**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 4](#_Toc190855147)

[Bài 1) Tạo ứng dụng đầu tiên 4](#_Toc190855148)

[1.1) Android Studio và Hello World 4](#_Toc190855149)

[1.2) Giao diện người dùng tương tác đầu tiên 5](#_Toc190855150)

[1.3) Trình chỉnh sửa bố cục 5](#_Toc190855151)

[1.4) Văn bản và các chế độ cuộn 5](#_Toc190855152)

[1.5) Tài nguyên có sẵn 5](#_Toc190855153)

[Bài 2) Activities 5](#_Toc190855154)

[2.1) Activity và Intent 5](#_Toc190855155)

[2.2) Vòng đời của Activity và trạng thái 5](#_Toc190855156)

[2.3) Intent ngầm định 5](#_Toc190855157)

[Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ 5](#_Toc190855158)

[3.1) Trình gỡ lỗi 5](#_Toc190855159)

[3.2) Kiểm thử đơn vị 5](#_Toc190855160)

[3.3) Thư viện hỗ trợ 5](#_Toc190855161)

[CHƯƠNG 2. Trải nghiệm người dùng 6](#_Toc190855162)

[Bài 1) Tương tác người dùng 6](#_Toc190855163)

[1.1) Hình ảnh có thể chọn 6](#_Toc190855164)

[1.2) Các điều khiển nhập liệu 6](#_Toc190855165)

[1.3) Menu và bộ chọn 6](#_Toc190855166)

[1.4) Điều hướng người dùng 6](#_Toc190855167)

[1.5) RecycleView 6](#_Toc190855168)

[Bài 2) Trải nghiệm người dùng thú vị 6](#_Toc190855169)

[2.1) Hình vẽ, định kiểu và chủ đề 6](#_Toc190855170)

[2.2) Thẻ và màu sắc 6](#_Toc190855171)

[2.3) Bố cục thích ứng 6](#_Toc190855172)

[Bài 3) Kiểm thử giao diện người dùng 6](#_Toc190855173)

[3.1) Espresso cho việc kiểm tra UI 6](#_Toc190855174)

[CHƯƠNG 3. Làm việc trong nền 6](#_Toc190855175)

[Bài 1) Các tác vụ nền 6](#_Toc190855176)

[1.1) AsyncTask 6](#_Toc190855177)

[1.2) AsyncTask và AsyncTaskLoader 6](#_Toc190855178)

[1.3) Broadcast receivers 6](#_Toc190855179)

[Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền 6](#_Toc190855180)

[2.1) Thông báo 6](#_Toc190855181)

[2.2) Trình quản lý cảnh báo 6](#_Toc190855182)

[2.3) JobScheduler 6](#_Toc190855183)

[CHƯƠNG 4. Lưu dữ liệu người dùng 7](#_Toc190855184)

[Bài 1) Tùy chọn và cài đặt 7](#_Toc190855185)

[1.1) Shared preferences 7](#_Toc190855186)

[1.2) Cài đặt ứng dụng 7](#_Toc190855187)

[Bài 2) Lưu trữ dữ liệu với Room 7](#_Toc190855188)

[2.1) Room, LiveData và ViewModel 7](#_Toc190855189)

[2.2) Room, LiveData và ViewModel 7](#_Toc190855190)

3.1) Trinfh gowx loi ……………………………………………………………………...

# LÀM QUEN

## Tạo ứng dụng đầu tiên

### 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.

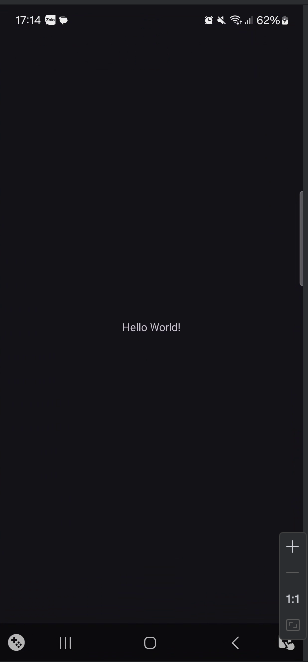
Những gì bạn sẽ làm

* 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 đã cài đặt thành công Android Studio, bạn sẽ tạo ứng dụng Hello World từ một mẫu, một dự án mới. Đó là một ứng dụng đơn giản hiển thị dòng chữ “Hello World” trên màn hình của thiết bị Android ảo hoặc thiết bị vật lý.

Khi mà ứng dụng chạy thành công thì sẽ hiển thị như sau:



**Nhiệm vụ 1: Cài đặt Android Studio**

Android Studio cung cấp một môi trường tích hợp (IDE) hoàn chỉnh bao gồm chỉnh sửa mã nâng cao và một bộ mẫu ứng dụng. Ngoài ra nó còn chứa các công cụ dành cho phát triển, sửa lỗi, kiểm thử và hiệu suất giúp việc phát triển các ứng dụng trở nên nhanh chóng và dễ dàng hơn. Bạn có thể thử nghiệm ứng dụng của mình với một loạt các trình giả lập đã được cấu hình sẵn hoặc trên thiết bị di động của riêng bạn, xây dụng ứng dụng sẵn sàng sử dụng và xuất bản trên cửa hàng Google Play.

**Chú ý:** Android Studio đang liên tục được cải tiến. Để biết thông tin mới nhất về yêu cầu hệ thống và hướng dẫn cài đặt, xem Android Studio.

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

Để khởi động và chạy với Android Studio, đầu tiên kiểm tra các yêu cầu hệ thống để đảm bảo rằng hệ thống đáp ứng chúng. Việc cài đặt tương tự cho tát cả các nền tẳng. Bất kì sự khác biệt nào cũng được ghi chú dưới đây:

1. Điều hướng đến Android developers site và làm theo các hướng dẫn để tải về 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 được chọn để cài đặt.
3. Sau khi hoàn thành việc cài đặt, Setup Wizard sẽ được tải xuống và cài đặt một số thành phần bổ sung bao gồm Android SDK. Hãy kiên nhẫn, điều này có thể mất một thời gian tùy thuộc vào tốc độ mạng của bạn và một số bước có thể bị lặp lại.
4. Khi quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng để tạ dự án đầu tiên của mình.

