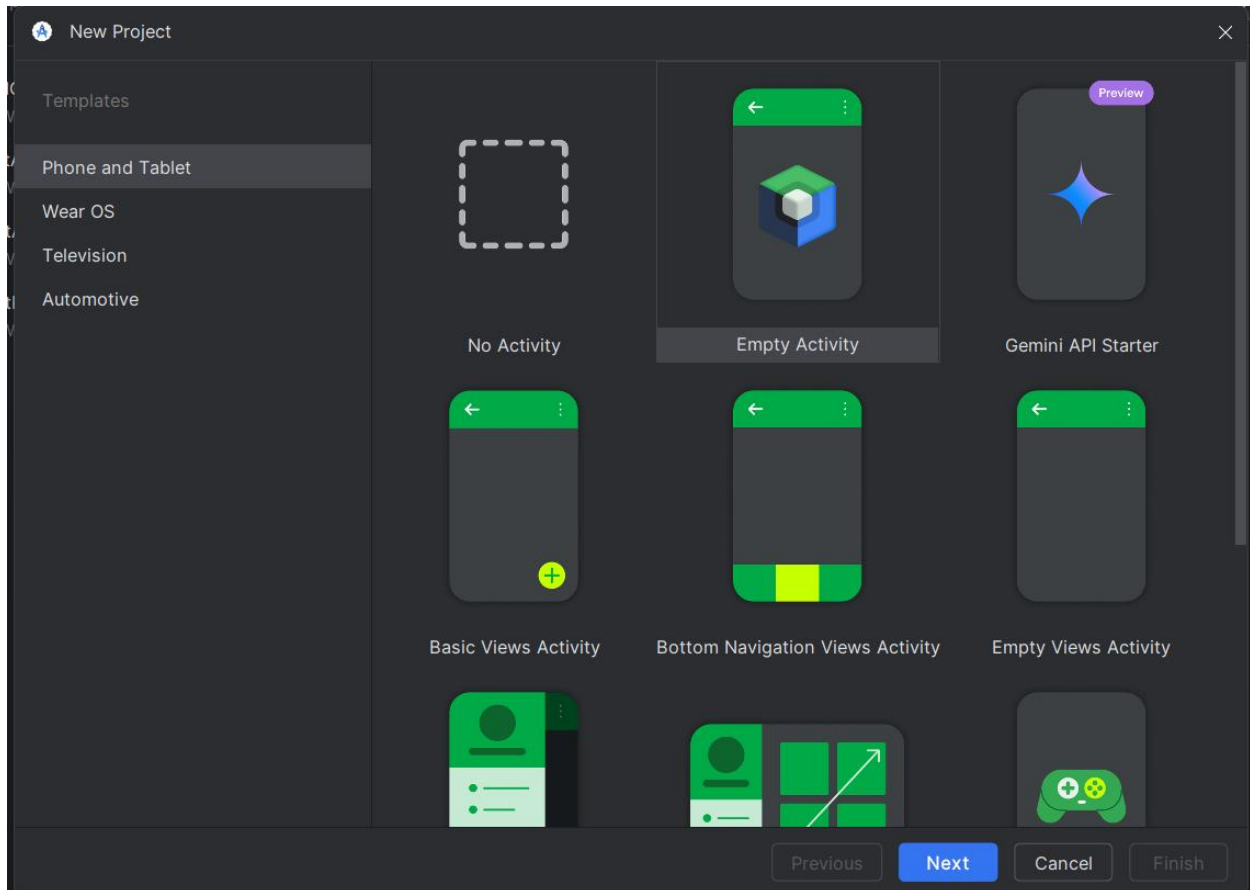


4. Xác minh rằng vị trí Dự án mặc định là nơi bạn muốn lưu trữ ứng dụng Hello World của bạn và các dự án Android Studio khác hoặc thay đổi thành thư mục bạn muốn.
5. Chấp nhận **android.example.com** mặc định cho **Company Domain** hoặc tạo một Company domain duy nhất. Nếu bạn không có kế hoạch phát hành ứng dụng của bạn, bạn có thể chấp nhận mặc định. Lưu ý rằng việc thay đổi tên gói ứng dụng của bạn sau này sẽ tốn thêm công việc.
6. Bỏ chọn các tùy chọn **Include C++ support** và Include Kotlin support, rồi nhấp chuột vào Next.
7. Trên màn hình Thiết bị Android mục tiêu, Điện thoại và Máy tính bảng phải được chọn. Đảm bảo rằng API 15: Android 4.0.3 IceCreamSandwich được đặt làm SDK tối thiểu; nếu không, hãy sử dụng menu bật lên để đặt.

Đây là các thiết lập được sử dụng bởi các ví dụ trong các bài học của khóa học này. Tính đến thời điểm viết bài này, các thiết lập này giúp ứng dụng Hello World của bạn

tương thích với 97% thiết bị Android đang hoạt động trên cửa hàng Google Play.

8. Bỏ chọn **Include Instant App support** và tất cả các tùy chọn khác. Sau đó, nhấp chuột vào **Next**. Nếu dự án của bạn yêu cầu các thành phần bổ sung cho SDK mục tiêu đã chọn, Android Studio sẽ tự động cài đặt chúng.
9. Cửa sổ **Add an Activity** xuất hiện. Hoạt động là một việc duy nhất, tập trung mà người dùng có thể thực hiện. Đây là thành phần quan trọng của bất kỳ ứng dụng Android nào. Hoạt động thường có bố cục liên quan đến nó để xác định cách các thành phần UI xuất hiện trên màn hình. Android Studio cung cấp các mẫu Hoạt động để giúp bạn bắt đầu. Đối với dự án Hello World, chọn **Empty Activity** như được hiển thị bên dưới và nhấp chuột vào “**Next**”.



10. Màn hình **Configure Activity** xuất hiện (khác nhau tùy thuộc vào mẫu bạn đã chọn ở bước trước). Theo mặc định, Activity trống do mẫu cung cấp

được đặt tên là MainActivity. Bạn có thể thay đổi tên này nếu muốn, nhưng bài học này sử dụng MainActivity.

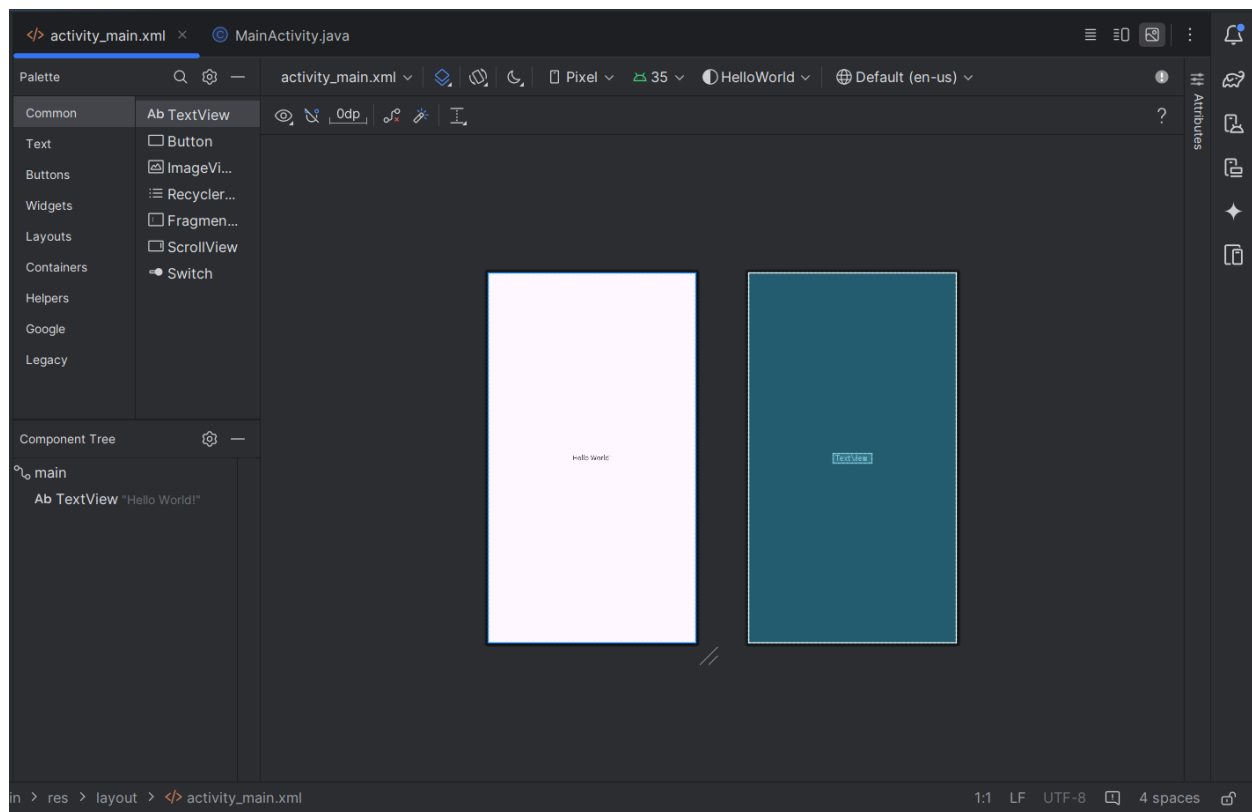
11. Đảm bảo rằng tùy chọn **Generate Layout file** được chọn. Tên bố cục theo mặc định là activity_main. Bạn có thể thay đổi tùy chọn này nếu muốn, nhưng bài học này sử dụng activity_main.
12. Đảm bảo rằng tùy chọn **Backwards Compatibility (App Compat)** được chọn. Điều này đảm bảo rằng ứng dụng của bạn sẽ tương thích ngược với các phiên bản Android trước đó.
13. Nhấp chuột vào **Finish**.

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

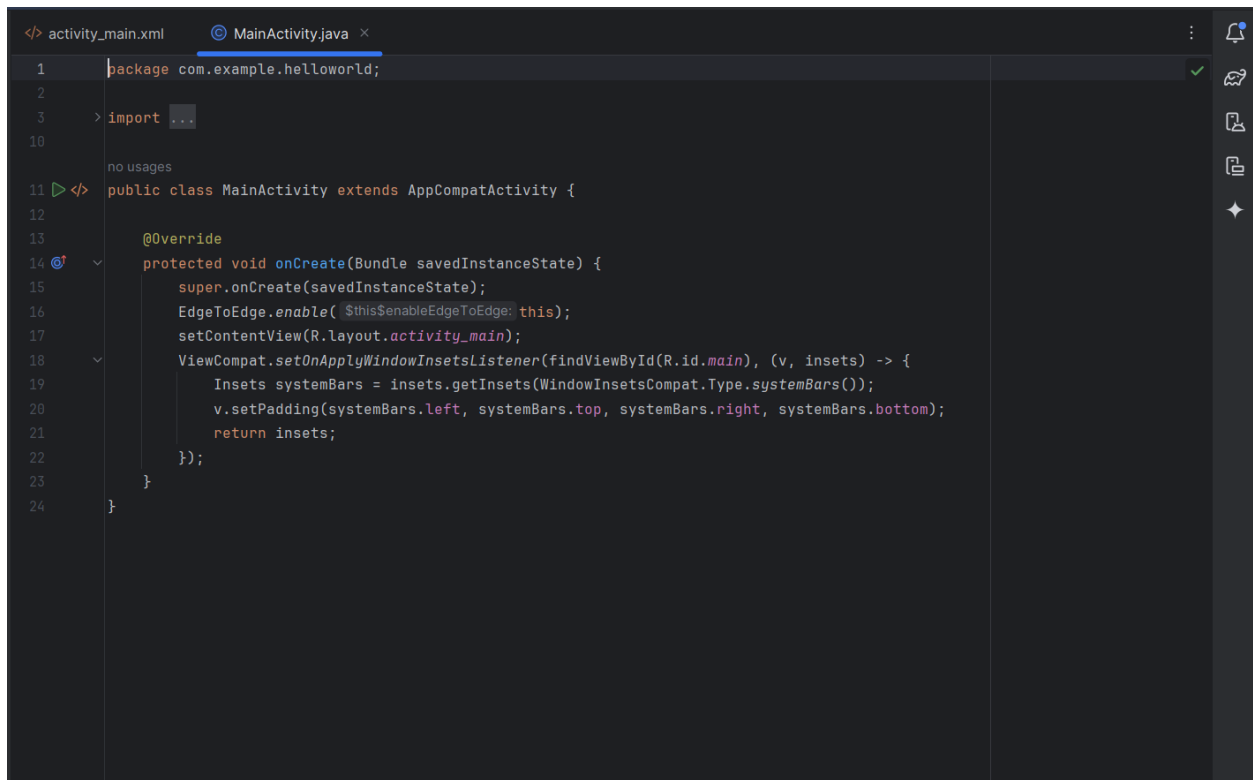
Mẹo: Xem trang Cấu hình nhà phát triển bản dựng của bạn để biết thông tin chi tiết. Bạn cũng có thể thấy thông báo "Mẹo trong ngày" với các phím tắt và các mẹo hữu ích khác. Nhấp vào Close để đóng thông báo.

