

Nhà phát triển Android

Khóa học cơ bản (Phiên bản 2)



Cập nhật lần cuối: Tue Sep 11, 2018

Khóa học này được tạo bởi nhóm Đào tạo nhà phát triển của Google.

- Để biết chi tiết về khóa học, bao gồm các liên kết đến tất cả các chương khái niệm, ứng dụng và trang trình bày, hãy xem [Kiến thức cơ bản về nhà phát triển Android \(Phiên bản 2\)](#).

developer.android.com/courses/ a dfv2

Lưu ý: Khóa học này sử dụng các thuật ngữ "codelab" và "thực hành" thay thế cho nhau.

Chúng tôi khuyên bạn nên sử dụng phiên bản online của khóa học này thay vì PDF tĩnh này để đảm bảo bạn đang sử dụng nội dung mới nhất.

Xem developer.android.com/courses/adf-v2.

Bài 1: Bắt đầu

PDF này chứa ảnh chụp nhanh một lần về các bài học trong **Unit 1: Bắt đầu**.

Bài học trong bài học này

Bài 1: Xây dựng ứng dụng đầu tiên của bạn

1.1: Android Studio và Hello World

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

1.2B: Trình chỉnh sửa bộ cục

1.3: Chế độ xem văn bản và cuộn

1.4: Tài nguyên có sẵn

Bài học 2: Sinh Hoạt

2.1: Hoạt động và ý định

2.2: Vòng đời và trạng thái hoạt động

2.3: Nỗ lực ngầm

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

3.1: Trình gỡ lỗi

3.2: Kiểm thử đơn vị

3.3: Thư viện hỗ trợ

Bài 1.1: Android Studio và Hello World

Giới thiệu

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

Những điều bạn nên biết

Bạn sẽ có thể:

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

Những gì bạn cần

- Máy tính chạy Windows hoặc Linux hoặc máy Mac chạy macOS. Xem [trang tải xuống Android Studio](#) để biết các yêu cầu hệ thống cập nhật.
- Truy cập Internet hoặc một cách thay thế để tải các bản cài đặt Android Studio và Java mới nhất vào máy tính của bạn.

Những gì bạn sẽ học

- Cách cài đặt và sử dụng Android Studio IDE.
- Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android.
- Cách tạo dự án Android từ mẫu.
- Cách thêm thông báo nhật ký vào ứng dụng của bạn cho mục đích gỡ lỗi.

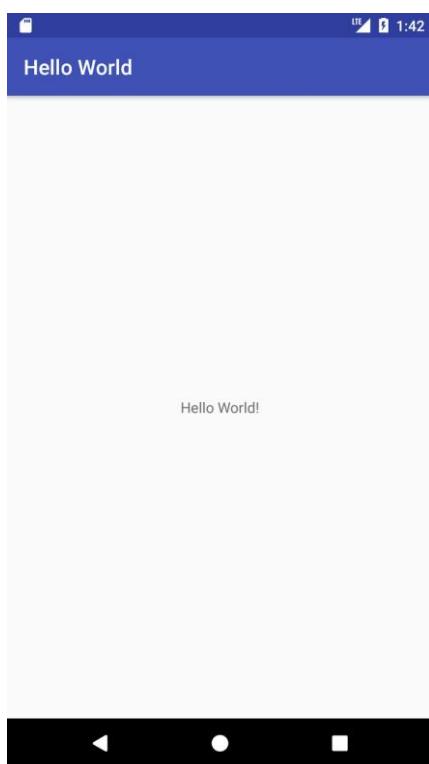
Bạn sẽ làm gì

- Cài đặt môi trường phát triển Android Studio.
- Tạo trình mô phỏng (thiết bị ảo) để chạy ứng dụng trên máy tính.
- Tạo và chạy ứng dụng Hello World trên các thiết bị ảo và vật lý.
- Khám phá bộ cục dự án.
- Tạo và xem thông báo nhật ký từ ứng dụng của bạn.
- Khám phá tệp AndroidManifest.xml.

Tổng quan về ứng dụng

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

Đây là ứng dụng hoàn thành sẽ trông như thế nào:



Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp (IDE) hoàn chỉnh bao gồm trình chỉnh sửa 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 kiểm tra ứng dụng của mình bằng nhiều trình mô phỏng được định cấu hình sẵn hoặc trên thiết bị di động của riêng mình, tạo ứng dụng chính thức và phát hành trên cửa hàng Google Play.

Lưu ý: Android Studio liên tục được cải thiện. Để biết thông tin mới nhất về yêu cầu hệ thống và hướng dẫn cài đặt, hãy xem một [Android Studio](#).

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

Để thiết lập và chạy Android Studio, trước tiên hãy kiểm tra các [yêu cầu về y stem](#) để đảm bảo rằng hệ thống của bạn đáp ứng các yêu cầu đó. Việc cài đặt tương tự cho tất cả các nền tảng. Bất kỳ sự khác biệt nào được lưu ý bên dưới.

1. Điều hướng đến [trang web dành cho nhà phát triển một ndroid](#) 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 được chọn để cài đặt.
3. Sau khi cài đặt xong, Trình hướng dẫn cài đặt sẽ tải xuống và cài đặt một số thành phần bổ sung bao gồm SDK Android. Hãy kiên nhẫn, quá trình này có thể mất một chút thời gian tùy thuộc vào tốc độ Internet của bạn và một số bước có vẻ đú thừa.
4. Khi quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng tạo dự án đầu tiên của mình.

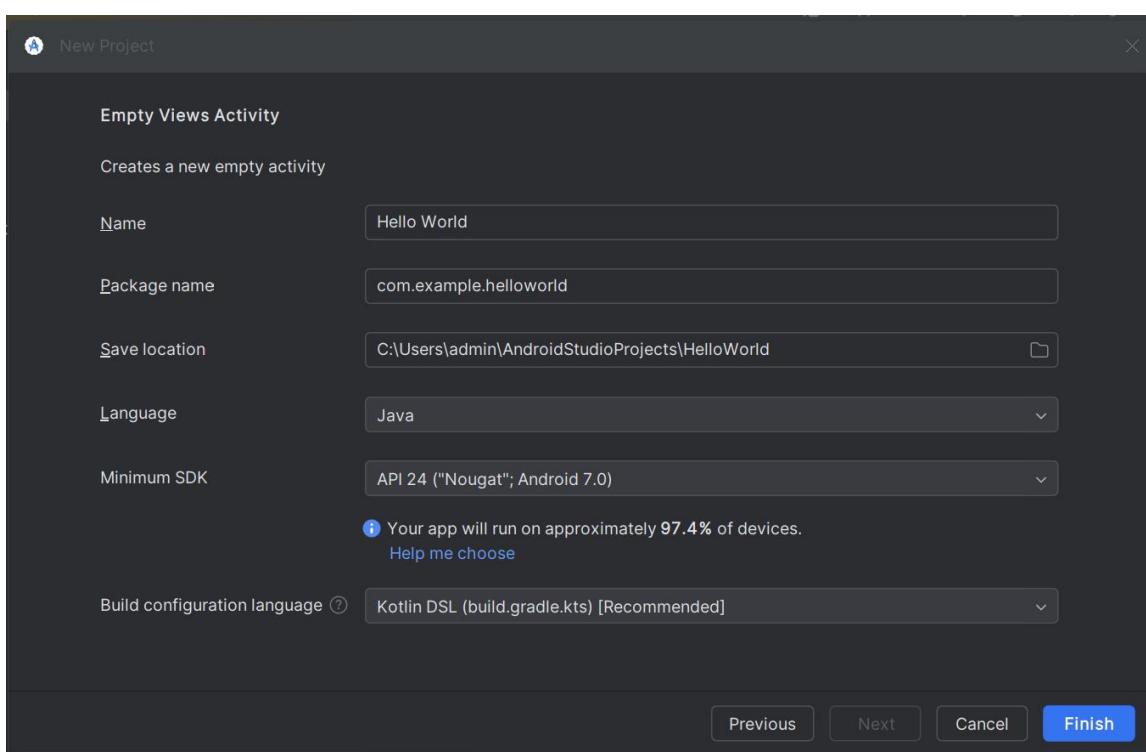
Khắc phục sự cố: Nếu bạn gặp sự cố với cài đặt của mình, hãy kiểm tra ghi [chú phâ hành A ndroid Studio](#) hoặc nhận trợ giúp từ người hướng dẫn của bạn.

Nhiệm vụ 2: Tạo ứng dụng Hello World

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

2.1 Tạo dự án ứng dụng

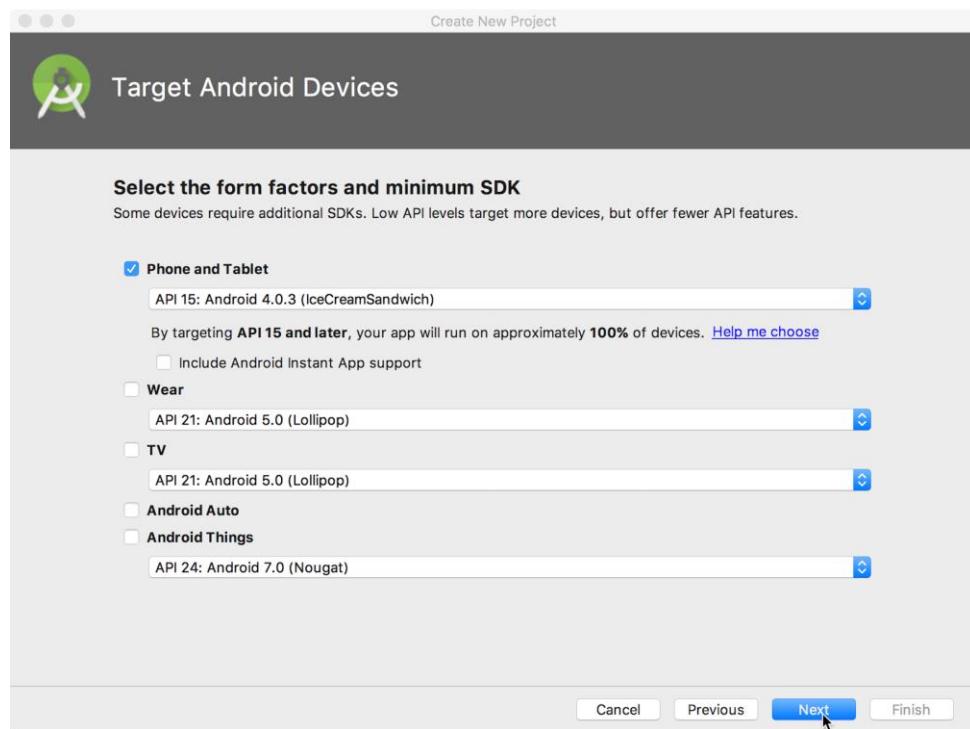
1. Mở Android Studio nếu chưa mở.
2. Trong cửa sổ chính Welcome to Android Studio, nhấp vào Start một dự án Android Studio mới.
3. Trong Create Android Project window, nhập Hello World cho tên ứng dụng A.



4. Xác minh rằng vị trí **Project mặc định** là nơi bạn muốn lưu trữ ứng dụng Hello World và các dự án Android Studio khác hoặc thay đổi vị trí đó thành thư mục ưa thích của bạn.
5. Chấp nhận `ndroid.example.com` mặc định cho **Company Domain** hoặc tạo một tên miền công ty duy nhất.

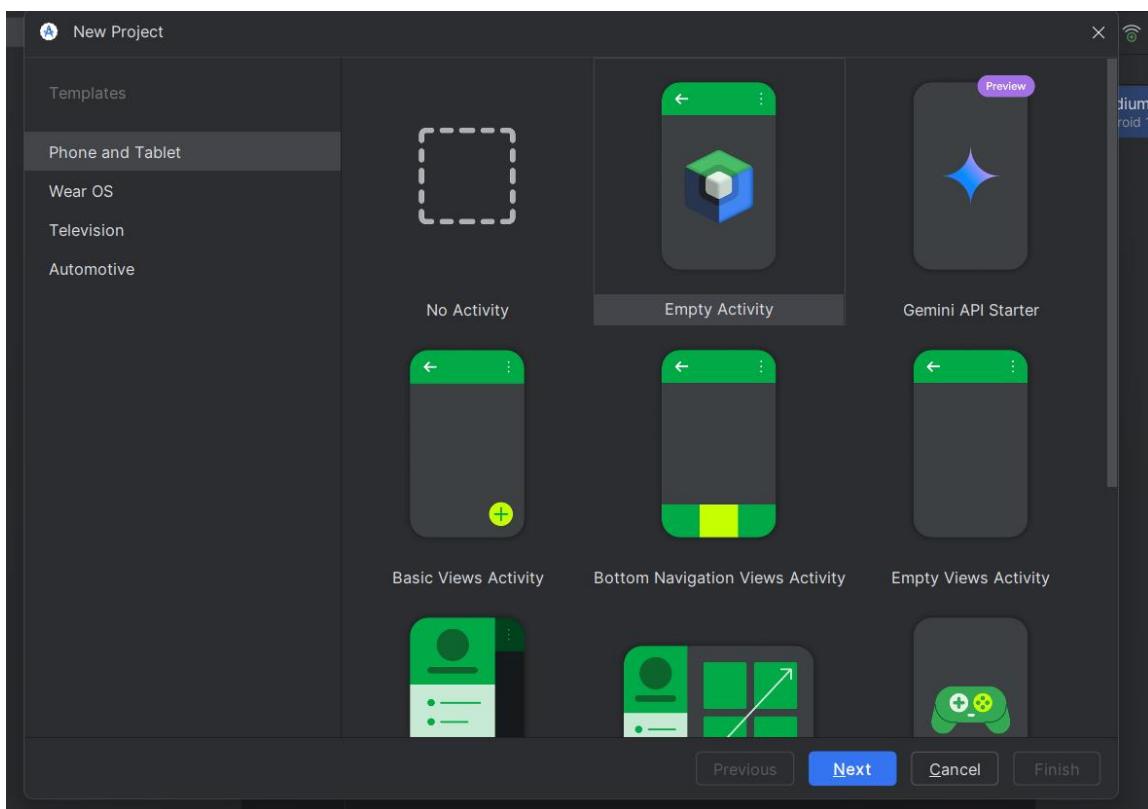
Nếu không định phát hành ứng dụng của mình, bạn có thể chấp nhận mặc định. Lưu ý rằng việc thay đổi tên gói của ứng dụng sau này là một công việc bổ sung.

6. Để bỏ chọn các tùy chọn để **I include C++ support** và **I include Kotlin support**, và nhấp vào **Next**.
7. Trên **màn hình Thiết bị** Android Target, **Phone** và **Tablet** sẽ 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 nó.



Đây là các cài đặt được sử dụng bởi các ví dụ trong các bài học cho khóa học này. Khi viết bài này, các cài đặt này làm cho ứng dụng Hello World của bạn tương thích với 97% thiết bị Android đang hoạt động trên Cửa hàng Google Play.

8. Bỏ chọn **hỗ trợ Úng dụng tức thì** 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 bạn đã chọn, Android Studio sẽ tự động cài đặt các thành phần đó.
9. Thêm **hoạt động** window xuất hiện. **Activity** là một điều duy nhất, tập trung mà người dùng có thể làm. Nó là một thành phần quan trọng của bất kỳ ứng dụng Android nào. Activity thường có một bố cục được liên kết với nó xác định cách các phần tử giao diện người dùng xuất hiện trên màn hình. Android Studio cung cấp Mẫu hoạt động để giúp bạn bắt đầu. Đối với dự án Hello World, hãy chọn **Empty Activity** như hình dưới đây và nhấp vào **Next**.



10. Màn hình **Hoạt động Configure** xuất hiện (khác nhau tùy thuộc vào mẫu bạn đã chọn ở bước trước). Theo mặc định, A ctivity trống được cung cấp bởi mẫu được đặt tên là MainActivity. Bạn có thể thay đổi điều này nếu muốn, nhưng bài học này sử dụng MainActivity.

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

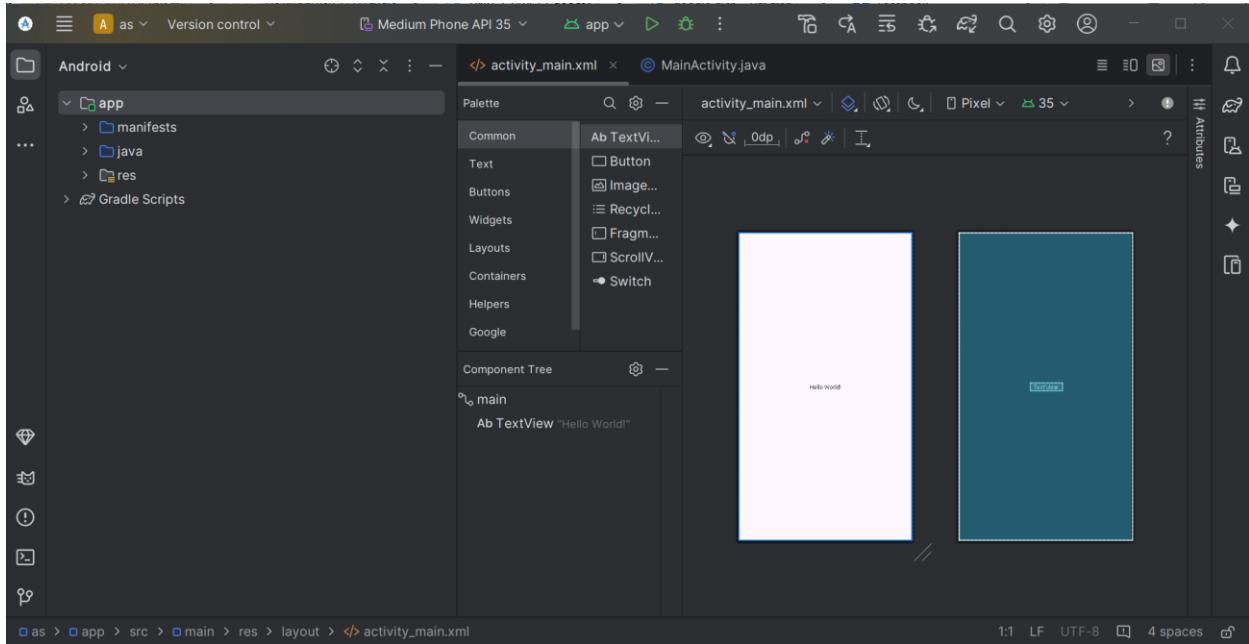
Android Studio tạo một thư mục cho các dự án của bạn và xây dựng dự án bằng [Gradle](#) (quá trình này có thể mất một chút thời gian).

Mẹo: Xem C trên [trang nhà phát triển](#) bản dựng của bạn để biết thông tin chi tiết.

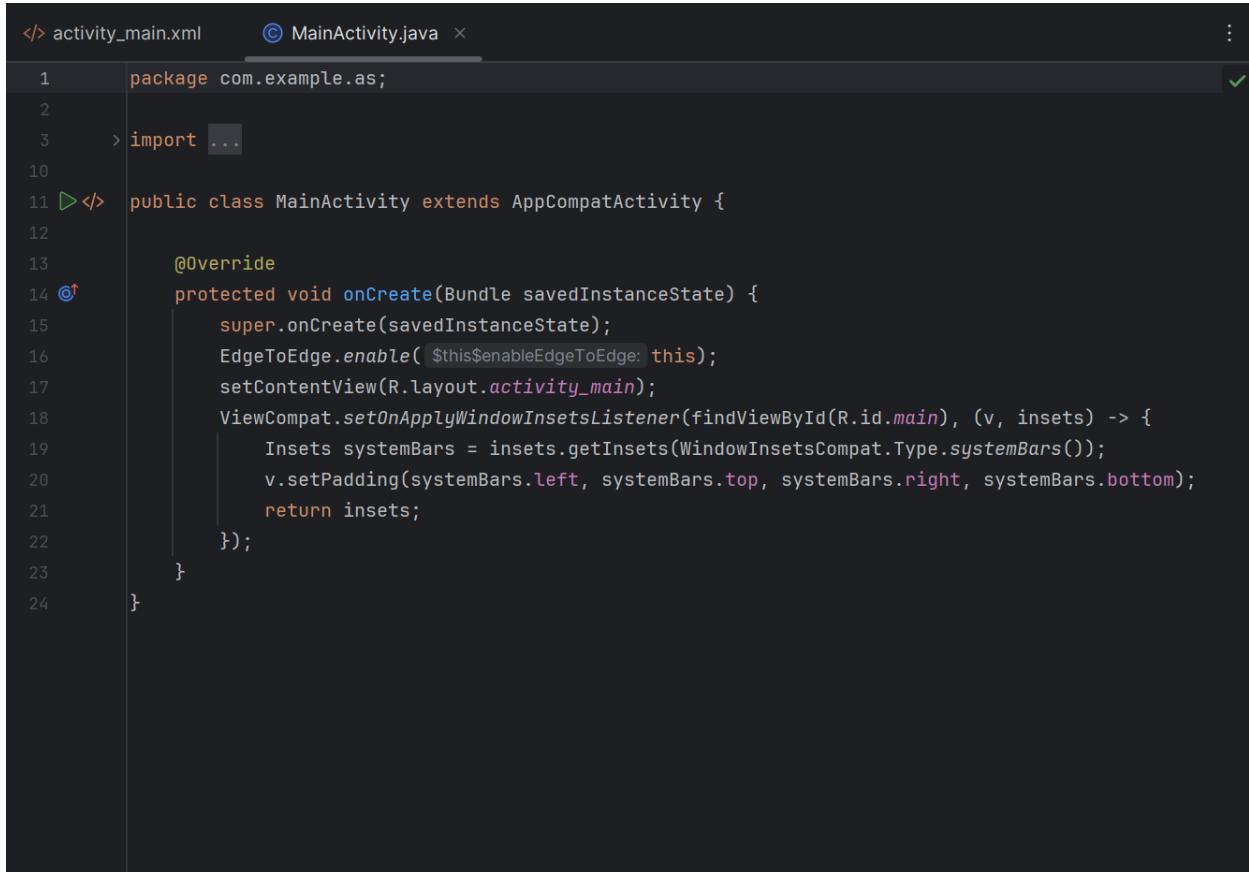
Bạn cũng có thể thấy thông báo "Mẹo trong ngày" với các phím tắt và các mẹo hữu ích khác. Nhập vào **Đóng** để đóng thư.

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

1. Nhập vào **tabactivity_main.xml** để xem trình chỉnh sửa bộ cục.
2. Nhập vào tab Design của **trình soạn thảo bộ cục**, nếu chưa được chọn, để hiển thị hình ảnh của bộ cục như hình dưới đây.



3. Nhập vào tab **MainActivity.java** để xem trình chỉnh sửa mã như hình bên dưới.



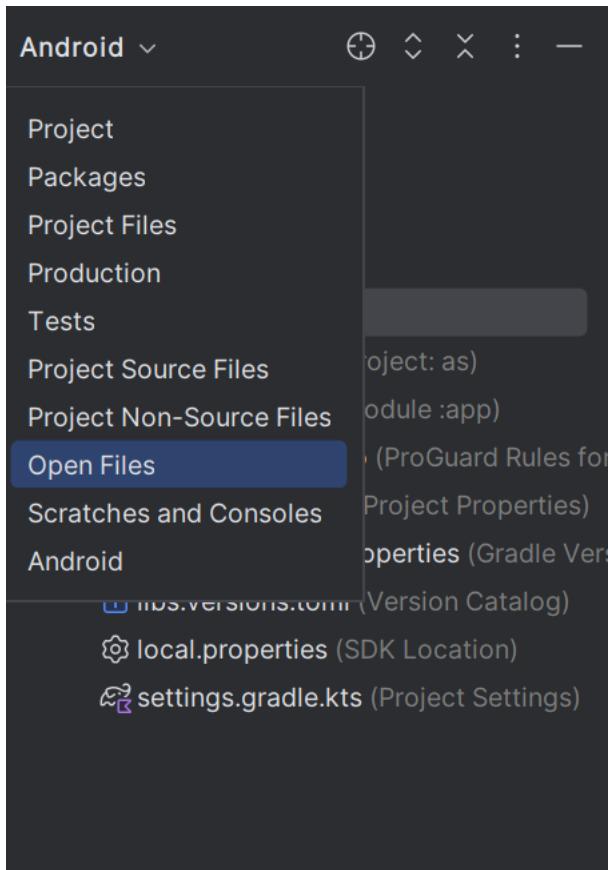
The screenshot shows the Android Studio interface with two tabs open: 'activity_main.xml' and 'MainActivity.java'. The 'MainActivity.java' tab is active, displaying Java code for a Main Activity. The code includes imports, package declaration, class definition, and an overridden onCreate method. The code uses ButterKnife annotations and ViewCompat.setOnApplyWindowInsetsListener to handle window insets.

```
</> activity_main.xml      © MainActivity.java × :  
1 package com.example.as;  
2  
3 > import ...  
10  
11 <> public class MainActivity extends AppCompatActivity {  
12  
13     @Override  
14     protected void onCreate(Bundle savedInstanceState) {  
15         super.onCreate(savedInstanceState);  
16         EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
17         setContentView(R.layout.activity_main);  
18         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {  
19             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());  
20             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
21             return insets;  
22         });  
23     }  
24 }
```

2.2 Khám phá khung Project > Android

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

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

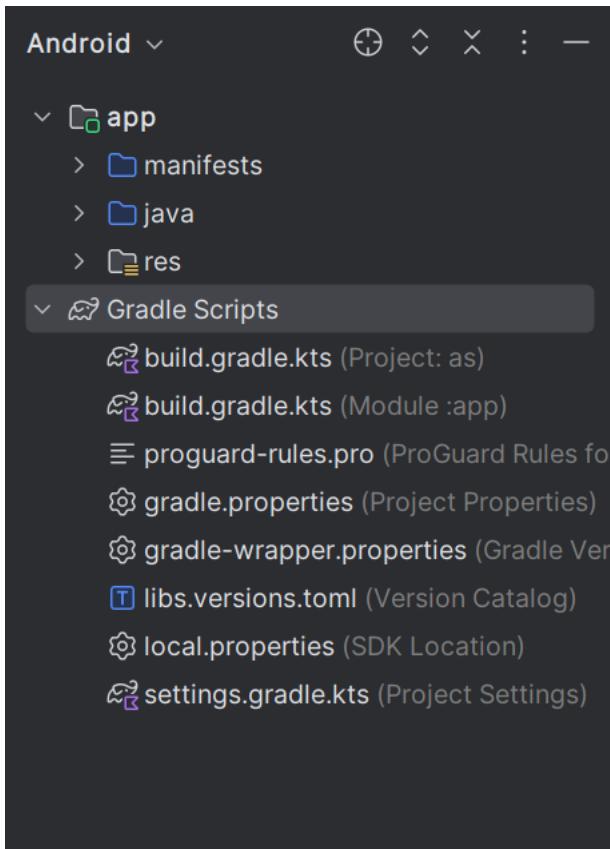


Lưu ý: Chương này và các chương khác đề cập đến ngăn Dự án, khi được đặt thành **Android**, là ngăn **Dự án > Android**.

2.3 Khám phá thư mục Tập lệnh Gradle

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

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



Làm theo các bước sau để khám phá hệ thống Gradle:

1. Nếu thư mục **Tập lệnh Gradle** không được mở rộng, hãy nhấp vào hình tam giác để mở rộng thư mục đó. 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 bản dựng Gradle cấp cao nhất. Hầu hết thời gian, bạn sẽ không cần thực hiện bất kỳ thay đổi nào đối với tệp này, nhưng vẫn hữu ích khi hiểu nội dung của nó.

Theo mặc định, tệp bản dựng cấp cao nhất sử dụng buildscript để xác định kho lưu trữ Gradle và các phần phụ thuộc chung cho tất cả các mô-đun trong dự án. Khi phần phụ thuộc của bạn không phải là 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 khôi kho lưu trữ của tệp này. Theo mặc định, các dự án Android Studio mới khai báo JCenter và Google (bao gồm các [Google Maven repository](#)) là vị trí kho lưu trữ:



```
Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work p Try Again Open 'Build' View Show Log in Explorer
```

```
1 // Top-level build file where you can add configuration options common to all sub-projects/modules.
2 plugins {
3     alias(libs.plugins.android.application) apply false
4 }
5 allprojects {
6     repositories {
7         google()
8         jcenter()
9     }
10 }
```

3. 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 định cấu hình cài đặt bản dựng cho từng mô-đun cụ thể (ứng dụng HelloWorld chỉ có một mô-đun). Việc định cấu hình các tùy chọn 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 tùy chọn cài đặt trong tệp AndroidManifest.xml hoặc tệp build.gradle cấp cao nhất.

Tệp này thường là tệp cần chỉnh sửa khi thay đổi cấu hình cấp ứng dụng, chẳng hạn như khai báo các phần phụ thuộc trong phần dependencies. Bạn có thể khai báo phần phụ thuộc thư viện bằng cách sử dụng một trong một số cấu hình phần phụ thuộc khác nhau. Mỗi cấu hình phần 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

sự phụ thuộc của tất cả các tệp ".jar" bên trong thư mục libs.

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

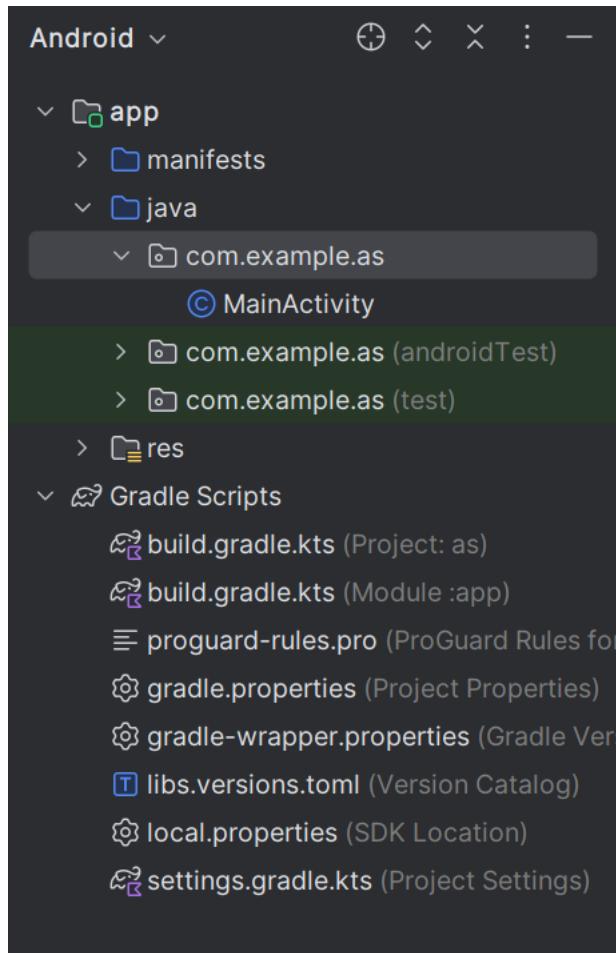
```
1  plugins {
2      alias(libs.plugins.android.application)
3  }
4
5  android {
6      namespace = "com.example.as"
7      compileSdk = 35
8
9      defaultConfig {
10         applicationId = "com.example.as"
11         minSdk = 24
12         targetSdk = 35
13         versionCode = 1
14         versionName = "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             androidTestImplementation 'com.android.support.test:runner:1.0.1' androidTestImplementation
27             'com.android.support.test.espresso:espresso-core:3.0.1' }
```

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

2.4 Khám phá ứng dụng và thư mục res

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

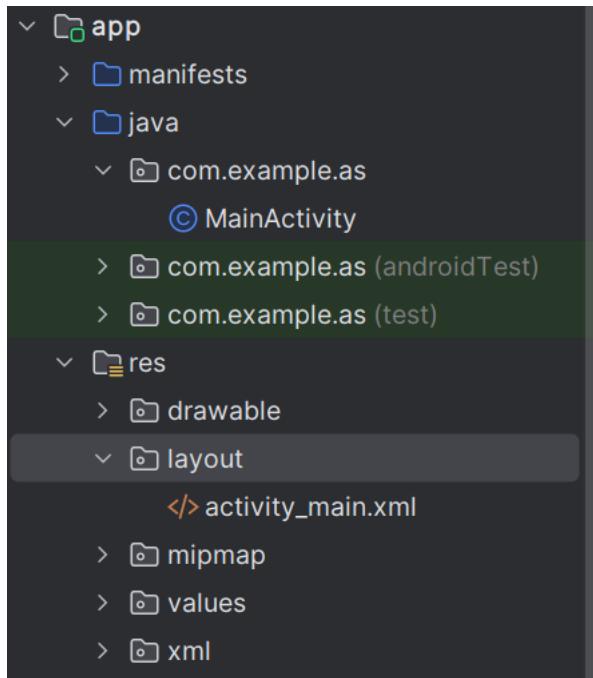
1. Mở rộng **thư mục app**, **thư mục java** và thư mục **com.example.android.helloworld** để xem tệp java **MainActivity**. Nhấp đúp vào tệp sẽ mở tệp đó trong trình soạn thảo mã.



Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như trong hình trên. Thư mục **com.example.hello.helloworld** (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục còn lại được sử dụng để kiểm tra 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 Activity

(màn hình) đầu tiên mà người dùng nhìn thấy, cũng khởi tạo các tài nguyên trên toàn ứng dụng, thường được gọi là **MainActivity** (phần mở rộng tệp bị bỏ qua trong ngăn **Project > Android**).

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



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

2.5 Khám phá thư mục tệp kê khai

Thư mục **manifests** chứa các tệp cung cấp thông tin cần thiết về ứng dụng của bạn cho Hệ thống Android, hệ thống phải có 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 thành phần A, phải được khai báo trong tệp XML này. Trong các bài học khóa học khác, bạn sẽ sửa đổi tệp này để thêm các tính năng và quyền tính năng. Để biết thêm giới thiệu, hãy xem [Tổng quan về bản kê khai pp](#).

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 \(AVD\)](#) để tạo một thiết bị ảo (còn được gọi là trình mô phỏng) mô phỏng cấu hình cho một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Xin lưu ý rằng Trình mô phỏng Android có các [yêu cầu riêng ngoài](#) các yêu cầu hệ thống cơ bản cho Android Studio.

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

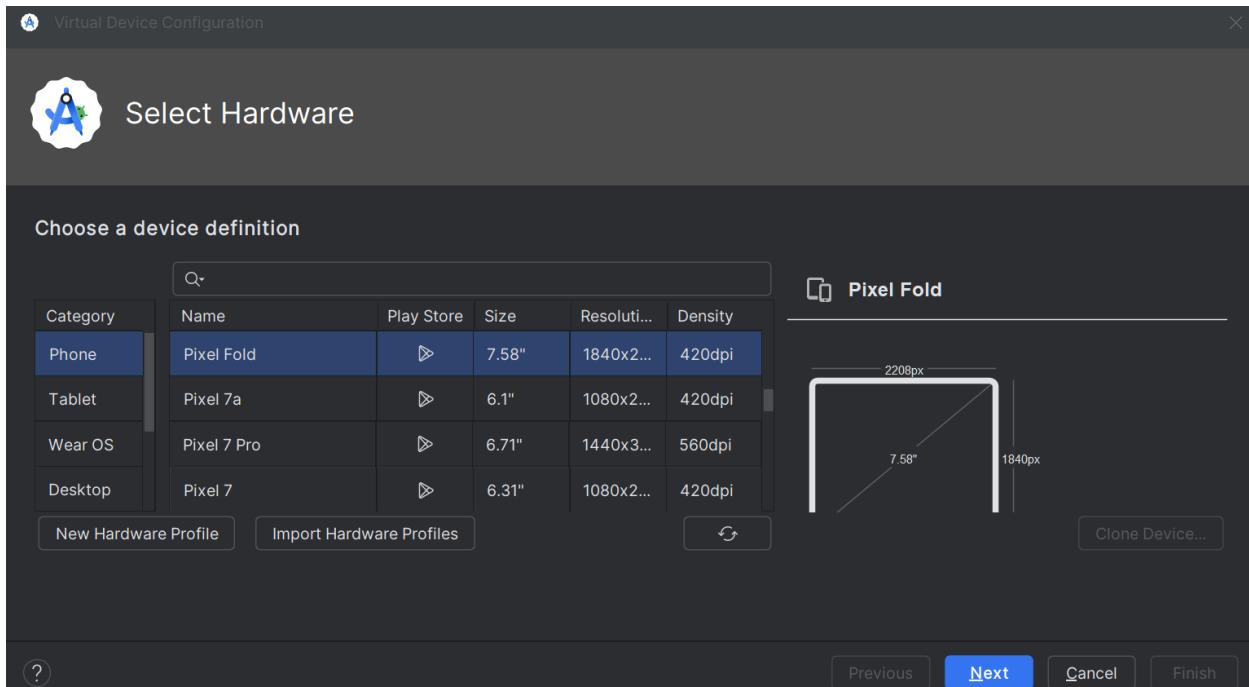
3.1 Tạo thiết bị ảo Android (AVD)

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

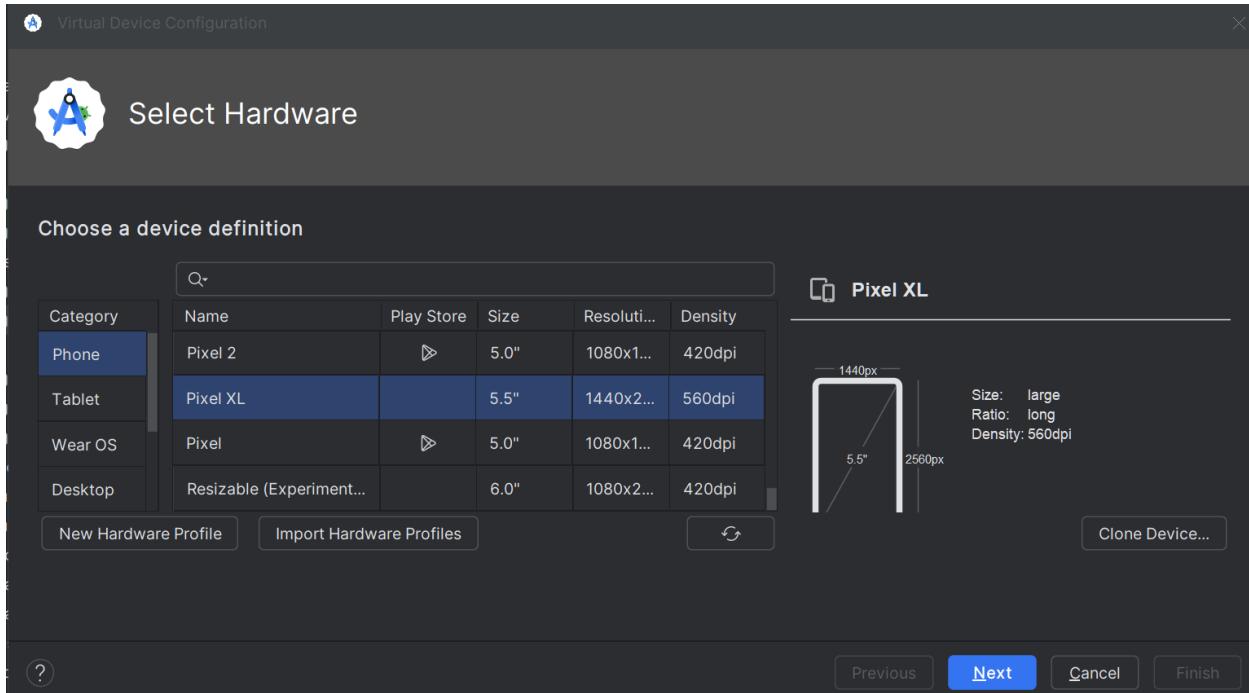
1. Trong Android Studio, hãy chọn **Tools > Trình quản lý AVD > Android** hoặc nhập vào biểu tượng Trình quản lý AVD



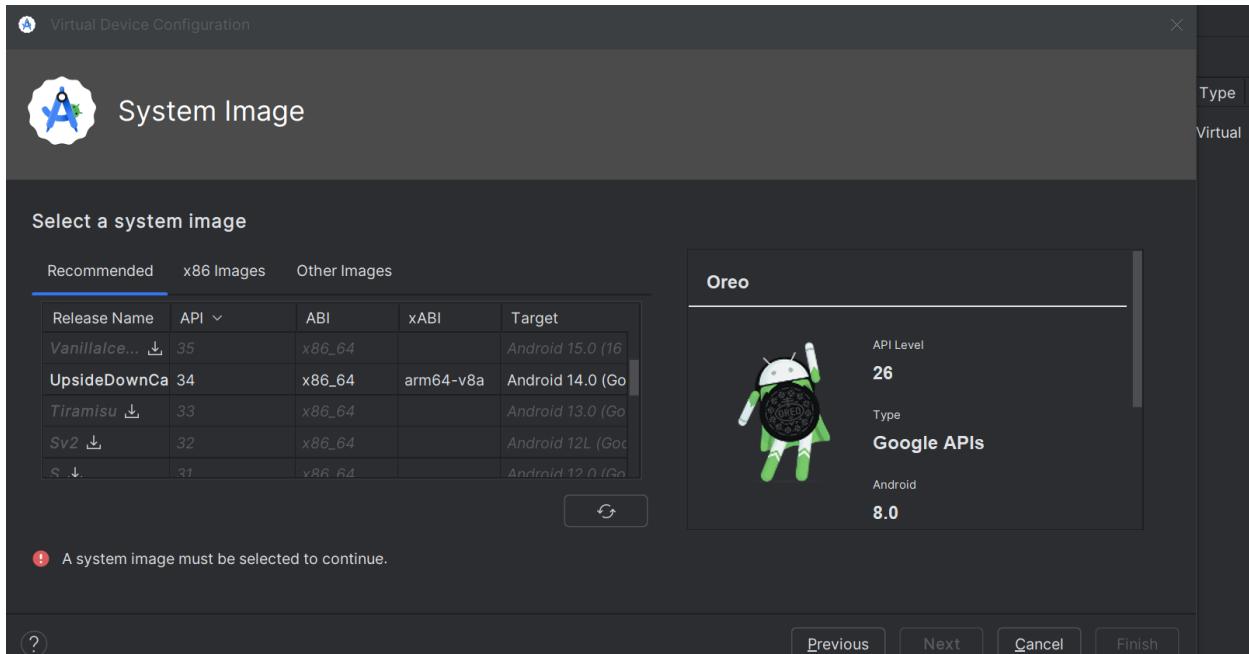
trên thanh công cụ. Y **thiết bị ảo của chúng tôi** xuất hiện. Nếu bạn đã tạo thiết bị ảo, màn hình hiển thị chúng (như trong hình bên dưới); nếu không, bạn sẽ thấy một danh sách trống.



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



3. Chọn một thiết bị như **Nexus 5x** hoặc **Pixel XL**, và nhấp vào **Next**. **System Image** screen xuất hiện.
4. Nhấp vào **tab Đè xuất** nếu nó chưa được chọn và chọn phiên bản hệ thống Android để chạy trên thiết bị ảo (chẳng hạn như **Oreo**).



Có nhiều phiên bản có sẵn hơn so với hiển thị trong tab **Rkhuyễn nghị**. Nhìn vào các tab **x 86 Images** và **Other Images** để xem chúng.

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

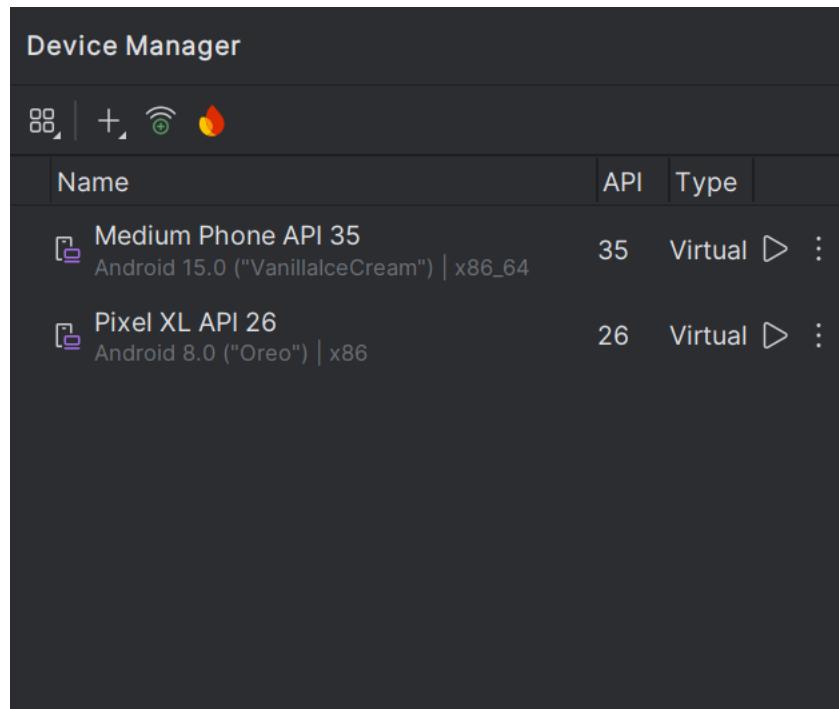
- Sau khi chọn hình ảnh hệ thống, hãy nhấp vào **Next**. **Cửa sổ Thiết bị ảo Android (AVD)** xuất hiện. Bạn cũng có thể thay đổi tên của AVD. Kiểm tra cấu hình của bạn và nhấp vào **Finish**.

3.2 Chạy ứng dụng trên thiết bị ảo

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

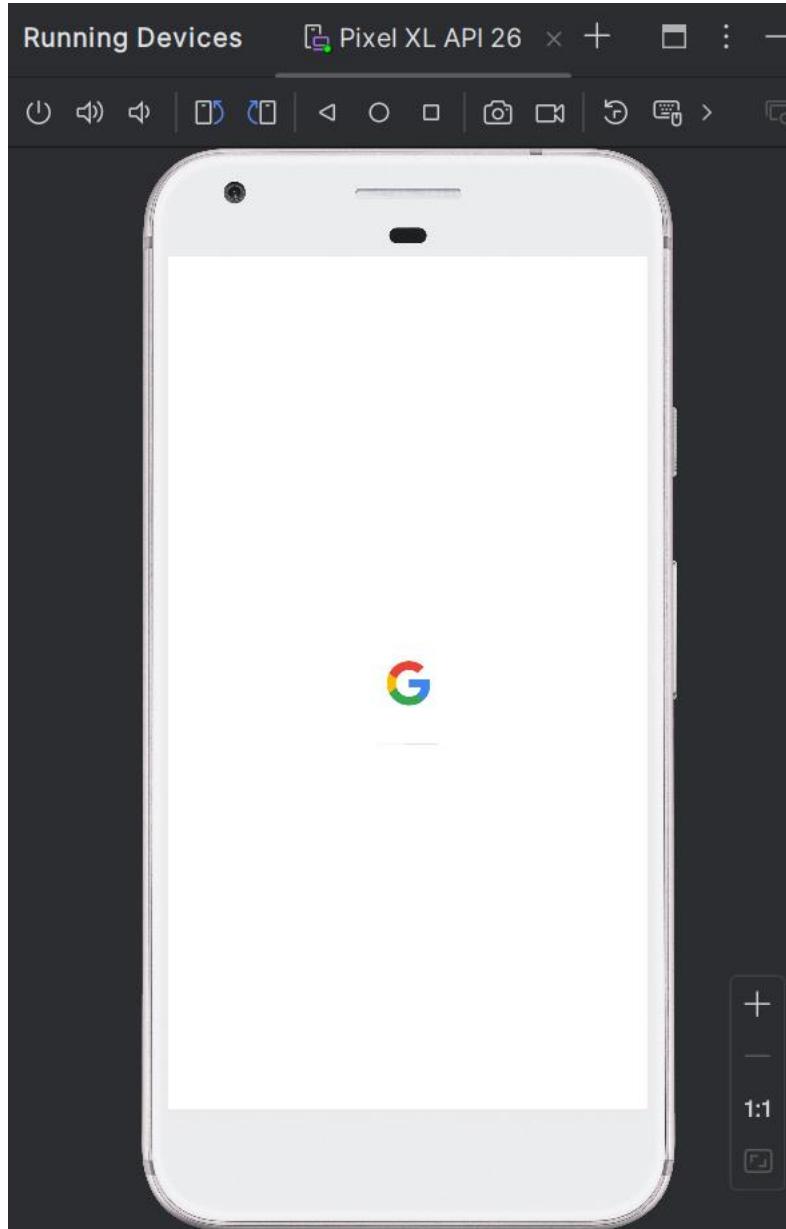
- Trong Android Studio, hãy chọn **Ứng dụng** > **Run** hoặc nhấp vào **biểu tượng Run**  trên thanh công cụ.

2. Cửa sổ **Select Deployment Target**, trong **A available Virtual Devices**, chọn thiết bị ảo mà bạn vừa tạo và nhấp vào **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, quá trình này có thể mất một lúc. Ứng dụng của bạn sẽ được xây dựng và sau khi trình mô phỏng đã sẵn sàng, Android Studio sẽ tải ứng dụng lên trình mô phỏng 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 từ đầu phiên của bạn. Bạn không nên đóng ứng dụng cho đến khi hoàn tất việc kiểm tra ứng dụng của mình để ứng dụng của bạn không phải trải qua quá trình khởi động thiết bị một lần nữa. Để đóng thiết bị ảo, hãy nhấp vào nút X tại

ở đầu trình giả lập, chọn **Quit** từ menu hoặc nhấn **Control-Q** trong Windows hoặc **Command-Q** trong macOS.

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

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

- Thiết bị Android như điện thoại hoặc máy tính bảng.
- Cáp dữ liệu để kết nối thiết bị Android 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 phàn cứng. Kiểm tra [tài liệu về Thiết bị phàn cứng Using](#). Bạn cũng có thể cần cài đặt trình điều khiển USB thích hợp cho thiết bị của mình. Đối với trình điều khiển USB dựa trên Windows, hãy xem Trình [điều khiển USB OEM](#).

4.1 Bật gỡ lỗi USB

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

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

1. Trên thiết bị của bạn, hãy mở **Settings**, tìm kiếm **Giới thiệu p hone**, nhấp vào **About phone** và nhấn vào **Số xây dựng** bảy lần.
2. Quay lại màn hình trước đó (**Settings/ System**). Các **tùy chọn Developer** xuất hiện trong danh sách. Nhấn vào **Tùy chọn Developer**.
3. Chọn **gỡ lỗi USB**.

4.2 Chạy ứng dụng của bạn trên thiết bị

Giờ đây, 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 có sẵn và các thiết bị đã được kết nối.
3. Chọn thiết bị của bạn và nhấp vào **OK**.

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

Troubleshooting

Nếu Android Studio không nhận dạng thiết bị của bạn, hãy thử các cách 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 là "không được phép", hãy làm theo các bước sau:

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

Bạn có thể cần cài đặt trình điều khiển USB thích hợp cho thiết bị của mình. Xem [tài liệu về Thiết bị phần cứng Using](#).

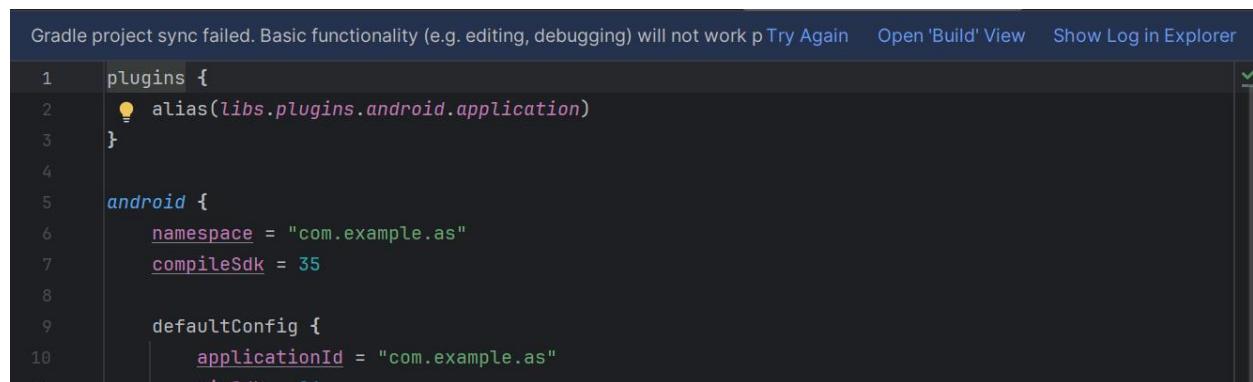
Nhiệm vụ 5: Thay đổi cấu hình Gradle của ứng dụng

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

5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Làm theo các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu nó chưa được mở và nhấp đúp vào tập tin **build.gradle(Module:app)**.
Nội dung của tệp xuất hiện trong trình soạn thảo mã.
2. Trong khối defaultConfig, thay đổi giá trị của minSdkVersion thành 17 như hình bên dưới (ban đầu nó được đặt thành 15).



```
Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work p Try Again Open 'Build' View Show Log in Explorer
1 plugins {
2     alias(libs.plugins.android.application)
3 }
4
5 android {
6     namespace = "com.example.as"
7     compileSdk = 35
8
9     defaultConfig {
10         applicationId = "com.example.as"
11         minSdk = 17
12     }
13 }
```

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

5.2 Đóng bộ hóa cấu hình Gradle mới

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

Để đóng bộ hóa các tệp dự án, hãy nhấp vào **Sync Now** trong thanh thông báo xuất hiện khi thực hiện thay đổi



(như trong hình trước) hoặc nhấp vào **biểu tượng** Dự án Sync với Tệp Gradle trên thanh công cụ.

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

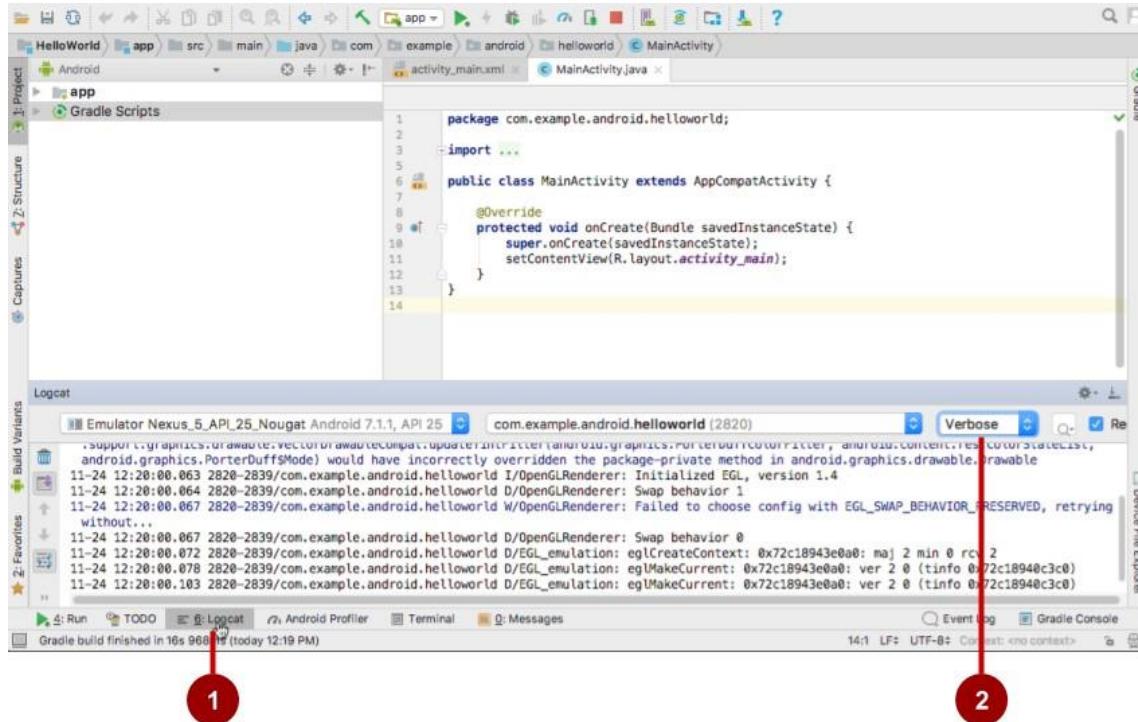
Để hiểu rõ hơn về Gradle, hãy xem tài liệu về [Tổng quan về hệ thống Build](#) và [Cách bản dựng Gradle](#).

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, câu lệnh này hiển thị thông báo trong **ngăn Logcat**. Thông báo nhật ký 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 ngoại lệ.

6.1 View ngăn Logcat

Để xem ngăn **Logcat**, hãy nhấp vào tab **Logcat** ở cuối cửa sổ Android Studio như trong hình bên dưới.



Trong hình trên:

1. Tab **L**ogcat để mở và đóng **L**ogcat, hiển thị thông tin về ứng dụng của bạn khi ứng dụng đang chạy. Nếu bạn thêm câu lệnh `Log` vào ứng dụng của mình, thông báo `Log` sẽ xuất hiện ở đây.
2. Menu mức `Log` được đặt thành **V**erbose (mặc định), hiển thị tất cả các thông báo `Log`. Các cài đặt khác bao gồm **D**ebug, **E**rror, **I**nfo và **W**arn.

6.2 Thêm câu lệnh nhật ký vào ứng dụng của bạn

Câu lệnh nhật ký trong mã ứng dụng của bạn sẽ hiển thị thông báo trong ngăn Logcat. Chẳng hạn:

```
Log.d("Hoạt động chính", "Xin chào thế giới");
```

Các phần của thông điệp là:

- Log: Lớp `Log` để gửi tin nhắn nhật ký đến ngăn Logcat.
- d: Cài đặt cấp độ **D ebug** Log để lọc thông báo nhật ký hiển thị trong ngăn Logcat. Các mức log khác là e cho **E rror**, w cho **W arn** và i cho **I nfo**.
- "MainActivity": Đối số đầu tiên là một thẻ có thể được sử dụng để lọc tin nhắn trong ngăn Logcat. Đây thường là tên của Activity mà thông điệp bắt nguồn từ đó. Tuy nhiên, bạn có thể làm cho điều này bất cứ thứ gì hữu ích cho bạn để gõ lỗi. Theo quy ước, các thẻ nhật ký được định nghĩa là hàng số cho khoảng cách A:

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

- "Xin chào thế giới": Đối số thứ hai là thông điệp thực tế.

Làm theo các bước sau:

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

```
Log.d("Hoạt động chính", "Xin chào thế giới");
```

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

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Log.d("Hoạt động chính", "Xin chào thế giới");
```

}

6. Nếu ngăn Logcat chưa mở, hãy nhấp vào tab **L ogcat** ở cuối Android Studio để mở.
7. Kiểm tra xem tên của mục tiêu và tên gói của ứng dụng có chính xác không.
8. Thay đổi mức L og trong ngăn **L ogcat** thành **D ebug** (hoặc để nguyên **V erbose** vì có rất ít thông báo nhật ký).
9. Chạy ứng dụng của bạn.

Thông báo sau sẽ xuất hiện trong ngăn Logcat:

11-24 14:06:59.001 4696-4696/? D>MainActivity: Xin chào thế giới

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 này.

Thử thách: Nếu bạn đã thiết lập và quen thuộc với quy trình phát triển cơ bản, hãy làm như sau:

1. Tạo một dự án mới trong Android Studio.
2. Thay đổi lời chào "Hello World" thành "Chúc mừng sinh nhật thành" và tên của ai đó có sinh nhật gần đây.
3. (Tùy chọn) Chụp ảnh màn hình ứng dụng đã hoàn thành của bạn và gửi email cho người mà bạn quên ngày sinh.
4. Một cách sử dụng phổ biến của L óp L og là ghi nhật ký các ngoai l ê J ava khi chúng xảy ra trong chương trình của bạn.

Có một số phương thức hữu ích, chẳng hạn như L og.e(), mà bạn có thể sử dụng cho mục đích này. Khám phá các phương thức bạn có thể sử dụng để bao gồm ngoại lệ với thông báo L og. 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, hãy truy cập [Android Studio](#) và làm theo hướng dẫn để tải xuống và cài đặt nó.
- Khi tạo một ứng dụng mới, hãy đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm SDK tối thiểu.
- Để xem hệ thống phân cấp Android của ứng dụng trong ngăn Dự án, hãy bấm 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.
- Tắt cả mã và tài nguyên cho ứng dụng nằm trong thư mục `app/res`. Thư mục `java` bao gồm các hoạt động, kiểm tra và các thành phần khác trong mã nguồn Java. Thư mục `res` chứa các tài nguyên, chẳng hạn như bố cục, chuỗi và hình ảnh.
- Chỉnh sửa tệp `AndroidManifest.xml` để thêm các tính năng, thành phần 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 \(AVD\) Android](#) để tạo thiết bị ảo (còn được gọi là trình mô phỏng) để chạy ứng dụng của bạn.
- Thêm [câu lệnh Log](#) vào ứng dụng của bạn, câu lệnh này hiển thị 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 tính năng Gỡ lỗi USB trên thiết bị. Mở **Settings > Khoảng p mài và nhấn Build số** bảy lần. Quay lại màn hình trước đó (**Settings**) và nhấn vào **Developer options**. Chọn **gỡ lỗi USB**.

Các khái niệm liên quan

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

Tìm hiểu thêm

Tài liệu Android Studio:

- [Trang tải xuống Android Studio](#)
- [Ghi chú phát hành Android Studio](#)
- [Làm quen với Android Studio](#)
- [Công cụ dòng lệnh Logcat](#)
- [Trình quản lý thiết bị ảo Android \(AVD\)](#)
- [Tổng quan về tệp kê khai ứng dụng](#)
- [Định cấu hình bản dựng của bạn](#)
- [Lớp nhật ký](#)
- [Tạo và quản lý thiết bị ảo](#)
- Khác:
 - [Làm cách nào để cài đặt Java?](#)
 - [Cài đặt phần mềm JDK và JAVA_HOME cài đặt](#)
 - [Trang web Gradle](#)
 - [Cú pháp Apache Groovy](#)
 - [Trang Wikipedia của Gradle](#)

Homework

Xây dựng và chạy ứng dụng

- Tạo một dự án Android mới từ Empty Template.
- Thêm câu lệnh ghi nhật ký cho các cấp độ nhật ký khác nhau trong `onCreate()` trong hoạt động chính.
- Tạo trình mô phỏng cho thiết bị, nhắm mục tiêu bất kỳ phiên bản Android nào bạn thích và chạy ứng dụng.
- Sử dụng tính năng lọc trong **Logcat** để tìm các câu lệnh nhật ký của bạn và điều chỉnh các cấp độ để chỉ hiển thị các câu lệnh gỡ lỗi hoặc ghi nhật ký lỗi.

Trả lời những câu hỏi này

Câu hỏi 1

Tên của tệp bô cục cho hoạt động chính là gì?

- `MainActivity.java`

- AndroidManifest.xml
- activity_main.xml
- xây dựng.gradle

Câu hỏi 2

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

- app_name
- xmlns:ứng dụng
- android:tên
- applicationId

Câu hỏi 3

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

- Màn hình thiết bị Android
- Trình quản lý AVD
- Trình quản lý SDK • Trình chỉnh sửa chủ đề

Câu hỏi 4

Giả sử rằng ứng dụng của bạn bao gồm câu lệnh ghi nhật ký sau:

```
Log.i("MainActivity", "Bộ cục MainActivity đã hoàn tất");
```

Bạn sẽ thấy câu lệnh "Bộ cục MainActivity đã hoàn tất" trong ngăn **Logcat** nếu menu Mirc nhật ký được đặt thành tùy chọn nào sau đây? (Gợi ý: nhiều câu trả lời là được.)

- Dài dòng
- Gõ lỗi

- Thông tin
- Cảnh báo
- Lỗi
- Khẳng định

Gửi ứng dụng của bạn để chấm điểm

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

- Một ctivity hiển thị "Hello World" trên màn hình.
- Ghi nhật ký các câu lệnh trong o nCreate() trong hoạt động chính.
- Cập nhật ký trong **ngăn Logcat** chỉ hiển thị các câu lệnh gõ lỗi hoặc ghi nhật ký lỗi.

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 *views* - mọi phần tử của màn hình đều là **View**. Lớp View đại diện cho khối xây dựng cơ bản cho tất cả các thành phần giao diện người dùng và lớp cơ sở cho các lớp cung cấp các thành phần giao diện người dùng tương tác như nút, hộp kiểm và trường nhập văn bản. Các lớp con View thường được sử dụng được mô tả trong một số bài học bao gồm:

- **TextView** để hiển thị văn bản.
- **EditText** để cho phép người dùng nhập và chỉnh sửa văn bản.
- **Nút** 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 được chứa trong một lớp mở rộng [Activity](#). Một Hoạt động thường được liên kết với bố cục của chế độ xem giao diện người dùng được xác định dưới dạng tệp XML (Ngôn ngữ đánh dấu eXtended). Tệp XML này thường được đặt tên theo chữ A của nó và xác định bố cục của các phần tử View trên màn hình.

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

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

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

Những điều bạn nên biết

Bạn nên làm quen với:

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

Những gì bạn sẽ học

- Cách tạo ứng dụng với hành vi tương tác. ● Cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục.
- Cách chỉnh sửa bố cục trong XML.
- Rất nhiều thuật ngữ mới. Kiểm tra [bảng thuật ngữ từ và khái niệm Vocabulary](#) hoặc các định nghĩa thân thiện.

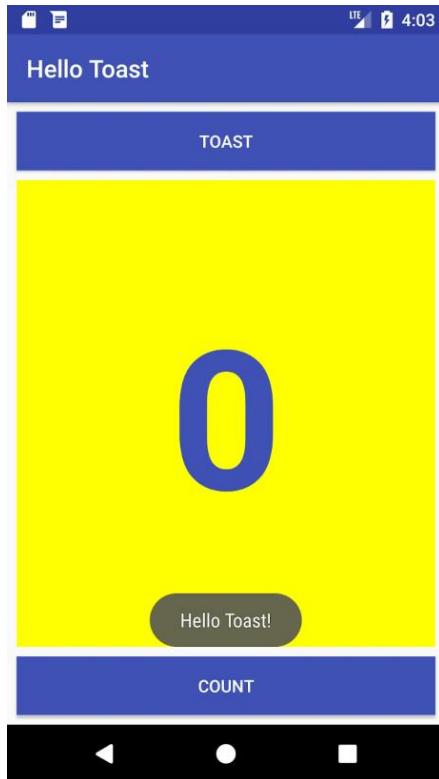
Bạn sẽ làm gì

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

Tổng quan về ứng dụng

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

Đây là những gì ứng dụng đã hoàn thành trông như thế nào:



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

Trong thực tế 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

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

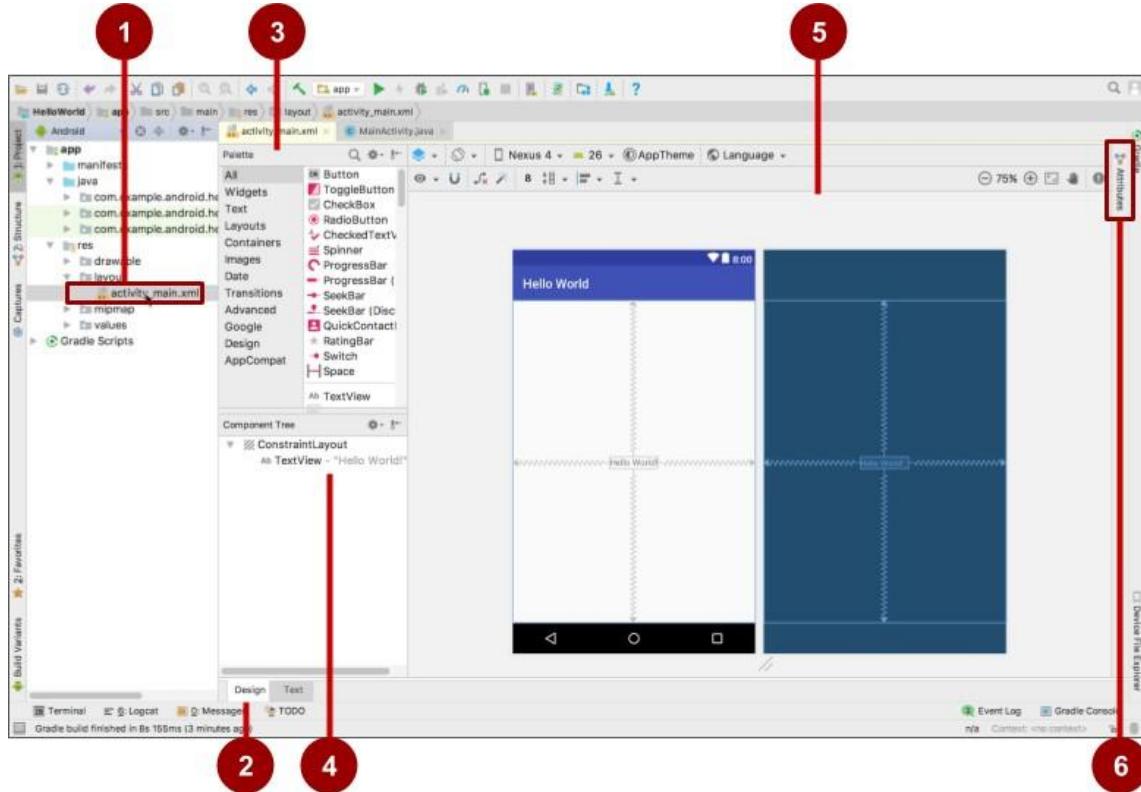
Thuộc tính	Giá trị
Tên ứng dụng	Xin chào Toast
Tên công ty	com.example.android (hoặc miền của riêng bạn)
SDK tối thiểu trên điện thoại và máy tính bảng	API15: Android 4.0.3 IceCreamSandwich
Mẫu	Hoạt động trống rỗng
Tạo hộp tệp bô cục	Chọn
Hộp tương thích ngược	Chọn

15. Chọn **R un > Chạy ứng dụng** hoặc nhấp vào **biểu tượng R un** trên  thanh công cụ để tạo và thực thi ứng dụng trên trình mô phỏng hoặc thiết bị của bạn.

1.2 Khám phá trình chỉnh sửa bô cục

Android Studio cung cấp trình chỉnh sửa bô cục để nhanh chóng tạo bô cục của các thành phần giao diện người dùng (UI) của ứng dụng. Nó cho phép bạn kéo các phần tử vào chế độ xem thiết kế trực quan và bô cục, định vị chúng trong bô cục, thêm ràng buộc và đặt thuộc tính. Các yếu tố C xác định vị trí của một phần tử giao diện người dùng trong bô cục. Một ràng buộc đại diện cho một kết nối hoặc căn chỉnh với một chế độ xem khác, bô cục cha hoặc một hướng dẫn vô hình.

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



- Trong **thư mục app > res > layout** trong **ngân Project > Android**, nhấp đúp vào tệp **activity_main.xml** để mở nó, nếu nó chưa được mở.
- Nhấp vào tab **Design** nếu nó chưa được chọn. Bạn sử dụng tab **Design** để thao tác với các phần tử và bố cục, và tab **Text** để chỉnh sửa mã XML cho bố cục.
- Ngân **Plettes** hiển thị các thành phần giao diện người dùng mà bạn có thể sử dụng trong bố cục của ứng dụng.
- Ngân **cây hỗ trợ C** hiển thị hệ thống phân cấp chế độ xem của các phần tử giao diện người dùng. Các phần tử View được tổ chức thành một hệ thống phân cấp cây gồm cha mẹ và con, trong đó con kế thừa các thuộc tính của cha mẹ của nó. Trong hình trên, TextView là con của ConstraintLayout. Các em sẽ học về các yếu tố này ở phần sau của bài học này.
- Các ngăn thiết kế và bẩn thiết kế của trình chỉnh sửa bố cục hiển thị các phần tử giao diện người dùng trong bố cục. Trong hình trên, bố cục chỉ hiển thị một phần tử: TextView hiển thị "Hello World". 6. Tab **Phân bố A** hiển thị ngăn Phân bố A để thiết lập thuộc tính cho phần tử giao diện người dùng.

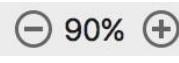
Mẹo: Xem B [sử dụng giao diện người dùng bằng Trình chỉnh sửa bố cục](#) để biết thông tin chi tiết về cách sử dụng trình chỉnh sửa bố cục và [xem Android Studio](#) để biết tài liệu đầy đủ về Android Studio.

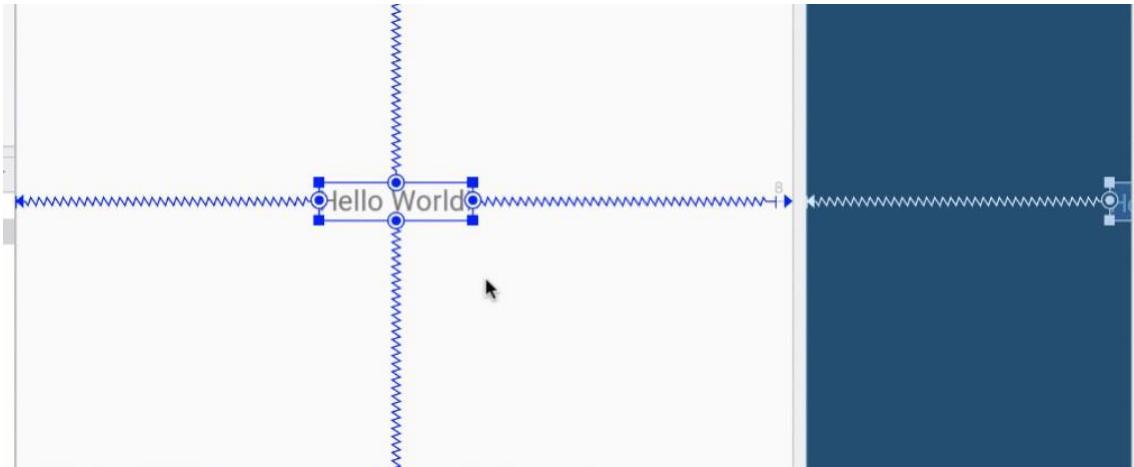
Nhiệm vụ 2: Thêm các thành phần View trong trình soạn thảo bố cục

Trong tác vụ này, bạn tạo bố cục giao diện người dùng cho ứng dụng HelloToast trong trình chỉnh sửa bố cục bằng cách sử dụng [ConstraintLayout](#). Bạn có thể tạo các ràng buộc theo cách thủ công, như được hiển thị sau hoặc tự động sử dụng **công cụ A utocomplete**.

2.1 Kiểm tra các ràng buộc của phần tử

Làm theo các bước sau:

1. Mở ctivity_main.xml từ ngăn **P roject > Android** nếu nó chưa mở. Nếu **Tab Thiết kế** chưa được chọn, hãy nhấp vào nó.
Nếu không có bản thiết kế, hãy nhấp vào **nút Select Design Surface**  trên thanh công cụ và chọn **D esign + Blueprint**.
2. Công cụ **A utocomplete**  cũng nằm trong thanh công cụ. Nó được bật theo mặc định. 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 có cái nhìn cận cảnh.
4. Chọn **T extView** trong ngăn Cây thành phần. "Hello World" T extView được đánh dấu trong các ngăn thiết kế và bản thiết kế và các ràng buộc cho phần tử có thể nhìn thấy.
5. Tham khảo hình động bên dưới cho bước này. Nhấp vào tay cầm hình tròn ở phía bên phải của T extView để xóa ràng buộc ngang liên kết chế độ xem với phía bên phải của bố cục. T extView nhảy sang bên trái vì nó không còn bị ràng buộc ở phía bên phải. Để thêm lại ràng buộc ngang, hãy nhấp vào cùng một tay cầm và kéo một đường sang phía bên phải của bố cục.



Trong ngăn blueprint hoặc thiết kế, các bộ điều khiển sau xuất hiện trên phần tử TextView:

- **Xử lý ràng buộc:** Để tạo một ràng buộc như trong hình động ở trên, hãy nhấp vào một tay cầm ràng buộc, được hiển thị dưới dạng hình tròn ở cạnh của một phần tử. Sau đó, kéo tay cầm đến một tay cầm ràng buộc khác hoặc đến ranh giới mẹ. Một đường ngoằn ngoèo đại diện cho ràng buộc.



- **Thay đổi kích thước bộ điều khiển:** Để thay đổi kích thước phần tử, hãy kéo các bộ điều khiển thay đổi kích thước hình vuông. Tay cầm thay đổi thành một góc cạnh trong khi bạn đang kéo nó.

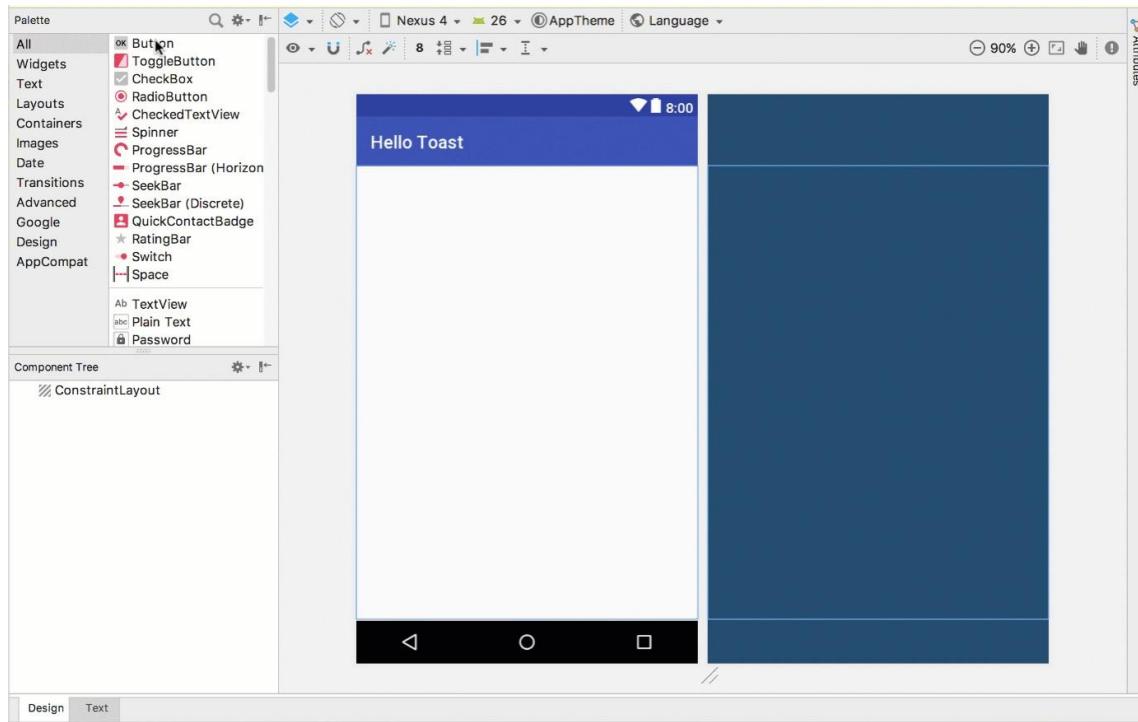


2.2 Thêm một nút vào bố cục

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

Làm theo các bước sau để thêm một Button:

1. Bắt đầu với một bảng sạch. Phần tử TextView là không cần thiết, vì vậy trong khi nó vẫn được chọn, nhấn phím **Delete** hoặc chọn **Edit > Delete**. Nay giờ bạn có một bố cục hoàn toàn trống.
2. Kéo một chữ **B** từ ngăn **Palette** đến bất kỳ vị trí nào trong bố cục. Nếu bạn thả chữ B 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 vào phía trên, bên trái và bên phải của bố cục như trong hình động bên dưới.



2.3 Thêm nút thứ hai vào bố cục

1. Kéo một chữ **B khác** từ ngăn **Palette** vào giữa bố cục như trong hình động bên dưới. Tự động kết nối có thể cung cấp các ràng buộc ngang cho bạn (nếu không, bạn có thể tự kéo chúng).
2. Kéo một ràng buộc dọc xuống cuối bố cục (tham khảo hình bên dưới).



Bạn có thể loại bỏ các ràng buộc khỏi một phần tử bằng cách chọn phần tử và di chuột con trỏ của bạn

trên đó để hiển thị nút Clear Constraints . Nhấp vào nút này để loại bỏ *các ràng buộc* *ll* trên phần tử đã chọn. Để xóa một ràng buộc duy nhất, hãy nhấp vào tay cầm cụ thể đặt ràng buộc.

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 rất hữu ích nếu bạn muốn làm lại tất cả các ràng buộc trong bố cục của mình.

Nhiệm vụ 3: Thay đổi thuộc tính phần tử giao diện người dùng

Ngăn A **ttributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng. Bạn có thể tìm thấy các thuộc tính (được gọi là *properties*) chung cho tất cả các chế độ xem trong [tài liệu lớp View](#).

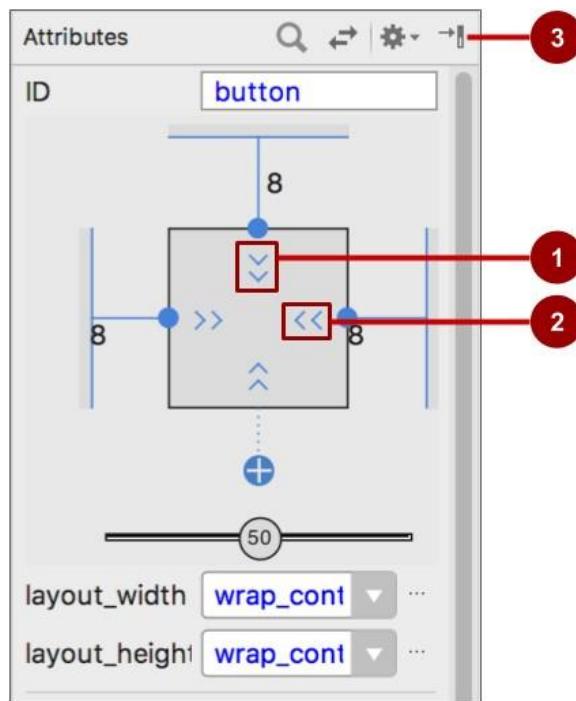
Trong nhiệm vụ này, bạn nhập các giá trị mới và thay đổi giá trị cho các thuộc tính quan trọng của B, có thể áp dụng cho hầu hết các loại V.

3.1 Thay đổi kích thước nút

Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước trên cả bốn góc của View để bạn có thể thay đổi kích thước View một cách nhanh chóng. Bạn có thể kéo các tay cầm trên mỗi góc của View để thay đổi kích thước của nó,

nhưng làm như vậy sẽ mã hóa cứng kích thước chiều rộng và chiều cao. Tránh mã hóa cứng kích thước cho hầu hết các phần tử View, vì kích thước được mã hóa cứng không thể thích ứng với các nội dung và kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn Phân bố A ở phía bên phải của trình chỉnh sửa bố cục để chọn chế độ kích thước không sử dụng kích thước được mã hóa cứng. **Ngăn Phân bố A** bao gồm một bảng điều chỉnh kích thước hình vuông được gọi là trình *kiểm tra chế độ xem* ở trên cùng. Các ký hiệu bên trong hình vuông đại diện cho cài đặt chiều cao và chiều rộng như sau:

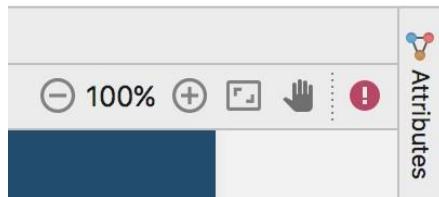


Trong hình trên:

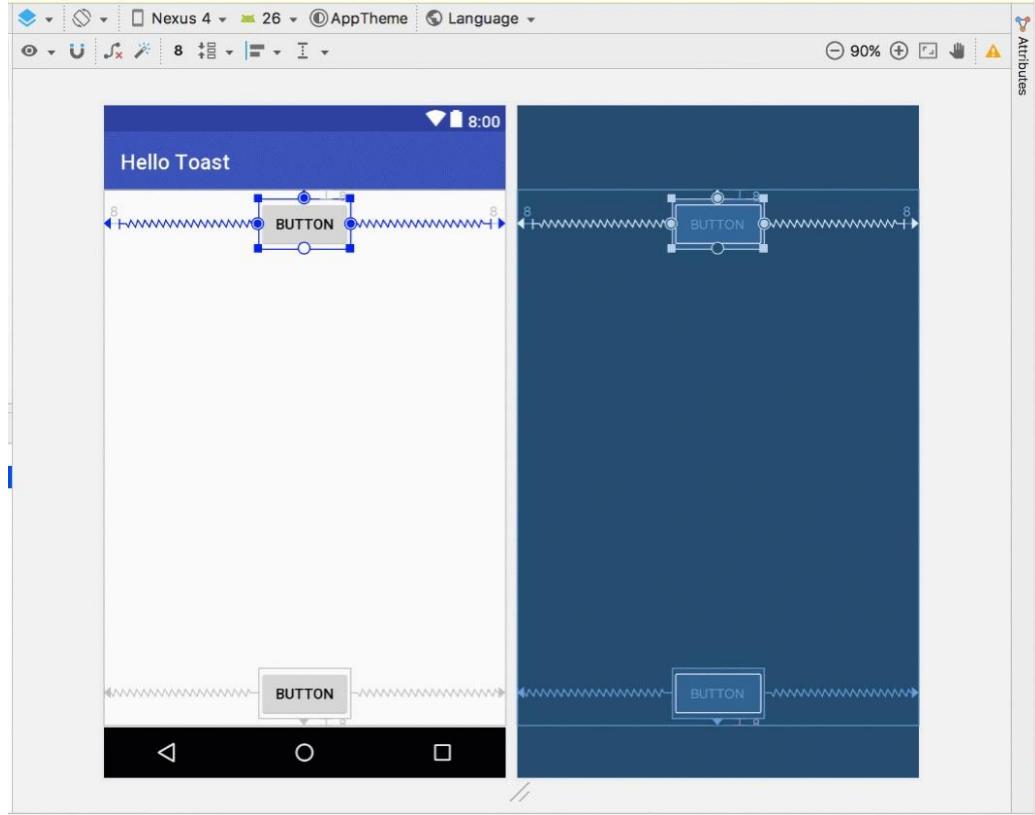
- Kiểm soát độ cao.** Điều khiển này chỉ định thuộc tính layout_height và xuất hiện trong hai phân đoạn ở cạnh trên và dưới của hình vuông. Các góc chỉ ra rằng điều khiển này được đặt thành wrap_content, có nghĩa là View sẽ mở rộng theo chiều dọc khi cần thiết để phù hợp với nội dung của nó. Dấu "8" cho biết ký quỹ tiêu chuẩn được đặt thành 8dp.
- Kiểm soát chiều rộng.** Điều khiển này chỉ định layout_width và xuất hiện trong hai phân đoạn ở bên trái và bên phải của hình vuông. Các góc chỉ ra rằng điều khiển này được đặt thành wrap_content, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần thiết để phù hợp với nội dung của nó, lên đến biên độ 8dp.
- Nút đóng ngăn Thuộc tính.** Bấm để đóng ngăn.

Làm theo các bước sau:

1. Chọn nút B trên cùng trong **ngăn Cây hỗ trợ C**.
2. Nhấp vào tab **A ttributes** ở phía bên phải của cửa sổ trình chỉnh sửa bộ cục.

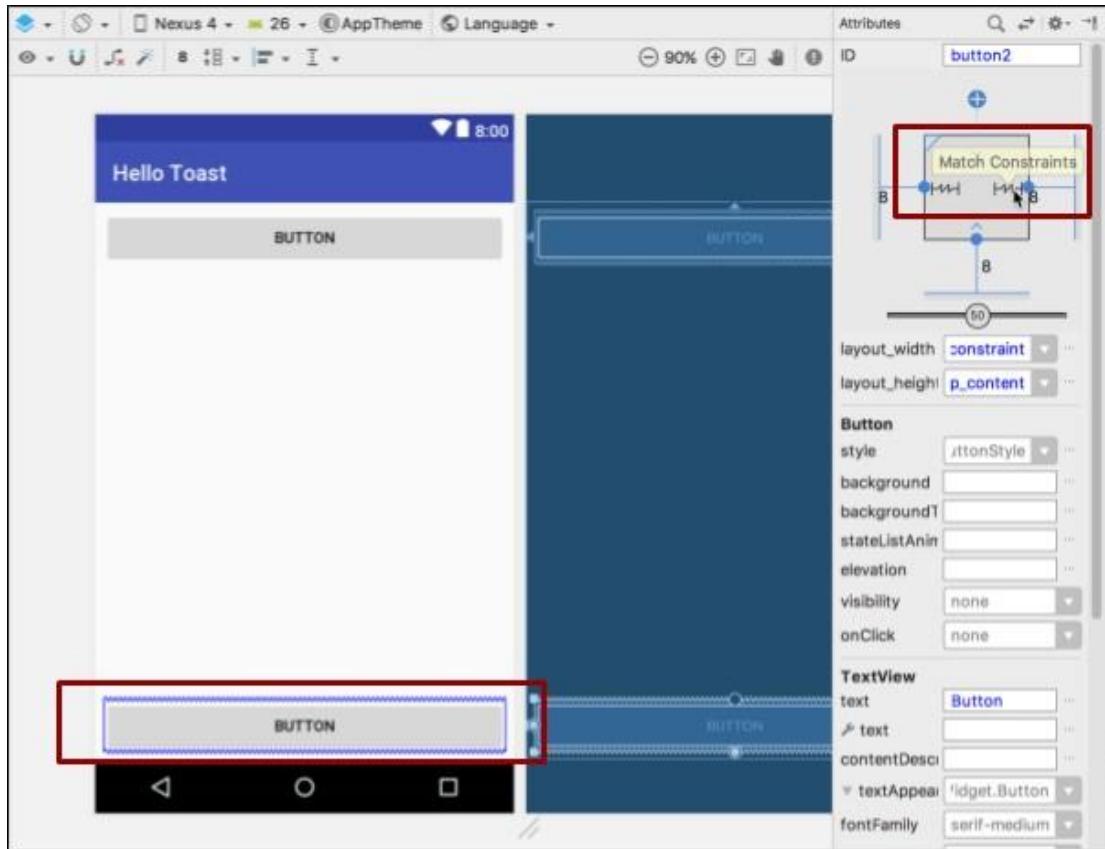


3. Nhấp vào điều khiển chiều rộng hai lần — lần nhấp đầu tiên thay đổi nó thành **F ixed** với các đường thẳng và lần nhấp thứ hai thay đổi nó thành **M atch Constraints** with spring coils, như thể hiện trong hình động bên dưới.



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

- Chọn nút B thứ hai và thực hiện các thay đổi tương tự đối với `layout_width` l như ở bước trước, như trong hình bên dưới.



Như được hiển thị trong các bước trước, các thuộc tính `layout_width` và `layout_height` **trong khung Phân bố A** 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 bộ cục, đó là `ConstraintLayout`:

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

Mẹo: Nếu bạn thay đổi thuộc tính layout_width bằng menu bật lên của nó, thuộc tính layout_width được đặt thành không vì không có thứ nguyên được đặt. Cài đặt này giống như match_constraint — chế độ xem có thể mở rộng càng nhiều càng tốt để đáp ứng các ràng buộc và cài đặt lè.

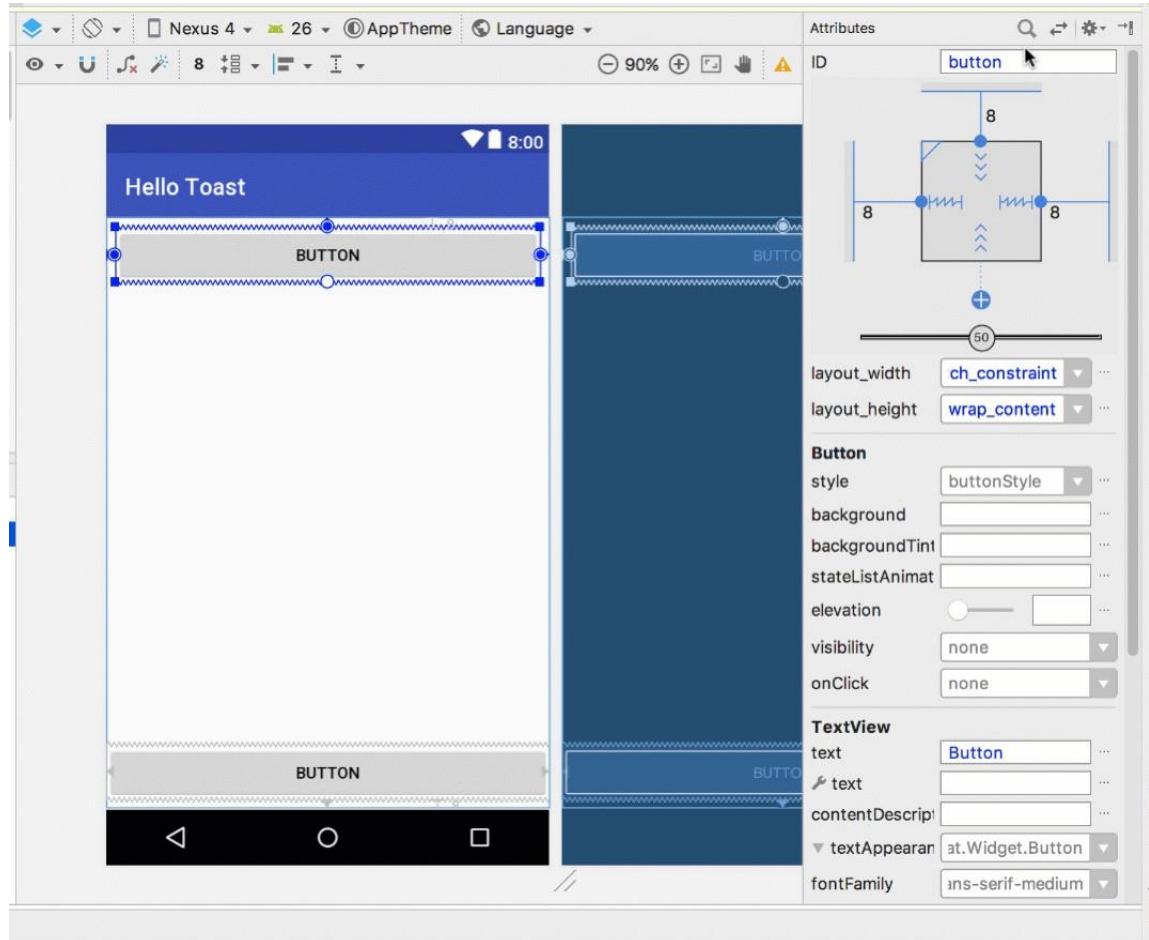
3.2 Thay đổi thuộc tính Nút

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

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

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

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



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

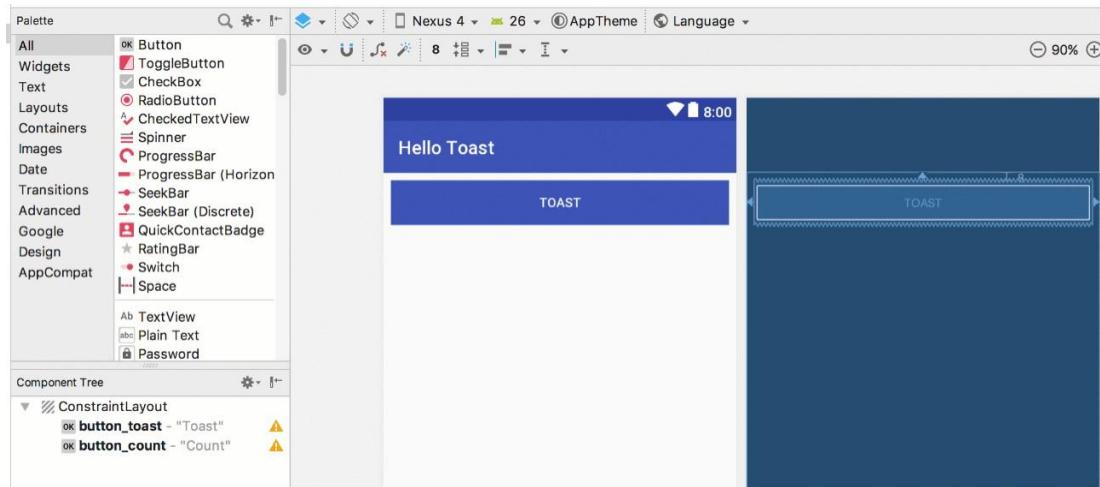
ColorPrimary là màu chính của chủ đề, một trong những màu cơ sở chủ đề được xác định trước được xác định trong tệp tài nguyên colors.xml. Nó được sử dụng cho thanh ứng dụng. Sử dụng màu cơ bản cho các yếu tố giao diện người dùng khác sẽ tạo ra một giao diện người dùng thống nhất. Bạn sẽ tìm hiểu thêm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

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

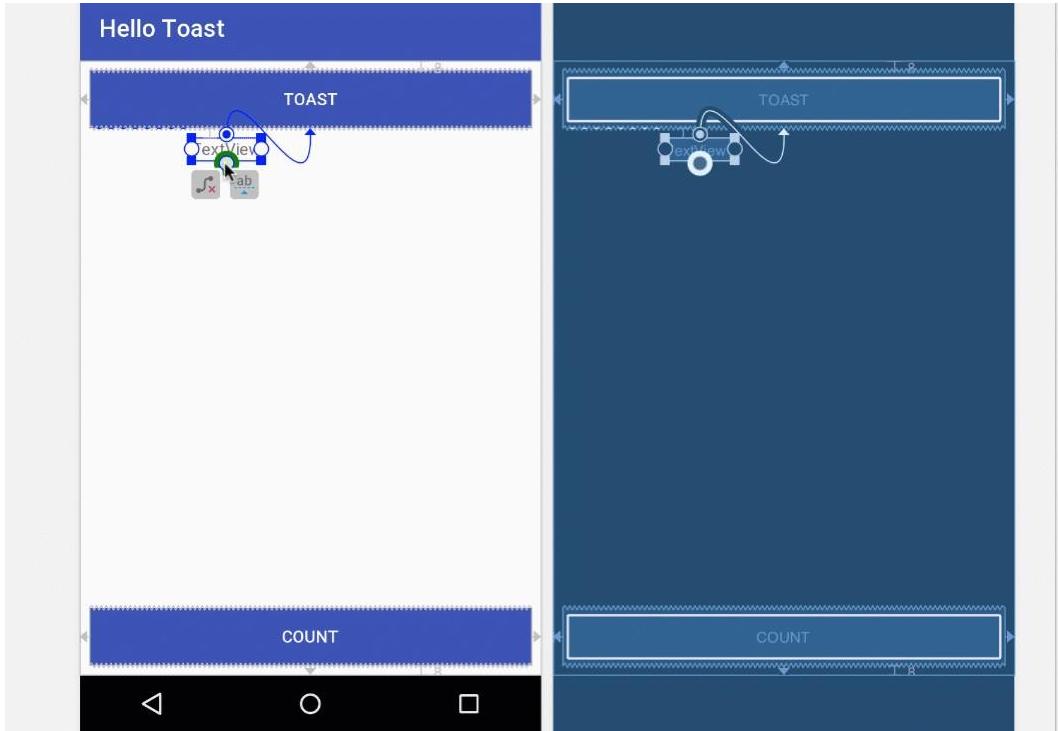
Một trong những lợi ích của ConstraintLayout là khả năng căn chỉnh hoặc hạn chế các phần tử liên quan đến các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một TextView ở giữa bố cục và hạn chế nó theo chiều ngang vào lề và theo chiều dọc với hai phần tử Button. Sau đó, bạn sẽ thay đổi các thuộc tính cho TextView trong ngăn Phân bố A.

4.1 Thêm TextView và các ràng buộc

- Như thể hiện trong hình động bên dưới, kéo TextView từ ngăn **Palette** đến phần trên của bố cục và kéo một ràng buộc từ trên cùng của TextView đến tay cầm ở dưới cùng của **Toast** Button. Điều này hạn chế TextView nằm bên dưới Button.



- Như thể hiện trong hình động bên dưới, kéo một ràng buộc từ dưới cùng của TextView đến tay cầm ở trên cùng của **Count** Button và từ các cạnh của TextView đến các cạnh của bố cục. Điều này hạn chế TextView ở giữa bố cục giữa hai phần tử Button.

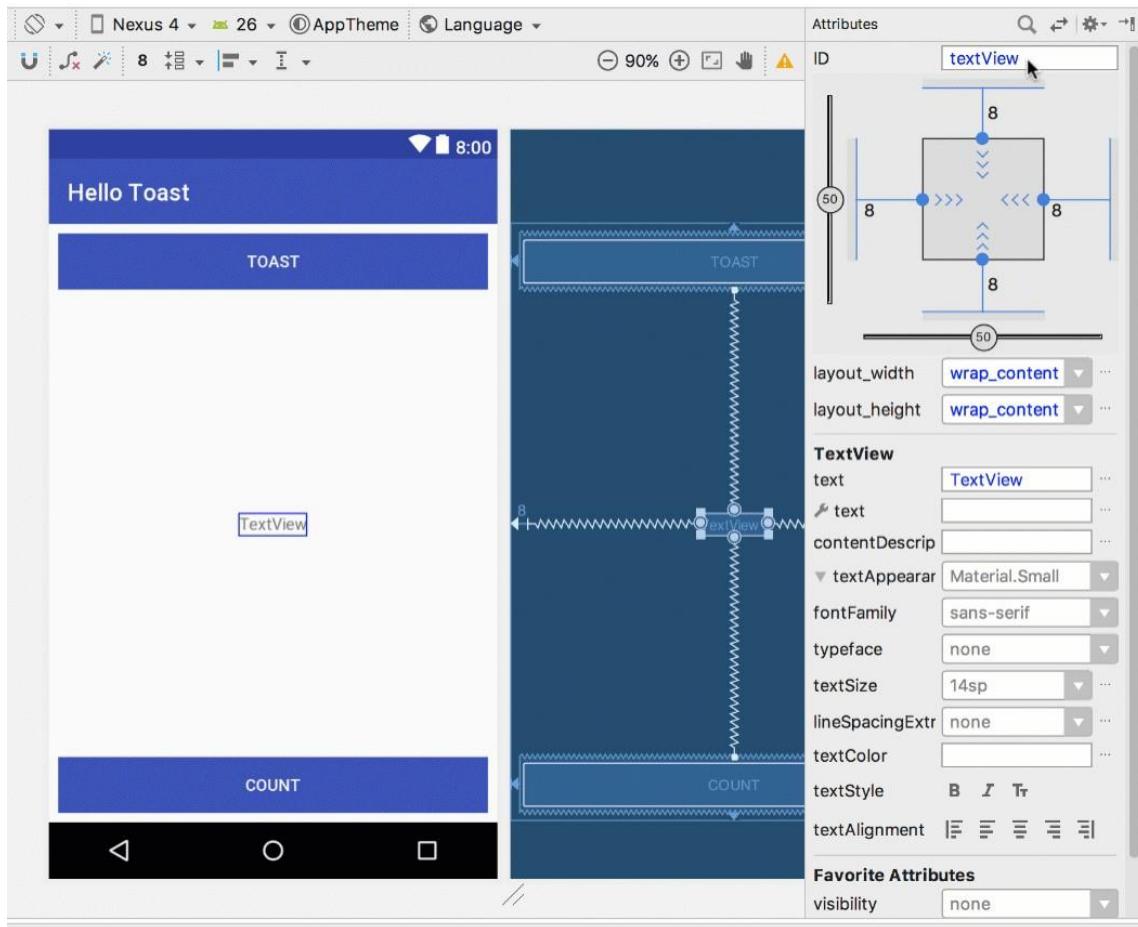


4.2 Đặt các thuộc tính TextView

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

1. Đặt ID thành **s how_count**.
2. Đặt text thành **0**.
3. Đặt textSize thành **16sp**.
4. Đặt textStyle thành **B** (in đậm) và textAlign thành **A LIGNCENTER** (căn giữa đoạn văn).
5. Thay đổi các điều khiển kích thước chê độ xem ngang và dọc (**layout_width** và **layout_height**) thành **match_constraint**.
6. Đặt textColor thành **@ color/colorPrimary**.

7. Cuộn xuống ngắn và nhấp vào View tất cả các thuộc tính, cuộn xuống trang thứ hai của các thuộc tính đến background, sau đó nhập # FFF00 cho màu vàng.
8. Cuộn xuống gravity, mở rộng gravity và chọn center_ver (đối với trung tâm-dọc).



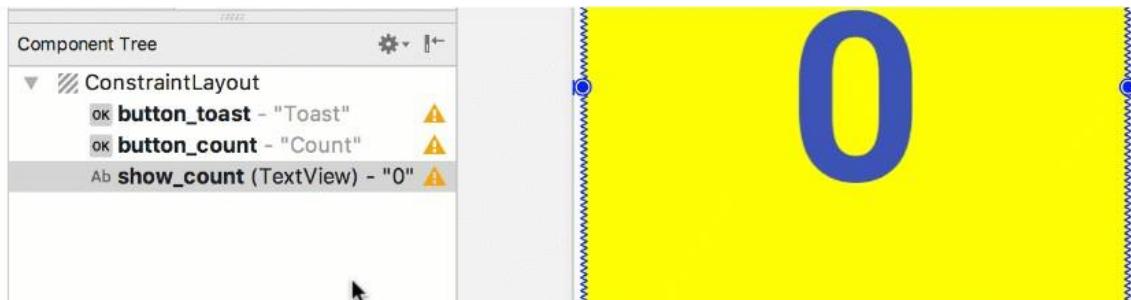
- textSize: Kích thước văn bản của TextView. Đối với bài học này, kích thước được đặt thành 160sp. Sp là viết tắt của *scale-independent pixel*, và giống như dp, là một đơn vị có tỷ lệ theo mật độ màn hình và sở thích kích thước phông chữ của người dùng. Sử dụng đơn vị dp khi bạn chỉ định kích thước phông chữ để kích thước được điều chỉnh cho cả mật độ màn hình và sở thích của người dùng.

- `textStyle` và `textAlignment`: Kiểu văn bản, được đặt thành **B** (in đậm) trong bài học này và căn chỉnh văn bản, được đặt thành **LIGNCENTER** (căn giữa đoạn văn).
- trọng lực: Thuộc tính `gravity` chỉ định cách một View được căn chỉnh trong `parent` của nó không phải là View hoặc ViewGroup. Trong bước này, bạn căn giữa TextView được căn giữa theo chiều dọc trong ConstraintLayout mẹ.

Bạn có thể nhận thấy rằng thuộc tính `background` nằm trên trang đầu tiên của ngăn Phân **bố A cho** Nút, nhưng trên trang thứ hai của ngăn Phân **bố A cho** TextView. Ngăn Phân **bố A** thay đổi cho từng loại View: Các thuộc tính phô biến nhất cho loại View xuất hiện trên trang đầu tiên và phần còn lại được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của ngăn A **Attributes**, hãy bấm vào biểu tượng trên thanh công cụ ở đầu ngăn.

Nhiệm vụ 5: Chính sửa bố cục trong XML

Bố cục 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 Cây thành phần. Di con trỏ của bạn qua các dấu chấm than này để xem 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 nên sử dụng tài nguyên.



Cách dễ nhất để khắc phục sự cố bố cục là chỉnh sửa bố cục trong XML. Mặc dù trình soạn thảo bố cục là một công cụ mạnh mẽ, nhưng một số thay đổi dễ dàng thực hiện trực tiếp trong mã nguồn XML.

5.1 Mở mã XML cho bố cục

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

Design Text
ở cuối trình chỉnh sửa bố cục.

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

The screenshot shows the Android Studio code editor with two tabs: 'activity_main.xml' and 'MainActivity.java'. The XML tab is active, displaying the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.hellotoast.MainActivity">

    <Button
        android:id="@+id/button_toast"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@color/colorPrimary"
        android:text="Toast"
        android:textColor="@android:color/white" />

    <Button
        android:id="@+id/button_count"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:background="@color/colorPrimary"
        android:text="Count"
        android:textColor="@android:color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/show_count"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginBottom="8dp" />
```

A yellow box highlights the string "Toast" in the first button's text attribute. A tooltip above the box reads: "Hardcoded string "Toast", should use @string resource [more...](#) (⌘F1)".

5.2 Trích xuất tài nguyên chuỗi

Thay vì mã hóa từng chuỗi, cách tốt nhất là sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Việc có các chuỗi trong một tệp riêng biệt giúp quản lý chúng dễ dàng 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 tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhập một lần vào từ "Toast" (cảnh báo đầu tiên được đánh dấu).
2. Nhấn **Alt-Enter** trong Windows hoặc **Option-Enter** trong macOS và chọn **Extract string resource** từ menu bật lên.
3. Nhập **button_label_toast** cho **tên nguồn điện tử R**.
4. Nhấp vào **OK**. 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 "Đếm" và `count_initial_value` cho "0".
6. Trong ngăn **Project > Android**, mở rộng `values` trong `res`, sau đó nhấp đúp vào `strings.xml` để xem tài nguyên chuỗi của bạn trong tệp `strings.xml`:

```
<tài nguyên>
<string name="app_name">Xin chào Toast</string>
<string name="button_label_toast">Toast</string>
<string name="button_label_count">Count</string>
<!-- tên chuỗi="count_initial_value">0</chuỗi>
</tài nguyên>
```

7. 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!":

```
<tài nguyên>
<string name="app_name">Xin chào Toast</string>
<string name="button_label_toast">Toast</string>
<string name="button_label_count">Count</string>
<!-- tên chuỗi="count_initial_value">0</chuỗi>
<string name="toast_message">Xin chào Toast!</string>
```

`</tài nguyên>`

Mẹo: Tài nguyên chuỗi bao gồm tên ứng dụng, tên này xuất hiện trong thanh ứng dụng ở đầu màn hình nếu bạn bắt đầu dự án ứng dụng của mình bằng Mẫu trống. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa tài nguyên `pp_name`.

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

Trong tác vụ này, bạn thêm một phương thức Java cho mỗi bài viết B trong MainActivity thực thi khi người dùng nhấp vào nút B.

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

Trình xử lý click là một phương thức được gọi khi người dùng nhấp hoặc nhấn vào phần tử giao diện người dùng có thể nhấp vào. Trong

Android Studio, bạn có thể chỉ định tên của phương thức trong trường `onClickListener` 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 B. Bạn sẽ sử dụng phương thức thứ hai 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. Với trình soạn thảo XML mở (tab Văn bản), tìm nút B với `android:id` được đặt thành `button_toast`:

```
< Nút
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    ...
    app:layout_constraintStart_toStartOf="cha mẹ"
    ứng dụng:layout_constraintTop_toTopOf="cha mẹ" />
```

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

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

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 **Create 'showToast(view)' in MainActivity** và nhấp vào **OK**

Hành động này tạo sơ khai 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 nút button_count B: Thêm thuộc tính android:onClick vào cuối và thêm trình xử lý nhấp chuột:

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

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

```
< Nút
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="@color/colorPrimary"
    android:text="@string/button_label_toast"
    android:textColor="@android:màu/trắng"
```

```
    app:layout_constraintEnd_toEndOf="cha mẹ"
    app:layout_constraintStart_toStartOf="cha mẹ" app:layout_constraintTop_toTopOf="cha mẹ"
    android:onClick="showToast"/>

< Nút
    android:id="@+id/button_count"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:text="@string/button_label_count"
    android:textColor="@android:màu/trắng"
    app:layout_constraintBottom_toBottomOf="cha mẹ"
    app:layout_constraintEnd_toEndOf="cha mẹ"
    app:layout_constraintStart_toStartOf="cha mẹ"
    android:onClick="countUp" />

<Ché độ xem văn bản
    android:id="@+id/show_count"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="#FFFF00"
    android:gravity="center_vertical"
    android:text="@string/count_initial_value"
    android:textAlignment="trung tâm"
    android:textColor="@color/colorPrimary"
    android:textSize="160sp"
    android:textStyle="đậm"
    ứng dụng:layout_constraintBottom_toTopOf="@+id/button_count"
    app:layout_constraintEnd_toEndOf="cha mẹ"
    app:layout_constraintStart_toStartOf="cha mẹ"
    ứng dụng:layout_constraintTop_toBottomOf="@+id/button_toast" />
```

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

```
gói com.example.android.hellotoast;
```

```
nhập android.support.v7.app.AppCompatActivity; nhập
android.os.Bundle; nhập android.view.View;

public class MainActivity extends AppCompatActivity {

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

    public void showToast(Xem chế độ xem) {
    }

    public void countUp(Xem chế độ xem) {
    }
}
```

6.2 Chính sửa trình xử lý Nút Toast

Bây giờ bạn sẽ chỉnh sửa phương thức `showToast()` — trình **xử lý nháp chuột** `Toast` `Button` 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 dung lượng cần thiết cho tin nhắn. 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 thông báo `Toast` để hiển thị kết quả chạm vào nút `B` hoặc thực hiện một hành động.

Làm theo các bước sau để chỉnh sửa trình **xử lý nháp chuột** `Toast` `B`:

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

```
public void showToast(Xem chế độ xem) {
}
```

2. Để tạo một thực thể của Toast, hãy gọi phương thức nhà máy makeText() trên lớp Toast. public void showToast(Xem chế độ xem) {

```
Bánh mì nướng = Toast.makeText(  
{}
```

Tuyên bố này không dày đủ cho đến khi bạn hoàn thành tất cả các bước.

3. Cung cấp Đồng nvariant của ứng dụng Một ctivity. Bởi vì một Toast hiển thị trên đầu Một ctivity UI, hệ thống cần thông tin về hiện tại Một ctivity. Khi bạn đã ở trong bối cảnh của Một ctivity ngữ cảnh bạn cần, sử dụng t của anh ấy như một lối tắt.

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

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

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

5. Cung cấp thời lượng cho màn hình. Ví dụ, Toast.LENGTH_SHORT hiển thị bánh mì nướng trong một thời gian tương đối ngắn.

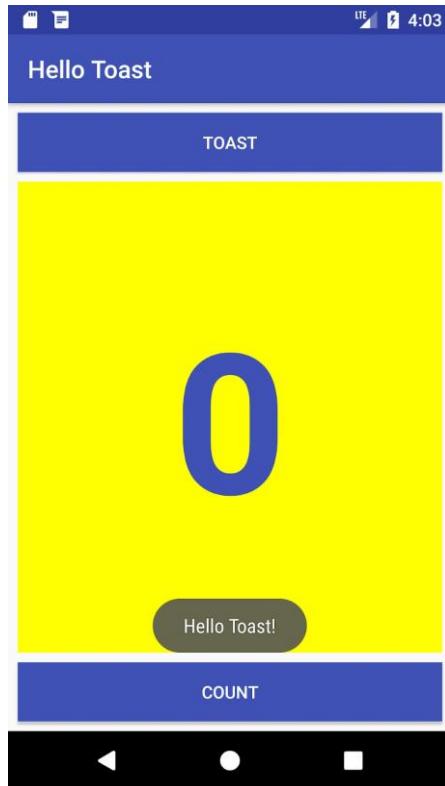
```
Toast toast = Toast.makeText(this, R.string.toast_message,  
Toast.LENGTH_SHORT);
```

Thời lượng của màn hình Toast có thể là Toast.LENGTH_LONG hoặc Toast.LENGTH_SHORT. Độ dài thực tế là khoảng 3.5 giây đối với Toast dài và 2 giây đối với Toast ngắn.

6. Hiển thị Toast bằng cách gọi [show\(\)](#). Sau đây là toàn bộ phương thức showToast():

```
public void showToast(Xem chế độ xem) {  
    Bánh mì nướng = Toast.makeText(this, R.string.toast_message, Toast.LENGTH_SHORT);  
    toast.show();  
}
```

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 chuột Count Button trong MainActivity — để nó hiển thị số lượng hiện tại sau khi nhấn vào C. Mỗi lần chạm sẽ tăng số lượng lên một. Mã cho trình xử lý phải:

- Theo dõi số lượng khi nó thay đổi.
- Gửi số lượng cập nhật đến TextView để hiển thị nó. Làm theo các bước sau để chỉnh sửa trình

xử lý nhấp chuột Count B:

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

```
public void countUp(Xem chế độ xem) {  
}
```

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

```
public void countUp(Xem chế độ xem) {  
    mĐếm ++;  
}
```

Nếu biểu tượng bóng đèn màu đỏ không xuất hiện, hãy chọn biểu thức mCount++. Bóng đèn màu đỏ cuối cùng cũng xuất hiện.

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

```
public class MainActivity extends AppCompatActivity {  
    tư nhân int mCount;
```

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

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

5. Cùng với biến trên, bạn cũng cần một biến thành viên riêng tư để tham chiếu của `show_count` `TextView`, mà bạn sẽ thêm vào trình xử lý nhấp chuột. Gọi biến này là `mShowCount`:

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

6. Bây giờ bạn đã có `m ShowCount`, bạn có thể lấy tham chiếu đến `TextView` bằng cách sử dụng ID bạn đã đặt trong tệp bố cục. Để có được tham chiếu này chỉ một lần, hãy chỉ định nó trong phương thức `onCreate()`. Như bạn đã học trong một bài học khác, phương thức `onCreate()` được sử dụng để *inflate* bố cục, có nghĩa là thiết lập chế độ xem nội dung của màn hình thành bố cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện người dùng khác trong bố cục, chẳng hạn như `TextView`. Xác định vị trí phương thức `onCreate()` trong `MainActivity`:

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

7. Thêm câu lệnh `findViewById` vào cuối phương thức:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
mShowCount = (TextView) findViewById(R.id.show_count); }
```

View, giống như một chuỗi, là một tài nguyên có thể có id. Lệnh `findViewById` lấy ID của một chế độ xem làm tham số của nó và trả về View. Bởi vì phương thức trả về một View, bạn phải chuyển kết quả sang kiểu View mà bạn mong đợi, trong trường hợp này là (TextView).

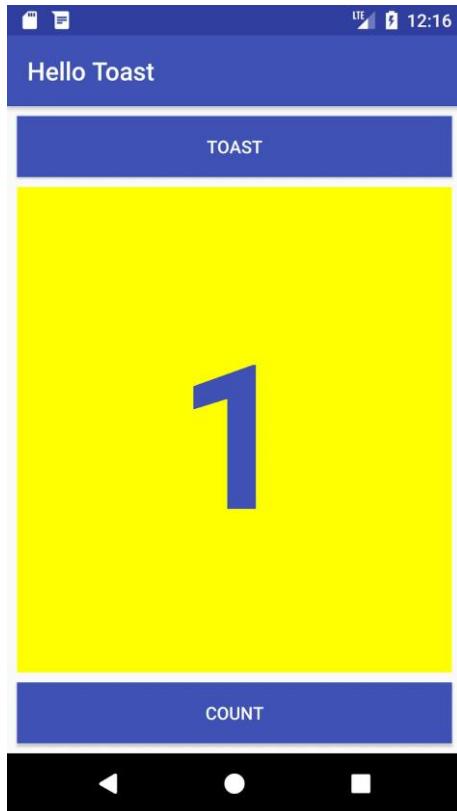
8. Bây giờ bạn đã gán cho m ShowCount TextView, bạn có thể sử dụng biến để đặt văn bản trong TextView thành giá trị của biến m Count. Thêm thông tin sau vào phương thức countUp():

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

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