**Nhiệm vụ 2: Tạo ứng dụng Hello World**

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

**Bước 1: Tạo dự án ứng dụng**

1. Mở Android Studio nếu nó chưa được mở.
2. Trong cửa sổ chính **Welcom to Android Studio**, nhấp chuột vào **Start a new Android Studio project**.
3. Trong cửa sổ **Create Android Studio**, gõ **Hello World** vào dòng **Application name**.
4. Xác nhận rằng **Project location** mặc định là nơi mà bạn muốn lưu trữ ứng dụng Hello World của bạn và các dự án Android Studio khác hoặc thay đổi thành thư mục được chọn bởi bạn.
5. Chấp nhận **android.example.com** mặc định cho tên miền **Company Domanic** hoặc tạo một tên miền công ty riêng biệt.

Nếu bạn không có kế hoạch về phát hành ứng dụng của bạn, bạn có thể chấp nhận mặc định. Hãy lưu ý rằng việc thay đổi gói tên của ứng dụng sau này sẽ tốn thêm công sức.

1. Bỏ chọn các tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấp vào **Next**.
2. Trên màn hình **Taget Android Devices**, hãy đảm bảo rằng **Phone and Tablet** được chọn. Đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt là Minimum SDK; nếu không hãy sử dụng hộp thoại menu để đặt lại.

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

1. Bỏ chọn tùy chọn **Include Instant App support** và tất cả các tùy chọn khác. Sau đó, nhấp vào Next. Nếu dự án của bạn yêu cầu các thành phần bổ sung cho SDK mục tiêu đã chọn, Andoid Studio sẽ tự động cài đặt chúng.
2. Cửa sổ **Add an Activity** xuất hiện. Activity là đơn vị duy nhất, tâp trung những thứ mà người dùng có thể thực hiện. Nó là một thành phần quan trọng của bất kỳ ứng dụng Android nào. Activity thường có một giao diện (layout) liên quan đến nó, định nghĩa cách mà các phần tử giao diện người dùng xuất hiện trên màn hình. Android Studio cung cấp các mẫu Activity để giúp bạn bắt đầu. Đối với dự án Hello World, chọn **Empty Activity** như hình dưới đây và nhấp vào **Next.**

A screenshot of a computer

AI-generated content may be incorrect.

1. Màn hình **Configure Activity** xuất hiện (khác nhau tùy thuộc vào mẫu bạn chọn ở bước trước). Theo mặc định, Activity trống do mẫu cung cấp có tên là **MainActivity**. Bạn có thể thay đổi tên này nếu muốn, nhưng bài học này sử dụng **MainActivity**.
2. Đảm bảo rằng tùy chọn **Generate Layout** file được chọn. Tên bố cục theo mặc định là activity\_main. Bạn có thể thay đổi nếu muốn, nhưng bài học này sử dụng activity\_main.
3. 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.
4. Nhấp vào **Finish**.

A screenshot of a computer

AI-generated content may be incorrect.

Android Studio tạo một thư mục cho các dự án của bạn và xây dụng dự án với Gradle (điều này có thể mất vài phút).

**Mẹo**: Xem trang **Configure your build developer** để biết thông tin chi tiết.

Bạn cũng có thể thấy thông báo “Tip of the day” với các phím tắt và các mẹo hữu ích khác. Nhấp vào **Close** để tắt thông báo.

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

1. Nhấp vào tab activity\_main.xml để xme trình chỉnh sửa bố cục.
2. Nhấp vào tab Design của trình chỉnh sửa bố cục nếu chưa chọn, để hiển thị bản trình bày đồ họa của bố cục, được hiển thị như bên dưới.

A screenshot of a computer

AI-generated content may be incorrect.

1. Nhấp vào MainActivity.java để xem trình chỉnh sửa mã, được hiển thị như bên dưới:

A screenshot of a computer

AI-generated content may be incorrect.

**Bước 2: Khám phá dự án > bảng điều khiển Android**

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

Nếu chưa chọn một dự án nào, nhấp vào tab **Project** trong cột tab dọc ở phía bên trái của cửa sổ Android Studio. Cửa sổ Project sẽ xuất hiện.

Để xem dự án trong phân cấp dự án Android chuẩn, hãy chọn **Android** từ hộp thoại menu bật lên phía trên của bảng Project, như hiển thị bên dưới.

A screenshot of a computer

AI-generated content may be incorrect.

**Chú ý**: Trong chương này và các chương khác đề cập đến khung Project, khi cài đặt là **Android** như **Project > bảng Android**.

**Bước 3: Khám phá thu mục Gradle Scripts**

Hệ thống xây dựng Gradle trong Android Studio giúp dễ dàng bao gồm các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác vào quá trình xây dựng của bạn dưới dạng phụ thuộc.

Khi bạn lần đầu tiên tạo một dự án ứng dụng, **Project > bảng Android** sẽ xuất hiện với thư mục **Gradle Scripts** được mở rộng như hình dưới đây:

A screenshot of a computer

AI-generated content may be incorrect.

Thực hiện theo từng bước để khám phá hệ thống Gradle:

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

Thư mục này chứa tất cả các tệp cần thiết cho xây dựng hệ thống.

1. Tìm tệp **build.gradle (Dự án: 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 duy nhất một tệp xây dựng Gradle cấp cao. Hầu hết thời gian, bạn sẽ không cần phải thực hiện bất kỳ thay đổi nào đối với tệp này nhưng vẫn rất hữu ích để hiểu nội dung của nó.

Theo mặc định, tệp xây dựng cấp cao sử dụng khổi buildscript để định nghĩa các kho lưu trữ và phụ thuộc Gradle chung cho tất cả các mô-đun trong dự án. Khi các yếu tố phụ thuộc của bạn là một thứ gì đó khác ngoài thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm kiếm các tệp trong các kh lưu trữ trực tuyến được chỉ định trong khối lưu trữ của tệp. Theo mặc định, các dự án Android Studio mới khai báo Jcenter và Google (bao gồm cả Google Maven responsitory) là các vị trí kho lưu trữ:

|  |
| --- |
| plugins **{** alias(libs.plugins.android.application) apply false **}** |

1. Tìm tệp build.gradle(Module:app).

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

Tệp này thường là tệp cần chỉnh sửa khi thay đổi các cấu hình cấp ứng dụng, chẳng hạn như khai báo các yếu tố trong phần denpendencies. Bạn có thể khai báo một thư viện phụ thuộc bằng nhiều cách cấu hình 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 implementation fileTree(dir: ‘libs’, include: ['\*.jar']) thêm một phụ thuộc cho tất cả các tệp “.jar” bên trong thư mục libs.