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

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



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

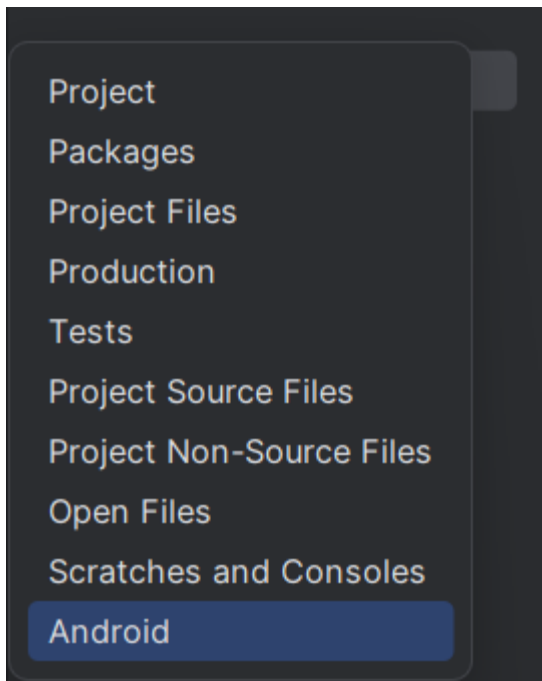


```
1 package com.example.helloworld;
2
3 > import ...
10
11 no usages
12
13 public class MainActivity extends AppCompatActivity {
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         EdgeToEdge.enable(this);
18         setContentView(R.layout.activity_main);
19         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
20             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
21             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
22             return insets;
23         });
24     }
25 }
```

2.2 Tìm hiểu dự án > Bảng điều khiển Android

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

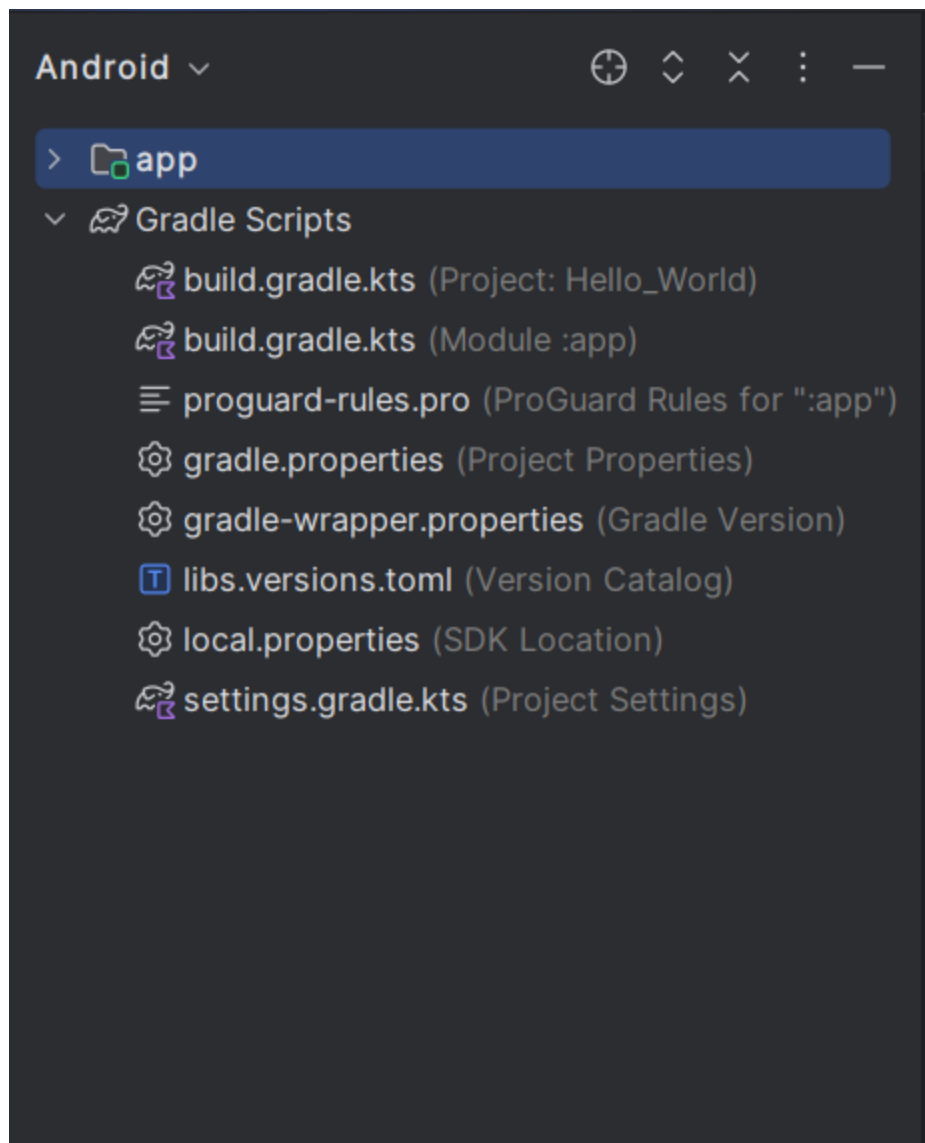
1. Nếu chưa chọn, hãy nhấp vào tab **Project** trong cột tab dọc ở phía bên trái của cửa sổ Android Studio. Ngăn Project sẽ xuất hiện.
2. Để xem dự án trong hệ thống phân cấp dự án Android chuẩn, hãy chọn Android từ menu bật lên ở đầu ngăn Project, như hiển thị bên dưới.



2.3 Tìm hiểu thư mục Gradle Scripts

Gradle xây dựng hệ thống 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 tiên 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 tìm hiểu 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. Tìm tệp build.gradle(Project: HelloWorld).

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

```
You can use the Project Structure dialog to view and edit your project configuration Open (Ctrl+Alt+Shift+S) Hide notification
1  pluginManagement {
2      repositories {
3          google {
4              content {
5                  includeGroupByRegex( groupRegex: "com\\.android\\.*)"
6                  includeGroupByRegex( groupRegex: "com\\.google\\.*)"
7                  includeGroupByRegex( groupRegex: "androidx\\.*)"
8              }
9          }
10         mavenCentral()
11         gradlePluginPortal()
12     }
13 }
14 dependencyResolutionManagement {
15     repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
16     repositories {
17         google()
18         mavenCentral()
19     }
20 }
21
22 rootProject.name = "Hello World"
23 include( ...projectPaths: ":app")
24
```

Trong những phiên bản Android Studio mới nhất thì JCenter không còn được hỗ trợ

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

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

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

phụ thuộc của tất cả các tập “.jar” bên trong thư mục libs.

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

```
1 plugins {
2     alias(libs.plugins.android.application)
3 }
4
5 android {
6     namespace = "com.example.helloworld"
7     compileSdk = 35
8
9     defaultConfig {
10         applicationId = "com.example.helloworld"
11         minSdk = 24
12         targetSdk = 35
13         versionCode = 1
14         versionName = "1.0"
15
16         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
17     }
18
19     buildTypes {
20         release {
21             isMinifyEnabled = false
22             proguardFiles(
23                 getDefaultProguardFile("proguard-android-optimize.txt"),
24                 "proguard-rules.pro"
25             )
26         }
27     }
28
29     compileOptions {
30         sourceCompatibility = JavaVersion.VERSION_11
31         targetCompatibility = JavaVersion.VERSION_11
32     }
33 }
```

```

27     }
28     compileOptions {
29         sourceCompatibility = JavaVersion.VERSION_11
30         targetCompatibility = JavaVersion.VERSION_11
31     }
32 }
33
34 dependencies {
35
36     implementation(libs.appcompat)
37     implementation(libs.material)
38     implementation(libs.activity)
39     implementation(libs.constraintlayout)
40     testImplementation(libs.junit)
41     androidTestImplementation(libs.ext.junit)
42     androidTestImplementation(libs.espresso.core)
43 }

```

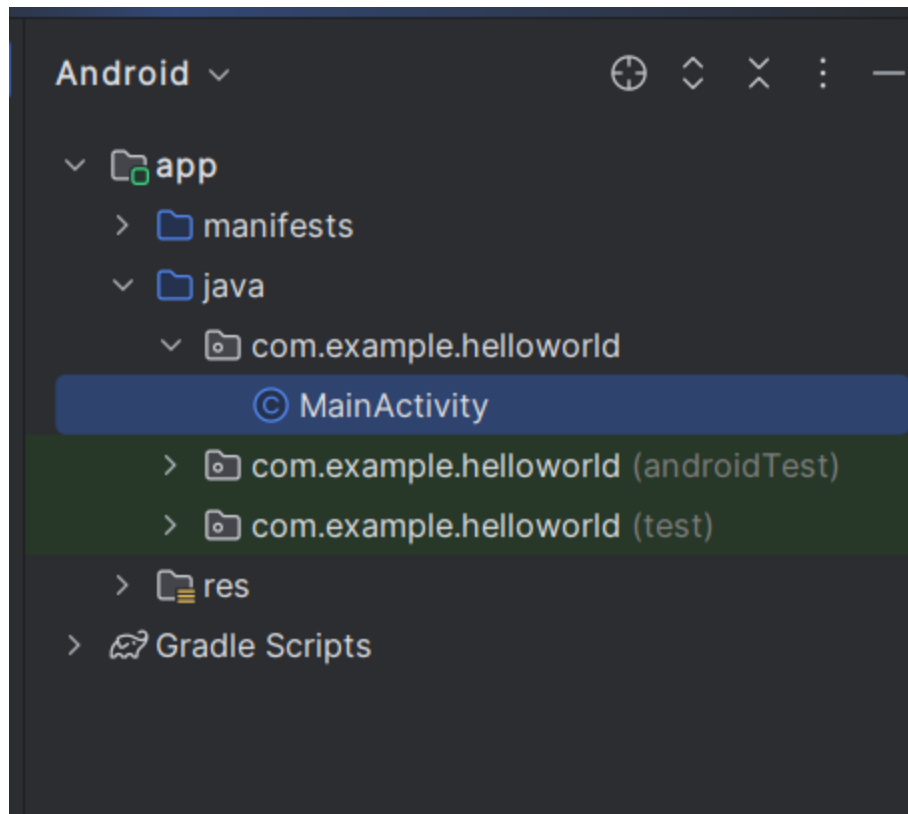
1:1 LF UTF-8 4 spaces

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

2.4 Khám phá các thư mục ứng dụng và res

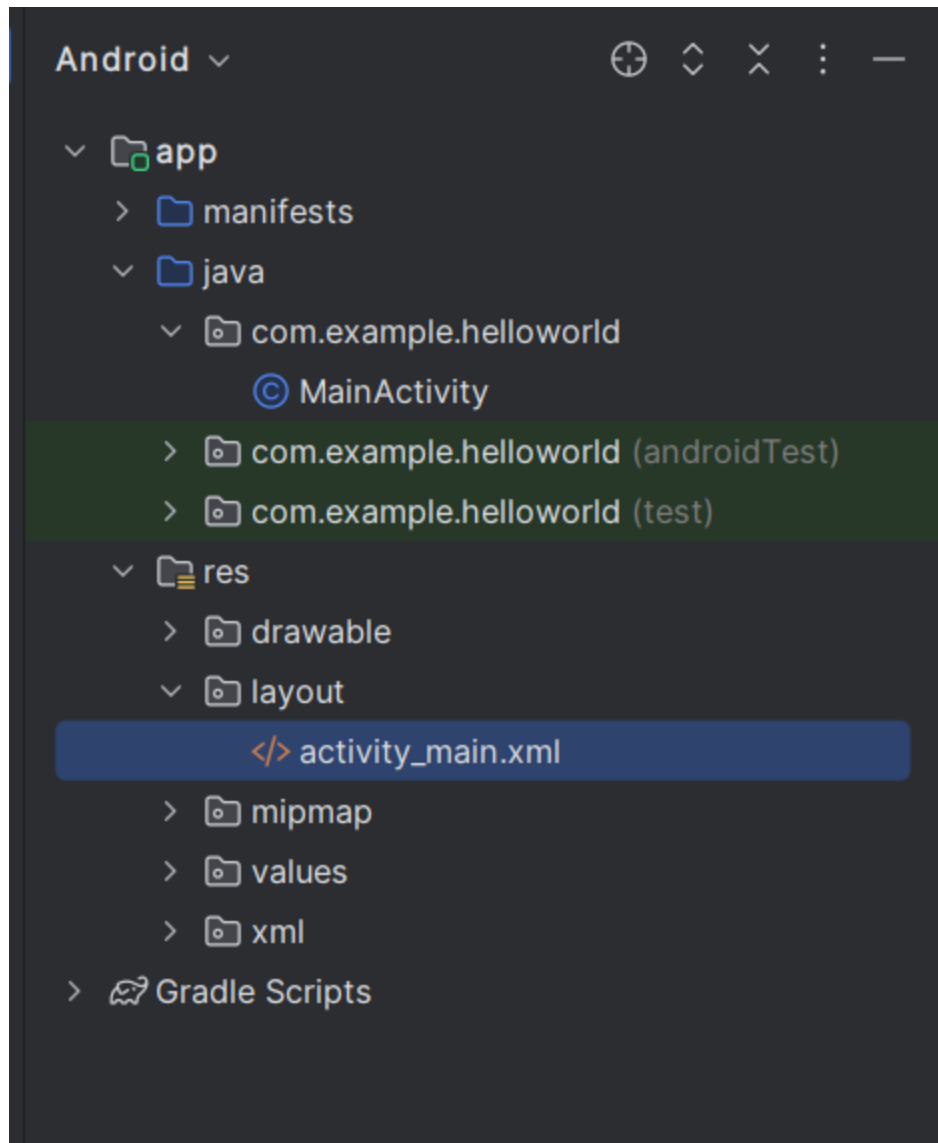
Tất cả mã và tài nguyên cho ứng dụng đều nằm trong thư mục app và res.

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



Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như được hiển thị 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 khác được sử dụng để thử nghiệm và được mô tả trong một bài học khác. Đối với ứng dụng Hello World, chỉ có một gói và nó chứa **MainActivity.java**. Tên của Hoạt động đầu tiên (màn hình) mà người dùng nhìn thấy, cũng khởi tạo các tài nguyên trên toàn ứng dụng, thường được gọi là **MainActivity** (phần mở rộng tệp bị bỏ qua trong ngăn Dự án > Android).

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



Thư mục **res** chứa các tài nguyên, chẳng hạn như 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 Tìm hiểu thư mục manifests

Thư mục manifests chứa các tệp cung cấp thông tin cần thiết về ứng dụng của bạn cho

hệ thống Android, mà hệ thống phải có trước khi có thể chạy bất kỳ mã nào của ứng dụng.

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

Tệp AndroidManifest.xml mô tả tất cả các thành phần của ứng dụng Android của bạn. Tất cả các thành phần cho một ứng dụng, chẳng hạn như mỗi Activity, phải được khai báo trong tệp XML này. Trong các bài học khác của khóa học, bạn sẽ sửa đổi tệp này để thêm các tính năng và quyền tính năng. Để biết phần giới thiệu, hãy xem [App Manifest Overview](#).

