

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



GIÁO TRÌNH
THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1. LÀM QUEN	1
Bài 1) Tạo ứng dụng đầu tiên.....	1
1.1) Android Studio và Hello World.....	1
1.2) Giao diện người dùng tương tác đầu tiên.....	17
1.3) Trình chỉnh sửa bộ cục	42
1.4) Văn bản và các chế độ cuộn.....	67
1.5) Tài nguyên có sẵn.....	82
Bài 2) Activities	90
2.1) Activity và Intent.....	90
2.2) Vòng đời của Activity và trạng thái	112
2.3) Intent ngầm định.....	122
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	136
3.1) Trình gỡ lỗi	136
3.2) Kiểm thử đơn vị	146
3.3) Thư viện hỗ trợ.....	146
CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG	147
Bài 1) Tương tác người dùng	147
1.1) Hình ảnh có thể chọn	147
1.2) Các điều khiển nhập liệu	147
1.3) Menu và bộ chọn	147
1.4) Điều hướng người dùng	147
1.5) RecyclerView	147
Bài 2) Trải nghiệm người dùng thú vị	147
2.1) Hình vẽ, định kiểu và chủ đề.....	147
2.2) Thẻ và màu sắc.....	147
2.3) Bộ cục thích ứng	147

Bài 3) Kiểm thử giao diện người dùng.....	147
3.1) Espresso cho việc kiểm tra UI.....	147
CHƯƠNG 3. LÀM VIỆC TRONG NỀN	147
Bài 1) Các tác vụ nền.....	147
1.1) AsyncTask	147
1.2) AsyncTask và AsyncTaskLoader	147
1.3) Broadcast receivers	147
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền	147
2.1) Thông báo.....	147
2.2) Trình quản lý cảnh báo.....	147
2.3) JobScheduler.....	147
CHƯƠNG 4. LUU DỮ LIỆU NGƯỜI DÙNG	148
Bài 1) Tùy chọn và cài đặt.....	148
1.1) Shared preferences.....	148
1.2) Cài đặt ứng dụng	148
Bài 2) Lưu trữ dữ liệu với Room.....	148
2.1) Room, LiveData và ViewModel	148
2.2) Room, LiveData và ViewModel	148

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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

Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

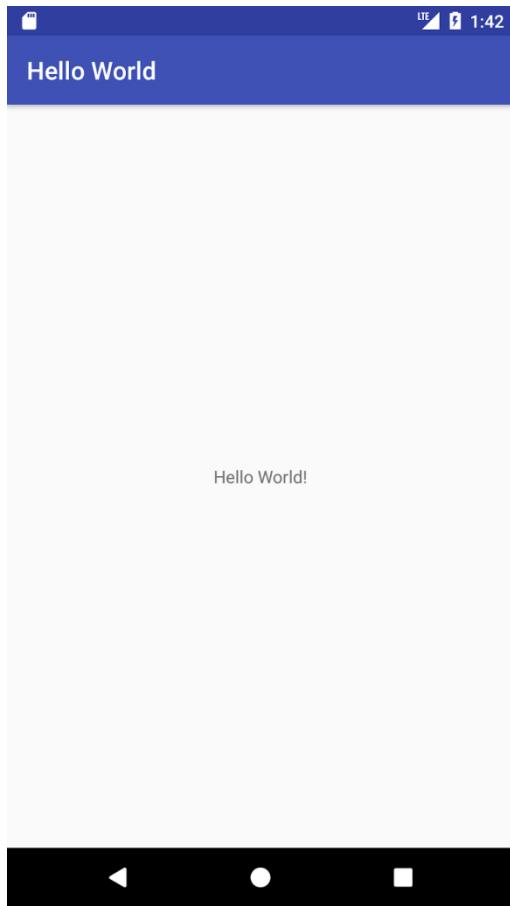
- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.
- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.

- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

Tổng quan ứng dụng

Sau khi cài đặt thành công Android Studio, bạn sẽ tạo 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 máy ảo Android hoặc thiết bị vật lý.

Ứng dụng hoàn thiện sẽ như thế này:



Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp hoàn chỉnh (IDE) bao gồm một trình soạn thảo mã nâng cao và một tập các mẫu ứng dụng. Ngoài ra, nó chứa các công cụ cho phát triển, gỡ lỗi, kiểm thử, và hiệu suất giúp phát triển ứng dụng nhanh và hiệu quả hơn. Bạn cũng có thể thử nghiệm ứng dụng của bạn với một loạt các trình giả lập được cấu hình sẵn hoặc trên thiết bị di động của bạn, xây dựng ứng dụng sản xuất và xuất bản trên cửa hàng Google Play.

Android Studio có sẵn cho các máy tính chạy Windows hoặc Linux, và cho máy Macs chạy MacOS. OpenJDK (Java Development Kit) mới nhất được tích hợp sẵn trong Android Studio.

Để bắt đầu và chạy Android Studio, đầu tiên kiểm tra yêu cầu hệ thống để đảm bảo rằng hệ thống của bạn đáp ứng chúng. Việc cài đặt tương tự nhau trên mọi nền tảng. Bất kỳ sự khác biệt nào đã được ghi chú ở bên dưới.

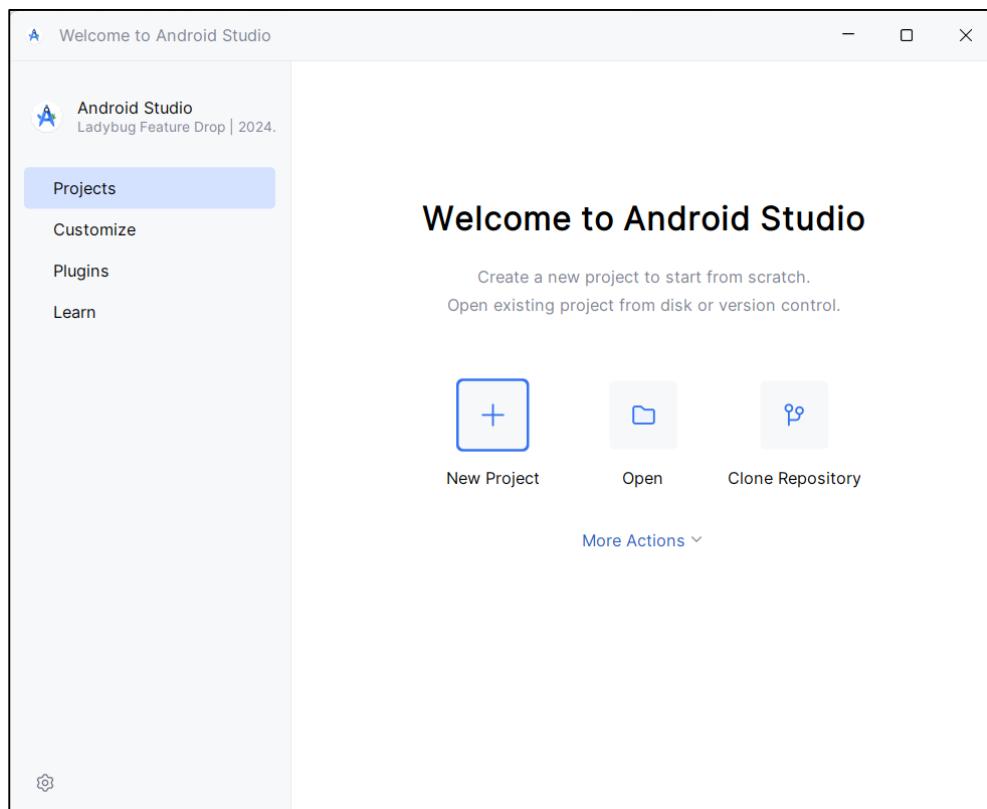
1. Truy cập trang web dành cho nhà phát triển Android và làm theo hướng dẫn để tải và cài đặt Android Studio.
2. Chấp nhận cấu hình mặc định cho tất cả các bước, và đảm bảo rằng các phần đã được chọn để cài đặt.
3. Sau khi kết thúc cài đặt, trình hướng dẫn cài đặt sẽ tải và cài đặt một số thành phần bổ sung bao gồm Android SDK. Hãy kiên nhẫn, việc này có thể mất thời gian tùy vào tốc độ internet của bạn, và một số bước có vẻ thưa thớt.
4. Khi hoàn tất tải xuống, Android Studio sẽ khởi động, và bạn sẵn sàng để tạo dự án đầu tiên của bạn.