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

|  |
| --- |
| plugins **{** alias(libs.plugins.android.application) **}** android **{** namespace 'com.example.helloworld'  compileSdk 35   defaultConfig **{** applicationId "com.example.helloworld"  minSdk 24  targetSdk 35  versionCode 1  versionName "1.0"   testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  **}** buildTypes **{** release **{** minifyEnabled false  proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  **}  }** compileOptions **{** sourceCompatibility JavaVersion.*VERSION\_11* targetCompatibility JavaVersion.*VERSION\_11* **} }** dependencies **{** implementation libs.appcompat  implementation libs.material  implementation libs.activity  implementation libs.constraintlayout  testImplementation libs.junit  androidTestImplementation libs.ext.junit  androidTestImplementation libs.espresso.core **}** |

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

**Bước 4: Khám phá thư mục app và thư mục res**

Tất cả mã và tài nguyên của 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.

A screenshot of a computer

AI-generated content may be incorrect.

Thư mục **java** chứa các tệp lớp Java được tổ chức trong ba thư mục con như hình minh họa trong hình trên. Thư mục **com.example.android.helloworld** (hoặc tên miền đã chỉ định) chứa tất cả các tệp của một gói ứng dụng. Hai thư mục còn lại được sử dụng cho mục đích kiểm thử và sẽ được mô tả trong một bài học khác. Đối với ứng dụng Hello World, chỉ có một goid duy nhất và nó chứa tệp MainActivity.java. Đây là Activity (màn hình) đầu tiên mà người dùng nhìn thấy, đồng thời khởi tạo các tài nguyên chung cho ứng dụng, thường được đặt tên là **MainActivity** (phần mở rộng tệp bị ẩn trong bảng **Project > Android**).

1. Mở rộng thư mục res và thư mục layout, và nhấp đúp chuột vào tệp activity\_main.xml để mở tệp đó trong trình chỉnh sửa layout.

A screenshot of a computer

AI-generated content may be incorrect.

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

**Bước 5: Khám phá thư mục mainfests**

Thư mục manifests chứ các file 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 **AndroidManifests.xml**.

Tệp AndroidManifests.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. Trogn các bài học khác của khóa học, bạn sẽ sửa đổi tệp này để thêm các tính năng và quyền tính năng. Để đọc phần giới thiệu, hãy xem App Manifest Overview.

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

Trong nhiệm vụ này, bạn sẽ sử dụng Android Virtual Device (AVD) manager để tạo một thiết bị ảo (cũng có thể hiểu 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 Android Emulator có các yêu cầu bổ sung ngoài các yêu cầu hệ thống cơ bản đối với Android Studio.

Sử dụng ADV Manager, bạn 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 kiểm tra 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.

**Bước 1: Tạo một thiết bị Android ảo (AVD)**

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

1. Trong Android Studio, chọn **Tools > Android > AVD Manager** hoặc nhấp vào biểu tượng AVD Manager trên thanh công cụ. Màn hình Your Virtual Devices 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.

A screenshot of a computer

AI-generated content may be incorrect.

1. Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** sẽ xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình trước. 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**).

A screenshot of a computer

AI-generated content may be incorrect.

1. Chọn một thiết bị như **Nexus 5x** hoặc **Pixel XL** và nhấp vào **Next**. Màn hình **System Image** xuất hiện.
2. Nhấp vào tab **Recommended** nếu chưa chọn và chọn phiên bản hệ thống Android nào để chạy trên thiết bị ảo (như **VanillaleCream**).

A screenshot of a computer

AI-generated content may be incorrect.

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

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

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

**Bước 2: Chạy ứng dụng trên thiết bị ảo**

Trong nhiệm vụ này, bạn sẽ hoàn tất việc chạy ứng dụng Hello World của bạn.

1. Trong Android Studio, chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run** trên thanh công cụ.
2. Cửa sổ **Select Deployment Target**, bên dưới **Available Virtual Devices**, chọn thiết bị ảo bạn vừa tạo và nhấp **OK**.

Trình giả lập khởi động và khởi động giống như một thiết bị vật lý. Tùy thuộc vào tốc độ máy tính của bạn, việc 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 đó. Bạn sẽ thấy ứng dụng Hello World như trong hình sau.

**Mẹo**: Khi thử nghiệm trên thiết bị ảo, bạn nên khởi động thiết bị một lần, ngay khi bắt đầu phiên làm việc. Bạn không nên đóng thiết bị cho đến khi hoàn tất thử nghiệm ứng dụng, để ứng dụng không phải trải qua quá trình khởi động thiết bị một lần nữa. Để đóng thiết bị ảo, hãy nhấp vào nút **X** ở đầu trình giả lập, chọn **Quit** từ menu hoặc nhấn **Control-Q** trong Windows hoặc Command-Q trong macOS.

**Nhiệm vụ 4: (Không bắt buộc) Sử dụng thiết bị vật lý**

Trong nhiệm vụ cuối này, bạn sẽ chạy ứng dụng của bạn trên một thiết bị di động như điện thoại hoặc máy tính bảng. Bạn nên luôn luôn thử nghiệm ứng dụng của bạn trên cả thiết bị ảo và thiết bị vật lý.

Những điều bạn cần có:

* Một thiết bị Android như điện thoại hoặc máy tính bảng.
* Một dây cáp truyền dữ liệu kết nối thiết bị Android của bạn với máy tính của bạn thông qua cổng USB.
* Nếu bạn đang sử dụng Hệ thống Linux hoặc Windows, bạn có thể cần thực hiện thêm các bước để chạy trên phần cúng thiết bị. Kiểm tra tài liệu **Using Hardware Devices**. Bạn cũng có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Đối với trình điều khiển USB dựa trên Windows, hãy xem **OEM USB Drivers**.

**Bước 1: Bật chế độ USB debugging**

Để Android Studio kết nối với thiết bị của bạn, bạn phải bật chế độ USB Debugging triên thiết bị Android của bạn. Tùy chọn này được kích hoạt trong phần **Developer options** trên thiết bị của bạn.

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

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

**Bước 2: Chạy ứng dụng của bạn trên một thiết bị**

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

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

A screen shot of a phone

AI-generated content may be incorrect.

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

**Khắc phục sự cố:**

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

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

Nếu máy tính vẫn không nhận diện thiết bị hoặc báo “unauthorized”, hãy làm theo các bước sau:

1. Rút thiết bị ra.
2. Trên thiết bị, mở **Developer Options** trong ứng dụng **Cài đặt (Settings)**.
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, hãy chấp nhận.

Bạn có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Xem thêm tại Using Hardware Devices documentation.