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

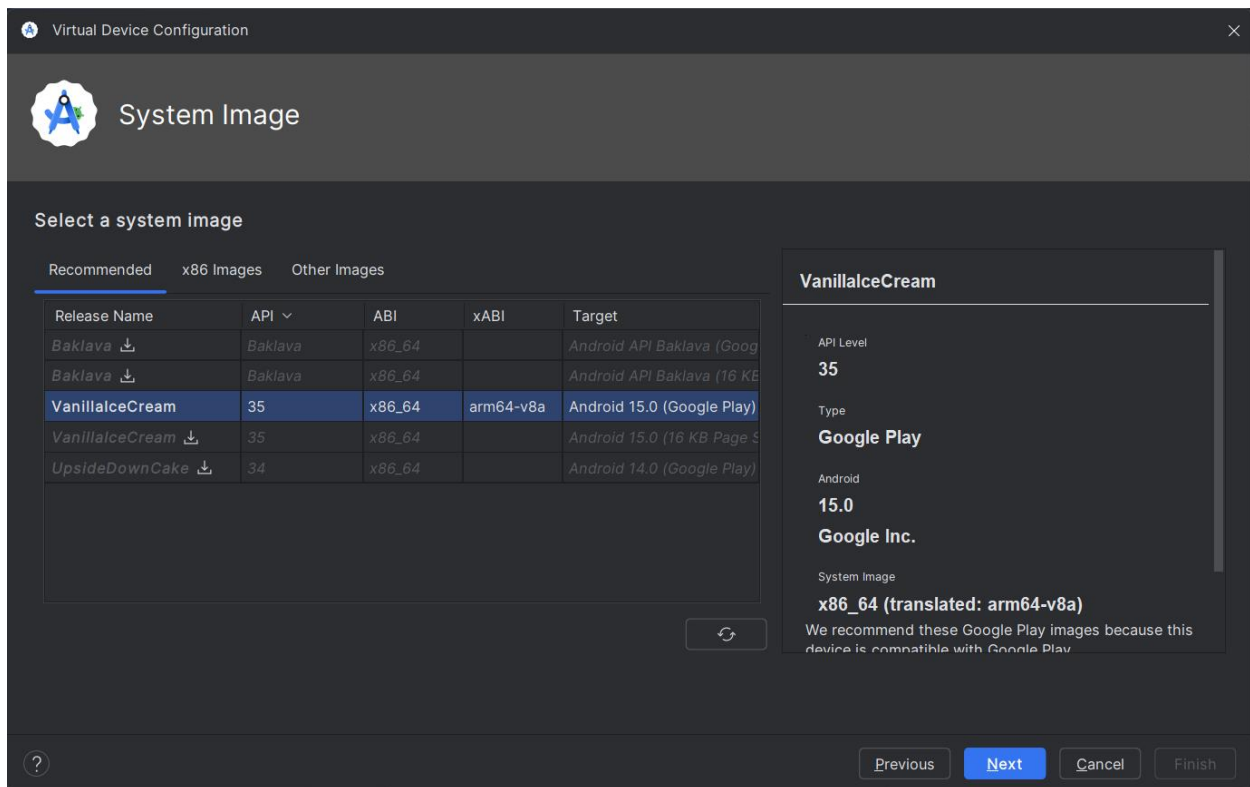
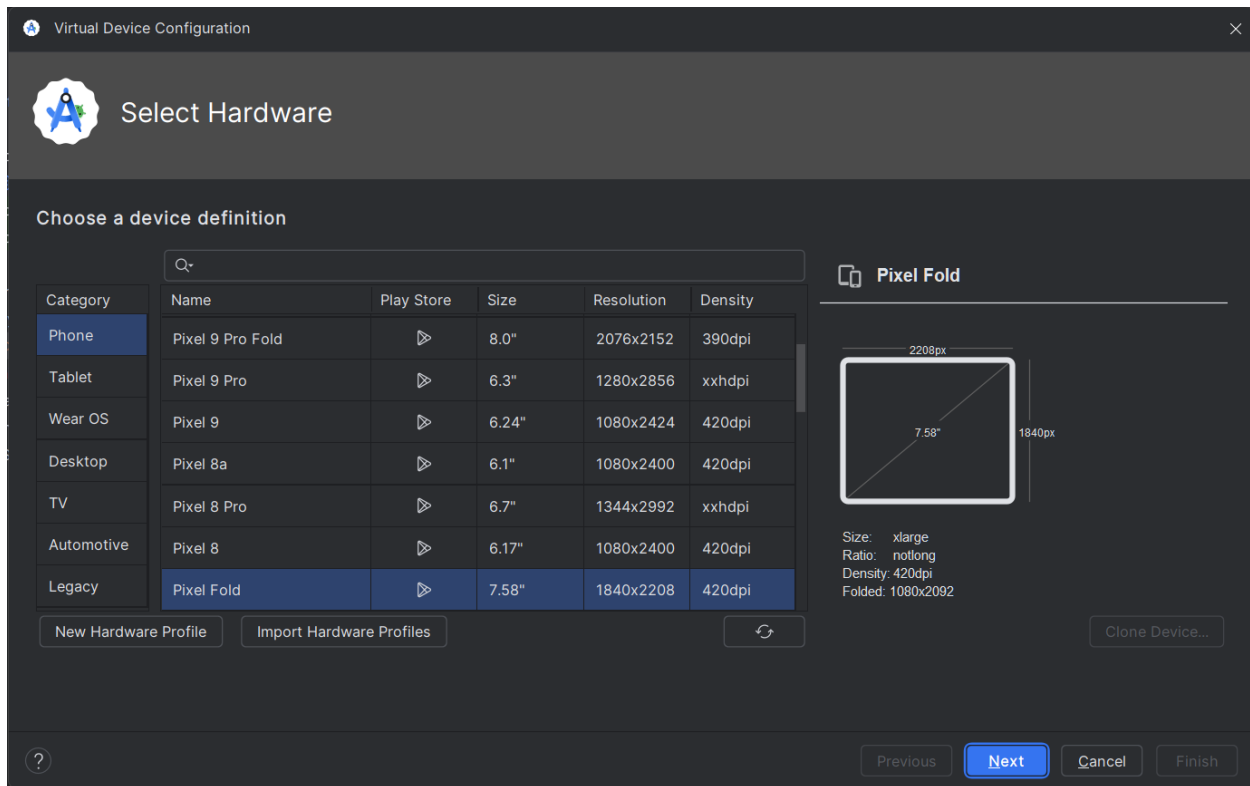
Trong tác vụ này, bạn sẽ sử dụng trình quản lý Thiết bị ảo Android (AVD) để tạo một thiết bị ảo (còn được gọi là trình giả lập) mô phỏng cấu hình cho một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng Trình giả lập Android có các yêu cầu bổ sung ngoài các yêu cầu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

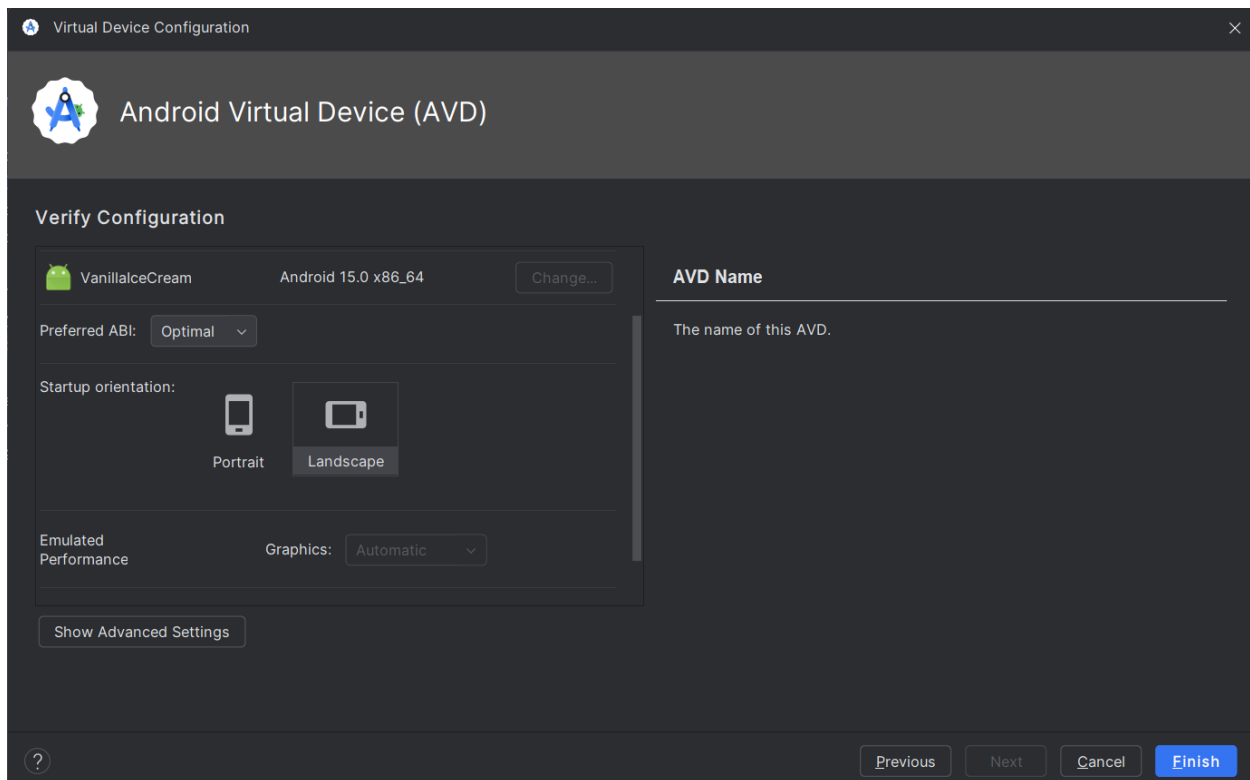
Khi sử dụng Trình quản lý AVD, bạn sẽ xác định các đặc điểm phần cứng của thiết bị, cấp độ API, bộ nhớ, giao diện và các thuộc tính khác của thiết bị và lưu dưới dạng thiết bị ảo. Với các thiết bị ảo, bạn có thể thử nghiệm ứng dụng trên các cấu hình thiết bị khác nhau (như máy tính bảng và điện thoại)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

3.1 Tạo thiết bị ảo Android (AVD)

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

1. Trong Android Studio, chọn Tool > Android > AVD Manager hoặc nhấp vào biểu tượng Trình quản lý AVD trên thanh công cụ. Màn hình Thiết bị ảo của bạn sẽ xuất hiện. Nếu bạn đã tạo thiết bị ảo, màn hình sẽ hiển thị chúng (như trong hình bên dưới); nếu không, bạn sẽ thấy danh sách trống.
2. Nhấp vào +Create Virtual Device. Cửa sổ Select Hardware sẽ xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng cung cấp một cột cho kích thước màn hình chéo (Size), độ phân giải màn hình tính bằng pixel (Resolution) và mật độ pixel (Density).





3. Chọn “Next” và tùy chọn cấu hình cho AVD cho phù hợp với thiết bị của bạn rồi nhấn Finish để kết thúc tạo AVD.

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** ứng dụng hoặc nhấp vào biểu tượng chạy trên thanh công cụ.
2. Nhấn vào biểu tượng **Device Manager** trên thanh công cụ bên phải và nhấn vào biểu tượng run máy ảo.

- **Stressors**
- **Stressors**
- **Stressors**



-
-
-



Trình giả lập khởi động và khởi động giống như một thiết bị vật lý. Tùy thuộc vào tốc độ máy tính của bạn, điều 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 từ đầu

phiên của bạn. 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 Thoát từ menu hoặc nhấn Control-Q trong Windows hoặc Command-Q trong macOS.

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

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

Bạn cần làm gì:

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

4.1 Bật gỡ lỗi USB

Để Android Studio giao tiếp với thiết bị của bạn, bạn phải bật Gỡ lỗi USB trên thiết bị Android của mình. Tính năng này được bật trong cài đặt Tùy chọn nhà phát triển của thiết bị.

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

1. Trên thiết bị của bạn, hãy mở **Setting**, tìm kiếm **About phone**, nhấp vào **About phone** và chạm vào **Build number** bảy lần.
2. Quay lại màn hình trước đó (Cài đặt / Hệ thống). Tùy chọn nhà phát triển xuất hiện trong danh sách.

Chạm vào Tùy chọn nhà phát triển.

3. Chọn **USB Debugging**.

4.2 Chạy ứng dụng của bạn trên thiết bị

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

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

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

Xử lý sự cố

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

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

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

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

Bạn có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Xem [tài liệu sử dụng thiết bị phần cứng](#)

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

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

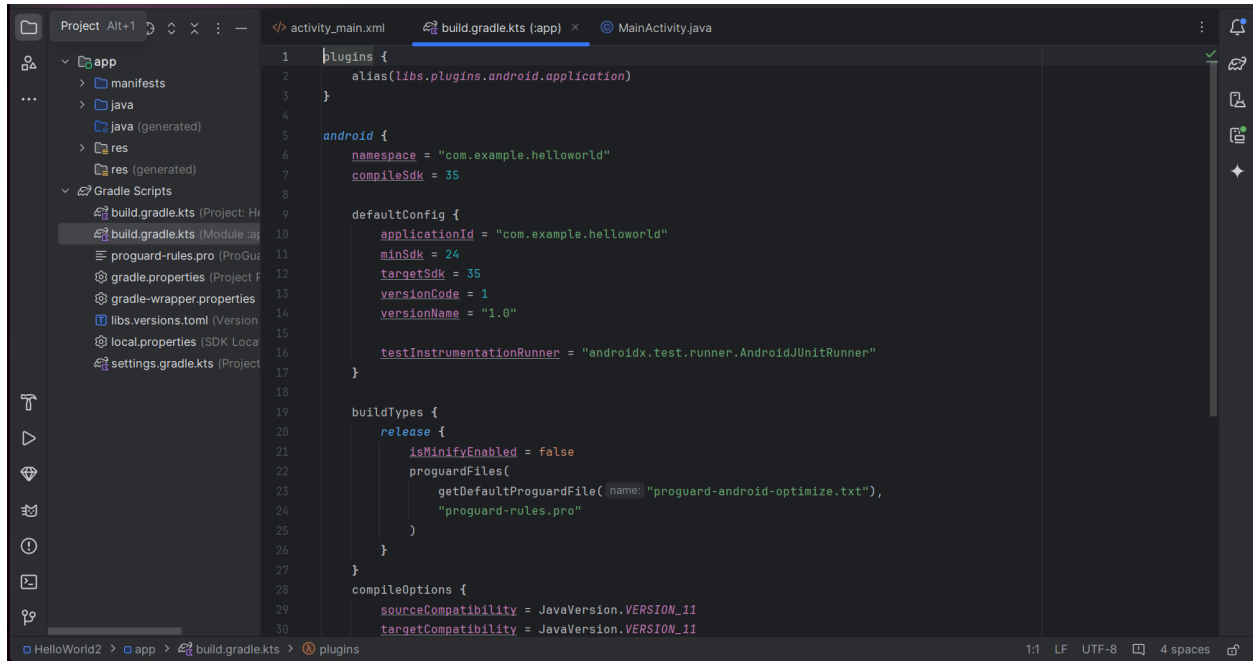
5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Thực hiện theo các bước sau:

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

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

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



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

5.2 Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi đối với các tệp cấu hình bản dựng trong một dự án, Android Studio yêu cầu bạn đồng bộ 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 Đồng bộ ngay trên thanh thông báo xuất hiện khi thực hiện thay đổi (như được hiển thị trong hình trước) hoặc nhấp vào biểu tượng Đồng bộ dự án với tệp Gradle trên thanh công cụ.