Nhiệm vụ 2: Tạo ứng dụng Hello World

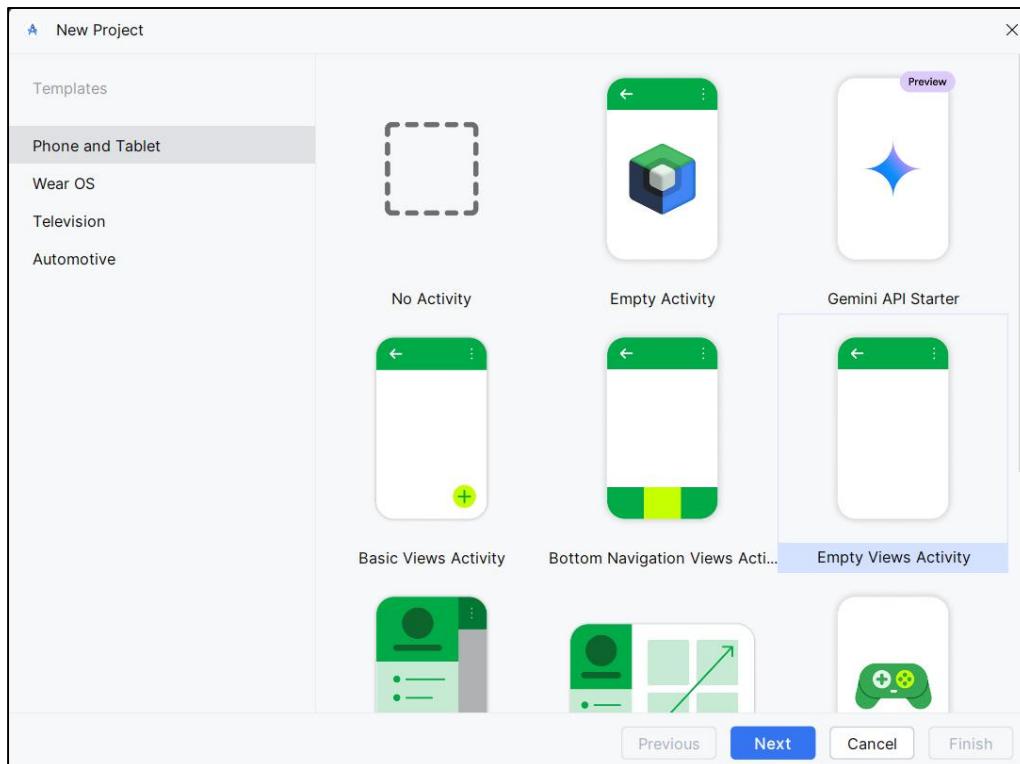
Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị “Hello World” để xác minh rằng Android Studio đã được cài đặt đúng, và để tìm hiểu cơ bản về phát triển với Android Studio.

Tạo dự án

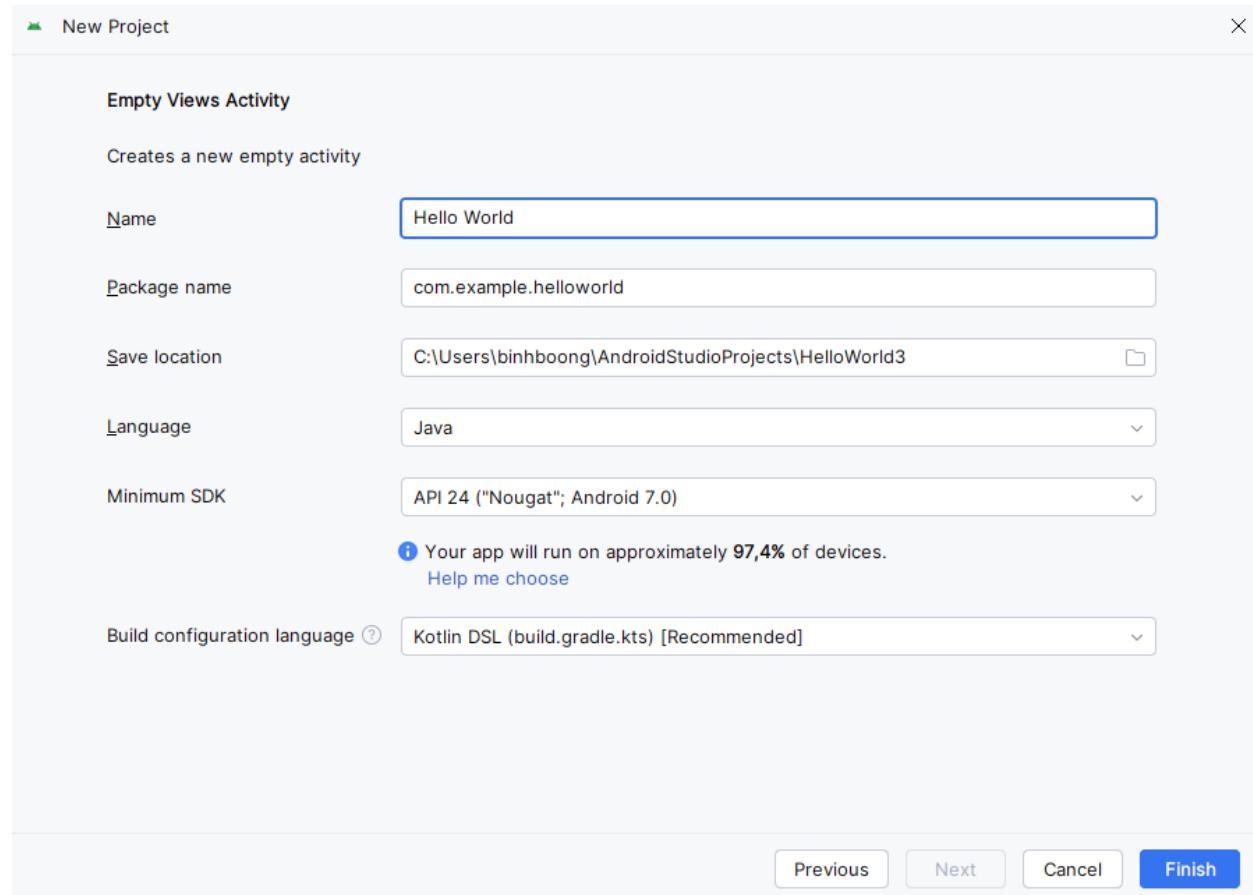
1. Mở Android Studio nếu nó chưa được mở.
2. Trong cửa sổ **Welcom to Android Studio**, chọn **New Project**.



3. Trong cửa sổ **New Project**, chọn **Phone and Tablet**, chọn **Empty Views Activity**. Tiếp theo nhấp **Next**.



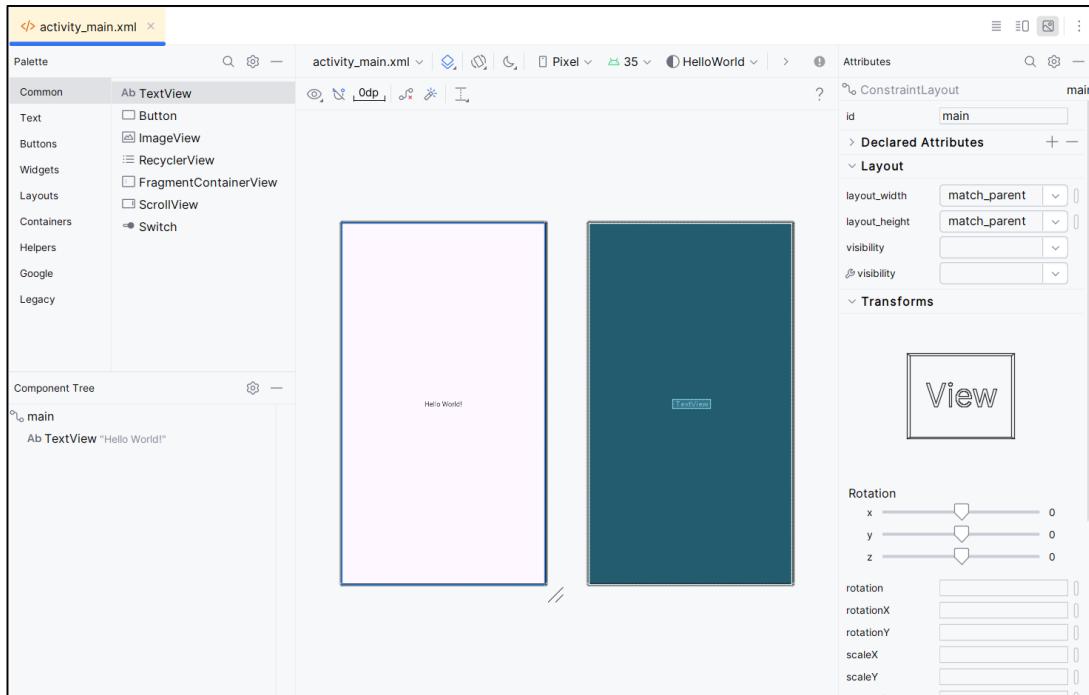
4. Nhập các thông tin cho ứng dụng. Tên ứng dụng (**Name**) là **Hello World**, tên gói (**Package name**) là **com.dvgiang.helloworld**. Chọn vị trí lưu thư mục dự án (**Save location**), ngôn ngữ (**Language**) ở đây là **Java**, phiên bản hệ điều hành Android tối thiểu mà ứng dụng có thể chạy (**Minimum SDK**) ở đây là **Android 7.0**. Tiếp theo nhấn **Finish** để tạo dự án.