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

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

**Bước 1: Thay đổi phiên bản tối thiểu SDK cho ứng dụng**

Thực hiện theo các bước sau:

1. Mở rộng thư mục **Gradle** **Scripts** nếu nó chưa được mở, sau đó nhấp đúp vào tệp **build.gradle (Module:app).**  
   Nội dung của tệp sẽ hiển thị trong trình chỉnh sửa mã.

A screenshot of a computer screen

AI-generated content may be incorrect.

1. Trong khối defaultConfig, thay đổi giá trị của minSdkVersion thành 24, như minh họa bên dưới (nếu giá trị ban đầu không được đặt là 24).

A screenshot of a computer

AI-generated content may be incorrect.

Trình chỉnh sửa mã hiển thị một thanh thông báo ở phía trên cùng với liên kết **Sync Now(Open(Ctrl+Alt+Shift+S))**.

**Bước 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 bạn đồng bộ (sync) các tệp của dự án để có thể nhập các thay đổi cấu hình build và chạy một số kiểm tra nhằm đảm bảo rằng cấu hình sẽ không gây ra lỗi khi biên dịch.

Để đồng bộ tệp dự án, hãy nhấp vào **Sync** **Now** trên 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 thanh công cụ.

Khi quá trình đồng bộ 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 tham khảo tài liệu Build System Overview và Configuring Gradle Builds.

**Nhiệm vụ 6: Thêm câu lệnh log vào ứng dụng của bạn**

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh Log vào ứng dụng của mình, giúp hiển thị thông báo trong bảng **Logcat**.

Các thông báo Log là một công cụ gỡ lỗi mạnh mẽ, cho phép bạn kiểm tra giá trị, theo dõi luồng thực thi và báo cáo các ngoại lệ.

**Bước 1: Xem bảng Logcat**

Để mở bảng **Logcat**, nhấp vào tab **Logcat** ở phía dưới cửa sổ Android Studio, như hình minh họa dưới đây:

A screenshot of a computer program

AI-generated content may be incorrect.

Trong hình trên:

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

Bước 2: Thêm câu lệnh log vào ứng dụng của bạn

Các câu lệnh Log trong mã ứng dụng sẽ hiển thị thông báo trong bảng Logcat. Ví dụ:

|  |
| --- |
| Log.d("MainActivity", "Hello World"); |

Các thành phần của thông báo log:

* Log: Lớp Log dùng để gửi thông báo log đến bảng Logcat
* d: Mức log **Debug** để lọc và hiển thị thông báo trong bảng Logcat. Các ức log khác bao gồm: **e** cho **Error**, **w** cho **Warn** và **i** cho **Info**.
* “MainActivity”: Đối số đầu tiên là một tag (nhãn), có thể được sử dụng để lọc thông báo trong bảng Logcat. Thông thường, tag này là tên của Activity nơi thông báo được ghi lại. Tuy nhiên, bạn có thể đặt bất kỳ giá trị nào hữu ích cho quá trình gỡ lỗi.

Theo quy ước, các tag log 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.

Thực hiện theo các bước sau:

1. Mở ứng dụng Hello World trong Android Studio và mở tệp MainActivity.
2. Để tự động thêm các thư viện import cần thiết vào dự án (chẳng hạn như android.util.Log để sử dụng Log), thực hiện: với Windows: Vào **File > Settings** và với macOS: Vào **Android Studio > Preferences**.
3. Chọn **Editor > General > Auto Import**. Tích chọn tất cả các ô và đặt **Insert imports on paste** thành **All**.
4. Nhấn **Apply**, sau đó nhấn **OK**.

A screenshot of a computer

AI-generated content may be incorrect.

1. Trong phương thức onCreate() của MainActivity, thêm câu lệnh sau:

|  |
| --- |
| Log.d("MainActivity", "Hello World"); |

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

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

1. Nếu bảng Logcat chưa mở, hãy nhấp vào tab **Logcat** ở phía dưới Android Studio để mở.
2. Kiểm tra xem tên mục tiêu (target) và tên gói (package name) của ứng dụng có chính xác không.
3. Thay đổi mức **Log** trong bảng **Logcat** thành **Debug** (hoặc giữ nguyên **Verbose** nếu có ít thông báo log).
4. Chạy ứng dụng của bạn.

Thông báo sau sẽ xuất hiện trong bảng Logcat:



**Thử thách lập trình**

**Lưu ý**: Tất cả các thử thách lập trình 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 xong 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. Thay đổi lời chào “Hello World” thành Happy Birthday to “ và thêm tên của một người vừa có sinh nhật gần đây.
3. (Tùy chọn) Chụp ảnh màn hình ứng dụng đã hoàn thành và gửi email cho ai đó mà bạn quên chúc mừng sinh nhật.
4. Một cách sử dụng phổ biến của lớp Log là ghi lại các ngoại lệ (Java exceptions) khi chúng xảy ra trong chương trình của bạn. Bạn có thể sử dụng một số phương thức hữu ích như Log.e() để thực hiện điều này. Khám phá các phương thức mà bạn có thể sử dụng để đính kèm một ngoại lệ (exception) vào thông báo Log. Sau đó, viết mã trong ứng dụng của bạn để kích hoạt và ghi lại một ngoại lệ.

**Tóm tắt**

* Để cài đặt Android Studio, truy cập trang Android Studio và làm theo 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 Minimum SDK.
* Để xem cấu trúc phân cấp của ứng dụng trong Project pane, nhấp vào tab **Project** 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 nguồn và tài nguyên của ứng dụng được đặt trong thư mục app và res. Thư mục java chứa các Activity, bài kiểm tra và các thành phần khác được viết bằng mã nguồn Java. Thư mục res chứa tài nguyên như bố cục (layouts), chuỗi ký tự (strings), và hình ảnh (images).
* Chỉnh sửa tệp AndroidManifest.xml để thêm các thành phần tính năng và quyền (permissions) cho ứng dụng Android. Tất cả các thành phần của ứng dụng, chẳng hạn như nhiều Activity, phải được khai báo trong tệp XML này.
* Sử dụng Android Virtual Device (AVD) Manager để tạo một thiết bị ảo (emulator) chạy ứng dụng của bạn.
* Thêm các câu lệnh Log vào ứng dụng để hiển thị thông báo trong bảng Logcat như một công cụ gỡ lỗi cơ bản.
* Để chạy ứng dụng trên thiết bị Android thực tế bằng Android Studio, hãy bật USB Debugging trên thiết bị: Mở **Cài đặt > Giới thiệu điện thoại (Settings > About phone)**, sau đó nhấn vào **Số bản dựng (Build number)** bảy lần. Quay lại màn hình trước (**Cài đặt**), nhấn vào **Tùy chọn nhà phát triển (Developer options),** sau đó bật **Gỡ lỗi USB (USB Debugging)**.