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

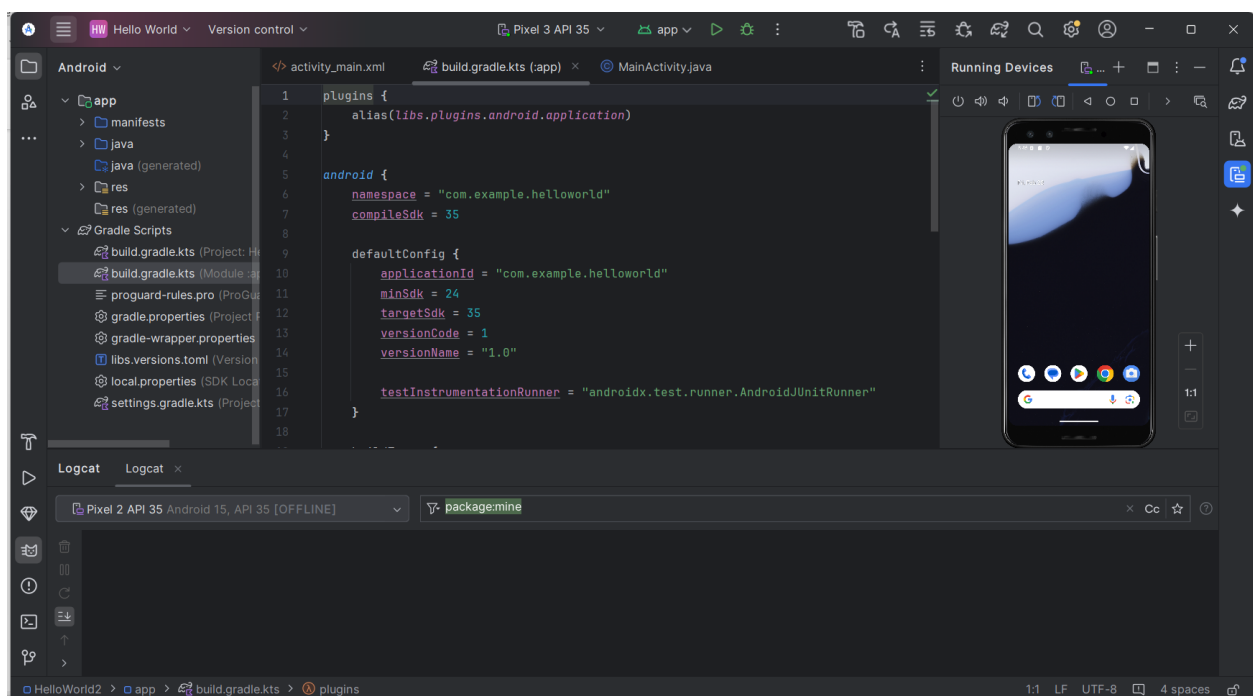
Để tìm hiểu sâu hơn về Gradle, hãy xem tài liệu tổng quan về hệ thống bản dựng và cấu hình bản dựng Gradle.

Nhiệm vụ 6: Thêm câu lệnh nhật ký vào ứng dụng của bạn

Trong nhiệm vụ này, bạn sẽ thêm 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 tab Logcat ở cuối cửa sổ Android Studio như được hiển thị 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 Log được đặt thành Verbose (mặc định), hiển thị tất cả thông báo Log. Các cài đặt khác bao gồm Gỡ lỗi, Lỗi, Thông tin và Cảnh báo.

6.2 Thêm câu lệnh log vào ứng dụng của bạn

Câu lệnh log trong mã ứng dụng của bạn sẽ hiển thị thông báo trong ngăn Logcat. Ví dụ: `Log.d("MainActivity", "Hello Word");`

Các phần của thông báo là:

- **Log:** Lớp **Log** để gửi thông báo nhật ký đến ngăn Logcat.
- **d:** Cài đặt mức Nhật ký gỡ lỗi để lọc hiển thị thông báo nhật ký trong ngăn Logcat. Các mức nhật ký khác là **e** cho Lỗi, **w** cho Cảnh báo và **i** cho Thông tin.
- **"MainActivity":** Đối số đầu tiên là thẻ có thể được sử dụng để lọc thông báo trong ngăn Logcat. Đây thường là tên của Hoạt động mà thông báo bắt nguồn. Tuy nhiên, bạn có thể biến điều này 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:

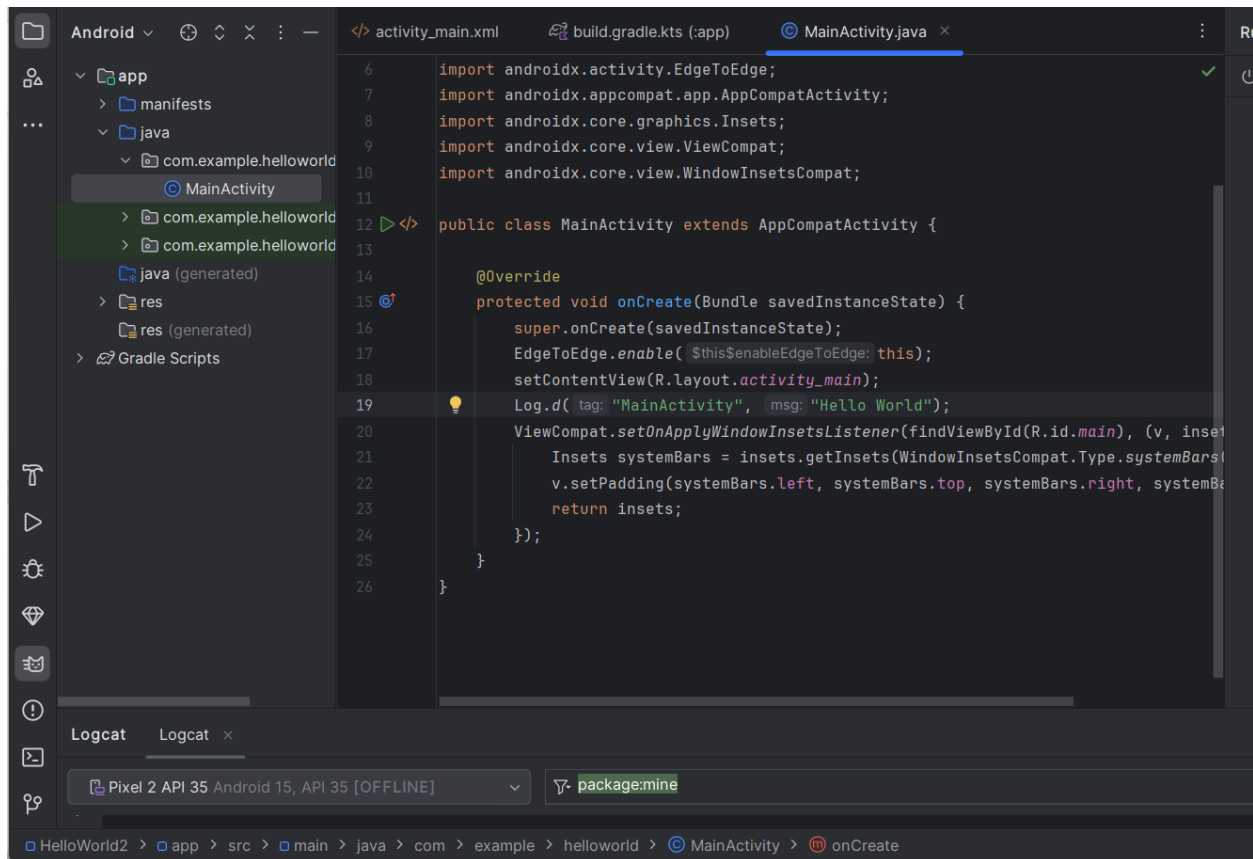
```
private static final String LOG_TAG = MainActivity.class.getSimpleName();
```

- **"Hello world":** Đối số thứ hai là thông báo thực tế.

Thực hiện theo các bước sau:

1. Mở ứng dụng Hello World của bạn trong Android studio và mở MainActivity.
2. Để tự động thêm các mục 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 đặt **Insert imports on paste** thành **All**.
4. Nhấp vào **Apply** rồi nhấp vào **OK**.
5. Trong phương thức `onCreate()` của MainActivity, hãy thêm câu lệnh sau:
`Log.d("MainActivity", "Hello World");`

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



6. Nếu ngăn Logcat chưa mở, hãy nhấp vào tab Logcat ở cuối Android Studio để mở nó.
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 Gỡ lỗi (hoặc để nguyên là **Verbose** vì có rất ít thông báo nhật ký).
9. Chạy ứng dụng 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.

Thử thách: Bây giờ bạn đã thiết lập và quen thuộc với quy trình phát triển cơ bản, hãy thực hiện các bước 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 sinh nhật gần đây.
3. (Tùy chọn) Chụp ảnh màn hình ứng dụng đã hoàn thành của bạn và gửi email cho người có sinh nhật mà bạn đã quên.
4. Một cách sử dụng phổ biến của lớp Log là ghi lại các ngoại lệ Java khi chúng xảy ra trong chương trình của bạn. Có một số phương pháp hữu ích, chẳng hạn như Log.e(), mà bạn có thể sử dụng cho mục đích này. Khám phá các phương pháp bạn có thể sử dụng để đưa ngoại lệ vào thông báo Nhật ký. Sau đó, viết mã vào ứng dụng của bạn để kích hoạt và ghi lại ngoại lệ.

Tóm tắt

- Để cài đặt Android Studio, hãy truy cập Android Studio và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo ứng dụng mới, hãy đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm SDK Tối thiểu.
- Để xem phân cấp Android của ứng dụng trong ngăn Dự án, hãy 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(Module:app) 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 đều nằm trong thư mục app và res. Thư mục java bao gồm các hoạt động, bài kiểm tra và các thành phần khác trong mã nguồn Java. Thư mục res chứa các tài nguyên, chẳng hạn như bố cục, chuỗi và hình ảnh.
- Chỉnh sửa tệp AndroidManifest.xml để thêm các thành phần tính năng và quyền 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 Nhật ký vào ứng dụng của bạn, hiển thị các thông báo trong ngăn Logcat như một công cụ cơ bản để gỡ lỗi.
- Để chạy ứng dụng của bạn trên thiết bị Android vật lý bằng Android Studio, hãy bật Gỡ lỗi USB trên thiết bị. Mở Setting > About phone và nhấn vào **Build**

number bảy lần. Quay lại màn hình trước đó (**Settings**) và nhấn vào **Developer options**. Chọn **USB Debugging**.

Các khái niệm liên quan

Tài liệu về khái niệm liên quan có trong [1.0: Giới thiệu to Android](#) và [1.1 Ứng dụng Android đầu tiên của bạn](#).

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

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 View. 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 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 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.

Mã Java hiển thị và điều khiển UI được chứa trong một lớp mở rộng Activity. Một Activity thường được liên kết với bố cục của các chế độ xem UI được định nghĩa là tệp XML (Ngôn ngữ đánh dấu mở rộng). Tệp XML này thường được đặt tên theo Activity của nó và định nghĩa bố cục của các thành phần View trên màn hình.

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

Trong các ứng dụng phức tạp hơn, một Activity có thể triển khai các hành động để phản hồi các lần 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 người dùng tương tác. Bạn tạo một ứng dụng bằng mẫu Empty Activity. Bạn cũng học cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục và cách chỉnh sửa bố cục trong XML. Bạn cần phát triển các kỹ năng này để có thể hoàn thành các bài thực hành khác trong khóa học này.

Những điều bạn cần biết

- Bạn cần phải quen thuộc với:
- Cách cài đặt và mở Android Studio.
- Cách tạo ứng dụng HelloWorld.
- Cách chạy ứng dụng HelloWorld.

Những điều bạn sẽ học

- Cách tạo ứng dụng có hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục.
- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới. Hãy xem bảng chú giải thuật ngữ và khái niệm từ vựng để biết các định nghĩa thân thiện.

Những điều bạn sẽ làm

- Tạo ứng dụng và thêm hai phần tử Nút và một TextView vào bố cục.
- Thao tác từng phần tử trong ConstraintLayout để giới hạn chúng vào lề và các phần tử khác.
- Thay đổi các thuộc tính phần tử UI.
- Chỉnh sửa bố cục của ứng dụng trong XML.
- Trích xuất các chuỗi được mã hóa cứng thành các tài nguyên chuỗi.
- Triển khai các phương thức xử lý nhấp để hiển thị thông báo trên màn hình khi người dùng chạm vào mỗi Nút.

Tổng quan về ứng dụng

Ứng dụng HelloToast bao gồm hai phần tử Button và một TextView. Khi người dùng chạm vào nút đầu tiên, ứng dụng sẽ hiển thị một thông báo ngắn (một Toast) trên màn hình. Chạm vào Nút thứ hai sẽ tăng bộ đếm "nhấp" được hiển thị trong TextView, bắt đầu từ số không.