```
public void countUp(Xem chế độ xem) {
    ++mĐếm;
    nếu (mShowCount != null)
        mShowCount.setText(Integer.toString(mCount)); }
```

9. Chạy ứng dụng để xác minh rằng số lượng tăng lên khi bạn nhấn vào nút C.



Mẹo: Để biết hướng dẫn chuyên sâu về cách sử dụng ConstraintLayout, hãy xem Lớp học lập trình [Using ConstraintLayout để thiết kế chế độ xem của bạn.](#)

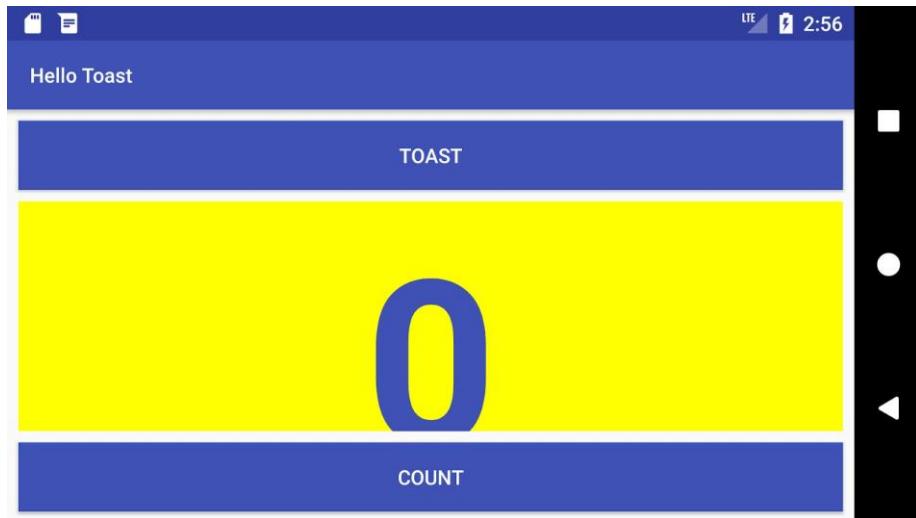
Mã giải pháp

Dự án Android Studio: [HelloToast](#)

Thử thách mã hóa

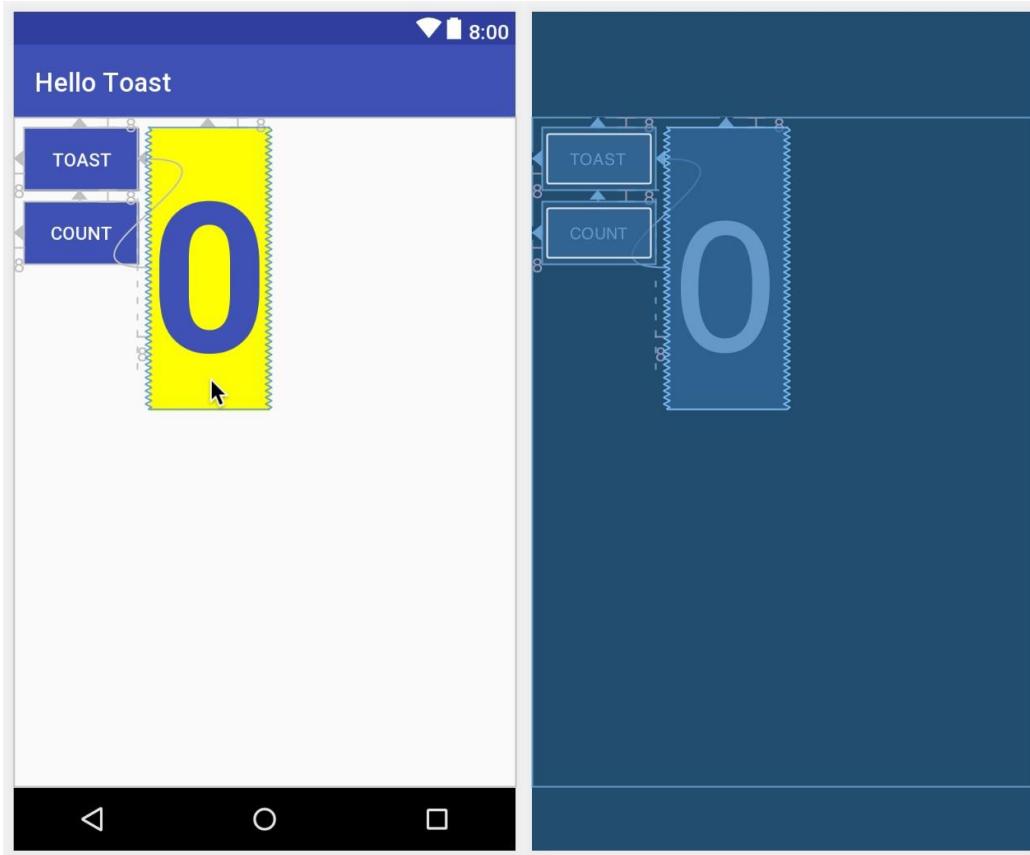
Lưu ý: Tất cả các thử thách mã hóa đều là tùy chọn và không phải là điều kiện tiên quyết cho các bài học sau này.

Ứng dụng HelloToast trông ổn khi thiết bị hoặc trình mô phỏng được định hướng theo chiều dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang hướng ngang, **nút Count Button** có thể chòng lên TextView dọc theo phía dưới như trong hình bên dưới.



Thử thách: Thay đổi bối cảnh sao cho nó trông đẹp ở cả hướng ngang và dọc:

1. Trên máy tính của bạn, tạo một bản sao của **thư mục dự án HelloToast** và đổi tên nó thành **HelloToastChallenge**.
2. Mở **HelloToastChallenge** trong Android Studio và tái cấu trúc nó. (Xem [Một ppendix: Tiện ích f](#) hoặc hướng dẫn về việc sao chép và tái cấu trúc một dự án.)
3. Thay đổi bối cảnh sao cho **chữ T oast Button** và **C ount Button** xuất hiện ở phía bên trái, như thể hiện trong hình bên dưới. TextView xuất hiện bên cạnh chúng, nhưng chỉ đủ rộng để hiển thị nội dung của nó. (Gợi ý: Sử dụng wrap_content.)
4. Chạy ứng dụng theo cả hướng ngang và dọc.





Mã giải pháp thách thức

Dự án Android Studio: [HelloToastChallenge](#)

Tóm tắt

View, ViewGroup và bộ cục:

- Tất cả các phần tử giao diện người dùng là các lớp con của [lớp View](#) và do đó kế thừa nhiều thuộc tính của lớp siêu View.
- Các phần tử View có thể được nhóm lại bên trong [ViewGroup](#), hoạt động như một vùng chứa. Mỗi quan hệ là cha mẹ-con, trong đó *p không phải* là ViewGroup và *c child* là View hoặc ViewGroup khác.

- Phương thực onCreate() được sử dụng để *inflate bố cục*, có nghĩa là đặt chế độ xem nội dung của màn hình thành bố cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các thành phần giao diện người dùng khác trong bố cục.
- V iew, giống như một chuỗi, là một tài nguyên có thể có id. Lệnh goi findViewById lấy ID của một chế độ xem làm tham số của nó và trả về V iew.

Sử dụng trình chỉnh sửa bố cục:

- Nhập vào tab **D esign** để thao tác với các phần tử và bố cục, và tab **T ext** để chỉnh sửa mã XML cho bố cục.
- Trong tab **D esign**, ngăn **P alettes** hiển thị các thành phần giao diện người dùng mà bạn có thể sử dụng trong bố cục của ứng dụng và ngăn **cây hỗ trợ** C hiển thị hệ thống phân cấp chế độ xem của các thành phần giao diện người dùng.
- Các ngăn thiết kế và bản thiết kế của trình chỉnh sửa bố cục hiển thị các yếu tố giao diện người dùng trong bố cục. ● Các **Một phần bố** hiển thị thẻ **Một phần bố** để đặt thuộc tính cho phần tử giao diện người dùng.
- Bộ điều khiển ràng buộc: Nhập vào một bộ điều khiển ràng buộc, được hiển thị dưới dạng hình tròn ở mỗi bên của một phần tử, sau đó kéo đến một bộ điều khiển ràng buộc khác hoặc đến ranh giới cha để tạo một ràng buộc. Ràng buộc được biểu thị bằng đường zigzag.
- Thay đổi kích thước bộ điều khiển: Bạn có thể kéo các bộ điều khiển thay đổi kích thước hình vuông để thay đổi kích thước phần tử. Trong khi kéo, tay cầm thay đổi thành một góc nghịch.
- Khi được bật, công cụ **Tự động kết nối** sẽ tự động tạo hai hoặc nhiều ràng buộc cho một phần tử giao diện người dùng đối với bố cục mẹ. Sau khi bạn kéo phần tử vào bố cục, nó sẽ tạo ra các ràng buộc dựa trên vị trí của phần tử.
- Bạn có thể loại bỏ các ràng buộc khỏi một phần tử bằng cách chọn phần tử và di chuột con trỏ trên nó để hiển thị nút Clear Constraints . Nhấp vào nút này để loại bỏ **các ràng buộc** ll trên phần tử đã chọn. Để xóa một ràng buộc duy nhất, hãy nhấp vào tay cầm cụ thể đặt ràng buộc.
- Ngăn A **ttributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng. Nó cũng bao gồm một bảng điều khiển kích thước hình vuông được gọi là thanh *trave* ở trên cùng. Các ký hiệu bên trong hình vuông đại diện cho cài đặt chiều cao và chiều rộng.

Đặt chiều rộng và chiều cao bố cục:

Các thuộc tính `layout_width` và `layout_height` thay đổi khi bạn thay đổi các điều khiển kích thước chiều cao và chiều rộng trong trình kiểm tra chế độ xem. Các thuộc tính này có thể lấy một trong ba giá trị cho `ConstraintLayout`:

- Cài đặt `match_constraint` mở rộng chế độ xem để lấp đầy khung nhìn mè của nó theo chiều rộng hoặc chiều cao—lên đến lè, nếu được đặt.
- Cài đặt `wrap_content` thu nhỏ kích thước chế độ xem để chế độ xem vừa đủ lớn để bao bọc nội dung của nó. Nếu không có nội dung, chế độ xem sẽ trở nên vô hình.
- Sử dụng một số `dip` cố định ([d pixel không phu thuộc vào độ nhạy cảm](#)) để chỉ định kích thước cố định, được điều chỉnh cho phù hợp với kích thước màn hình của thiết bị.

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. Làm theo các bước sau:

1. Nhập một lần vào chuỗi được mã hóa cứng để giải nén, nhấn **Alt-Enter (Option-Enter trên máy Mac)** và chọn **E xtract tài nguyên chuỗi** từ menu bật lên.
2. **Đặt tên nguồn điện tử R.**
3. Nhập vào **OK**. Thao tác này sẽ tạo ra một tài nguyên chuỗi 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`

Xử lý nhấp chuột:

- Trình `xử lý click` là một phương thức được gọi khi người dùng nhấp hoặc nhấn vào một phần tử giao diện người dùng.
- Chỉ định trình xử lý nhấp chuột cho một phần tử giao diện người dùng chẳng hạn như `Button` bằng cách nhập tên của nó vào trường `onClick` trong ngăn **phân bổ A** của tab **E sign D** hoặc trong trình chỉnh sửa XML bằng cách thêm thuộc tính `android:onClick` vào một phần tử giao diện người dùng như `Button`.
- Tạo trình xử lý nhấp chuột trong ctivity A chính bằng cách sử dụng tham số `View`. Ví dụ: `public void showToast(View view) {...}`.
- Bạn có thể tìm thấy thông tin về tất cả các thuộc tính `Button` trong [tài liệu lớp Button](#) và tất cả các thuộc tính `TextView` trong [tài liệu lớp TextView](#).

Hiển thị tin nhắn Toast:

T [Toast](#) cung cấp một cách để hiển thị một thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ lấp đầy dung lượng cần thiết cho tin nhắn. Để tạo một phiên bản của một chiếc yến mạch T, hãy làm theo các bước sau:

1. Gọi [phương thức nhà máy](#) `makeText()` trên lớp [Toast](#).
2. Cung cấp [Đòng văn bản](#) của ứng dụng Một ctivity và thông báo để hiển thi (chẳng hạn như tài nguyên chuỗi).
3. Cung cấp thời lượng của màn hình, ví dụ [Toast.LENGTH_SHORT](#) trong một thời gian ngắn. Thời lượng có thể là `Toast.LENGTH_LONG` hoặc `Toast.LENGTH_SHORT`.
4. Hiển thị `Toast` bằng cách gọi [show\(\)](#).

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [1.2: Bố cục và tài nguyên cho giao diện người dùng](#).

Tìm hiểu thêm

Tài liệu dành cho nhà phát triển Android:

- [Android Studio](#)
- [Xây dựng giao diện người dùng bằng Layout Editor](#)
- [Xây dựng giao diện người dùng đáp ứng với ConstraintLayout](#)
- [Bố trí](#)
- [Cánh](#)
- [Nút • TextView](#)
- [Tài nguyên Android](#)
- [Tài nguyên R.color chuẩn Android](#)
- [Hỗ trợ các mật độ khác nhau](#)
- [Sự kiện đầu vào Android](#)
- [Ngữ cảnh](#)

Khác:

- Lớp học lập trình: [Using ConstraintLayout để thiết kế chế độ xem của bạn](#)
- [Thuật ngữ từ vựng và khái niệm](#)

Lớp học lập trình tiếp theo là [Android fundamentals 1.2 Phần B: Trình chỉnh sửa bố cục](#).

Bài 1.2 Phần B: Trình soạn thảo bố cục

Giới thiệu

Như bạn đã học trong [1.2 Phần A: Giao diện người dùng tương tác đầu tiên của bạn](#), bạn có thể xây dựng giao diện người dùng (UI) bằng cách sử dụng

[ConstraintLayout](#) trong trình chỉnh sửa bố cục, đặt các phần tử giao diện người dùng trong bố cục bằng cách sử dụng kết nối ràng buộc với các phần tử khác và với các cạnh bố cục. ConstraintLayout được thiết kế để giúp bạn dễ dàng kéo các phần tử giao diện người dùng vào trình chỉnh sửa bố cục.

ConstraintLayout là một [ViewGroup](#), là một View đặc biệt có thể chứa các đối tượng View khác (được gọi là *children* hoặc *child views*). Thực tế này cho thấy nhiều tính năng hơn của ConstraintLayout và trình soạn thảo bố cục.

Thực tiễn này cũng giới thiệu hai [lớp con](#) ViewGroup khác:

- [LinearLayout](#): Một nhóm căn chỉnh các phần tử View con bên trong nó theo chiều ngang hoặc chiều dọc.
- [RelativeLayout](#): Một nhóm các phần tử View con trong đó mỗi phần tử View được định vị và căn chỉnh so với phần tử View khác trong ViewGroup. Vị trí của các phần tử View con được mô tả trong mối quan hệ với nhau hoặc với ViewGroup mẹ.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo ứng dụng Hello World bằng Android Studio.
- Chạy ứng dụng trên trình mô phỏng hoặc thiết bị.
- Tạo bố cục đơn giản cho ứng dụng bằng ConstraintLayout.
- Trích xuất và sử dụng tài nguyên chuỗi.

Những gì bạn sẽ học

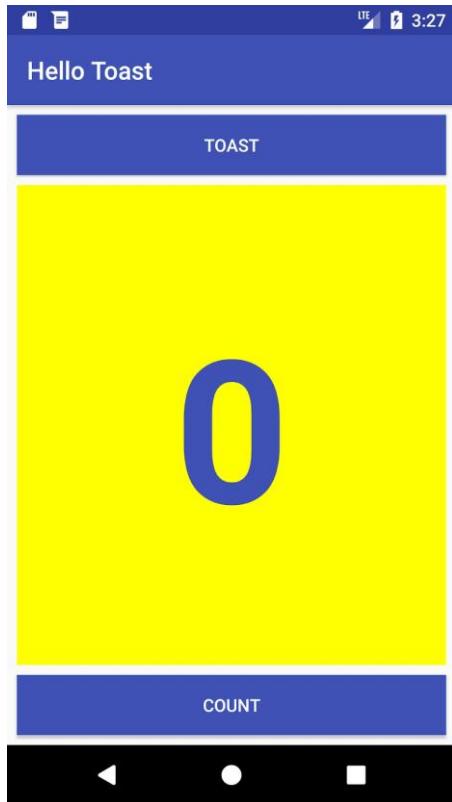
- Cách tạo một biến thể bô cục cho hướng ngang (ngang).
- Cách tạo biến thể bô cục cho máy tính bảng và màn hình lớn hơn.
- Cách sử dụng ràng buộc đường cơ sở để căn chỉnh các thành phần giao diện người dùng với văn bản.
- Cách sử dụng các nút gói và căn chỉnh để căn chỉnh các phần tử trong bô cục.
- Cách định vị chế độ xem trong L inearLayout.
- Cách định vị chế độ xem trong R elativeLayout.

Bạn sẽ làm gì

- Tạo biến thể bô cục cho hướng hiển thị ngang.
- Tạo biến thể bô cục cho máy tính bảng và màn hình lớn hơn.
- Sửa đổi bô cục để thêm ràng buộc vào các thành phần giao diện người dùng.
- Sử dụng các ràng buộc đường cơ sở C onstraintLayout để căn chỉnh các phần tử với văn bản.
- Sử dụng gói C onstraintLayout và căn chỉnh các nút để căn chỉnh các phần tử.
- Thay đổi bô cục để sử dụng L inearLayout.
- Định vị các phần tử trong L inearLayout.
- Thay đổi bô cục để sử dụng R elativeLayout.
- Sắp xếp lại các chế độ xem trong bô cục chính để tương đối với nhau.

Tổng quan về ứng dụng

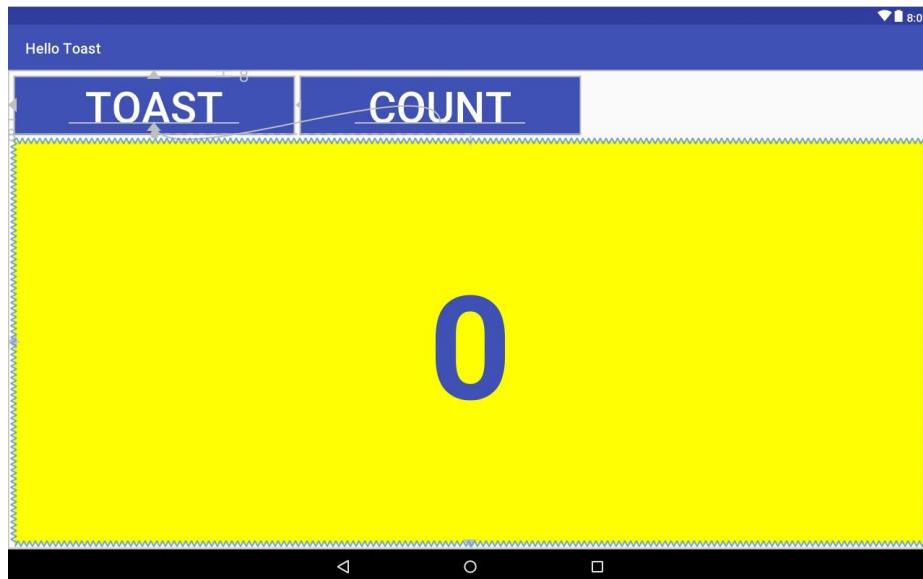
Ứng dụng Hello Toast trong bài học trước sử dụng [C onstraintLayout](#) để sắp xếp các phần tử giao diện người dùng trong bô cục Activity, như trong hình bên dưới.



Để thực hành thêm với ConstraintLayout, bạn sẽ tạo một biến thể của bộ cục này cho hướng ngang như trong hình bên dưới.



Bạn cũng sẽ học cách sử dụng các ràng buộc đường cơ sở và một số tính năng căn chỉnh của ConstraintLayout bằng cách tạo một biến thể bô cục khác cho màn hình máy tính bảng.

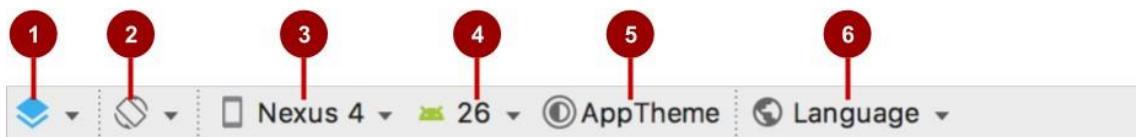


Bạn cũng tìm hiểu về các lớp con ViewGroup khác như [LinearLayout](#) và [RelativeLayout](#), đồng thời thay đổi bô cục ứng dụng Hello Toast để sử dụng chúng.

Nhiệm vụ 1: Tạo biến thể bố cục

Trong bài học trước, thử thách mã hóa yêu cầu thay đổi bố cục của ứng dụng Hello Toast để nó vừa vặn theo hướng ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học một cách dễ dàng hơn để tạo các biến thể của bố cục cho các hướng ngang (còn được gọi là *landscape*) và dọc (còn được gọi là *portrait*) cho điện thoại và cho màn hình lớn hơn như máy tính bảng.

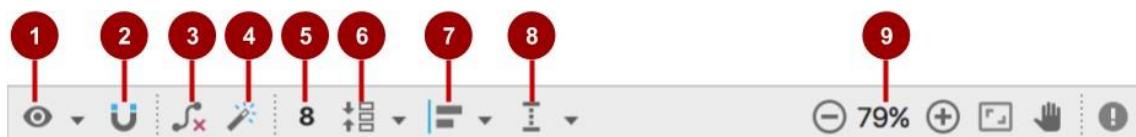
Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trong hai thanh công cụ trên cùng của trình soạn thảo bố cục. Thanh công cụ trên cùng cho phép bạn định cấu hình giao diện của bản xem trước bố cục trong trình chỉnh sửa bố cục:



Trong hình trên:

1. **Chọn Bố cục thiết kế:** Chọn **Design** để hiển thị bản xem trước màu của bố cục của bạn hoặc **Blueprint** để chỉ hiển thị đường viền cho từng phần tử giao diện người dùng. Để xem các ngăn khác cạnh nhau, hãy chọn **Design + Blueprint**.
2. **Định hướng trong Trình chỉnh sửa:** Chọn **Portrait** hoặc **Landscape** để hiển thị bản xem trước theo hướng dọc hoặc ngang. Điều này rất hữu ích để xem trước bố cục mà không cần phải chạy ứng dụng trên trình mô phỏng hoặc thiết bị. Để tạo bố cục thay thế, hãy chọn **Create Landscape Variation** hoặc các biến thể khác.
3. **Thiết bị trong Trình chỉnh sửa:** Chọn loại thiết bị (điện thoại / máy tính bảng, Android TV hoặc Android Wear).
4. **Phiên bản API trong Trình chỉnh sửa:** Chọn phiên bản Android để sử dụng để hiển thị bản xem trước.
5. **Chủ đề trong Trình chỉnh sửa:** Chọn một chủ đề (chẳng hạn như **AppTheme**) để áp dụng cho bản xem trước.
6. **Ngôn ngữ trong Trình chỉnh sửa:** Chọn ngôn ngữ và ngôn ngữ cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có sẵn trong tài nguyên chuỗi (xem bài học về bản địa hóa để biết chi tiết về cách thêm ngôn ngữ). Bạn cũng có thể chọn **Review là Từ phải sang trái** để xem bố cục như thẻ ngôn ngữ RTL đã được chọn.

Thanh công cụ thứ hai cho phép bạn định cấu hình giao diện của các phần tử giao diện người dùng trong **ConstraintLayout**, đồng thời thu phóng và xoay bản xem trước:



Trong hình trên:

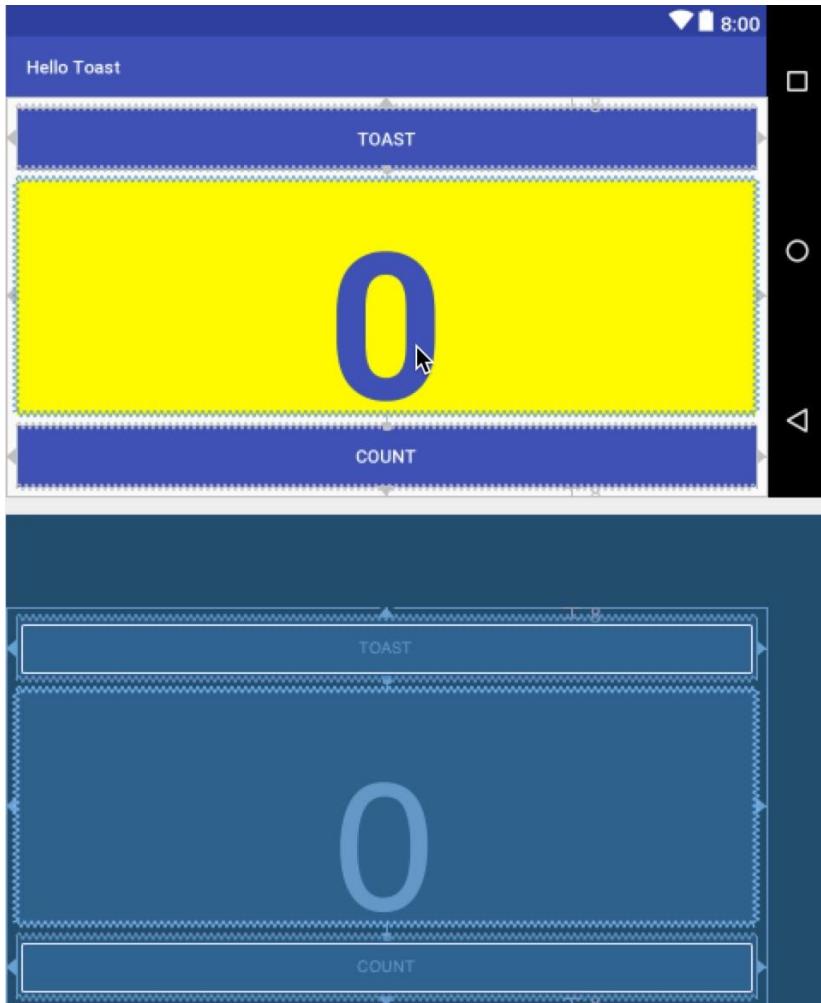
1. **Hiển thị:** Chọn **S cách ràng buộc** và **S cách Lề** để hiển thị chúng trong bản xem trước hoặc để dùng hiển thị chúng.
2. **Tự động kết nối:** Bật hoặc tắt Tự động kết nối. Khi bật Tự động kết nối, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như Button) vào bất kỳ phần nào của bố cục để tạo các ràng buộc đối với bố cục mẹ.
3. **Xóa tất cả các ràng buộc:** Xóa tất cả các ràng buộc trong toàn bộ bố cục.
4. **Suy ra các ràng buộc:** Tạo ràng buộc bằng suy luận.
5. **Lề mặc định:** Đặt lề mặc định.
6. **Đóng gói:** Đóng gói hoặc mở rộng các yếu tố đã chọn.
7. **Căn chỉnh:** Căn chỉnh các phần tử đã chọn.
8. **Nguyên tắc:** Thêm hướng dẫn đọc hoặc ngang.
9. **Điều khiển thu phóng/xoay:** Phóng to hoặc thu nhỏ.

Mẹo: Để tìm hiểu thêm về cách sử dụng trình chỉnh sửa bố cục, hãy xem B [sử dụng giao diện người dùng bằng Trình chỉnh sửa bố cục](#). Để tìm hiểu thêm về cách tạo bố cục bằng ConstraintLayout, hãy xem bài viết B [sử dụng giao diện người dùng đáp ứng với ConstraintLayout](#).

1.1 Xem trước bố cục theo hướng ngang

Để xem trước bố cục ứng dụng Hello Toast theo hướng ngang, hãy làm theo các bước sau:

1. Mở ứng dụng Hello Toast từ bài học trước.
2. Mở tệp bố cục ctivity_main.xml. Nhập vào **tab Design** nếu nó chưa được chọn.
3. Nhập vào **nút Orientation** trong Trình chỉnh sửa  trên thanh công cụ trên cùng.
4. Chọn **Portrait** trong menu thả xuống. Bố cục xuất hiện theo hướng ngang như hình dưới đây. Để quay lại hướng dọc, hãy chọn **Landscape**.



1.2 Tạo biến thể bố cục cho hướng ngang

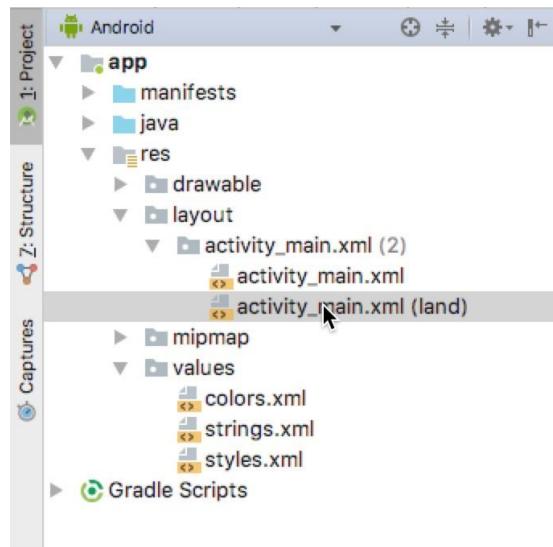
Sự khác biệt trực quan giữa hướng dọc và chiều ngang cho bố cục này là chữ số (0) trong phần tử `s how_count TextView` quá thấp đối với hướng ngang — quá gần với nút Đếm . Tùy thuộc vào thiết bị hoặc trình mô phỏng bạn sử dụng, phần tử `TextView` có thể xuất hiện quá lớn hoặc không căn giữa vì kích thước văn bản được cố định ở 160sp.

Để khắc phục điều này đối với các hướng ngang trong khi vẫn giữ nguyên hướng dọc, bạn có thể tạo biến thể của bộ cục ứng dụng Hello Toast khác với hướng ngang. Làm theo các bước sau:

1. Nhấp vào **nút Orientation** trong Trình chỉnh sửa  trên thanh công cụ trên cùng.
2. Chọn **Create Landscape Variation**.

Một cửa sổ trình chỉnh sửa mới mở ra với **tab layout/activity_main.xml** hiển thị bộ cục cho hướng ngang (ngang). Bạn có thể thay đổi bộ cục này, dành riêng cho hướng ngang, mà không cần thay đổi hướng dọc (dọc) ban đầu.

3. Trong ngăn **Project > Android**, hãy nhìn vào bên trong **thư mục bộ cục res >** và bạn sẽ thấy rằng Android Studio đã tự động tạo biến thể cho bạn, được gọi là **activity_main.xml (lnd)**.



1.3 Xem trước bộ cục cho các thiết bị khác nhau

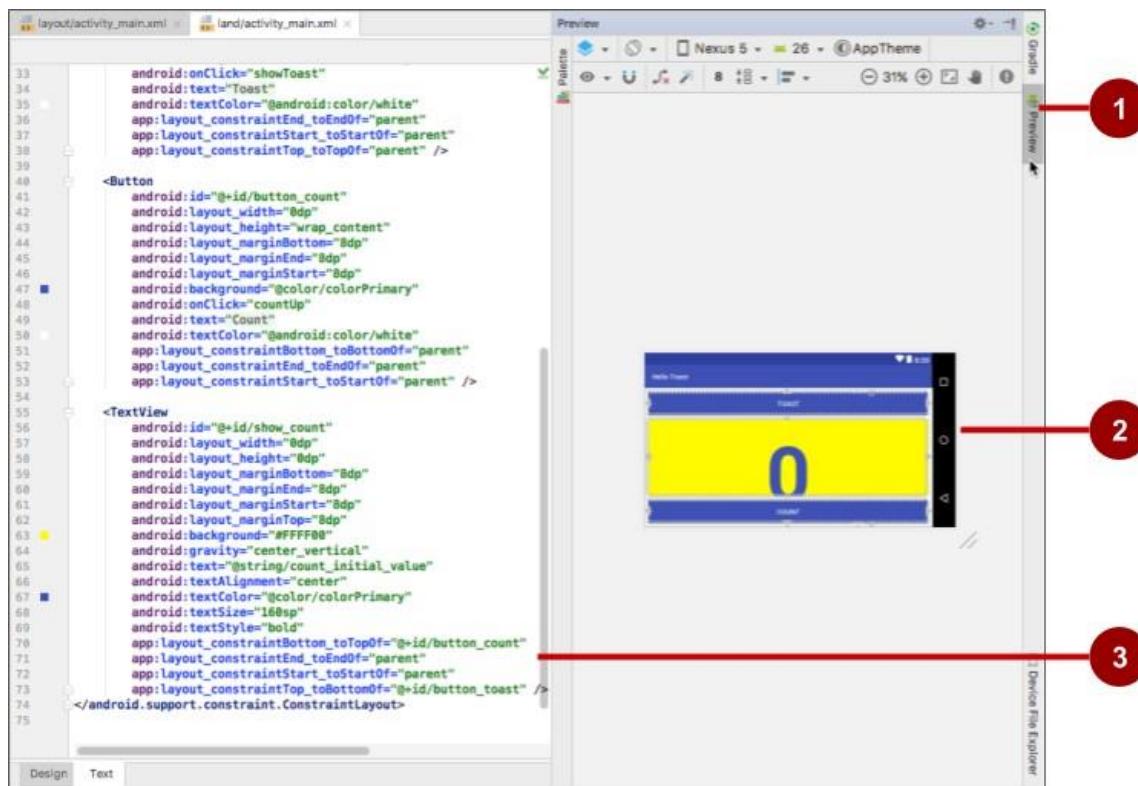
Bạn có thể xem trước bộ cục cho các thiết bị khác nhau mà không cần phải chạy ứng dụng trên thiết bị hoặc trình mô phỏng. Làm theo các bước sau:

1. Tab **land/activity_main.xml** vẫn sẽ được mở trong trình soạn thảo bộ cục; nếu không, hãy nhấp đúp vào **tệp activity_main.xml (lnd)** trong **thư mục layout**.

2. Nhấp vào **nút Device** trong Trình chỉnh sửa trên thanh công cụ trên cùng.
3. Chọn một thiết bị khác trong menu thả xuống. Ví dụ: chọn **Nexus 4**, **Nexus 5**, sau đó chọn **Pixel** để xem sự khác biệt trong bản xem trước. Những khác biệt này là do kích thước văn bản cố định cho **TextView**.

1.4 Thay đổi bố cục cho hướng ngang

Bạn có thể sử dụng ngăn Attributes trong **tab Design** để đặt hoặc thay đổi thuộc tính, nhưng đôi khi có thể nhanh hơn khi sử dụng tab **Text** để chỉnh sửa mã XML trực tiếp. Tab **Text** hiển thị mã XML và cung cấp tab **xem xét P** ở phía bên phải của cửa sổ để hiển thị bản xem trước bố cục, như trong hình bên dưới.



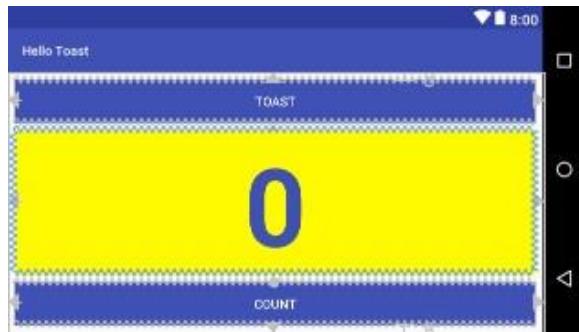
Hình trên cho thấy những điều sau:

1. Tab **Đánh giá P**, bạn sử dụng để hiển thị ngăn xem trước

2. Ngăn xem trước
3. Mã XML

Để thay đổi bố cục, hãy làm theo các bước sau:

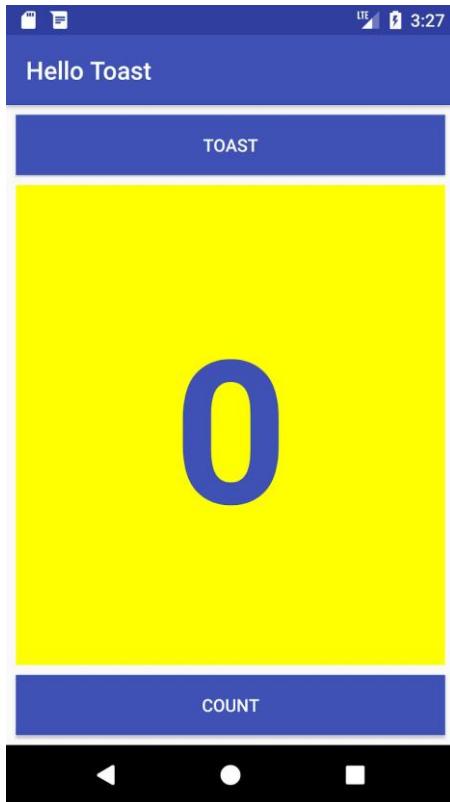
1. Tab **layout/activity_main.xml** vẫn sẽ được mở trong trình soạn thảo bố cục; nếu không, hãy nhấp đúp vào **tệp activity_main.xml (land)** trong **thư mục layout**.
2. Nhấp vào tab **Text** và tab **Preview** (nếu chưa được chọn).
3. Tìm phần tử **TextView** trong mã XML.
4. Thay đổi thuộc tính `android:textSize="160sp"` thành `android:textSize="120sp"`. Bạn xem trước bố cục hiển thị kết quả:



5. Chọn các thiết bị khác nhau trong menu thả xuống **Device** trong **Trình chỉnh sửa** để xem bố cục trông như thế nào trên các thiết bị khác nhau theo hướng ngang.

Trong ngăn trình chỉnh sửa, **tab layout/activity_main.xml** hiển thị bố cục cho hướng ngang. Tab **activity_main.xml** hiển thị bố cục không thay đổi cho hướng dọc. Bạn có thể chuyển đổi qua lại bằng cách nhấp vào các tab.

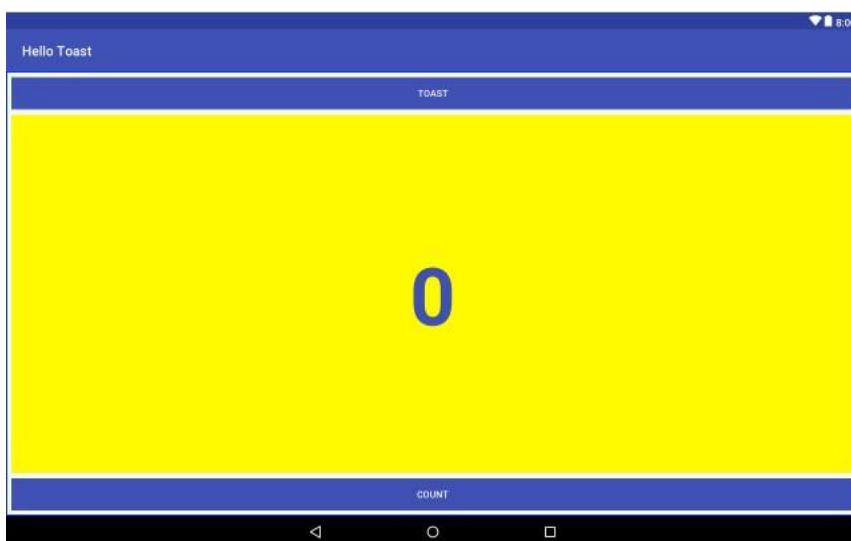
6. Chạy ứng dụng trên trình mô phỏng hoặc thiết bị và chuyển hướng từ dọc sang ngang để xem cả hai bố cục.



1.5 Tạo biến thể bố cục cho máy tính bảng

Như bạn đã học trước đây, bạn có thể xem trước bố cục cho các thiết bị khác nhau bằng cách nhấp vào nút **D** trong

Nút Trình chỉnh sửa **Nexus 5** trên thanh công cụ trên cùng. Nếu bạn chọn một thiết bị như **Nexus 10** (máy tính bảng) từ menu, bạn có thể thấy rằng bố cục không lý tưởng cho màn hình máy tính bảng — văn bản của mỗi chữ B quá nhỏ và sự sắp xếp của các yếu tố Button ở trên cùng và dưới cùng không lý tưởng cho máy tính bảng màn hình lớn.



Để khắc phục điều này cho máy tính bảng trong khi vẫn giữ nguyên hướng ngang và dọc kích thước điện thoại, bạn có thể tạo biến thể bố cục hoàn toàn khác cho máy tính bảng. Làm theo các bước sau:

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị các ngăn thiết kế và bản thiết kế.
2. Nhấp vào **nút Orientation** trong Trình chỉnh sửa trên thanh công cụ trên cùng.
3. Chọn **Create layout x-large Variation**.

Một cửa sổ trình chỉnh sửa mới mở ra với **tab x lớn / activity_main.xml** hiển thị bố cục cho thiết bị có kích thước máy tính bảng. Biên tập viên cũng chọn một thiết bị máy tính bảng, chẳng hạn như Nexus 9 hoặc Nexus 10, để xem trước. Bạn có thể thay đổi bố cục này, dành riêng cho máy tính bảng, mà không cần thay đổi các bố cục khác.

1.6 Thay đổi biến thể bố cục cho máy tính bảng

Bạn có thể sử dụng ngăn Thuộc tính trong **tab E sign D** để thay đổi thuộc tính cho bố cục này.

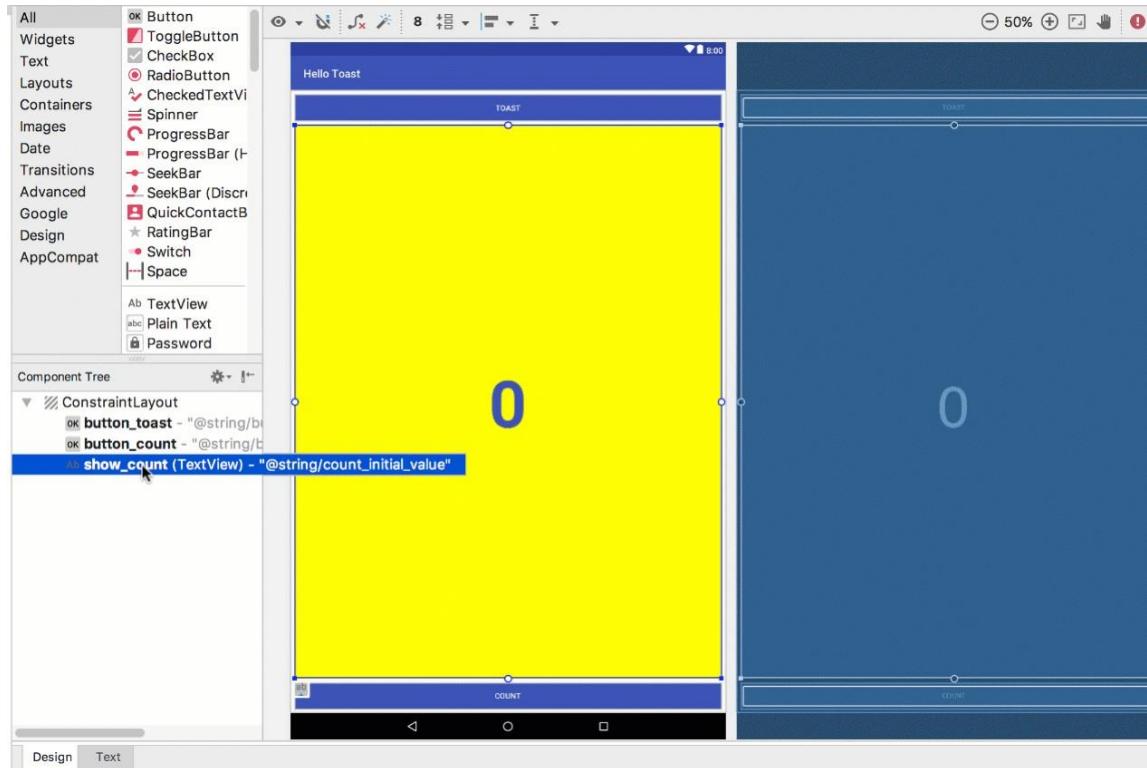
1. Tắt công cụ Tự động kết nối trên thanh công cụ. Đối với bước này, hãy đảm bảo rằng công cụ đã bị tắt:



2. Xóa tất cả các ràng buộc trong bố cục bằng cách nhấp vào **nút C lear All Constraints** trên thanh công cụ.

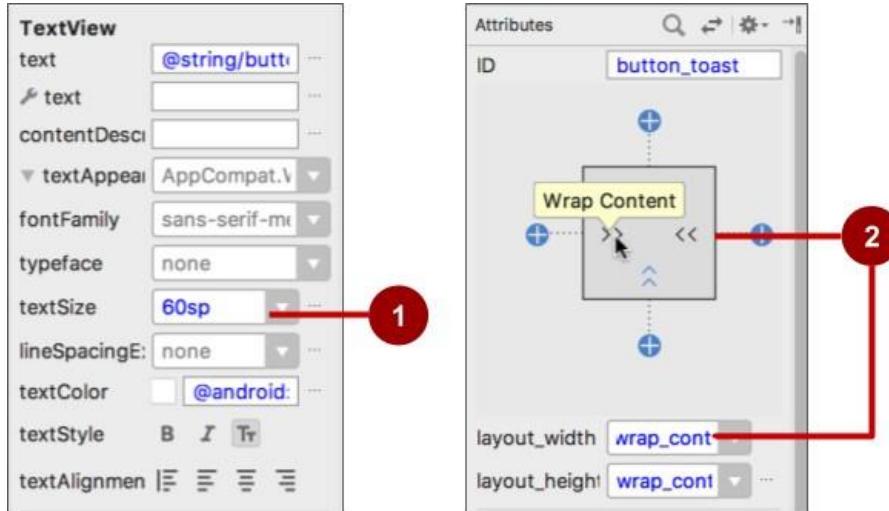
Với các ràng buộc được loại bỏ, bạn có thể di chuyển và thay đổi kích thước các phần tử trên bố cục một cách tự do.

3. Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước trên cả bốn góc của một phần tử để thay đổi kích thước của nó. Trong **Component Tree**, chọn **T extView** được gọi là **s how _count**. Để lấy **T extView** ra khỏi đường đi để bạn có thể tự do kéo các phần tử **B utton**, hãy kéo một góc của nó để thay đổi kích thước của nó, như được hiển thị trong hình động bên dưới.



Thay đổi kích thước phần tử mã hóa cùng kích thước chiều rộng và chiều cao. Tránh mã hóa cùng kích thước cho hầu hết các yếu tố, vì bạn không thể dự đoán kích thước được mã hóa cùng sẽ trông như thế nào trên màn hình có kích thước và mật độ khác nhau. Bạn đang làm điều này bây giờ chỉ để di chuyển phần tử ra khỏi đường và bạn sẽ thay đổi kích thước trong một bước khác.

- Chọn button_toast trong **Component Tree**, nhấp vào tab **Attributes** để mở ngăn **Attributes** và thay đổi `textSize` thành **60sp** (# 1 trong hình bên dưới) và `layout_width` thành **wrap_content** (# 2 trong hình bên dưới).



Như được hiển thị ở phía bên phải của hình trên (2), bạn có thể nhấp vào điều khiển chiều rộng của trình kiểm tra chế độ xem, xuất hiện trong hai phân đoạn ở bên trái và bên phải của hình vuông, cho đến khi nó hiển thị **Wrap Content**. Thay vào đó, bạn có thể chọn **wrap_content** từ menu **layout_width**.

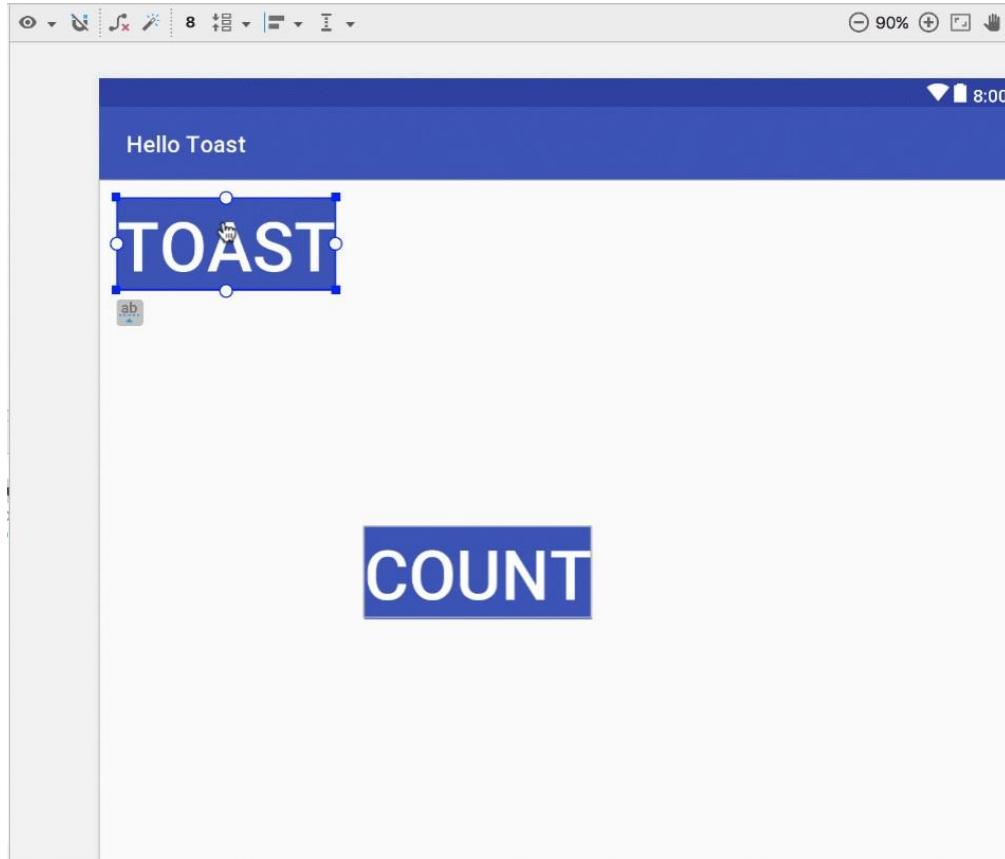
Bạn sử dụng **wrap_content** để nếu văn bản Button được bandle hóa sang một ngôn ngữ khác, Nút sẽ xuất hiện rộng hơn hoặc mỏng hơn để phù hợp với từ trong các ngôn ngữ khác nhau.

- Chọn nút `button_count` B trong **Component Tree**, thay đổi `textSize` thành **60sp** và `layout_width` thành **wrap_content** và kéo chữ B phía trên `TextView` đến một khoảng trống trong bố cục.

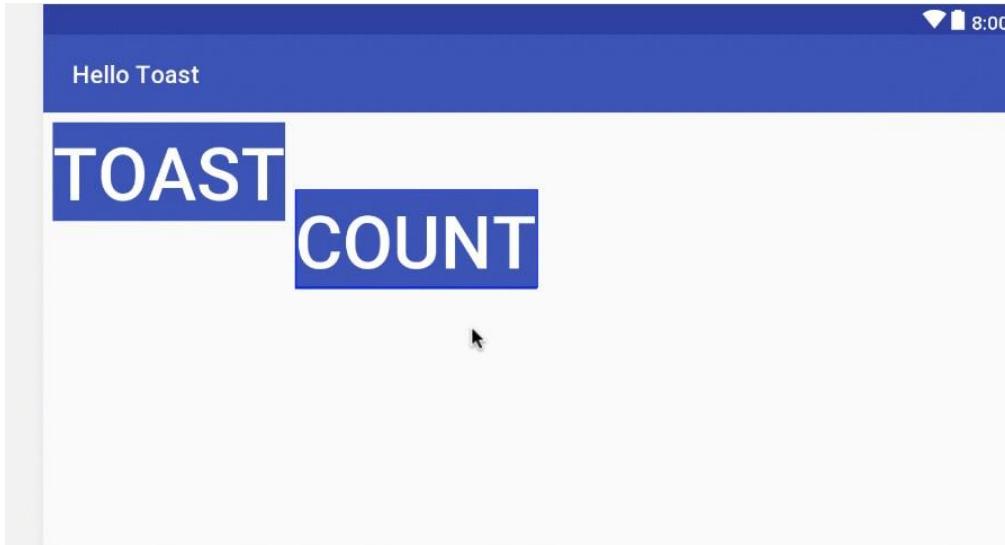
1.7 Sử dụng ràng buộc cơ sở

Bạn có thể căn chỉnh một phần tử giao diện người dùng có chứa văn bản, chẳng hạn như `TextView` hoặc `Button`, với một phần tử giao diện người dùng khác có chứa văn bản. **Ràng buộc baseline** cho phép bạn hạn chế các phần tử để các đường cơ sở văn bản khớp nhau.

- Hạn chế nút `button_toast` B ở phía trên và bên trái của bố cục, kéo nút `button_count` B đến một khoảng trống gần nút `button_toast` B và hạn chế `button_count` B với nút `button_toast` B ở phía bên trái của nút `button_toast` B, như thể hiện trong hình động:



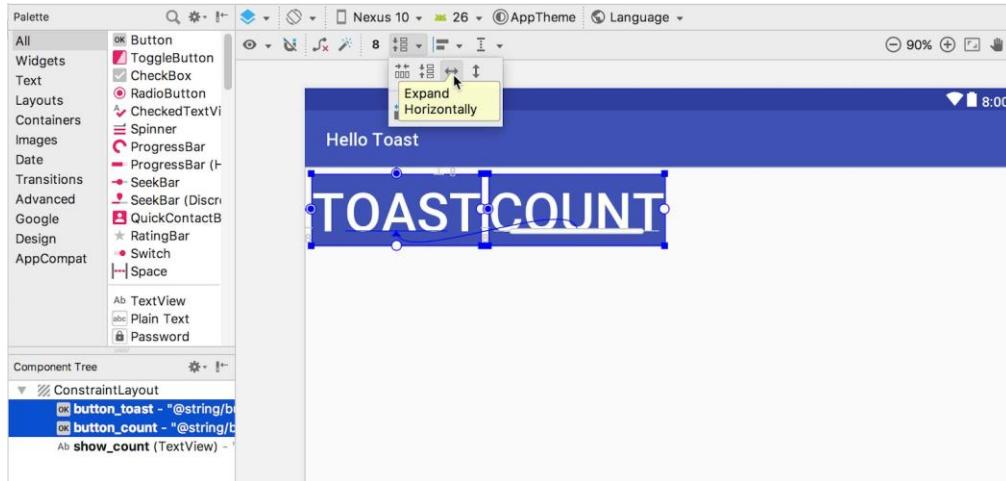
2. Sử dụng *ràng buộc baseline*, bạn có thể hạn chế button_count B utton sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của button_toast B utton. Chọn phần tử button_count , sau đó di con trỏ của bạn qua phần tử cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.
3. Nhấp vào nút ràng buộc đường cơ sở. Tay cầm cơ sở xuất hiện, nhấp nháy màu xanh lục như trong hình động. Nhấp và kéo một đường ràng buộc đường cơ sở đến đường cơ sở của phần tử button_toast.



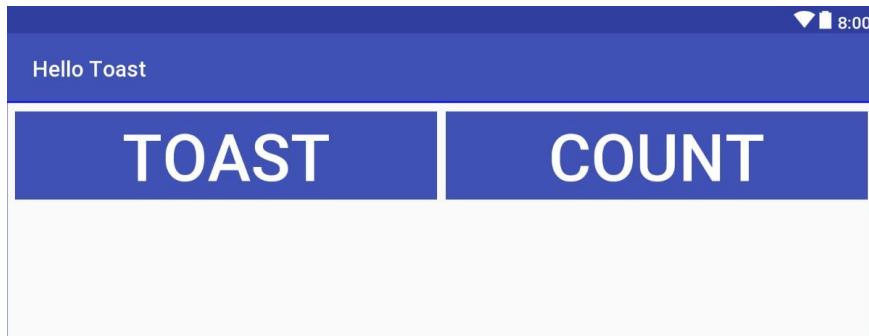
1.8 Mở rộng các nút theo chiều ngang

Nút đóng gói trên thanh công cụ cung cấp các tùy chọn để đóng gói hoặc mở rộng các phần tử giao diện người dùng đã chọn. Bạn có thể sử dụng nó để sắp xếp đều các phần tử B utton theo chiều ngang trên bố cục.

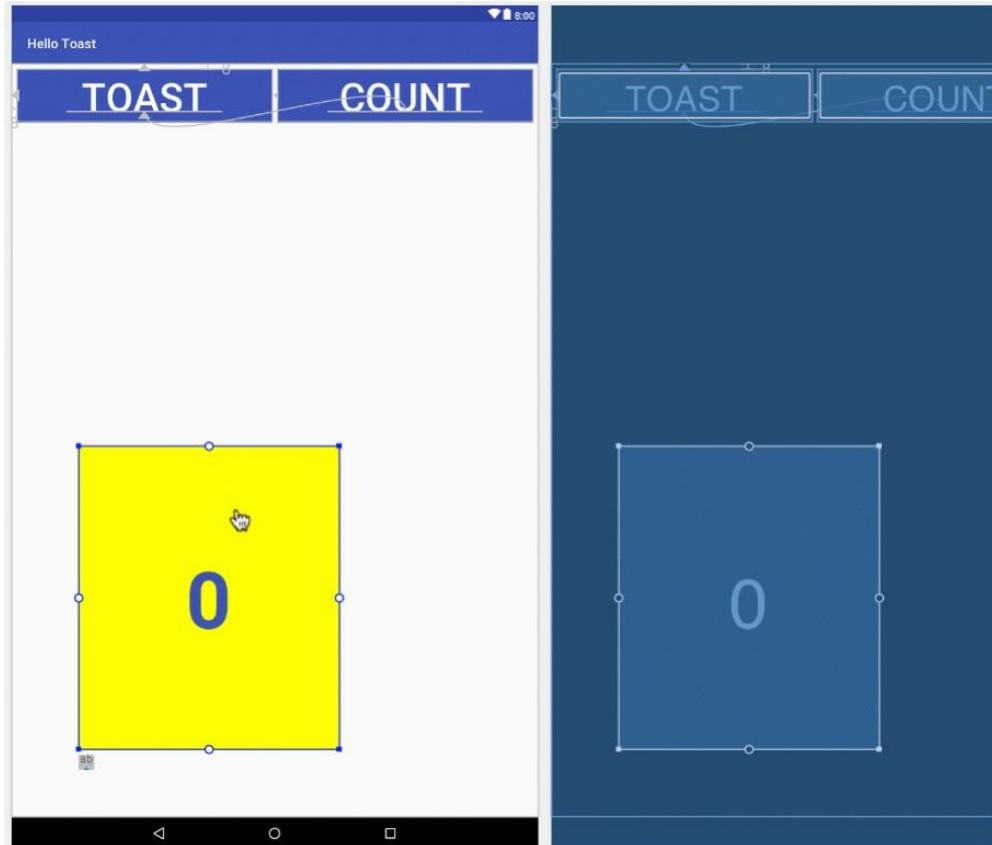
1. Chọn nút b utton_count B trong **Cây đồng thời C** và Shift-chọn nút b utton_toast để cả hai đều được chọn.
2. Nhấp vào nút gói trên thanh công cụ và chọn **E xpand Horizontal** như trong hình bên dưới.



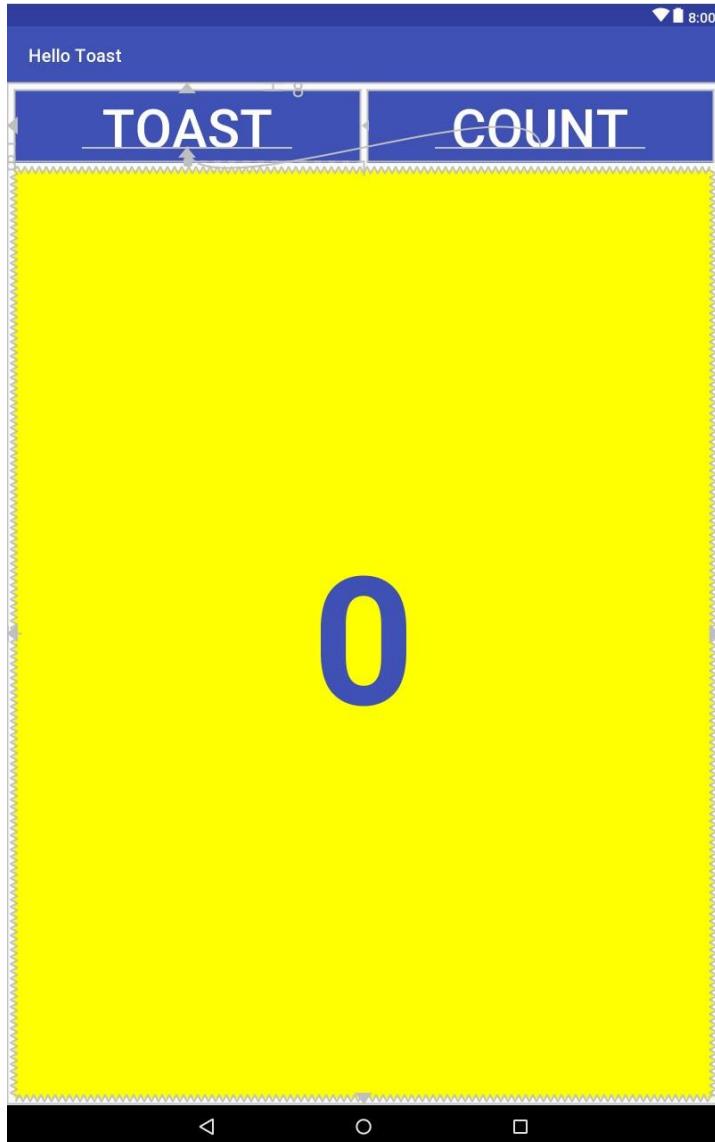
Các phần tử Button mở rộng theo chiều ngang để lắp đầy bố cục như hình dưới đây.



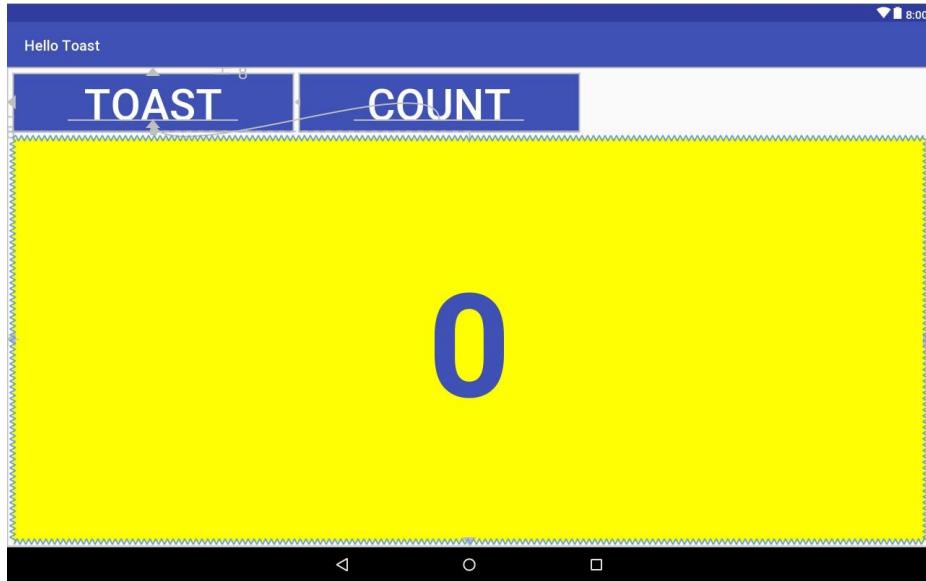
3. Để hoàn thành bố cục, hãy ràng buộc s how_count TextView ở cuối b utton_toast
- Nút và ở hai bên và dưới cùng của bố cục, như thể hiện trong hình động bên dưới.



4. Các bước cuối cùng là thay đổi `s how_count TextView layout_width` và `layout_height` thành **Match Constraints** và `textSize` thành **200sp**. Bố cục cuối cùng trông giống như hình bên dưới.



5. Nhấp vào **nút Orientation in Editor** ở thanh công cụ trên cùng và chọn **Switch to Landscape**. Bố cục máy tính bảng xuất hiện theo hướng ngang như hình dưới đây. (Bạn có thể chọn **phù thủy Switch Chân dung** để trở lại hướng dọc.).



- Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng sau khi chạy ứng dụng để xem ứng dụng trông như thế nào trên các loại thiết bị khác nhau. Bạn đã tạo thành công một ứng dụng có thể chạy với giao diện người dùng phù hợp trên điện thoại và máy tính bảng có kích thước và mật độ màn hình khác nhau.

Mẹo: Để biết hướng dẫn chuyên sâu về cách sử dụng ConstraintLayout, hãy xem [Using ConstraintLayout để thiết kế chế độ xem của bạn](#).

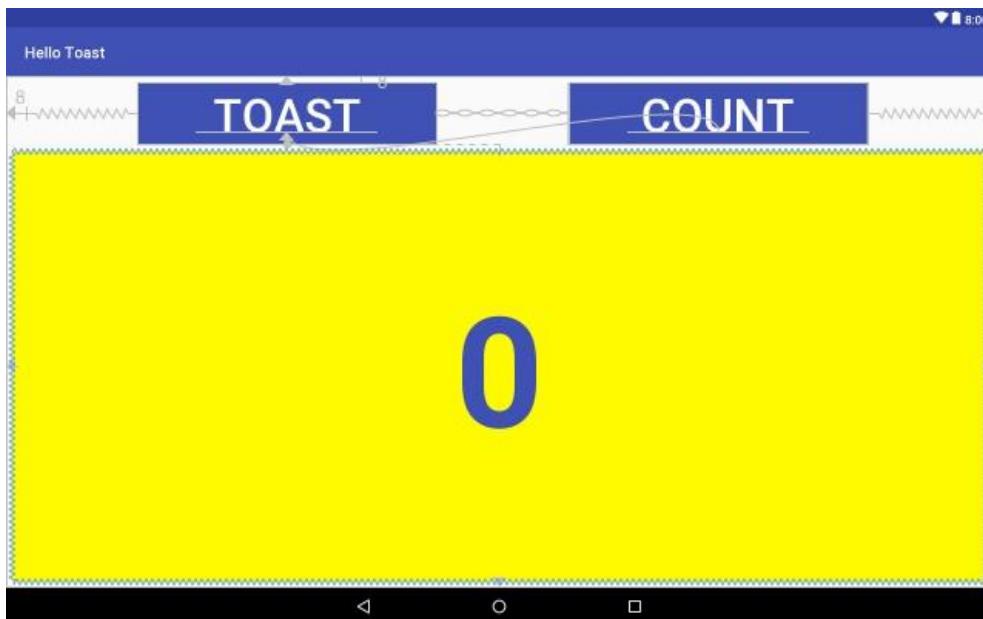
Mã giải pháp nhiệm vụ 1

Dự án Android Studio: [HelloToast](#)

Thử thách lập trình 1

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 này.

Thách thức: Để phù hợp với hướng ngang (ngang) cho máy tính bảng, bạn có thể căn giữa Các phần tử nút trong ctivity_main.xml (xlarge) để chúng xuất hiện như trong hình bên dưới. **Gợi ý:** Chọn các phần tử, nhập vào nút căn chỉnh trên thanh công cụ và chọn **C enter theo chiều ngang**.



Mã giải pháp thử thách 1

Dự án Android Studio: [HelloToastChallenge2](#)

Nhiệm vụ 2: Thay đổi bố cục thành LinearLayout

LinearLayout là một ViewGroup sắp xếp bộ sưu tập các chế độ xem của nó trong một hàng ngang hoặc dọc. LinearLayout là một trong những bố cục phổ biến nhất vì nó đơn giản và nhanh chóng. Nó thường được sử dụng trong một nhóm chế độ xem khác để sắp xếp các phần tử giao diện người dùng theo chiều ngang hoặc chiều dọc.

Cần có LinearLayout để có các thuộc tính sau:

- layout_width
- layout_height
- Hướng

layout_width và layout_height có thể lấy một trong các giá trị sau:

- match_parent: Mở rộng chế độ xem để lấp đầy chế độ xem chính của nó theo chiều rộng hoặc chiều cao. Khi LinearLayout là chế độ xem gốc, nó sẽ mở rộng đến kích thước của màn hình (chế độ xem mẹ).
- wrap_content: Thu nhỏ kích thước chế độ xem để chế độ xem vừa đủ lớn để bao bọc nội dung của nó. Nếu không có nội dung, chế độ xem sẽ trở nên vô hình.
- Số lượng dp cố định ([đơn vị cảm không phụ thuộc vào điểm ảnh](#)): Chỉ định kích thước cố định, được điều chỉnh cho mật độ màn hình của thiết bị. Ví dụ: 16dp có nghĩa là 16 pixel không phụ thuộc vào mật độ.

Oriantation có thể là:

- ngang: Chế độ xem được sắp xếp từ trái sang phải.
- dọc: Chế độ xem được sắp xếp từ trên xuống dưới.

Trong tác vụ này, bạn sẽ thay đổi nhóm chế độ xem gốc ConstraintLayout cho ứng dụng Hello Toast thành LinearLayout để bạn có thể thực hành sử dụng LinearLayout.

2.1 Thay đổi nhóm chế độ xem gốc thành LinearLayout

1. Mở ứng dụng Hello Toast từ tác vụ trước đó.
2. Mở tệp bố cục activity_main.xml (nếu chưa mở) và bấm vào tab **Văn bản** ở cuối ngăn chỉnh sửa để xem mã XML. Ở trên cùng của mã XML là dòng thẻ sau:

```
<android.support.constraint.ConstraintLayout xmlns:android="http://...>
```

3. Thay đổi thẻ android.support.constraint.ConstraintLayout < thành <L inearLayout để mã trông như sau:

```
<LinearLayout xmlns:android="http:..."
```

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

```
    android:orientation="dọc"
```

Sau khi thực hiện những thay đổi này, một số thuộc tính XML cho các phần tử khác được gạch chân màu đỏ vì chúng được sử dụng với ConstraintLayout và không liên quan đến Linearlayout.

2.2 Thay đổi thuộc tính phần tử cho LinearLayout

Làm theo các bước sau để thay đổi các thuộc tính phần tử giao diện người dùng để chúng hoạt động với Linearlayout:

1. Mở ứng dụng Hello Toast từ tác vụ trước đó.
2. Mở tệp bố cục activity_main.xml (nếu nó chưa được mở) và nhấp vào tab Text.
3. Tìm phần tử button_toast Button và thay đổi thuộc tính sau:

Gốc	Thay đổi thành
android:layout_width="0dp"	android:layout_width="match_parent"

4. Xóa các thuộc tính sau khỏi phần tử button_toast:

app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent"

5. Tìm phần tử button_count Button và thay đổi thuộc tính sau:

Gốc	Thay đổi thành
android:layout_width="0dp"	android:layout_width="match_parent"

6. Xóa các thuộc tính sau khỏi phần tử button_count:

app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintEnd_toEndOf="parent" app:layout_constraintStart_toStartOf="parent"

7. Tìm phần tử show_count TextView và thay đổi các thuộc tính sau:

Gốc	Thay đổi thành
android:layout_width="0dp"	android:layout_width="match_parent"
android:layout_width="0dp"	android:layout_height="wrap_content"

8. Xóa các thuộc tính sau khỏi phần tử show_count:

```
app:layout_constraintBottom_toTopOf="@+id/button_count" app:layout_constraintEnd_toEndOf="parent"  
app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toBottomOf="@+id/button_toast"
```

9. Nhập vào tab **P review** ở phía bên phải của cửa sổ Android Studio (nếu chưa được chọn) để xem trước bố cục cho đến nay:



2.3 Thay đổi vị trí của các phần tử trong LinearLayout

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

Để thay đổi vị trí của chúng sao cho nút **C ount** ở dưới cùng, hãy làm theo các bước sau:

1. Mở ứng dụng Hello Toast từ tác vụ trước đó.

2. Mở tệp bố cục ctivity_main.xml (nếu nó chưa được mở) và nhấp vào tab **Văn bản**.
3. Chọn nút button_count Button và tất cả các thuộc tính của nó, từ thẻ Nút < đến và bao gồm thẻ đóng /> và chọn **E dit > Cut**.
4. Nhấp sau thẻ đóng /> của phần tử TextView nhưng trước khi đóng </LinearLayout> và chọn **E dit > Paste**.
5. (Tùy chọn) Để khắc phục bất kỳ vấn đề thụt lề hoặc khoảng cách nào cho mục đích thẩm mỹ, hãy chọn **C ode > Định dạng lại Mã** để định dạng lại mã XML với khoảng cách và thụt lề thích hợp.

Mã XML cho các phần tử giao diện người dùng bây giờ trông như sau:

```
<Nút
    android:id="@+id/button_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="@color/colorPrimary"
    android:onClick="hiển thị bánh mì nướng"
    android:text="@string/button_label_toast"
    android:textColor="@android:color/white" />

<Ché độ xem văn bản
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="#FFFF00"
    android:gravity="center_vertical"
    android:text="@string/count_initial_value"
    android:textAlignment="trung tâm"
    android:textColor="@color/colorPrimary"
    android:textSize="160sp"
    android:textStyle="bold" />

<Nút
    android:id="@+id/button_count"
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:onClick="đếm"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white" />
```

Bằng cách di chuyển nút button_count B bên dưới TextView, bỏ cục bảy giờ giàn với những gì bạn đã có trước đây, với nút **C ount** ở phía dưới. Bạn xem trước của bỏ cục bảy giờ trông như sau:



2.4 Thêm trọng lượng cho phần tử TextView

Chỉ định gravity và weight thuộc tính cho phép bạn kiểm soát việc sắp xếp các chế độ xem và nội dung trong LinearLayout.

Thuộc tính android:gravity chỉ định sự liên kết của nội dung của View trong chính View. Trong bài học trước, bạn đặt thuộc tính này cho s how_count TextView để căn giữa nội dung (chữ số 0) ở giữa Văn bảnView:

```
android:gravity="center_vertical"
```

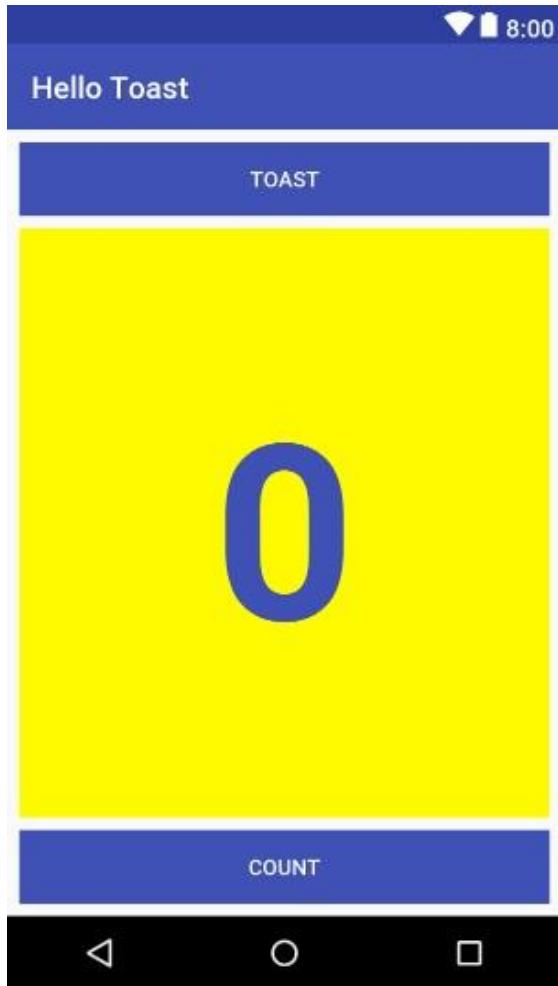
Thuộc tính android:layout_weight trong chỉ định bao nhiêu không gian bổ sung trong LinearLayout sẽ được phân bổ cho View. Nếu chỉ có một View có thuộc tính này, nó sẽ nhận được tất cả không gian màn hình bổ sung. Đối với nhiều phần tử View, không gian được chia theo tỷ lệ. Ví dụ: nếu mỗi phần tử Button có trọng số là 1 và TextView 2, tổng cộng là 4, thì các phần tử Button nhận được 1/4 khoảng trống mỗi phần và một nửa TextView.

Trên các thiết bị khác nhau, bố cục có thể hiển thị phần tử s how_count TextView như lấp đầy một phần hoặc hầu hết khoảng trống giữa các nút **T oast** và **C**. Để mở rộng TextView để lấp đầy không gian có sẵn bất kể thiết bị nào được sử dụng, hãy chỉ định thuộc tính android:gravity cho TextView. Làm theo các bước sau:

1. Mở ứng dụng Hello Toast từ tác vụ trước đó.
2. Mở tệp bố cục ctivity_main.xml (nếu nó chưa được mở) và nhập vào tab **Văn bản**.
3. Tìm phần tử s how_count TextView và thêm thuộc tính sau:

```
android:layout_weight="1"
```

Bản xem trước bây giờ trông giống như hình sau.



Phần tử `CountView` chiếm tất cả không gian giữa các nút. Bạn có thể xem trước bố cục cho các thiết bị khác nhau, như bạn đã làm trong tác vụ trước bằng cách nhấp vào nút **Device** trong **Trình chỉnh sửa** **Nexus 5** trên thanh công cụ trên cùng của ngăn xem trước và chọn một thiết bị khác. Không vấn đề gì thiết bị nào bạn chọn cho bản xem trước, phần tử `CountView` sẽ chiếm tất cả không gian giữa các nút.

Mã giải pháp nhiệm vụ 2

Mã XML trong một ctivity_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="dọc"
    tools:context="com.example.android.hellotoast.MainActivity">

    < Nút
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@color/colorPrimary"
        android:onClick="hiển thị bánh mì nướng"
        android:text="@string/button_label_toast"
        android:textColor="@android:color/white" />

    <Ché độ xem văn bản
        android:id="@+id/show_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="#FFFF00"
        android:text="@string/count_initial_value"
        android:textAlignment="trung tâm"
        android:textColor="@color/colorPrimary"
        android:textSize="160sp"
        android:textStyle="đậm"
        android:layout_weight="1"/>

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

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:onClick="đếm"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white" />
</LinearLayout>
```

Nhiệm vụ 3: Thay đổi bố cục thành RelativeLayout

R [relativeLayout](#) là một nhóm chế độ xem trong đó mỗi chế độ xem được định vị và căn chỉnh tương đối với các chế độ xem khác trong nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục với R [relativeLayout](#).

3.1 Thay đổi LinearLayout thành RelativeLayout

Một cách dễ dàng để thay đổi L [inearLayout](#) thành R [elativeLayout](#) là thêm các thuộc tính XML vào tab **T ext**.

1. Mở tệp bố cục `activity_main.xml` và nhấp vào **tab T ext** ở cuối ngăn chỉnh sửa để xem mã XML.
2. Thay đổi `<LinearLayout` ở trên cùng thành `<RelativeLayout` để câu lệnh trông như sau:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Cuộn xuống để đảm bảo rằng thẻ kết thúc `</LinearLayout>` cũng đã thay đổi thành `</RelativeLayout>`; nếu chưa, hãy thay đổi nó theo cách thủ công.

3.2 Sắp xếp lại các chế độ xem trong RelativeLayout

Một cách dễ dàng để sắp xếp lại và định vị các chế độ xem trong RelativeLayout là thêm các thuộc tính XML vào tab Text.

- Nhấp vào tab **P review** ở bên cạnh trình chỉnh sửa (nếu nó chưa được chọn) để xem bản xem trước bô cục, bây giờ trông giống như hình bên dưới.



Với sự thay đổi thành RelativeLayout, trình soạn thảo bô cục cũng thay đổi một số thuộc tính chế độ xem.
Chẳng hạn:

- COUNT Button (button_count) phủ lên TOAST Button, đó là lý do tại sao bạn không thể nhìn thấy TOAST Button (button_toast).
- Phần trên cùng của TextView (show_count) phủ lên các phần tử Button.

2. Thêm thuộc tính `android:layout_below` vào `button_count` để định vị Button ngay bên dưới `show_count` TextView. Thuộc tính này là một trong một số thuộc tính để định vị các chế độ xem trong `RelativeLayout` bạn đặt các chế độ xem liên quan đến các chế độ xem khác.
`android:layout_below="@+id/show_count"`

3. Thêm thuộc tính `android:layout_centerHorizontal` vào cùng một nút B để căn giữa view theo chiều ngang trong cha mẹ của nó, trong trường hợp này là nhóm view `RelativeLayout`.

```
android:layout_centerHorizontal="đúng"
```

Mã XML đầy đủ cho `button_count` như sau:

```
< Nút
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:onClick="đếm"
    android:text="@string/button_label_count"
    android:textColor="@android:màu/trắng" android:layout_below="@+id/show_count"
    android:layout_centerHorizontal="true"/>
```

4. Thêm các thuộc tính sau vào `show_count` TextView:

```
android:layout_below="@+id/button_toast" android:layout_alignParentLeft="đúng"
    android:layout_alignParentStart="đúng"
```

Một `android:layout_alignParentLeft` căn chỉnh chế độ xem ở phía bên trái của

RelativeLayout nhóm chế độ xem mẹ. Mặc dù bản thân thuộc tính này là đủ để căn chỉnh chế độ xem

Ở phía bên trái, bạn có thể muốn chế độ xem căn chỉnh ở phía bên phải *nếu* ứng dụng đang chạy trên thiết bị đang sử dụng ngôn ngữ từ phải sang trái. Do đó, thuộc tính android:layout_alignParentStart làm cho cạnh "bắt đầu" của chế độ xem này khớp với cạnh bắt đầu của cha mẹ. *Start* là cạnh trái của màn hình nếu tùy chọn là từ trái sang phải hoặc nó là cạnh phải của màn hình nếu tùy chọn là từ phải sang trái.

- Xóa thuộc tính android:layout_weight="1" khỏi TextView, không liên quan đến RelativeLayout. Bản xem trước bố cục bây giờ trông giống như hình sau:



Mẹo: R elativeLayout giúp bạn tương đối dễ dàng nhanh chóng đặt các thành phần giao diện người dùng trong bố cục. Để tìm hiểu thêm về cách định vị chế độ xem trong R elativeLayout, hãy xem "[Positioning Views](#)" trong chủ đề "Bố cục tương đối" của Hướng dẫn API.