**Các khái niệm liên quan**

Tài liệu về các khái niệm liên quan có trong 1.0: Introduction to Android và 1.1 Your first Android app.

**Tìm hiểu thêm**

Tài liệu về Android Studio:

Khác:

**Bài tập về nhà**

**Xây dựng và chạy một ứng dụng:**

* Tạo một dự án Android mới từ mẫu Empty Template.
* Thêm các câu lệnh log với các cấp độ khác nhau trong phương thức onCreate() của MainActivity.
* Tạo một thiết bị giả lập (emulator) với phiên bản Android tùy chọn và chạy ứng dụng.
* Sử dụng bộ lọc trong **Logcat** để tìm các câu lệnh log của bạn và điều chỉnh mức hiển thị chỉ bao gồm debug hoặc error.

**Trả lời các câu hỏi**

Câu 1: Tên của tệp tin layout cho main activity là gì ?

* MainActivity.java
* AndroidManifest.xml
* activity\_main.xml
* build.gradle

Câu 2: Tên của tài nguyên chuỗi (string resource) xác định tên của ứng dụng là gì ?

* app\_name
* xmlns:app
* android:name
* applicationId

Câu 3: Bạn sử dụng công cụ nào để tạo một trình giả lập (emulator) mới ?

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

Câu 4: Giả sử ứng dụng của bạn có câu lệnh ghi log sau:

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

Bạn sẽ thấy dòng log “MainActivity layout is complete” trong Logcat nếu mức Log được đặt thành tùy chọn nào sau đây? (Gợi ý: Có thể có nhiều đáp án đúng.)

* Verbose
* Debug
* Info
* Warn
* Error
* Assert

**Nộp ứng dụng của bạn để chấm điểm**

Hãy kiểm tra để đảm bảo ứng dụng có các yêu cầu sau:

* Một Activity hiển thị “Hello World” trên màn hình.
* Câu lệnh log trong phương thức onCreate() của MainActivity.
* Mức Log trong **Logcat** chỉ hiển thị lệnh debug hoặc error.

### Giao diện người dùng tương tác đầu tiên

**Giới thiệu**

Giao diện người dùng (UI) xuất hiện trên màn hình thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là *views* – mọi phần tử trên màn hình đầu là một View. Lớp View đại diện cho khối xây dựng cơ bản của tất cả các thành phần UI và là lớp cơ sở cho các lớp cung cấp các thành phần UI tương tác như button, checkbox và trường nhập văn bản. Các lớp con View thường được sử dụng, được mô tả trong nhiều 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 phần tử 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 phần tử View khác và định vị chúng.

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

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị một bố cục được định nghĩa trong tệp layout activity\_main.xml, trong đó bao gồm một TextView hiển thị văn bản "Hello World".

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

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên—một ứng dụng cho phép người dùng tương tác. Bạn sẽ tạo một ứng dụng sử dụng mẫu Empty Activity. Ngoài ra, bạn sẽ học cách sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế bố cục và chỉnh sửa bố 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**

Bạn cần làm quen với:

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

**Những gì bạn sẽ học**

* Cách tạo một ứng dụng có hành vi tương tác.
* Cách sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế bố cục)
* Cách chỉnh sửa bố cục trong XML.
* Nhiều thuật ngữ mới. Hãy xem phần Vocabulary words and concepts glossary để tìm hiểu các định nghĩa một cách dễ hiểu.

**Những gì bạn sẽ làm**

* Tạo một ứng dụng và thêm hai phần tử Button cùng một TextView vào bố cục.
* Điều chỉnh từng phần tử trong ConstraintLayout để ràng buộc chúng với lề và các phần tử khác.
* Thay đổi thuộc tính của các phần tử UI.
* Chỉnh sửa bố cục của ứng dụng trong XML.
* Trích xuất chuỗi văn bản được mã cứng thành tài nguyên chuỗi.
* Triển khai phương thức xử lý sự kiện khi nhấn nút để hiển thị thông báo trên màn hình khi người dùng chạm vào từng Button.

**Tổng quan về ứng dụng**

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

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

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

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

**Bước 1: Tạo dự án Android Studio**

1. Chạy Android Studio và tạo một dự án mới với các thông số sau:

|  |  |
| --- | --- |
| Attribute | Value |
| 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 |

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

**Bước 2: Khám phá trình chỉnh sửa bố cục**

Android Studio cung cấp trình chỉnh sửa bố cục (layout editor) để xây dựng nhanh chóng bố cục giao diện người dùng (UI) của ứng dụng. Công cụ này cho phép bạn kéo các phần tử vào chế độ xem thiết kế trực quan và bản thiết kế, định vị chúng trong bố cục, thêm ràng buộc (constraints) và thiết lập thuộc tính.

Constraints xác định vị trí của một phần tử UI trong bố cục. Một ràng buộc đại diện cho một kết nối hoặc căn chỉnh với một View khác, bố cục cha, hoặc một đường hướng dẫn vô hình.

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

1. Trong bảng điều hướng **Project > Android**, điều hướng đến thư mục **app > res > layout**, sau đó nhấp đúp vào tệp **activity\_main.xml** để mở, nếu nó chưa được mở.

2. Nhấp vào tab Design nếu nó chưa được chọn. Bạn sử dụng tab Design để thao tác các phần tử và bố cục, còn tab Text để chỉnh sửa mã XML của bố cục.

3. Ngăn Palettes hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục của ứng dụng.

4. Ngăn Component Tree hiển thị hệ thống phân cấp của các phần tử UI. Các phần tử View được tổ chức theo một cây phân cấp với quan hệ cha - con, trong đó phần tử con kế thừa thuộc tính từ phần tử cha. Trong hình trên, TextView là phần tử con của ConstraintLayout. Bạn sẽ tìm hiểu thêm về các phần tử này trong bài học này.

5. Các ngăn thiết kế và bản thiết kế của trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình trên, bố cục chỉ chứa một phần tử: một TextView hiển thị dòng chữ "Hello World".