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

Nhiệm vụ 1: Tạo và tìm hiểu 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:

Application Name	Hello Toast
Company Name	com.example.android (or your own domain)
Phone and Tablet Minimum SDK	API15: Android 4.0.3 IceCreamSandwich
Template	Empty Activity
Generate Layout file box	Selected
Backwards Compatibility box	Selected

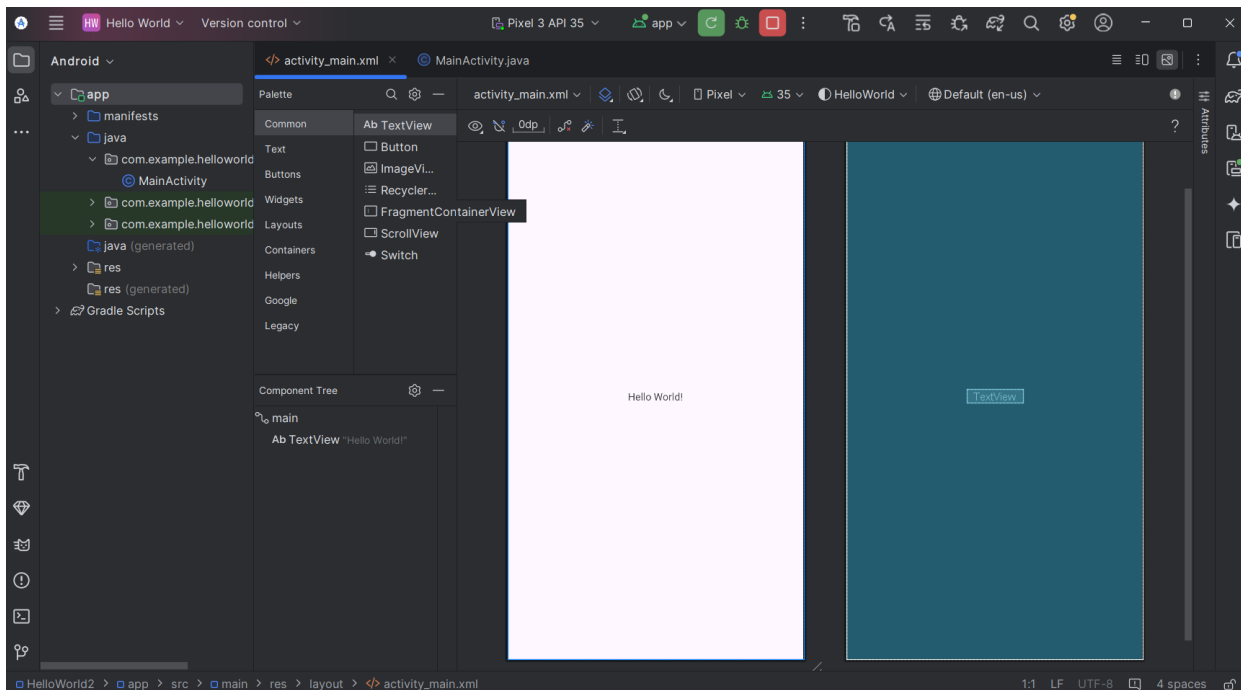
Chọn **Run > Run app** hoặc nhấp vào biểu tượng Run trên thanh công cụ để xây dựng và thực thi ứng dụng trên trình giả lập hoặc thiết bị của bạn.

1.2 Tìm hiểu 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 ứng dụng. Nó cho phép bạn kéo các thành phần vào chế độ

xem thiết kế trực quan và bản thiết kế, định vị chúng trong bố cục, thêm ràng buộc và đặt thuộc tính. Ràng buộc xác định vị trí của thành phần UI trong bố cục. Ràng buộc thể hiện kết nối hoặc căn chỉnh với chế độ xem khác, bố cục cha hoặc hướng dẫn vô hình.

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



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

Trong tác vụ này, bạn tạo bố cục UI cho ứng dụng HelloToast trong trình chỉnh sửa bố cục bằng các tính năng ConstraintLayout. Bạn có thể tạo các ràng buộc theo cách thủ công, như được hiển thị sau, hoặc tự động bằng công cụ **Autoconnect**.

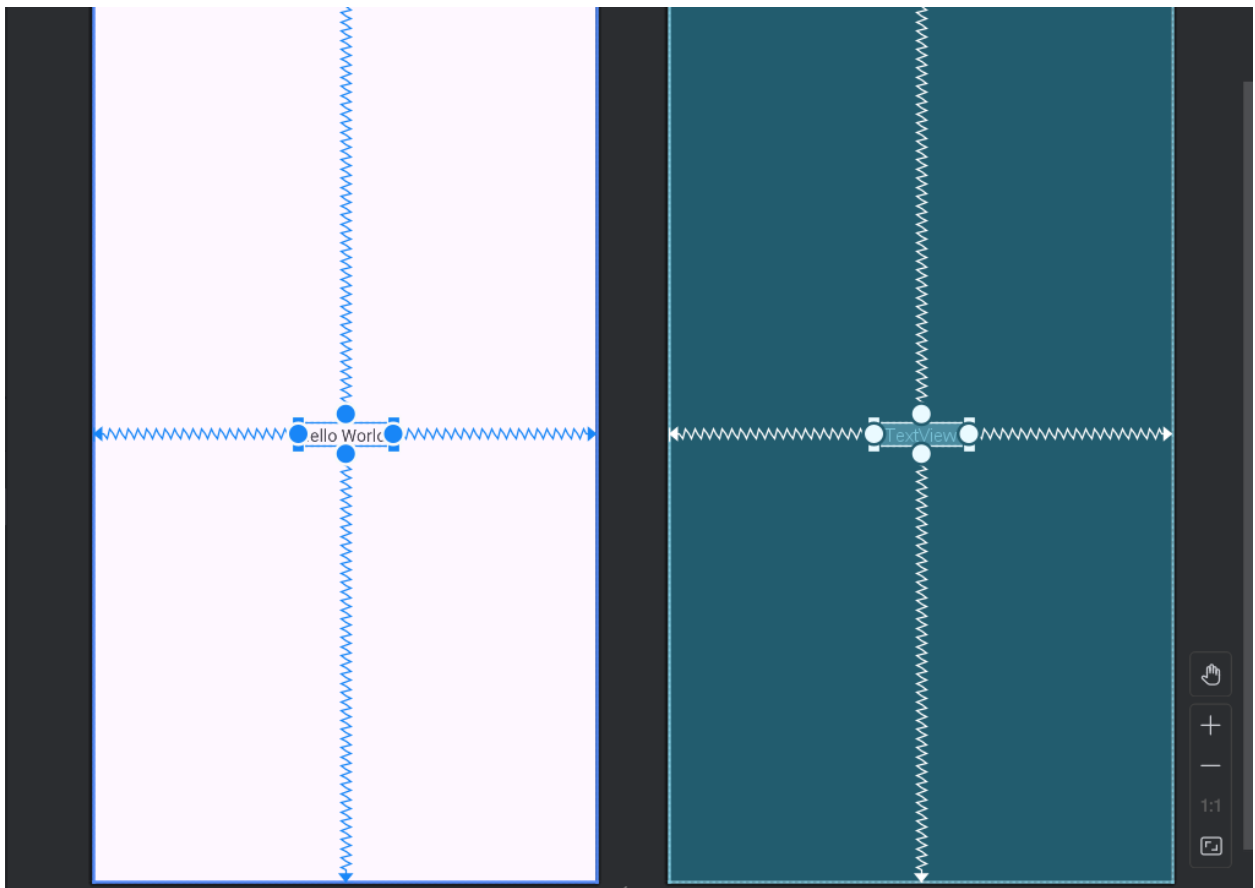
2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện theo các bước sau:

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

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

2. Công cụ Autoconnect cũng nằm trong thanh công cụ. Theo mặc định, công cụ này được bật. Đối với bước này, hãy đảm bảo rằng công cụ không bị tắt.
3. Nhấp vào nút phóng to để phóng to các ngăn thiết kế và bản thiết kế để xem cận cảnh.
4. Chọn TextView trong ngăn Component Tree. TextView "Hello World" được tô sáng trong các ngăn thiết kế và bản thiết kế và các ràng buộc cho phần tử sẽ hiển thị.
5. Tham khảo hình ảnh động bên dưới để biết bước này. Nhấp vào tay cầm hình tròn ở bên phải của TextView để xóa ràng buộc theo chiều ngang liên kết chế độ xem với bên phải của bố cục. TextView nhảy sang bên trái vì nó không còn bị ràng buộc ở bên phải nữa. Để thêm lại ràng buộc theo chiều ngang, hãy nhấp vào cùng tay cầm đó và kéo một đường sang bên phải của bố cục.



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

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



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

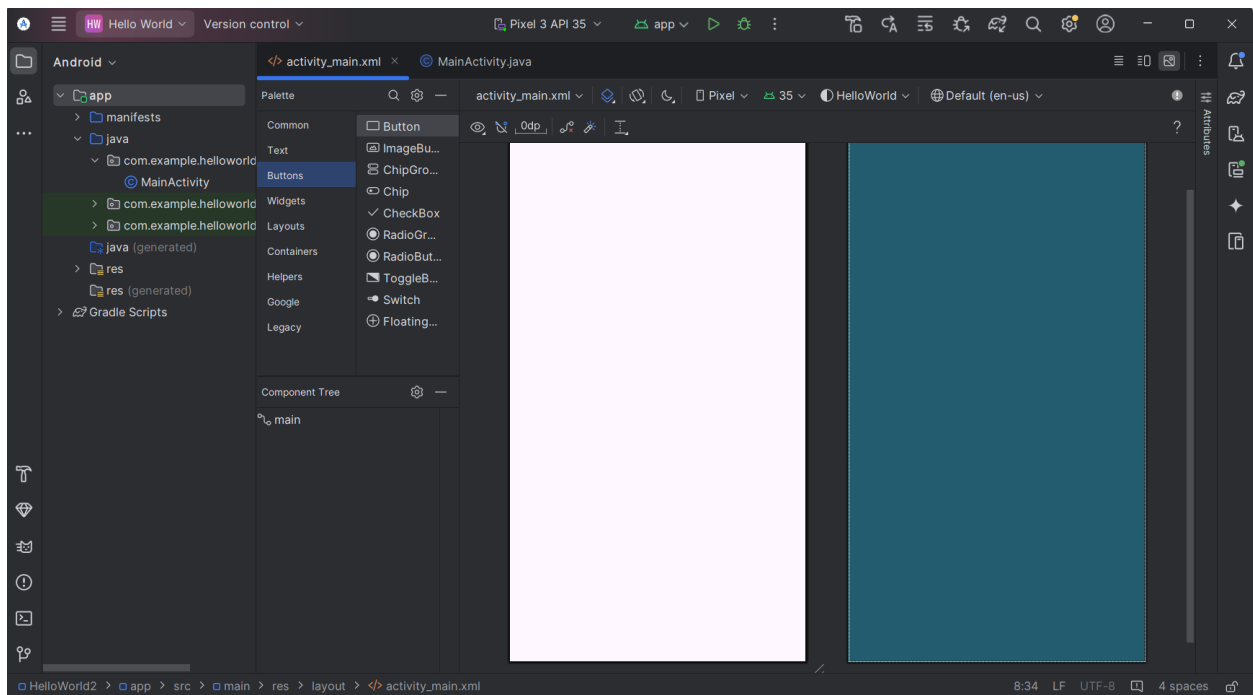


2.2 Thêm Nút vào bố cục

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

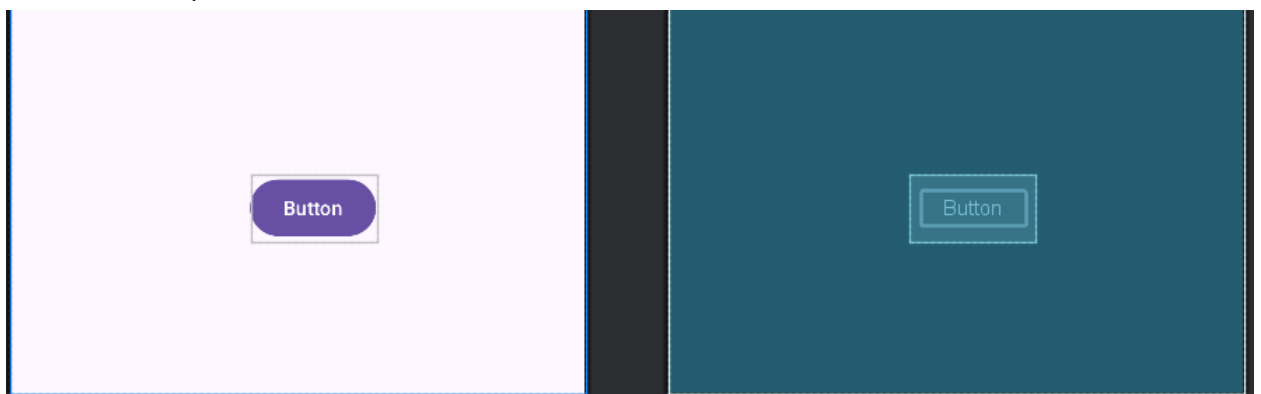
Thực hiện theo các bước sau để thêm Nút:

1. Bắt đầu với một bảng trắng. Thành phần TextView không cần thiết, vì vậy khi thành phần này vẫn được chọn, hãy nhấn phím **Delete** hoặc chọn **Edit > Delete**. Bây giờ bạn có một bố cục hoàn toàn trống.
2. Kéo **Button** từ ngăn **Palette** đến bất kỳ vị trí nào trong bố cục. Nếu bạn thả 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 cùng, bên trái và bên phải của bố cục như 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 trong hình động bên dưới. Autoconnect có thể cung cấp các ràng buộc theo chiều ngang cho bạn (nếu không, bạn có thể tự kéo chúng).
2. Kéo một ràng buộc theo chiều dọc xuống dưới cùng của bố cục (tham khảo hình bên dưới).



Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử và di con trỏ lên trên để hiển thị nút Xóa ràng buộc. Nhấp vào nút này để xóa tất cả các ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc duy nhất, hãy nhấp vào tay cầm cụ thể đặt ràng buộc đó. Để xóa tất cả các ràng buộc trong toàn bộ bố cục, hãy

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

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

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

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

3.1 Thay đổi kích thước Nút

Trình chỉnh sửa bố cục cung cấp các nút điều chỉnh kích thước ở cả bốn góc của Chế độ xem để bạn có thể thay đổi kích thước chế độ xem một cách nhanh chóng. Bạn có thể kéo các nút điều chỉnh ở mỗi góc của Chế độ xem để thay đổi kích thước, nhưng làm như vậy sẽ mã hóa cứng các kích thước chiều rộng và chiều cao. Tránh mã hóa cứng các kích thước cho hầu hết các thành phần Chế độ xem, vì các kích thước được mã hóa cứng không thể thích ứng với các kích thước nội dung và màn hình khác nhau. Thay vào đó, hãy sử dụng ngăn **Attributes** ở bên phải của trình chỉnh sửa bố cục để chọn chế độ kích thước không sử dụng các kích thước được mã hóa cứng. Ngăn **Attributes** bao gồm một bảng điều khiển kích thước hình vuông được gọi là trình kiểm tra chế độ xem ở trên cùng. Các ký hiệu bên trong hình vuông biểu thị các thiết lập chiều cao và chiều rộng như sau:

Attributes

Button

button

id

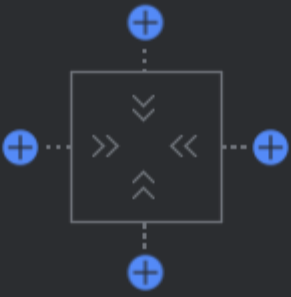
button

> Declared Attributes

+ -

Layout

Constraint Widget



Constraints

! Not Horizontally Constrained

! Not Vertically Constrained

layout_width

wrap_content

▼

◻

layout_height

wrap_content

▼

◻

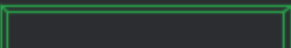
visibility

▼

visibility

▼

Transforms

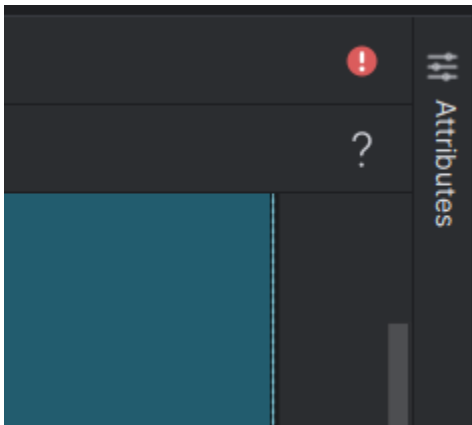


Trong hình trên:

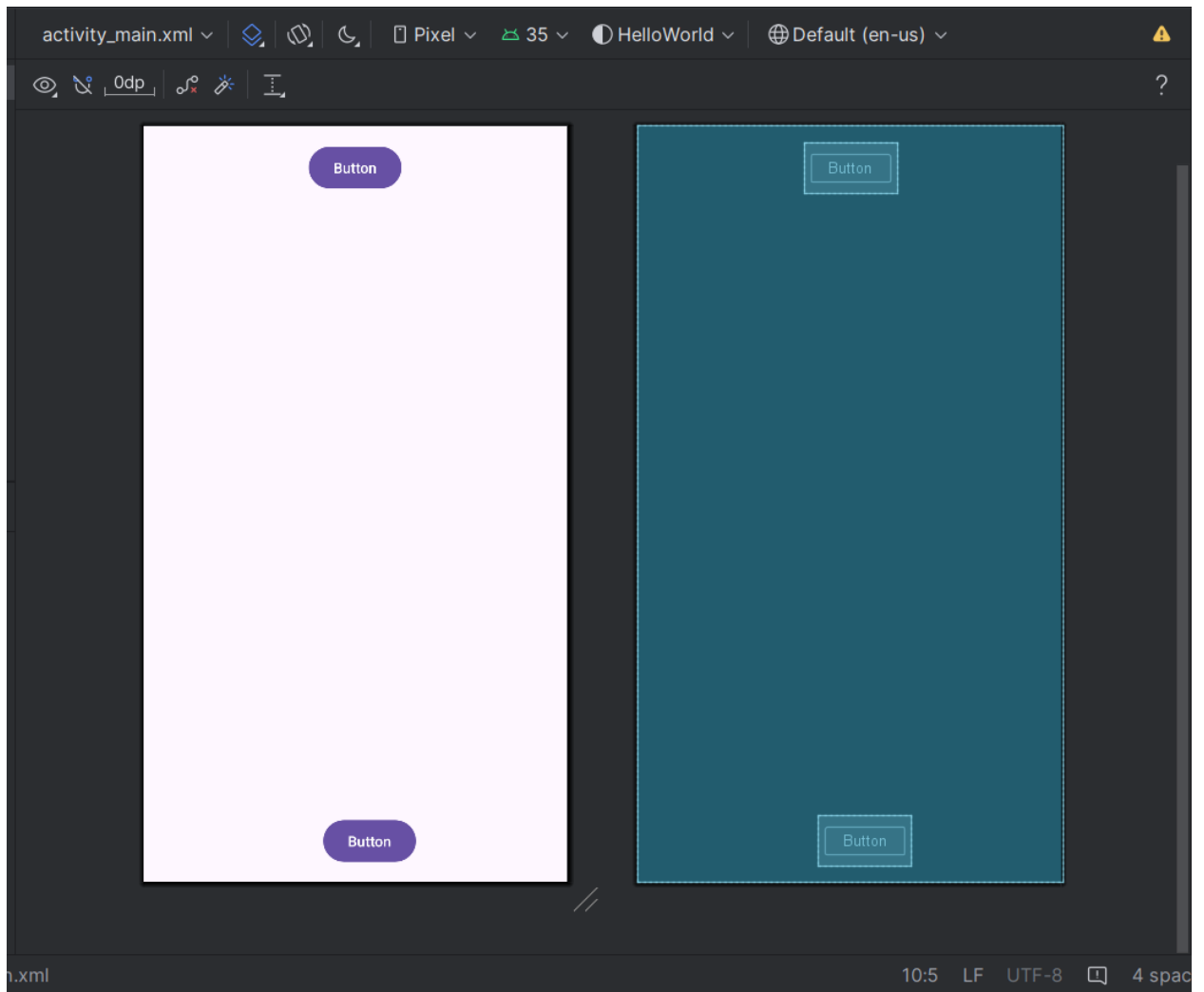
1. Kiểm soát chiều cao. Kiểm soát này chỉ định thuộc tính `layout_height` và xuất hiện ở hai phân đoạn ở phía trên và phía dưới của hình vuông. Các góc cho biết rằng kiểm soát này được đặt thành `wrap_content`, nghĩa là Chế độ xem sẽ mở rộng theo chiều dọc khi cần để phù hợp với nội dung của nó. "8" biểu thị lề chuẩn được đặt thành 8dp.
2. Kiểm soát chiều rộng. Kiểm soát này chỉ định `layout_width` và xuất hiện trong hai phân đoạn ở bên trái và bên phải của hình vuông. Các góc cho biết kiểm soát này được đặt thành `wrap_content`, có nghĩa là Chế độ xem sẽ mở rộng theo chiều ngang khi cần để vừa với nội dung của nó, lên đến lề 8dp.
3. Nút đóng ngăn thuộc tính. Nhấp để đóng ngăn.

Thực hiện theo các bước sau:

1. Chọn Nút trên cùng trong ngăn cây thành phần.
2. Nhấp vào tab thuộc tính ở bên phải cửa sổ trình chỉnh sửa bố cục.

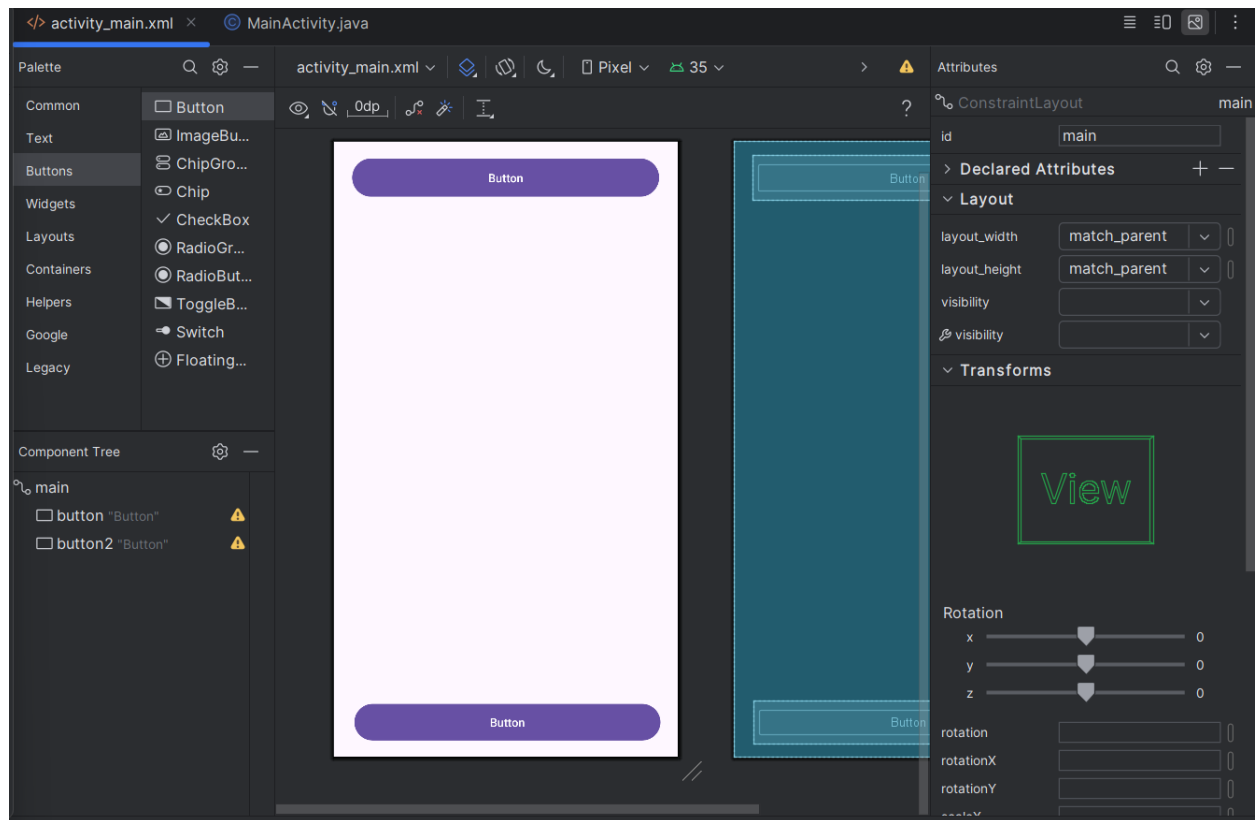


3. Nhấp vào điều khiển chiều rộng hai lần—lần nhấp đầu tiên sẽ thay đổi thành Cố định với các đường thẳng và lần nhấp thứ sẽ thay đổi thành Phù hợp với các ràng buộc với các cuộn lò xo, như thể hiện trong hình động bên dưới.



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

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



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

- Thiết lập `match_constraint` mở rộng phần tử View để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao—lên đến một lề, nếu có. Phần tử cha trong trường hợp này là `ConstraintLayout`. Bạn tìm hiểu thêm về `ConstraintLayout` trong tác vụ tiếp theo.
- Thiết lập `wrap_content` thu nhỏ kích thước của phần tử View để nó chỉ đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, phần tử View sẽ trở nên vô hình.
- Để chỉ định kích thước cố định điều chỉnh theo kích thước màn hình của thiết bị, hãy sử dụng số lượng cố định pixel không phụ thuộc vào mật độ (đơn vị dp). Ví dụ: 16dp nghĩa là 16 pixel không phụ thuộc vào mật độ.

Mẹo: Nếu bạn thay đổi thuộc tính `layout_width` bằng menu bật lên, thuộc tính `layout_width` sẽ được đặt thành 0 vì không có kích thước nào được đặt. Thiết lập này

giống như `match_constraint`—chế độ xem có thể mở rộng tối đa để đáp ứng các ràng buộc và thiết lập lề.

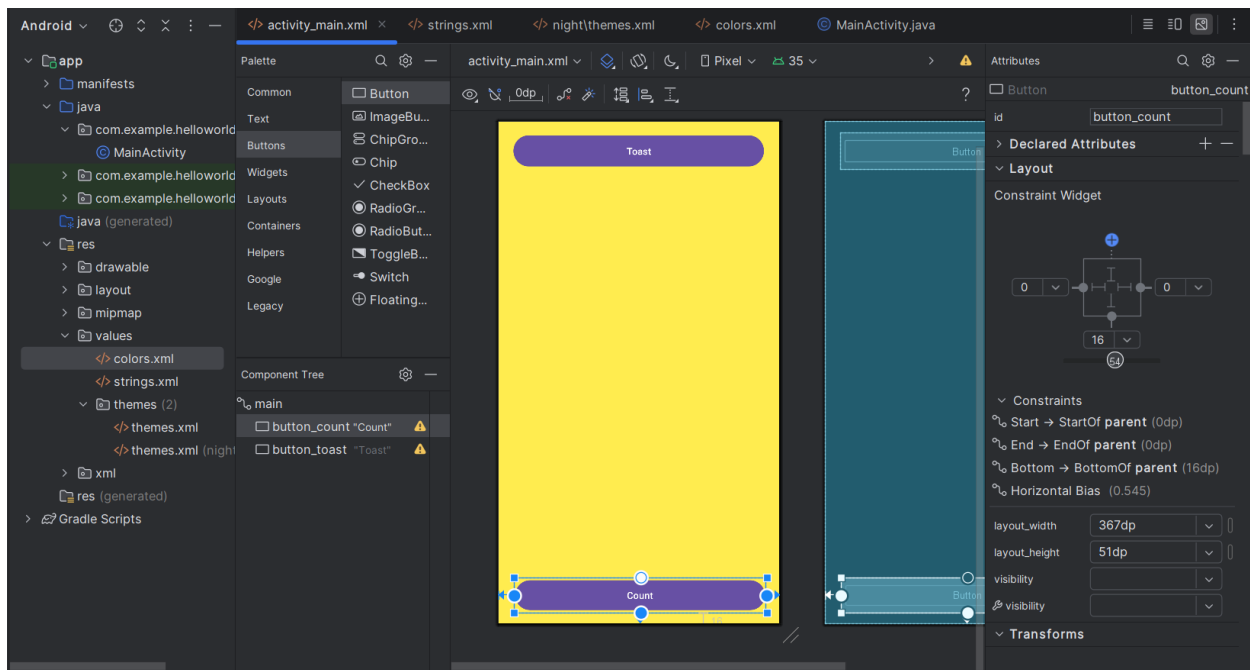
3.2 Thay đổi các thuộc tính của Button

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

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

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

1. Sau khi chọn Nút đầu tiên, hãy chỉnh sửa trường ID ở đầu ngăn Thuộc tính thành `button_toast` cho thuộc tính `android:id`, được sử dụng để xác định phần tử trong bố cục.
2. Đặt thuộc tính nền 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 văn bản 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 sở chủ đề được xác định trước được xác định trong tệp tài nguyên `colors.xml`. Nó được sử dụng cho thành ứng dụng. Sử dụng màu cơ sở cho các thành phần UI khác sẽ tạo ra một UI thống nhất. Bạn sẽ tìm hiểu thêm về chủ đề ứng dụng và Material Design trong một bài học khác