Mã giải pháp nhiệm vụ 3

Mã XML trong một ctivity_main.xml:

```
<RelativeLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="dọc"
    tools:context="com.example.android.hellotoast.MainActivity">

    < Nút
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@color/colorPrimary"
        android:onClick="hiển thị bánh mì nướng"
        android:text="@string/button_label_toast"
        android:textColor="@android:color/white" />

    <Ché độ xem văn bản
        android:id="@+id/show_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="#FFFF00"
```

```
    android:text="@string/count_initial_value"
    android:textAlignment="trung tâm"
    android:textColor="@color/colorPrimary"
    android:textSize="160sp"
    android:textStyle="đậm"
    android:layout_below="@+id/button_toast"
    android:layout_alignParentLeft="đúng"
    android:layout_alignParentStart="true" />

< Nút
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:onClick="đếm"
    android:text="@string/button_label_count"
    android:textColor="@android:màu/trắng" android:layout_below="@+id/show_count"
    android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

Tóm tắt

Sử dụng trình chỉnh sửa bố cục để xem trước và tạo mẫu mã:

- Để xem trước bố cục ứng dụng với hướng ngang trong trình chỉnh sửa bố cục, hãy nhập vào biểu tượng **Orientation in Editor** ở thanh công cụ trên cùng và chọn **Switch to Landscape**. Chọn **Switch to Portrait** để trở lại hướng dọc.
- Để tạo biến thể của bố cục khác nhau cho hướng ngang, hãy nhập vào nút **Orientation in Editor** và chọn **Create Landscape Variation**. Một cửa sổ trình chỉnh sửa mới mở ra với tab **land/activity_main.xml** hiển thị bố cục cho hướng ngang (ngang).

- Để xem trước bố cục cho các thiết bị khác nhau mà không cần phải chạy ứng dụng trên thiết bị hoặc trình mô phỏng, hãy nhấp vào **nút Device** trong Trình chỉnh sửa trên thanh công cụ trên cùng và chọn một thiết bị.
- Để tạo biến thể của bố cục khác cho máy tính bảng (màn hình lớn hơn), hãy nhấp vào biểu tượng **Orientation** trong nút **Editor** và chọn **Create layout x-large Variation**. Một biến tập viên mới cửa sổ mở ra với tab **X lớn/activity_main.xml** hiển thị bố cục cho thiết bị có kích thước bảng máy tính bảng.

Sử dụng ConstraintLayout:

- Để xóa tất cả các ràng buộc trong bố cục có gốc **ConstraintLayout**, hãy nhấp vào nút **Clear All Constraints** trên thanh công cụ.
- Bạn có thể căn chỉnh một phần tử giao diện người dùng có chứa văn bản, chẳng hạn như **Text View** hoặc **Button**, với một phần tử giao diện người dùng khác có chứa văn bản. Ràng buộc **baseline** cho phép bạn hạn chế các phần tử để các đường cơ sở văn bản khớp nhau.
- Để tạo ràng buộc đường cơ sở, hãy di con trỏ qua phần tử giao diện người dùng cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.
- Nút đóng gói trên thanh công cụ cung cấp các tùy chọn để đóng gói hoặc mở rộng các phần tử giao diện người dùng đã chọn. Bạn có thể sử dụng nó để sắp xếp đều các phần tử **Button** theo chiều ngang trên bố cục.

Sử dụng LinearLayout:

- LinearLayout** là một **ViewGroup** sắp xếp bộ sưu tập các chế độ xem của nó trong một hàng ngang hoặc dọc.
- LinearLayout** được yêu cầu phải có các thuộc tính **layout_width**, **layout_height** và **orientation**.
- match_parent** cho **layout_width** hoặc **layout_height**: Mở rộng View để lấp đầy cha của nó theo chiều rộng hoặc chiều cao. Khi **LinearLayout** là gốc View, nó sẽ mở rộng đến kích thước của màn hình (View mẹ).
- Wrap_content** cho **layout_width** hoặc **layout_height**: Thu nhỏ kích thước để View vừa đủ lớn để bao bọc nội dung của nó. Nếu không có nội dung, View trở nên vô hình.

- Số điểm ảnh dp cố định ([đơn vị ảnh không phụ thuộc vào điểm ảnh](#)) cho layout_width hoặc layout_height: Chỉ định kích thước cố định, được điều chỉnh cho mật độ màn hình của thiết bị. Ví dụ: 16dp có nghĩa là 16 pixel không phụ thuộc vào mật độ.
- Orientation cho LinearLayout có thể là horizontal để sắp xếp các phần tử từ trái sang phải, hoặc vertical để sắp xếp các phần tử từ trên xuống dưới.
- Chỉ định gravity và weight thuộc tính cho phép bạn kiểm soát thêm việc sắp xếp các chế độ xem và nội dung trong LinearLayout.
- Thuộc tính android:gravity chỉ định sự liên kết của nội dung của View trong Xem chính nó.
- Thuộc tính android:layout_weight trong chỉ ra bao nhiêu không gian bổ sung trong LinearLayout sẽ được phân bổ cho View. Nếu chỉ có một View có thuộc tính này, nó sẽ nhận được tất cả không gian màn hình bổ sung. Đối với nhiều phần tử View, không gian được chia theo tỷ lệ. Ví dụ, nếu hai Mỗi phần tử nút có trọng lượng là 1 và một TextView 2, tổng cộng là 4, các phần tử Button nhận được 1/4 khoảng trống mỗi phần tử và một nửa TextView.

Sử dụng RelativeLayout:

- RelativeLayout là một ViewGroup trong đó mỗi chế độ xem được định vị và căn chỉnh so với các chế độ xem khác trong nhóm.
- Sử dụng android:layout_alignParentTop để căn chỉnh View với trên cùng của cha mẹ.
- Sử dụng android:layout_alignParentLeft để căn chỉnh View ở phía bên trái của cha mẹ.
- Sử dụng android:layout_alignParentStart để làm cho cạnh bắt đầu của View khớp với cạnh bắt đầu của mẹ. Thuộc tính này hữu ích nếu bạn muốn ứng dụng của mình hoạt động trên các thiết bị sử dụng các tùy chọn ngôn ngữ hoặc ngôn ngữ khác nhau. Start là cạnh trái của màn hình nếu tùy chọn là từ trái sang phải hoặc nó là cạnh phải của màn hình nếu tùy chọn là từ phải sang trái.

Các khái niệm liên quan

Tài liệu khái niệm liên quan có trong [1.2: Bộ cục và tài nguyên cho giao diện người dùng](#).

Tìm hiểu thêm

Tài liệu Android Studio:

- [Làm quen với Android Studio](#)
- [Tạo biểu tượng ứng dụng bằng Image Asset Studio](#)

Tài liệu dành cho nhà phát triển Android:

- [Bố trí](#)
- [Xây dựng giao diện người dùng bằng Layout Editor](#)
- [Xây dựng giao diện người dùng đáp ứng với ConstraintLayout](#)
- [Bố trí](#)
- [Bố cục tuyến tính](#)
- [Bố cục tương đối](#)
- [Cạnh](#)
- [Nút • TextView](#)
- [Hỗ trợ mật độ điểm ảnh khác nhau](#) Khác:
 - Lớp học lập trình: [Using ConstraintLayout để thiết kế chế độ xem Android của bạn](#)
 - [Thuật ngữ từ vựng và khái niệm](#)

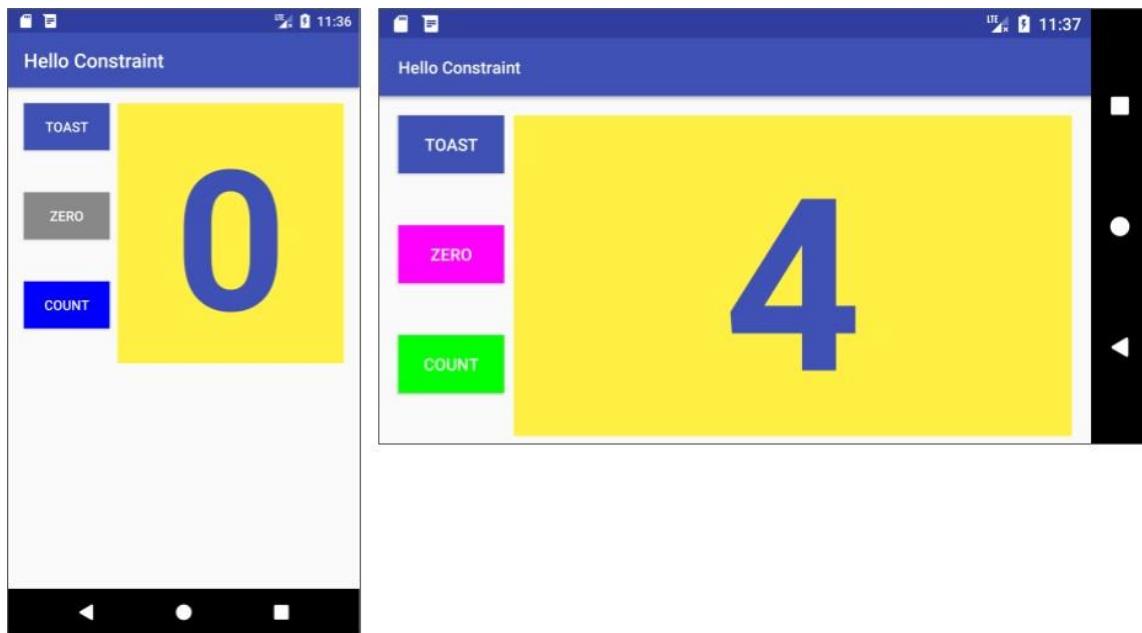
Homework

Thay đổi ứng dụng

Mở [ứng dụng](#) HelloToast.

1. Thay đổi tên của dự án thành **HelloConstraint** và tái cấu trúc dự án thành **Hello Constraint**. ([F](#)or hướng dẫn về cách sao chép và tái cấu trúc một dự án, xem A [ppendix: Tiêu chí](#).)
2. Sửa đổi bố cục activity_main.xml để căn chỉnh các phần tử **Toast** và **Count Button** dọc theo phía bên trái của `s how_count TextView` hiển thị "0". Tham khảo các hình dưới đây để biết bố cục.
3. Bao gồm một chữ **B** thứ ba được gọi là **Zero** xuất hiện giữa **các phần tử Toast** và **Count Button**.
4. Phân phối các phần tử **Button** theo chiều dọc giữa trên cùng và dưới cùng của `s how_count TextView`.
5. Đặt **Zero** **Button** ban đầu có nền màu xám.
6. Đảm bảo rằng bạn bao gồm **Zero** **Button** cho hướng ngang trong `activity_main.xml` (đất liền) và cả cho màn hình có kích thước bằng máy tính bảng trong `activity_main (xlarge)`.
7. Làm cho **Zero** **Button** thay đổi giá trị trong `s how_count TextView` thành 0.

8. Cập nhật trình xử lý nhấp chuột cho C **ount** B utton để nó thay đổi màu nền *o wn* của nó , tùy thuộc vào việc số đếm mới là lẻ hay chẵn.
Gợi ý: Không sử dụng `findViewById` để tìm C **ount** B utton. Bạn có thể sử dụng thứ gì khác không?
Vui lòng sử dụng hằng số trong lớp `C olor` cho hai màu nền khác nhau.
9. Cập nhật trình xử lý nhấp chuột cho C **ount** B utton để đặt màu nền cho Nút Z ero thành màu khác ngoài màu xám để biết nó hiện đang hoạt động. Gợi ý: Bạn có thể sử dụng `findViewById` trong trường hợp này.
10. Cập nhật trình xử lý nhấp chuột cho Z ero B utton để đặt lại màu thành màu xám, sao cho màu xám khi số đếm bằng không.



Trả lời những câu hỏi này

Câu hỏi 1

Hai thuộc tính ràng buộc bô cục nào trên Z ero B utton định vị nó theo chiều dọc bằng khoảng cách giữa hai phần tử B utton còn lại? (Chọn 2 câu trả lời.)

- Ứng dụng: `layout_constraintBottom_toTopOf="@+id/button_count"`
- `android:layout_marginBottom="8dp"`

- android:layout_marginStart="16dp"
- ứng dụng:layout_constraintTop_toBottomOf="@+id/button_toast"
- android:layout_marginTop="8dp"

Câu hỏi 2

Thuộc tính ràng buộc bối cảnh nào trên **Z ero** Button định vị nó theo chiều ngang thẳng hàng với hai phần tử Button còn lại?

- app:layout_constraintLeft_toLeftOf="cha mẹ"
- ứng dụng:layout_constraintBottom_toTopOf="@+id/button_count"
- android:layout_marginBottom="8dp"
- ứng dụng:layout_constraintTop_toBottomOf="@+id/button_toast"

Câu hỏi 3

Chữ ký chính xác cho một phương thức được sử dụng với thuộc tính XML android:onClick là gì?

- public void callMethod()
- public void callMethod(Xem chế độ xem)
- private void callMethod(Xem chế độ xem)
- public boolean callMethod(Chế độ xem)

Câu hỏi 4

Trình xử lý nhấp chuột cho C ount Button bắt đầu bằng chữ ký phương thức sau:

```
public void countUp (Xem chế độ xem)
```

Kỹ thuật nào sau đây hiệu quả hơn khi sử dụng trong trình xử lý này để thay đổi màu nền của phần tử Button? Chọn một:

- Sử dụng `findViewById` để tìm `COUNT_BUTTON`. Gán kết quả cho một biến `view`, sau đó sử dụng `setBackgroundColor()`.
- Sử dụng tham số `view` được chuyển đến trình xử lý nhập chuột với `setBackgroundColor()`; `view.setBackgroundColor()`

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kiểm tra xem ứng dụng có các tính năng sau không:

- Nó hiển thị nút **Zero**.
- Nút **Zero** nằm giữa các nút **Count** và **Reset**.
- Ứng dụng này bao gồm triển khai `activity_main.xml`, `activity_main.xml` (đất) và `activity_main.xml` (xlarge).
- Ứng dụng bao gồm việc triển khai phương pháp xử lý nhập chuột cho nút **Zero** để đặt lại số đếm về 0. Phương thức phải hiển thị số lượng không trong `show_count TextView`. Trình xử lý nhập chuột cũng phải đặt lại màu nền của nút **Zero** thành màu xám.
- Phương pháp xử lý nhập chuột cho nút **Count** đã được cập nhật để thay đổi màu nền của riêng nó tùy thuộc vào việc số đếm mới là lẻ hay chẵn. Phương thức này phải sử dụng tham số `view` để truy cập nút. Phương pháp này cũng phải thay đổi nền của nút **Zero** thành màu khác với màu xám.

Bài 1.3: Chế độ xem văn bản và cuộn

Giới thiệu

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

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

Bạn thường sẽ sử dụng chế độ xem cuộn cho các câu chuyện tin tức, bài báo hoặc bất kỳ văn bản dài nào không hoàn toàn phù hợp với màn hình. Bạn cũng có thể sử dụng chế độ xem cuộn để cho phép người dùng nhập nhiều dòng văn bản hoặc kết hợp các thành phần giao diện người dùng (chẳng hạn như trường văn bản và nút) trong chế độ xem cuộn.

Lớp [ScrollView](#) cung cấp bộ cục cho chế độ xem cuộn. ScrollView là một lớp con của [FrameLayout](#). Chỉ đặt *one* view như một chế độ xem con bên trong nó — một chế độ xem con chứa toàn bộ nội dung để cuộn. Bản thân chế độ xem con này có thể là một [ViewGroup](#) (chẳng hạn như [LinearLayout](#)) chứa các phần tử giao diện người dùng.

Bộ cục phức tạp có thể gặp vấn đề về hiệu suất với chế độ xem con như hình ảnh. Một lựa chọn tốt cho View trong ScrollView là [LinearLayout](#) được sắp xếp theo hướng dọc, trình bày các mục mà người dùng có thể cuộn qua (chẳng hạn như các phần tử [TextView](#)).

Với ScrollView, tất cả các thành phần giao diện người dùng đều nằm trong bộ nhớ và trong hệ thống phân cấp chế độ xem ngay cả khi chúng không được hiển thị trên màn hình. Điều này làm cho ScrollView trở nên lý tưởng để cuộn các trang văn bản dạng tự do một cách mượt mà, vì văn bản đã có trong bộ nhớ. Tuy nhiên, ScrollView có thể sử dụng nhiều bộ nhớ, điều này có thể ảnh hưởng đến hiệu suất của phần còn lại của ứng dụng. Để hiển thị danh sách dài các mục mà người dùng có thể thêm, xóa hoặc chỉnh sửa, hãy cân nhắc sử dụng [RecyclerView](#), được mô tả trong một bài học riêng.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo ứng dụng Hello World bằng Android Studio.
- Chạy ứng dụng trên trình mô phỏng hoặc thiết bị.
- Triển khai [TextView](#) trong bộ cục cho ứng dụng.
- Tạo và sử dụng tài nguyên chuỗi.

Những gì bạn sẽ học

- Cách sử dụng mã XML để thêm nhiều phần tử [TextView](#).
- Cách sử dụng mã XML để xác định một cuộn [View](#).
- Cách hiển thị văn bản dạng tự do với một số thẻ định dạng HTML.
- Cách tạo kiểu cho màu nền và màu văn bản của [TextView](#).
- Làm thế nào để bao gồm một liên kết web trong văn bản.

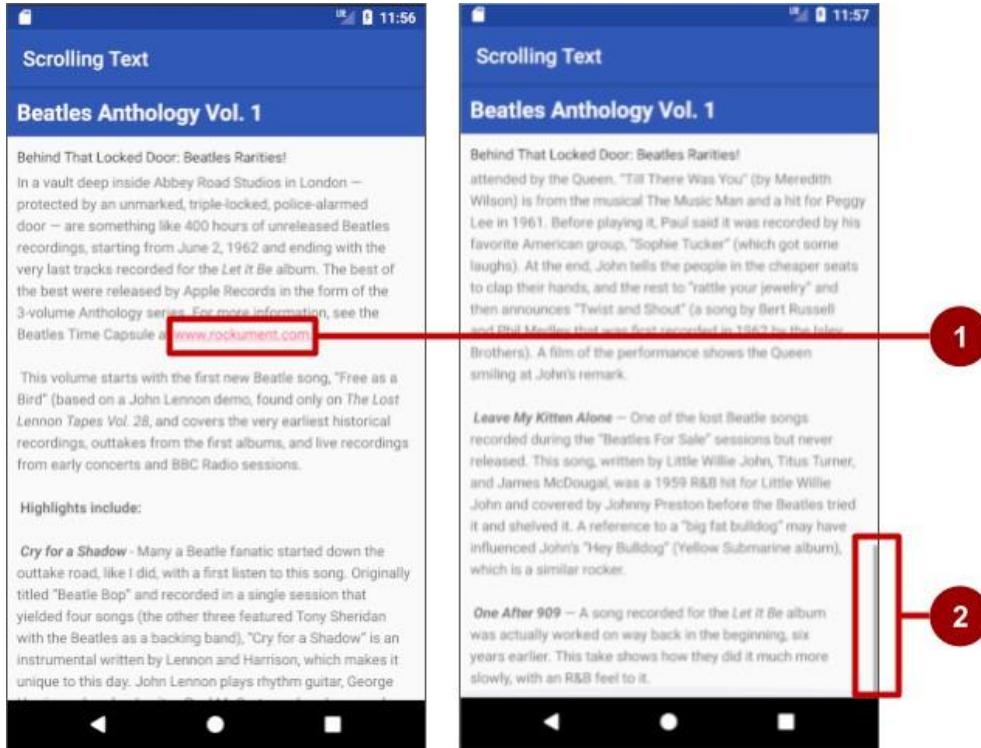
Bạn sẽ làm gì

- Tạo ứng dụng ScrollingText.
- Thay đổi [ConstraintLayout](#) [ViewGroup](#) thành [RelativeLayout](#).

- Thêm hai phần tử TextView cho tiêu đề và tiêu đề phụ của bài viết.
- Sử dụng kiểu và màu sắc TextViewAppearance cho tiêu đề và tiêu đề phụ của bài viết.
- Sử dụng thẻ HTML trong chuỗi văn bản để kiểm soát định dạng.
- Sử dụng thuộc tính LineSpacingExtra để thêm khoảng cách dòng cho dễ đọc.
- Thêm ScrollView vào bố cục để cho phép cuộn phần tử TextView.
- Thêm thuộc tính utoLink để cho phép các URL trong văn bản hoạt động và có thể nhấp vào.

Tổng quan về ứng dụng

Ứng dụng Văn bản cuộn minh họa thành phần giao diện người dùng ScrollView. ScrollView là một ViewGroup mà trong ví dụ này chứa một TextView. Nó hiển thị một trang văn bản dài — trong trường hợp này là đánh giá album nhạc — mà người dùng có thể cuộn theo chiều dọc để đọc bằng cách vuốt lên và xuống. Một thanh cuộn xuất hiện ở lề bên phải. Ứng dụng cho thấy cách bạn có thể sử dụng văn bản được định dạng với các thẻ HTML tối thiểu để đặt văn bản thành in đậm hoặc in nghiêng và với các ký tự dòng mới để phân tách đoạn văn. Bạn cũng có thể bao gồm các liên kết web đang hoạt động trong văn bản.

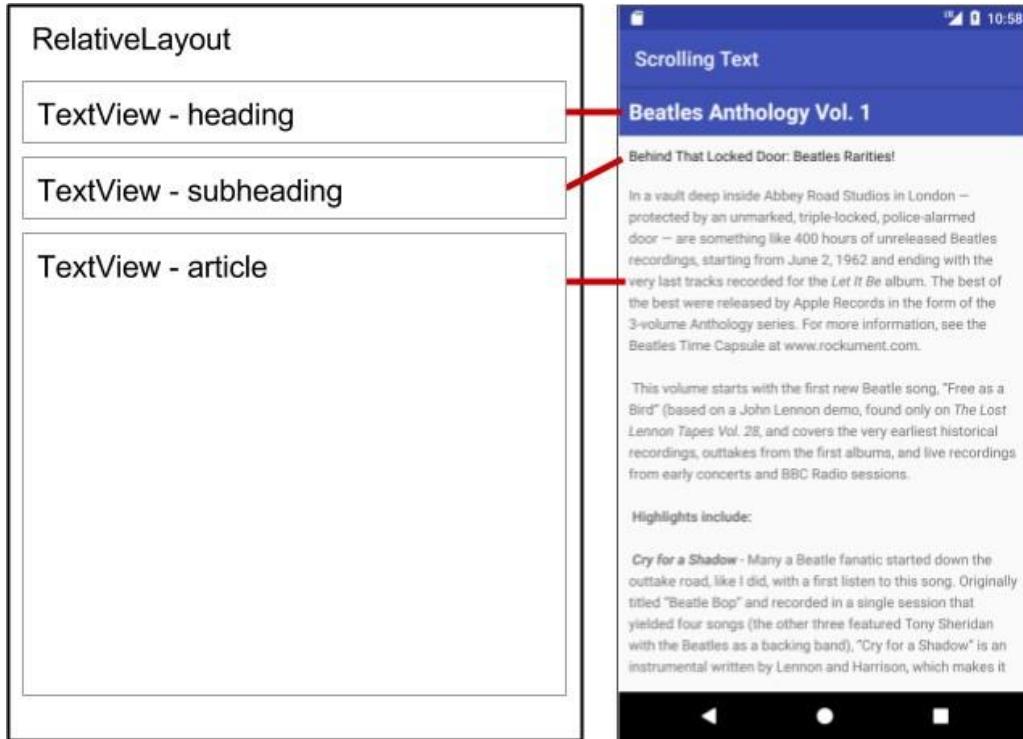


Trong hình trên, xuất hiện như sau:

1. Một liên kết web đang hoạt động được nhúng trong văn bản dạng tự do
2. Thanh cuộn xuất hiện khi cuộn văn bản

Nhiệm vụ 1: Thêm và chỉnh sửa các phần tử TextView

Trong thực tế này, bạn sẽ tạo một dự án Android cho ứng dụng ScrollingText, thêm các phần tử TextView vào bộ cục cho tiêu đề và phụ đề bài viết, đồng thời thay đổi phần tử TextView "Hello World" hiện có để hiển thị một bài viết dài. Hình dưới đây là sơ đồ bộ cục.



Bạn sẽ thực hiện tất cả các thay đổi này trong mã XML và trong tệp strings.xml. Bạn sẽ chỉnh sửa mã XML cho bối cảnh trong ngăn Văn bản, mà bạn hiển thị bằng cách nhấp vào tab Text, thay vì nhấp vào tab Thiết kế cho ngăn Thiết kế. Một số thay đổi đối với các phần tử và thuộc tính giao diện người dùng dễ thực hiện trực tiếp trong ngăn Văn bản bằng mã nguồn XML.

1.1 Tạo các phần tử dự án và TextView

Trong nhiệm vụ này, bạn sẽ tạo dự án và các phần tử TextView, đồng thời sử dụng các thuộc tính TextView để tạo kiểu cho văn bản và nền.

Mẹo: Để tìm hiểu thêm về các thuộc tính này, hãy xem tài liệu tham khảo TextView.

- Trong Android Studio, hãy tạo một dự án mới với các thông số sau:

Thuộc tính	Giá trị
Tên ứng dụng	Cuộn văn bản
Tên công ty	android.example.com (hoặc miền của riêng bạn)
SDK tối thiểu trên điện thoại và máy tính bảng	API15: Android 4.0.3 IceCreamSandwich
Mẫu	Hoạt động trống rỗng
Hộp kiểm Tạo tệp bô cục	Chọn
Hộp kiểm Tương thích ngược (AppCompat)	Chọn

2. Trong **thư mục app > res > layout trong ngăn Project > Android**, mở tệp **activity_main.xml** và nhấp vào tab **Text** để xem mã XML.

Ở trên cùng, hoặc *root*, của hệ thống phân cấp View là ConstraintLayout ViewGroup:

```
android.support.constraint.ConstraintLayout
```

3. Thay đổi ViewGroup này thành RelativeLayout. Dòng mã thứ hai bây giờ trông giống như thế này:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

RelativeLayout cho phép bạn đặt các phần tử giao diện người dùng tương đối với nhau hoặc tương đối với cha

RelativeLayout .

Phần tử "Hello World" TextView mặc định được tạo bởi mẫu Empty Layout vẫn có các thuộc tính ràng buộc (chẳng hạn như `pp:layout_constraintBottom_toBottomOf="parent"`). Để lo lắng—you sẽ xóa chúng trong bước tiếp theo.

- Xóa dòng mã XML sau đây, có liên quan đến ConstraintLayout:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

Khôi mã XML ở trên cùng bây giờ trông như thế này:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.scrollingtext.MainActivity">
```

- Thêm phần tử TextView phía trên TextView "Hello World" bằng cách nhập < TextView. Một khôi TextView xuất hiện kết thúc bằng /> và hiển thị các thuộc tính layout_width và layout_height, cần thiết cho TextView.
- Nhập các thuộc tính sau cho TextView. Khi bạn nhập từng thuộc tính và giá trị, các đề xuất sẽ xuất hiện để hoàn thành tên hoặc giá trị thuộc tính.

Thuộc tính TextView #1	Giá trị
Android:layout_width	"match_parent"

Android:layout_height	"wrap_content"
android:id	"@+id/article_heading"
Android:Nền	"@color/colorPrimary"
android:textColor	"@android: màu / trắng"
android:đệm	"10dp"
android:textAppearance	"@android:style/TextAppearance.DeviceDefault.Large"
android:textStyle	"đậm"
android:văn bản	"Tiêu đề bài viết"

7. Trích xuất tài nguyên chuỗi cho chuỗi được mã hóa cứng của thuộc tính android:text "Tiêu đề bài viết" trong TextView để tạo một mục nhập cho nó trong **strings.xml**.

Đặt con trỏ vào chuỗi được mã hóa cứng, nhấn Alt-Enter (Option-Enter trên máy Mac) và chọn **Extract string resource**. Đảm bảo rằng tùy chọn **Create the resource in directories** được chọn, sau đó chỉnh sửa tên tài nguyên cho giá trị chuỗi thành **một rticle_title**.

Tài nguyên chuỗi được mô tả chi tiết trong Tài [nguyên String](#).

8. Trích xuất tài nguyên thứ nguyên cho chuỗi mã hóa cứng của thuộc tính android:padding "10dp" trong TextView để tạo file **dimens.xml** và thêm một mục nhập vào đó.

Đặt con trỏ vào chuỗi được mã hóa cứng, nhấn Alt-Enter (Option-Enter trên máy Mac) và chọn **Extract tài nguyên thứ nguyên**. Đảm bảo rằng tùy chọn **Create the resource in directories** được chọn, sau đó chỉnh sửa Resource name thành **p adding_regular**.

9. Thêm một phần tử TextView khác phía trên TextView "Hello World" và bên dưới TextView bạn đã tạo ở các bước trước. Thêm các thuộc tính sau vào TextView:

Thuộc tính TextView #2	Giá trị
layout_width	"match_parent"
layout_height	"wrap_content"
android:id	"@+id/article_subheading"
Android:layout_below	"@+id/article_heading"
android:đệm	"@dimen/padding_regular"
android:textAppearance	"@android:style/TextAppearance.DeviceDefault"
android:văn bản	"Phụ đề bài viết"

Vì bạn đã trích xuất tài nguyên thứ nguyên cho chuỗi "10dp" thành padding_regular trong TextView đã tạo trước đó, bạn có thể sử dụng "@dimen/padding_regular" cho thuộc tính android:padding trong TextView này.

10. Trích xuất tài nguyên chuỗi cho chuỗi được mã hóa cứng của thuộc tính android:text " Article Subtitle" trong TextView thành **một rticle_subtitle**.

11. Trong phần tử "Hello World" TextView, xóa các thuộc tính layout_constraint:

```
app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent" app:layout_constraintTop_toTopOf="parent"
```

12. Thêm các thuộc tính TextView sau vào phần tử TextView "Hello World" và thay đổi thuộc tính android:text:

Thuộc tính TextView	Giá trị

android:id	"@+id/bài viết"
Android:layout_below	"@id/article_subheading"
android:lineSpacingExtra	"5sp"
android:đệm	"@dimen/padding_regular"
android:văn bản	Thay đổi thành "Văn bản bài viết"

13. Trích xuất tài nguyên chuỗi cho " Văn bản bài viết" đến **một rticle_text**và trích xuất tài nguyên thứ nguyên cho " 5sp" đến **Lin e_spacing**.
14. Định dạng lại và căn chỉnh mã bằng cách chọn **C ode > Định dạng lại mã**. Bạn nên định dạng lại và căn chỉnh mã của mình để bạn và những người khác dễ hiểu hơn.

1.2 Thêm văn bản của bài viết

Trong một ứng dụng thực sự truy cập các bài báo trên tạp chí hoặc báo, các bài báo xuất hiện có thể đến từ một nguồn trực tuyến thông qua nhà cung cấp nội dung hoặc có thể được lưu trữ trong cơ sở dữ liệu trên thiết bị.

Đối với thực tế này, bạn sẽ tạo bài viết dưới dạng một chuỗi dài duy nhất trong tài nguyên strings.xml.

1. Trong **thư mục app > res > values**, hãy mở **s trings.xml**.
2. Mở bất kỳ tệp văn bản nào có một lượng lớn văn bản hoặc mở [tệp s trings.xml của ứng dụng ScrollingText đã hoàn thành](#).
3. Nhập các giá trị cho các chuỗi một rticle_title và một rticle_subtitle với tiêu đề và phụ đề được tạo ra hoặc sử dụng các giá trị trong tệp strings.xml của đã hoàn thành
Ứng dụng ScrollingText. Đặt giá trị chuỗi thành văn bản một dòng không có thẻ HTML hoặc nhiều dòng.
4. Nhập hoặc sao chép và dán văn bản cho chuỗi rticle_text.

Bạn có thể sử dụng văn bản trong tệp văn bản của mình hoặc sử dụng văn bản được cung cấp cho chuỗi `article_text` trong tệp `strings.xml` của ứng dụng `ScrollingText` đã hoàn thành. Yêu cầu duy nhất cho tác vụ này là văn bản phải đủ dài để không vừa với màn hình.

Hãy ghi nhớ những điều sau (tham khảo hình bên dưới để biết người yêu cùample):

- Khi bạn nhập hoặc dán văn bản vào tệp `strings.xml`, các dòng văn bản không ngắt quãng đến dòng tiếp theo—chúng kéo dài ra ngoài lề bên phải. Đây là hành vi chính xác—mỗi dòng văn bản mới bắt đầu từ lề trái đại diện cho toàn bộ đoạn văn. Nếu bạn muốn văn bản trong `strings.xml` được bao bọc, bạn có thể nhấn `Return` để nhập kết thúc dòng cứng hoặc định dạng văn bản trước tiên trong trình soạn thảo văn bản có kết thúc dòng cứng.
- Nhập `\n` để biểu thi phần cuối của một dòng và một `\n` khác để biểu thi một dòng trống. Bạn cần thêm các ký tự cuối dòng để giữ cho các đoạn văn không chạy vào nhau.
- Nếu bạn có dấu nháy đơn (' ') trong văn bản của mình, bạn phải thoát nó bằng cách trước nó bằng dấu gạch chéo ngược (\'). Nếu bạn có dấu ngoặc kép trong văn bản của mình, bạn cũng phải thoát nó (\"). Bạn cũng phải thoát khỏi bất kỳ ký tự không phải ASCII nào khác. Xem [phản tạo kiểu và tạo kiểu](#) F của [tài nguyên S tring](#) để biết thêm chi tiết.
- Nhập thẻ HTML `` và `` xung quanh các từ nên được in đậm.
- Nhập thẻ HTML `<i>` và `</i>` xung quanh các từ phải in nghiêng. Nếu bạn sử dụng dấu nháy đơn cong trong cụm từ nghiêng, hãy thay thế chúng bằng dấu nháy đơn thẳng.
- Bạn có thể kết hợp in đậm và in nghiêng bằng cách kết hợp các thẻ, như trong `<i>... words...</i>`. Các thẻ HTML khác bị bỏ qua.
- Bao gồm toàn bộ văn bản trong `<string name="article_text"></string>` trong tệp `strings.xml`.
- Bao gồm một liên kết web để kiểm tra, chẳng hạn như `w ww.google.com`. (Ví dụ dưới đây sử dụng `www.rockument.com`.) *D* không sử dụng thẻ HTML, vì bất kỳ thẻ HTML nào ngoại trừ thẻ in đậm và nghiêng đều bị bỏ qua và được trình bày dưới dạng văn bản, đây không phải là điều bạn muốn.

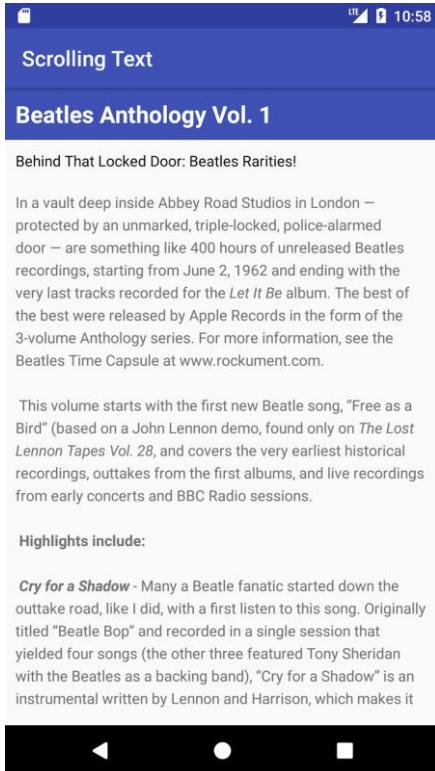
The screenshot shows the Android Studio interface with the strings.xml file open. The code editor displays the following XML:

```
<resources>
    <string name="app_name">Scrolling Text</string>
    <string name="article_title">Beatles Anthology Vol. 1</string>
    <string name="article_subtitle">Behind That Locked Door: Beatles Rarities!</string>
    <string name="article_text">In a vault deep inside Abbey Road Studios in London – protected by an unmarked, triple-locked door – lies a time capsule containing a collection of rare Beatles recordings. This volume starts with the first new Beatle song, "Free as a Bird" (based on a John Lennon demo, found only on The Lost Songs). Highlights include:</b>
        <b><i>Cry for a Shadow</i></b> – Many a Beatle fanatic started down the outtake road, like I did, with a first listen to this one.
        <b><i>My Bonnie</i></b> and <b><i>Ain't She Sweet</i></b> – At the same session, the Beatles played on "My Bonnie" (the first song John ever wrote).
        <b><i>Searchin</i></b> – A Jerry Leiber – Mike Stoller comedy song that was a hit for the Coasters in 1957, and a popular Beatles cover.
        <b><i>Love Me Do</i></b> – An early version of the song, played a bit slower and with more of a blues feeling, and a cool title.
        <b><i>She Loves You – Till There Was You – Twist and Shout</i></b> – Live at the Princess Wales Theatre by Leicester Square.
        <b><i>Leave My Kitten Alone</i></b> – One of the lost Beatle songs recorded during the "Beatles For Sale" sessions but never released.
        <b><i>One After 909</i></b> – A song recorded for the <i>Let It Be</i> album was actually worked on way back in the beginning.
    </string>
</resources>
```

A yellow warning icon is positioned next to the string value for the key "article_text". The tooltip message reads: "For more information, see the Beatles Time Capsule at www.rockument.com". The status bar at the top right of the editor says "Open editor Hide notification".

1.3 Chạy ứng dụng

Chạy ứng dụng. Bài viết xuất hiện, nhưng người dùng không thể cuộn bài viết vì bạn chưa bao gồm ScrollView (bạn sẽ thực hiện trong tác vụ tiếp theo). Cũng lưu ý rằng nhấn vào liên kết web hiện không làm được gì. Bạn cũng sẽ khắc phục điều đó trong nhiệm vụ tiếp theo.



Mã giải pháp nhiệm vụ 1

Một tệp bố cục `activity_main.xml` trông như sau:

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

```
    android:layout_height="match_parent"
    tools:context="com.example.android.scrollingtext.MainActivity">

    <Ché độ xem văn bản
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/article_heading"
        android:background="@color/colorPrimary"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_title"
        android:textAppearance=
            "@android:style/TextAppearance.DeviceDefault.Large" android:textColor="@android:color/white"
        android:textStyle="bold" />

    <Ché độ xem văn bản
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/article_subheading"
        android:layout_below="@+id/article_heading"
        android:padding="@dimen/padding_regular" android:text="@string/article_subtitle"
        android:textAppearance=
            "@android:style/TextAppearance.DeviceDefault" />

    <Ché độ xem văn bản
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:id="@+id/bài viết"
        android:layout_below="@+id/article_subheading"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_text" />
</RelativeLayout>
```

Nhiệm vụ 2: Thêm ScrollView và liên kết web đang hoạt động

Trong nhiệm vụ trước, bạn đã tạo ứng dụng ScrollingText với các phần tử TextView cho tiêu đề bài viết, phụ đề và văn bản bài viết dài. Bạn cũng đã bao gồm một liên kết web, nhưng liên kết đó chưa hoạt động. Bạn sẽ thêm mã để làm cho nó hoạt động.

Ngoài ra, bên thân TextView không thể cho phép người dùng cuộn văn bản bài viết để xem tất cả. Bạn sẽ thêm một ViewGroup mới có tên là ScrollView vào bộ cục XML sẽ làm cho TextView có thể cuộn được.

2.1 Thêm thuộc tính autoLink cho các liên kết web đang hoạt động

Thêm thuộc tính android:autoLink="web" vào phần tử TextView. Mã XML cho TextView này bây giờ trông như sau:

```
<Chế độ xem văn bản
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/bài viết"
    android:autoLink="web"
    android:layout_below="@+id/article_subheading"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />
```

2.2 Thêm ScrollView vào bộ cục

Để làm cho View (chẳng hạn như TextView) có thể cuộn được, hãy nhúng View *nhìn bên cạnh* ScrollView.

1. Thêm một ScrollView giữa article_subheading TextView và article TextView. Khi bạn vào <ScrollView>, Android Studio sẽ tự động thêm </ScrollView> ở cuối và hiển thị các thuộc tính android:layout_width và android:layout_height kèm theo các gợi ý.
2. Chọn wrap_content từ các gợi ý cho cả hai thuộc tính.

Mã cho hai phần tử TextView và ScrollView bây giờ trông như thế này:

```
<Ché độ xem văn bản
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/article_subheading"
    android:layout_below="@+id/article_heading"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_subtitle"
    android:textAppearance=
        "@android:style/TextAppearance.DeviceDefault"/>

<Ché độ xem cuộn
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"></ScrollView>
<Ché độ xem văn bản
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/bài viết"
    android:autoLink="web"
    android:layout_below="@+id/article_subheading"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />
```

3. Di chuyển phần kết thúc </ScrollView> mã sau một article TextView để các thuộc tính TextView của article hoàn toàn nằm bên trong ScrollView.
4. Xóa thuộc tính sau khỏi article TextView và thêm nó vào ScrollView:

```
    android:layout_below="@+id/article_subheading"
```

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0.

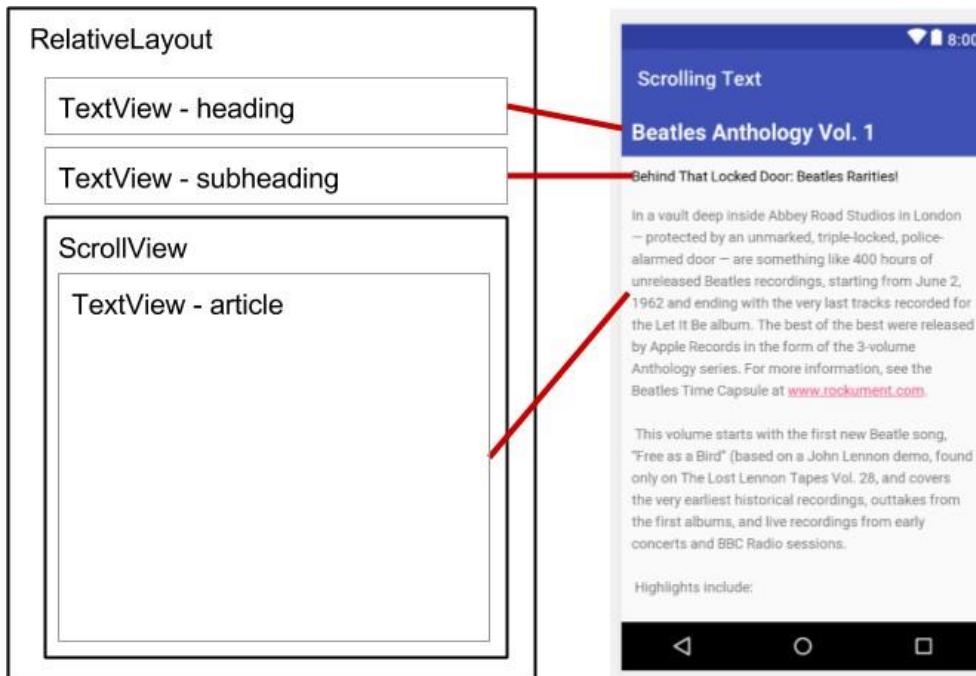
PDF này là *Ảnh chụp nhanh* một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2

Trang 127

Với thuộc tính trên, phần tử ScrollView sẽ xuất hiện bên dưới tiêu đề phụ của bài viết. Bài viết nằm bên trong phần tử ScrollView.

5. Chọn **C ode > Reformat Code** để định dạng lại mã XML để rticle TextView hiện xuất hiện thụt lề bên trong mã < ScrollView.
6. Nhập vào tab **Đánh giá P** ở phía bên phải của trình chỉnh sửa bố cục để xem trước bố cục.

Bố cục bây giờ trông giống như phía bên phải của hình sau:



2.3 Chạy ứng dụng

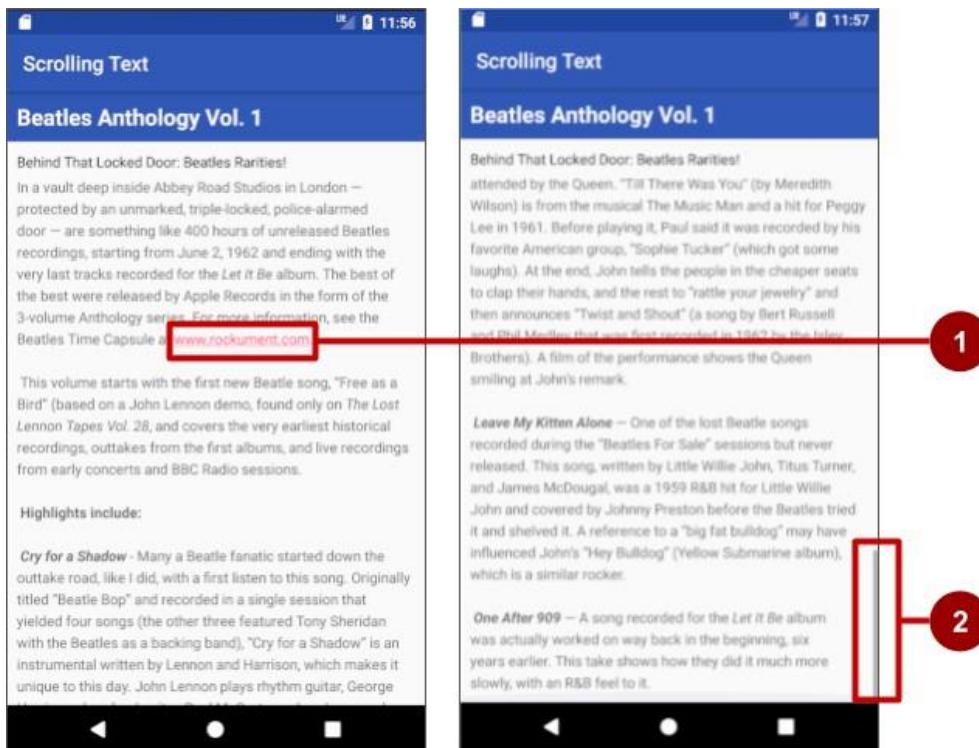
Để kiểm tra cách văn bản cuộn:

- Chạy ứng dụng trên thiết bị hoặc trình mô phỏng.

Vuốt lên và xuống để cuộn bài viết. Thanh cuộn xuất hiện ở lề phải khi bạn cuộn.

Nhấn vào web liên kết để chuyển đến web trang. Thuộc tính android:autoLink biến bất kỳ URL dễ nhận biết nào trong TextView (chẳng hạn như www.rockument.com) thành một liên kết web.

- Xoay thiết bị hoặc trình mô phỏng của bạn trong khi chạy ứng dụng. Lưu ý cách chế độ xem cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.
- Chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng. Lưu ý cách chế độ xem cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.



Trong hình trên, xuất hiện như sau:

- Một liên kết web đang hoạt động được nhúng trong văn bản dạng tự do
- Thanh cuộn xuất hiện khi cuộn văn bản

Mã giải pháp nhiệm vụ 2

Mã XML cho bộ cục với chế độ xem cuộn như sau:

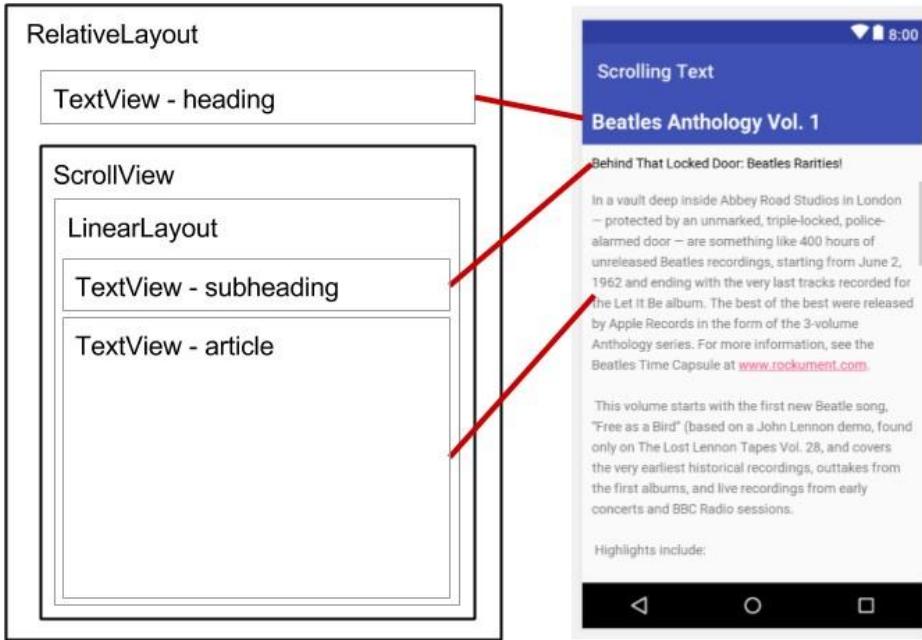
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context="com.example.android.scrollingtext.MainActivity">  
  
<Ché độ xem văn bản  
    android:id="@+id/article_heading"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="@color/colorPrimary"  
    android:padding="@dimen/padding_regular"  
    android:text="@string/article_title"  
    android:textAppearance=  
        "@android:style/TextAppearance.DeviceDefault.Large"  
    android:textColor="@android:màu/trắng"  
    android:textStyle="bold" />  
  
<Ché độ xem văn bản  
    android:id="@+id/article_subheading"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_below="@id/article_heading"  
    android:padding="@dimen/padding_regular" android:text="@string/article_subtitle"  
    android:textAppearance=  
        "@android:style/TextAppearance.DeviceDefault" />  
  
<Ché độ xem cuộn  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@id/article_subheading">  
<Ché độ xem văn bản  
    android:id="@+id/bài viết"  
    android:layout_width="wrap_content" android:layout_height="wrap_content"  
    android:autoLink="web"
```

```
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />
  </ScrollView>
</RelativeLayout>
```

Nhiệm vụ 3: Cuộn nhiều phần tử

Như đã lưu ý trước đây, một ScrollView chỉ có thể chứa một View con (chẳng hạn như một Article TextView mà bạn đã tạo). Tuy nhiên, View đó có thể là một ViewGroup khác chứa các phần tử View, chẳng hạn như [LinearLayout](#). Bạn có thể tìm một ViewGroup như LinearLayout với ScrollView, do đó cuộn mọi thứ bên trong LinearLayout.

Ví dụ: nếu bạn muốn tiêu đề phụ của bài viết cuộn cùng với bài viết, hãy thêm LinearLayout trong ScrollView và di chuyển tiêu đề phụ và bài viết vào LinearLayout. LinearLayout trở thành View con duy nhất trong ScrollView như trong hình bên dưới và người dùng có thể cuộn toàn bộ LinearLayout: tiêu đề phụ và bài viết.



3.1 Thêm LinearLayout vào ScrollView

1. Mở tệp **activity_main.xml** của dự án ứng dụng ScrollingText và chọn tab **Text** để chỉnh sửa mã XML (nếu chưa được chọn).
2. Thêm một **LinearLayout** phía trên một **Article TextView** trong **ScrollView**. Khi bạn nhập **<LinearLayout**, Android Studio sẽ tự động thêm **</LinearLayout>** vào cuối và hiển thị các thuộc tính **android:layout_width** và **android:layout_height** kèm theo các đề xuất. Chọn **match_parent** và **wrap_content** từ các đề xuất cho chiều rộng và chiều cao của nó, tương ứng. Mã ở đầu **ScrollView** bây giờ trông như thế này:

```
<Chế độ xem cuộn
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/article_subheading">

    <Bố cục tuyến tính
        android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content">></LinearLayout>
<Chế độ xem văn bản
    android:id="@+id/bài viết"
```

Bạn sử dụng `match_parent` để khớp chiều rộng của `ViewGroup` mẹ. Bạn sử dụng `wrap_content` để thay đổi kích thước `LinearLayout` để nó vừa đủ lớn để bao bọc nội dung của nó.

3. Di chuyển phần kết thúc `</LinearLayout>` mã sau `một Article TextView` nhưng *b trước khi* kết thúc `</ScrollView>`.

`LinearLayout` hiện bao gồm một `Article TextView` và hoàn toàn nằm bên trong `ScrollView`.

4. Thêm thuộc tính `android:orientation="vertical"` vào `LinearLayout` để đặt hướng của nó thành dọc.
5. Chọn **Code > Định dạng lại Mã** để thực hiện mã một cách chính xác.

`LinearLayout` bây giờ trông như thế này:

```
<Chế độ xem cuộn
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/article_subheading">
    <Bố cục tuyến tính
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
            <Chế độ xem văn bản
                android:id="@+id/bài viết"
                android:layout_width="wrap_content" android:layout_height="wrap_content"
                android:autoLink="web"
                android:lineSpacingExtra="@dimen/line_spacing"
                android:padding="@dimen/padding_regular"
```

Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị

```
    android:text="@string/article_text" />
  </LinearLayout>

</ScrollView>
```

3.2 Di chuyển các phần tử giao diện người dùng trong LinearLayout

LinearLayout hiện chỉ có một phần tử giao diện người dùng — một Article TextView. Bạn muốn bao gồm ArticleSubheading TextView trong LinearLayout để cả hai sẽ cuộn.

- Để di chuyển ArticleSubheading TextView, chọn mã, chọn **Edit > Cut**, nhấp vào phía trên một Article TextView bên trong LinearLayout và chọn **Edit > Paste**.
- Xóa thuộc tính android:layout_below="@+id/article_heading" khỏi ArticleSubheading TextView. Bởi vì TextView này hiện nằm trong LinearLayout, thuộc tính này sẽ xung đột với các thuộc tính LinearLayout.
- Thay đổi thuộc tính bố cục ScrollView từ android:layout_below="@+id/article_subheading" thành android:layout_below="@+id/article_heading". Bây giờ tiêu đề phụ là một phần của LinearLayout, thì ScrollView phải được đặt bên dưới tiêu đề, không phải tiêu đề phụ. Mã XML cho ScrollView bây giờ là sau:

```
<Ché độ xem cuộn
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_below="@+id/article_heading">
  <Bố cục truyền tính
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

  <Ché độ xem văn bản
    android:id="@+id/article_subheading"
    android:layout_width="match_parent" android:layout_height="wrap_content"
```

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

Trang 134

```
    android:padding="@dimen/padding_regular" android:text="@string/article_subtitle"
    android:textAppearance=
        "@android:style/TextAppearance.DeviceDefault" />

    <Ché độ xem văn bản
        android:id="@+id/bài viết"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:autoLink="web"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:padding="@dimen/padding_regular" android:text="@string/article_text" />
    </LinearLayout>

</ScrollView>
```

4. Chạy ứng dụng.

Vuốt lên và xuống để cuộn bài viết và lưu ý rằng tiêu đề phụ bảy giờ cuộn cùng với bài viết trong khi tiêu đề vẫn giữ nguyên.

Mã giải pháp

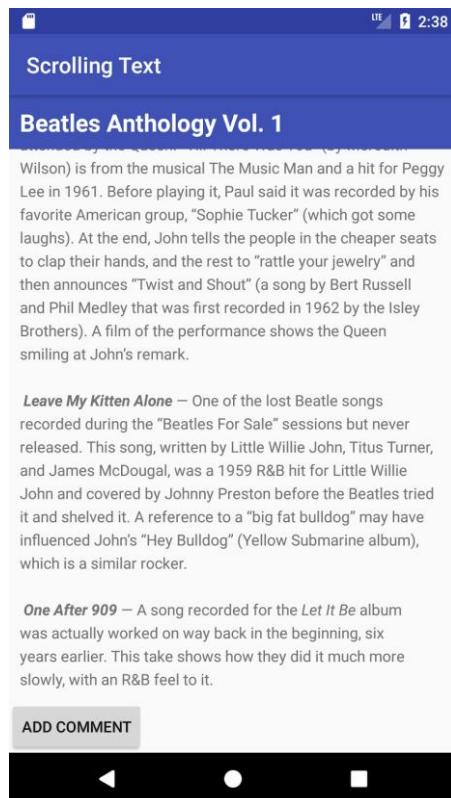
Dự án Android Studio: [ScrollingText](#)

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 này.

Thử thách: Add một phần tử giao diện người dùng khác — A Button — vào LinearLayout bên trong ScrollView để nó cuộn theo văn bản.

- Làm cho Button xuất hiện bên dưới bài viết. Người dùng cuộn đến cuối bài viết để xem Button.
- Sử dụng văn bản **A dd Comment** cho Button. Đối với thử thách này, không cần phải tạo ra phương pháp xử lý nút; tất cả những gì bạn phải làm là đặt phần tử Button vào vị trí thích hợp trong bố cục.



Mã giải pháp thách thức

Dự án Android Studio: [ScrollingTextChallenge](#)

Tóm tắt

- Sử dụng [ScrollView](#) để cuộn một View con duy nhất (chẳng hạn như TextView). Một ScrollView chỉ có thể chứa một con View hoặc ViewGroup.
- Sử dụng ViewGroup chẳng hạn như [LinearLayout](#) làm View con trong ScrollView để cuộn nhiều phần tử View. Bao gồm các phần tử trong LinearLayout.
- Hiển thị văn bản dạng tự do trong TextView với các thẻ định dạng HTML cho chữ in đậm và in nghiêng.
- Sử dụng \n làm ký tự cuối dòng trong văn bản dạng tự do để giữ cho đoạn văn không chạy vào đoạn tiếp theo.
- Sử dụng thuộc tính android:autoLink="web" để làm cho các liên kết web trong văn bản có thể nhấp được.

Các khái niệm liên quan

Tài liệu khái niệm liên quan nằm trong [1.3 Văn bản và chế độ xem cuộn](#).

Tìm hiểu thêm

Tài liệu Android Studio:

- [Trang tải xuống Android Studio](#)
- [Làm quen với Android Studio](#)

Tài liệu dành cho nhà phát triển Android:

- [Chế độ xem cuộn\(ScrollView\)](#)
- [Bộ cung cấp tính năng](#)

- [Bố cục tương đối](#)
- [Cánh](#)
- [Nút](#) ● [TextView](#)
- [Tài nguyên chuỗi](#)
- [Bố cục tương đối](#)

Khác:

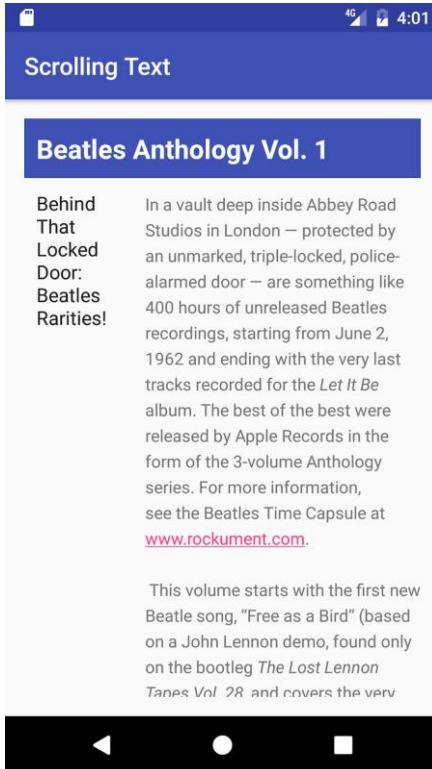
- Blog dành cho các nhà phát triển Android: [Link hóa văn bản của bạn!](#)
- Pathway mã: [Working với TextView](#)

Homework

Thay đổi ứng dụng

Mở [ứng dụng ScrollView2](#) mà bạn đã tạo trong [bài học](#) Working với TextView Elements.

1. Thay đổi tiêu đề phụ để nó bao bọc trong một cột ở bên trái rộng 100 dp, như hình dưới đây.
2. Đặt văn bản của bài viết ở bên phải của tiêu đề phụ như hình dưới đây.



Trả lời những câu hỏi này

Câu hỏi 1

Bạn có thể sử dụng bao nhiêu chế độ xem trong ScrollView? Chọn một:

- Chỉ một chế độ xem
- Một chế độ xem hoặc một nhóm chế độ xem
- Bao nhiêu tùy thích

Câu hỏi 2

Bạn sử dụng thuộc tính XML nào trong LinearLayout để hiển thị các chế độ xem cạnh nhau? Chọn một:

- android:orientation="ngang"
- android:orientation="dọc"
- android:layout_width="wrap_content"

Câu hỏi 3

Bạn sử dụng thuộc tính XML nào để xác định chiều rộng của LinearLayout bên trong chế độ xem cuộn? Chọn một:

- android:layout_width="wrap_content"
- android:layout_width="match_parent"
- android:layout_width="200dp"

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kiểm tra xem ứng dụng có các tính năng sau không:

- Bộ cục hiển thị tiêu đề phụ ở cột bên trái và văn bản bài viết ở cột bên phải, như trong hình trên.
- ScrollView bao gồm một LinearLayout với hai phần tử TextView.
- Hướng LinearLayout được đặt thành ngang.

Bài 1.4: Học cách giúp đỡ bản thân

Giới thiệu

Những điều bạn nên biết

Bạn sẽ có thể:

- Tìm hiểu quy trình làm việc cơ bản của Android Studio.
- Tạo ứng dụng từ đầu bằng cách sử dụng mẫu Hoạt động trống.

- Sử dụng trình chỉnh sửa bố cục.

Những gì bạn sẽ học

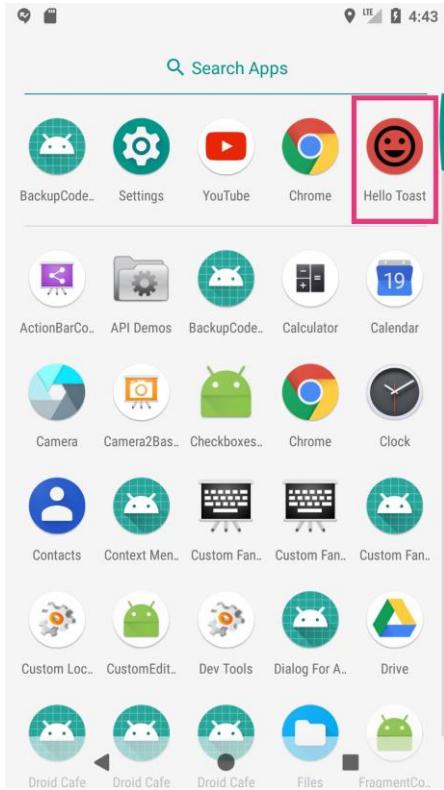
- Tìm thông tin và tài nguyên dành cho nhà phát triển ở đâu.
- Cách thêm biểu tượng trình khởi chạy vào ứng dụng của bạn.
- Cách tìm kiếm trợ giúp khi bạn đang phát triển ứng dụng Android.

Bạn sẽ làm gì

- Khám phá một số tài nguyên có sẵn cho các nhà phát triển Android ở mọi cấp độ.
- Thêm biểu tượng trình chạy cho ứng dụng của bạn.

Tổng quan về ứng dụng

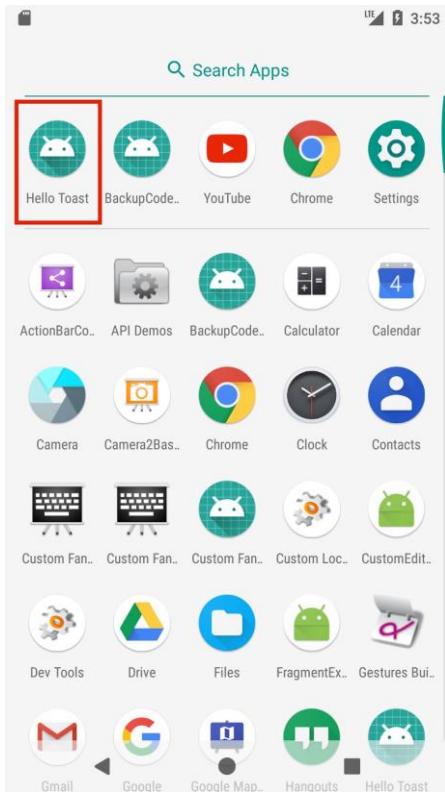
Bạn sẽ thêm biểu tượng launcher vào ứng dụng HelloToast mà bạn đã tạo trước đó hoặc vào một ứng dụng mới.



Nhiệm vụ 1: Thay đổi biểu tượng trình khởi chạy

Mỗi ứng dụng mới bạn tạo bằng Android Studio đều bắt đầu bằng một biểu tượng trình chạy tiêu chuẩn đại diện cho ứng dụng. Biểu tượng trình chạy xuất hiện trong danh sách cửa hàng Google Play. Khi người dùng tìm kiếm trên cửa hàng Google Play, biểu tượng cho ứng dụng của bạn sẽ xuất hiện trong kết quả tìm kiếm.

Khi người dùng đã cài đặt ứng dụng, biểu tượng trình khởi chạy sẽ xuất hiện trên thiết bị ở nhiều nơi khác nhau bao gồm màn hình chính và màn hình Tìm kiếm Ứng dụng. Ví dụ: ứng dụng HelloToast xuất hiện trong màn hình Tìm kiếm ứng dụng của trình mô phỏng với biểu tượng tiêu chuẩn cho các dự án ứng dụng mới, như minh họa bên dưới.



Thay đổi biểu tượng trình chạy là một quy trình từng bước đơn giản giới thiệu cho bạn các tính năng nội dung hình ảnh của Android Studio. Trong nhiệm vụ này, bạn cũng tìm hiểu thêm về cách truy cập tài liệu chính thức của Android.

1.1 Khám phá tài liệu chính thức của Android

Bạn có thể tìm thấy tài liệu chính thức dành cho nhà phát triển Android tại developer.android.com. Tài liệu này chứa rất nhiều thông tin được Google cập nhật.

16. Đi đến developer.android.com/design/.

Phần này nói về Material Design, là một triết lý thiết kế khái niệm phác thảo cách các ứng dụng nên trông và hoạt động trên thiết bị di động. Điều hướng các liên kết để tìm hiểu thêm về Material Design. Ví dụ: truy cập phần [Style](#) để tìm hiểu thêm về việc sử dụng màu sắc và các chủ đề khác.

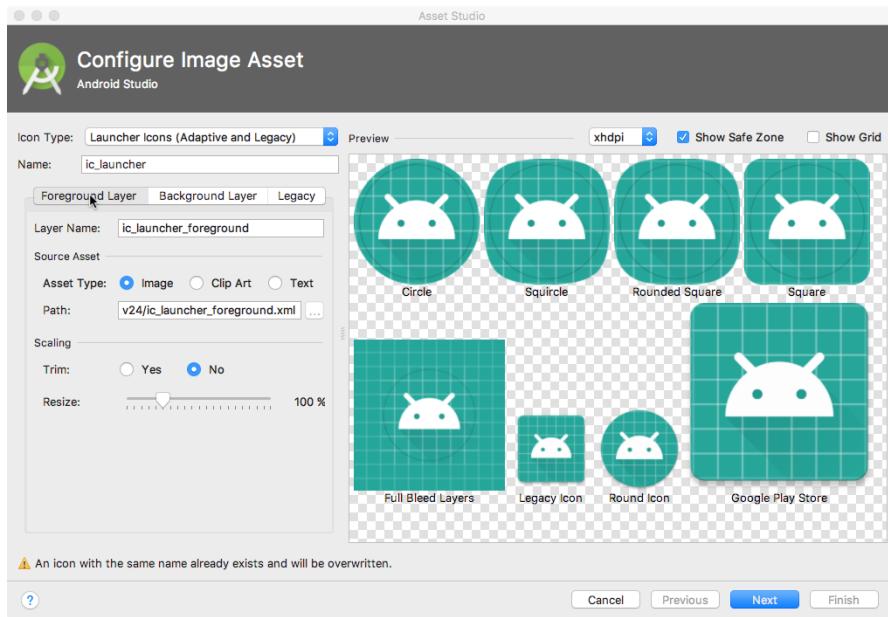
17. Truy cập developer.android.com/docs/ để tìm thông tin API, tài liệu tham khảo, hướng dẫn, hướng dẫn công cụ và mẫu mã.
18. Truy cập developer.android.com/distribute/ để tìm thông tin về việc đưa ứng dụng lên [Google Play](#), hệ thống phân phối kỹ thuật số của Google dành cho các ứng dụng được phát triển bằng Android SDK. Sử dụng [Google Play Console](#) để phát triển cơ sở người dùng của bạn và bắt đầu [kiếm tiền](#).

1.2 Thêm nội dung hình ảnh cho biểu tượng launcher

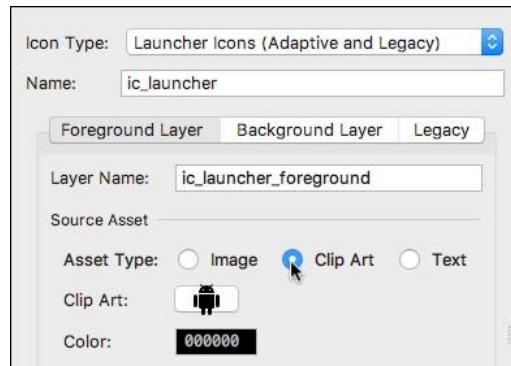
Để thêm hình ảnh clip-art làm biểu tượng trình khởi chạy, hãy làm theo các bước sau:

1. Mở dự án ứng dụng HelloToast từ bài học trước về cách sử dụng trình chỉnh sửa bộ cục hoặc tạo dự án ứng dụng mới.
2. Trong ngăn Project > **Android**, **right nhấp vào** (hoặc **Control-click**) thư mục **res** và chọn **New > Nội dung hình ảnh**. Cửa sổ Định cấu hình nội dung hình ảnh xuất hiện.

Trang 143

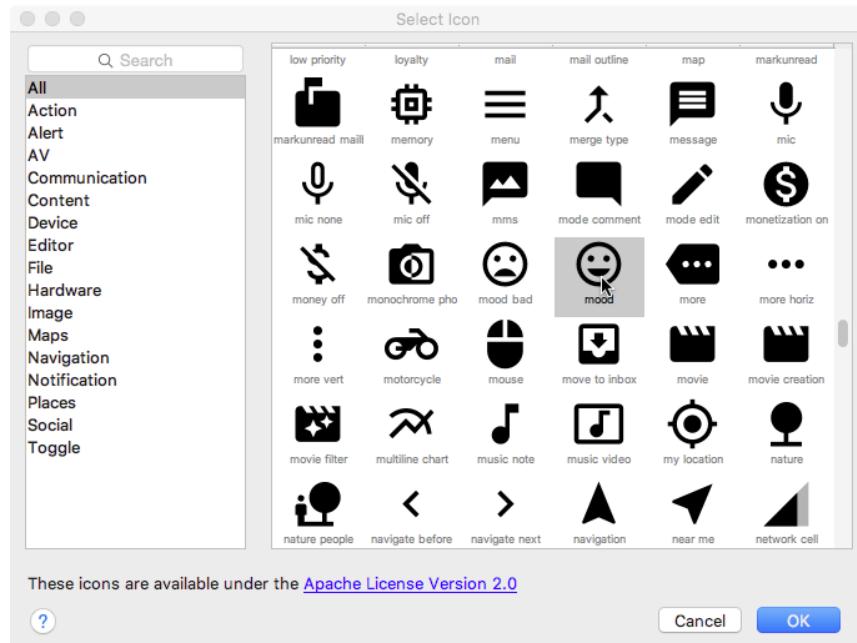


3. Trong trường **I**con **T**ype, chọn **L**auncher **I**cons (**A**daptive & **L**elegacy) nếu nó chưa được chọn.
4. Nhấp vào tab **F**oreground **L**ayer, chọn **C**lip **A**rt cho **A**sset **T**ype.



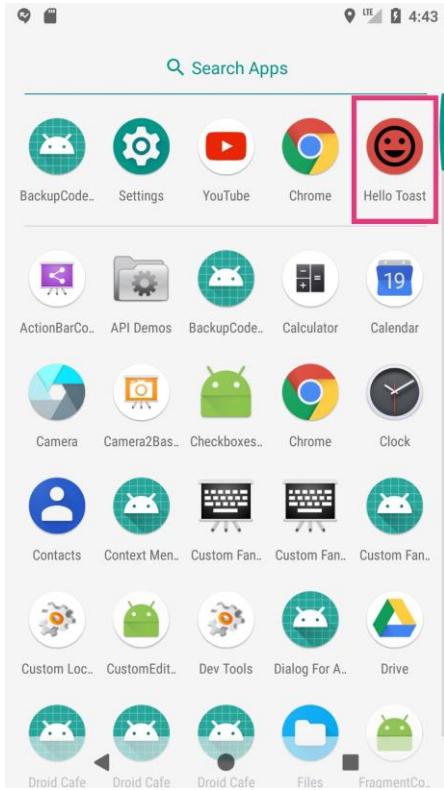
5. Nhấp vào biểu tượng trong trường **Nghệ thuật mội C**. Các biểu tượng xuất hiện từ bộ biểu tượng thiết kế vật liệu.

- Duyệt qua cửa sổ Chọn biểu tượng, chọn một biểu tượng thích hợp (chẳng hạn như biểu tượng tâm trạng để gọi ý tâm trạng tốt), sau đó bấm vào **OK**.



- Nhấp vào tab **B**ackground Layer, chọn **C**olor làm **A**sset Type, sau đó nhấp vào chip màu để chọn màu để sử dụng làm layer nền.
- Nhấp vào tab **L**evelacy và xem lại cài đặt mặc định. Xác nhận rằng bạn muốn tạo các biểu tượng cũ, hình tròn và Cửa hàng Google Play. Nhấp vào **N**ext khi hoàn tất.
- Chạy ứng dụng.

Android Studio tự động thêm hình ảnh trình chạy vào thư mục **mipmap** cho các mật độ khác nhau. Do đó, biểu tượng khởi chạy ứng dụng sẽ thay đổi thành biểu tượng mới sau khi bạn chạy ứng dụng, như hình dưới đây.



Mẹo: Xem [Launcher Icons](#) để tìm hiểu thêm về cách thiết kế các biểu tượng launcher hiệu quả.

Nhiệm vụ 2: Sử dụng mẫu dự án

Android Studio cung cấp các mẫu cho các thiết kế hoạt động và ứng dụng phổ biến và được đề xuất. Sử dụng các mẫu tích hợp giúp tiết kiệm thời gian và giúp bạn tuân theo các phương pháp thiết kế tốt nhất.

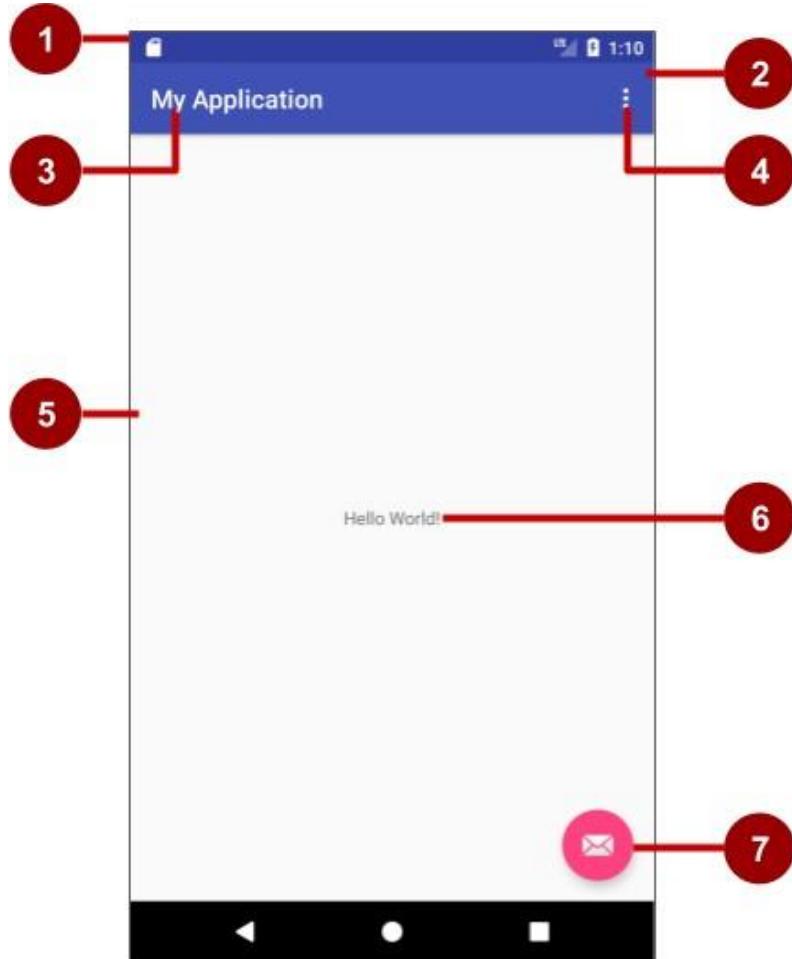
Mỗi mẫu kết hợp một hoạt động khung và giao diện người dùng. Bạn đã sử dụng mẫu Hoạt động trống. Mẫu Hoạt động cơ bản có nhiều tính năng hơn và kết hợp các tính năng ứng dụng được đề xuất, chẳng hạn như menu tùy chọn xuất hiện trong thanh ứng dụng.

2.1 Khám phá kiến trúc Hoạt động cơ bản

Mẫu Basic Activity là một mẫu đa năng do Android Studio cung cấp để hỗ trợ bạn bắt đầu phát triển ứng dụng của mình.

1. Trong Android Studio, hãy tạo một dự án mới bằng mẫu Basic Activity.
2. Xây dựng và chạy ứng dụng.
3. Xác định các bộ phận được dán nhãn trong hình và bảng bên dưới. Tìm các ứng dụng tương đương trên màn hình thiết bị hoặc trình mô phỏng của bạn. Kiểm tra mã Java tương ứng và các tệp XML được mô tả trong bảng.

Làm quen với mã nguồn Java và các tệp XML sẽ giúp bạn mở rộng và tùy chỉnh mẫu này cho nhu cầu của riêng bạn.



Kiến trúc của mẫu Hoạt động cơ bản

#	Mô tả giao diện người dùng	Tham khảo mã
1	Thanh trạng thái Hệ thống Android cung cấp và kiểm soát thanh trạng thái.	Không hiển thị trong mã mẫu. Bạn có thể truy cập nó từ hoạt động của bạn. Ví dụ, bạn có thể sử dụng thanh trạng thái , nếu cần.

2	AppBarLayout > Thanh công cụ Thanh ứng dụng (còn được gọi là thanh hành động) cung cấp cấu trúc trực quan, các yếu tố hình ảnh được chuẩn hóa và điều hướng. Đối với ngược	Trong ctivity_main.xml, hãy tìm android.support.v7.widget.Toolbar
---	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------

reative Commons để cập

nhật mới nhất.

	tương thích, AppBarLayout trong mẫu nhúng một Toolbar với chức năng tương tự như ActionBar .	trong android.support.design.widget.AppBarL ayout. Thay đổi thanh công cụ để thay đổi giao diện của thanh ứng dụng. Đối với một ví dụ, hãy xem Hướng dẫn A pp Bar .
3	Tên ứng dụng Điều này bắt nguồn từ tên gói của bạn, nhưng có thể là bất cứ thứ gì bạn chọn.	Trong một ndroidManifest.xml: android:label="@string/app_name"
4	Nút tràn menu tùy chọn Các mục menu cho hoạt động, cũng như các tùy chọn toàn cầu, chẳng hạn như S earch và S ettings cho ứng dụng. Các mục menu ứng dụng của bạn sẽ đi vào menu này.	Trong MainActivity.java: onOptionsItemSelected() thực hiện những gì xảy ra khi một mục menu được chọn. Res > menu > menu_main.xml Tài nguyên chỉ định các mục menu cho menu tùy chọn.
5	Bộ cục ViewGroup CoordinatorLayout ViewGroup là một bộ cục giàu tính năng cung cấp cơ chế cho các phần tử View (UI) tương tác. Giao diện người dùng của ứng dụng của bạn nằm bên trong tệp content_main.xml có trong tệp ViewGroup.	Trong một ctivity_main.xml: Không có chế độ xem nào được chỉ định trong bộ cục này; thay vào đó, nó bao gồm một bộ cục khác với một lệnh bộ cục bao gồm để bao gồm @layout/content_main nơi các chế độ xem được chỉ định. Điều này tách chế độ xem hệ thống khỏi chế độ xem duy nhất cho ứng dụng của bạn.
6	Chế độ xem văn bản Trong ví dụ, được sử dụng để hiển thị "Hello World". Thay thế điều này bằng các thành phần giao diện người dùng cho ứng dụng của bạn.	Trong content_main.xml: Tất cả các thành phần giao diện người dùng của ứng dụng được xác định trong tệp này.

7	Nút hành động nổi (FAB)	Trong ctivity_main.xml dưới dạng phần tử giao diện người dùng bằng biểu tượng clip-art. MainActivity.java bao gồm một sơ khai trong onCreate() đặt trình nghe onClick() cho FAB.
---	-------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

2.2 Tùy chỉnh ứng dụng do mẫu tạo ra

Thay đổi giao diện của ứng dụng do mẫu Hoạt động cơ bản tạo ra. Ví dụ: bạn có thể thay đổi màu của thanh ứng dụng để phù hợp với thanh trạng thái (trên một số thiết bị có màu tối hơn của cùng một màu cơ bản). Bạn cũng có thể muốn xóa nút hành động nổi nếu bạn không định sử dụng nó.

1. Thay đổi màu của thanh ứng dụng (Toolbar) trong ctivity_main.xml bằng cách thay đổi android:background thành "?attr/colorPrimaryDark", màu này đặt màu thanh ứng dụng thành màu cơ bản tối hơn phù hợp với thanh trạng thái:

```
android:background="?attr/colorPrimaryDark"
```

2. Để xóa nút hành động nổi, hãy bắt đầu bằng cách xóa mã sơ khai trong onCreate() đặt trình nghe onClick() cho nút. Mở **MainActivity** và xóa khỏi mã sau:

```
FloatingActionButton fab = (FloatingActionButton)
    findViewById(R.id.fab);
fab.setOnClickListener(mới View.OnClickListener() {
    @Override
    public void onClick(Xem chế độ xem) {
        Snackbar.make(xem, "Thay thế bằng hành động của riêng bạn",
            Snackbar.LENGTH_LONG).setAction("Hành động", null).show();
    }
});
```

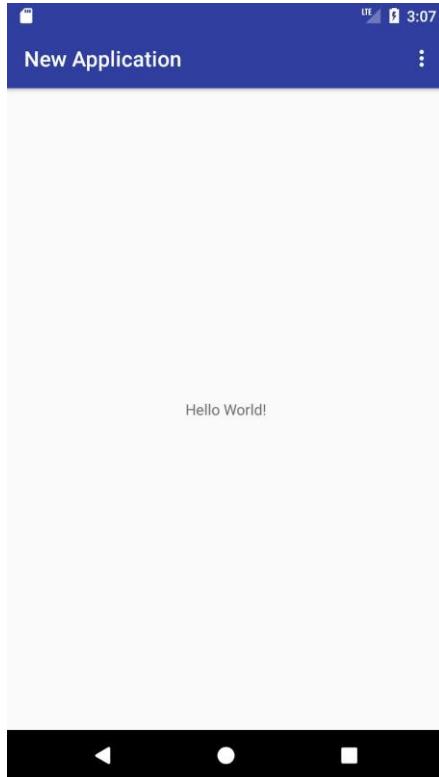
3. Để xóa nút hành động nổi khỏi bố cục, hãy xóa khói mã XML sau đây khỏi activity_main.xml:

```
<android.support.design.widget.FloatingActionButton  
    android:id="@+id/fab"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="dưới cùng | kết thúc"  
    android:layout_margin="@dimen/fab_margin"  
    app:srcCompat="@android:drawable/ic_dialog_email" />
```

4. Thay đổi tên của ứng dụng được hiển thị trong thanh ứng dụng bằng cách thay đổi tài nguyên chuỗi app_name trong strings.xml thành như sau:

```
<string name="app_name">Ứng dụng mới</string>
```

5. Chạy ứng dụng. Nút hành động nổi không còn xuất hiện, tên đã thay đổi và màu nền thanh ứng dụng đã thay đổi.



Mẹo: Xem [Tài nguyên truy cập để](#) biết chi tiết về cú pháp XML để truy cập tài nguyên.

2.3 Khám phá cách thêm hoạt động bằng mẫu

Đối với các bài thực hành cho đến nay, bạn đã sử dụng các mẫu Hoạt động trống và Hoạt động cơ bản. Trong các bài học sau, các mẫu bạn sử dụng khác nhau, tùy thuộc vào nhiệm vụ.

Các mẫu hoạt động này cũng có sẵn từ bên trong dự án của bạn để bạn có thể thêm nhiều hoạt động hơn vào ứng dụng của mình sau khi thiết lập dự án ban đầu. (Bạn tìm hiểu thêm về Một lớp học ctivity trong một chương khác.)

1. Tạo dự án ứng dụng mới hoặc chọn dự án hiện có.
2. Trong ngăn Project > Android, right nhấp vào thư mục Java.
3. Chọn New > Activity > Gallery.

4. Thêm một ctivity. Ví dụ: nhập vào **Hoạt động ngăn kéo hàng không** N để thêm A ctivity với ngăn điều hướng vào ứng dụng của bạn.
5. Nhấp đúp vào các tệp bô cục cho A ctivity để hiển thị chúng trong trình chỉnh sửa bô cục.

Nhiệm vụ 3: Học hỏi từ mã mẫu

Android Studio và tài liệu Android cung cấp nhiều mẫu mã mà bạn có thể nghiên cứu, sao chép và kết hợp với các dự án của mình.

3.1 Mẫu mã Android

Bạn có thể khám phá hàng trăm mẫu mã trực tiếp từ bên trong Android Studio.

1. Trong Android Studio, hãy chọn **File > New > Import Sample**.
2. Duyệt qua các mẫu.
3. Chọn một mẫu và nhập vào **Next**.
4. Chấp nhận mặc định và nhập vào **Finish**.

Lưu ý: Các mẫu có ở đây có nghĩa là điểm khởi đầu cho sự phát triển hơn nữa. Chúng tôi khuyến khích bạn thiết kế và xây dựng ý tưởng của riêng bạn vào chúng.

3.2 Sử dụng Trình quản lý SDK để cài đặt tài liệu ngoại tuyến

Cài đặt Android Studio cũng cài đặt các tính năng cần thiết của Android SDK (Bộ công cụ phát triển phần mềm). Tuy nhiên, các thư viện và tài liệu bổ sung có sẵn và bạn có thể cài đặt chúng bằng cách sử dụng Trình quản lý SDK.

1. Chọn **T ools > Trình quản lý SDK > Android**.
2. Ở cột bên trái, nhấp vào **SDK ndroid**.
3. Chọn và sao chép đường dẫn cho Vị trí SDK Android ở đầu màn hình, vì bạn sẽ cần nó để định vị tài liệu trên máy tính của mình:



4. Nhấp vào tab **Nền tảng S DK**. Bạn có thể cài đặt các phiên bản bổ sung của hệ thống Android từ đây.
5. Nhấp vào tab **S DK Update Sites**. Android Studio thường xuyên kiểm tra các trang web được liệt kê và đã chọn để biết các bản cập nhật.
6. Nhấp vào tab **Công cụ S DK**. Bạn có thể cài đặt các Công cụ SDK bổ sung không được cài đặt theo mặc định, cũng như phiên bản ngoại tuyến của tài liệu dành cho nhà phát triển Android.
7. Chọn hộp kiểm cho "Tài liệu cho SDK Android" nếu nó chưa được cài đặt và nhấp vào **A pply**.
8. Khi quá trình cài đặt hoàn tất, nhấp vào **F inish**.
9. Điều hướng đến thư mục **s dk** mà bạn đã sao chép ở trên và mở **thư mục d ocs**.
10. Tìm **trong dex.html** và mở nó.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

Trang 153

Nhiệm vụ 4: Nhiều tài nguyên khác

- Kênh [YouTube dành cho nhà phát triển Android](#) là một nguồn hướng dẫn và mẹo tuyệt vời.
- Nhóm Android đăng tin tức và mẹo trong [blog Android chuyên nghiệp](#).
- [Stack Overflow](#) là một cộng đồng các lập trình viên giúp đỡ lẫn nhau. Nếu bạn gặp sự cố, khả năng cao là ai đó đã đăng câu trả lời. Hãy thử đăng một câu hỏi như "Làm cách nào để thiết lập và sử dụng ADB qua WiFi?" hoặc "Rò rỉ bộ nhớ phô biến nhất trong quá trình phát triển Android là gì?"
- Và cuối cùng nhưng không kém phần quan trọng, hãy nhập câu hỏi của bạn vào tìm kiếm của Google và công cụ tìm kiếm của Google sẽ thu thập các kết quả có liên quan từ tất cả các tài nguyên này. Ví dụ: "Phiên bản hệ điều hành Android phô biến nhất ở Ấn Độ là gì?"

4.1 Tìm kiếm trên Stack Overflow bằng thẻ

Đi tới [Stack Overflow](#) và nhập [android] vào hộp tìm kiếm. Đầu ngoặc [] cho biết bạn muốn tìm kiếm các bài đăng đã được gắn thẻ là về Android.

Bạn có thể kết hợp thẻ và cụm từ tìm kiếm để làm cho tìm kiếm của bạn cụ thể hơn. Hãy thử các tìm kiếm sau:

- [android] và [bộ cục]
- [Android] "Xin chào thế giới"

Để tìm hiểu thêm về nhiều cách mà bạn có thể tìm kiếm trên Stack Overflow, hãy xem trung [tâm trợ giúp Stack Overflow](#).

Tóm tắt

- Tài liệu chính thức của nhà phát triển Android: [developer.android.com](#)
- Material Design là một triết lý thiết kế khái niệm phác thảo cách các ứng dụng nên giao diện và hoạt động trên thiết bị di động.

- [Cửa hàng Google Play](#) là hệ thống phân phối kỹ thuật số của Google cho các ứng dụng được phát triển bằng SDK Android.
- Android Studio cung cấp các mẫu cho các thiết kế hoạt động và ứng dụng phổ biến và được đề xuất. Các mẫu này cung cấp mã hoạt động cho các trường hợp sử dụng phổ biến.
- Khi tạo dự án, bạn có thể chọn mẫu cho hoạt động đầu tiên của mình.
- Trong khi bạn đang phát triển thêm ứng dụng của mình, các hoạt động và các thành phần ứng dụng khác có thể được tạo từ các mẫu tích hợp.
- Android Studio chứa nhiều mẫu mã mà bạn có thể nghiên cứu, sao chép và kết hợp với các dự án của mình.

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [1.4: Tài nguyên giúp bạn học](#).

Tìm hiểu thêm

Tài liệu Android Studio:

- [Làm quen với Android Studio](#)
- [Thông tin cơ bản về quy trình làm việc dành cho nhà phát triển](#)

Tài liệu dành cho nhà phát triển Android:

- [Trang web dành cho nhà phát triển Android](#)
- [Đào tạo dành cho nhà phát triển của Google](#)
- [Bố trí](#)
- [Tổng quan về tài nguyên ứng dụng](#)
- [Bố trí](#) • [Menu](#) • [Chế độ xem văn bản](#)
- [Tài nguyên chuỗi](#)
- [Tệp kê khai ứng dụng](#)

Mẫu mã:

- [Mã nguồn cho các bài tập trên GitHub](#)
- [Mẫu mã Android dành cho nhà phát triển](#)

Video:

- [Kênh YouTube dành cho nhà phát triển Android](#)
- [Các khóa học trực tuyến của Udacity](#)

Khác:

- [Blog chính thức của Android](#)
- [Blog dành cho nhà phát triển Android](#)
- [Lớp học lập trình dành cho Google Developers](#)
- [Ngăn xếp tràn](#) • [Từ vựng Android](#)

Homework

Tải ứng dụng mẫu và khám phá tài nguyên

1. Tải một trong các ứng dụng mẫu vào Android Studio.
2. Mở một trong các tệp hoạt động Java trong ứng dụng. Tìm một lớp, loại hoặc quy trình mà bạn không quen thuộc và tra cứu trong tài liệu dành cho nhà phát triển Android.
3. Truy cập Stack Overflow và tìm kiếm các câu hỏi về cùng một chủ đề.
4. Thay đổi biểu tượng trình khởi chạy. Sử dụng biểu tượng có sẵn trong phần nội dung hình ảnh của Android Studio.

Trả lời những câu hỏi này

Câu hỏi 1

Trong dự án Android Studio, bạn có thể sử dụng lệnh menu nào để mở danh sách các ứng dụng mẫu? Chọn một:

- Mở > tệp
- Mẫu nhập > mới > tệp
- Tệp > mô-đun nhập > mới

- Tệp > dự án nhập > mới

Câu hỏi 2

Mẫu Hoạt động cơ bản cung cấp những nút nào như một phần của giao diện người dùng? Chọn hai:

- Các nút điều hướng
- Nút tràn menu tùy chọn
- Nút hành động nổi
- Nút lớp nút có văn bản "Nút"

Câu hỏi 3

Nguồn tài liệu nào là tài liệu chính thức dành cho các nhà phát triển Android? Chọn một:

- stackoverflow.com
- officialandroid.blogspot.com
- developer.android.com
- github.com

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kết quả là một ứng dụng mới hoặc một phiên bản Hello Toast, với biểu tượng trình khởi chạy mới xuất hiện trong màn hình Ứng dụng Tìm kiếm của thiết bị Android.

Bài học 2.1: Các sinh hoạt và ý định Giới thiệu

A activity đại diện cho một màn hình duy nhất trong ứng dụng của bạn mà người dùng của bạn có thể thực hiện một tác vụ tập trung duy nhất như chụp ảnh, gửi email hoặc xem bản đồ. Một hoạt động thường được hiển thị cho người dùng dưới dạng cửa sổ toàn màn hình.

Một ứng dụng thường bao gồm nhiều màn hình được liên kết lồng lèo với nhau. Mỗi màn hình là một hoạt động. Thông thường, một hoạt động trong ứng dụng được chỉ định là hoạt động "chính" (MainActivity.java), hoạt động này được hiển thị cho người dùng khi ứng dụng được khởi chạy. Sau đó, hoạt động chính có thể bắt đầu các hoạt động khác để thực hiện các hành động khác nhau.

Mỗi khi một hoạt động mới bắt đầu, hoạt động trước đó sẽ bị dừng, nhưng hệ thống sẽ giữ lại hoạt động trong một ngăn xếp ("ngăn xếp lui"). Khi một hoạt động mới bắt đầu, hoạt động mới đó sẽ được đẩy lên ngăn xếp ngược và thu hút sự tập trung của người dùng. Ngăn xếp ngược tuân theo logic ngăn xếp cơ bản "vào sau, xuất trước". Khi người dùng hoàn tất hoạt động hiện tại và nhấn nút Quay lại, hoạt động đó sẽ được bật ra khỏi ngăn xếp và phá hủy, và hoạt động trước đó sẽ tiếp tục.

Một hoạt động được bắt đầu hoặc kích hoạt bằng một *intent*. Intent là một thông báo không đồng bộ mà bạn có thể sử dụng trong hoạt động của mình để yêu cầu một hành động từ một hoạt động khác hoặc từ một số thành phần ứng dụng khác. Bạn sử dụng ý định để bắt đầu một hoạt động từ một hoạt động khác và chuyển dữ liệu giữa các hoạt động.

An intent có thể là explicit hoặc implicit:

- Ý định explicit là ý định mà bạn biết mục tiêu của ý định đó. Nghĩa là, các em đã biết tên lớp học đủ điều kiện của sinh hoạt cụ thể đó.
- Ý định implicit là một trong đó bạn không có tên của thành phần mục tiêu, nhưng bạn có một hành động chung để thực hiện.

Trong thực tế này, bạn tạo ra các ý định rõ ràng. Bạn tìm hiểu cách sử dụng các ý định ngầm trong một thực tế sau.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy ứng dụng trong Android Studio.
- Sử dụng trình soạn thảo bố cục để tạo bố cục trong ConstraintLayout
- Chính sửa mã XML bố cục.
- Thêm chức năng onClick vào một Button.

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0.
PDF này là Ánh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2

Trang 158

Những gì bạn sẽ học

- Cách tạo Activity mới trong Android Studio.
- Cách xác định hoạt động cha và con cho điều hướng Lên.
- Làm thế nào để bắt đầu một Activity A với một Intent rõ ràng.
- Làm thế nào để truyền dữ liệu giữa mỗi Activity với một Intent rõ ràng.

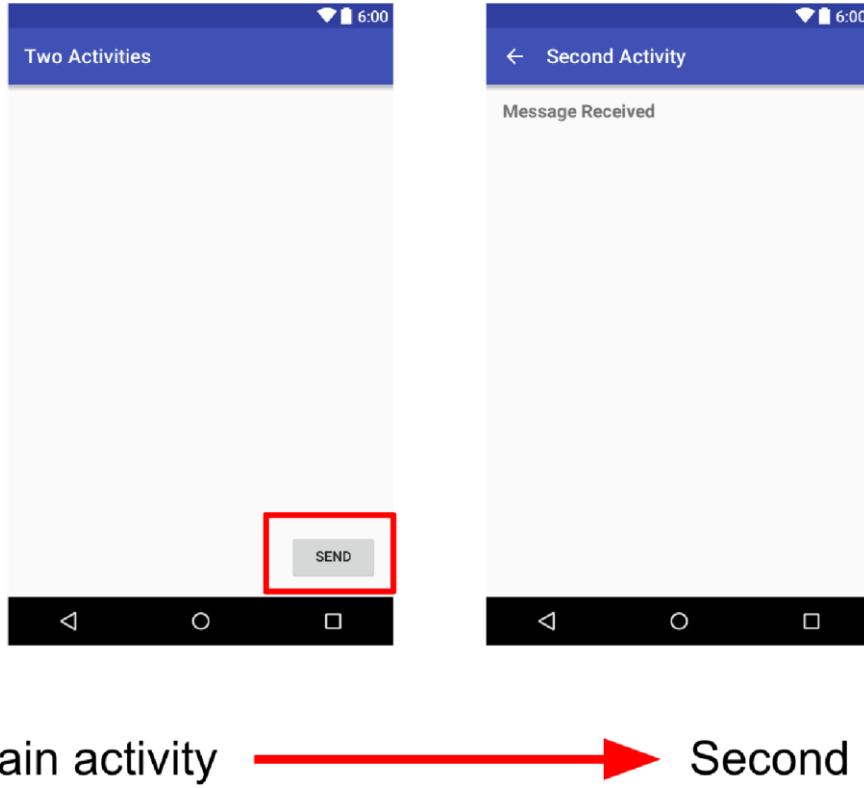
Bạn sẽ làm gì

- Tạo một ứng dụng Android mới với một Activity A chính và một Activity A thứ hai.
- Truyền một số dữ liệu (một chuỗi) từ Activity A chính sang thứ hai bằng cách sử dụng một N và hiển thị dữ liệu đó trong Activity A thứ hai.
- Gửi một bit dữ liệu khác nhau thứ hai trở lại Activity A chính, cũng sử dụng một chữ I.

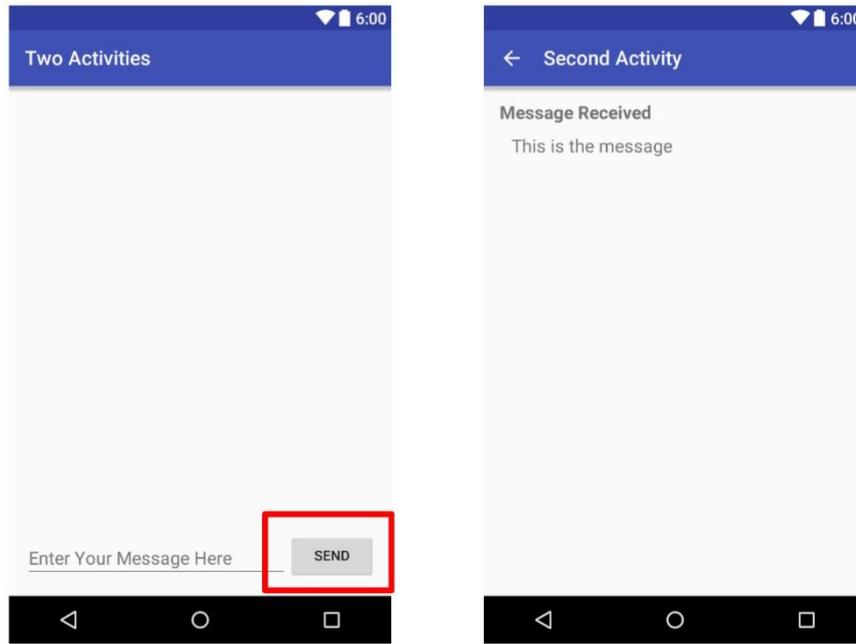
Tổng quan về ứng dụng

Trong chương này, bạn tạo và xây dựng một ứng dụng có tên là Two Activities, không có gì ngạc nhiên khi chứa hai triển khai Activity. Bạn xây dựng ứng dụng theo ba giai đoạn.

Trong giai đoạn đầu tiên, bạn tạo một ứng dụng có hoạt động chính chứa một nút, **S end**. Khi người dùng nhấp vào nút này, hoạt động chính của bạn sẽ sử dụng một ý định để bắt đầu hoạt động thứ hai.

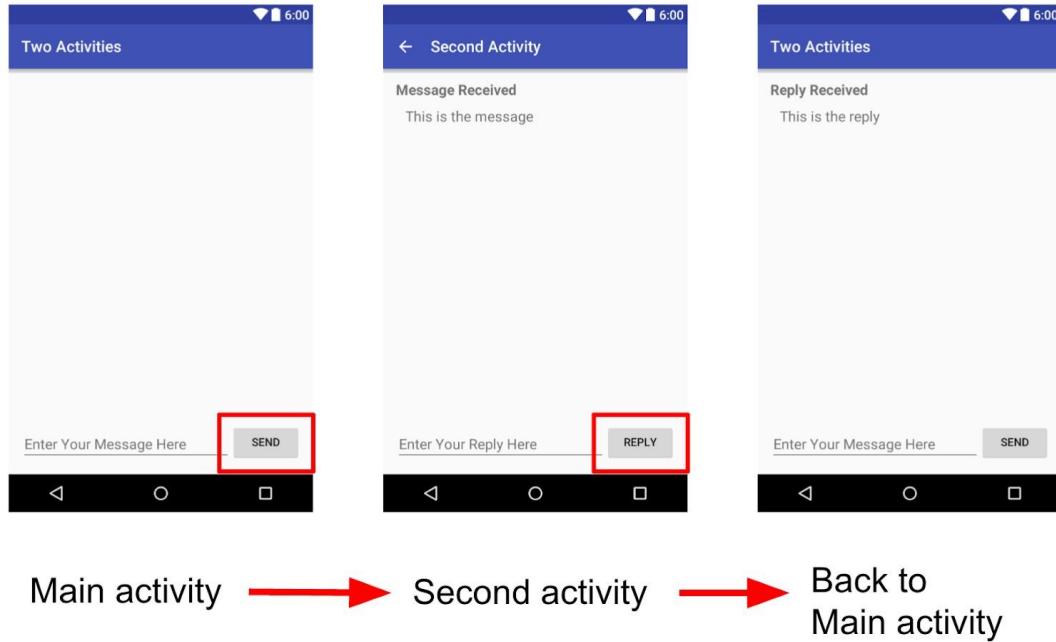


Trong giai đoạn thứ hai, bạn thêm chế độ xem EditText vào hoạt động chính. Người dùng nhập tin nhắn và nhấp vào **Send**. Hoạt động chính sử dụng ý định để bắt đầu hoạt động thứ hai và gửi tin nhắn của người dùng đến hoạt động thứ hai. Hoạt động thứ hai hiển thị tin nhắn mà nó nhận được.



Main activity → Second activity

Trong giai đoạn cuối cùng của việc tạo ứng dụng Two Activities, bạn thêm EditText và nút Reply vào hoạt động thứ hai. Bây giờ người dùng có thể nhập tin nhắn trả lời và nhấn vào Reply, và câu trả lời được hiển thị trên hoạt động chính. Tại thời điểm này, bạn sử dụng ý định để chuyển câu trả lời trở lại từ hoạt động thứ hai sang hoạt động chính.



Nhiệm vụ 1: Tạo dự án TwoActivities

Trong nhiệm vụ này, bạn thiết lập dự án ban đầu với một ctivity A chính, xác định bộ cục và xác định phương thức khung cho sự kiện nút o nClick.

1.1 Tạo dự án TwoActivities

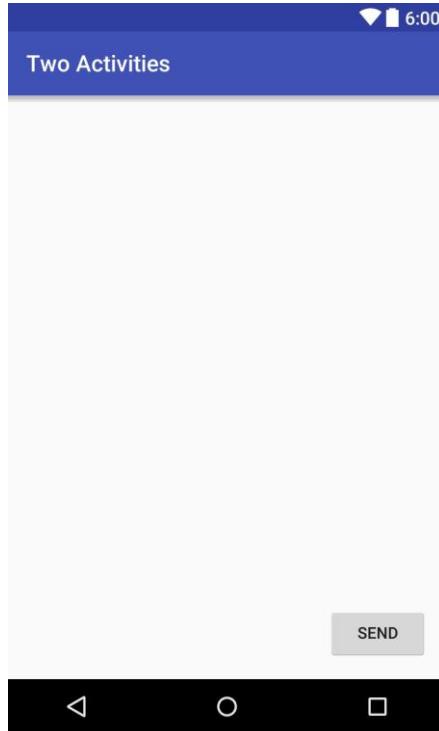
1. Khởi động Android Studio và tạo một dự án Android Studio mới.

Đặt tên cho ứng dụng của bạn là **Hoạt động** và chọn cùng cài đặt **Phone và Máy tính bảng** mà bạn đã sử dụng trong các bài thực hành trước đây. Thư mục dự án được tự động đặt tên là **TwoActivities** và tên ứng dụng xuất hiện trên thanh ứng dụng sẽ là "Hai Hoạt động".

2. Chọn **Empty Activity** cho mẫu Activity. Nhập vào **Name**.
3. Chấp nhận tên Activity mặc định (**MainActivity**). Đảm bảo các **tùy chọn Tệp bối cảnh Generate** và **Tương thích ngược (AppCompat)** được chọn.
4. Nhập vào **Finish**.

1.2 Xác định bối cảnh cho Hoạt động chính

1. Mở **res > bối cảnh > activity_main.xml** trong ngăn **Project > Android**. Trình chỉnh sửa bối cảnh xuất hiện.
2. Nhập vào tab **Design** nếu nó chưa được chọn và xóa **Text View** (cái có nội dung "Hello World") trong ngăn **Cây tùy chỉnh**.
3. Khi Tự động kết nối được bật (cài đặt mặc định), hãy kéo chữ **B** từ ngăn **Palette** sang góc dưới bên phải của bối cảnh. Tự động kết nối tạo ra các ràng buộc cho **Button**.
4. Trong ngăn **Attributes**, đặt **ID** thành **button_main**, **layout_width** và **layout_height** thành **wrap_content** và nhập **Send** cho trường **Value**. Bối cảnh bây giờ sẽ trông như thế này:



- Nhập vào tab **T ext** để chỉnh sửa mã XML. Thêm thuộc tính sau vào chữ B utton:

```
android:onClick="launchSecondActivity"
```

Giá trị thuộc tính được gạch chân màu đỏ vì phương thức `launchSecondActivity()` chưa được tạo. Bỏ qua lỗi này ngay bây giờ; bạn sửa nó trong nhiệm vụ tiếp theo.

- Trích xuất tài nguyên chuỗi, như được mô tả trong thực tế trước đó, cho "Gửi" và sử dụng tên `button_main` cho tài nguyên.

Mã XML cho Button sẽ trông như sau:

```
< Nút
```

Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị

```
    android:id="@+id/button_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/button_main"
    android:onClick="launchSecondActivity"
    app:layout_constraintBottom_toBottomOf="cha me"
    app:layout_constraintRight_toRightOf="cha me" />
```

1.3 Xác định hành động Nút

Trong tác vụ này, bạn triển khai `launchSecondActivity()` m e thod mà bạn đã tham chiéu trong bō cúc cho thuộc tính `android:onClick`.

- Nháp vào "**launchSecondActivity**" trong mã XML `activity_main.xml`.
- Nhấn Alt + Enter (Option + Enter trên máy Mac) và chọn **Create 'launchSecondActivity (View)' in 'MainActivity'**.

Tệp `MainActivity` sẽ mở ra và Android Studio tạo một phương thức khung cho trình xử lý `launchSecondActivity()`.

- Bên trong `launchSecondActivity()`, thêm câu lệnh `Log.d` có nội dung "Button Clicked!"

```
Log.d(LOG_TAG, "Đã nháp nút!");
```

`LOG_TAG` sẽ hiển thị màu đỏ. Bạn thêm định nghĩa cho biến đó trong bước sau.

- Ở đầu lớp `MainActivity`, thêm một hằng số cho biến `LOG_TAG`:

```
riêng tĩnh cuối cùng String LOG_TAG =
    MainActivity.class.getSimpleName();
```

a

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

Trang 165

Hàng số này sử dụng tên của chính lớp làm thẻ.

5. Chạy ứng dụng của bạn. Khi bạn nhấp vào **nút kết thúc S**, bạn sẽ thấy thông báo "Button Clicked!" trong ngăn **Logcat**. Nếu có quá nhiều dấu ra trong màn hình, hãy nhập **MainActivity** vào hộp tìm kiếm và ngăn **Logcat** sẽ chỉ hiển thị các dòng khớp với thẻ đó.

Mã cho **MainActivity** sẽ trông như sau:

```
gói com.example.android.twoactivities;

nhập android.support.v7.app.AppCompatActivity; nhập
android.os.Bundle; nhập android.util.Log; nhập android.view.View;

public class MainActivity extends AppCompatActivity {
    riêng tĩnh cuối cùng String LOG_TAG =
        MainActivity.class.getSimpleName();

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

    public void launchSecondActivity(Ché độ xem xem) {
        Log.d(LOG_TAG, "Đã nhấp nút!");
    }
}
```

Nhiệm vụ 2: Tạo và khởi chạy Hoạt động thứ hai

Mỗi hoạt động mới bạn thêm vào dự án của mình có bộ cục và tệp Java riêng, tách biệt với bộ cục của hoạt động chính. Chúng cũng có các yếu tố hoạt động \leftrightarrow riêng trong tệp AndroidManifest.xml. Cũng như hoạt động chính, việc triển khai hoạt động mới mà bạn tạo trong Android Studio cũng mở rộng từ lớp AppCompatActivity.

Mỗi hoạt động trong ứng dụng của bạn chỉ được kết nối lỏng lẻo với các hoạt động khác. Tuy nhiên, bạn có thể xác định một hoạt động làm cha của một hoạt động khác trong tệp AndroidManifest.xml. Mỗi quan hệ cha con này cho phép Android thêm gợi ý điều hướng như mũi tên hướng trái trên thanh tiêu đề cho từng hoạt động.

Một hoạt động giao tiếp với các hoạt động khác (trong cùng một ứng dụng và trên các ứng dụng khác nhau) với một ý định. An Intent có thể là explicit hoặc implicit:

- Ý định explicit là một ý định trong đó bạn biết mục tiêu của ý định đó; nghĩa là, bạn đã biết tên lớp đủ điều kiện của hoạt động cụ thể đó.
- Ý định implicit là một trong đó bạn không có tên của thành phần mục tiêu, nhưng có một hành động chung để thực hiện.

Trong nhiệm vụ này, bạn thêm một hoạt động thứ hai vào ứng dụng của chúng tôi, với bộ cục riêng của nó. Bạn sửa đổi

AndroidManifest.xml để xác định hoạt động chính là hoạt động mẹ của hoạt động thứ hai. Sau đó, bạn sửa đổi phương thức launchSecondActivity() trong MainActivity để bao gồm một ý định khởi chạy hoạt động thứ hai khi bạn nhấp vào nút.

2.1 Tạo Hoạt động thứ hai

1. Nhập vào thư mục app cho dự án của bạn và chọn File > Hoạt động > mới > Hoạt động trống.
2. Đặt tên cho Activity S econdActivity mới. Đảm bảo Generate Layout File và Backwards Khả năng tương thích (AppCompat) được kiểm tra. Tên bộ cục được điền dưới dạng activity_second. Đừng kiểm tra tùy chọn Hoạt động L.
3. Nhập vào Finish. Android Studio thêm cả bộ cục Activity mới (một activity_second.xml) và một tệp Java mới (SecondActivity.java) vào dự án của bạn cho Activity mới. Nó cũng cập nhật tệp AndroidManifest.xml để bao gồm Activity mới.

2.2 Sửa đổi AndroidManifest.xml file

1. Mở m **anifests > AndroidManifest.xml**.
2. Tìm thành phần <activity> mà Android Studio đã tạo cho phần tử A thứ hai.

<activity android:name=". Hoạt động thứ hai"></hoạt động>

3. Thay thế toàn bộ phần tử > hoạt động <bằng phần tử sau:

```
<activity android:name=". Hoạt động thứ hai"
    android:label = "Hoạt động thứ hai"
    android:parentActivityName=". Hoạt động chính">
    <siêu dữ liệu
        android:name="android.support.PARENT_ACTIVITY"
        android:id =
            "com.example.android.twoactivities.MainActivity" />
</hoạt động>
```

Thuộc tính label thêm tiêu đề của Activity vào thanh ứng dụng.

Với thuộc tính parentActivityName, bạn chỉ ra rằng hoạt động chính là hoạt động gốc của hoạt động thứ hai. Mỗi quan hệ này được sử dụng để điều hướng Lên trong ứng dụng của bạn: thanh ứng dụng cho hoạt động thứ hai sẽ có mũi tên hướng trái để người dùng có thể điều hướng "lên trên" đến hoạt động chính.

Với phần tử siêu dữ liệu <>, bạn cung cấp thông tin tùy ý bổ sung về hoạt động dưới dạng cặp khóa-giá trị. Trong trường hợp này, các thuộc tính siêu dữ liệu thực hiện điều tương tự như thuộc tính android:parentActivityName — chúng xác định mối quan hệ giữa hai hoạt động để điều hướng lên trên. Các thuộc tính siêu dữ liệu này là bắt buộc đối với các phiên bản Android cũ hơn, vì thuộc tính android:parentActivityName chỉ có sẵn cho API cấp 16 trở lên.

4. Trích xuất tài nguyên chuỗi cho "Hoạt động thứ hai" trong mã trên và sử dụng activity2_name làm tên tài nguyên.

2.3 Xác định bối cảnh cho Hoạt động thứ hai

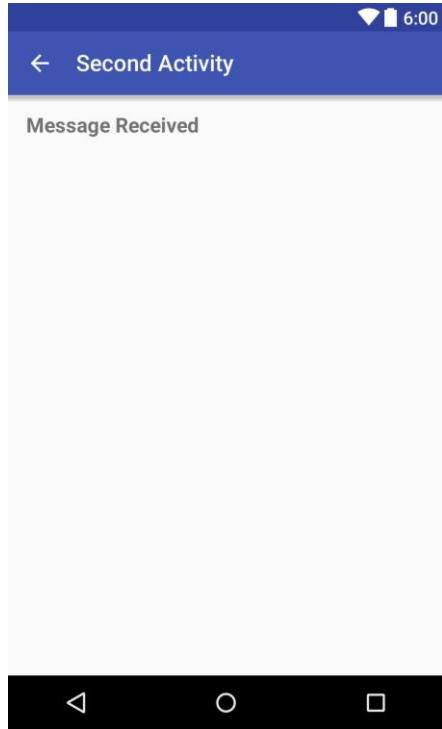
1. Mở **một activity_second.xml** và nhấp vào tab **Design** nếu nó chưa được chọn.

2. Kéo **TextView** từ **ngăn Palette** đến góc trên cùng bên trái của bố cục và thêm các ràng buộc vào phía trên và bên trái của bố cục. Đặt các thuộc tính của nó trong **ngăn Phân bố A** như sau:

Thuộc tính	Giá trị
Id	text_header
Ký quỹ trên	16
Lề trái	8
layout_width	wrap_content
layout_height	wrap_content
Nhắn tin	Tin nhắn nhận được
textXuất hiện	AppCompat.Trung bình
textStyle	B (in đậm)

Giá trị của **textAppearance** là một thuộc tính chủ đề Android đặc biệt xác định các kiểu phông chữ cơ bản. Các em tìm hiểu thêm về các chủ đề trong bài học sau.

Bố cục bây giờ sẽ trông như thế này:



3. Nhập vào tab Text để chỉnh sửa mã XML và trích xuất chuỗi "Message Received" vào một tài nguyên có tên text_header.
4. Thêm thuộc tính android:layout_marginLeft="8dp" vào TextView để bổ sung cho thuộc tính layout_marginStart cho các phiên bản Android cũ hơn.

Mã XML cho ctivity_second.xml phải như sau:

```
<android.support.constraint.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/resauto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context="com.example.android.twoactivities.SecondActivity">  
    <Ché độ xem văn bản  
        android:id="@+id/text_header"  
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="16dp"
    android:text="@string/text_header"
    android:textAppearance=
        "@style/TextAppearance.AppCompat.Medium"
    android:textStyle="đậm"
    app:layout_constraintStart_toStartOf="cha mẹ"
    ứng dụng:layout_constraintTop_toTopOf="cha mẹ" />
</android.support.constraint.ConstraintLayout>
```

2.4 Thêm Ý định vào Hoạt động chính

Trong nhiệm vụ này, bạn thêm một Intent rõ ràng vào activity A chính. Này được sử dụng để kích hoạt activity A thứ hai khi nhập vào nút **kết thúc S**.

1. Mở **MainActivity**.
2. Tạo một Intent mới trong phương thức `launchSecondActivity()`.

Hàm tạo Intent nhận hai đối số cho một Intent rõ ràng: một Context ứng dụng và thành phần cụ thể sẽ nhận được Intent đó. Ở đây bạn nên sử dụng `this` làm Context, và `SecondActivity.class` làm lớp cụ thể:

```
Ý định = ý định mới(this, SecondActivity.class);
```

3. Gọi phương thức `startActivity()` với Intent mới làm đối số.

```
startActivity(ý định);
```

4. Chạy ứng dụng.

Khi bạn nhấp vào nút **kết thúc S**, MainActivity sẽ gửi Intent và hệ thống Android khởi chạy SecondActivity, xuất hiện trên màn hình. Để quay lại MainActivity, hãy bấm vào **nút Up** (mũi tên trái trên thanh ứng dụng) hoặc nút Quay lại ở cuối màn hình.

Nhiệm vụ 3: Gửi dữ liệu từ Hoạt động chính sang Hoạt động thứ hai

Trong nhiệm vụ cuối cùng, bạn đã thêm một ý định rõ ràng vào MainActivity để khởi chạy SecondActivity. Bạn cũng có thể sử dụng ý định để kết thúc dữ liệu từ hoạt động này sang hoạt động khác trong khi khởi chạy nó.

Đối tượng ý định của bạn có thể chuyển dữ liệu đến hoạt động mục tiêu theo hai cách: trong trường dữ liệu hoặc trong các tùy chọn bổ sung ý định. Dữ liệu ý định là một URI cho biết data cụ thể sẽ được hành động. Nếu thông tin bạn muốn chuyển đến một hoạt động thông qua một ý định không phải là URI hoặc bạn có nhiều hơn một phần thông tin bạn muốn gửi, bạn có thể đưa thông tin bổ sung đó vào extras thay thế.

Ý định extras là các cặp khóa/giá trị trong chữ **Bundle**. Bundle là một tập hợp dữ liệu, được lưu trữ dưới dạng cặp khóa/giá trị. Để chuyển thông tin từ hoạt động này sang hoạt động khác, bạn đặt các khóa và giá trị vào Bundle bổ sung ý định từ hoạt động gửi, sau đó đưa chúng trở lại trong hoạt động nhận.

Trong tác vụ này, bạn sửa đổi ý định rõ ràng trong MainActivity để bao gồm dữ liệu bổ sung (trong trường hợp này là chuỗi do người dùng nhập) trong ý định bổ sung Bundle. Sau đó, bạn sửa đổi SecondActivity để lấy dữ liệu đó trả lại mục đích bổ sung và hiển thị nó trên màn hình.

3.1 Thêm EditText vào bộ cục MainActivity

- Mở **activity_main.xml**.
- Kéo phần tử Plain Text (EditText) từ ngăn Palette xuống cuối bộ cục và thêm các ràng buộc vào phía bên trái của bộ cục, dưới cùng của bộ cục và phía bên trái của **Send** Button. Đặt các thuộc tính của nó trong **Phân bố A** như sau:

Thuộc tính	Giá trị
Id	editText_main
Lề phải	8

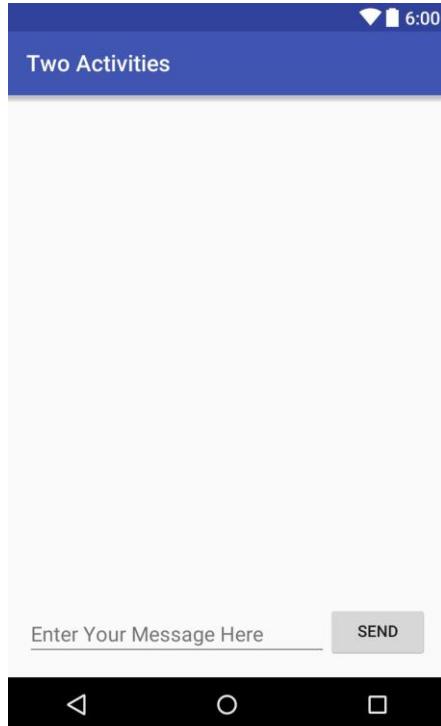
Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị

Lề trái	8
Lề dưới	16
layout_width	match_constraint
layout_height	wrap_content
Loại đầu vào	textLongMessage
kháy	Nhập tin nhắn của bạn tại đây
Nhắn tin	(Xóa bất kỳ văn bản nào trong trường này)

Bô cục mới trong ctivity_main.xml trông như thế này:

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.



3. Nhập vào tab **T ext** để chỉnh sửa mã XML và trích xuất chuỗi "Nhập tin nhắn của bạn ở đây" vào một tài nguyên có tên `e ditText_main`.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

Trang 173

Mã XML cho bộ cục sẽ trông giống như sau.

```
<android.support.constraint.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.MainActivity">

    < Nút
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/button_main"
        android:onClick="launchSecondActivity"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />
    <EditText
        android:id="@+id/editText_main"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:ems="10"
        android:hint="@string/editText_main"
        android:inputType="textLongMessage"
        app:layout_constraintBottom_toBottomOf="cha me"
        app:layout_constraintEnd_toStartOf="@+id/button2" app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

3.2 Thêm một chuỗi vào các tính năng bổ sung của Ý định

I intent extra là các cặp khóa/giá trị trong một chữ B. B undle là một tập hợp dữ liệu, được lưu trữ dưới dạng cặp khóa/giá trị. Để truyền thông tin từ Activity này sang Activity khác, bạn đặt các khóa và giá trị vào Intent extra B undle từ Activity A gửi, và sau đó đưa chúng trở lại khi nhận

Hoạt động.

1. Mở MainActivity.
2. Thêm một hàng số public ở đầu lớp để xác định khóa cho Intent extra:

```
public static final String EXTRA_MESSAGE =  
        "com.example.android.twoactivities.extra.MESSAGE";
```

3. Thêm một biến riêng ở đầu lớp để giữ EditText:

```
riêng EditText mMessageEditText;
```

4. Trong phương thức onCreate(), sử dụng findViewById() để lấy tham chiếu đến EditText và gán nó cho biến riêng đó:

```
mMessageEditText = findViewById(R.id.editText_main);
```

5. Trong phương thức launchSecondActivity(), ngay bên dưới nút Intent mới, lấy văn bản từ EditText dưới dạng chuỗi:

```
Thông báo chuỗi = mMessageEditText.getText().toString();
```

6. Thêm chuỗi đó vào Intent như một phần bổ sung với key XTRA_MESSAGE làm khóa và chuỗi làm giá trị:

```
intent.putExtra(EXTRA_MESSAGE, tin nhắn);
```

Phương thức onCreate() trong MainActivity bây giờ sẽ trông như sau:

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

Phương thức launchSecondActivity() trong MainActivity bây giờ sẽ trông như sau:

```
public void launchSecondActivity(Ché độ xem xem) {  
    Log.d(LOG_TAG, "Đã nhấp nút!");  
    Ý định = ý định mới(this, SecondActivity.class);  
    Thông báo chuỗi = mMessageEditText.getText().toString();  
    intent.putExtra(EXTRA_MESSAGE, tin nhắn);  
    startActivity(ý định); }
```

3.3 Thêm TextView vào SecondActivity cho tin nhắn

1. Mở một activity_second.xml.

2. Kéo **một TextView** khác vào bố cục bên dưới `text_header` `TextView` và thêm các ràng buộc vào phía bên trái của bố cục và vào cuối `text_header`.
3. Đặt các thuộc tính `TextView` mới trong ngăn Phân **bô A** như sau:

Thuộc tính	Giá trị
Id	<code>text_message</code>
Ký quỹ trên	8
Lề trái	8
<code>layout_width</code>	<code>wrap_content</code>
<code>layout_height</code>	<code>wrap_content</code>
Nhấn tin	(Xóa bất kỳ văn bản nào trong trường này)
<code>textXuất hiện</code>	<code>AppCompat.Trung bình</code>

Bố cục mới trông giống như trong tác vụ trước, bởi vì `TextView` mới (chưa) không chứa bất kỳ văn bản nào và do đó không xuất hiện trên màn hình.

Mã XML cho bố cục `activity_second.xml` sẽ trông giống như sau:

```
<android.support.constraint.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/resauto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context="com.example.android.twoactivities.SecondActivity">  
  
<Ché độ xem văn bản  
    android:id="@+id/text_header"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"
```

```
    android:layout_marginTop="16dp"
    android:text="@string/text_header"
    android:textAppearance=
        "@style/TextAppearance.AppCompat.Medium"
    android:textStyle="đậm"
    app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent" />

<Ché độ xem văn bản
    android:id="@+id/text_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    app:layout_constraintStart_toStartOf="cha me"
    app:layout_constraintTop_toBottomOf="@+id/text_header" />
</android.support.constraint.ConstraintLayout>
```

3.4 Sửa đổi SecondActivity để nhận các tính năng bổ sung và hiển thị thông báo

1. Mở **S econdActivity** để thêm mã vào phương thức `onCreate()`.
2. Nhận `I ntent` đã kích hoạt `Activity` A này:

```
Ý định = getIntent();
```

3. Lấy chuỗi chứa thông điệp từ các tính năng bổ sung của `I ntent` bằng cách sử dụng `MainActivity.EXTRA_MESSAGE` biến tĩnh làm khóa:

```
Thông báo chuỗi = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

a

for the latest updates.

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0. PDF này là Ánh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2

4. Sử dụng `findViewById()` để lấy tham chiếu đến `TextView` cho thông báo từ bộ cục:

```
TextView textView = findViewById(R.id.text_message);
```

5. Đặt văn bản của `TextView` thành chuỗi từ `Intent extra`:

```
textView.setText(tin nhắn);
```

6. Chạy ứng dụng. Khi bạn nhập một tin nhắn trong `MainActivity` và nhấp vào **S end**, `SecondActivity` sẽ khởi chạy và hiển thị thông báo đó.

Phương thức `SecondActivity` `onCreate()` sẽ trông như sau:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_second);  
    Ý định = getIntent();  
    Thông báo chuỗi = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);  
    TextView textView = findViewById(R.id.text_message);  
    textView.setText(tin nhắn); }
```

Nhiệm vụ 4: Trả dữ liệu trả lại Hoạt động chính

Bây giờ bạn đã có một ứng dụng khởi chạy một hoạt động mới và gửi dữ liệu đến hoạt động đó, bước cuối cùng là trả lại dữ liệu từ hoạt động thứ hai trả lại hoạt động chính. Bạn cũng sử dụng ý định và ý định *cho* tác vụ này.

4.1 Thêm EditText và Button vào bố cục SecondActivity

1. Mở `strings.xml` và thêm tài nguyên chuỗi cho văn bản Button và gọi ý cho EditText mà bạn sẽ thêm vào SecondActivity:

```
<string name="button_second">Trả lời</chuỗi>
<string name="editText_second">Nhập câu trả lời của bạn vào đây</string>
```

2. Mở `activity_main.xml` và `activity_second.xml`.
3. Sao chép EditText và Button từ tệp bố cục activity_main.xml và Paste chúng vào bố cục activity_second.xml.
4. Trong một activity_second.xml, sửa đổi các giá trị thuộc tính cho chữ B như sau:

Giá trị thuộc tính cũ	Giá trị thuộc tính mới
<code>android:id="@+id/button_main"</code>	<code>android:id="@+id/button_second"</code>
<code>android:onClick="launchSecondActivity"</code>	<code>android:onClick="trả lời"</code>
<code>android:text="@string/button_main"</code>	<code>android:text="@string/button_second"</code>

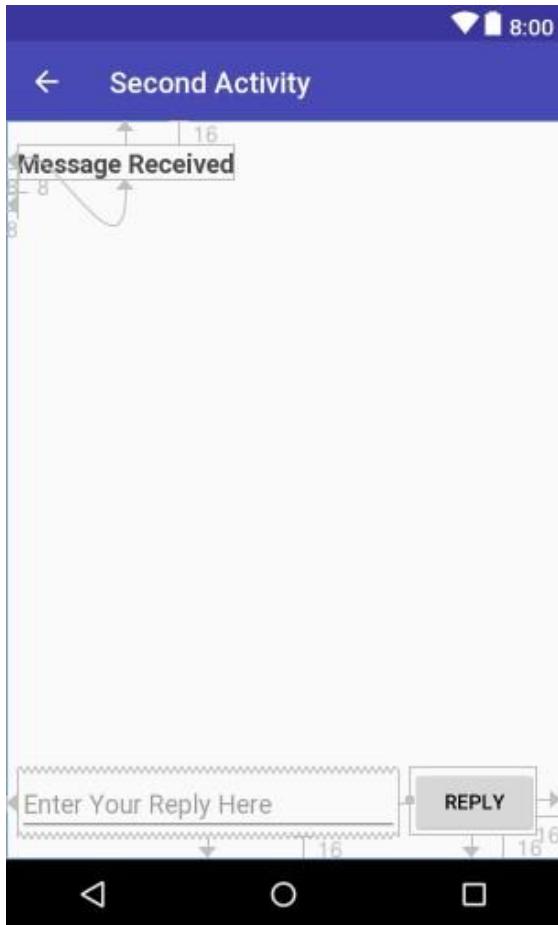
5. Trong ctivity_second.xml, sửa đổi các giá trị thuộc tính cho EditText như sau:

Giá trị thuộc tính cũ	Giá trị thuộc tính mới
android:id="@+id/editText_main"	android:id="@+id/editText_second"
app:layout_constraintEnd_toStartOf="@+id/nút"	ứng dụng:layout_constraintEnd_toStartOf="@+id/button_second"
android: gọi ý = "@string/editText_main"	android: gọi ý = "@string/editText_second"

6. Trong trình soạn thảo bô cục XML, nhập vào **r eturnReply**, nhấn Alt+Enter (Option+Return trên Mac) và chọn **C reate 'returnReply(View)' trong 'SecondActivity'**.

Android Studio tạo một phương thức khung cho trình xử lý returnReply(). Bạn thực hiện phương pháp này trong nhiệm vụ tiếp theo.

Bô cục mới cho ctivity_second.xml trông như sau:



Mã XML cho tệp bố cục activity_second.xml như sau:

```
<android.support.constraint.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.SecondActivity">
    <Ché độ xem văn bản
```

```
    android:id="@+id/text_header"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="16dp"
    android:text="@string/text_header"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium" android:textStyle="đậm"
    app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent" />

<Ché đố xem văn bản
    android:id="@+id/text_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    app:layout_constraintStart_toStartOf="cha me"
    app:layout_constraintTop_toBottomOf="@+id/text_header" />

< Nút
    android:id="@+id/button_second"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginRight="16dp"
    android:text="@string/button_second"
    android:onClick="trả lời"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent" />
<EditText
    android:id="@+id/editText_second"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginEnd="8dp" android:layout_marginStart="8dp"
    android:ems="10"
    android:hint="@string/editText_second"
    android:inputType="textLongMessage"
    app:layout_constraintBottom_toBottomOf="cha me"
```

```
        úng dụng:layout_constraintEnd_toStartOf="@+id/button_second"
        app:layout_constraintStart_toStartOf="cha mè" />
</android.support.constraint.ConstraintLayout>
```

4.2 Tạo Ý định phản hồi trong Hoạt động thứ hai

Dữ liệu phản hồi từ ctivity A thứ hai trả lại ctivity A chính được gửi trong một Intent extra. Bạn xây dựng return Intent này và đưa dữ liệu vào đó theo cách tương tự như cách bạn làm cho việc gửi Intent.

1. Mở S econdActivity.
2. Ở đầu lớp, thêm một hàng số công cộng để xác định khóa cho Intent extra:

```
public static final String EXTRA_REPLY =
    "com.example.android.twoactivities.extra.REPLY";
```

3. Thêm một biến riêng ở đầu lớp để giữ EditText.

```
riêng EditText mReply;
```

4. Trong phương thức onCreate(), trước mã Intent, hãy sử dụng findViewById() để lấy tham chiếu đến EditText và gán nó cho biến riêng đó:

```
mReply = findViewById(R.id.editText_second);
```

5. Trong phương thức returnReply(), lấy văn bản của EditText dưới dạng chuỗi:

```
Trả lời chuỗi = mReply.getText().toString();
```

6. Trong phương thức `onActivityResult()`, tạo một intent mới cho phản hồi—*don't reuse* sử dụng lại đối tượng Intent mà bạn nhận được từ yêu cầu ban đầu.

```
Ý định replyIntent = ý định mới ();
```

7. Thêm chuỗi reply từ EditText vào ý định mới dưới dạng Intent extra. Bởi vì extras là các cặp khóa/giá trị, ở đây khóa là EXTRA_REPLY và giá trị là reply:

```
replyIntent.putExtra(EXTRA_REPLY, trả lời);
```

8. Đặt kết quả thành RESULT_OK để cho biết rằng phản hồi đã thành công. Lớp Activity xác định mã kết quả, bao gồm RESULT_OK và RESULT_CANCELED.

```
setResult(RESULT_OK, trả lờiIntent);
```

9. Gọi `finish()` để đóng Activity và quay lại MainActivity.

```
kết thúc();
```

Mã cho SecondActivity bây giờ sẽ như sau:

```
public class SecondActivity mở rộng AppCompatActivity {
    public static final String EXTRA_REPLY =
        "com.example.android.twoactivities.extra.REPLY";
    riêng EditText mReply;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SetContentView(R.layout.activity_second);
        mReply = findViewById(R.id.editText_second);
        Ý định = getIntent();
        Thông báo chuỗi = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
        TextView textView = findViewById(R.id.text_message);    textView.setText(tin nhắn);
    }

    public void returnReply(Xem ché độ xem) {
        Trả lời chuỗi = mReply.getText().toString();
        Ý định replyIntent = ý định mới ();
        replyIntent.putExtra(EXTRA_REPLY, trả lời);
        setResult(RESULT_OK, trả lờiIntent);
        kết thúc();
    }
}
```

4.3 Thêm các phần tử TextView để hiển thị câu trả lời

MainActivity cần một cách để hiển thị câu trả lời mà SecondActivity gửi. Trong tác vụ này, bạn thêm các phần tử TextView vào bố cục activity_main.xml để hiển thị câu trả lời trong MainActivity. Để làm cho nhiệm vụ này dễ dàng hơn, bạn sao chép các phần tử TextView mà bạn đã sử dụng trong SecondActivity.

1. Mở **strings.xml** và thêm tài nguyên chuỗi cho tiêu đề trả lời:

```
<string name="text_header_reply">Reply Received</string>
```

2. Mở **một ctivity_main.xml** và **một ctivity_second.xml**.
3. Sao chép hai phần tử `TextView` từ tệp bố cục `activity_second.xml` và dán chúng vào bố cục `activity_main.xml` phía trên Button.
4. Trong `activity_main.xml`, sửa đổi các giá trị thuộc tính cho `TextView` đầu tiên như sau:

Giá trị thuộc tính cũ	Giá trị thuộc tính mới
<code>android:id="@+id/text_header"</code>	<code>android:id="@+id/text_header_reply"</code>
<code>android:text="</code> <code>"@string/text_header"</code>	<code>android:text="</code> <code>"@string/text_header_reply"</code>

5. Trong một `activity_main.xml`, sửa đổi các giá trị thuộc tính cho `TextView` thứ hai như sau:

Giá trị thuộc tính cũ	Giá trị thuộc tính mới
<code>android:id="@+id/text_message"</code>	<code>android:id="@+id/text_message_reply"</code>
<code>android:layout_constraintTop_toBottomOf="@+id/text_header"</code>	<code>android:layout_constraintTop_toBottomOf="@+id/text_header_reply"</code>

6. Thêm thuộc tính `android:visibility` vào mỗi `TextView` để làm cho chúng không hiển thị ban đầu.
(Việc hiển thị chúng trên màn hình, nhưng không có bất kỳ nội dung nào, có thể gây nhầm lẫn cho người dùng.)

`android:visibility="vô hình"`

Bạn sẽ làm cho các phần tử TextView này hiển thị sau khi dữ liệu phản hồi được truyền trở lại từ lần A thứ hai.

Bố cục Activity_main.xml trông giống như trong tác vụ trước đó — mặc dù bạn đã thêm hai phần tử TextView mới vào bố cục. Vì bạn đặt các yếu tố này thành vô hình, chúng không xuất hiện trên màn hình.

Sau đây là mã XML cho tệp activity_main.xml:

```
<android.support.constraint.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.twoactivities.MainActivity">
    <Ché độ xem văn bản
        android:id="@+id/text_header_reply"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header_reply"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:textStyle="đậm"
        android:visibility="vô hình"
        app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent" />
    <Ché độ xem văn bản
        android:id="@+id/text_message_reply"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:visibility="vô hình"
        app:layout_constraintStart_toStartOf="cha me"
        app:layout_constraintTop_toBottomOf="@+id/text_header_reply" />
    < Nút
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/button_main"
```

```
    android:onClick="launchSecondActivity"
    app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintRight_toRightOf="parent" />
<EditText
    android:id="@+id/editText_main"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginEnd="8dp" android:layout_marginStart="8dp"
    android:ems="10"
    android:hint="@string/editText_main"
    android:inputType="textLongMessage"
    app:layout_constraintBottom_toBottomOf="cha mẹ"
    ứng dụng:layout_constraintEnd_toStartOf="@+id/button2"
    app:layout_constraintStart_toStartOf="cha mẹ" />
</android.support.constraint.ConstraintLayout>
```

4.4 Nhận câu trả lời từ Ý định bổ sung và hiển thị nó

Khi bạn sử dụng một Intent rõ ràng để bắt đầu một Activity A khác, bạn có thể không mong đợi nhận lại bất kỳ dữ liệu nào — bạn chỉ đang kích hoạt A Activity đó. Trong trường hợp đó, bạn sử dụng startActivity() để bắt đầu Activity mới, như bạn đã làm trước đó trong thực tế này. Tuy nhiên, nếu bạn muốn lấy lại dữ liệu từ Hoạt động đã kích hoạt, bạn cần bắt đầu nó bằng startActivityForResult().

Trong tác vụ này, bạn sửa đổi ứng dụng để khởi động SecondActivity mong đợi kết quả, trích xuất dữ liệu trả về từ N và hiển thị dữ liệu đó trong các phần tử TextView mà bạn đã tạo trong tác vụ trước.

1. Mở MainActivity.
2. Thêm một hằng số công khai ở đầu lớp để xác định khóa cho một loại phản hồi cụ thể mà bạn quan tâm:

```
public static final int TEXT_REQUEST = 1;
```

3. Thêm hai biến riêng tư để giữ các phần tử trả lời và trả lời TextView :

```
TextView riêng mReplyHeadTextView; TextView riêng  
mReplyTextView;
```

4. Trong phương thức `onCreate()`, hãy sử dụng `findViewById()` để lấy các tham chiếu từ bố cục đến phần tử trả lời và trả lời `EditText`. Gán các phiên bản chế độ xem đó cho các biến riêng tư:

```
mReplyHeadTextView = findViewById(R.id.text_header_reply); mReplyTextView =  
findViewById(R.id.text_message_reply);
```

Phương thức `onCreate()` đầy đủ bây giờ sẽ trông như sau:

```
@Override  
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    mMessageEditText = findViewById(R.id.editText_main);  
    mReplyHeadTextView = findViewById(R.id.text_header_reply);  
    mReplyTextView = findViewById(R.id.text_message_reply); }
```

5. Trong phương thức `launchSecondActivity()`, thay đổi lệnh gọi `startActivity()` thành `startActivityForResult()` và bao gồm khóa `TEXT_REQUEST` làm đối số: `startActivityForResult(intent, TEXT_REQUEST)`;
6. Ghi đè phương thức `onActivityResult()` bằng chữ ký này:

```
@Override null công khai onActivityResult(int requestCode,  
        int resultCode, Dữ liệu ý định) { }
```

Bà đối số cho `onActivityResult()` chứa tất cả thông tin bạn cần để xử lý dữ liệu trả về: `requestCode` bạn đặt khi khởi chạy Activity với `startActivityForResult()`, `resultCode` được đặt trong Activity đã khởi chạy (thường là một trong các `RESULT_OK` hoặc `RESULT_CANCELED`) và dữ liệu Intent chứa dữ liệu được trả về từ vụ phỏng Activity.

7. Bên trong `onActivityResult()`, hãy gọi `super.onActivityResult()`:

```
super.onActivityResult(requestCode, resultCode, data);
```

8. Thêm mã để kiểm tra `TEXT_REQUEST` để đảm bảo bạn xử lý đúng kết quả, trong trường hợp có một số. Đồng thời kiểm tra `RESULT_OK`, để đảm bảo rằng yêu cầu đã thành công:

```
if(requestCode == TEXT_REQUEST) { if(resultCode == RESULT_OK) {  
    }  
}
```

Lớp `Activity` xác định mã kết quả. Mã có thể là `RESULT_OK` (yêu cầu thành công), `RESULT_CANCELED` (người dùng đã hủy thao tác) hoặc `RESULT_FIRST_USER` (để xác định mã kết quả của riêng bạn).