Android Studio sẽ tạo một thư mục chứa dự án của bạn, và xây dựng dự án với Gradle.

Trình chỉnh sửa Android Studio xuất hiện, làm theo các bước sau:

1. Nhấn vào tab **activity_main.xml** để xem trình chỉnh sửa bô cục.



2. Nhấn tab **MainActivity.java** để xem trình chỉnh sửa mã.

```

1 package com.example.helloworld;
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);
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 }

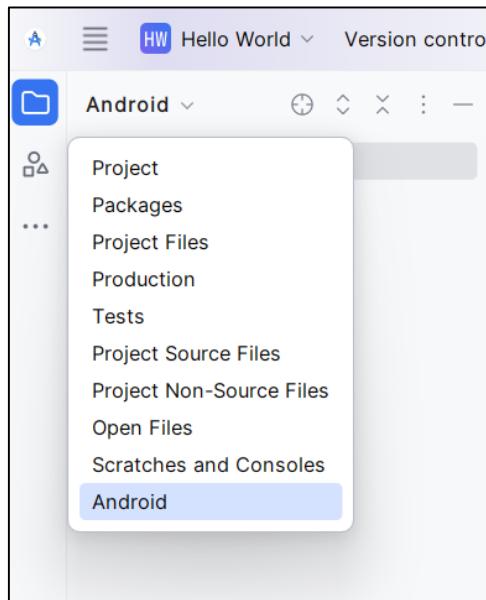
```

Khám phá dự án > Khung Android

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

1. Nếu chưa được chọn, nhấn tab **Project** trong cột tab dọc ở phía trên bên trái của cửa sổ Android Studio. Khung Project sẽ xuất hiện.

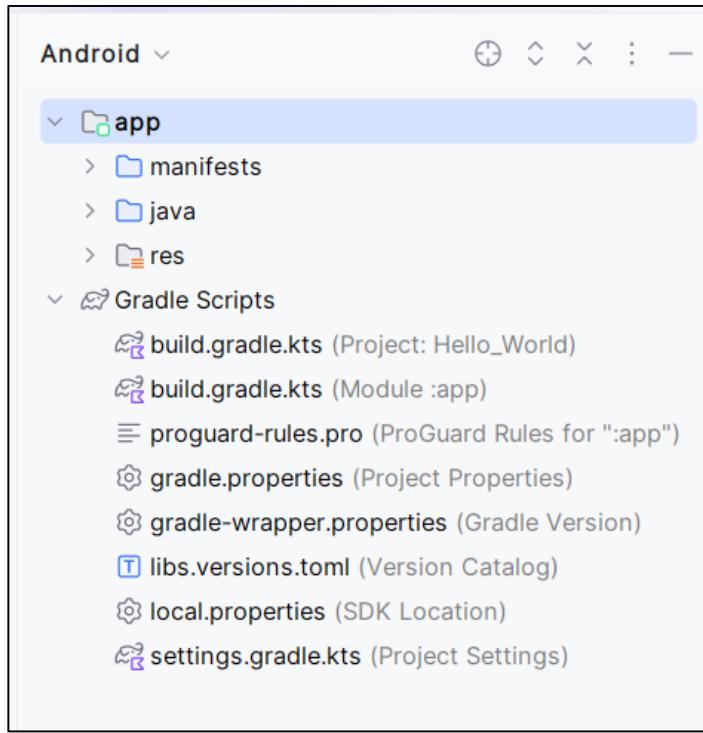
2. Để xem dự án trong hệ thống phân cấp dự án Adroid chuẩn, chọn Android từ menu hiện lên.



Khám phá thư mục Gradle Script

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

Khi bạn tạo dự án đầu tiên, **Project > Android** sẽ xuất hiện với thư mục **Gradle Scripts** như ảnh bên dưới.



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

1. Nếu thư mục **Gradle Scripts** không được mở rộng, nhấn vào hình tam giác để mở nó. Thư mục này chứa tất cả các tệp cần thiết chi hệ thống xây dựng.
2. Tìm đến tệp **build.gradle.kts(Project: Hello_World)**

Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô-đun tạo nên dự án của bạn. Mỗi dự án Android Studio đều chứa một tệp Gradle build cấp cao nhất. Hầu hết, bạn sẽ không cần thay đổi bất cứ gì trên tệp này, nhưng vẫn hữu ích khi hiểu nội dung của nó.

Mặc định, tệp xây dựng cấp cao nhất sử dụng khái niệm **buildscript** để xác định kho lưu trữ Gradle và các phụ thuộc chung cho tất cả các mô-đun trong dự án. Khi phụ thuộc của bạn khác với thư viện cục bộ hoặc cây tập tin, Gradle tìm kiếm các tệp trong bất kỳ kho lưu trữ trực tuyến nào được chỉ định trong khái niệm kho lưu trữ của tệp này.

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

Ngoài tệp **build.gradle.kts** mức dự án, mỗi mô-đun có một tệp **build.gradle.kts** của nó, cho phép bạn cài đặt cấu hình bản dựng cho mỗi mô-đun cụ thể (ứng dụng Hello World chỉ có một mô-đun). Cấu hình các cài đặt bản dựng này cho phép bạn cung cấp các tùy chọn đóng gói tùy chỉnh, như các loại xây dựng bổ sung. Bạn cũng có thể ghi đè các cài đặt trong tệp **AndroidManifest.xml** hoặc **build.gradle.kts** cấp cao nhất.

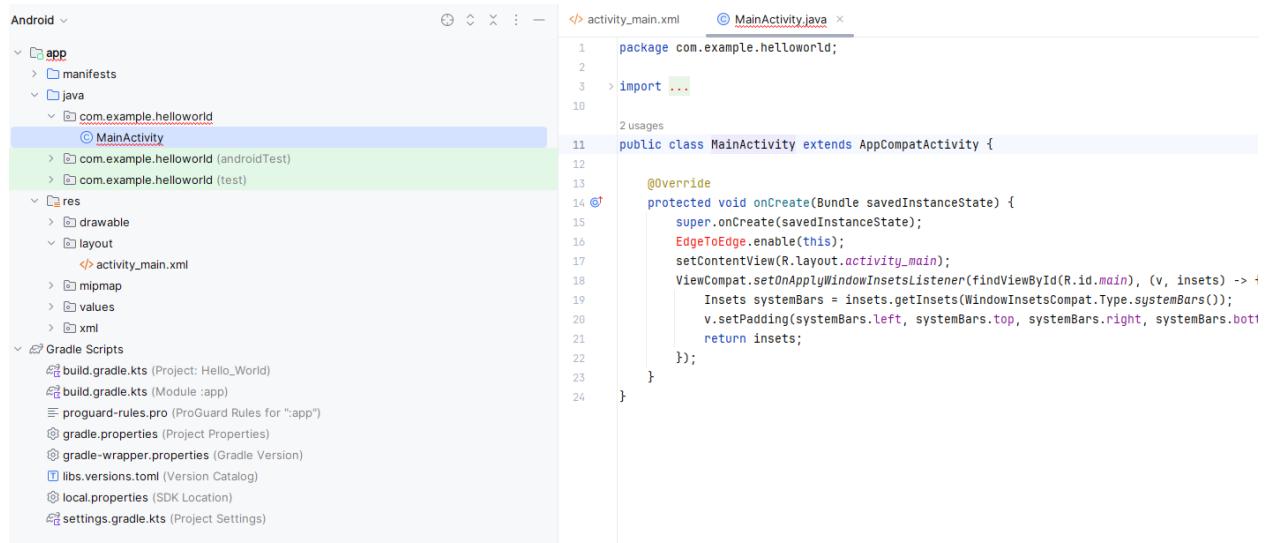
Tệp này thường phổ biến để chỉnh sửa khi thay đổi các cấu hình mức ứng dụng, như khai báo các phụ thuộc trong phần **dependencies**. Bạn cũng có thể khai báo thư viện phụ thuộc sử dụng một trong nhiều cấu hình phụ thuộc khác nhau. Mỗi cấu hình cung cấp cho Gradle các chỉ dẫn khác nhau cách sử dụng thư viện.