6. Tab Attributes hiển thị ngăn Attributes để thiết lập các thuộc tính cho một phần tử UI.

**Mẹo**: Xem tài liệu 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 tài liệu đầy đủ về Android Studio.

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

Trong nhiệm vụ này, bạn sẽ tạo bố cục giao diện người dùng cho ứng dụng HelloToast bằng cách sử dụng các tính năng của ConstraintLayout trong trình chỉnh sửa bố cục. Bạn có thể tạo ràng buộc (constraints) theo cách thủ công như hướng dẫn bên dưới hoặc tự động bằng công cụ Autoconnect.

**Bước 1: Kiểm tra ràng buộc của phần tử**

Thực hiện các bước sau:

1. Mở tệp activity\_main.xml từ bảng điều hướng **Project > Android** nếu nó chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào nó.

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

1. Công cụ **Autoconnect** cũng nằm trên thanh công cụ và được bật theo mặc định. Đảm bảo rằng công cụ này không bị tắt.
2. Nhấp vào nút Zoom In để phóng to chế độ xem thiết kế và bản thiết kế để quan sát rõ hơn.
3. Chọn **TextView** trong ngăn Component Tree. Phần tử TextView có nội dung "Hello World" sẽ được tô sáng trong các ngăn thiết kế và bản thiết kế, đồng thời các ràng buộc của phần tử sẽ hiển thị.
4. Làm theo hướng dẫn trong hình động bên dưới. Nhấp vào tay cầm hình tròn ở bên phải của TextView để xóa ràng buộc ngang đang gắn phần tử này vào cạnh phải của bố cục. Khi ràng buộc này bị xóa, TextView sẽ nhảy sang bên trái vì nó không còn bị ràng buộc với cạnh phải nữa. Để thêm lại ràng buộc ngang, nhấp vào cùng tay cầm đó và kéo một đường đến cạnh phải của bố cục.

Trong ngăn bản thiết kế (blueprint) hoặc thiết kế (design), các tay cầm sau xuất hiện trên phần tử TextView:

* **Constraint handle**: Để tạo một ràng buộc như trong hình động trên, hãy nhấp vào tay cầm ràng buộc (Constraint handle), được hiển thị dưới dạng một vòng tròn ở cạnh của phần tử. Sau đó, kéo tay cầm đến một tay cầm ràng buộc khác hoặc đến ranh giới của phần tử cha. Một đường ziczac sẽ đại diện cho ràng buộc.
* **Resizing handle** (Tay cầm thay đổi kích thước): Để thay đổi kích thước của phần tử, hãy kéo các tay cầm hình vuông. Khi kéo, tay cầm sẽ thay đổi thành một góc xiên.

**Bước 2: Thêm một Button vào bố cục**

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

Thực hiện các bước sau để thêm một Button:

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

**Bước 3: Thêm một Button thứ hai vào bố cục**

1. Kéo thêm một Button từ bảng điều hướng Palette vào giữa bố cục như hình minh họa hình động bên dưới. Autoconnect có thể tự động tạo ràng buộc ngang cho bạn. Nếu không, bạn có thể tự kéo ràng buộc ngang.
2. Kéo một ràng buộc dọc từ Button đến cạnh dưới của bố cục (tham khảo hình bên dưới).

Bạn có thể xóa các ràng buộc khỏi một phẩn tử bằng cách chọn phần tử đó và di chuyển con trỏ chuột qua để hiển thị nút **Clear** **Constraints**. Nhấp vào nút này để xóa tất cả ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc cụ thể, hãy nhấp vào tay cầm thiết lập ràng buộc đó.

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

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

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

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

**Bước 1: Thay đổi kích thước Button**

Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước **(Resizing handle)** ở cả bốn góc của một View, giúp bạn có thể thay đổi kích thước View một cách nhanh chóng. Bạn có thể kéo các tay cầm ở mỗi góc của View để thay đổi kích thước của nó, nhưng làm như vậy sẽ đặt kích thước chiều rộng và chiều cao dưới dạng giá trị cố định (hardcoded). Hạn chế sử dụng kích thước cố định cho hầu hết các phần tử View, vì các kích thước này không thể thích ứng với nội dung và kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn **Attributes** ở bên phải của trình chỉnh sửa bố cục để chọn một chế độ kích thước không sử dụng giá trị cố định. Ngăn **Attributes** bao gồm một bảng kích thước hình vuông, được gọi là *view inspector*, nằm ở phía trên.Các ký hiệu bên trong hình vuông này đại diện cho các cài đặt chiều cao và chiều rộng như sau:

Trong hình trên:

1. **Height control** (Điều khiển chiều cao): Điều khiển này xác định thuộc tính layout\_height và xuất hiện dưới dạng hai đoạn trên các cạnh trên và dưới của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành wrap\_content, có nghĩa là View sẽ mở rộng theo chiều dọc khi cần thiết để phù hợp với nội dung của nó. Số "8" biểu thị một lề tiêu chuẩn được đặt thành 8dp.
2. **Width control** (Điều khiển chiều rộng): Điều khiển này xác định thuộc tính layout\_width và xuất hiện dưới dạng hai đoạn trên các cạnh trái và phải của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành wrap\_content, điều này có nghĩa là View sẽ mở rộng theo chiều ngang khi cần thiết để phù hợp với nội dung của nó, tối đa đến một lề 8dp.
3. Nút đóng ngăn **Attributes**: Nhấp vào để đóng ngăn

Thực hiện các bước sau:

1. Chọn Button trên cùng trong ngăn **Component Tree**.
2. Nhấp vào tab **Attributes** ở phía 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 nó thành **Fixed** với các đường thẳng và lần nhấp thứ hai thay đổi nó thành **Match Constraints** với các lò xo, như minh họa trong hình động bên dưới.

Do thay đổi kết quả điều khiển chiều rộng, thuộc tính layout\_width trong ngăn **Attributes** hiển thị giá trị match\_constraint, và phần tử Button mở rộng theo chiều ngang để lấp đầy không gian giữa cạnh trái và cạnh phải của bố cục.

1. Chọn Button thứ hai và thực hiện các thay đổi tương tự đối với layout\_width như ở bước trước, như minh họa trong hình dưới đây.