9. Bên trong khối if bên trong, nhận thêm Intent từ phản hồi Intent (data). Ở đây chìa khóa cho phần bổ sung là hàng số `EXTRA_REPLY` từ `SecondActivity`:

```
Trả lời chuỗi = data.getStringExtra(SecondActivity.EXTRA_REPLY);
```

10. Đặt khả năng hiển thị của tiêu đề trả lời thành true:

```
mReplyHeadTextView.setVisibility(View.VISIBLE);
```

11. Đặt văn bản trả lời TextView thành reply và đặt khả năng hiển thị của nó thành true:

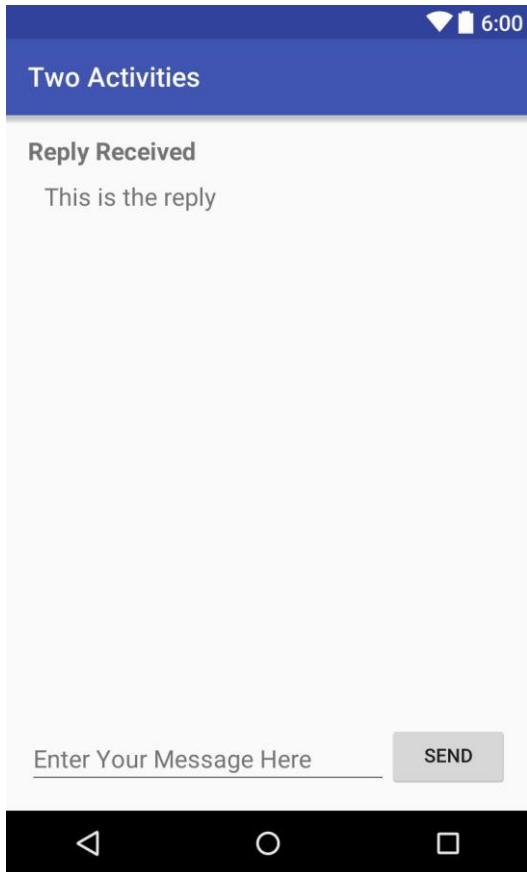
```
mReplyTextView.setText(trả lời); mReplyTextView.setVisibility(View.VISIBLE);
```

Phương thức onActivityResult() đầy đủ bây giờ sẽ trông như sau:

```
@Override null công khai onActivityResult(int requestCode,
        int resultCode, Dữ liệu ý định) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == TEXT_REQUEST) { if
        (resultCode == RESULT_OK) {
            Trả lời chuỗi =
                data.getStringExtra(SecondActivity.EXTRA_REPLY);
            mReplyHeadTextView.setVisibility(View.VISIBLE);
            mReplyTextView.setText(trả lời);
            mReplyTextView.setVisibility(View.VISIBLE);
        }
    }
}
```

12. Chạy ứng dụng.

Bây giờ, khi bạn gửi tin nhắn đến Activity thứ hai và nhận được câu trả lời, Activity chính sẽ cập nhật để hiển thị câu trả lời.



Mã giải pháp

Dự án Android Studio: [TwoActivities](#)

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 này.

Thử thách: C tạo ra một ứng dụng có ba phần tử Button được dán nhãn Text One, Text Two và Text Three. Khi bất kỳ phần tử nào trong số các yếu tố Button này được nhấp vào, hãy khởi động một Activity A thứ hai. Activity thứ hai đó phải chứa một ScrollView hiển thị một trong ba đoạn văn bản (Bạn có thể bao gồm lựa chọn đoạn văn của mình). Sử dụng chữ I để khởi chạy Activity A thứ hai với các tính năng bổ sung để chỉ ra đoạn nào trong ba đoạn văn cần hiển thị.

Tóm tắt

Tổng quan:

- Activity là một thành phần ứng dụng cung cấp một màn hình tập trung vào một tác vụ của người dùng.
- Mỗi Activity có tệp bố cục giao diện người dùng riêng.
- Bạn có thể chỉ định hệ thống triển khai Activity của mình một mối quan hệ cha / con để bắt đầu hướng dẫn trong ứng dụng của bạn.
- View có thể được hiển thị hoặc ẩn với thuộc tính android:visibility.

Để thực hiện Activity:

- Chọn File > Hoạt động > mới để bắt đầu từ mẫu và tự động thực hiện các bước sau.
- Nếu không bắt đầu từ một mẫu, hãy tạo một lớp Java Activity, triển khai một giao diện người dùng cơ bản cho hoạt động trong một tệp bố cục XML được liên kết và khai báo Activity mới trong AndroidManifest.xml.

Ý định:

- An Intent cho phép bạn yêu cầu một hành động từ một thành phần khác trong ứng dụng của bạn, ví dụ: để bắt đầu một thành phần A từ một thành phần khác. Một Intent có thể rõ ràng hoặc ngầm.

- Với một Intent rõ ràng, bạn chỉ ra thành phần mục tiêu cụ thể để nhận dữ liệu.
- Với một Intent ngầm, bạn chỉ định chức năng bạn muốn nhưng không phải thành phần mục tiêu.
- Một Intent có thể bao gồm dữ liệu để thực hiện một hành động (dưới dạng URI) hoặc thông tin bổ sung dưới dạng Intent extras.
- Ý định extras là các cặp khóa/giá trị trong một chữ B được gửi cùng với chữ I.

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [2.1: Hoạt động và ý định](#).

Tìm hiểu thêm

Tài liệu Android Studio:

- [Làm quen với Android Studio](#)

Tài liệu dành cho nhà phát triển Android:

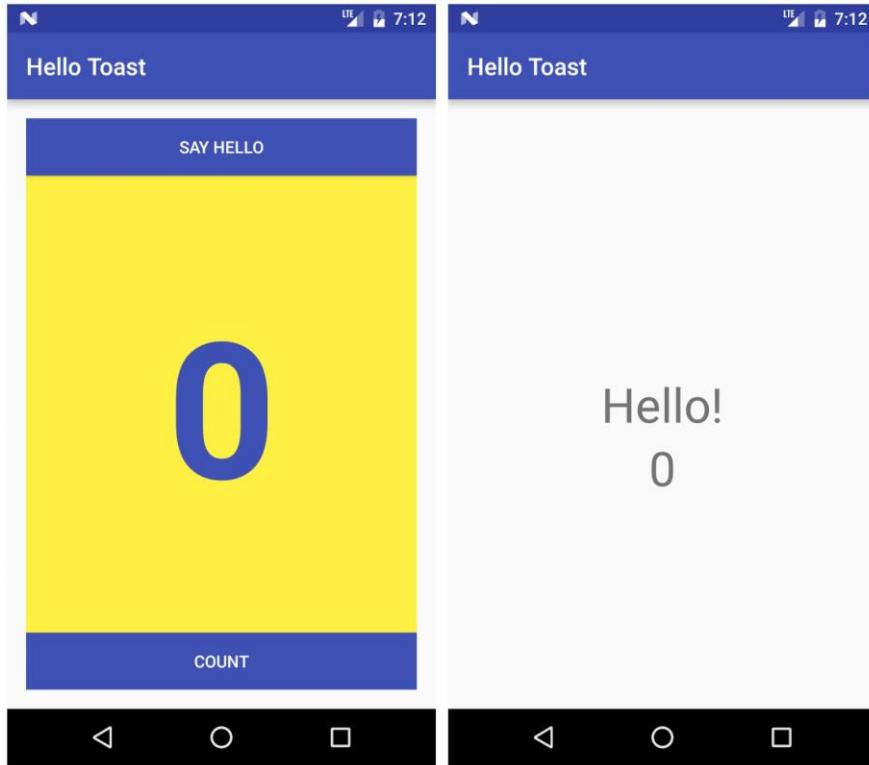
- [Nguyên tắc cơ bản về ứng dụng](#)
- [Hoạt động](#)
- [Ý định và bộ lọc ý định](#)
- [Thiết kế điều hướng Quay lại và Lên](#)
- [Hoạt động](#)
- [Kiên quyết](#)
- [Chế độ xem cuộn\(ScrollView\)](#)
- [Cánh](#)
- [Nút • TextView](#)
- [Tài nguyên chuỗi](#)

Homework

Xây dựng và chạy ứng dụng

Mở ứng dụng [HelloToast](#) mà bạn đã tạo trong lớp học lập trình thực tế trước đó.

1. Sửa đổi nút **T oast** để nó khởi chạy một chữ A mới để hiển thị từ "Xin chào!" và số lượng hiện tại, như hình dưới đây.
2. Thay đổi văn bản trên nút T oast thành **S ay Hello**.



Trả lời những câu hỏi này

Câu hỏi 1

Những thay đổi nào được thực hiện khi bạn thêm Activity thứ hai vào ứng dụng của mình bằng cách chọn **File > New >**

Hoạt động và mẫu Activity ? Chọn một:

- Loại thứ hai được thêm vào dưới dạng lớp Java. Bạn vẫn cần thêm tệp bố cục XML.
- Tệp bố cục XML thứ hai được tạo và thêm một lớp Java. Bạn vẫn cần xác định chữ ký lớp.
- Tệp Activity thứ hai được thêm vào dưới dạng lớp Java, tệp bố cục XML được tạo và tệp AndroidManifest.xml được thay đổi để khai báo tệp Activity thứ hai.

- Tập bộ cục XML Activity thứ hai được tạo và tệp AndroidManifest.xml được thay đổi để khai báo Activity thứ hai.

Câu hỏi 2

Điều gì xảy ra nếu bạn xóa các phần tử `android:parentActivityName` và `meta-data < ... >` khỏi khai báo Activity thứ hai trong tệp `AndroidManifest.xml`? Chọn một:

- Đầu Activity thứ hai không còn xuất hiện khi bạn cố gắng bắt đầu nó với một biểu tượng rõ ràng Ý định.
- Tập bộ cục XML Activity thứ hai bị xóa.
- Nút Quay lại không còn hoạt động trong Activity Activity thứ hai để đưa người dùng trở lại Activity chính.
- Nút Lên trên thanh ứng dụng không còn xuất hiện trong biểu tượng Activity thứ hai để đưa người dùng trở lại biểu tượng Activity gốc.

Câu hỏi 3

Bạn sử dụng phương thức constructor nào để tạo ra một Intent rõ ràng mới? Chọn một:

- ý định mới ()
- new Intent (Context context, Class<?> class)
- ý định mới (Hành động chuỗi, uri uri)
- ý định mới (Hành động chuỗi)

Câu hỏi 4

Trong bài tập về nhà của ứng dụng `HelloToast`, làm thế nào để bạn thêm giá trị hiện tại của số lượng vào Intent? Chọn một:

- Như dữ liệu Intent
- Như Intent T EXT_REQUEST
- Như một hành động Intent

- Như một phần bổ sung của tôi

Câu hỏi 5

Trong bài tập về nhà của ứng dụng HelloToast, làm thế nào để bạn hiển thị số lượng hiện tại trong "Xin chào" thứ hai Hoạt động? Chọn một:

- Nhận Intent mà Activity đã được khởi chạy.
- Lấy giá trị đếm hiện tại từ Intent.
- Cập nhật TextView cho số lượng.
- Tất cả những điều trên.

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

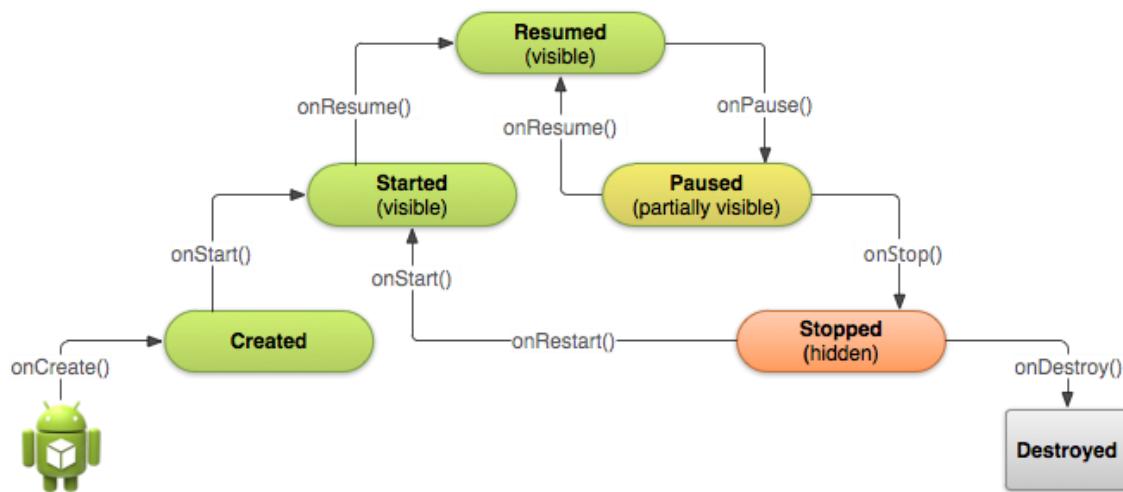
Kiểm tra xem ứng dụng có các tính năng sau không:

- Nó hiển thị nút **S ay H ello** thay vì nút **T oast**.
- Lần thứ hai bắt đầu khi nhấn nút **S ay H ello** và nó hiển thị thông báo "Xin chào!" và số lượng hiện tại từ MainActivity.
- Các tệp bô cục Java và XML thứ hai đã được thêm vào dự án.
- Tệp bô cục XML cho Activity thứ hai chứa hai phần tử TextView, một với chuỗi "Hello!" và phần tử thứ hai với count.
- Nó bao gồm một triển khai của một phương thức xử lý nhập chuột cho **nút S ay Hello** (trong Hoạt động chính).
- Nó bao gồm một triển khai của phương thức onCreate() cho Activity A thứ hai và cập nhật số lượng TextView với count từ MainActivity.

Bài 2.2: Vòng đời và trạng thái hoạt động

Giới thiệu

Trong thực tế này, bạn tìm hiểu thêm về *vòng đời activity*. Vòng đời là tập hợp các trạng thái mà một hoạt động có thể ở trong toàn bộ vòng đời của nó, từ khi nó được tạo ra đến khi nó bị phá hủy và hệ thống lấy lại tài nguyên của nó. Khi người dùng di chuyển giữa các hoạt động trong ứng dụng của bạn (cũng như vào và ra khỏi ứng dụng của bạn), các hoạt động sẽ chuyển đổi giữa các trạng thái khác nhau trong vòng đời của chúng.



Mỗi giai đoạn trong vòng đời của hoạt động có một phương thức gọi lại tương ứng: `onCreate()`, `onStart()`, `onPause()`, v.v. Khi một hoạt động thay đổi trạng thái, phương thức gọi lại được liên kết sẽ được gọi. Bạn đã thấy một trong các phương thức sau: `onCreate()`. Bằng cách ghi đè bất kỳ phương thức gọi lại vòng đời nào trong các lớp Activity, bạn có thể thay đổi hành vi mặc định của hoạt động để phản hồi các hành động của người dùng hoặc hệ thống.

Trạng thái hoạt động cũng có thể thay đổi để đáp ứng các thay đổi về cấu hình thiết bị, ví dụ: khi người dùng xoay thiết bị từ dọc sang ngang. Khi những thay đổi cấu hình này xảy ra, hoạt động sẽ bị hủy và tạo lại ở trạng thái mặc định và người dùng có thể mất thông tin mà họ đã nhập vào hoạt động. Để tránh gây nhầm lẫn cho người dùng, điều quan trọng là bạn phải phát triển ứng dụng để tránh mất dữ liệu không mong muốn. Sau đó trong thực tế này, bạn sẽ thử nghiệm các thay đổi về cấu hình và tìm hiểu cách duy trì trạng thái của hoạt động để đáp ứng các thay đổi về cấu hình thiết bị và các sự kiện vòng đời hoạt động khác.

Trong thực tế này, bạn thêm các câu lệnh ghi nhật ký vào ứng dụng TwoActivities và quan sát các thay đổi về vòng đời hoạt động khi bạn sử dụng ứng dụng. Sau đó, bạn bắt đầu làm việc với những thay đổi này và khám phá cách xử lý đầu vào của người dùng trong những điều kiện này.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy một dự án ứng dụng trong Android Studio.
- Thêm câu lệnh nhật ký vào ứng dụng của bạn và xem các nhật ký đó trong **ngăn Logcat**.
- Hiểu và làm việc với Activity và Intent, đồng thời thoải mái tương tác với họ.

Những gì bạn sẽ học

- Vòng đời Activity hoạt động như thế nào.
- Khi một Activity A bắt đầu, tạm dừng, dừng lại và bị phá hủy.
- Giới thiệu về các phương thức gọi lại vòng đời được liên kết với Activity thay đổi.
- Ảnh hưởng của các hành động (chẳng hạn như thay đổi cấu hình) có thể dẫn đến các sự kiện vòng đời Activity.
- Cách giữ lại trạng thái Activity trên các sự kiện vòng đời.

Bạn sẽ làm gì

- Thêm mã vào ứng dụng TwoActivities từ thực tế trước đó để triển khai các lệnh gọi lại vòng đời Hoạt động khác nhau để bao gồm các câu lệnh ghi nhật ký.
- Quan sát các thay đổi trạng thái khi ứng dụng của bạn chạy và khi bạn tương tác với từng trạng thái A trong ứng dụng của mình.
- Sửa đổi ứng dụng của bạn để giữ lại trạng thái phiên bản của Activity được tạo lại bất ngờ để phản hồi hành vi của người dùng hoặc thay đổi cấu hình trên thiết bị.

Tổng quan về ứng dụng

Trong thực tế này, bạn thêm vào ứng dụng [TwoActivities](#). Ứng dụng có giao diện và hoạt động gần giống như trong lớp học lập trình trước. Nó chứa hai Activity và cung cấp cho người dùng khả năng gửi giữa chúng. Những thay đổi bạn thực hiện đối với ứng dụng trong thực tế này sẽ không ảnh hưởng đến hành vi hiển thị của người dùng.

Nhiệm vụ 1: Thêm callback vòng đời vào TwoActivities

Trong nhiệm vụ này, bạn sẽ triển khai tất cả các phương thức gọi lại vòng đời Activity để in thông báo vào logcat khi các phương thức đó được gọi. Các thông báo nhật ký này sẽ cho phép bạn xem khi nào trạng thái Vòng đời Hoạt động thay đổi và những thay đổi trạng thái vòng đời đó ảnh hưởng như thế nào đến ứng dụng của bạn khi ứng dụng chạy.

1.1 (Tùy chọn) Sao chép dự án TwoActivities

Đối với các nhiệm vụ trong thực tế này, bạn sẽ sửa đổi dự án [TwoActivities](#) hiện có mà bạn đã xây dựng trong thực tế trước. Nếu bạn muốn giữ nguyên dự án TwoActivities trước đó, hãy làm theo các bước trong [Phụ lục: Tiên ích](#) để tạo bản sao của dự án.

1.2 Triển khai callback vào MainActivity

1. Mở dự án TwoActivities trong Android Studio và mở **MainActivity** trong **MainActivity** trong **ngắn Project > Android**.
2. Trong phương thức `onCreate()`, thêm các câu lệnh nhật ký sau:

```
Log.d(LOG_TAG, "-----");
Log.d(LOG_TAG, "onCreate");
```

3. Thêm một phần

ide cho lệnh gọi lại `onStart()`, với một câu lệnh đến nhật ký cho sự kiện đó:

```
@Override trống công khai
onStart(){
    super.onStart();
    Log.d(LOG_TAG, "onStart");
}
```

Đối với lỗi tắt, hãy chọn **Code > Override Methods** trong Android Studio. Một hộp thoại xuất hiện với tất cả các phương thức có thể có mà bạn có thể ghi đè trong lớp của mình. Chọn một hoặc nhiều phương thức gọi lại từ danh sách sẽ chèn một mẫu hoàn chỉnh cho các phương thức đó, bao gồm cả lệnh gọi bắt buộc đến siêu lớp.

4. Sử dụng phương thức `onStart()` làm mẫu để triển khai các lệnh gọi lại vòng đời `onPause()`, `onRestart()`, `onResume()`, `onStop()` và `onDestroy()`.

Tất cả các phương thức gọi lại đều có chữ ký giống nhau (ngoại trừ tên). Nếu bạn **COPY** và **DÁN** `onStart()` để tạo các phương thức gọi lại khác, đừng quên cập nhật nội dung để gọi đúng phương thức trong superclass và ghi lại phương thức chính xác.

1. Chạy ứng dụng của bạn.
2. Nhấp vào tab **Logcat** ở cuối Android Studio để hiển thị ngăn **Logcat**. Bạn sẽ thấy ba thông báo nhật ký hiển thị ba trạng thái vòng đời mà Activity đã chuyển qua khi nó bắt đầu:

```
D/Hoạt động chính: -----  
D/MainActivity: onCreate  
D/MainActivity: onStart  
D/Hoạt động chính: onResume
```

1.3 Triển khai callback vòng đời trong SecondActivity

Bây giờ bạn đã triển khai các phương thức gọi lại vòng đời cho MainActivity, hãy làm tương tự cho Hoạt động thứ hai.

1. Mở **SecondActivity**.
2. Ở đầu lớp, thêm một hằng số cho biến `LOG_TAG`:

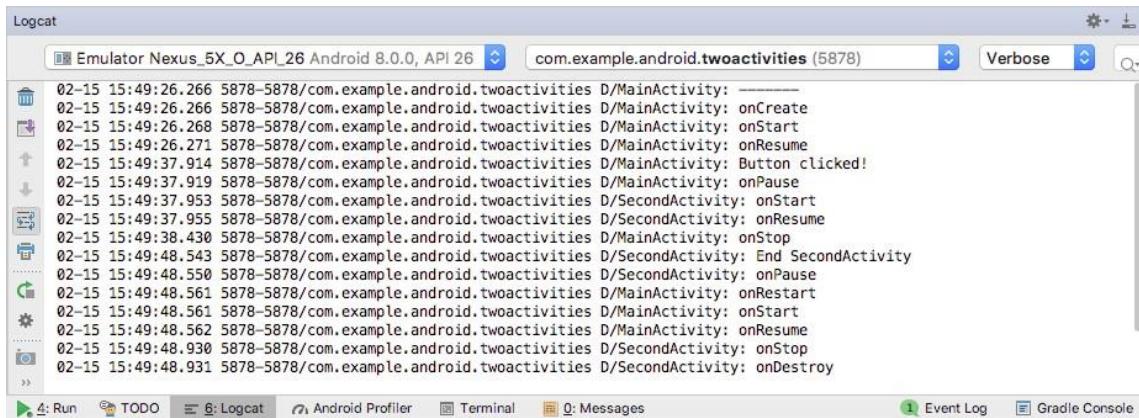
```
Private static final String LOG_TAG = SecondActivity.class.getSimpleName();
```

3. Thêm lệnh gọi lại vòng đời và câu lệnh nhật ký vào lệnh A thứ hai. (Bạn có thể **C opy** và **Paste** các phương thức callback từ `MainActivity`.)
4. Thêm một câu lệnh log vào phương thức `returnReply()` ngay trước phương thức `finish()`:

```
Log.d(LOG_TAG, "Kết thúc hoạt động thứ hai");
```

1.4 Quan sát nhật ký khi ứng dụng chạy

1. Chạy ứng dụng của bạn.
2. Nhấp vào tab **Logcat** ở cuối Android Studio để hiển thị ngăn **Logcat**.
3. Nhập **Activity** vào hộp tìm kiếm.
Logcat Android có thể rất dài và lộn xộn. Vì biến `LOG_TAG` trong mỗi lớp chứa các từ `MainActivity` hoặc `SecondActivity`, từ khóa này cho phép bạn lọc nhật ký chỉ cho những thứ bạn quan tâm.



Thử nghiệm sử dụng ứng dụng của bạn và lưu ý rằng các sự kiện vòng đời xảy ra để phản hồi với các hành động khác nhau. Đặc biệt, hãy thử những điều sau:

- Sử dụng ứng dụng bình thường (gửi tin nhắn, trả lời bằng tin nhắn khác).
- Sử dụng nút Quay lại để quay lại từ ctivity A thứ hai sang ctivity A chính.

- Sử dụng mũi tên Lên trong thanh ứng dụng để quay lại từ ctivity A thứ hai sang ctivity A chính.
- Xoay thiết bị trên cả kích thước A chính và thứ hai vào các thời điểm khác nhau trong ứng dụng của bạn và quan sát những gì xảy ra trong nhật ký và trên màn hình.
- Nhấn nút tổng quan (nút vuông ở bên phải Trang chủ) và đóng ứng dụng (nhấn vào dấu X).
- Quay lại màn hình chính và khởi động lại ứng dụng của bạn.

MẸO: Nếu đang chạy ứng dụng của mình trong trình mô phỏng, bạn có thể mô phỏng quá trình xoay bằng Control+F11 hoặc Control+Function+F11.

Mã giải pháp nhiệm vụ 1

Các đoạn mã sau đây hiển thị mã giải pháp cho tác vụ đầu tiên.

Hoạt động chính

Các đoạn mã sau đây hiển thị mã được thêm vào MainActivity, nhưng không phải toàn bộ lớp.

Phương thức onCreate():

```
@Override được bảo vệ void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Ghi lại thời điểm bắt đầu của phương thức onCreate().  
    Log.d(LOG_TAG, "-----");  
    Log.d(LOG_TAG, "onCreate");
```

```
Khởi tạo tất cả các biến chế độ xem.  
mMessageEditText = findViewById(R.id.editText_main);  
mReplyHeaderView = findViewById(R.id.text_header_reply);  
mReplyTextView = findViewById(R.id.text_message_reply); }
```

Các phương pháp vòng đời khác:

```
@Override được bảo vệ void  
onStart() {  
    super.onStart();  
    Log.d(LOG_TAG, "onStart");  
}  
  
@Override được bảo vệ khoảng  
trống onPause() {  
    super.onPause();  
    Log.d(LOG_TAG, "onPause");  
}  
  
@Override được bảo vệ void  
onRestart() {  
    super.onRestart();  
    Log.d(LOG_TAG, "onRestart"); }  
  
@Override được bảo vệ void  
onResume() {  
    super.onResume();  
    Log.d(LOG_TAG, "onResume"); }  
  
@Override được bảo vệ khoảng  
trống onStop() {  
    super.onStop();  
    Log.d(LOG_TAG, "đã ngắt");  
}  
  
@Override được bảo vệ khoảng trống  
onDestroy() {  
  
    super.onDestroy();  
    Log.d(LOG_TAG, "onDestroy");  
}
```

Hoạt động thứ hai

Các đoạn mã sau đây hiển thị mã được thêm vào SecondActivity, nhưng không phải toàn bộ lớp.

Ở đâu lớp SecondActivity :

```
Private static final String LOG_TAG = SecondActivity.class.getSimpleName();
```

Phương thức returnReply():

```
public void returnReply(Xem chế độ xem) {
    Trả lời chuỗi = mReply.getText().toString();
    Ý định replyIntent = ý định mới ();
    replyIntent.putExtra(EXTRA_REPLY, trả lời);
    setResult(RESULT_OK, trả lờiIntent);
    Log.d(LOG_TAG, "Kết thúc hoạt động thứ hai");
    kết thúc();
}
```

Các phương pháp vòng đời khác:

Tương tự như đối với MainActivity, ở trên.

Nhiệm vụ 2: Lưu và khôi phục trạng thái phiên bản Activity

Tùy thuộc vào tài nguyên hệ thống và hành vi của người dùng, mỗi hoạt động A trong ứng dụng của bạn có thể bị phá hủy và xây dựng lại thường xuyên hơn nhiều so với bạn nghĩ.

Bạn có thể nhận thấy hành vi này trong phần trước khi xoay thiết bị hoặc trình giả lập. Xoay thiết bị là một ví dụ về sự *thay đổi cấu hình* của thiết bị. Mặc dù xoay là phổ biến nhất, nhưng tất cả các thay đổi cấu hình đều dẫn đến việc Activity hiện tại bị phá hủy và tái tạo như thế nào mới. Nếu bạn không tính đến hành vi này trong mã của mình, khi xảy ra thay đổi cấu hình, bộ cục Activity của bạn có thể trở lại giao diện mặc định và giá trị ban đầu, đồng thời người dùng có thể mất vị trí, dữ liệu hoặc trạng thái tiến trình trong ứng dụng của bạn.

Trạng thái của mỗi Activity được lưu trữ dưới dạng một tập hợp các cặp khóa/giá trị trong một đối tượng `Bundle` được gọi là trạng thái thực thể *Hoạt động*. Hệ thống lưu thông tin trạng thái mặc định vào trạng thái phiên bản B ngay trước khi Activity A bị dừng và chuyển thông tin B đó đến phiên bản A Activity mới để khôi phục.

Để tránh bị mất dữ liệu trong Activity khi nó bị phá hủy và tạo lại bất ngờ, bạn cần triển khai phương thức `onSaveInstanceState()`. Hệ thống gọi phương thức này trên Activity của bạn (giữa `onPause()` và `onStop()`) khi có khả năng Activity có thể bị phá hủy và tạo lại.

Dữ liệu bạn lưu trong trạng thái phiên bản chỉ dành riêng cho phiên bản này của Activity cụ thể này trong phiên ứng dụng hiện tại. Khi bạn dừng và khởi động lại phiên ứng dụng mới, trạng thái phiên bản A Activity sẽ mất và A Activity sẽ trở lại giao diện mặc định. Nếu bạn cần lưu dữ liệu người dùng giữa các phiên ứng dụng, hãy sử dụng tùy chọn được chia sẻ hoặc cơ sở dữ liệu. Bạn tìm hiểu về cả hai điều này trong một thực tế sau.

2.1 Lưu trạng thái phiên bản Activity bằng `onSaveInstanceState()`

Bạn có thể nhận thấy rằng việc xoay thiết bị hoàn toàn không ảnh hưởng đến trạng thái của Activity A thứ hai. Điều này là do bộ cục và trạng thái A Activity thứ hai được tạo ra từ bộ cục và Ý định đã kích hoạt nó. Ngay cả khi A Activity được tạo lại, Intent vẫn ở đó và dữ liệu trong Intent đó vẫn được sử dụng mỗi khi phương thức `onCreate()` trong Activity A thứ hai được gọi.

Ngoài ra, bạn có thể nhận thấy rằng trong mỗi đoạn A, bất kỳ văn bản nào bạn đã nhập vào tin nhắn hoặc trả lời các phần tử `EditText` đều được giữ lại ngay cả khi thiết bị được xoay. Điều này là do thông tin trạng thái của một số phần tử `View` trong bộ cục của bạn được tự động lưu sau các thay đổi cấu hình và giá trị hiện tại của `EditText` là một trong những trường hợp đó.

Vì vậy, trạng thái A Activity duy nhất mà bạn quan tâm là các phần tử `TextView` cho tiêu đề trả lời và văn bản trả lời trong A Activity chính. Cả hai phần tử `TextView` đều vô hình theo mặc định; chúng chỉ xuất hiện khi bạn gửi một tin nhắn trả lại A Activity chính từ Activity A thứ hai.

Trong tác vụ này, bạn thêm mã để giữ nguyên trạng thái phiên bản của hai phần tử TextView này bằng cách sử dụng onSaveInstanceState().

1. Mở **MainActivity**.
2. Thêm cách triển khai khung này của onSaveInstanceState() vào mục A hoặc sử dụng **Code > Override Methods** để chèn ghi đè khung xương.

```
@Override  
public void onSaveInstanceState(Bundle outState) {  
    super.onSaveInstanceState(outState);}
```

3. Kiểm tra xem tiêu đề hiện có hiển thị hay không, và nếu có thì đặt trạng thái hiển thị đó vào trạng thái Bundle bằng phương thức `putBoolean()` và khóa "reply_visible".

```
if (mReplyHeadTextView.getVisibility() == View.VISIBLE) {  
    outState.putBoolean("reply_visible", đúng);}
```

Hãy nhớ rằng tiêu đề và văn bản trả lời được đánh dấu là vô hình cho đến khi có câu trả lời từ lần thứ hai. Nếu tiêu đề hiển thị, thì có dữ liệu trả lời cần được lưu. Lưu ý rằng chúng ta chỉ quan tâm đến trạng thái hiển thị đó - văn bản thực tế của tiêu đề không cần phải được lưu, vì văn bản đó không bao giờ thay đổi.

4. Bên trong cùng một kiểm tra đó, thêm văn bản trả lời vào bảng B.
`outState.putString("reply_text", mReplyTextView.getText().toString());`

Nếu tiêu đề hiển thị, bạn có thể giả định rằng bản thân thư trả lời cũng hiển thị. Bạn không cần kiểm tra hoặc lưu trạng thái hiển thị hiện tại của thư trả lời. Chỉ có văn bản thực tế của tin nhắn đi vào trạng thái B với phím "reply_text".

Bạn chỉ lưu trạng thái của những phần tử View có thể thay đổi sau khi Activity được tạo. Các phần tử View khác trong ứng dụng của bạn (EditText, Button) có thể được tạo lại từ bộ cục mặc định bất cứ lúc nào.

Lưu ý rằng hệ thống sẽ lưu trạng thái của một số phần tử View, chẳng hạn như nội dung của EditText.

2.2 Khôi phục trạng thái phiên bản Activity trong onCreate()

Khi bạn đã lưu trạng thái phiên bản Activity, bạn cũng cần khôi phục nó khi Activity được tạo lại. Bạn có thể thực hiện việc này trong `onCreate()` hoặc bằng cách triển khai lệnh gọi lại `onRestoreInstanceState()`, được gọi sau `onStart()` sau khi Activity được tạo.

Hầu hết thời gian, nơi tốt hơn để khôi phục trạng thái Activity là trong `onCreate()`, để đảm bảo rằng giao diện người dùng, bao gồm cả trạng thái, có sẵn càng sớm càng tốt. Đôi khi thuận tiện khi thực hiện điều đó trong `onRestoreInstanceState()` sau khi tất cả các quá trình khởi tạo đã được thực hiện hoặc cho phép các lớp con quyết định có sử dụng triển khai mặc định của bạn hay không.

- Trong phương thức `onCreate()`, sau khi các biến View được khởi tạo bằng `findViewById()`, hãy thêm một thử nghiệm để đảm bảo rằng `savedInstanceState` không phải là null.

```
Khởi tạo tất cả các biến chế độ xem.  
mMessageEditText = findViewById(R.id.editText_main); mReplyHeadTextView =  
findViewById(R.id.text_header_reply); mReplyTextView =  
findViewById(R.id.text_message_reply);  
Khôi phục trạng thái.  
if (savedInstanceState != null) { }
```

Khi Activity của bạn được tạo, hệ thống sẽ chuyển trạng thái Bundle đến `onCreate()` làm đối số duy nhất của nó. Lần đầu tiên `onCreate()` được gọi và ứng dụng của bạn khởi động, Bundle là null— không có trạng thái hiện có trong lần đầu tiên ứng dụng của bạn khởi động. Các lệnh gọi tiếp theo đến `onCreate()` có một gói được điền vào dữ liệu bạn đã lưu trữ trong `onSaveInstanceState()`.

- Bên trong kiểm tra đó, lấy khả năng hiển thị hiện tại (đúng hoặc sai) ra khỏi Bundle bằng phím "reply_visible".

```
if (savedInstanceState != null) {  
    boolean isVisible =
```

```
|    savedInstanceState.getBoolean("reply_visible"); }
```

3. Thêm một thử nghiệm bên dưới dòng trước đó cho biến isVisible.

```
|if (isVisible) {  
|}
```

Nếu có một r eply_visible khóa trong trạng thái B undle (và i sVisible do đó là t rue), bạn sẽ cần khôi phục trạng thái.

4. Bên trong kiểm tra i sVisible, làm cho tiêu đề hiển thị.

```
|mReplyHeadTextView.setVisibility(View.VISIBLE);
```

5. Nhận tin nhắn trả lời bằng văn bản từ B undle bằng phím "reply_text" và đặt TextView trả lời để hiển thị chuỗi đó.

```
mReplyTextView.setText(savedInstanceState.getString("reply_text"));
```

6. Làm cho câu trả lời TextView hiển thị:

```
mReplyTextView.setVisibility(View.VISIBLE);
```

7. Chạy ứng dụng. Hãy thử xoay thiết bị hoặc trình mô phỏng để đảm bảo rằng thông báo trả lời (nếu có) vẫn còn trên màn hình sau khi Activity được tạo lại.

Mã giải pháp nhiệm vụ 2

Các đoạn mã sau đây hiển thị mã giải pháp cho tác vụ này.

Hoạt động chính

Các đoạn mã sau đây hiển thị mã được thêm vào MainActivity, nhưng không phải toàn bộ lớp.

Phương thức onSaveInstanceState():

```
@Override  
public void onSaveInstanceState(Bundle outState) {  
    super.onSaveInstanceState(outState);  
    Nếu tiêu đề hiển thị, tin nhắn cần được lưu.  
    Nếu không, chúng ta vẫn đang sử dụng bố cục mặc định.  
    if (mReplyHeadTextView.getVisibility() == View.VISIBLE) {  
        outState.putBoolean("reply_visible", đúng);  
        outState.putString("reply_text",  
            mReplyTextView.getText().toString());  
    }  
}
```

}

Phương thức `onCreate()`:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d(LOG_TAG, "");
    Log.d(LOG_TAG, "onCreate");

    Khởi tạo tất cả các biến chế độ xem.
    mMessageEditText = findViewById(R.id.editText_main);
    mReplyHeadTextView = findViewById(R.id.text_header_reply);
    mReplyTextView = findViewById(R.id.text_message_reply);
    Khôi phục trạng thái đã lưu.
    Xem onSaveInstanceState() để biết những gì được lưu.
    if (savedInstanceState != null) {
        boolean isVisible =
            savedInstanceState.getBoolean("reply_visible");
        Hiển thị cả chế độ xem tiêu đề và thư. Nếu isVisible là // false hoặc thiếu trong gói, hãy sử dụng bộ
        cục mặc định.     nếu (isVisible) {
            mReplyHeadTextView.setVisibility(View.VISIBLE);
            mReplyTextView.setText(savedInstanceState
                .getString("reply_text"));
            mReplyTextView.setVisibility(View.VISIBLE);
        }
    }
}
```

Dự án hoàn chỉnh:

Dự án Android Studio: [TwoActivitiesLifecycle](#)

Trang 213

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 này.

Thử thách: Tạo một ứng dụng danh sách mua sắm đơn giản với một hoạt động chính cho danh sách mà người dùng đang xây dựng và một hoạt động thứ hai cho danh sách các mặt hàng mua sắm phổ biến.

- Hoạt động chính phải chứa danh sách cần xây dựng, danh sách này phải được tạo thành từ mười phần tử TextView trống.
- Nút **Mục A dd** trên hoạt động chính khởi chạy hoạt động thứ hai chứa danh sách các mặt hàng mua sắm phổ biến (**C heese, R ice, A pples**, v.v.). Sử dụng các phần tử Button để hiển thị các mục.
- Chọn một mục sẽ đưa người dùng trở lại hoạt động chính và cập nhật TextView trống để bao gồm mục đã chọn.

Sử dụng Intent để chuyển thông tin từ một ctivity A sang ctivity khác. Đảm bảo rằng trạng thái hiện tại của danh sách mua sắm được lưu khi người dùng xoay thiết bị.

Tóm tắt

- Vòng đời Hoạt động là một tập hợp các trạng thái mà Activity di chuyển qua, bắt đầu khi nó được tạo lần đầu tiên và kết thúc khi hệ thống Android thu hồi tài nguyên cho Activity đó.

- Khi người dùng di chuyển từ một ctivity A này sang một ctivity khác và bên trong và bên ngoài ứng dụng của bạn, mỗi ctivity A di chuyển giữa các trạng thái trong vòng đời A ctivity.
- Mỗi trạng thái trong vòng đời A ctivity có một phương thức gọi lại tương ứng mà bạn có thể ghi đè trong lớp A ctivity của mình.
- Các phương thức vòng đời là o nCreate(), o nStart(), o nPause(), o nRestart(), o nResume(), onStop(), o nDestroy().
- Ghi đè phương thức gọi lại vòng đời cho phép bạn thêm hành vi xảy ra khi Activity của bạn chuyển sang trạng thái đó.
- Bạn có thể thêm các phương thức ghi đè khung xương vào các lớp của mình trong Android Studio bằng C **ode > Ghi đè**.

- Những thay đổi cấu hình thiết bị như xoay dẫn đến việc Activity bị phá hủy và tái tạo như thế nào là mới.
- Một phần của trạng thái Activity được giữ nguyên khi thay đổi cấu hình, bao gồm các giá trị hiện tại của các phần tử EditText. Đối với tất cả các dữ liệu khác, bạn phải tự lưu dữ liệu đó một cách rõ ràng.
- Lưu trạng thái phiên bản Activity trong phương thức onSaveInstanceState().
- Dữ liệu trạng thái phiên bản được lưu trữ dưới dạng các cặp khóa/giá trị đơn giản trong bảng B. Sử dụng các phương pháp Bundle để đưa dữ liệu vào và lấy dữ liệu trở lại từ Bundle.
- Khôi phục trạng thái phiên bản trong onCreate(), đây là cách ưu tiên hoặc onRestoreInstanceState().

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [2.2: Vòng đời và trạng thái hoạt động](#).

Tìm hiểu thêm

Tài liệu Android Studio:

- [Làm quen với Android Studio](#)

Tài liệu dành cho nhà phát triển Android:

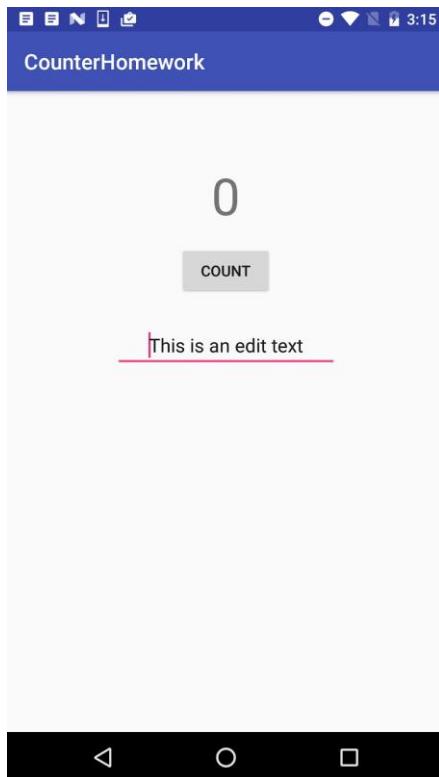
- [Nguyên tắc cơ bản về ứng dụng](#)
- [Hoạt động](#)
- [Hiểu vòng đời hoạt động](#)
- [Ý định và bộ lọc ý định](#)
- [Xử lý các thay đổi cấu hình](#)
- [Hoạt động](#)
- [Kiên quyết](#)

Homework

Xây dựng và chạy ứng dụng

1. Tạo một ứng dụng có bố cục chứa bộ đếm TextView, Button để tăng bộ đếm và EditText. Xem ảnh chụp màn hình bên dưới làm ví dụ. Bạn không cần phải sao chép chính xác bố cục.
2. Thêm trình xử lý nhấp chuột cho nút B để tăng bộ đếm.

3. Chạy ứng dụng và tăng bộ đếm. Nhập một số văn bản vào EditText.
4. Xoay thiết bị. Lưu ý rằng bộ đếm được đặt lại, nhưng EditText thì không.
5. Triển khai onSaveInstanceState() để lưu trạng thái hiện tại của ứng dụng.
6. Cập nhật onCreate() để khôi phục trạng thái của ứng dụng.
7. Đảm bảo rằng khi bạn xoay thiết bị, trạng thái ứng dụng được giữ nguyên.



Trả lời những câu hỏi này

Câu hỏi 1

Nếu bạn chạy ứng dụng bài tập về nhà trước khi triển khai onSaveInstanceState(), điều gì sẽ xảy ra nếu bạn xoay thiết bị?
Chọn một:

- EditText không còn chứa văn bản bạn đã nhập, nhưng bộ đếm được giữ nguyên.
- Bộ đếm được đặt lại về 0 và EditText không còn chứa văn bản bạn đã nhập.

- Bộ đếm được đặt lại về 0, nhưng nội dung của EditText được giữ nguyên.
- Bộ đếm và nội dung của EditText được giữ nguyên.

Câu hỏi 2

Các phương pháp vòng đời Activity được gọi là gì khi thay đổi cấu hình thiết bị (chẳng hạn như xoay vòng) xảy ra? Chọn một:

- Android ngay lập tức tắt Activity của bạn bằng cách gọi `onStop()`. Mã của bạn phải khởi động lại Activity.
- Android tắt Activity của bạn bằng cách gọi `onPause()`, `onStop()` và `onDestroy()`. Mã của bạn phải khởi động lại Activity.
- Android tắt Activity của bạn bằng cách gọi `onPause()`, `onStop()` và `onDestroy()`, sau đó khởi động lại, gọi `onCreate()`, `onStart()` và `onResume()`.
- Android ngay lập tức gọi `onResume()`.

Câu hỏi 3

Khi nào trong vòng đời Activity là `onSaveInstanceState()` được gọi? Chọn một:

- `onSaveInstanceState()` được gọi trước phương thức `onStop()`.
- `onSaveInstanceState()` được gọi trước phương thức `onResume()`.
- `onSaveInstanceState()` được gọi trước phương thức `onCreate()`.
- `onSaveInstanceState()` được gọi trước phương thức `onDestroy()`.

Câu hỏi 4

Phương pháp vòng đời Activity nào là tốt nhất để sử dụng để lưu trữ dữ liệu trước khi Activity được hoàn thành hoặc phá hủy? Chọn một:

- `onPause()` hoặc `onStop()`
- `onResume()` hoặc `onCreate()`
- `onDestroy()`

- onStart() hoặc onRestart()

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kiểm tra xem ứng dụng có các tính năng sau không:

- Nó hiển thị một bộ đếm, một Button để tăng bộ đếm đó và một EditText.
- Nhập vào nút B sẽ tăng bộ đếm lên 1.
- Khi thiết bị được xoay, cả trạng thái bộ đếm và EditText đều được giữ lại.
- Việc triển khai MainActivity.java sử dụng phương thức onSaveInstanceState() để lưu trữ giá trị bộ đếm.
- Việc thực hiện các bài kiểm tra onCreate() cho sự tồn tại của savedInstanceState Bundle. Nếu Bundle đó tồn tại, giá trị bộ đếm sẽ được khôi phục và lưu vào TextView.

Bài 2.3: Ý định ngầm

Giới thiệu

Trong phần trước, bạn đã tìm hiểu về ý định rõ ràng. Trong một ý định rõ ràng, bạn thực hiện một hoạt động trong ứng dụng của mình hoặc trong một ứng dụng khác bằng cách gửi một ý định có tên lớp đủ điều kiện của hoạt động. Trong phần này, bạn tìm hiểu thêm về ý định và cách sử dụng chúng để thực hiện các hoạt động.

Với ý định ngầm, bạn bắt đầu một hoạt động mà không biết ứng dụng hoặc hoạt động nào sẽ xử lý tác vụ. Ví dụ: nếu bạn muốn ứng dụng của mình chụp ảnh, gửi email hoặc hiển thị vị trí trên bản đồ, bạn thường không quan tâm ứng dụng hoặc hoạt động nào thực hiện tác vụ.

Ngược lại, hoạt động của bạn có thể khai báo một hoặc nhiều bộ lọc ý định trong tệp AndroidManifest.xml để quảng cáo rằng hoạt động có thể chấp nhận ý định ngầm và xác định các loại ý định mà hoạt động sẽ chấp nhận.

Để so khớp yêu cầu của bạn với một ứng dụng được cài đặt trên thiết bị, hệ thống Android sẽ khớp ý định ngầm của bạn với một hoạt động có bộ lọc ý định cho biết rằng họ có thể thực hiện hành động. Nếu nhiều ứng dụng khớp, người dùng sẽ thấy một công cụ chọn ứng dụng cho phép họ chọn ứng dụng mà họ muốn sử dụng để xử lý ý định.

Trong thực tế này, bạn xây dựng một ứng dụng gửi ý định ngầm để thực hiện từng tác vụ sau:

- Mở URL trong trình duyệt web.
- Mở một vị trí trên bản đồ.

- Chia sẻ văn bản.

Chia sẻ — gửi một phần thông tin cho người khác qua email hoặc phương tiện truyền thông xã hội — là một tính năng phổ biến trong nhiều ứng dụng. Đối với hành động chia sẻ, bạn sử dụng lớp `S hareCompat.IntentBuilder`, giúp bạn dễ dàng xây dựng ý định ngầm để chia sẻ dữ liệu.

Cuối cùng, bạn tạo một bộ nhận ý định đơn giản chấp nhận ý định ngầm cho một hành động cụ thể.

Những điều bạn nên biết

Bạn sẽ có thể:

- Sử dụng trình chỉnh sửa bố cục để sửa đổi bố cục.
- Chính sửa mã XML của bố cục.
- Thêm một nút B và một trình xử lý nhấp chuột.
- Tạo và sử dụng Activity.
- Tạo và gửi một Intent giữa một Activity A và một Activity khác.

Những gì bạn sẽ học

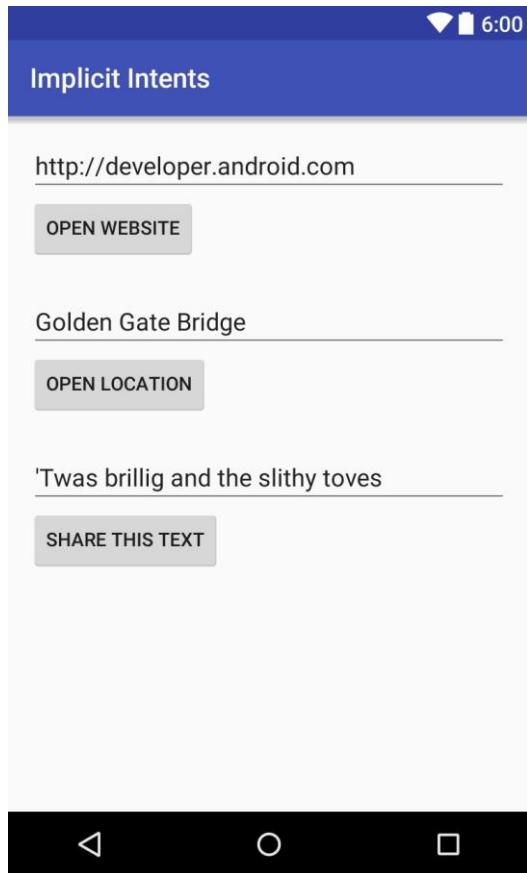
- Làm thế nào để tạo một Intent ngầm, và sử dụng các hành động và danh mục của nó.
- Cách sử dụng lớp trợ giúp `S hareCompat.IntentBuilder` để tạo một Intent ngầm để chia sẻ dữ liệu.
- Cách quảng cáo rằng ứng dụng của bạn có thể chấp nhận Intent ngầm bằng cách khai báo bộ lọc Intent trong tệp `A ndroidManifest.xml`.

Bạn sẽ làm gì

- Tạo một ứng dụng mới để thử nghiệm với Intent ngầm.
- Thực hiện một Intent ngầm để mở một trang web và một cái khác mở một vị trí trên bản đồ.
- Triển khai một hành động để chia sẻ một đoạn văn bản.
- Tạo một ứng dụng mới có thể chấp nhận một Intent ngầm để mở một trang web.

Tổng quan về ứng dụng

Trong phần này, bạn tạo một ứng dụng mới với một Activity và ba tùy chọn cho các hành động: mở trang web, mở một vị trí trên bản đồ và chia sẻ một đoạn văn bản. Tất cả các trường văn bản đều có thẻ chỉnh sửa (EditText), nhưng chứa các giá trị mặc định.



Nhiệm vụ 1: Tạo dự án và bô cục

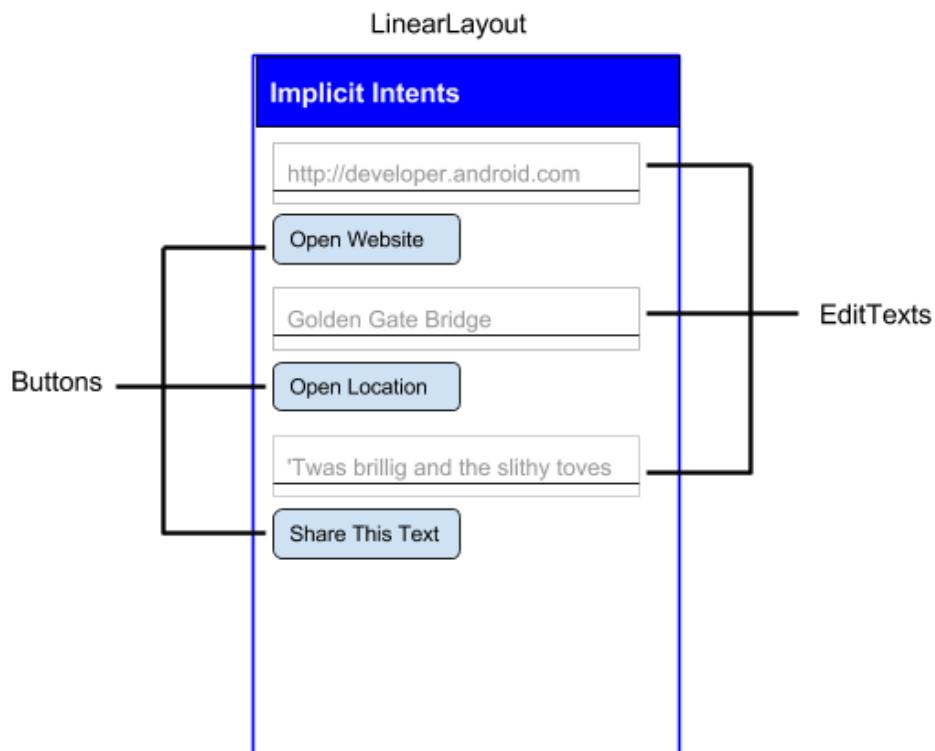
Đối với bài tập này, bạn tạo một dự án và ứng dụng mới có tên là Implicit Intents, với bô cục mới.

1.1 Tạo dự án

1. Khởi động Android Studio và tạo một dự án Android Studio mới. Đặt tên cho ứng dụng của bạn **tôi implicit Intents**.
2. Chọn **Empty Activity** cho mẫu dự án. Nhập vào **Name**.
3. Chấp nhận tên **Activity** mặc định (**MainActivity**). Đảm bảo hộp **Bộ lọc Generate file** được chọn. Nhập vào **Finish**.

1.2 Tạo bố cục

Trong tác vụ này, hãy tạo bố cục cho ứng dụng. Sử dụng **LinearLayout**, ba phần tử **Button** và ba phần tử **EditText**, như sau:



- Mở giá trị pp > res > strings.xml trong ngăn Project > Android và thêm các tài nguyên chuỗi sau:

```
<string name="edittext_uri">http://developer.android.com</chuỗi>
<string name="button_uri">Mở website</string>
<string name="edittext_loc">Cầu Cảng Vàng</string>
<string name="button_loc">Open Location</string>

<string name="edittext_share">\'Twas brillig and the slithy toves</string>
<string name="button_share">chia sẻ văn bản này</string>
```

- Mở res > bộ cục > activity_main.xml trong ngăn Project > Android. Nhấp vào tab Text để chuyển sang mã XML.
- Thay đổi android.support.constraint.ConstraintLayout thành LinearLayout, như bạn đã học trong thực tế trước đó.
- Thêm thuộc tính android:orientation với giá trị "vertical". Thêm thuộc tính android:padding có giá trị "16dp".

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/resauto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="dọc"
    android:padding="16dp"
    tools:context="com.example.android.implicitintents.MainActivity">
```

- Xóa TextView hiển thị "Hello World".
- Thêm một tập hợp các yếu tố giao diện người dùng vào bộ cục cho nút Trang web Open. Bạn cần một phần tử EditText và một phần tử Button. Sử dụng các giá trị thuộc tính sau:

Thuộc tính EditText	Giá trị
---------------------	---------

android:id	"@+id/website_edittext"
Android:layout_width	"match_parent"
Android:layout_height	"wrap_content"
android:văn bản	"@string/edittext_uri"
Thuộc tính nút	Giá trị
android:id	"@+id/open_website_button"
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"
Android:layout_marginBottom	"24dp"
android:văn bản	"@string/button_uri"
android:bậtNháp chuột	"trang web mở"

Giá trị cho thuộc tính android:onClick sẽ vẫn được gạch chân màu đỏ cho đến khi bạn xác định phương thức gọi lại trong một tác vụ tiếp theo.

- Thêm một tập hợp các phần tử giao diện người dùng (EditText và Button) vào bộ cục cho nút **Vị trí bút O**. Sử dụng các thuộc tính tương tự như trong bước trước, nhưng sửa đổi chúng như hình dưới đây. (Bạn có thể sao chép các giá trị từ **O bút Website** và sửa đổi chúng.)

Thuộc tính EditText	Giá trị
android:id	"@+id/location_edittext"
android:văn bản	"@string/edittext_loc"
Thuộc tính nút	Giá trị

android:id	"@+id/open_location_button"
android:văn bản	"@string/button_loc"
android:bậtNháp chuột	"mởVị trí"

Giá trị cho thuộc tính android:onClick sẽ vẫn được gạch chân màu đỏ cho đến khi bạn xác định phương thức gọi lại trong một tác vụ tiếp theo.

8. Thêm một tập hợp các phần tử giao diện người dùng (EditText và Button) vào bộ cục cho nút **Share This**. Sử dụng các thuộc tính được hiển thị bên dưới. (Bạn có thể sao chép các giá trị từ **O bút Website** và sửa đổi chúng.)

Thuộc tính EditText	Giá trị
android:id	"@+id/share_edittext"
android:văn bản	"@string/edittext_share"
Thuộc tính nút	Giá trị
android:id	"@+id/share_text_button"
android:văn bản	"@string/button_share"
android:bậtNháp chuột	"chia sẻ văn bản"

Tùy thuộc vào phiên bản Android Studio của bạn, mã activity_main.xml của bạn sẽ trông giống như sau. Các giá trị cho các thuộc tính android:onClick sẽ vẫn được gạch chân màu đỏ cho đến khi bạn xác định các phương thức gọi lại trong một tác vụ tiếp theo.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/resauto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="dọc"
    android:padding="16dp"
    tools:context="com.example.android.implicitintents.MainActivity">

    <EditText
        android:id="@+id/website_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/edittext_uri"/>
    < Nút
        android:id="@+id/open_website_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        android:text="@string/button_uri"
        android:onClick="openWebsite"/>
    <EditText
        android:id="@+id/location_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/edittext_uri"/>
    < Nút
        android:id="@+id/open_location_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        android:text="@string/button_loc" android:onClick="openLocation"/>
    <EditText
        android:id="@+id/share_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/edittext_share"/>
```

```
< Nút
    android:id="@+id/share_text_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="@string/button_share"
    android:onClick="shareText"/>

</LinearLayout>
```

Nhiệm vụ 2: Triển khai nút Mở trang web

Trong tác vụ này, bạn triển khai phương pháp xử lý khi nhấp chuột cho nút đầu tiên trong bố cục, **O pen**

Trang mạng. Hành động này sử dụng Intent ngầm để gửi URI đã cho đến một Activity A có thể xử lý Intent ngầm đó (chẳng hạn như trình duyệt web).

2.1 Định nghĩa openWebsite()

1. Nhập vào "openWebsite" trong mã XML activity_main.xml.
2. Nhấn Alt + Enter (Option + Enter trên máy Mac) và chọn **Create 'openWebsite (View)' in 'MainActivity'**.

Tệp MainActivity sẽ mở ra và Android Studio tạo một phương thức khung cho trình xử lý openWebsite().

```
public void openWebsite(Xem chế độ xem) {  
}
```

3. Trong MainActivity, thêm một biến riêng ở đầu lớp để giữ đối tượng EditText cho URI trang web.

```
EditText riêng mWebsiteEditText;
```

4. Trong phương thức onCreate() cho MainActivity, hãy sử dụng findViewById() để lấy tham chiếu đến thực thể EditText và gán nó cho biến riêng đó:

```
mWebsiteEditText = findViewById(R.id.website_edittext);
```

2.2 Thêm mã vào openWebsite()

1. Thêm một câu lệnh vào phương thức openWebsite() mới lấy giá trị chuỗi của EditText:

```
Url chuỗi = mWebsiteEditText.getText().toString();
```

2. Mã hóa và phân tích cú pháp chuỗi đó thành một đối tượng Uri:

```
Trang web Uri = Uri.parse(url);
```

3. Tạo một Intent mới với Intent. ACTION_VIEW làm hành động và URI là dữ liệu:

Ý định = ý định mới(Intent.ACTION_VIEW, trang web);

Công hàm Intent này khác với hàm bạn đã sử dụng để tạo ra một hàm Intent rõ ràng. Trong hàm tạo trước, bạn đã chỉ định ngữ cảnh hiện tại và một thành phần cụ thể (lớp Activity) để gửi Intent. Trong hàm khởi tạo này, bạn chỉ định một hành động và dữ liệu cho hành động đó. Các hành động được xác định bởi lớp Intent và có thể bao gồm ACTION_VIEW (để xem dữ liệu đã cho), ACTION_EDIT (để chỉnh sửa dữ liệu đã cho) hoặc ACTION_DIAL (để quay số điện thoại). Trong trường hợp này, hành động là ACTION_VIEW vì bạn muốn hiển thị trang web được chỉ định bởi URI trong biến webpage.

4. Sử dụng phương thức resolveActivity() và trình quản lý gói Android để tìm một Activity có thể xử lý Intent ngầm của bạn. Đảm bảo rằng yêu cầu được giải quyết thành công.

```
if(intent.resolveActivity(getApplicationContext()) != null) { }
```

Yêu cầu này khớp dữ liệu và hành động Intent của bạn với bộ lọc Intent cho các ứng dụng đã cài đặt trên thiết bị. Bạn sử dụng nó để đảm bảo có ít nhất một loại Activity có thể xử lý các yêu cầu của bạn.

5. Bên trong câu lệnh if, gọi startActivity() để gửi Intent.

```
startActivity(ý định);
```

6. Thêm một khối else để in thông báo Log nếu không thể giải quyết vấn đề Intent.

```
} khác {
    Log.d("ImplicitIntents", "Không thể xử lý việc này!"); }
```

Phương thức openWebsite() bây giờ sẽ trông như sau. (Nhận xét được thêm vào để rõ ràng.)

```
public void openWebsite(Xem chế độ xem) {  
    Nhận văn bản URL.  
    Url chuỗi = mWebsiteEditText.getText().toString();  
  
    Phân tích cú pháp URI và tạo ý định.  
    Trang web Uri = Uri.parse(url);  
    Ý định = ý định mới(Intent.ACTION_VIEW, trang web);  
    Tìm một hoạt động để đưa ra ý định và bắt đầu hoạt động đó.  
    if (intent.resolveActivity(getApplicationContext()) != null) {  
        startActivityForResult(ý định);  
    } khác {  
        Log.d("ImplicitIntents", "Không thể xử lý ý định này!");  
    }  
}
```

Nhiệm vụ 3: Triển khai nút Mở vị trí

Trong tác vụ này, bạn triển khai phương thức xử lý khi nhấp chuột cho nút thứ hai trong giao diện người dùng, **Open Vị trí**. Phương thức này gần giống với phương thức `openWebsite()`. Sự khác biệt là việc sử dụng URI địa lý để chỉ ra vị trí bản đồ. Bạn có thể sử dụng URI địa lý với vĩ độ và kinh độ hoặc sử dụng chuỗi truy vấn cho một vị trí chung. Trong ví dụ này, chúng tôi đã sử dụng cái sau.

3.1 Định nghĩa `openLocation()`

1. Nhập vào "openLocation" trong mã XML `activity_main.xml`.
2. Nhấn Alt + Enter (Option + Enter trên máy Mac) và chọn **Create 'openLocation (View)' trong Hoạt động chính.**

Android Studio tạo một phương thức khung trong `MainActivity` cho trình xử lý `openLocation()`.

```
public void openLocation(Xem ché độ xem) {  
}
```

3. Thêm một biến riêng tư ở đầu MainActivity để giữ đối tượng EditText cho URI vị trí.

```
riêng EditText mLocationEditText;
```

4. Trong phương thức onCreate(), sử dụng findViewById() để lấy tham chiếu đến thực thể EditText và gán nó cho biến riêng đó:

```
mLocationEditText = findViewById(R.id.location_edittext);
```

3.2 Thêm mã vào openLocation()

1. Trong phương thức openLocation() mới, thêm một câu lệnh để lấy giá trị chuỗi của mLocationEditText EditText.

```
Stri Ng Loc = mLocationEditText.getText().toString();
```

2.
Pa
RSE
T nǔ vào một đối tượng Uri với truy vấn tìm kiếm địa lý:

```
Uri địa chỉUri = Uri.parse("geo:0,0?q=" + lôc);
```

3. Tạo một

Tạo Intent với Intent.ACTION_VIEW làm hành động và lọc làm dữ liệu.

```
Intent nt_kiên quyết= mới_Ý định (Intent.ACTION_VIEW, địa chỉUri);
```

4. Giải quyết Intent và kiểm tra để đảm bảo rằng Intent đã được giải quyết thành công. Nếu vậy, startActivity(), nếu không hãy ghi lại thông báo lỗi.

```
if(intent.resolveActivity(getApplicationContext()) != null) {  
    startActivity(ý định);  
} khác {  
    Log.d("ImplicitIntents", "Không thể xử lý ý định này!"); }
```

Phương thức openLocation() bây giờ sẽ trông như sau (các nhận xét được thêm vào để rõ ràng):

```
public void openLocation(Xem ché độ xem) {  
    Lấy chuỗi chỉ một vị trí. Đầu vào không được xác thực; nó là  
    được chuyển đến trình xử lý vị trí nguyên vẹn.  
    Chuỗi loc = mLocationEditText.getText().toString();  
    Phân tích cú pháp vị trí và tạo ý định.  
    Uri addressUri = Uri.parse("geo:0,0?q=" + place);  
    Ý định = ý định mới(Intent.ACTION_VIEW, addressUri);  
    Tìm một hoạt động để xử lý ý định và bắt đầu hoạt động đó.  
    if (intent.resolveActivity(getApplicationContext()) != null) { startActivity(intent);  
    } khác {  
        Log.d("ImplicitIntents", "Không thể xử lý ý định này!");  
    }  
}
```

Nhiệm vụ 4: Triển khai nút Chia sẻ văn bản này

Hành động chia sẻ là một cách dễ dàng để người dùng chia sẻ các mục trong ứng dụng của bạn với mạng xã hội và các ứng dụng khác. Mặc dù bạn có thể tạo một hành động chia sẻ trong ứng dụng của riêng mình bằng cách sử dụng Intent ngầm, nhưng Android cung cấp [lớp trợ giúp ShareCompat.IntentBuilder](#) để giúp việc triển khai chia sẻ trở nên dễ dàng. Bạn có thể sử dụng ShareCompat.IntentBuilder để tạo một Intent và khởi chạy một bộ chọn để cho phép người dùng chọn ứng dụng đích để chia sẻ.

Trong tác vụ này, bạn thực hiện chia sẻ một chút văn bản trong chính sửa văn bản, sử dụng lớp ShareCompat.IntentBuilder.

4.1 Định nghĩa shareText()

1. Nhập vào "shareText" trong mã XML activity_main.xml.
2. Nhấn Alt + Enter (Option + Enter trên máy Mac) và chọn **Create 'shareText (View)' trong Hoạt động chính.**

Android Studio tạo một phương thức khung trong MainActivity cho trình xử lý shareText().

```
công khai Void shareText(Xem xem) {  
}
```

3. Quảng
các

để giữ EditText.

```
tư Chinh sửaVăn bản mShareTextEdit;
```

4. Trong o
đến t

nCreate(), sử dụng findViewById() để lấy tham chiếu đến thực thể EditText và gán nó
Mù riêng v có thể sử dụng:

```
mShareTextEdit = findViewById(R.id.share_edittext);
```

4.2 Thêm mã vào shareText()

- Trong phương thức shareText() mới, thêm một câu lệnh để lấy giá trị chuỗi của mShareTextEdit EditText.