4. Nhấn vào dấu tam giác để đóng **Gradle Scripts**

Khám phá thư mục app và res

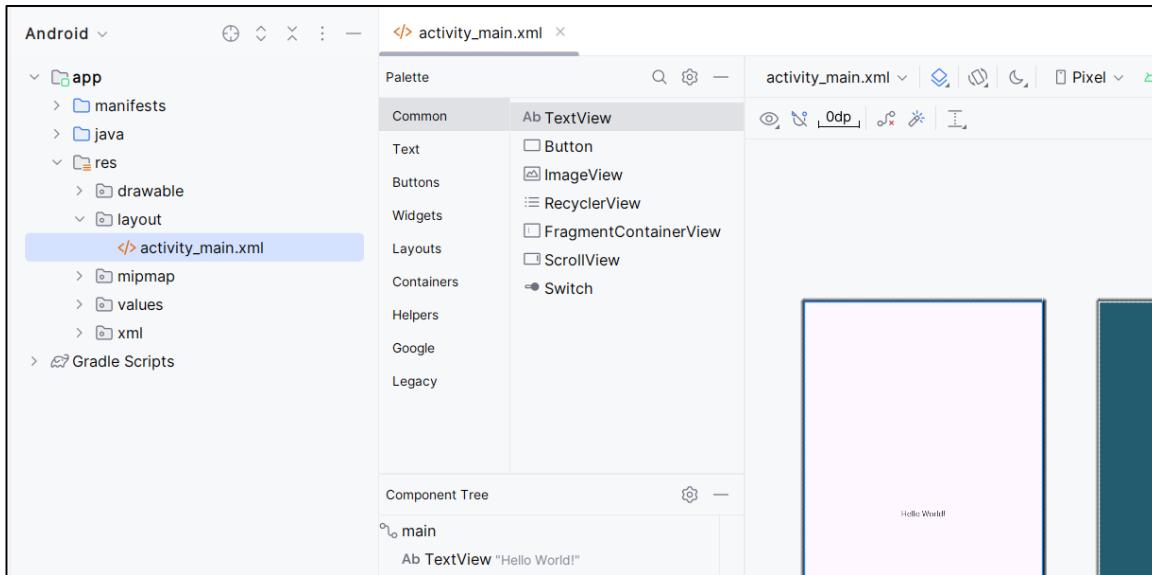
Tất cả mã và tài nguyên cho ứng dụng nằm trong thư mục **app** và **res**.

1. Mở rộng thư mục **app**, **java**, và **com.dvgiang.helloworld** để thấy tệp **java** **MainActivity**, nhấn đúp chuột để mở tệp trong trình chỉnh sửa mã.



Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như thể hiện trong hình trên. Thư mục **com.dvgiang.helloworld** (hoặc tên bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục còn lại được sử dụng để thử nghiệm. Đối với ứng dụng Hello World, chỉ có một gói chứa **MainActivity.java**. Tên của **Activity(screen)** đầu tiên mà người dùng nhìn thấy, cũng khởi tạo các tài nguyên trên toàn ứng dụng, thường được gọi là **MainActivity**.

2. Mở rộng thư mục **res**, **layout**, nhấn đúp chuột vào tệp **activity_main.xml** để mở trong trình chỉnh sửa bối cảnh.



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

Khám phá thư mục **manifests**

Thư mục **manifests** chứa các tệp cung cấp thông tin cần thiết về ứng dụng của bạn cho hệ thống Android, hệ thống phải có thông tin này trước khi có thể chạy bất kỳ mã nào của ứng dụng.

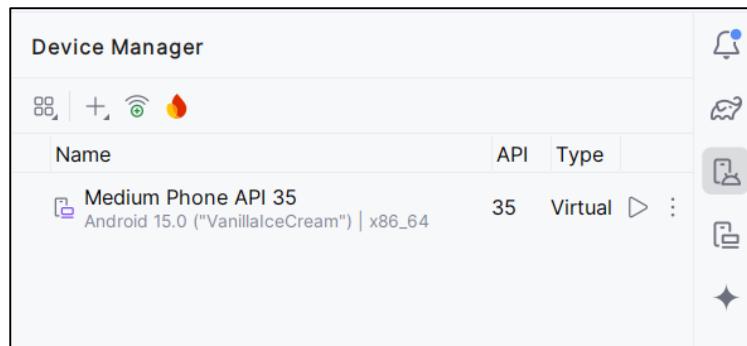
1. Mở rộng thư mục **manifests**.
2. Mở tệp **AndroidManifest.xml**.

Tệp **AndroidManifest.xml** mô tả tất cả các thành phần của ứng dụng Android của bạn. Tất cả các thành phần cho một ứng dụng, như mỗi Activity, phải được khai báo trong tệp XML này.

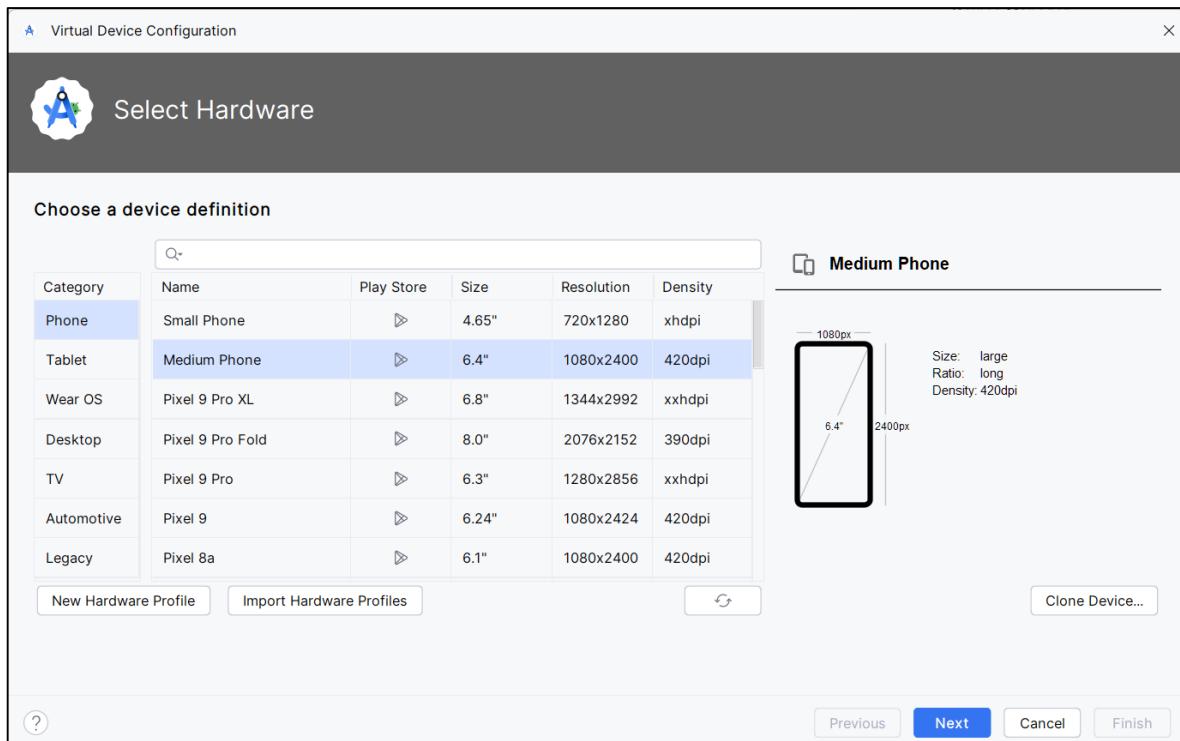
Nhiệm vụ 3: Tạo một thiết bị Android ảo (AVD)