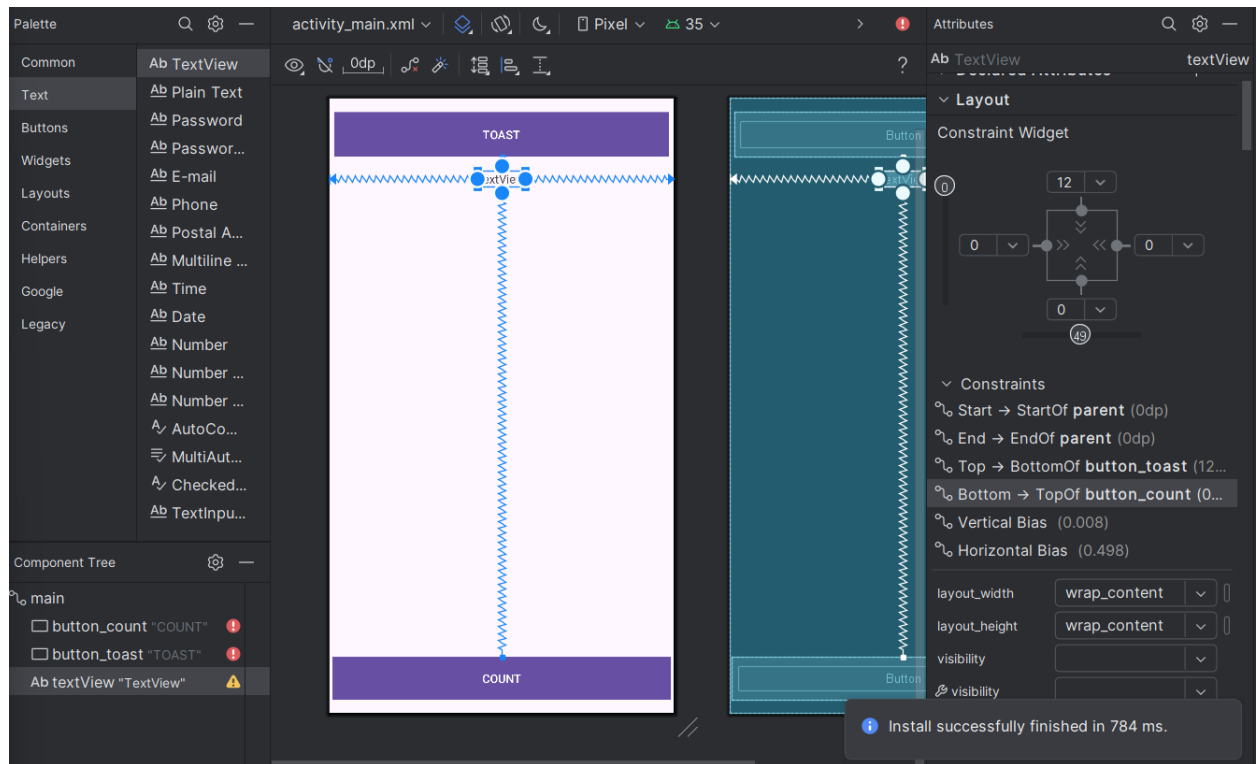
Nhiệm vụ 4: Thêm `TextEdit` và thiết lập các thuộc tính

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 theo chiều ngang với lề và theo chiều dọc với hai phần tử `Button`. Sau đó, bạn sẽ thay đổi các thuộc tính cho `TextView` trong ngăn Thuộc tính.

4.1 Thêm `TextView` và các ràng buộc

- Như được hiển thị trong hình động bên dưới, hãy kéo một `TextView` từ ngăn Palette đến phần trên của bố cục và kéo một ràng buộc từ đầu `TextView` đến tay cầm ở dưới cùng của Nút `Toast`. Thao tác này ràng buộc `TextView` nằm bên dưới nút.

2. Như được hiển thị trong hình động bên dưới, hãy kéo một ràng buộc từ dưới cùng của TextView đến tay cầm ở đầu Nút Count và từ các cạnh của TextView đến các cạnh của bố cục. Điều này hạn chế TextView ở giữa bố cục giữa

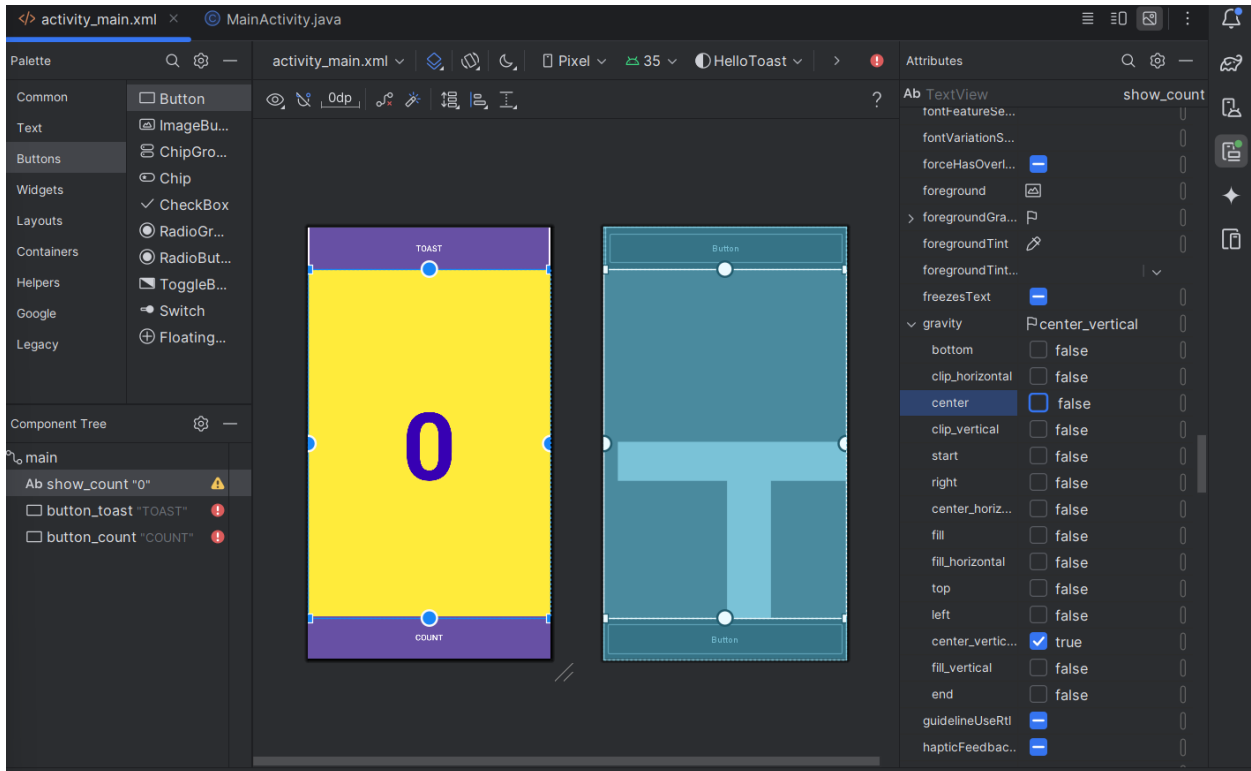


4.2 Đặt thuộc tính TextView

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

1. Đặt ID thành show_count.
2. Đặt văn bản thành 0.
3. Đặt textSize thành 160sp.
4. Đặt textStyle thành B (in đậm) và textAlignment thành ALIGNCENTER (căn giữa đoạn văn).
5. Thay đổi các điều khiển kích thước chế độ xem theo chiều ngang và chiều dọc (layout_width và layout_height) thành match_constraint.
6. Đặt textColor thành @color/colorPrimary.

7. Cuộn xuống gần và nhấp vào View all attributes (Xem tất cả các thuộc tính), cuộn xuống trang thứ hai của attributes thành background (thuộc tính thành nền), sau đó nhập #FFFF00 để có màu vàng.
8. Cuộn xuống gravity (trọng lực), mở rộng gravity (trọng lực) và chọn center_ver (cho center-vertical (tâm dọc)).

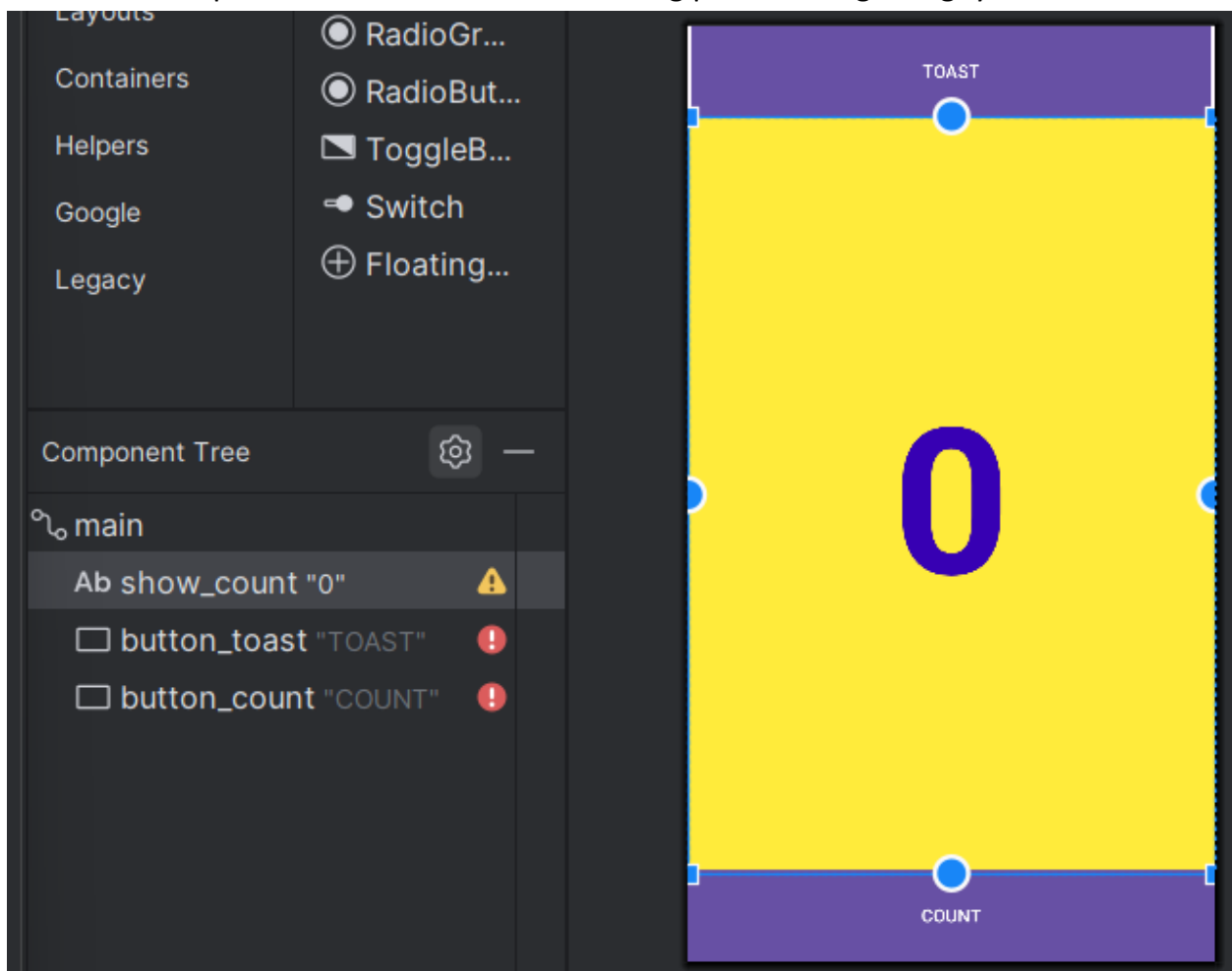


- **extSize:** Kích thước văn bản của TextView. Đối với bài học này, kích thước được đặt thành 160sp. Sp viết tắt của pixel không phụ thuộc vào tỷ lệ, và giống như dp, là một đơn vị tỷ lệ với mật độ màn hình và tùy chọn kích thước phông chữ của người dùng. Sử dụng các đơn vị dp khi bạn chỉ định kích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

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

Nhiệm vụ 5: Chỉnh sửa bố cục trong XML

Bố cục ứng dụng Hello Toast gần hoàn thiện! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử UI trong Cây thành phần. Di con trỏ qua các dấu chấm than này để xem các thông báo cảnh báo, như được hiển thị bên dưới. Cảnh báo tương tự xuất hiện cho cả ba phần tử: chuỗi được mã hóa cứng phải sử dụng tài nguyên.



Cách dễ nhất để khắc phục sự cố 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 trong mã nguồn XML.

5.1 Mở mã XML cho bố cục

Đối với nhiệm vụ này, hãy mở tệp `activity_main.xml` nếu tệp chưa được mở và nhấp vào tab Text bản ở cuối trình chỉnh sửa bố cục.

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

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/show_count"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:background="#FFEB3B"
        android:gravity="center_vertical"
        android:text="0"
        android:textAlignment="center"
        android:textColor="@color/design_default_color_primary_dark"
        android:textSize="16sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/button_count"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button_toast"
        app:layout_constraintVertical_bias="0.0"
        tools:ignore="RtlCompat" />
    <Button
        android:id="@+id/button_toast"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

5.2 Trích xuất tài nguyên chuỗi


Thay vì mã hóa cứng chuỗi, cách tốt nhất là sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Việc có các chuỗi trong một tệp riêng giúp quản lý chúng dễ dàng hơn, đặc biệt là nếu bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc

để dịch và bản địa hóa ứng dụng của bạn, vì bạn cần tạo tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhấp một lần vào từ "Toast" (cảnh báo được tô sáng 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. Nhấp vào OK. Một tài nguyên chuỗi được tạo trong tệp values/res/string.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 ngăn Project > Android, hãy mở rộng các giá trị trong res, sau đó nhấp đúp vào strings.xml để xem tài nguyên chuỗi của bạn trong tệp strings.xml:

```
1 <resources>
2     <string name="app_name">Hello Toast</string>
3     <string name="count_initial_value">0</string>
4      <string name="button_label_toast">TOAST</string>
5     <string name="button_label_count">COUNT</string>
6
7 </resources>
```

Bạn cần một chuỗi khác để sử dụng trong tác 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!":

```

1  <resources>
2      <string name="app_name">Hello Toast</string>
3      <string name="count_initial_value">0</string>
4      <string name="button_label_toast">TOAST</string>
5       <string name="button_label_count">COUNT</string>
6      <string name="toast_message">Hello Toast!</string>
7  </resources>

```

Mẹo: Tài nguyên chuỗi bao gồm tên ứng dụng, xuất hiện trên 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 xử lý onClick cho các nút

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

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

Trình xử lý nhấp là phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick trong ngăn **Attributes** của tab **Design**. Bạn cũng có thể chỉ định tên của phương thức xử lý trong trình soạn thảo XML bằng cách thêm thuộc tính android:onClick vào Nút. Bạn sẽ sử dụng phương thức sau vì bạn chưa tạo các phương thức xử lý và trình soạn thảo XML cung cấp một cách tự động để tạo các phương thức đó.