```
Chuỗi txt = mShareTextEdit.getText().toString();
```

2. Xác định loại mime của văn bản để chia sẻ:

```
String mimeType = "text/plain";
```

3. Gọi ShareCompat.IntentBuilder bằng các phương thức sau:

```
ShareCompat.IntentBuilder .from(cái này)
.setType(mimeType)
.setChooserTitle("Chia sẻ văn bản này với: ")
.setText(txt)
.startChooser();
```

4. Trích xuất giá trị của .setChooserTitle vào tài nguyên chuỗi.

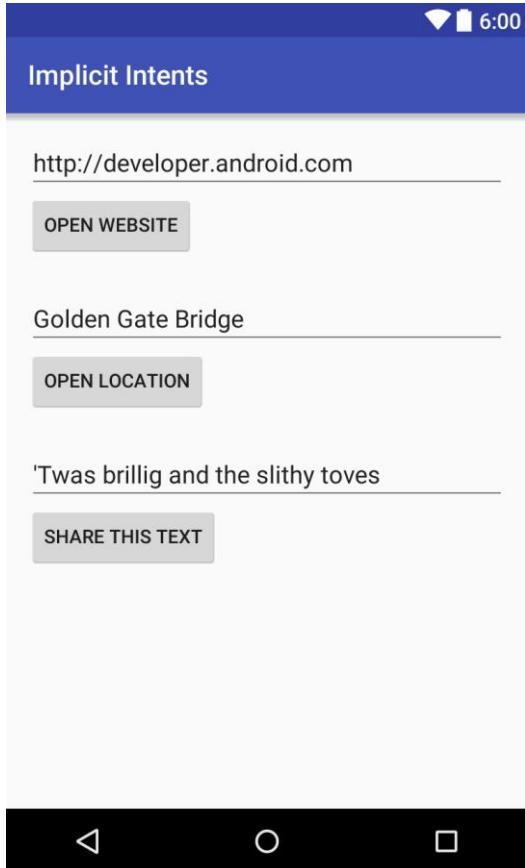
Lệnh gọi đến ShareCompat.IntentBuilder sử dụng các phương thức sau:

Phương pháp	Sự miêu tả
tù()	A activity khởi động chia sẻ này Intent (t của anh ấy).
setType()	Loại MIME của mục được chia sẻ.
setChooserTitle()	Tiêu đề xuất hiện trên bộ chọn ứng dụng hệ thống.
setText()	Văn bản thực tế sẽ được chia sẻ
startChooser()	Hiển thị bộ chọn ứng dụng hệ thống và gửi Intent.

Định dạng này, với tất cả các phương thức setter của trình tạo được xâu chuỗi với nhau trong một câu lệnh, là một cách viết tắt dễ dàng để tạo và khởi chạy Intent. Bạn có thể thêm bất kỳ phương thức bổ sung nào vào danh sách.

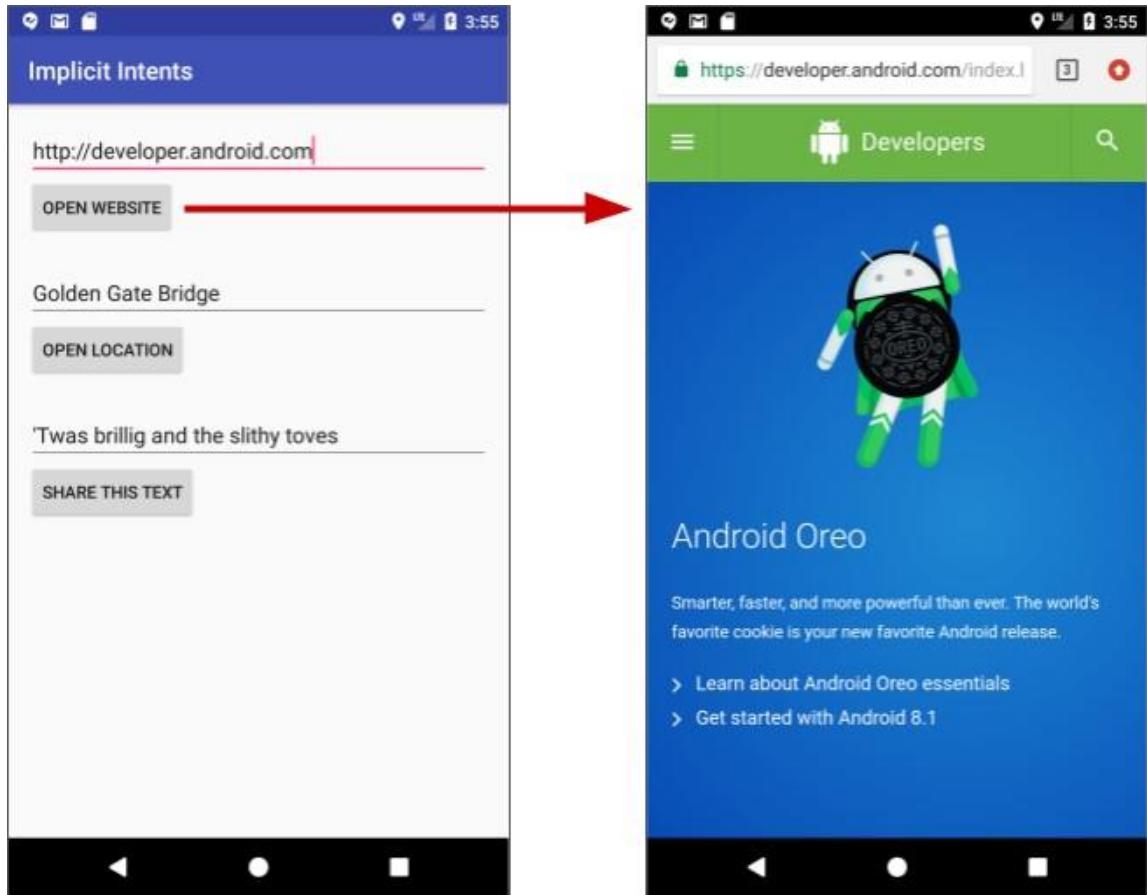
Phương thức shareText() bây giờ sẽ trông như sau:

```
public void shareText(Ché độ xem xem) {  
    Chuỗi txt = mShareTextEdit.getText().toString();  
    String mimeType = "text/plain";  
    Chia sẻCompat.IntentBuilder  
        .from(cái này)  
        .setType(mimeType)  
        .setChooserTitle(R.string.share_text_with)  
        .setText(txt)  
        .startChooser(); }
```



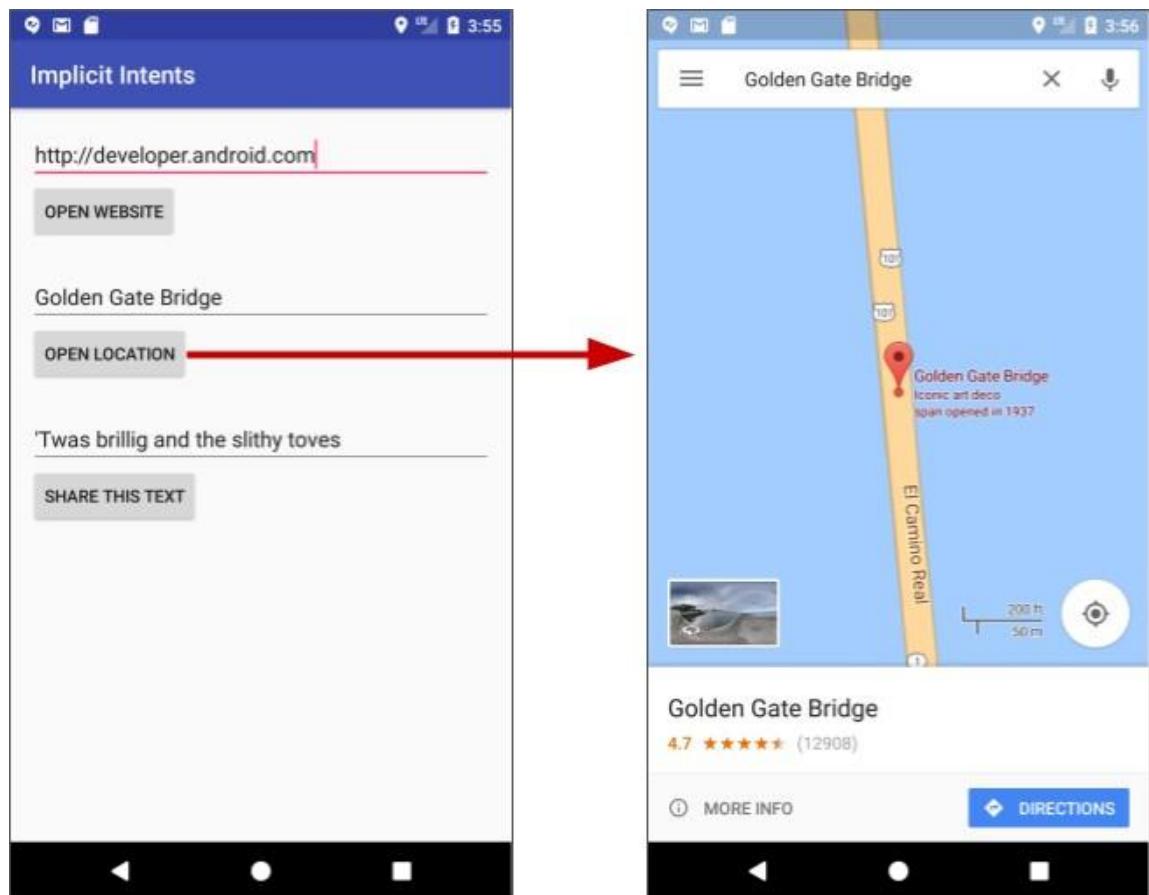
4.3 Chạy ứng dụng

1. Chạy ứng dụng.
2. Nhập vào Open Website button để khởi chạy trình duyệt với URL trang web trong EditText phía trên Button. Trình duyệt và trang web sẽ xuất hiện như hình dưới đây.



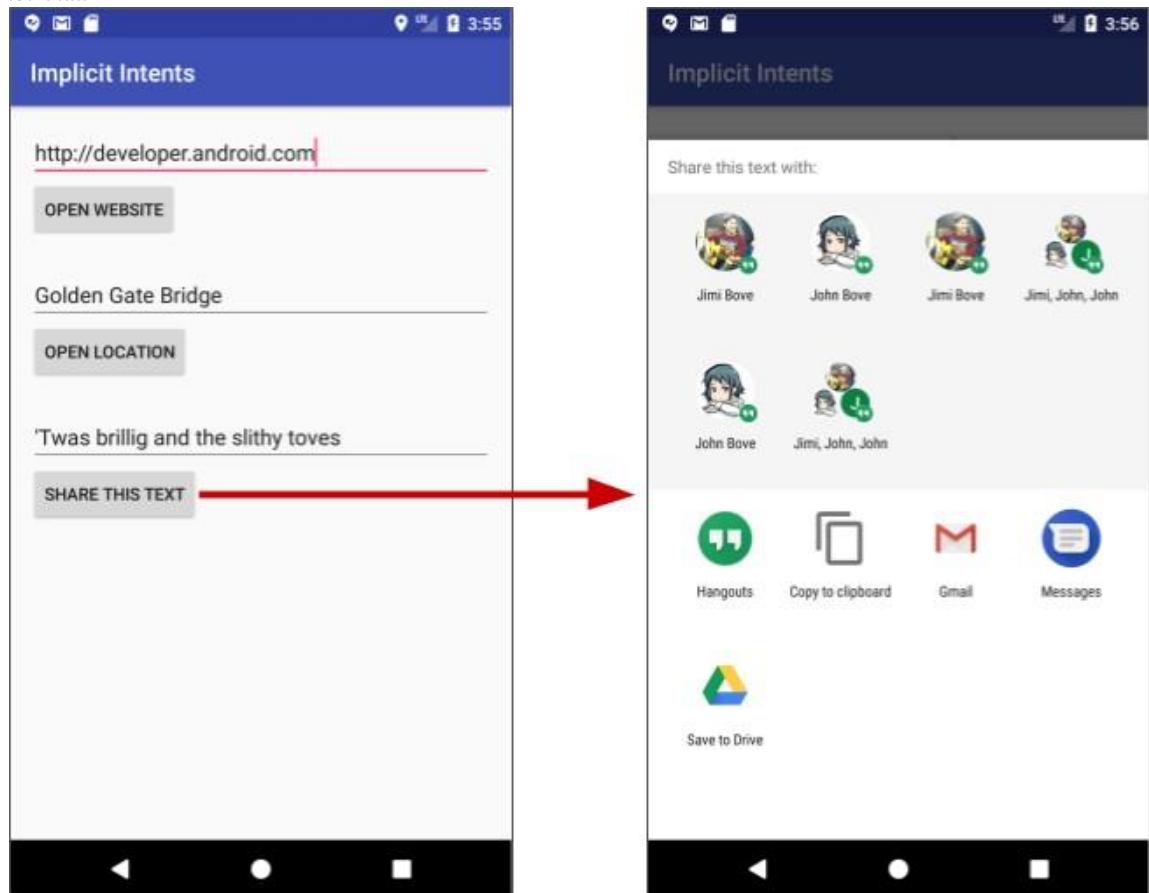
3. Nhập vào **nút Vị trí Open** để khởi chạy bản đồ với vị trí trong EditText phía trên Button. Bản đồ với vị trí sẽ xuất hiện như hình dưới đây.

Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị



- Nhấn vào **S**hare This Text để khởi chạy hộp thoại với các lựa chọn chia sẻ văn bản. Hộp thoại với các lựa chọn sẽ xuất hiện như hình dưới đây.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#). PDF này là ảnh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2 để cập nhật mới nhất.



Mã giải pháp nhiệm vụ 4

Dự án Android Studio: [ImplicitIntents](#)

Nhiệm vụ 5: Nhận ý định ngầm

Cho đến nay, bạn đã tạo một ứng dụng sử dụng Intent ngầm để khởi chạy một số ứng dụng khác

Hoạt động. Trong nhiệm vụ này, bạn xem xét vấn đề từ cách khác: cho phép Activity trong ứng dụng của bạn phản hồi với một Intent ngầm được gửi từ một số ứng dụng khác.

Activity trong ứng dụng của bạn luôn có thể được kích hoạt từ bên trong hoặc bên ngoài ứng dụng của bạn với một Ý định. Để cho phép Activity nhận được Intent ngầm, bạn xác định bộ lọc Intent trong tệp AndroidManifest.xml của ứng dụng để cho biết loại Intent ngầm nào mà Activity của bạn quan tâm đến việc xử lý.

Để khớp yêu cầu của bạn với một ứng dụng cụ thể được cài đặt trên thiết bị, hệ thống Android sẽ khớp Intent ngầm của bạn với Activity có bộ lọc Intent cho biết rằng họ có thể thực hiện hành động đó. Nếu có nhiều ứng dụng được cài đặt phù hợp, người dùng sẽ thấy một bộ chọn ứng dụng cho phép họ chọn ứng dụng mà họ muốn sử dụng để xử lý ứng dụng đó.

Khi một ứng dụng trên thiết bị gửi Intent ngầm, hệ thống Android sẽ khớp hành động và dữ liệu của Intent đó với bất kỳ ứng dụng Activity nào bao gồm các bộ lọc Intent phù hợp. Khi

Bộ lọc ý định cho một Activity A khớp với Intent:

- Nếu chỉ có một khớp Activity, Android cho phép Activity tự xử lý Intent.
- Nếu có nhiều kết quả trùng khớp, Android sẽ hiển thị trình chọn ứng dụng để cho phép người dùng chọn ứng dụng mà họ muốn thực hiện hành động đó.

Trong nhiệm vụ này, bạn tạo một ứng dụng rất đơn giản nhận được một Intent ngầm để mở URI cho một trang web. Khi được kích hoạt bởi một Intent ngầm, ứng dụng đó sẽ hiển thị URI được yêu cầu dưới dạng một chuỗi trong TextView.

5.1 Tạo dự án và bố cục

1. Tạo một dự án Android Studio mới với tên ứng dụng **Implicit Intents Receiver** và chọn **Empty Activity** cho mẫu dự án.
2. Chấp nhận tên Activity mặc định (MainActivity). Nhập vào Next.
3. Đảm bảo hộp **Generate file** được chọn. Nhập vào Finish.
4. Mở **activity_main.xml**.
5. Trong TextView ("Hello World") hiện có, xóa thuộc tính android:text. Không có văn bản nào trong TextView này theo mặc định, nhưng bạn sẽ thêm URI từ Intent trong onOptionsItemSelected().
6. Để yên các thuộc tính layout_constraint, nhưng thêm các thuộc tính sau:

Thuộc tính	Giá trị
------------	---------

android:id	"@+id/text_uri_message"
android:kích thước văn bản	"18sp"
android:textStyle	"đậm"

5.2 Sửa đổi AndroidManifest.xml để thêm bộ lọc Ý định

1. Mở tệp AndroidManifest.xml .
2. Lưu ý rằng MainActivity đã có bộ lọc Intent này:

```
<intentfilter>
<hành động android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" /></intentfilter>
```

Bộ lọc Intent này, là một phần của tệp khai dự án mặc định, chỉ ra rằng Activity này là điểm vào chính cho ứng dụng của bạn (nó có hành động Intent là "android.intent.action.MAIN") và Activity này sẽ xuất hiện dưới dạng mục cấp cao nhất trong trình khởi chạy (danh mục của nó là " android.intent.category.LAUNCHER").

3. Thêm thẻ `< intent-filter >` thứ hai bên trong activity> < và bao gồm các yếu tố sau:

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.BROWSABLE" />
<data android:scheme="http" android:host="developer.android.com" />
```

Các dòng này xác định một bộ lọc Intent cho Activity A, nghĩa là loại Intent mà Hoạt động có thể xử lý. Bộ lọc Intent này tuyên bố các yếu tố sau:

Loại bộ lọc	Giá trị	Trận đấu
hành động	"android.intent.action.VIEW"	Bất kỳ Intent nào với hành động xem.
loại	"android.intent.category.DEFAULT"	Bất kỳ ngầm nào với Intent. Danh mục này phải được đưa vào để Activity của bạn nhận được một Intent ngầm.
loại	"android.intent.category.BROWSABLE"	Yêu cầu liên kết có thể duyệt từ các trang web, email hoặc các nguồn khác.
dữ liệu	android:scheme="http" android:host="developer.android.com"	URI có chứa lược đồ http và tên máy chủ developer.android.com.

Lưu ý rằng bộ lọc dữ liệu có hạn chế về cả loại liên kết mà nó sẽ chấp nhận và tên máy chủ cho các URI đó. Nếu bạn muốn người nhận của mình có thể chấp nhận bất kỳ liên kết nào, bạn có thể bỏ qua phần tử <data>.

Phần xin của AndroidManifest.xml bây giờ sẽ trông như sau:

```
<application  
    android:allowBackup="true"  
    android:icon="@mipmap/ic_launcher"  
    android:label="@string/app_name"  
    android:roundIcon="@mipmap/ic_launcher_round"  
    android:supportsRtl="true"  
    android:theme="@style/AppTheme">  
    <activity android:name=".Hoạt động chính">  
        <intent-filter>
```

```
<hành động android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intentfilter>

<intentfilter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" /> <category
    android:name="android.intent.category.BROWSABLE" />
        <dữ liệu android:scheme="http"
            android:host="developer.android.com" /></intentfilter>
</hoạt động>
</ ứng dụng>
```

5.3 Xử lý ý định

Trong phương thức `onCreate()` cho Activity của bạn, hãy xử lý Intent đến cho bất kỳ dữ liệu hoặc tính năng bổ sung nào mà nó bao gồm. Trong trường hợp này, Intent ngầm đến có URI được lưu trữ trong dữ liệu Intent.

1. Mở MainActivity.
2. Trong phương thức `onCreate()`, lấy N đến đã được sử dụng để kích hoạt Activity:

```
Ý định = getIntent();
```

3. Lấy dữ liệu i. Intent data luôn là một đối tượng URI:

```
uri uri = intent.getData();
```

4. Kiểm tra để đảm bảo rằng biến uri không phải là null. Nếu kiểm tra đó vượt qua, hãy tạo một chuỗi từ đối tượng URI đó:

```
if(uri != null) {  
    Chuỗi uri_string = "URI: " + uri.toString(); }
```

5. Trích xuất phần "URI:" của phần trên vào tài nguyên chuỗi (uri_label).
6. Bên trong cùng một khái niệm đó, lấy TextView cho thông báo:

```
TextView textView = findViewById(R.id.text_uri_message);
```

7. Cũng bên trong i

f b khóa, đặt văn bản của TextView đó thành URI:

```
textView.setText(uri_string);
```

Phương thức onCreate() cho MainActivity bây giờ sẽ trông như sau:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Ý định = getIntent();  
    uri uri = intent.getData();  
    if (uri != null) {  
        Chuỗi uri_string = getString(R.string.uri_label)  
            + uri.toString();  
        TextView textView = findViewById(R.id.text_uri_message);  
        textView.setText(uri_string);  
    }  
}
```

5.4 Chạy cả hai ứng dụng

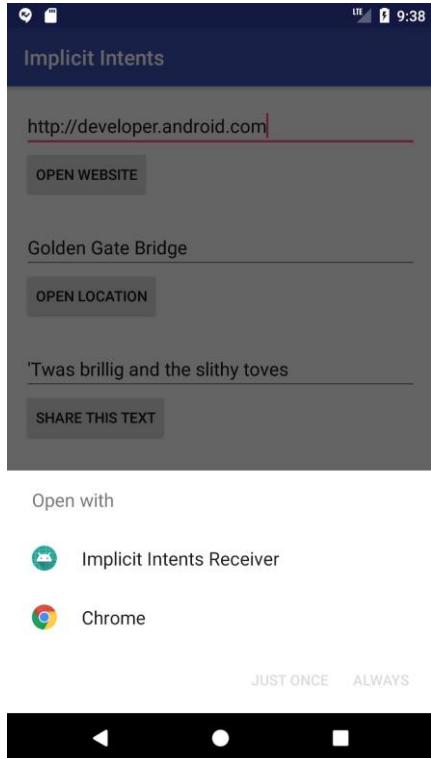
Để hiển thị kết quả nhận được Intent ngầm, bạn sẽ chạy cả ứng dụng Thu ý định ngầm và Ứng dụng Ý định ngầm trên trình mô phỏng hoặc thiết bị của bạn.

1. Chạy ứng dụng Bộ thu ý định ngầm.

Tự chạy ứng dụng hiển thị một đoạn A trong không có văn bản. Điều này là do Activity đã được kích hoạt từ trình khởi chạy hệ thống chứ không phải với Intent từ một ứng dụng khác.

2. Chạy ứng dụng Ý định ngầm và nhấp vào **Trang web Open** với URI mặc định.

Một công cụ chọn ứng dụng xuất hiện hỏi bạn muốn sử dụng trình duyệt mặc định (Chrome trong hình bên dưới) hay ứng dụng Thu nhận ý định ngầm. Chọn **Tôi implicit Bộ thu ý định** và nhấp vào **Just Once**. Ứng dụng Thu nhận ý định ngầm sẽ khởi chạy và thông báo hiển thị URI từ yêu cầu ban đầu.



3. Nhấn vào nút Quay lại và nhập URI khác. Nhập vào **O pen Website**.

Ứng dụng nhận có một bộ lọc Intent rất hạn chế chỉ khớp với giao thức URI chính xác (`http://`) và máy chủ (`developer.android.com`). bất kỳ URI nào khác sẽ mở trong trình duyệt web mặc định.

Mã giải pháp nhiệm vụ 5

Dự án Android Studio: [ImplicitIntentsReceiver](#)

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 này.

Thử thách: Trong thử thách thực tế trước đó, bạn đã tạo một trình tạo ứng dụng danh sách mua sắm với một Hoạt động để hiển thị danh sách và một hoạt động khác để chọn một mục. Thêm EditText và Button vào danh sách mua sắm Activity để xác định vị trí một cửa hàng cụ thể trên bản đồ.

Tóm tắt

- Một Intent ngầm cho phép bạn kích hoạt Activity nếu bạn biết hành động, nhưng không phải ứng dụng cụ thể hoặc Activity sẽ xử lý hành động đó.
- Một Activity có thể nhận được Intent ngầm phải xác định các bộ lọc Intent trong tệp AndroidManifest.xml phù hợp với một hoặc nhiều hành động và danh mục Intent.
- Hệ thống Android khớp nội dung của bộ lọc Intent và Intent của bất kỳ Activity nào có sẵn để xác định Activity nào sẽ kích hoạt. Nếu có nhiều hơn một Activity có sẵn, hệ thống sẽ cung cấp một bộ chọn để người dùng có thể chọn một.
- Lớp ShareCompat.IntentBuilder giúp dễ dàng xây dựng một Intent ngầm để chia sẻ dữ liệu lên phương tiện truyền thông xã hội hoặc email.

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [2.3: Implicit intents](#).

Tìm hiểu thêm

Tài liệu dành cho nhà phát triển Android:

- [Nguyên tắc cơ bản về ứng dụng](#)
- [Hoạt động](#)
- [Hiểu vòng đời hoạt động](#)
- [Ý định và bộ lọc ý định](#)
- [Cho phép các ứng dụng khác bắt đầu hoạt động của bạn](#)
- [Ý định Google Maps dành cho Android](#)
- [Hoạt động](#)

- [Kiên quyết](#)
- [<bộ lọc ý định>](#)
- [<hoạt động>](#)
- [Uri](#)
- [Chia sẻCompat.IntentBuilder](#)

Homework

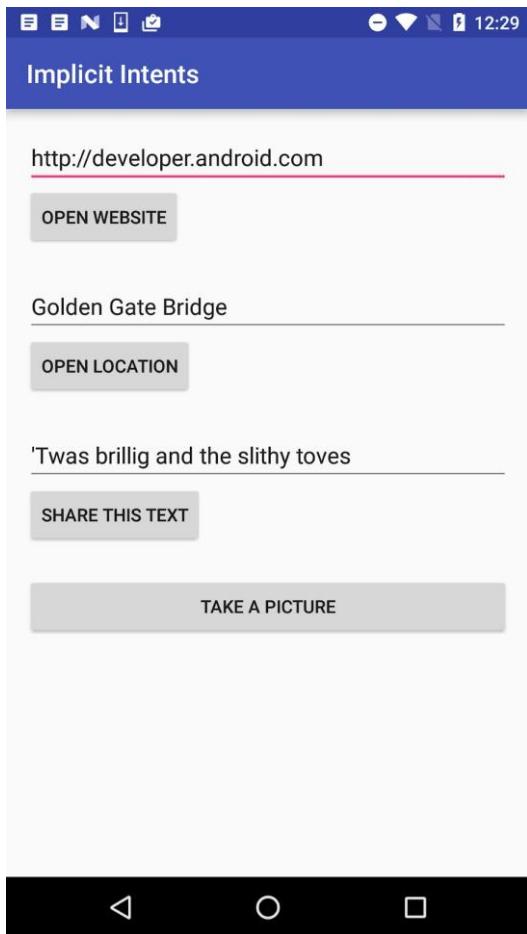
Xây dựng và chạy ứng dụng

Mở ứng dụng [ImplicitIntents](#) mà bạn đã tạo.

1. Thêm một nút khác ở cuối màn hình.
2. Khi nhấp vào nút B, hãy khởi chạy ứng dụng máy ảnh để chụp ảnh. (Bạn không cần phải trả lại ảnh về ứng dụng gốc.)

Ghi:

Nếu bạn sử dụng trình mô phỏng Android để kiểm tra máy ảnh, hãy mở cấu hình trình mô phỏng trong trình quản lý Android AVD, chọn **Cài đặt nâng cao**, sau đó chọn **E mulated** cho cả camera trước và sau. Khởi động lại trình mô phỏng của bạn nếu cần.



Trả lời những câu hỏi này

Câu hỏi 1

Bạn sử dụng phương thức constructor nào để tạo một Intent ngầm để khởi chạy ứng dụng máy ảnh?

- ý định mới ()
- new Intent(Context context, Class<?> class)

- ý định mới (Hành động chuỗi, uri uri)
- ý định mới (Hành động chuỗi)

Câu hỏi 2

Khi bạn tạo ra một đối tượng Intent ngầm, điều nào sau đây là đúng?

- Không chỉ định Activity cụ thể hoặc thành phần khác để khởi chạy.
- Thêm hành động Intent hoặc Intent loại (hoặc cả hai).
- Giải quyết vấn đề Intent với hệ thống trước khi gọi startActivity() hoặc startActivityForResult().
- Tất cả những điều trên.

Câu hỏi 3

Bạn sử dụng hành động nào để chụp ảnh bằng ứng dụng máy ảnh?

- Ý định takePicture = ý định mới (Intent.ACTION_VIEW);
- Ý định takePicture = ý định mới (Intent.ACTION_MAIN);
- Ý định takePicture = ý định mới (MediaStore.ACTION_IMAGE_CAPTURE);
- Ý định takePicture = ý định mới (Intent.ACTION_GET_CONTENT);

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kiểm tra xem ứng dụng có các tính năng sau không:

- Nó hiển thị nút Take a Picture ở cuối ứng dụng.
- Khi được nhập vào, nút sẽ khởi chạy ứng dụng máy ảnh trên thiết bị.

- Trước khi gửi ý định, phương thức `onClickListener()` cho **Button** đảm bảo rằng một ứng dụng có sẵn trên thiết bị, sử dụng các phương thức `resolveActivity()` và `getPackageManager()`.

Bài 3.1: Trình gỡ lỗi

Giới thiệu

Trong các bài thực hành trước đây, bạn đã sử dụng [lớp Log](#) để in thông tin vào nhật ký hệ thống, thông tin này xuất hiện trong [ngăn Logcat](#) trong Android Studio khi ứng dụng của bạn chạy. Thêm câu lệnh ghi nhật ký vào ứng dụng là một cách để tìm lỗi và cải thiện hoạt động của ứng dụng. Một cách khác là sử dụng trình gỡ lỗi được tích hợp trong Android Studio.

Trong thực tế này, bạn sẽ tìm hiểu cách gỡ lỗi ứng dụng của mình trong trình mô phỏng và trên thiết bị, đặt và xem các điểm ngắt, thực hiện từng bước qua mã và kiểm tra các biến.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo một dự án Android Studio.
- Sử dụng trình soạn thảo bố cục để làm việc với các phần tử `EditText` và `Button`.
- Xây dựng và chạy ứng dụng của bạn trong Android Studio, trên cả trình mô phỏng và trên thiết bị.
- Đọc và phân tích dấu vết ngăn xếp, bao gồm cả lần cuối, lần trước.
- Thêm câu lệnh nhật ký và xem nhật ký hệ thống ([ngăn Logcat](#)) trong Android Studio.

Những gì bạn sẽ học

- Cách chạy ứng dụng ở chế độ gỡ lỗi trong trình mô phỏng hoặc trên thiết bị.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#). PDF này là Ảnh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2 để cập nhật mới nhất.

Trang 250

- Cách thực hiện ứng dụng của bạn.
- Cách thiết lập và sắp xếp các điểm ngắt.
- Cách kiểm tra và sửa lỗi các biến trong trình gõ lỗi.

Bạn sẽ làm gì

- Xây dựng ứng dụng SimpleCalc.
- Đặt và xem các điểm ngắt trong mã cho SimpleCalc.
- Bước qua mã của bạn khi nó chạy.
- Kiểm tra các biến và đánh giá biểu thức.
- Xác định và khắc phục sự cố trong ứng dụng mẫu.

Tổng quan về ứng dụng

Ứng dụng SimpleCalc có hai phần tử EditText và bốn phần tử Button. Khi bạn nhập hai số và nhấp vào A Button, ứng dụng sẽ thực hiện phép tính cho B button đó và hiển thị kết quả.

Trang 251



Nhiệm vụ 1: Khám phá dự án và ứng dụng SimpleCalc

Đối với thực tế này, bạn sẽ không tự xây dựng ứng dụng SimpleCalc. Dự án hoàn chỉnh có sẵn tại [SimpleCalc](#). Trong tác vụ này, bạn mở dự án SimpleCalc trong Android Studio và khám phá một số tính năng chính của ứng dụng.

1.1 Tải xuống và mở dự án SimpleCalc

1. Tải xuống [SimpleCalc](#) và giải nén tệp.

creativecommons

Commons để cập nhật

mới nhất.

2. Khởi động Android Studio và chọn **File > Open**.
3. Điều hướng đến thư mục cho SimpleCalc, chọn tệp thư mục đó và nhấp vào **OK**. Dự án SimpleCalc được xây dựng.
4. Mở **ngân Project > Android** nếu nó chưa mở.

Cảnh báo: Ứng dụng này chứa các lỗi mà bạn sẽ tìm thấy và khắc phục. Nếu chạy ứng dụng trên thiết bị hoặc trình mô phỏng, bạn có thể gặp phải hành vi không mong muốn, bao gồm cả sự cố trong ứng dụng.

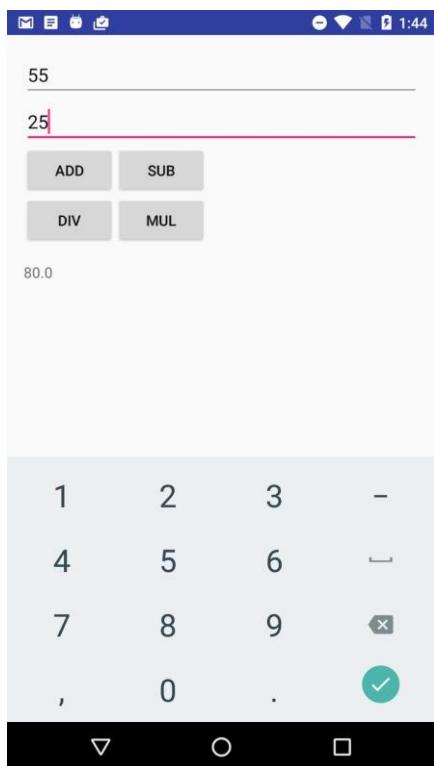
1.2 Khám phá bố cục

1. Mở **activity_main.xml**.
2. Nhấp vào tab **Text** để xem mã XML.
3. Nhấp vào tab **Danh giá** P để xem trước bố cục.

Kiểm tra mã và thiết kế XML bố cục và lưu ý những điều sau:

- Bố cục chứa hai phần tử `EditText` cho đầu vào, bốn phần tử `Button` cho các phép tính và một phần tử `TextView` để hiển thị kết quả.
- Mỗi phép tính `Button` có một trình xử lý nhấp chuột `onClick` riêng (`onAdd`, `onSub`, v.v.)
- `TextView` cho kết quả không có bất kỳ văn bản nào trong đó theo mặc định.
- Hai phần tử `EditText` có thuộc tính `android:inputType` và giá trị "`numberDecimal`". Thuộc tính này chỉ ra rằng `EditText` chỉ chấp nhận các số làm đầu vào. Bàn phím xuất hiện trên màn hình sẽ chỉ chứa số. Bạn sẽ tìm hiểu thêm về các kiểu đầu vào cho các phần tử `EditText` trong phần thực tế sau.

Trang 253



1.3 Khám phá mã ứng dụng

1. Mở rộng **thư mục app > java** trong ngăn **Project > Android**. Ngoài lớp **MainActivity**, dự án này còn bao gồm một lớp **calculator** tiện ích C.
2. Mở **C alculator** và kiểm tra mã. Lưu ý rằng các phép toán mà máy tính có thể thực hiện được xác định bởi số Operator và tất cả các phương thức hoạt động đều là public. 3. Mở **M ainActivity** và kiểm tra mã và nhận xét.

Lưu ý những điều sau:

creativecommons để cập nhật mới nhất.

Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị

- Tất cả các trình xử lý nhấp chuột android:onClick được xác định đều gọi phương thức compute() riêng, với tên hoạt động là một trong những giá trị từ trình phát âm C. Liệt kê toán tử.
- Phương thức compute() gọi phương thức private getOperand() (lần lượt gọi getOperandText()) để truy xuất các giá trị số từ các phần tử EditText.
- Sau đó, phương thức compute() sử dụng phù thủy số trên tên toán hạng để gọi phương thức thích hợp trong thực thể alculator C (m Calculator).
- Các phương thức tính toán trong lớp tính toán C thực hiện số học thực tế và trả về một giá trị.
- Phần cuối cùng của phương thức compute() cập nhật TextView với kết quả tính toán.

1.4 Chạy ứng dụng

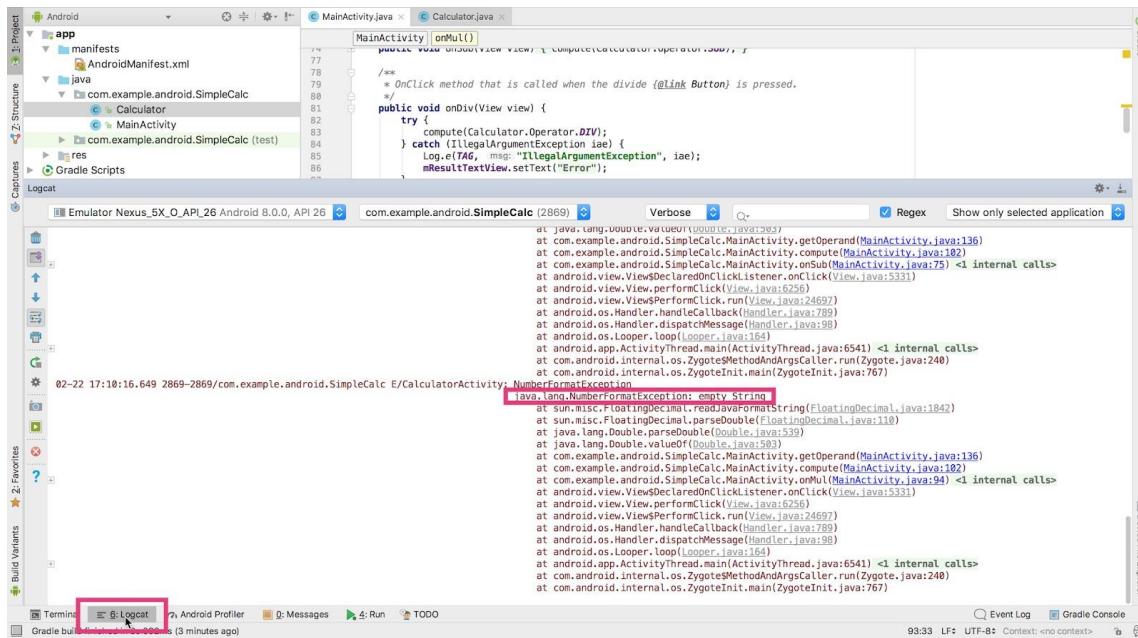
Chạy ứng dụng và làm theo các bước sau:

1. Nhập cả giá trị số nguyên và dấu phẩy động để tính toán.
2. Nhập các giá trị dấu phẩy động có phân số thập phân lớn (ví dụ: **1 .6753456**)
3. Chia một số cho không.
4. Để trống một hoặc cả hai phần tử EditText và thử bất kỳ phép tính nào.
5. Nhấp vào tab **Logcat** ở cuối cửa sổ Android Studio để mở ngăn **Logcat** (nếu chưa mở). Kiểm tra dấu vết ngăn xếp tại thời điểm ứng dụng báo cáo lỗi.

Nếu một hoặc cả hai phần tử EditText trong SimpleCalc trống, ứng dụng sẽ báo cáo một ngoại lệ, như thể hiện trong hình bên dưới và nhật ký hệ thống hiển thị trạng thái của ngăn xếp thực thi tại thời điểm ứng dụng tạo ra lỗi. Theo dõi ngăn xếp thường cung cấp thông tin quan trọng về lý do tại sao xảy ra lỗi.

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0.
 PDF này là Ảnh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2

Trang 255



Nhiệm vụ 2: Chạy SimpleCalc trong trình gõ lỗi

Trong tác vụ này, bạn sẽ giới thiệu về trình gõ lỗi trong Android Studio và tìm hiểu cách đặt điểm ngắt và chạy ứng dụng ở chế độ gõ lỗi.

2.1 Khởi động và chạy ứng dụng của bạn ở chế độ gỡ lỗi

- Trong Android Studio, hãy chọn Run > Debug app hoặc nhấp vào **biểu tượng D ebug** trên thanh công cụ.
- Nếu ứng dụng của bạn đã chạy, bạn sẽ được hỏi có muốn khởi động lại ứng dụng ở chế độ gỡ lỗi hay không. Nhấp vào **ứng dụng R estart**.

Android Studio sẽ xây dựng và chạy ứng dụng của bạn trên trình mô phỏng hoặc trên thiết bị. Gỡ lỗi là giống nhau trong cả hai trường hợp. Trong khi Android Studio đang khởi tạo trình gỡ lỗi, bạn có thể thấy thông báo "Đang chờ trình gỡ lỗi" trên thiết bị trước khi có thể sử dụng ứng dụng của mình.

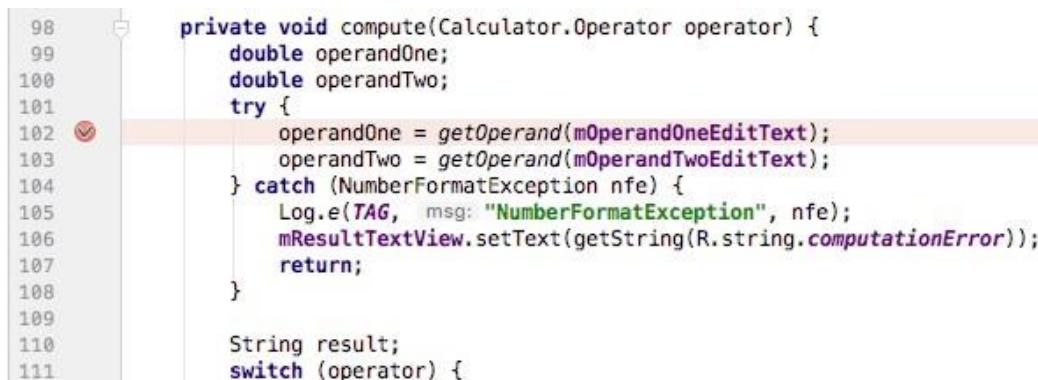
- Nhấp vào thẻ **D ebug** ở cuối cửa sổ Android Studio để hiển thị **ngăn D ebug** (hoặc chọn View > Tool Windows > Debug). Tab **D ebugger** trong ngăn đã được chọn, hiển thị ngăn **D ebugger**.

2.2 Đặt điểm ngắt

Điểm ngắt là một vị trí trong mã mà bạn muốn tạm dừng quá trình thực thi bình thường của ứng dụng để thực hiện các hành động khác như kiểm tra biến hoặc đánh giá biểu thức hoặc thực thi mã từng dòng để xác định nguyên nhân gây ra lỗi thời gian chạy. Bạn có thể đặt điểm ngắt trên bất kỳ dòng mã thực thi nào.

- Mở **MainActivity** và nhấp vào dòng thứ tư của phương thức `c ompute()` (dòng ngay sau câu lệnh `t ry`).
- Nhấp vào máng xói bên trái của ngăn trình chỉnh sửa tại dòng đó, bên cạnh số dòng. Một chấm màu đỏ xuất hiện ở dòng đó, cho biết một điểm ngắt. Dấu chấm màu đỏ bao gồm dấu kiểm nếu ứng dụng đã chạy ở chế độ gỡ lỗi.

Thay vào đó, bạn có thể chọn Run > Toggle Line Breakpoint hoặc nhấn Control-F8 (Command-F8 trên máy Mac) để đặt hoặc xóa điểm ngắt tại một dòng.



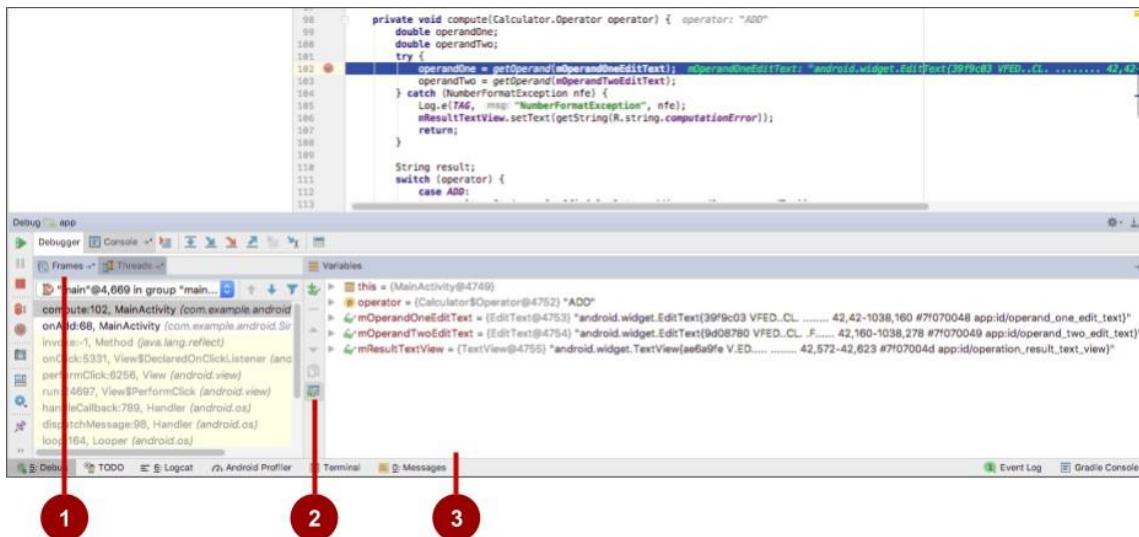
```
98     private void compute(Calculator.Operator operator) {  
99         double operandOne;  
100        double operandTwo;  
101        try {  
102            operandOne = getOperand(mOperandOneEditText);  
103            operandTwo = getOperand(mOperandTwoEditText);  
104        } catch (NumberFormatException nfe) {  
105            Log.e(TAG, msg: "NumberFormatException", nfe);  
106            mResultTextView.setText(getString(R.string.computationError));  
107            return;  
108        }  
109        String result;  
110        switch (operator) {  
111    }
```

Nếu bạn nhập nhầm vào một điểm ngắt, bạn có thể hoàn tác nó bằng cách nhấp vào điểm ngắt. Nếu bạn nhập vào một dòng mã không thể thực thi, dấu chấm màu đỏ sẽ bao gồm một "x" và một cảnh báo xuất hiện rằng dòng mã đó không thể thực thi.

- Trong ứng dụng SimpleCalc, nhập số vào các phần tử EditText và nhấp vào một trong các phần tử tính toán B.

Trang 257

Quá trình thực thi ứng dụng của bạn sẽ dừng khi nó đạt đến điểm ngắt mà bạn đã đặt và trình gõ lỗi hiển thị trạng thái hiện tại của ứng dụng tại điểm ngắt đó như trong hình bên dưới.



Hình trên cho thấy ngăn **D ebug** với các tab **D ebugger** và **C**. Tab **D ebugger** được chọn, hiển thị ngăn **D ebugger** với các tính năng sau:

- Tab khung:** Nhập để hiển thị ngăn **F rames** với khung ngăn xếp thực thi hiện tại cho một luồng nhất định. Ngăn xếp thực thi hiển thị từng lớp và phương thức đã được gọi trong ứng dụng của bạn và trong thời gian chạy Android, với phương thức gần đây nhất ở trên cùng.

Nhấp vào tab **T hreads** để thay thế ngăn **F rames** bằng ngăn **T hreads**. Ứng dụng của bạn hiện đang chạy trong luồng chính và ứng dụng đang thực thi phương thức `c ompute()` trong Hoạt động chính.

2. **Nút Đồng hồ:** Bấm để hiển thị khung **Watches** trong khung **Variables**, hiển thị các giá trị cho bất kỳ đồng hồ biến nào bạn đã đặt. Đồng hồ cho phép bạn theo dõi một biến cụ thể trong chương trình của mình và xem biến đó thay đổi như thế nào khi chương trình của bạn chạy.
3. **Variables pane:** Hiển thị các biến trong phạm vi hiện tại và giá trị của chúng. Ở giai đoạn này của quá trình thực thi ứng dụng của bạn, các biến có sẵn là: `this` (đối với `Activity`), `operator` (tên toán tử từ `Calculator`.
Toán tử mà phương thức được gọi từ đó), cũng như các biến toàn cục cho các phần tử `EditText` và `TextView`. Mỗi biến trong ngăn này có biểu tượng mở rộng để mở rộng danh sách các thuộc tính đối tượng cho biến. Hãy thử mở rộng một biến để khám phá các thuộc tính của biến đó.

2.3 Tiếp tục thực thi ứng dụng của bạn

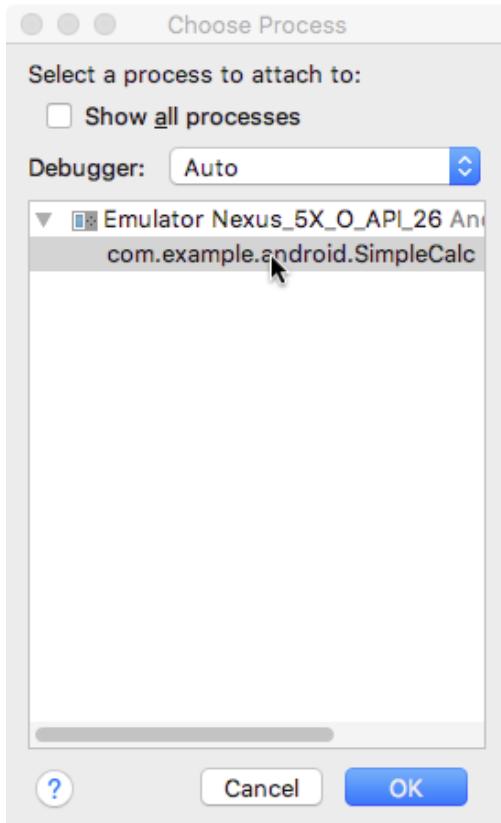
Tiếp tục thực thi ứng dụng của bạn bằng cách chọn **R un > Resume Program** hoặc nhấp vào biểu tượng **R esume**  ở phía bên trái của cửa sổ trình gỡ lỗi.

Ứng dụng SimpleCalc tiếp tục chạy và bạn có thể tương tác với ứng dụng cho đến lần thực thi mã tiếp theo đến điểm ngắt.

2.4 Gỡ lỗi ứng dụng đang chạy

Nếu ứng dụng của bạn đang chạy trên thiết bị hoặc trình mô phỏng và bạn quyết định muốn gỡ lỗi ứng dụng đó, bạn có thể chuyển ứng dụng đang chạy sang chế độ gỡ lỗi.

1. Chạy ứng dụng SimpleCalc bình thường, với biểu tượng **R un** .
2. Chọn **R un > Attach debugger to Android process** hoặc nhấp vào nút **A ttach**  i con trên thanh công cụ.
3. Chọn quy trình của ứng dụng từ hộp thoại xuất hiện (hiển thị bên dưới). Nhấp vào **OK**



Ngăn **D ebug** xuất hiện với ngăn **D ebugger** đang mở và giờ đây bạn có thể gỡ lỗi ứng dụng của mình như thế bạn đã khởi động ứng dụng ở chế độ gỡ lỗi.

Lưu ý: Nếu ngăn **D ebug** không tự động xuất hiện, hãy nhấp vào tab **D ebug** ở cuối màn hình. Nếu nó chưa được chọn, hãy nhấp vào tab **D ebugger** trong ngăn **D ebug** để hiển thị ngăn Debugger .

Nhiệm vụ 3: Khám phá các tính năng của trình gỡ lỗi

Trong tác vụ này, chúng ta sẽ khám phá các tính năng khác nhau trong trình gỡ lỗi Android Studio, bao gồm thực thi ứng dụng của bạn từng dòng, làm việc với điểm ngắt và kiểm tra các biến.

3.1 Bước qua quá trình thực thi ứng dụng của bạn

Sau một điểm ngắt, bạn có thể sử dụng trình gỡ lỗi để thực thi từng dòng mã trong ứng dụng của mình và kiểm tra trạng thái của các biến khi ứng dụng chạy.

1. Gỡ lỗi ứng dụng của bạn trong Android Studio với điểm ngắt bạn đã đặt trong tác vụ cuối cùng.
2. Trong ứng dụng, nhập số vào cả hai phần tử EditText và nhấp vào nút **A dd**.

Quá trình thực thi ứng dụng của bạn dừng tại điểm ngắt mà bạn đã đặt trước đó và **ngăn E bugger D hiển thị** trạng thái hiện tại của ứng dụng. Dòng hiện tại được đánh dấu trong mã của bạn.

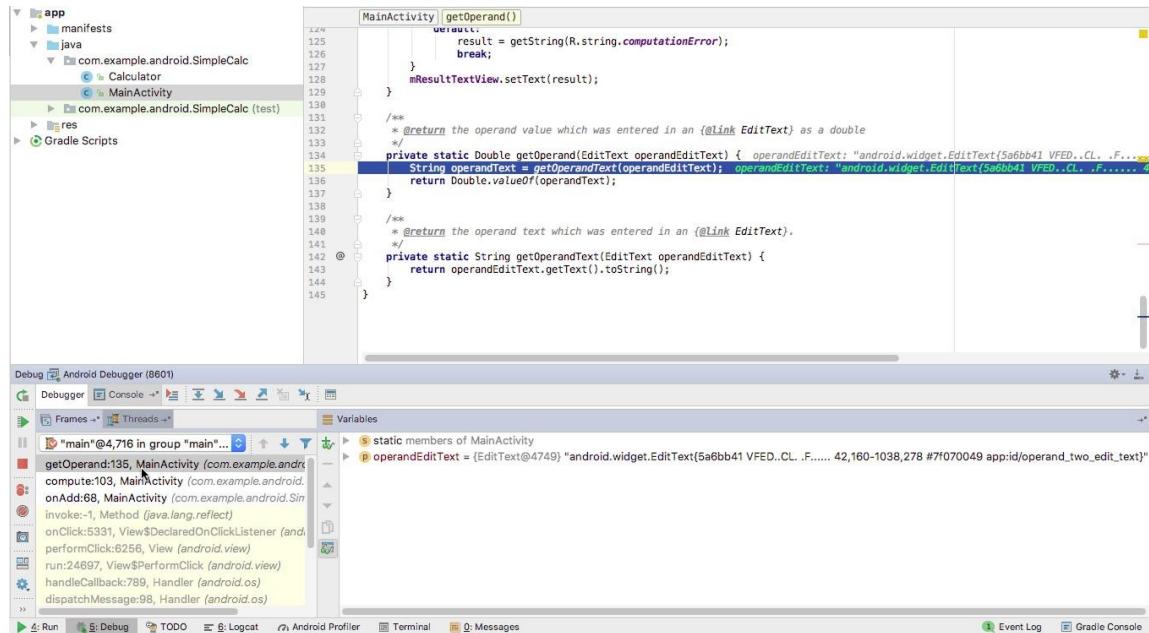
3. Nhấp vào nút **S tep Over**  ở đầu cửa sổ trình gỡ lỗi.

Trình gỡ lỗi thực thi dòng hiện tại trong phương thức compute() (trong đó điểm ngắt là, gán cho o perandOne) và phần đánh dấu di chuyển đến dòng tiếp theo trong mã (gán cho o perandTwo). Ngăn **V ariables** cập nhật để phản ánh trạng thái thực thi mới và giá trị hiện tại của các biến cũng xuất hiện sau mỗi dòng mã nguồn của bạn bằng chữ nghiêng. Bạn cũng có thể sử dụng **R un > Step Over** hoặc nhấn F 8 để bước qua mã của mình.

4. Tại dòng tiếp theo (bài tập cho o perandTwo), nhấp vào biểu tượng **S tep Into** .

S tep Into nhảy vào việc thực thi một cuộc gọi phương thức trong dòng hiện tại (so với việc chỉ thực hiện phương thức đó và vẫn ở trên cùng một dòng). Trong trường hợp này, vì việc gán đó bao gồm lệnh gọi đến getOperand(), trình gỡ lỗi sẽ cuộn mã MainActivity đến định nghĩa phương thức đó.

Khi bạn bước vào một phương thức, **ngăn F rames** sẽ cập nhật để chỉ ra khung mới trong ngăn xếp lệnh gọi (ở đây là getOperand()) và ngăn **V có thể hiển thị** các biến có sẵn trong phạm vi phương thức mới. Bạn có thể nhấp vào bất kỳ dòng nào trong ngăn **F rames** để xem điểm trong khung ngăn xếp trước đó mà phương thức đã được gọi.



Bạn cũng có thể sử dụng **Run > Step Into** hoặc F7 để bước vào một phương thức.

- Nhập vào **Step Over** để chạy từng dòng trong `getOperand()`. Lưu ý rằng khi phương thức hoàn tất, trình gõ lỗi sẽ đưa bạn về điểm mà bạn bước vào phương thức đầu tiên và tắt cả các bảng cập nhật để hiển thị thông tin mới.
- Nhập vào **Step Over** hai lần để di chuyển điểm thực thi đến dòng đầu tiên bên trong câu lệnh `case` cho `ADD`.
- Nhập vào **Step Into** .

Trình gõ lỗi thực thi phương thức thích hợp được xác định trong lớp `Calculator`, mở tệp `Calculator.java` và cuộn đến điểm thực thi trong lớp đó. Một lần nữa, các ngăn khác nhau cập nhật để phản ánh trạng thái mới.

- Sử dụng biểu tượng **Step Out** để thực hiện phần còn lại của phương thức tính toán đó và bật ra trả lời lại phương thức `compute()` trong `MainActivity`. Sau đó, bạn có thể tiếp tục gõ lỗi phương thức `compute()` từ nơi bạn đã dừng lại.

Bạn cũng có thể sử dụng **Run > Step Out** hoặc nhấn **Shift-F8** để bước ra khỏi một phương thức thực thi.

creativecommons để cập

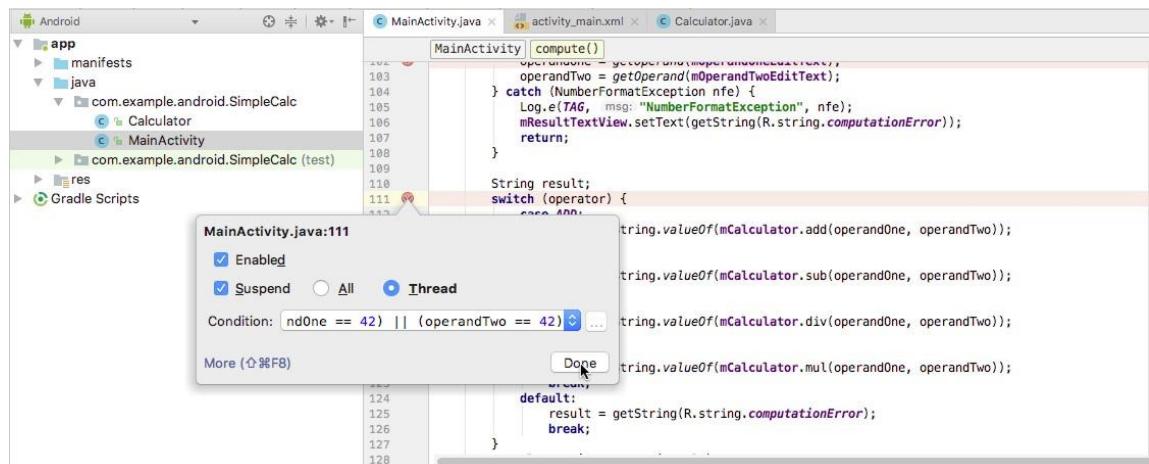
nhật mới nhất.

3.2 Làm việc với Breakpoint

Sử dụng điểm ngắt để cho biết vị trí bạn muốn làm gián đoạn quá trình thực thi ứng dụng trong mã để gỡ lỗi phần đó của ứng dụng đó.

1. Tìm điểm ngắt bạn đã đặt trong tác vụ cuối cùng — ở đầu phương thức compute() trong MainActivity.
2. Thêm một điểm ngắt vào đầu câu lệnh phù thủy.
3. Nhấp chuột phải vào điểm ngắt mới đó để nhập một điều kiện, như thể hiện trong hình bên dưới và nhập thử nghiệm sau vào trường C :

(operandOne == 42) ||(toán hạngHai == 42)



4. Nhập vào **D** một.

Điểm ngắt thứ hai này là một *điểm ngắt bắt đầu*. Quá trình thực thi ứng dụng của bạn sẽ chỉ dừng tại điểm ngắt này nếu kiểm tra trong điều kiện là đúng. Trong trường hợp này, biểu thức chỉ đúng nếu một hoặc các toán hạng khác mà bạn đã nhập là 4 2. Bạn có thể nhập bất kỳ biểu thức Java nào làm điều kiện miễn là nó trả về boolean.

5. Chạy ứng dụng của bạn ở chế độ gỡ lỗi (**R un > Debug**) hoặc nhấp vào **R esume** nếu ứng dụng đã chạy. Trong ứng dụng, nhập hai số khác với 42 và nhấp vào nút **A dd**. Việc thực thi dừng tại điểm ngắt đầu tiên trong phương thức compute().

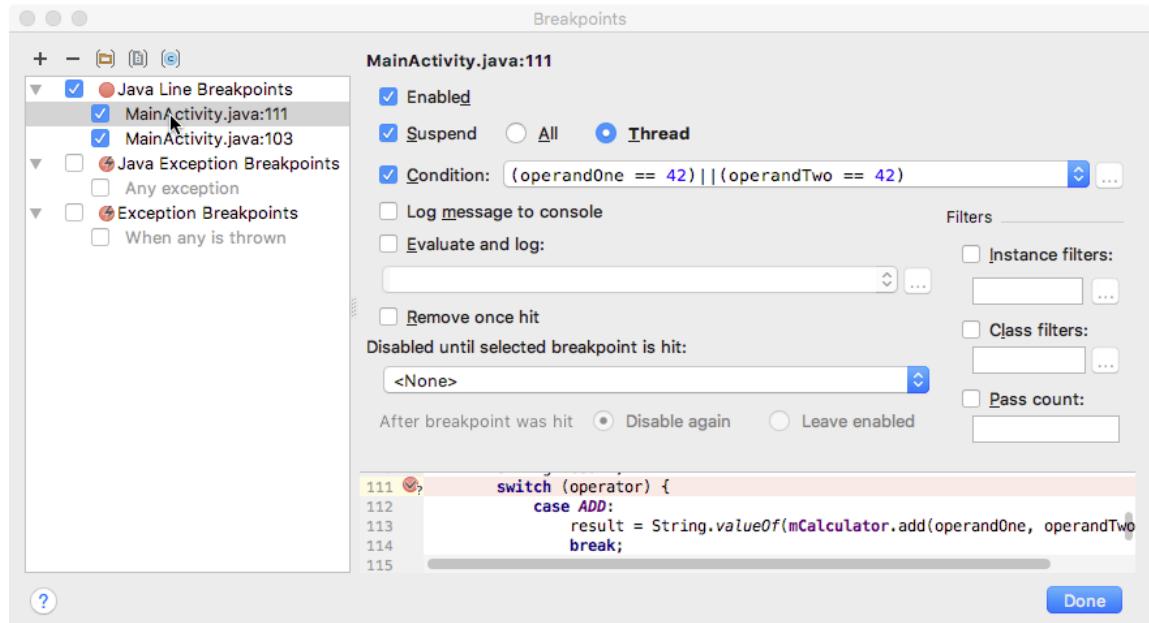
creative Commons
Khóa học cơ bản về nhà phát triển Android (V2) *Đơn vị*

6. Nhập vào **R esume** để tiếp tục gỡ lỗi ứng dụng. Quan sát rằng quá trình thực thi không dừng lại ở điểm ngắt thứ hai của bạn vì điều kiện không được đáp ứng.
7. Trong ứng dụng, nhập **4 2** vào `EditText` đầu tiên và nhấp vào bất kỳ nút B nào. Nhập vào **R esume** để tiếp tục thực thi sau điểm ngắt đầu tiên. Quan sát rằng điểm ngắt thứ hai tại câu lệnh `phù thủy s — điểm ngắt bắn đĩa c — tạm dừng thực thi` vì điều kiện đã được đáp ứng.
8. **Nhấp chuột phải** (hoặc **C ontrol-click**) điểm ngắt đầu tiên trong `c ommute()` và bỏ chọn **E nabled**. Bấm **Xong**. Quan sát rằng biểu tượng điểm ngắt bây giờ có một chấm màu xanh lá cây với một đường viền màu đỏ.

Tắt điểm ngắt cho phép bạn tạm thời "tắt tiếng" điểm ngắt đó mà không thực sự xóa nó khỏi mã của bạn. Nếu bạn xóa hoàn toàn một điểm ngắt, bạn cũng mất bất kỳ điều kiện nào bạn đã tạo cho điểm ngắt đó, vì vậy việc tắt nó thường là lựa chọn tốt hơn.

Bạn cũng có thể tắt tiếng tất cả các điểm ngắt trong ứng dụng của mình cùng một lúc với **biểu tượng M ute Breakpoints**.

9. Nhập vào **V iew Breakpoints** ở cạnh trái của cửa sổ trình gỡ lỗi. Cửa sổ **B reakpoints** xuất hiện.
Cửa sổ **B reakpoints** cho phép bạn xem tất cả các điểm ngắt trong ứng dụng của mình, bật hoặc tắt các điểm ngắt riêng lẻ và thêm các tính năng bổ sung của điểm ngắt bao gồm các điều kiện, phần phụ thuộc vào các điểm ngắt khác và ghi nhật ký.



Để đóng cửa sổ **B breakpoints**, hãy nhấp vào **D** một.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#). PDF này là ảnh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2 để cập nhật mới nhất.

Trang 264

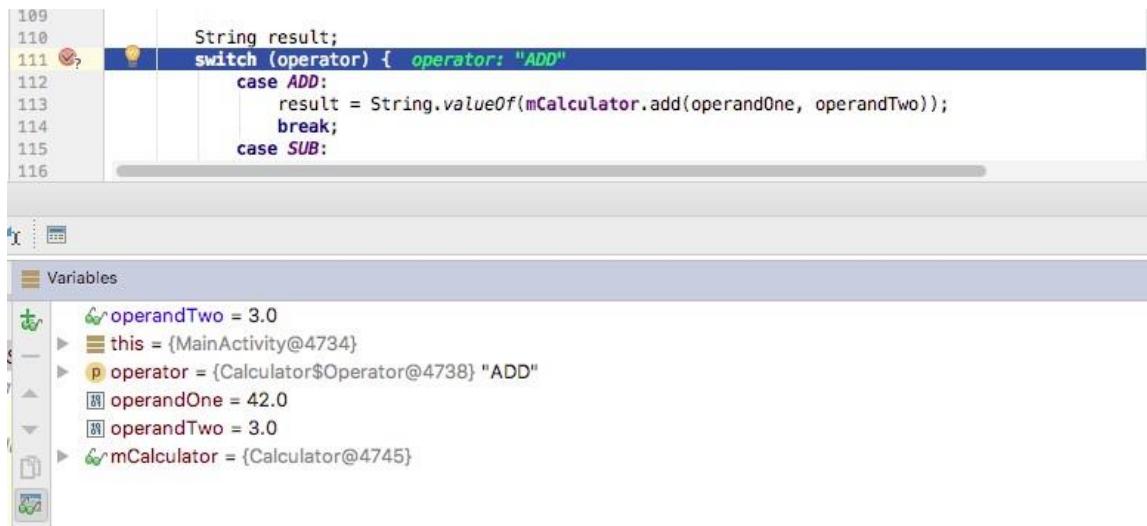
3.3 Kiểm tra và sửa đổi các biến

Trình gõ lỗi Android Studio cho phép bạn kiểm tra trạng thái của các biến trong ứng dụng khi ứng dụng đó chạy.

1. Chạy ứng dụng SimpleCalc ở chế độ gõ lỗi nếu ứng dụng chưa chạy.
2. Trong ứng dụng, nhập hai số, một trong số đó là **42** và nhấp vào nút **Add**.

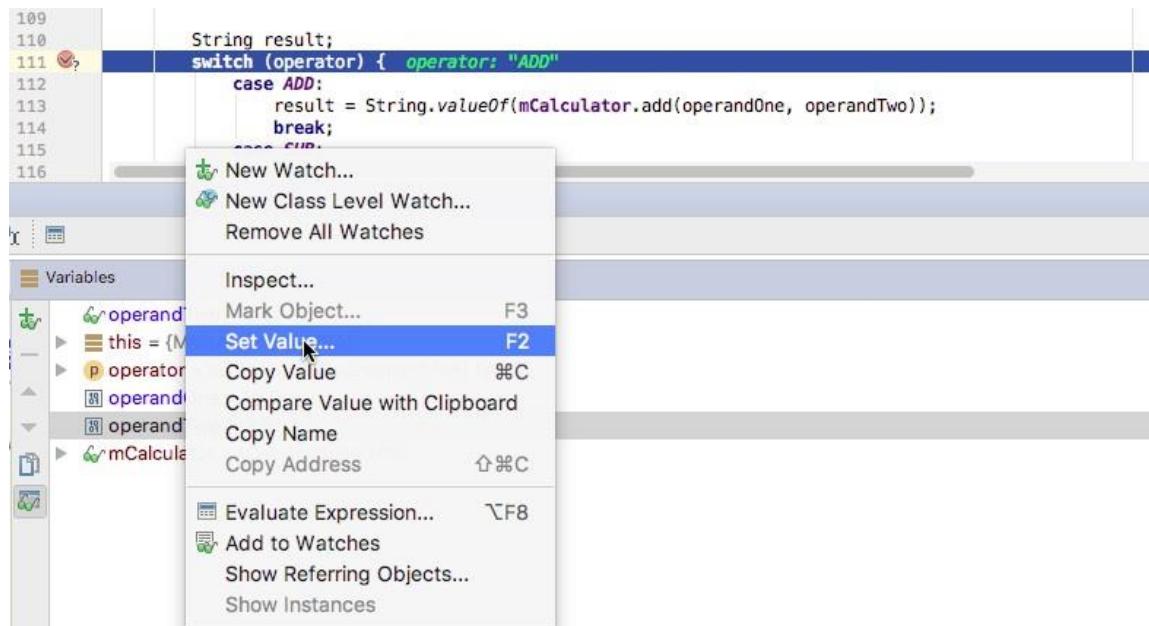
Điểm ngắt đầu tiên trong `c compute()` vẫn bị tắt tiếng. Việc thực hiện dừng ở điểm ngắt thứ hai (điểm ngắt có điều kiện tại câu lệnh `switch`), và trình gõ lỗi xuất hiện.

3. Quan sát trong ngăn **Variables** rằng các biến `operandOne` và `operandTwo` có các giá trị bạn đã nhập vào ứng dụng.



4. Biến `operator` của anh ấy là một đối tượng `MainActivity`. Nhấp vào biểu tượng mở rộng để xem danh sách các biến thành viên của đối tượng đó. Nhấp vào biểu tượng mở rộng một lần nữa để đóng danh sách.
5. Nhấp chuột phải (hoặc C khi nhấp vào) biến `operandOne` trong ngăn **Variables** và chọn **Set Value**.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).



6. Thay đổi giá trị của operandOne thành **1 0** và nhấn **R eturn**.
7. Thay đổi giá trị của operandTwo thành **1 0** theo cách tương tự và nhấn **R eturn**.
8. Quan sát rằng kết quả trong ứng dụng bây giờ dựa trên các giá trị biến mà bạn đã thay đổi trong trình gõ lỗi; ví dụ: vì bạn đã nhấp vào nút **A dd** button ở Bước 2, kết quả trong ứng dụng bây giờ là **20**.
9. Nhập vào biểu tượng **R esume** để tiếp tục chạy ứng dụng của bạn.
10. Trong ứng dụng, các mục gốc (bao gồm **4 2**) được giữ nguyên trong các phần tử EditText. (Giá trị của chúng chỉ được thay đổi trong trình gõ lỗi.) Nhấp vào nút **A dd**. Việc thực hiện lại dừng tại điểm ngắt.

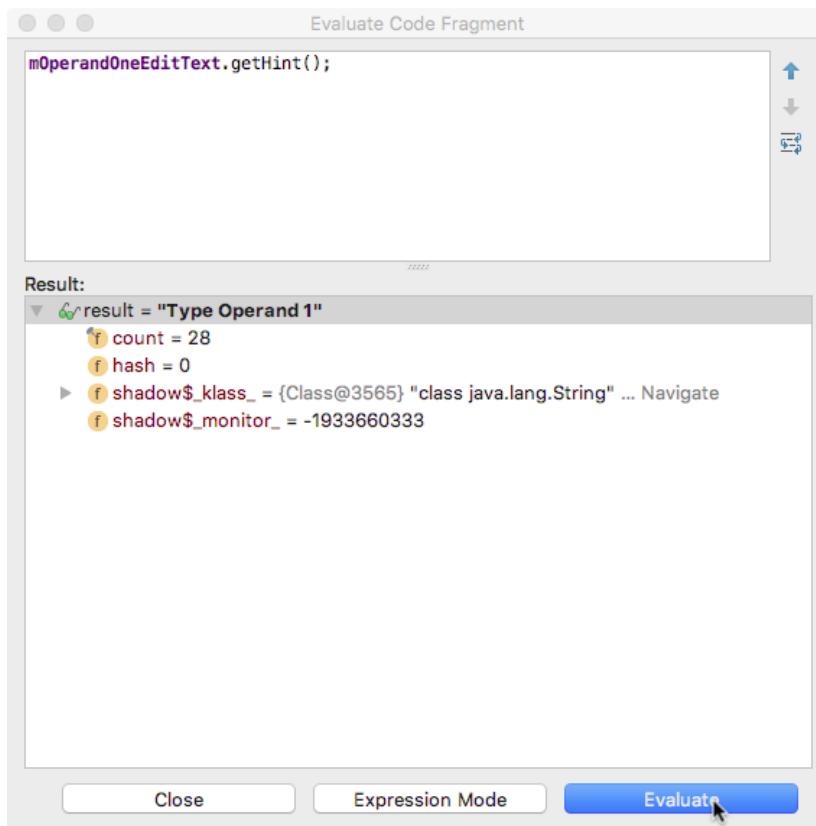
11. Nhấp vào biểu tượng **E valuate Expression** hoặc chọn **R un > Evaluate Expression**. Bạn cũng có thể **nhấp chuột phải** (hoặc **nhấp vào C**) bất kỳ biến nào và chọn **E valuate Expression**.

Cửa sổ **E valuate Code Fragment** xuất hiện. Sử dụng tính năng này để khám phá trạng thái của các biến và đối tượng trong ứng dụng của bạn, bao gồm cả việc gọi các phương thức trên các đối tượng đó. Bạn có thể nhập bất kỳ mã nào vào cửa sổ này.

creativecommons để cập

nhật mới nhất.

Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị



12. Nhập câu lệnh `m OperandOneEditText.getHint();` vào trường trên cùng của **cửa sổ Evaluate Code Fragment** (như trong hình trên) và nhấp vào **Evaluate**.
13. Trường Kết quả hiển thị kết quả của biến thức đó. Gọi ý cho `EditText` này là chuỗi "Type Operand 1", như được định nghĩa ban đầu trong XML cho `EditText` đó.

Kết quả bạn nhận được từ việc đánh giá một biến thức dựa trên trạng thái hiện tại của ứng dụng. Tùy thuộc vào giá trị của các biến trong ứng dụng tại thời điểm bạn đánh giá biến thức, bạn có thể nhận được các kết quả khác nhau.

Cũng lưu ý rằng nếu bạn sử dụng **Biểu thức đánh giá E** để thay đổi giá trị của các biến hoặc thuộc tính đối tượng, bạn sẽ thay đổi trạng thái đang chạy của ứng dụng.

14. Nhập vào C **thua** để đóng cửa sổ **Evaluate Code Fragment**.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#)
để cập nhật mới nhất.

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 này.

Thử thách: Vào cuối Tác vụ 1, bạn đã thử chạy ứng dụng SimpleCalc mà không có giá trị nào trong một trong các phần tử EditText, dẫn đến lỗi. Sử dụng trình gõ lỗi để thực thi mã và xác định chính xác lý do tại sao lỗi này xảy ra. Khắc phục lỗi gây ra lỗi này.

Tóm tắt

- Xem thông tin ghi nhật ký trong Android Studio bằng cách nhấp vào **tab Logcat**.
- Chạy ứng dụng của bạn ở chế độ gõ lỗi bằng cách nhấp vào biểu tượng Gõ lỗi hoặc chọn **R un > Gõ lỗi ứng dụng**.
- Nhấp vào tab **D ebug** để hiển thị ngăn **D ebug**. Nhấp vào tab **D ebugger** trong ngăn **D ebug** để hiển thị ngăn **D ebugger** (nếu chưa được chọn).
- Ngăn **debugger D** hiển thị (ngăn xếp) **các rames F**, **các tệp V** trong một khung cụ thể và **các cổng W** (theo dõi chủ động một biến trong khi chương trình chạy).
- Điểm ngắt là một vị trí trong mã mà bạn muốn tạm dừng quá trình thực thi bình thường của ứng dụng để thực hiện các hành động khác. Đặt hoặc xóa điểm ngắt gõ lỗi bằng cách nhấp vào máng xôi bên trái của cửa sổ trình chỉnh sửa ngay bên cạnh dòng mục tiêu

Khái niệm liên quan

Tài liệu về khái niệm liên quan có trong [3.1: Trình gõ lỗi Android Studio](#).

creativecommons.org/licenses/by/4.0/deed.vi

*Commons để cập nhật
mới nhất.*

Tìm hiểu thêm

Tài liệu Android Studio:

- [Hướng dẫn sử dụng Android Studio](#)
- [Gỡ lỗi ứng dụng của bạn](#)
- [Ghi và xem nhật ký](#)
- [Phân tích dấu vết ngắn xép](#)
- [Cầu gỡ lỗi Android](#)
- [Hỗ trợ Android](#)
- [Trình phân tích mạng](#)
- [Trình phân tích CPU](#)
- [Chế độ xem theo dõi](#)

Khác:

- Video: [Debugging và thử nghiệm trong Android Studio](#)

Homework

Xây dựng và chạy ứng dụng

Mở [ứng dụng](#) SimpleCalc.

1. Trong MainActivity, đặt một điểm ngắt trên dòng đầu tiên của phương thức onAdd().
2. Chạy ứng dụng trong trình gỡ lỗi. Thực hiện thao tác thêm trong ứng dụng. Quá trình thực hiện dừng lại ở điểm ngắt.
3. Nhập vào Step Into để theo dõi quá trình thực hiện ứng dụng từng bước. Lưu ý rằng Step Into mở và thực thi các tệp từ khung Android, cho phép bạn xem bản thân Android hoạt động như thế nào trên mã của mình.
4. Kiểm tra cách ngăn Debug thay đổi khi bạn bước qua mã cho khung ngắn xép hiện tại và các biến cục bộ.
5. Kiểm tra cách mã trong ngắn trình chỉnh sửa được chú thích khi mỗi dòng được thực thi.
6. Nhập vào Step Out để quay lại ứng dụng của bạn nếu ngắn xép thực thi quá sâu để hiểu.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Trả lời những câu hỏi này

Câu hỏi 1

Chạy ứng dụng SimpleCalc mà không cần trình gỡ lỗi. Để trống một hoặc cả hai phần tử EditText và thử bất kỳ phép tính nào. Tại sao lỗi xảy ra?

- java.lang.NumberFormatException: chuỗi trống
- W / OpenGLRenderer: Không thể chọn cấu hình với EGL_SWAP_BEHAVIOR_PRESERVED
- Ứng dụng có thể đang thực hiện quá nhiều công việc trên chủ đề chính của nó.
- Dung lượng bộ nhớ đệm mã đã được tăng lên 128KB.

Câu hỏi 2

Bạn thực hiện chức năng nào trong ngăn Debug để thực thi dòng hiện tại nơi có điểm ngắt, và sau đó dừng lại ở dòng tiếp theo trong mã? Chọn một:

- **Bước vào**
- **Bước qua**
- **Bước ra**
- **Tiếp tục**

Câu hỏi 3

Bạn thực hiện chức năng nào trong ngăn Debug để chuyển đến việc thực thi một phương thức gọi từ dòng hiện tại nơi có điểm ngắt? Chọn một:

- **Bước vào**
- **Bước qua**
- **Bước ra**
- **Tiếp tục**

reative Commons

Trang 270

Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Không có ứng dụng nào để gửi cho bài tập về nhà này.

Bài 3.2: Kiểm tra đơn vị

Giới thiệu

Kiểm tra mã của bạn có thể giúp bạn phát hiện lỗi sớm trong quá trình phát triển, khi lỗi ít tốn kém nhất để giải quyết. Khi ứng dụng của bạn trở nên lớn hơn và phức tạp hơn, việc kiểm tra sẽ cải thiện tính mạnh mẽ của mã. Với các thử nghiệm trong mã, bạn có thể thực hiện các phần nhỏ của ứng dụng một cách riêng lẻ và bạn có thể kiểm tra theo những cách có thể tự động hóa và lặp lại.

Android Studio và Thư viện hỗ trợ thử nghiệm Android hỗ trợ một số loại thử nghiệm và khung thử nghiệm khác nhau. Trong thực tế này, bạn sẽ khám phá chức năng kiểm thử tích hợp của Android Studio, đồng thời tìm hiểu cách viết và chạy kiểm thử đơn vị cục bộ.

Kiểm thử đơn vị cục bộ là các bài kiểm tra được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với Máy ảo Java (JVM). Bạn sử dụng kiểm thử đơn vị cục bộ để kiểm tra các phần của ứng dụng không cần quyền truy cập vào khung Android hoặc thiết bị hoặc trình giả lập chạy Android, ví dụ như logic nội bộ. Bạn cũng sử dụng kiểm thử đơn vị cục bộ để kiểm tra các phần của ứng dụng mà bạn có thể tạo các đối tượng giả mạo ("mô phỏng" hoặc sơ khai) giả vờ hoạt động giống như các đối tượng tương đương của framework.

Unit testing được viết bằng JUnit, một khung kiểm thử đơn vị phổ biến cho Java.

Những điều bạn nên biết

Bạn sẽ có thể:

a

for the latest updates.

- Tạo một dự án Android Studio.
- Xây dựng và chạy ứng dụng của bạn trong Android Studio, trên cả trình mô phỏng và trên thiết bị. ● Điều hướng **Project > Android** trong Android Studio.

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0. PDF này là Ánh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2

Trang 271

Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị

- Tìm các thành phần chính của dự án Android Studio, bao gồm ndroidManifest.xml A, tài nguyên, tệp Java và tệp Gradle.

Những gì bạn sẽ học

- Cách tổ chức và chạy thử nghiệm trong Android Studio.
- Hiểu kiểm thử đơn vị là gì.
- Viết kiểm thử đơn vị cho mã của bạn.

Bạn sẽ làm gì

- Chạy các bài kiểm tra ban đầu trong ứng dụng SimpleCalc.
- Thêm nhiều bài kiểm tra hơn vào ứng dụng SimpleCalc.
- Chạy các bài kiểm tra đơn vị để xem kết quả.

Tổng quan về ứng dụng

Thực tế này sử dụng ứng dụng [SimpleCalc](#) từ lớp học lập trình thực tế trước đó ([Android fundamentals 3.1: Trình gõ lõi](#)). Bạn có thể sửa đổi ứng dụng đó tại chỗ hoặc tạo một bản sao thư mục dự án của mình trước khi tiếp tục.

Nhiệm vụ 1: Khám phá và chạy CalculatorTest

Bạn viết và chạy các thử nghiệm của mình (cả kiểm thử đơn vị và kiểm thử đo lường) bên trong Android Studio, cùng với mã cho ứng dụng của bạn. Mỗi dự án Android mới đều bao gồm các lớp mẫu cơ bản để thử nghiệm mà bạn có thể mở rộng hoặc thay thế cho mục đích sử dụng của riêng mình.

Trong tác vụ này, bạn quay lại ứng dụng SimpleCalc, bao gồm một lớp kiểm thử đơn vị cơ bản.

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0. PDF này là Ánh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2 Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị

1.1 Khám phá các bộ nguồn và CalculatorTest

Bộ nguồn là tập hợp mã trong dự án của bạn dành cho các mục tiêu bản dựng khác nhau hoặc các mục tiêu khác "hương vị" của ứng dụng của bạn. Khi Android Studio tạo dự án của bạn, nó sẽ tạo ba nhóm nguồn:

- Bộ nguồn main, cho mã và tài nguyên của ứng dụng của bạn.
- Bộ nguồn (test) cho các bài kiểm tra đơn vị cục bộ của ứng dụng. Tập hợp nguồn hiển thị (test) sau tên gói.
- Bộ nguồn (androidTest) dành cho các bài kiểm tra được đo lường trên Android. Tập hợp nguồn hiển thị (androidTest) sau tên gói.

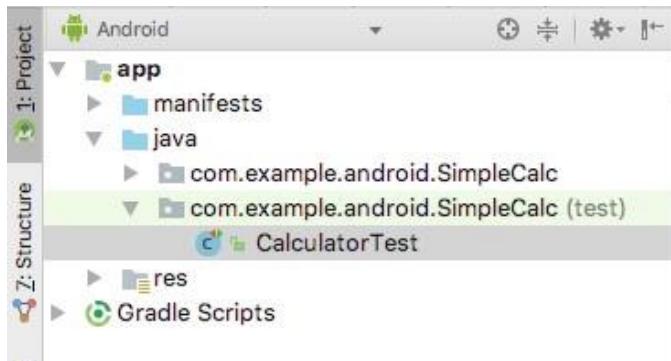
Trong tác vụ này, bạn sẽ khám phá cách các tập hợp nguồn được hiển thị trong Android Studio, kiểm tra cấu hình Gradle để thử nghiệm và chạy các bài kiểm tra đơn vị cho ứng dụng SimpleCalc.

Lưu ý: Bộ nguồn (androidTest) đã bị xóa khỏi ví dụ này để đơn giản. Nó được giải thích chi tiết hơn trong một bài học khác.

1. Mở dự án [SimpleCalc](#) trong Android Studio, nếu bạn chưa làm như vậy.

1. Mở ngăn Project > Android và mở rộng các thư mục app và java.

Thư mục java trong chế độ xem Android liệt kê tất cả các nhóm nguồn trong ứng dụng theo tên gói. Trong trường hợp này (như hiển thị bên dưới), mã ứng dụng nằm trong bộ nguồn com.android.example.SimpleCalc. Mã thử nghiệm nằm trong tập nguồn với test xuất hiện trong ngoặc đơn sau tên gói: com.android.example.SimpleCalc (test).



Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0.
 PDF này là Ánh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2
 Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị

2. Mở rộng thư mục com.android.example.SimpleCalc (test).

Thư mục này là nơi bạn đặt các bài kiểm tra đơn vị cục bộ của ứng dụng. Android Studio tạo một lớp kiểm thử mẫu cho bạn trong thư mục này cho các dự án mới, nhưng đối với SimpleCalc, lớp kiểm thử được gọi là CalculatorTest.

1. Mở C alculatorTest.

Kiểm tra mã và lưu ý những điều sau:

- Các gói nhập khẩu duy nhất là từ các gói org.junit, org.hamcrest và android.test. Không có phần phụ thuộc nào trên các lớp khung Android.
- Chú thích @RunWith(JUnit4.class) cho biết trình chạy sẽ được sử dụng để chạy các bài kiểm tra trong lớp này. Trình chạy thử nghiệm là một thư viện hoặc bộ công cụ cho phép thử nghiệm xảy ra và kết quả được in vào nhật ký. Đối với các thử nghiệm có yêu cầu thiết lập hoặc cơ sở hạ tầng phức tạp hơn (chẳng hạn như Espresso), bạn sẽ sử dụng các trình chạy thử nghiệm khác nhau. Đối với ví dụ này, chúng ta đang sử dụng trình chạy thử nghiệm JUnit4 cơ bản.
- Chú thích @ SmallTest chỉ ra rằng tất cả các thử nghiệm trong lớp này là các bài kiểm tra đơn vị không có phụ thuộc và chạy trong mili giây. Chú thích @ SmallTest, @ MediumTest và @ LargeTest là các quy ước giúp bạn dễ dàng gộp các nhóm thử nghiệm thành các bộ có chức năng tương tự.
- The setup() method is used to set up the environment before testing, and includes the @Before annotation. In this case the setup creates a new instance of the Calculator class and assigns it to the m Calculator member variable.

- The `a ddTwoNumbers()` method is an actual test, and is annotated with `@ Test`. Only methods in a test class that have an `@ Test` annotation are considered tests to the test runner. Note that by convention test methods do not include the word "test."
- The first line of a `ddTwoNumbers()` calls the `a dd()` method from the `C alculator` class. You can only test methods that are `p ublic` or `p ackage-protected`. In this case the `C alculator` is a `p ublic` class with `p ublic` methods, so all is well.
- The second line is the assertion for the test. Assertions are expressions that must evaluate and result in `t rue` for the test to pass. In this case the assertion is that the result you got from the `add` method (`1 + 1`) matches the given number `2`. You'll learn more about how to create assertions later in this practical.

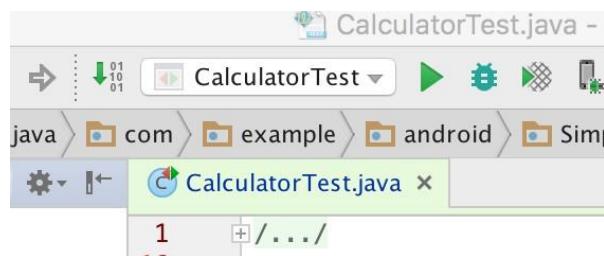
*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

1.2 Run tests in Android Studio

In this task you'll run the unit tests in the test folder and view the output for both successful and failed tests.

1. In the **P**roject > **A**ndroid pane, **r**ight-click (or **C**ontrol-click) **C**alculatorTest and select **R**un '**C**alculatorTest'.

The project builds, if necessary, and the **C**alculatorTest pane appears at the bottom of the screen. At the top of the pane, the drop-down list for available execution configurations also changes to **C**alculatorTest.



All the tests in the CalculatorTest class run, and if those tests are successful, the progress bar at the top of the view turns green. (In this case, there is currently only one test.) A status message in the footer also reports "Tests Passed."



1. Open **C**alculatorTest if it is not already open, and change the assertion in a**d**dTwoNumbers() to:

```
assertThat(resultAdd, is(equalTo(3d)));
```

a

for the latest updates.

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Page 275

2. In the run configurations dropdown menu at the top of the screen, select **C alculatorTest** (if it is not already selected) and click **R un** .

The test runs again as before, but this time the assertion fails (3 is not equal to 1 + 1.) The progress bar in the run view turns red, and the testing log indicates where the test (assertion) failed and why.

3. Change the assertion in `ddTwoNumbers()` back to the correct test and run your tests again to ensure they pass.
4. In the run configurations dropdown, select **a pp** to run your app normally.

Task 2: Add more unit tests to CalculatorTest

With unit testing, you take a small bit of code in your app such as a method or a class, and isolate it from the rest of your app, so that the tests you write makes sure that one small bit of the code works in the way you'd expect.

Typically, a unit test calls a method with a variety of different inputs, and verifies that the method does what you expect and returns what you expect it to return.

In this task you learn more about how to construct unit tests. You'll write additional unit tests for the `Calculator` utility methods in the `SimpleCalc` app, and run those tests to make sure that they produce the output you expect.

Note: Unit testing, test-driven development, and the JUnit 4 API are all large and complex topics and outside the scope of this course.

2.1 Add more tests for the `add()` method

Although it is impossible to test every possible value that the `add()` method may ever see, it's a good idea to test for input that might be unusual. For example, consider what happens if the `add()` method gets arguments:

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Page 276

- With negative operands
- With floating-point numbers

*reative Commons for the
latest updates.*

Android Developer Fundamentals Course (V2) Unit

- With exceptionally large numbers
- With operands of different types (a float and a double, for example)
- With an operand that is zero
- With an operand that is infinity

In this task we'll add more unit tests for the `a dd()` method to test different kinds of inputs.

1. Add a new method to `C alculatorTest` called `a ddTwoNumbersNegative()`. Use this skeleton:

```
@Test
public void addTwoNumbersNegative() { }
```

This test method has a similar structure to a `ddTwoNumbers()`: it is a `public` method, with no parameters, that returns `v oid`. It is annotated with `@ Test`, which indicates it is a single unit test.

Why not just add more assertions to a `ddTwoNumbers()`? Grouping more than one assertion into a single method can make your tests harder to debug if only one assertion fails, and obscures the tests that do succeed. The general rule for unit tests is to provide a test method for every individual assertion.

2. Run all tests in `C alculatorTest`, as before.

In the test window both `a ddTwoNumbers` and `a ddTwoNumbersNegative` are listed as available (and passing) tests in the left panel. The `a ddTwoNumbersNegative` test still passes even though it doesn't contain any code—a test that does nothing is still considered a successful test.

3. Add a line to a ddTwoNumbersNegative() to invoke the add() method in the Calculator class with a negative operand.

```
double resultAdd = mCalculator.add(-1d, 2d);
```

This work is licensed under a Creative Commons Attribution 4.0 International License. This PDF is one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Page 277

The d notation after each operand indicates that these are numbers of type double.
Because the add() method is defined with double parameters, float or int will also work.
Indicating the type explicitly enables you to test other types separately, if you need to.

4. Add an assertion with a assertThat().

```
assertThat(resultAdd, is(equalTo(1d)));
```

The assertThat() method is a JUnit4 assertion that claims the expression in the first argument is equal to the one in the second argument. Older versions of JUnit used more specific assertion methods (assertEquals(), assertEqualsNull(), or assertEquals()) , but assertThat() is a more flexible, more debuggable and often easier to read format.

The assertThat() method is used with *matchers*. Matchers are the chained method calls in the second operand of this assertion, i s(equalTo()). The Hamcrest framework defines the available matchers you can use to build an assertion. ("Hamcrest" is an anagram for "matchers.") Hamcrest provides many basic matchers for most basic assertions. You can also define your own custom matchers for more complex assertions.

In this case the assertion is that the result of the add() operation (-1 + 2) equals 1.

5. Add a new unit test to CalculatorTest for floating-point numbers:

*This work is licensed under a Creative Commons Attribution 4.0 International License.
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Page 278

```
@Test  
public void addTwoNumbersFloats() {  
    double resultAdd = mCalculator.add(1.111f, 1.111d);  
    assertThat(resultAdd, is(equalTo(2.222d))); }
```

Again, a very similar test to the previous test method, but with one argument to a `dd()` that is explicitly type `float` rather than `double`. The `dd()` method is defined with parameters of type `double`, so you can call it with a `float` type, and that number is promoted to a `double`. 6. Click **R un**  to run all the tests again.

creative

*Commons for the latest
updates.*

This time the test failed, and the progress bar is red. This is the important part of the error message:

```
java.lang.AssertionError:  
Expected: is <2.222>  
but: was <2.2219999418258665>
```

Arithmetic with floating-point numbers is inexact, and the promotion resulted in a side effect of additional precision. The assertion in the test is technically false: the expected value is not equal to the actual value.

The question this raises is: When you have a precision problem with promoting float arguments, is that a problem with your code, or a problem with your test? In this particular case both input arguments to the add() method from the SimpleCalc app will always be type double, so this is an arbitrary and unrealistic test. However, if your app was written such that the input to the add() method could be either double or float, and you only care about *some* precision, you need to provide some wiggle room to the test so that "close enough" counts as a success.

7. Change the assertThat() method to use the closeTo() matcher:

```
assertThat(resultAdd, is(closeTo(2.222, 0.01)));
```

You need to make a choice for the matcher. Click on **closeTo** twice (until the entire expression is underlined), and press Alt+Enter (Option+Return on a Mac). Choose **isCloseTo.closeTo (org.hamcrest.number)**.

8. Click **Run**  to run all the tests again.

This time the test passes.

With the closeTo() matcher, rather than testing for exact equality you can test for equality within a specific delta. In this case the closeTo() matcher method takes two arguments: the expected value and the amount of delta. In the example above, that delta is just two decimal points of precision.

This work is licensed under

Creative Commons Attribution 4.0 International License.

*This work is licensed under a C Attribution 4.0 International License.
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

for the latest updates.

2.2 Add unit tests for the other calculation methods

Use what you learned in the previous task to fill out the unit tests for the `Calculator` class.

1. Add a unit test called `subTwoNumbers()` that tests the `sub()` method.
2. Add a unit test called `subWorksWithNegativeResults()` that tests the `sub()` method where the given calculation results in a negative number.
3. Add a unit test called `mulTwoNumbers()` that tests the `mul()` method.
4. Add a unit test called `mulTwoNumbersZero()` that tests the `mul()` method with at least one argument as zero.
5. Add a unit test called `divTwoNumbers()` that tests the `div()` method with two non-zero arguments.
6. Add a unit test called `divTwoNumbersZero()` that tests the `div()` method with a double dividend and zero as the divisor.

All of these tests should pass, except `divTwoNumbersZero()` which causes an illegal argument exception for dividing by zero. If you run the app, enter zero as Operand 2, and click **D iv** to divide, the result is an error.

Task 2 solution code

Android Studio project: [SimpleCalcTest](#)

The following code snippet shows the tests for this task:

```
@Test
```

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

reative

*Commons for the latest
updates.*

```
public void addTwoNumbers() {
    double resultAdd = mCalculator.add(1d, 1d);
    assertThat(resultAdd, is(equalTo(2d)));
}
@Test public void addTwoNumbersNegative() {
    double resultAdd = mCalculator.add(-1d, 2d);
    assertThat(resultAdd, is(equalTo(1d)));
}
@Test
public void addTwoNumbersFloats() {
    double resultAdd = mCalculator.add(1.111f, 1.111d); assertThat(resultAdd, is(closeTo(2.222, 0.01)));
}
@Test public void subTwoNumbers() {
    double resultSub = mCalculator.sub(1d, 1d);
    assertThat(resultSub, is(equalTo(0d)));
}
@Test public void subWorksWithNegativeResult() {    double
resultSub = mCalculator.sub(1d, 17d);
    assertThat(resultSub, is(equalTo(-16d)));
}
@Test
public void mulTwoNumbers() {
    double resultMul = mCalculator.mul(32d, 2d);
    assertThat(resultMul, is(equalTo(64d)));
}
@Test public void
divTwoNumbers() {
    double resultDiv = mCalculator.div(32d,2d);
    assertThat(resultDiv, is(equalTo(16d)));
}
@Test
public void divTwoNumbersZero() {
    double resultDiv = mCalculator.div(32d,0);
    assertThat(resultDiv, is(equalTo(Double.POSITIVE_INFINITY)));
}
```

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

This work is licensed under

Creative Commons Attribution 4.0 International License.

for the latest updates.

Coding challenges

Note: All coding challenges are optional and are not prerequisites for later lessons.

Challenge 1: Dividing by zero is always worth testing for, because it is a special case in arithmetic. How might you change the app to more gracefully handle divide by zero? To accomplish this challenge, start with a test that shows what the right behavior should be.

Remove the `divideTwoNumbersZero()` method from `CalculatorTest`, and add a new unit test called `divideByZeroThrows()` that tests the `divide()` method with a second argument of zero, with the expected result as `IllegalArgumentException.class`. This test will pass, and as a result it will demonstrate that any division by zero will result in this exception.

After you learn how to write code for an [Exception](#) handler, your app can handle this exception gracefully by, for example, displaying a [Toast](#) message to the user to change Operand 2 from zero to another number.

Challenge 2: Sometimes it's difficult to isolate a unit of code from all of its external dependencies. Rather than organize your code in complicated ways just so you can test it more easily, you can use a mock framework to create fake ("mock") objects that pretend to be dependencies. Research the [Mockito](#) framework, and learn how to set it up in Android Studio. Write a test class for the `calcButton()` method in `SimpleCalc`, and use Mockito to simulate the Android context in which your tests will run.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#). This PDF is one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Page 282

Summary

Android Studio has built-in features for running local unit tests:

- Local unit tests use the JVM of your local machine. They don't use the Android framework.
- Unit tests are written with JUnit, a common unit testing framework for Java.
- JUnit tests are located in the (test) folder in the Android Studio **P**roject > **A**ndroid pane.
- Local unit tests only need these packages: org.junit, org.hamcrest, and android.test.
- The `@RunWith(JUnit4.class)` annotation tells the test runner to run tests in this class.
- `@SmallTest`, `@MediumTest`, and `@LargeTest` annotations are conventions that make it easier to bundle similar groups of tests
- The `@SmallTest` annotation indicates all the tests in a class are unit tests that have no dependencies and run in milliseconds.
- Instrumented tests are tests that run on an Android-powered device or emulator.
Instrumented tests have access to the Android framework.
- A test runner is a library or set of tools that enables testing to occur and the results to be printed to the log.

Related concept

The related concept documentation is in [3.2: App testing](#).

Learn more

Android Studio documentation:

- [Android Studio User Guide](#)
- [Write and View Logs](#)

a

for the latest updates.

*This work is licensed under [Creative Commons Attribution 4.0 International License](#).
Android Developer Fundamentals Course (V2) Unit*

Android developer documentation:

- [Best Practices for Testing](#)
- [Getting Started with Testing](#)
- [Building Local Unit Tests](#)

Other:

- [JUnit 4 Home Page](#) ● [JUnit 4 API Reference](#) ● [java.lang.Math](#)
- [Java Hamcrest](#)
- [Mockito Home Page](#)
- Video: [Android Testing Support - Testing Patterns](#)
- [Android Testing Codelab](#)
- [Android Tools Protip: Test Size Annotations](#)
- [The Benefits of Using assertThat over other Assert Methods in Unit Tests](#)

Homework

Build and run an app

Open the [SimpleCalc](#) app from the practical on using the debugger. You're going to add a **POW** button to the layout. The button calculates the first operand raised to the power of the second operand. For example, given operands of 5 and 4, the app calculates 5 raised to the power of 4, or 625.

Before you write the implementation of your power button, consider the kind of tests you might want to perform using this calculation. What unusual values may occur in this calculation?

1. Update the `Calculator` class in the app to include a `pow()` method. Hint: Consult the documentation for the [java.lang.Math](#) class.
2. Update the `MainActivity` class to connect the **POW** button to the calculation.

Now write each of the following tests for your `pow()` method. Run your test suite each time you write a test, and fix the original calculation in your app if necessary:

- A test with positive integer operands.

- A test with a negative integer as the first operand.
- A test with a negative integer as the second operand.
- A test with 0 as the first operand and a positive integer as the second operand.

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Page 284

- A test with 0 as the second operand.
- A test with 0 as the first operand and -1 as the second operand. (Hint: consult the documentation for Double.POSITIVE_INFINITY.)
- A test with -0 as the first operand and any negative number as the second operand.

Answer these questions

Question 1

Which statement best describes a local unit test? Choose one:

- Tests that run on an Android-powered device or emulator and have access to the Android framework.
- Tests that enable you to write automated UI test methods.
- Tests that are compiled and run entirely on your local machine with the Java Virtual Machine (JVM).

Question 2

Source sets are collections of related code. In which source set are you likely to find unit tests? Choose one:

- app/res
- com.example.android.SimpleCalcTest
- com.example.android.SimpleCalcTest (test)
- com.example.android.SimpleCalcTest (androidTest)

Question 3

Which annotation is used to mark a method as an actual test? Choose one:

- `@RunWith(JUnit4.class)`
- `@SmallTest`
- `@Before`
- `@Test`

This work is licensed under

Creative Commons Attribution 4.0 International License.

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Submit your app for grading

Guidance for graders

Check that the app has the following features:

- It displays a **P OW B utton** that provides an exponential ("power of") calculation.
- The implementation of **M ainActivity** includes a click handler for the **P OW B utton**.
- The implementation of **C alculator** includes a **p ow()** method that performs the calculation.
- The **C alculatorTest()** method includes separate test methods for the **p ow()** method in the **Calculator** class that perform tests for negative and 0 operands, and for the case of 0 and -1 as the operands.

Lesson 3.3: Support libraries

Introduction

The Android SDK includes the Android Support Library, which is a collection of several libraries.

These libraries provide features that aren't built into the Android framework, including the following:

- Backward-compatible versions of framework components, so that apps running on older versions of the Android platform can support features made available in newer versions of the platform
- Additional layout and user interface elements
- Support for different device form factors, such as TV devices or wearables
- Components to support Material Design elements
- Other features, including palette support, annotations, percentage-based layout dimensions, and preferences

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

What you should already know

You should be able to:

- Create an Android Studio project.
- Use the layout editor to work with EditText and Button elements.
- Build and run your app in Android Studio, on both an emulator and on a device.
- Navigate the Project > Android pane in Android Studio.
- Find the major components of an Android Studio project, including AndroidManifest.xml, resources, Java files, and Gradle files.

What you'll learn

- How to verify that the Android Support Library is available in your Android Studio installation.
- How to indicate support library classes in your app.
- How to tell the difference between the values for compileSdkVersion, targetSdkVersion, and minSdkVersion.
- How to recognize deprecated or unavailable APIs in your code.
- More about the Android support libraries.

What you'll do

- Create a new app with one TextView and one Button.
- Verify that the Android Support Repository (containing the Android Support Library) is available in your Android Studio installation. • Explore the build.gradle files for your app project.
- Manage class or method calls that are unavailable for the version of Android your app supports.
- Use a compatibility class from the support library to provide backward-compatibility for your app.

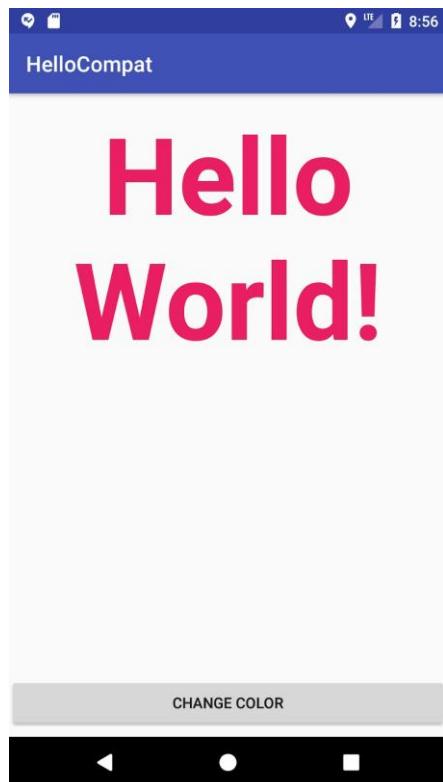
eat ve

ommons Attri Android

Developer
Fundamentals Course
(V2) Unit

App overview

In this practical you'll create an app called HelloCompat with one Text View that displays "Hello World" on the screen, and one Button that changes the color of the text. There are 20 possible colors, defined as resources in the color.xml file, and each button click randomly picks one of those colors.



The methods to get a color value from the app's resources have changed with different versions for the Android framework. This example uses the ContextCompat class in the Android Support Library, which allows you to use a method that works for all versions.

a

for the latest updates.

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Page 288

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.*

Page 293

Task 1: Set up your project to use support libraries

For this task you'll set up a new project for the HelloCompat app and implement the layout and basic behavior.

1.1 Verify that the Android Support Repository is available

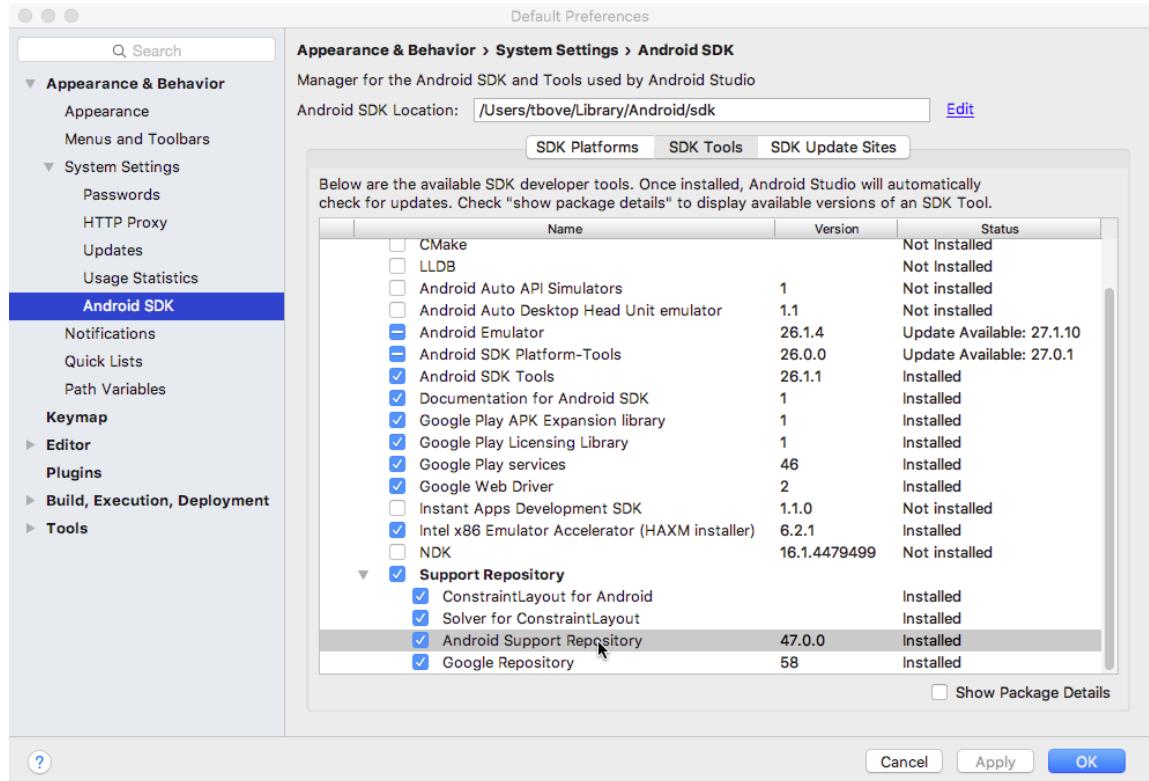
The Android Support Library is downloaded as part of the Android SDK, and available in the Android SDK manager. In Android Studio, you'll use the Android Support Repository—the local repository for the support libraries—to get access to the libraries from within your Gradle build files. In this task you'll verify that the Android Support Repository is downloaded and available for your projects.

15. In Android Studio, select **T ools > Android > SDK Manager**, or click the SDK Manager  icon.

The Android SDK **D efault Preferences** pane appears.

16. Click the **S DK Tools** tab and expand **S upport Repository**, as shown in the figure below.

eat ve
ommons Attr for the
latest updates.



17. Look for **Android **S**upport **R**epository in the list.**

If **I**nstalled appears in the Status column, you're all set. Click **C**ancel.

If **N**ot **i**nstalled or **U**pdate **A**vailable appears, click the checkbox next to **A**ndroid **S**upport **R**epository. A download icon should appear next to the checkbox. Click **O**K.

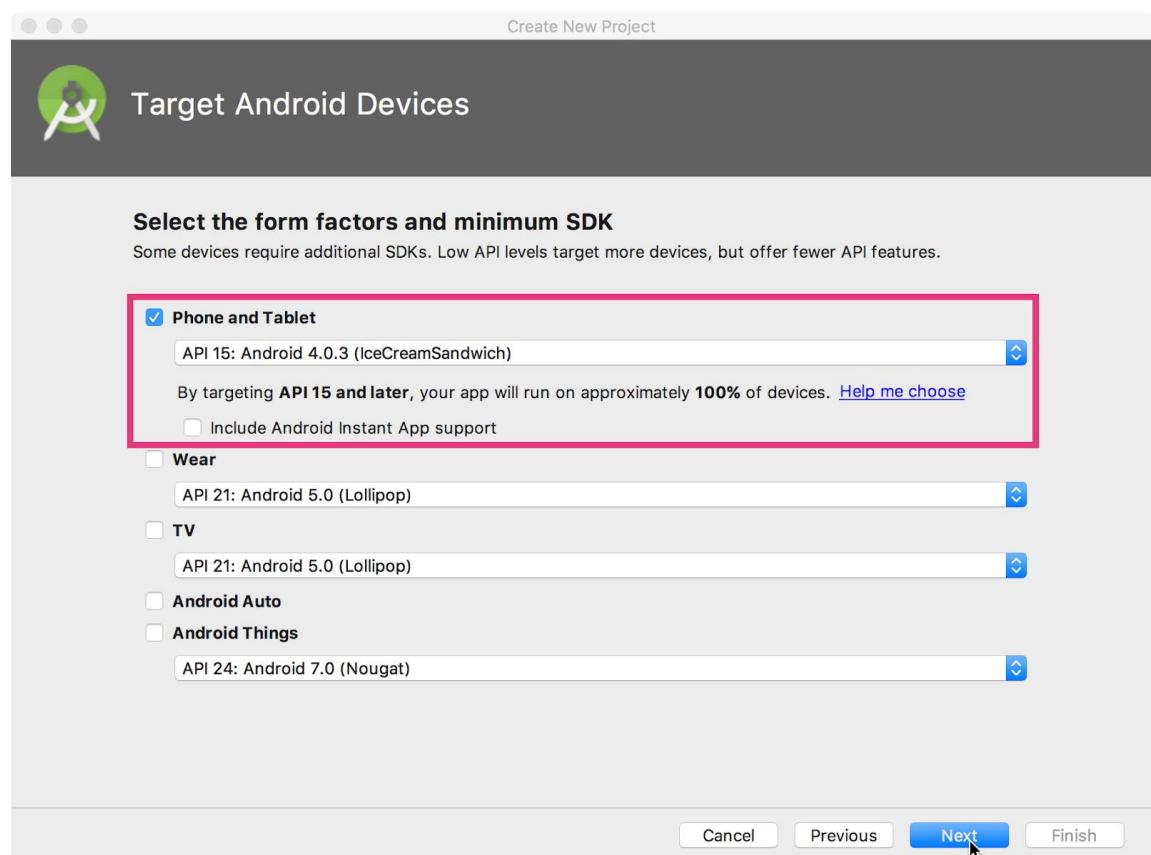
18. Click **OK again, and then **F**inish when the support repository has been installed.**

1.2 Set up the project and examine build.gradle

1. Create a new project called **HelloCompat**.

On the Target Android Devices page, **API 15: Android 4.0.3 (IceCreamSandwich)** is selected for the minimum SDK. As you've learned in previous lessons, this is the oldest version of the Android platform your app will support.

*This work is licensed under Creative Commons Attribution 4.0 International License.
Android Developer Fundamentals Course (V2) Unit*



1. Click **Next**, and choose the **Empty Activity** template.
2. Click **Next**, and ensure that the **Generate Layout file** and **Backwards Compatibility (App Compat)** options are checked. The latter option ensures that your app will be backwards-compatible with previous versions of Android.
3. Click **Finish**.

Explore build.gradle (Module:app)

1. In Android Studio, make sure the **Project > Android** pane is open.
2. Expand **Gradle Scripts** and open the **build.gradle (Module: app)** file.

*eat ve
ommons Attr for the
latest updates.*

Note that `build.gradle` for the overall project (`build.gradle (Project: helpcompat)`) is a different file from the `build.gradle` for the app module. In the `build.gradle (Module: app)` file.

3. Locate the `compileSdkVersion` line near the top of the file. For example:

```
compileSdkVersion 26
```

The `compile` version is the Android framework version your app is compiled with in Android Studio. For new projects the compile version is the most recent set of framework APIs you have installed. This value affects *only* Android Studio itself and the warnings or errors you get in Android Studio if you use older or newer APIs.

4. Locate the `minSdkVersion` line in the `defaultConfig` section a few lines down.

```
minSdkVersion 15
```

The `minimum` version is the oldest Android API version your app runs under. It's the same number you chose in Step 1 when you created your project. The Google Play store uses this number to make sure that your app can run on a given user's device. Android Studio also uses this number to warn you about using deprecated APIs.

5. Locate the `targetSdkVersion` line in the `defaultConfig` section. For example:

```
targetSdkVersion 26
```

The *target* version indicates the API version your app is designed and tested for. If the API of the Android platform is higher than this number (that is, your app is running on a newer device), the platform may enable compatibility behaviors to make sure that your app continues to work the way it was designed to. For example, Android 6.0 (API 23) provides a new runtime permissions model. If your app targets a lower API level, the platform falls back to the older install-time permissions model.

Although the target SDK can be the same number as the compile SDK, it is often a lower number that indicates the most recent version of the API for which you have tested your app.

This work is licensed under

Creative Commons Attribution 4.0 International License.

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

6. Locate the `dependencies` section of `build.gradle`, near the end of the file. For example:

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'com.android.support:appcompat-v7:26.1.0'  
    implementation  
        'com.android.support.constraint:constraint-layout:1.0.2'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.1'  
    androidTestImplementation  
        'com.android.support.test.espresso:espresso-core:3.0.1' }
```

The `dependencies` section for a new project includes several dependencies to enable testing with Espresso and JUnit, as well as the v7 appcompat support library. The version numbers for these libraries in your project may be different than those shown here.

The v7 appcompat support library provides backward-compatibility for older versions of Android all the way back to API 9. It includes the v4 compat library as well, so you don't need to add both as a dependency.

7. Update the version numbers, if necessary.

If the current version number for a library is lower than the currently available library version number, Android Studio highlights the line and warns you that a new version is available ("A newer version of `com.android.support:appcompat-v7` is available"). Edit the version number to the updated version.

Tip: You can also click anywhere in the highlight line and press Alt+Enter (Option+Return on a Mac). Select **Change to x.x.x.x** from the menu, where `x.x.x.x` is the most up-to-date version available.

8. Update the `compileSdkVersion` number, if necessary.

The major version number of the support library (the first number) must match your `compileSdkVersion`. When you update the support library version, you may also need to update `compileSdkVersion` to match.

9. Click **Sync Now** to sync your updated Gradle files with the project, if prompted.

creative

*Commons for the latest
updates.*

10. Install missing SDK platform files, if necessary.

If you update `compileSdkVersion`, you may need to install the SDK platform components to match. Click **I nstall missing platform(s) and sync project** to start this process.

Task 2: Implement the layout and MainActivity

For this task you'll implement the layout and basic behavior for the `MainActivity` class.

2.1 Change the layout and colors

In this task you will modify the `activity_main.xml` layout for the app.

1. Open a `activity_main.xml` in the **P**roject > **A**ndroid pane.
2. Click the **D**esign tab (if it is not already selected) to show the layout editor.
3. Select the "Hello World" `TextView` in the layout and open the **A**ttributes pane. 4. Change the `TextView` attributes as follows:

Attribute field	Enter the following:
ID	<code>hello_textview</code>
textStyle	B (bold)
textAlignment	Center the paragraph icon
textSize	100sp

This adds the `android:id` attribute to the `TextView` with the `id` set to `hello_textview`, changes the text alignment, makes the text bold, and sets a larger text size of `100sp`.

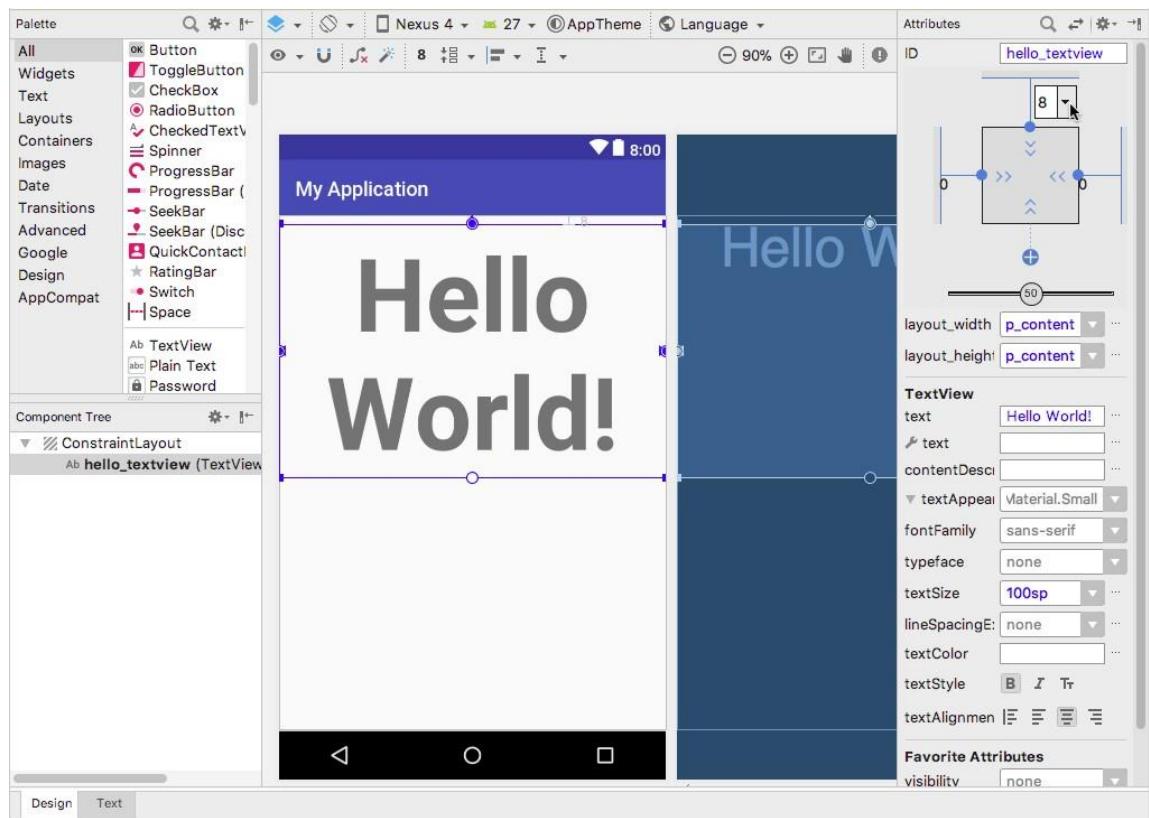
a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

This work is licensed under [Creative Commons Attribution 4.0 International License](#).

for the latest updates.

5. Delete the constraint that stretches from the bottom of the hello_textview TextView to the bottom of the layout, so that the TextView snaps to the top of the layout, and choose 8 (8dp) for the top margin as shown below.



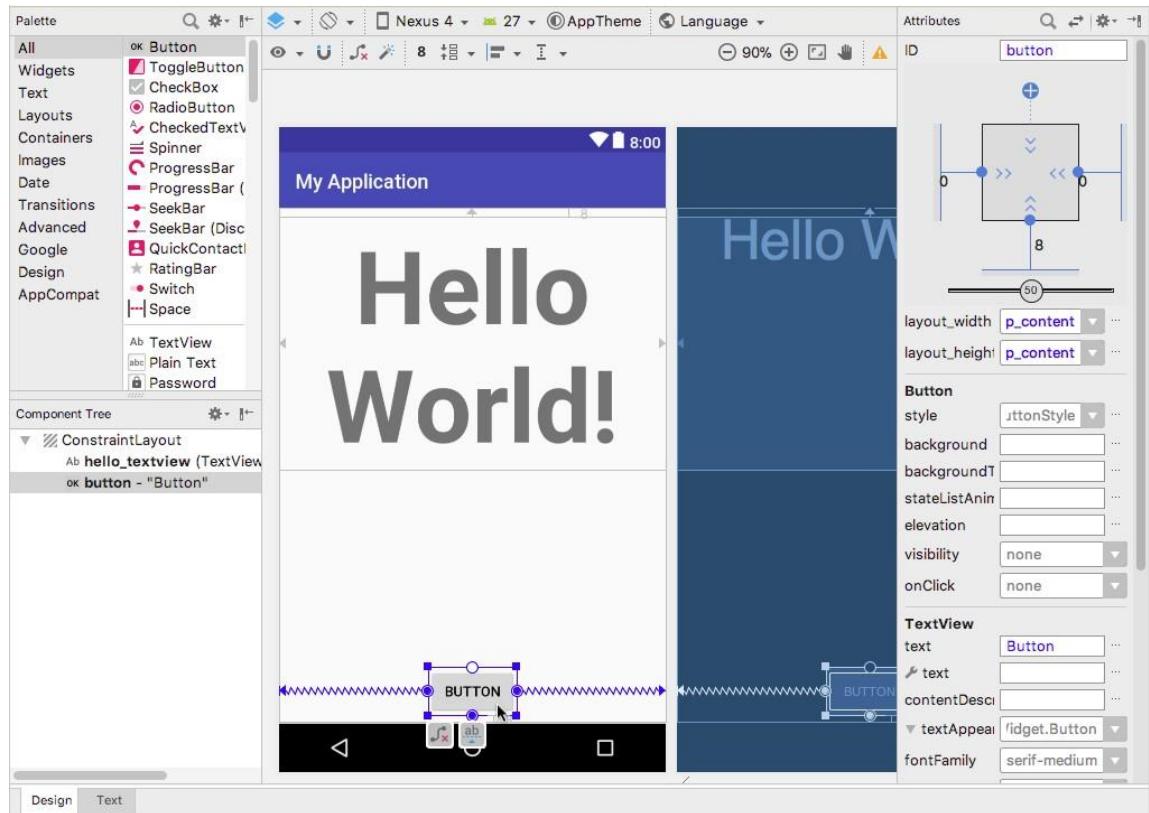
6. Drag a Button to the bottom of the layout, and add constraints to the left and right sides and bottom of the layout, as shown in the figure below.

*This work is licensed under a [Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

creativecommons.org/licenses/by/nd/4.0/
Creative Commons for the
latest updates.

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2



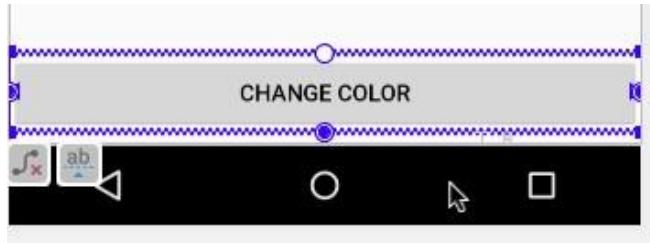
7. Change the **layout_width** attribute in the **Attributes** pane for the **Button** to **match_constraint**.
8. Change the other attributes in the **Attributes** pane for the **Button** as follows:

Attribute field	Enter the following:
ID	color_button
text	"Change Color"

The **Button** should now appear in the layout as shown below:

This work is licensed under a [Creative Commons Attribution 4.0 International License](#). This PDF is one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Android Developer Fundamentals Course (V2) Unit



9. In a previous lesson you learned how to extract a string resource from a literal text string. Click the **T ext** tab to switch to XML code, and extract the "Hello Text!" and "Change Color" strings in the **T extView** and **B utton**, and enter string resource names for them.
10. Add the following `android:onClick` attribute to the **B utton**:

```
android:onClick="changeColor"
```

11. To add colors, expand **r es** and **v alues** in the **P roject > Android** pane, and open **c olors.xml**.
12. Add the following color resources to the file:

```
<color name="red">#F44336</color>
<color name="pink">#E91E63</color>
<color name="purple">#9C27B0</color>
<color name="deep_purple">#673AB7</color>
<color name="indigo">#3F51B5</color>
<color name="blue">#2196F3</color>
<color name="light_blue">#03A9F4</color>
<color name="cyan">#00BCD4</color>
<color name="teal">#009688</color>
<color name="green">#4CAF50</color>
<color name="light_green">#8BC34A</color>
<color name="lime">#CDDC39</color>
<color name="yellow">#FFEB3B</color>
```

This work is licensed under a [Creative Commons Attribution 4.0 International License](#). This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

```
<color name="amber">#FFC107</color>
<color name="orange">#FF9800</color>
<color name="deep_orange">#FF5722</color>
<color name="brown">#795548</color>
<color name="grey">#9E9E9E</color>
<color name="blue_grey">#607D8B</color> <color name="black">#000000</color>
```

creative

Commons Android

Developer

Fundamentals

Course (V2) Unit

These color values and names come from the recommended color palettes for Android apps defined at [Material Design - Style - Color](#). The codes indicate color RGB values in hexadecimal.

2.2 Add behavior to MainActivity

In this task you'll finish setting up the project by adding private variables and implementing onCreate() and onSaveInstanceState().

1. Open **MainActivity**.
2. Add a private variable at the top of the class to hold the TextView object.

```
private TextView mHelloTextView;
```

3. Add the following color array just after the private variable:

a

for the latest updates.

```
private String[] mColorArray = {"red", "pink", "purple", "deep_purple", "indigo", "blue", "light_blue", "cyan",  
    "teal", "green",  
    "light_green", "lime", "yellow", "amber", "orange", "deep_orange",  
    "brown", "grey", "blue_grey", "black" };
```

Each color name corresponds to the name of a color resource in color.xml.

4. In the onCreate() method, use findViewById() to get a reference to the TextView instance and assign it to that private variable:

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

```
mHelloTextView = findViewById(R.id.hello_textview);
```

1. Also in `onCreate()`, restore the saved instance state, if any:

```
// restore saved instance state (the text color)
if (savedInstanceState != null) {
    mHelloTextView.setTextColor(savedInstanceState.getInt("color")); }
```

2. Add the `onSaveInstanceState()` method to `MainActivity` to save the text color:

```
@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    // save the current text color    outState.putInt("color", mHelloTextView.getCurrentTextColor()); }
```

Task 2 solution code

The following is the solution code for the XML layout and a code snippet in the `MainActivity` class for the `HelloCompat` app so far.

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Page 299

Android Developer Fundamentals Course (V2) Unit

XML layout

The XML layout for the `activity_main.xml` file is shown below. The `changeColor` click handler for the `Button` is underlined in red because it hasn't yet been defined. You define it in the next task.

```

<android.support.constraint.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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.myapplication.MainActivity">

    <TextView
        android:id="@+id/hello_textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/hello_world"
        android:textAlignment="center"
        android:textSize="100sp"
        android:textStyle="bold"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/color_button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:text="@string/change_color"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"      app:layout_constraintRight_toRightOf="parent"
        android:onClick="changeColor"/>
</android.support.constraint.ConstraintLayout>

```

MainActivity

The MainActivity class includes the following private variables at the top of the class:

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
 This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

```
// Text view for Hello World. private TextView
mHelloTextView;
// array of color names, these match the color resources in color.xml private String[] mColorArray =
{"red", "pink", "purple", "deep_purple",
 "indigo", "blue", "light_blue", "cyan", "teal", "green",
 "light_green", "lime", "yellow", "amber", "orange", "deep_orange",
 "brown", "grey", "blue_grey", "black"};
```

The MainActivity class includes the following onCreate() and onSaveInstanceState() methods:

```
@Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mHelloTextView = findViewById(R.id.hello_textview);
    // restore saved instance state (the text color)
    if (savedInstanceState != null) {
        mHelloTextView.setTextColor(savedInstanceState.getInt("color"));
    }
}

@Override public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    // save the current text color
    outState.putInt("color", mHelloTextView.getCurrentTextColor()); }
```

Task 3: Implement Button behavior

The **C hange C olor** button in the HelloCompat app picks one of the 20 colors from the `color.xml` resource file at random and sets the color of the text to that color. In this task you'll implement the behavior for `B utton` click handler.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#). This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Page 301

Android Developer Fundamentals Course (V2) Unit

2.1 Add the changeButton() click handler

1. Open `a ctivity_main.xml`, if it is not already open. Click the `T ext` tab to show the XML code.
2. Click on "changeColor" in the `android:onClick` attribute inside the `B utton` element.
3. Press `Alt+Enter` (`Option+Enter` on a Mac), and select **C reate onClick event handler**.
4. Choose **M ainActivity** and click **O K**.

This creates a placeholder method stub for the `c hangeColor()` method in `M ainActivity`:

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

2.2 Implement the Button action

1. Switch to `M ainActivity`.
2. In the `c hangeColor()` method, create a random number object by using the `R andom` class (a Java class) to generate simple random numbers.

```
Random random = new Random();
```

a

for the latest updates.

3. Us

the ran doinst ance to pick a random color from the m ColorArray array:

```
String colorName = mColorArray[random.nextInt(20)];
```

The `nextInt()` method with the argument 20 gets another random integer between 0 and 19. You use that integer as the index of the array to get a color name.

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

4. Get the resource identifier (an integer) for the color name from the resources:

```
int colorResourceName = getResources().getIdentifier(colorName, "color",  
getApplicationContext().getPackageName());
```

When your app is compiled, the Android system converts the definitions in your XML files into resources with internal integer IDs. There are separate IDs for both the names and the values. This line matches the color strings from the `colorName` array with the corresponding color name IDs in the XML resource file. The `getResources()` method gets all the resources for your app. The `getIdentifier()` method looks up the color name (the string) in the color resources ("color") for the current package name.

5. Get the integer ID for the actual color from the resources and assign it to a `colorRes` variable, and use the `getTheme()` method to get the theme for the current application context.

```
int colorRes =  
    getResources().getColor(colorResourceName, this.getTheme());
```

The `getResources()` method gets the set of resources for your app, and the `getColor()` method retrieves a specific color from those resources by the ID of the color name. However, `getColor()` has a red underlined highlight.

If you point at `getColor()`, Android Studio reports: "Call requires API 23 (current min is 15)". Because your `minSdkVersion` is 15, you get this message if you try to use any APIs that were introduced after API 15. You can still compile your app, but because this version of `getColor()` is not available on devices prior to API 23, your app will crash when the user taps the **C hange Color** button.

At this stage you could check for the platform version and use the right version of `getColor()` depending on where the app is running. A better way to support both older and newer Android APIs without warnings is to use one of the compatibility classes in the support library.

6. Change the `colorRes` assignment line to use the `ContextCompat` class:

*reative
Commons for the latest
updates.*

Android Developer Fundamentals Course (V2) Unit

```
int colorRes = ContextCompat.getColor(this, colorResourceName);
```

`ContextCompat` provides many compatibility methods to address API differences in the application context and app resources. The `getColor()` method in `ContextCompat` takes two arguments: the current context (here, the `Activity` instance, `this`), and the name of the color.

The implementation of this method in the support library hides the implementation differences in different versions of the API. You can call this method regardless of your compile SDK or minimum SDK versions with no warnings, errors, or crashes.

1. Set the color of the `TextView` to the color resource ID:

```
mHelloTextView.setTextColor(colorRes);
```

2. Run the app on a device or emulator, and click the **Change Color** button.

The **Change Color** button should now change the color of the text in the app, as shown below.

*This work is licensed under a Creative Commons Attribution 4.0 International License.
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Page 304



Solution code

MainActivity solution

The following is the `changeColor()` click handler in `MainActivity`:

*This work is licensed under a Creative Commons Attribution 4.0 International License.
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Page 305

```
/**  
 * This method handles the click of the Change Color button by  
 * picking a random color from a color array.  
 * @param view The view that was clicked
```

*creativecommons.org for the
latest updates.*

*a
for the latest updates.*

```
/* public void changeColor(View view) {  
    // Get a random color name from the color array (20 colors).  
    Random random = new Random();  
    String colorName = mColorArray[random.nextInt(20)];  
    // Get the color identifier that matches the color name.  
    int colorResourceName = getResources().getIdentifier(colorName,  
        "color", getApplicationContext().getPackageName());  
  
    // Get the color ID from the resources.  
    int colorRes = ContextCompat.getColor(this, colorResourceName);  
    // Set the text color.  
    mHelloTextView.setTextColor(colorRes); }
```

Android Studio project

Android Studio project: [HelloCompat](#)

Coding challenge

Note: All coding challenges are optional and are not prerequisites for later lessons.

Challenge: Rather than using `ContextCompat` for to get the color resource, use a test of the values in the `Build` class to perform a different operation if the app is running on a device that supports a version of Android older than API 23.

Summary

Installing the Android Support Library:

- Use the SDK Manager to install the Android Support Repository. Choose **T ools > Android > SDK Manager**, click the **S DK Tools** tab, and expand **S upport Repository**.

This work is licensed under [Creative Commons Attribution 4.0 International License](#).

- If **I nstalled** appears in the Status column for **A ndroid Support Repository**, click **C ancel**; if **N ot installed** or **U pdate Available** appears, click the checkbox. A download icon should appear next to the checkbox. Click **O K**.

Android uses three directives to indicate how your app should behave for different API versions:

- `minSdkVersion`: the minimum API version your app supports.
- `compileSdkVersion`: the API version your app should be compiled with.
- `targetSdkVersion`: the API version your app was designed for.

To manage dependencies in your project:

- Expand **G radle Scripts** in the **P roject > Android** pane, and open the **b uild.gradle (Module: app)** file.
- You can add dependencies in the `d ependencies` section.

The `C ontextCompat` class provides methods for compatibility with context and resource-related methods for both old and new API levels.

Related concept

The related concept documentation is in [3 .3: The Android Support Library](#).

Learn more

Android Studio documentation:

- [Android Studio User Guide](#)

Android developer documentation:

*creative
Commons Android
Developer
Fundamentals
Course (V2) Unit*

- [Android Support Library](#) (introduction)
- [Support Library Setup](#)
- [Support Library Features](#)
- [Supporting Different Platform Versions](#)
- [Package Index](#) (all API packages that start with android.support)

Other:

- [Picking your compileSdkVersion, minSdkVersion, and targetSdkVersion](#)
- [Understanding the Android Support Library](#)
- [All the Things Compat](#)

Homework

Run an app

Open the [HelloCompat](#) app you created in the practical on using support libraries.

1. Set a debugger breakpoint on the line in the `changeColor()` method that actually changes the color:

```
int colorRes = ContextCompat.getColor(this, colorResourceName);
```

2. Run the app in debug mode on a device or emulator that's running an API version 23 or newer. Click **S tep Into** to step into the `getColor()` method and follow the method calls deeper into the stack. Examine how the

`ContextCompat` class determines how to get the color from the resources, and which other framework classes it uses.

Some classes may produce a warning that the "source code does not match the bytecode." Click **Step Out** to return to a known source file, or keep clicking **Step Into** until the debugger returns on its own.

3. Repeat the previous step for a device or emulator running an API version older than 23. Note the different paths that the framework takes to accomplish getting the color.

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Answer these questions

Question 1

Which class appears when you first **Step Into** the `ContextCompat.getColor()` method? Choose one:

- `MainActivity`
- `ContextCompat`
- `AppCompatActivity`
- `Context`

Question 2

In the class that appears, which statement is executed if the build version is API version 23 or newer? Choose one:

- `return context.getColor(id);`
- `return context.getResources().getColor(id);`
- `throw new IllegalArgumentException("permission is null");`
- `return mResources == null ? super.getResources() : mResources;`

Question 3

If you change the `ContextCompat.getColor()` method back to the `getColor()` method, what will happen when you run the app? Choose one:

- If your `m_inSdkVersion` is 15, the word `getColor` is underlined in red in the code editor. Hover your pointer over it, and Android Studio reports "Call requires API 23 (current min is 15)".
- The app will run without error on emulators and devices using API 23 or newer.
- The app will crash when the user taps **C hange Color** if the emulator or device is using API 17.
- All of the above.

*creative Commons
Android Developer Fundamentals Course (V2) Unit*

Submit your app for grading

Guidance for graders

No app to submit for this homework assignment.

End of Unit

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

a

for the latest updates.