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

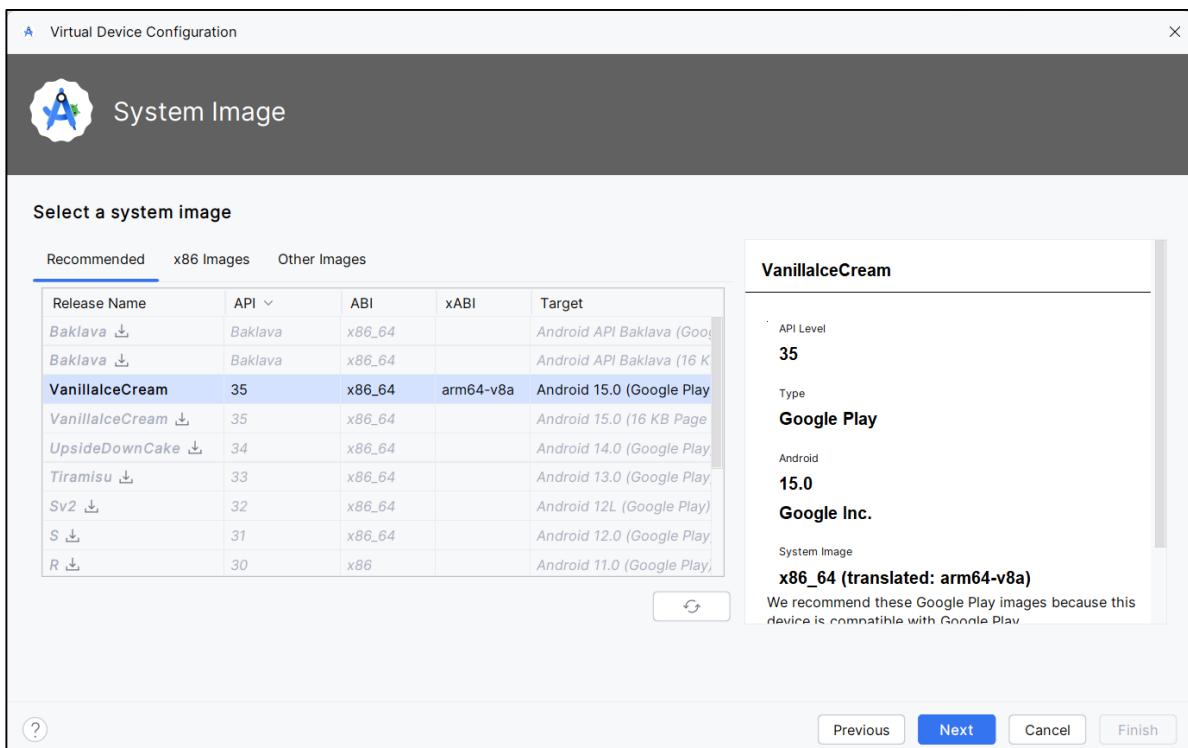
1. Trong Android Studio, chọn **Tools > Device Manager**, hoặc chọn vào biểu tượng Device Manager () trên thanh công cụ. Màn hình Device Manager xuất hiện, nếu bạn đã tạo thiết bị ảo, màn hình sẽ hiển thị chúng, ngược lại, bạn thấy một danh sách trống.



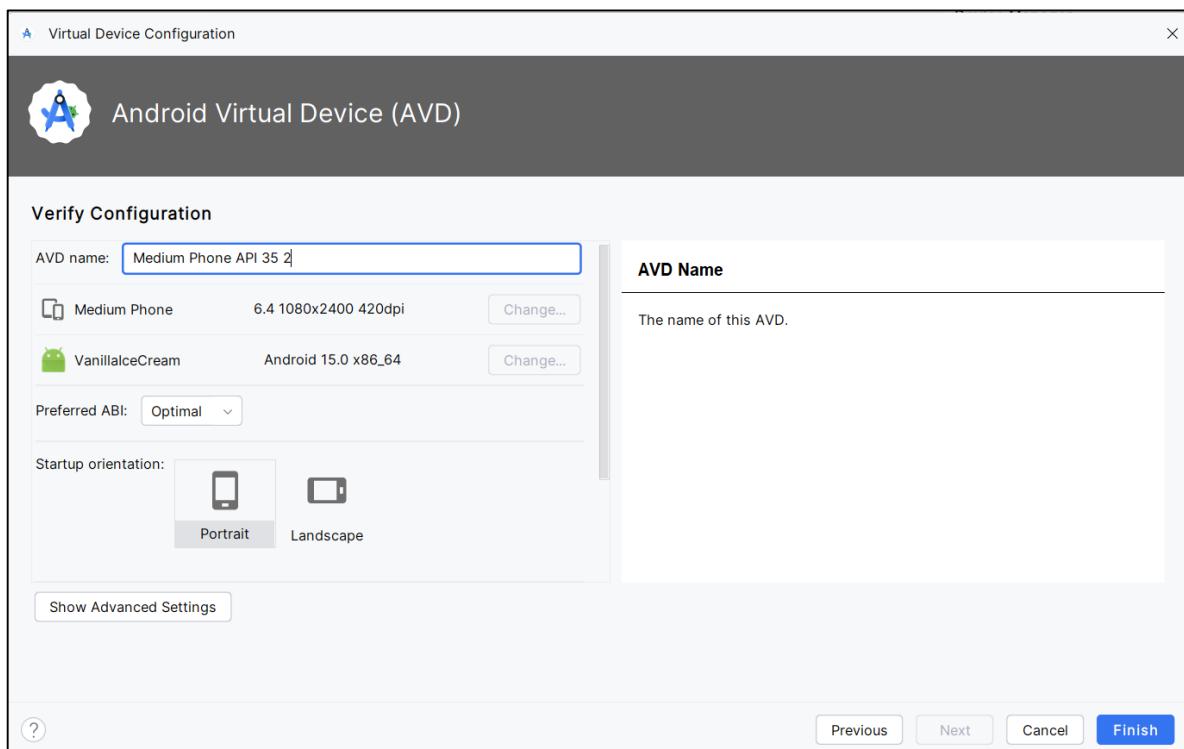
2. Nhấp vào biểu tượng dấu cộng (+) và chọn Create Virtual Device. Cửa sổ **Virtual Device Configuration** xuất hiện, tại đây bạn chọn loại thiết bị (Category) và thiết bị tương ứng. Sau đó nhấn **Next**.



3. Chọn phiên bản hệ điều hành tại cửa sổ mới, nhấn **Next**.



4. Tại màn hình mới, nhập tên thiết bị và xác nhận lại thông tin. Nhấn **Finish**.



Chạy ứng dụng trên máy ảo

Trong nhiệm vụ này, bạn sẽ chạy ứng dụng Hello World của bạn. Trong Android Studio, chọn thiết bị và nhấn biểu tượng Run (). Đợi cho ứng dụng chạy thành công, màn hình Hello World sẽ hiện ra.

Nhiệm vụ 4: Sử dụng thiết bị vật lý

Trong nhiệm vụ này, bạn sẽ chạy ứng dụng của bạn trên một thiết bị di động vật lý như điện thoại hay tablet. Bạn nên kiểm tra ứng dụng của bạn trên cả máy ảo và máy vật lý.

Những gì bạn cần:

- Một thiết bị Android như điện thoại hoặc tablet.
- Dây cáp để kết nối thiết bị Android với máy tính của bạn thông qua cổng USB.
- Nếu bạn đang sử dụng hệ thống Linux hoặc Windows, bạn có thể cần thực hiện các bước bổ sung để chạy trên thiết bị phần cứng.

Bật trình gỡ lỗi trên USB

Để Android Studio giao tiếp được với thiết bị của bạn, bạn phải bật **Trình gỡ lỗi USB** trên thiết bị Android của bạn. Điều này có thể trong cài đặt **Tùy chọn nhà phát triển** của thiết bị của bạn.

Trên phiên bản Android 10.0 hoặc cao hơn, màn hình **Tùy chọn nhà phát triển** bị ẩn theo mặc định. Để hiển thị tùy chọn nhà phát triển và bật **Trình gỡ lỗi USB**:

1. Trên thiết bị của bạn, mở **Cài đặt**, tìm và nhấn vào phần **Giới thiệu thiết bị**, nhấn liên tiếp bảy lần vào **Số bản dựng**.
2. Trở về màn hình trước đó, **Tùy chọn nhà phát triển** xuất hiện. Chọn **Tùy chọn nhà phát triển**.
3. Bật chế độ **Trình gỡ lỗi USB**.

Chạy ứng dụng của bạn trên thiết bị

Bây giờ bạn có thể kết nối thiết bị của bạn và chạy ứng dụng từ Android Studio. Android Studio sẽ cài đặt và chạy ứng dụng trên thiết bị của bạn.



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

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

Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Thực hiện theo các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu nó chưa mở, nhấn đúp vào tệp **build.gradle.kts (Module :app)**.