1. Khi trình soạn thảo XML mở (tab Text), hãy tìm nút có android:id được đặt thành button_toast:

```

29      <Button
30          android:id="@+id/button_toast"
31          android:layout_width="406dp"
32          android:layout_height="71dp"
33          android:background="@color/design_default_color_on_primary"
34          android:text="@string/button_label_toast"
35          android:textAlignment="center"
36          android:textColor="@color/white"
37          app:layout_constraintEnd_toEndOf="parent"
38          app:layout_constraintStart_toStartOf="parent"
39          app:layout_constraintTop_toTopOf="parent" />
40

```

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

```

40          android:onClick="showToast"/>
41

```

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

Nếu biểu tượng bóng đèn màu đỏ không xuất hiện, hãy nhấp vào tên phương thức ("showToast"). Nhấn Alt-Enter (Option-Enter trên máy Mac), chọn Tạo 'showToast(view)' trong MainActivity và nhấp vào OK.

Hành động này tạo một stub phương thức giữ chỗ 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`: Thêm thuộc tính `android:onClick` vào cuối và thêm trình xử lý nhấp:

```

52          android:onClick="countUp"/>
53

```

Mã XML cho các thành phần UI trong `ConstraintLayout` hiện trông như thế này:

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

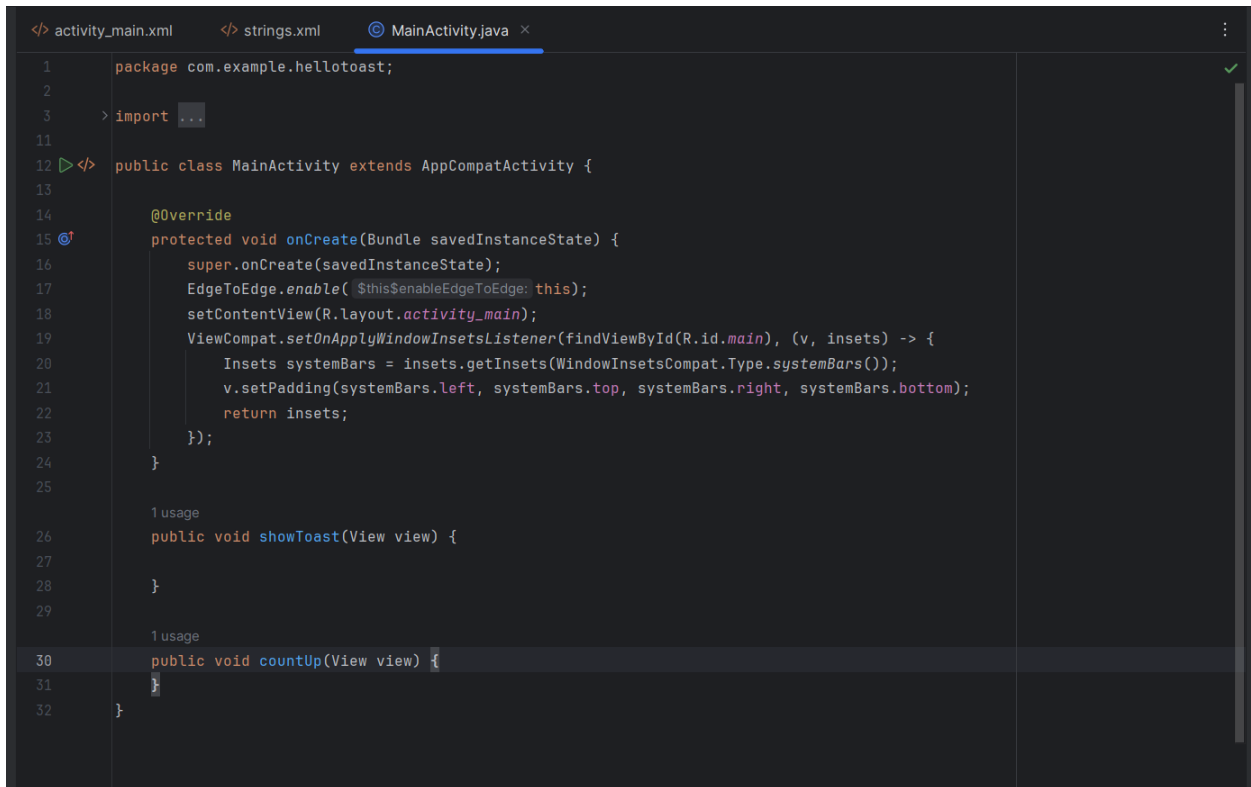
    <Button
        android:id="@+id/button_toast"
        android:layout_width="406dp"
        android:layout_height="71dp"
        android:background="@color/design_default_color_on_primary"
        android:text="@string/button_label_toast"
        android:textAlignment="center"
        android:textColor="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:onClick="showToast"/>

    <Button
        android:id="@+id/button_count"
        android:layout_width="411dp"
        android:layout_height="71dp"
        android:background="@color/white"
        android:text="@string/button_label_count"
        android:textColor="@color/design_default_color_on_primary"
        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:background="#FFEB3B"
        android:gravity="center"
        android:text="@string/count_initial_value"
        android:textAlignment="center"
        android:textColor="@color/design_default_color_primary_dark"
        android:textSize="160sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/button_count"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button_toast"
        app:layout_constraintVertical_bias="0.0"
        tools:ignore="RtlCompat" />

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

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



```
1 package com.example.hellotoast;
2
3 > import ...
11
12 <> public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         EdgeToEdge.enable(this);
18         setContentView(R.layout.activity_main);
19         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
20             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
21             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
22             return insets;
23         });
24     }
25
26     1 usage
27     public void showToast(View view) {
28
29     }
30
31     1 usage
32     public void countUp(View view) {
33
34     }
35 }
```

6.2 Chỉnh sửa trình xử lý Nút Toast

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

Làm theo các bước sau để chỉnh sửa trình xử lý nhấp chuột Nút Toast:

1. Xác định vị trí phương thức `showToast()` mới được tạo.

```

public void showToast(View view) {
}

```

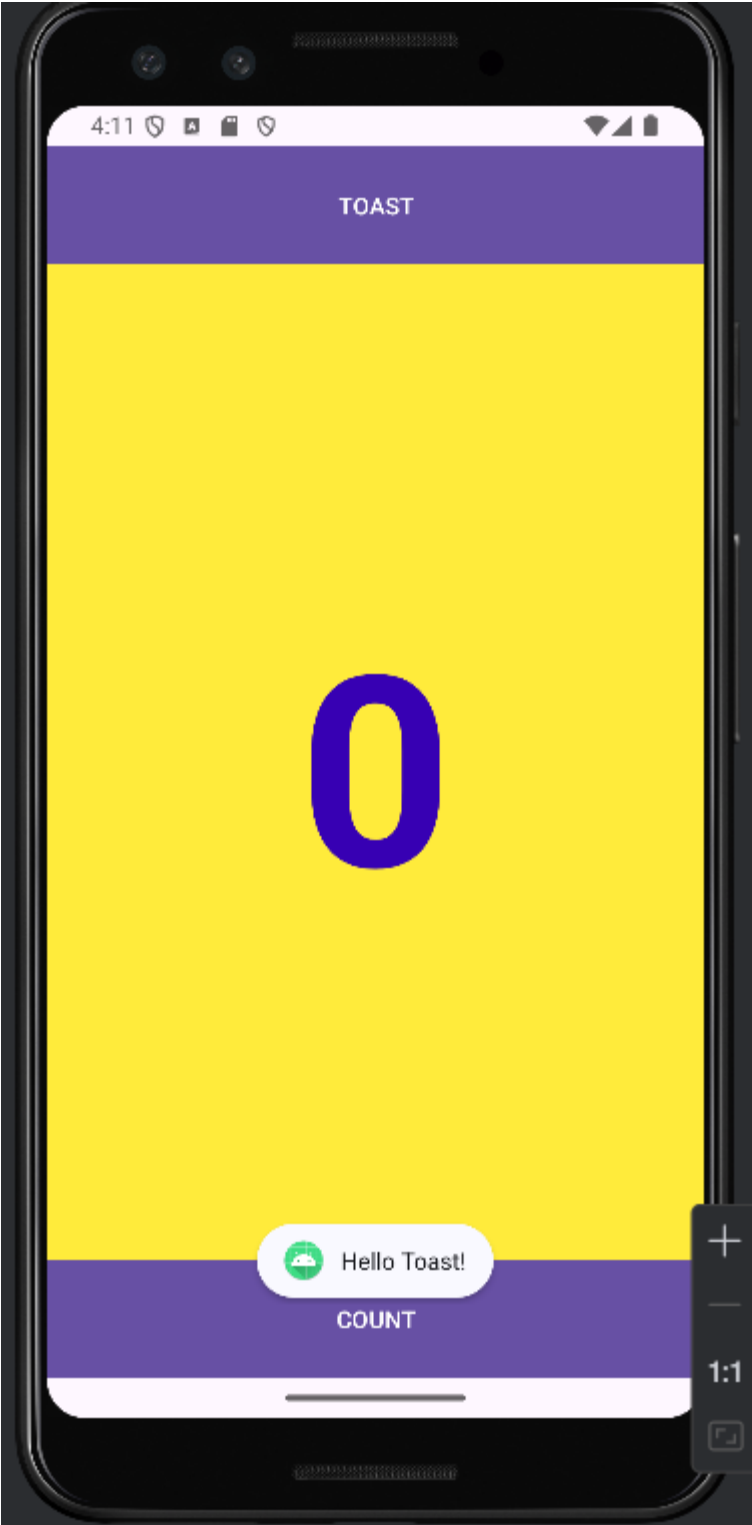
2. Để tạo một phiên bản Toast, hãy gọi phương thức `makeText()` trên lớp `Toast`.
Câu lệnh này chưa hoàn tất cho đến khi bạn hoàn tất tất cả các bước.
3. Cung cấp ngữ cảnh của hoạt động ứng dụng. Vì Toast hiển thị ở trên cùng của giao diện người dùng hoạt động, hệ thống cần thông tin về hoạt động hiện tại. Khi bạn đã ở trong ngữ cảnh của hoạt động mà ngữ cảnh bạn cần, hãy sử dụng điều này như một phím tắt.
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`.
5. Cung cấp thời lượng hiển thị. Ví dụ: `Toast.LENGTH_SHORT` hiển thị toast trong thời gian tương đối ngắn.
Thời lượng hiển thị Toast có thể là `Toast.LENGTH_LONG` hoặc `Toast.LENGTH_SHORT`.
Độ dài thực tế là khoảng 3,5 giây cho Toast dài và 2 giây cho Toast ngắn.
6. Hiển thị Toast bằng cách gọi `show()`. Sau đây là toàn bộ phương thức `showToast()`:

```

27  public void showToast(View view) {
28      Toast toast = Toast.makeText(context: this, R.string.toast_message, Toast.LENGTH_SHORT);
29      toast.show();
30  }
31

```

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



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()` mới tạo
2. Để theo dõi số đếm, bạn cần một biến thành viên riêng. Mỗi lần nhấn nút Đếm sẽ làm tăng giá trị của biến này. Nhập nội dung sau, nội dung này sẽ được tô sáng bằng màu đỏ và hiển thị biểu tượng bóng đèn màu đỏ:

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

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

```
13 </> public class MainActivity extends AppCompatActivity {  
14  
15     1 usage  
    private int mCount;  
16
```

4. Thay đổi câu lệnh biến thành viên riêng tư để khởi tạo biến thành số 0:

```
13 </> public class MainActivity extends AppCompatActivity {  
14  
15     1 usage  
    private int mCount = 0;  
16
```

5. Cùng với biến ở trên, bạn cũng cần một biến thành viên riêng tư để tham chiếu đến Show_count TextView, mà bạn sẽ thêm vào trình xử lý nhấp chuột. Gọi biến này mShowCount:
6. Bây giờ bạn đã có mShowCount, bạn có thể lấy tham chiếu đến TextView bằng ID bạn đã đặt trong tệp bố cục. Để lấy tham chiếu này chỉ một lần, hãy chỉ định tham chiếu đó trong phương thức onCreate() như bạn đã tìm hiểu trong bài học khác, phương thức onCreate() được sử dụng để làm phòng bố cục, có nghĩa là đặt chế độ xem nội dung của màn hình thành bố cục XML. Bạn cũng có thể sử dụng phương thức này để lấy tham chiếu đến các thành phần UI khác trong bố cục, chẳng hạn như TextView. Xác định vị trí phương thức onCreate() trong MainActivity:

```
20  protected void onCreate(Bundle savedInstanceState) {  
21      super.onCreate(savedInstanceState);  
22      EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
23      setContentView(R.layout.activity_main);  
24      ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {  
25          Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());  
26          v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
27          return insets;  
28      }));  
29  }
```

7. Thêm câu lệnh findViewById vào cuối phương thức:

```
19  @Override  
20  protected void onCreate(Bundle savedInstanceState) {  
21      super.onCreate(savedInstanceState);  
22      EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
23      setContentView(R.layout.activity_main);  
24      mShowCount = (TextView) findViewById(R.id.show_count);  
25      ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {  
26          Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());  
27          v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
28          return insets;  
29      }));  
30  }
```

View, giống như một chuỗi, là một tài nguyên có thể có một id. Lệnh gọi findViewById lấy ID của một view làm tham số và trả về View. Vì phương thức trả về một View, bạn phải ép 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 để đặt văn bản trong TextView thành giá trị của biến mCount. Thêm nội dung sau vào phương thức countUp():

Toàn bộ phương thức countUp() hiện trông như thế này:

```
1 usage
37     public void countUp(View view) {
38         mCount++;
39         if(mShowCount != null) mShowCount.setText("" + mCount);
40     }
41 }
```

9. Chạy ứng dụng để xác minh số đếm tăng lên khi bạn chạm vào nút **Count**.



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

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

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