Như đã thấy trong các bước trước, các thuộc tính layout\_width và layout\_height trong ngăn **Attributes** thay đổi khi bạn điều chỉnh các điều khiển chiều rộng và chiều cao trong inspector. Các thuộc tính này có thể nhận một trong ba giá trị trong bố cục ConstraintLayout:

* Giá trị match\_constraint mở rộng phần tử View để lấp đầy không gian của parent theo chiều rộng (width) hoặc chiều cao tối đa (heigh-up) đến một lề nếu có lề được đặt. Trong trường hợp này, parent là ConstraintLayout. Bạn sẽ tìm hiểu thêm về ConstraintLayout trong nhiệm vụ tiếp theo.
* Giá trị wrap\_content thu nhỏ kích thước của phần tử View sao cho vừa đủ để bao bọc nội dung của nó. Nếu không có nội dung, phần tử View sẽ trở nên vô hình.
* Kích thước cố định để chỉ định một kích thước cố định có thể điều chỉnh theo kích thước màn hình của thiết bị, sử dụng một số cố định với đơn vị density-independent pixels (dp units). Ví dụ, 16dp có nghĩa là 16 pixel độc lập với mật độ màn hình.

**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 0 vì không có kích thước cố định được thiết lập. Cài đặt này tương đương với match\_constraint—phần tử View có thể mở rộng tối đa để đáp ứng các ràng buộc và thiết lập lề.

**Bước 2: Thay đổi các thuộc tính của Button**

Để xác định duy nhất mỗi View trong bố cục của một Activity, mỗi View hoặc lớp con của View (chẳng hạn như Button) cần có một ID duy nhất. Ngoài ra, để có thể sử dụng được, 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 **Attributes** 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 android:id, background, textColor, và text.

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

1. Sau khi chọn Button đầu tiên, chỉnh sửa trường ID ở đầu ngăn **Attributes** thành **button\_toast** cho thuộc tính android:id, thuộc tính này dùng để xác định phần tử trong bố cục.

2. Đặt thuộc tính background thành **@color/colorPrimary**. (Khi bạn nhập **@c**, các tùy chọn sẽ xuất hiện để bạn dễ dàng chọn.)

3. Đặt thuộc tính textColor thành **@android:color/white**.

4. Chỉnh sửa thuộc tính text thành **Toast**.

5.Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng **button\_count** làm ID, **Count** cho thuộc tính text, và cùng màu nền cũng như màu chữ như các bước trước đó.

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

**Nhiệm vụ 4: Thêm một TextEdit và thiết lập các ràng buộc của nó**

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

**Bước 1: Thêm một TextView và các ràng buộc**

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

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

**Bước 2: Thiết lập các thuộc tính của TextView**

Khi TextView đang được chọn, mở bảng **Attributes** nếu nó chưa được mở. Thiết lập các thuộc tính cho TextView như minh họa trong hình động bên dưới. Những thuộc tính mà bạn chưa gặp trước đây sẽ được giải thích sau hình minh họa:

1. Đặt ID thành **show\_count**.

2. Đặt text thành **0**.

3. Đặt textSize thành **160sp**.

4. Đặt textStyle thành **B** (đậm) và textAlignment thành ALIGNCENTER (căn giữa đoạn văn bản).

5. Thay đổi chế độ kích thước ngang và dọc (layout\_width và layout\_height) thành **match\_contraint**.

6. Đặt textColor thành **@color/colorPrimary**.

7. Cuộn xuống bảng và nhấp vào **View all attributes**, tiếp tục cuộn xuống trang thuộc tính thứ hai đến background, sau đó nhập **#FFF00** để chọn một tông màu vàng.

8. Cuộn xuống đến gravity, mở rộng gravity và chọn **center\_ver** (để căn giữa theo chiều dọc).

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

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

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

Bố cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử giao diện người dùng trong Component Tree. Di chuột qua các dấu chấm than này để xem thông báo cảnh báo, như hình minh họa bên dưới. Cả ba phần tử đều có cùng một cảnh báo: chuỗi được mã hóa cứng (hardcoded strings) nên sử dụng tài nguyên.

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

**Bước 1: Mở mã XML của bố cục**

Đối với nhiệm vụ này, mở tệp activity\_main.xml nếu nó chưa được mở, rồi nhấp vào tab **Text** ở dưới cùng của trình chỉnh sửa bố cục.

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

**Bước 2: Trích xuất tài nguyên chuỗi (string resources)**

Thay vì mã hóa cứng các chuỗi, một phương pháp tốt nhất là sử dụng tài nguyên chuỗi, giúp đại diện cho các chuỗi đó. Việc lưu trữ chuỗi trong một tệp riêng biệt giúp dễ dàng quản lý hơn, đặc biệt 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 một tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhấp một lần vào từ "Toast" (cảnh báo đầu tiên được đánh dấu).
2. **Nhấn Alt + Enter** trên Windows hoặc **Option + Enter** trên macOS và chọn **Extract string resource** từ hộp thoại menu bật lên.
3. Nhập **button\_label\_toast** cho **Resource name**.
4. Nhấp **OK**. Một tài nguyên chuỗi sẽ được tạo trong tệp values/res/strings.xml, và chuỗi trong mã của bạn sẽ được thay thế bằng một tham chiếu đến tài nguyên:

@string/button\_label\_toast

1. Trích xuất các chuỗi còn lại: button\_label\_count cho "Count", và count\_initial\_value cho "0".
2. Trong bảng **Project > Android**, mở rộng thư mục **values** bên trong **res**, sau đó nhấp đúp vào **strings.xml** để xem các tài nguyên chuỗi của bạn trong tệp strings.xml.
3. Bạn cần một chuỗi khác để sử dụng trong một nhiệm vụ tiếp theo nhằm hiển thị thông báo. Thêm vào tệp strings.xml một tài nguyên chuỗi mới có tên toast\_message cho cụm từ "Hello Toast!":

Mẹo: Các tài nguyên chuỗi bao gồm tên ứng dụng, tên này sẽ xuất hiện trên thanh ứng dụng ở đầu màn hình nếu bạn bắt đầu dự án ứng dụng bằng Empty Template. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa app\_name resource.

**Nhiệm vụ 6: Thêm onClick handlers cho các nút**

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

**Bước 1: Thêm thuộc tính onClick và trình xử lý sự kiện cho mỗi Button**

Một click handler là một 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 của **Attributes** trong tab **Design**. Bạn cũng có thể chỉ định tên của phương thức xử lý trong XML editor bằng cách thêm thuộc tính android:onClick vào Button. Bạn sẽ sử dụng cách thứ hai vì hiện tại bạn chưa tạo các phương thức xử lý, và XML editor cung cấp một cách tự động để tạo các phương thức này.

1. Khi trình chỉnh sửa XML đang mở (tab Text), tìm Button có thuộc tính android:id

được đặt thành button\_toast.