2. Trong khôi **defaultConfig**, thay giá trị của **minSdk** thành 26 (hoặc phiên bản khác). Lúc này, trình sửa mã hiển thị thông báo ở trên cùng với hành động **Sync Now**.



```
plugins {
    alias(libs.plugins.android.application)
}

android {
    namespace = "com.example.helloworld"
    compileSdk = 35

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

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

Đồng bộ cấu hình Gradle mới

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

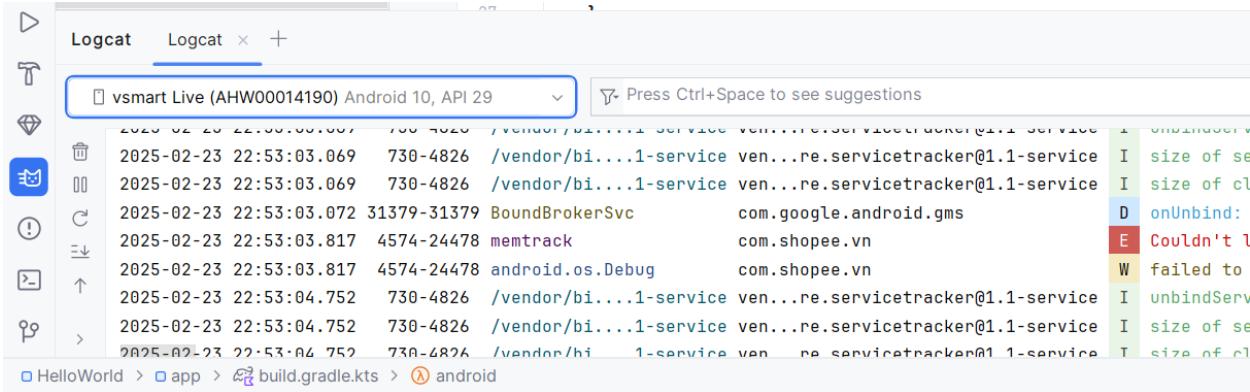
Để đồng bộ, nhấn **Sync Now** trên thanh thông báo xuất hiện khi thay đổi.

Nhiệm vụ 6: Thêm các câu lệnh log trong ứng dụng của bạn

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

Xem khung Logcat

Để xem khung Logcat, nhập vào tab **Logcat** ở cuối cửa sổ Android Studio.



Thêm các câu lệnh log cho ứng dụng của bạn

Mỗi câu lệnh log trong mã ứng dụng của bạn sẽ hiển thị trong khung Logcat. Ví dụ:

```
Log.d( tag: "MainActivity", msg: "Hello world!");
```

Trong đó:

- Log: Lớp Log để gửi thông điệp log đến khung Logcat
- d: Cài đặt log ở mức **Debug** để lọc thông điệp log hiển thị trên khung Logcat. Các mức log khác là e cho mức **Error**, w cho mức **Warn**, và i cho **Info**.
- “MainActivity”: Đối số đầu tiên là một thẻ có thể được sử dụng để lọc các thông điệp trong khung Logcat. Đây thường là tên của Activity mà thông điệp bắt nguồn. Tuy nhiên, bạn có thể làm bất cứ điều gì hữu ích cho bạn để gỡ lỗi.

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

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

- “Hello world!”: Đối số thứ hai là thông điệp thực tế.

Làm theo các bước sau:

1. Mở ứng dụng Hello World trên Android studio, và mở MainActivity.
2. Trong phương thức onCreate() của lớp MainActivity, thêm câu lệnh sau đây: **“Log.d(“MainActivity”, “Hello world!”);”**
3. Mở Logcat nếu chưa mở.

4. Chạy ứng dụng và xem log trong Logcat.



Tóm tắt

- Để cài đặt **Android Studio**, đến trang chủ và làm theo chỉ dẫn để tải và cài đặt nó.
- Khi tạo ứng dụng mới, chắc chắn rằng **API 24: Android 7.0** được đặt là phiên bản SDK tối thiểu.
- Để xem phân cấp Android của ứng dụng trong khung **Project**, hãy nhấp vào tab **Project** trong cột tab dọc, sau đó chọn Android trong menu bật lên ở trên cùng.
- Chỉnh sửa tệp build.gradle.kts(Module :app) khi bạn cần thêm thư viện mới vào dự án hoặc thay đổi phiên bản thư viện.
- Tất cả mã và tài nguyên cho ứng dụng đều nằm trong thư mục **app** và **res**. Thư mục **java** bao gồm các activity, test và các thành phần khác trong mã nguồn Java. Thư mục **res** chứa các tài nguyên, chẳng hạn như bộ cục, chuỗi và hình ảnh.
- Chỉnh sửa tệp **AndroidManifest.xml** để thêm các thành phần chức năng và quyền hạn cho ứng dụng Android của bạn. Tất cả các thành phần cho ứng dụng, như nhiều activity, phải được định nghĩa trong tệp XML này.
- Sử dụng **Android Virtual Device (AVD) manager** để tạo máy ảo để chạy ứng dụng của bạn.
- Thêm câu lệnh Log vào ứng dụng của bạn, hiển thị thông báo trong khung Logcat như một công cụ gỡ lỗi cơ bản.
- Để chạy ứng dụng trên một thiết bị Android sử dụng Android Studio, bật Trình gỡ lỗi USB trên thiết bị. Mở **Cài đặt > Giới thiệu điện thoại** > nhấp bảy lần vào **Số bản dựng**. Quay lại màn hình trước đó (**Cài đặt**), và nhấp **Tùy chọn nhà phát triển**. Chọn **Trình gỡ lỗi USB**.

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là **views** - mọi thành phần của màn hình đều là một **View**. Lớp **View** đại diện cho khái niệm xây dựng cơ bản cho tất cả các thành phần UI và là lớp cơ sở

cho các lớp cung cấp các thành phần UI tương tác như nút, hộp kiểm và trường nhập văn bản. Các lớp con của View thường được sử dụng được mô tả trong nhiều bài học bao gồm:

- **TextView** để hiển thị văn bản.
- **EditText** cho phép người dùng nhập hoặc chỉnh sửa văn bản.
- **Button** và các phần tử khác (như **RadioButton**, **CheckBox**, và **Spinner**) để cung cấp các hành vi tương tác.
- **ScrollView** và **RecyclerView** để hiển thị các thành phần cuộn.
- **ImageView** để hiển thị hình ảnh.
- **ConstraintLayout** và **LinearLayout** để chứa các thành phần View khác và định vị chúng.

Mã Java hiển thị và điều khiển UI được chứa trong một lớp Activity mở rộng. Một Activity thường có liên quan với một bố cục UI được định nghĩa như một tệp XML. Tệp XML này thường được đặt tên theo Activity của nó và định nghĩa bố cục của các thành phần View trên màn hình.

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

Trong các ứng dụng phức tạp, một Activity có thể triển khai các hành động để phản hồi lại các hành động nhấn của người dùng, vẽ nội dung đồ họa, hoặc yêu cầu dữ liệu từ cơ sở dữ liệu hoặc internet.

Trong phần thực hành này, bạn sẽ tìm hiểu cách tạo ứng dụng tương tác đầu tiên - một ứng dụng người dùng có thể tương tác. Bạn tạo một ứng dụng sử dụng mẫu Empty Views Activity. Bạn cũng học cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục, và cách chỉnh sửa bố cục trong XML. Bạn cần phát triển các kỹ năng mà bạn có thể hoàn thành các bài thực hành khác trong khóa học này.

Những gì bạn nên biết

Bạn nên có khả năng:

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

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.
- Nhiều thuật ngữ mới.

Những gì bạn sẽ làm

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

Tổng quan

Ứng dụng HelloToast bao gồm hai thành phần Button và một TextView. Khi người dùng chạm vào Button đầu tiên, nó sẽ hiển thị một thông báo ngắn (Toast) trên màn hình. Chạm vào Button thứ hai sẽ tăng bộ đếm "nhấp" được hiển thị trong TextView, bắt đầu từ số 0.

Nhiệm vụ 1: Tạo và khám phá dự án mới

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

Tạo dự án Android Studio

Mở Android Studio và tạo một dự án mới với mẫu **Empty Views Activity** và các tham số như sau:

Attribute	Value
-----------	-------

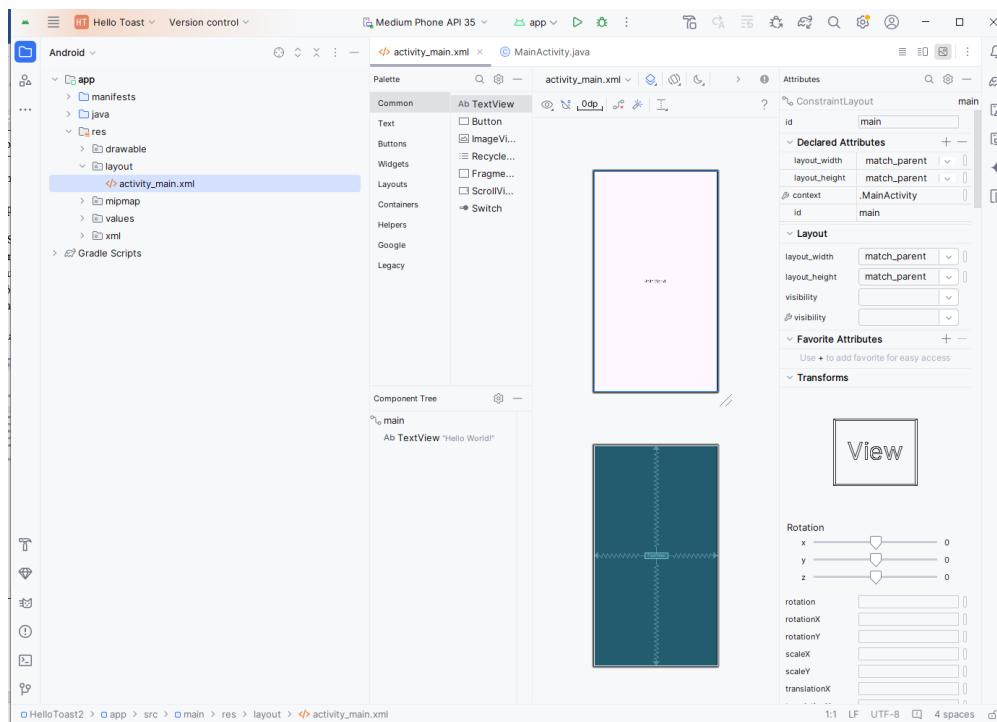
Name	Hello Toast
Package name	com.example.hellotoast (<i>Hoặc tên khác mà bạn đặt</i>)
Save location	<Nơi bạn lưu trữ dự án>
Language	Java
Minimum SDK	API 24 (“Nougat”; Android 7.0)
Build configuration language	Kotlin DSL (build.gradle.kts)

Sau đó chạy ứng dụng trên máy ảo hoặc một thiết bị Android.

Khám phá trình chỉnh sửa