1. Thêm thuộc tính android:onClick vào cuối phần tử button\_toast, sau thuộc tính cuối cùng và trước ký hiệu kết thúc />.
2. Nhấp vào biểu tượng bóng đèn màu đỏ xuất hiện bên cạnh thuộc tính. Chọn **Create click handler**, chọn **MainActivity**, rồi nhấp **OK**.

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

Hành động này sẽ tạo một phương thức khung (placeholder) cho phương thức showToast() trong MainActivity, như được hiển thị ở cuối các bước này.

1. Lặp lại hai bước cuối với nút button\_count: Thêm thuộc tính android:onClick vào cuối và thêm trình xử lý sự kiện nhấp (click handler).

Mã XML cho các phần tử giao diện người dùng trong ConstraintLayout hiện trông như sau:

1. Nếu MainActivity.java chưa được mở, hãy mở rộng thư mục **java** trong Project > Android view, sau đó mở rộng **com.example.android.hellotoast** và nhấp đúp vào **MainActivity**. Trình chỉnh sửa mã sẽ xuất hiện với nội dung mã trong MainActivity.

**Bước 2: Chỉnh sửa trình xử lý sự kiện của Toast Button**

Bây giờ, bạn sẽ chỉnh sửa phương thức showToast() - trình xử lý sự kiện khi nhấn nút **Toast** trong MainActivity - để hiển thị một thông báo. Toast cung cấp một cách hiển thị thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ chiếm không gian đủ để hiển thị nội dung tin nhắn, trong khi activity hiện tại vẫn hiển thị và tương tác được. Toast hữu ích để kiểm tra tính tương tác của ứng dụng—bạn có thể thêm một thông báo Toast để hiển thị kết quả khi người dùng nhấn vào một Button hoặc thực hiện một hành động.

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

1. Xác định vị trí phương thức showToast() vừa được tạo.
2. Để tạo một thể hiện của Toast, hãy gọi phương thức tạo makeText() trên lớp Toast.

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

1. Cung cấp context của Activity trong ứng dụng. Vì Toast hiển thị trên giao diện Activity, hệ thống cần thông tin về Activity hiện tại. Khi bạn đang ở trong Activity mà bạn cần context, hãy sử dụng this như một cách viết tắt.
2. Cung cấp thông điệp cần hiển thị, chẳng hạn như một string resource (chuỗi toast\_message mà bạn đã tạo ở bước trước). String resource toast\_message được xác định bằng R.string.toast\_message.
3. Cung cấp thời gian hiển thị. Ví dụ, Toast.LENGTH\_SHORT sẽ hiển thị Toast trong một khoảng thời gian ngắn.

Thời gian hiển thị của một Toast có thể là Toast.LENGTH\_LONG hoặc Toast.LENGTH\_SHORT. Thời gian thực tế khoảng 3,5 giây đối với Toast dài và 2 giây đối với Toast ngắn.

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

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

**Bước 3: Chỉnh sửa trình xử lý nút đếm (Count Button handler)**

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

Mã của trình xử lý sự kiện cần:

* Theo dõi giá trị của số đếm khi nó thay đổi.
* Gửi số đếm đã cập nhật đến TextView để hiển thị.

Thực hiện các bước sau để chỉnh sửa trình xử lý sự kiện của nút Count:

1. Xác định vị trí phương thức countUp() vừa được tạo.
2. Để theo dõi số đếm, bạn cần một biến thành viên riêng (private). Mỗi lần nhấn nút Count, giá trị của biến này sẽ tăng lên.

Hãy nhập đoạn mã sau, đoạn này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn màu đỏ:

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

1. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'mCount'** từ menu bật lên. Thao tác này sẽ tạo một biến thành viên private ở đầu MainActivity, và Android Studio sẽ mặc định biến này là kiểu số nguyên (int):
2. Thay đổi câu lệnh khai báo biến thành viên private để khởi tạo biến với giá trị bằng 0:
3. Cùng với biến trên, bạn cũng cần một biến thành viên private để tham chiếu đến TextView có ID là show\_count, biến này sẽ được sử dụng trong trình xử lý sự kiện nhấn nút. Đặt tên biến là mShowCount.
4. Bây giờ, khi đã có mShowCount, bạn có thể lấy tham chiếu đến TextView bằng ID đã đặt trong tệp bố cục (layout file). Để chỉ lấy tham chiếu này một lần, hãy khai báo nó trong phương thức onCreate(). Như bạn sẽ tìm hiểu trong bài học khác, phương thức onCreate() được sử dụng để inflate layout, nghĩa là thiết lập nội dung của màn hình theo bố cục XML.  
   Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện khác trong bố cục, chẳng hạn như TextView. Tìm phương thức onCreate() trong MainActivity:
5. Thêm câu lệnh findViewById vào cuối phương thức onCreate():

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

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

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

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

Mẹo: Để có hướng dẫn chuyên sâu về cách sử dụng ConstraintLayout, hãy xem Codelab Using ConstraintLayout to design your views.

**Bài giải**

**Thử thách**

**Tóm tắt**

**Khái niệm liên quan**

**Tìm hiểu thêm**

### Trình chỉnh sửa bố cục

### Văn bản và các chế độ cuộn

### Tài nguyên có sẵn

## Activities

### Activity và Intent

### Vòng đời của Activity và trạng thái

### Intent ngầm định

## Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ

### Trình gỡ lỗi

### Kiểm thử đơn vị

### Thư viện hỗ trợ

# TRẢI NGHIỆM NGƯỜI DÙNG

## Tương tác người dùng

### Hình ảnh có thể chọn

### Các điều khiển nhập liệu

### Menu và bộ chọn

### Điều hướng người dùng

### RecycleView

## Trải nghiệm người dùng thú vị

### Hình vẽ, định kiểu và chủ đề

### Thẻ và màu sắc

### Bố cục thích ứng

## Kiểm thử giao diện người dùng

### Espresso cho việc kiểm tra UI

# LÀM VIỆC TRONG NỀN

## Các tác vụ nền

### AsyncTask

### AsyncTask và AsyncTaskLoader

### Broadcast receivers

## Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền

### Thông báo

### Trình quản lý cảnh báo

### JobScheduler

# LƯU DỮ LIỆU NGƯỜI DÙNG

## Tùy chọn và cài đặt

### Shared preferences

### Cài đặt ứng dụng

## Lưu trữ dữ liệu với Room

### Room, LiveData và ViewModel

### Room, LiveData và ViewModel