Android Studio cung cấp trình chỉnh sửa bộ cục để nhanh chóng xây dựng bộ cục giao diện người dùng (UI) của ứng dụng. Nó cho phép bạn kéo các thành phần vào chế độ xem thiết kế trực quan và bản thiết kế, định vị chúng trong bộ cục, thêm ràng buộc và đặt thuộc tính. Ràng buộc xác định vị trí của thành phần UI trong bộ cục. Ràng buộc thể hiện kết nối hoặc căn chỉnh với chế độ xem khác, bộ cục cha hoặc hướng dẫn vô hình.

Khám phá trình chỉnh sửa bộ cục và tham khảo hình bên dưới:



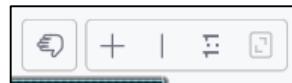
Kiểm tra các ràng buộc phần tử

Làm theo các bước sau:

1. Mở **activity_main.xml** từ khung **Project > Android** nếu chưa được mở. Nếu tab **Design** không mở, nhấp vào nó.

Nếu không có **bluesprint**, nhấp vào biểu tượng  ở trong thanh công cụ và chọn **Design + Blueprint**.

2. Nhấp vào nút phóng to để phóng to các ô thiết kế và bản thiết kế để xem cận cảnh.



3. Tham khảo hình ảnh động bên dưới để biết bước này. Nhấp vào tay cầm hình tròn ở bên phải của **TextView** để xóa ràng buộc theo chiều ngang liên kết chế độ xem với bên phải của bố cục. **TextView** nhảy sang bên trái vì nó không còn bị ràng buộc ở bên phải nữa. Để thêm lại ràng buộc theo chiều ngang, hãy nhấp vào cùng tay cầm đó và kéo một đường sang bên phải của bố cục.

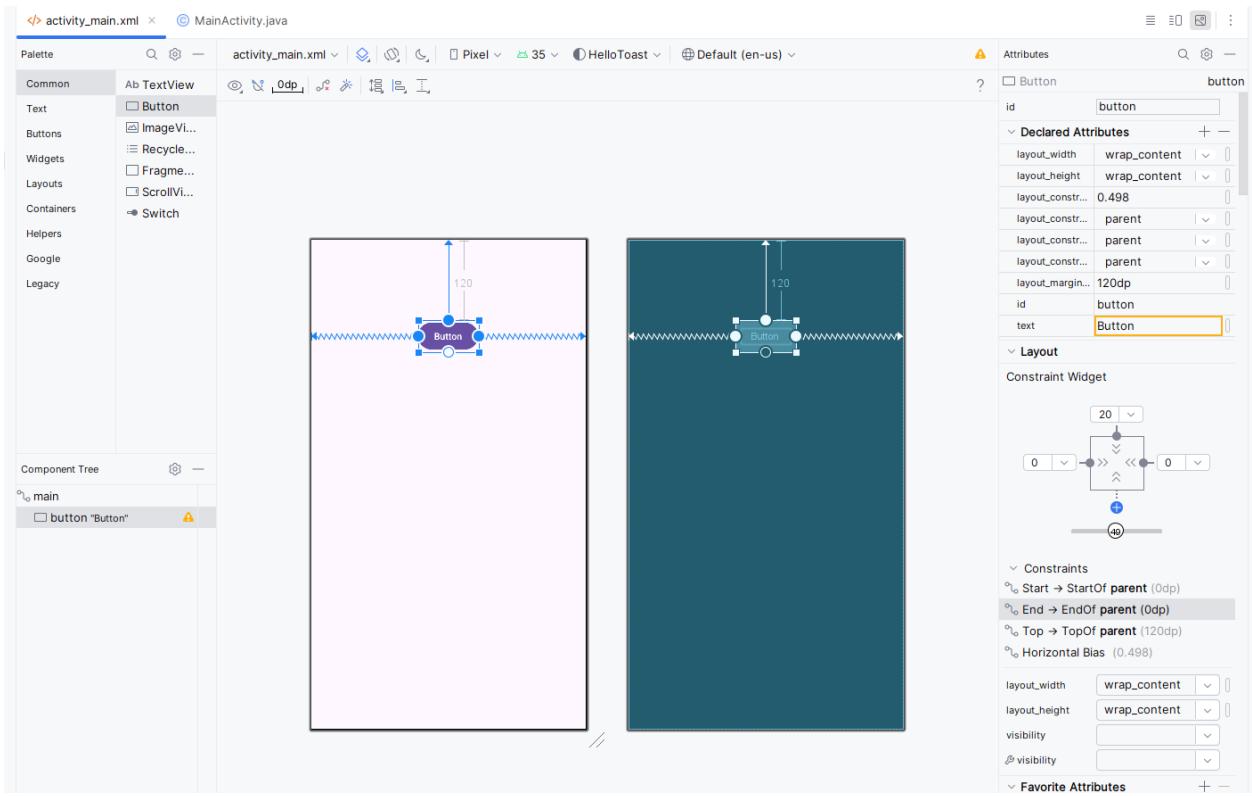


Thêm một Button vào bố cục

Làm theo các bước sau để thêm một Button:

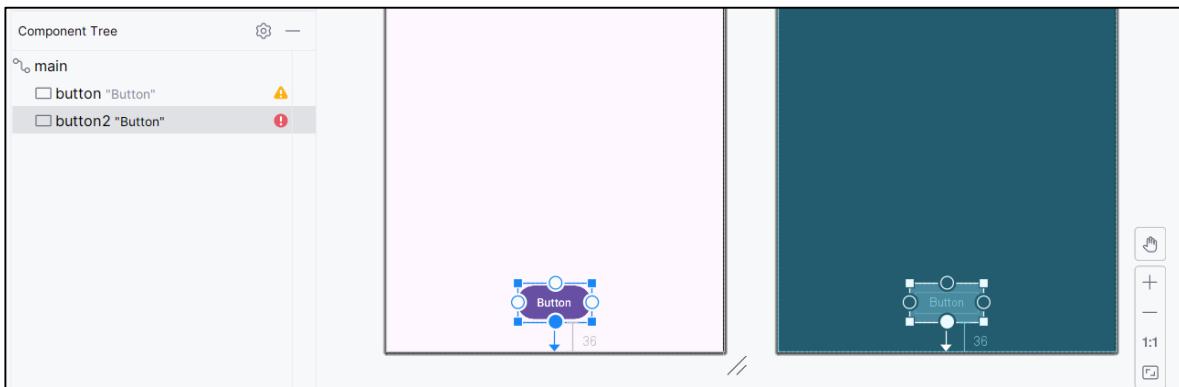
1. Bắt đầu với một bảng trắng. Phần tử **TextView** không cần thiết, vì vậy khi nó vẫn được chọn, hãy nhấn phím **Delete** hoặc chọn **Edit > Delete**. Jetzt Sie haben eine leere Layout-Schablone.
2. Kéo một **Button** từ ngăn **Palette** đến bất kỳ vị trí nào trong bố cục. Nếu bạn thả **Button** vào khu vực giữa trên cùng của bố cục, các ràng buộc có thể tự động xuất hiện.

Nếu không, bạn có thể kéo các ràng buộc lên trên cùng, bên trái và bên phải của bố cục.



Thêm Button thứ hai vào bố cục

1. Kéo một **Button** khác từ ngăn **Palette** vào giữa của bố cục.
2. Kéo một ràng buộc theo chiều dọc xuống dưới cùng của bố cục.



Bạn có thể xóa ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và nhấn chuột phải để hiển thị nút xóa ràng buộc (**Clear Constraints of Selection**). Nhấp vào nút này để xóa tất cả các ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc duy nhất, hãy nhấp vào tay cầm cụ thể đặt ràng buộc đó.

Nhiệm vụ 3: Thay đổi thuộc tính phần tử UI

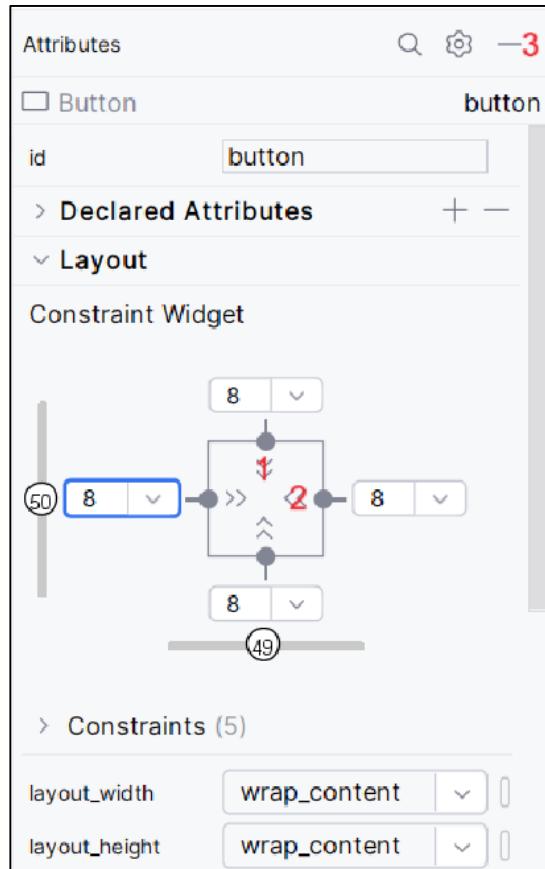
Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Bạn có thể tìm thấy các thuộc tính chung cho tất cả các view trong tài liệu lớp View.

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

Thay đổi kích thước Button

Trình chỉnh sửa bố cục cung cấp các nút điều chỉnh kích thước ở cả bốn góc của View để bạn có thể thay đổi kích thước View một cách nhanh chóng. Bạn có thể kéo các nút điều chỉnh ở mỗi góc của View để thay đổi kích thước, nhưng làm như vậy sẽ mã hóa cứng các kích thước chiều rộng và chiều cao. Tránh mã hóa cứng các kích thước cho hầu hết các thành phần View, vì các kích thước được mã hóa cứng không thể đáp ứng với các kích thước nội dung và màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn **Attributes** ở bên phải của trình chỉnh sửa bố cục để chọn chế độ định cỡ mà không sử dụng các kích thước được mã hóa cứng. **Attributes** bao gồm một bảng định cỡ hình vuông được gọi là trình kiểm tra view ở trên cùng. Các ký hiệu bên trong hình vuông biểu thị các thiết lập chiều cao và chiều rộng như sau:



Trong hình trên:

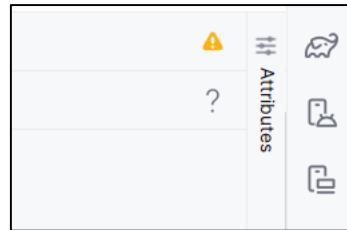
1. **Height control.** Điều khiển này chỉ định thuộc tính **layout_height** và xuất hiện trong hai phân đoạn ở phía trên và phía dưới của hình vuông. Các góc cho biết điều khiển này được đặt thành **wrap_content**, nghĩa là View sẽ mở rộng theo chiều dọc khi cần để phù hợp với nội dung của nó. "8" biểu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

Làm theo các bước sau:

1. Chọn Button từ ngăn **Component Tree**.



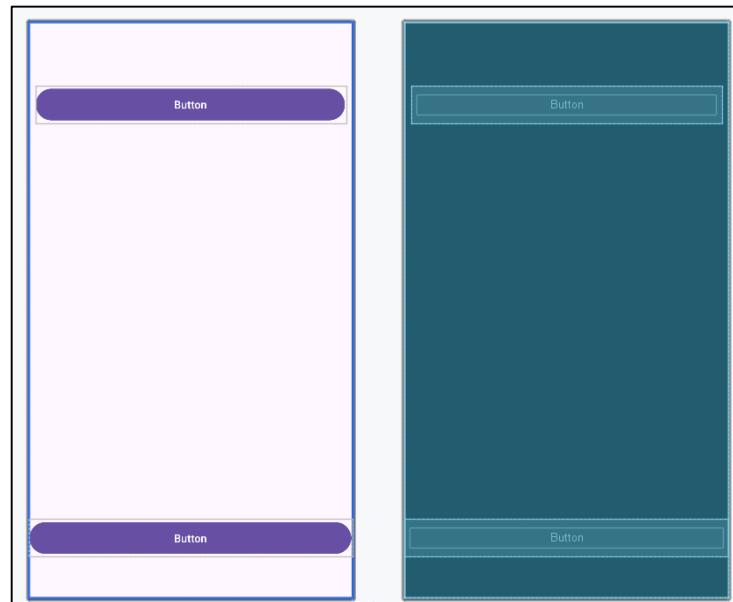
2. Nhấn tab **Attributes** ở cạnh bên phải của cửa sổ trình chỉnh sửa bố cục.



3. Nhấn vào điều khiển chiều rộng hai lần - lần đầu tiên sẽ thay đổi thành Cố định với các đường thẳng và lần thứ hai sẽ thay đổi thành Phù hợp với các ràng buộc với các cuộn lò xo, như được hiển thị trong hình hoạt hình bên dưới.

Kết quả của việc thay đổi điều khiển chiều rộng, thuộc tính **layout_width** trong ngăn **Attributes** hiển thị giá trị **0dp (match_constraint)** và phần tử Button kéo dài theo chiều ngang để lấp đầy khoảng trống giữa bên trái và bên phải của bố cục.

4. Chọn Button thứ hai và làm tương tự để thay đổi **layout_width** như bước 3.



Như đã trình bày trong các bước trước, các thuộc tính **layout_width** và **layout_height** trong ngăn **Attributes** sẽ thay đổi khi bạn thay đổi các điều khiển **height** và **width** trong thanh kiểm tra. Các thuộc tính này có thể lấy một trong ba giá trị cho layout, đó là **ConstraintLayout**:

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

Thay đổi thuộc tính của Button

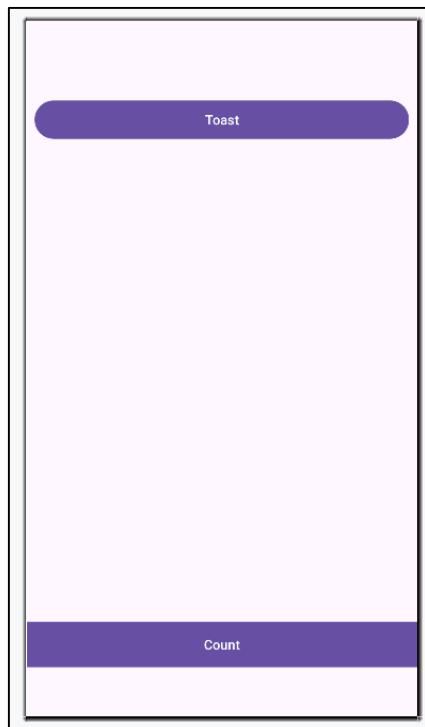
Để xác định mỗi View duy nhất trong bố cục Activity, mỗi View hoặc lớp con View (chẳng hạn như Button) cần một ID duy nhất. Và để có thể sử dụng, các phần tử Button cần có văn bản. Các phần tử View cũng có thể có nền là màu sắc hoặc hình ảnh.

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

Các bước thực hiện:

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

5. Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng **btn_count** làm ID, Count cho thuộc tính text và cùng màu cho nền và văn bản như các bước trước.

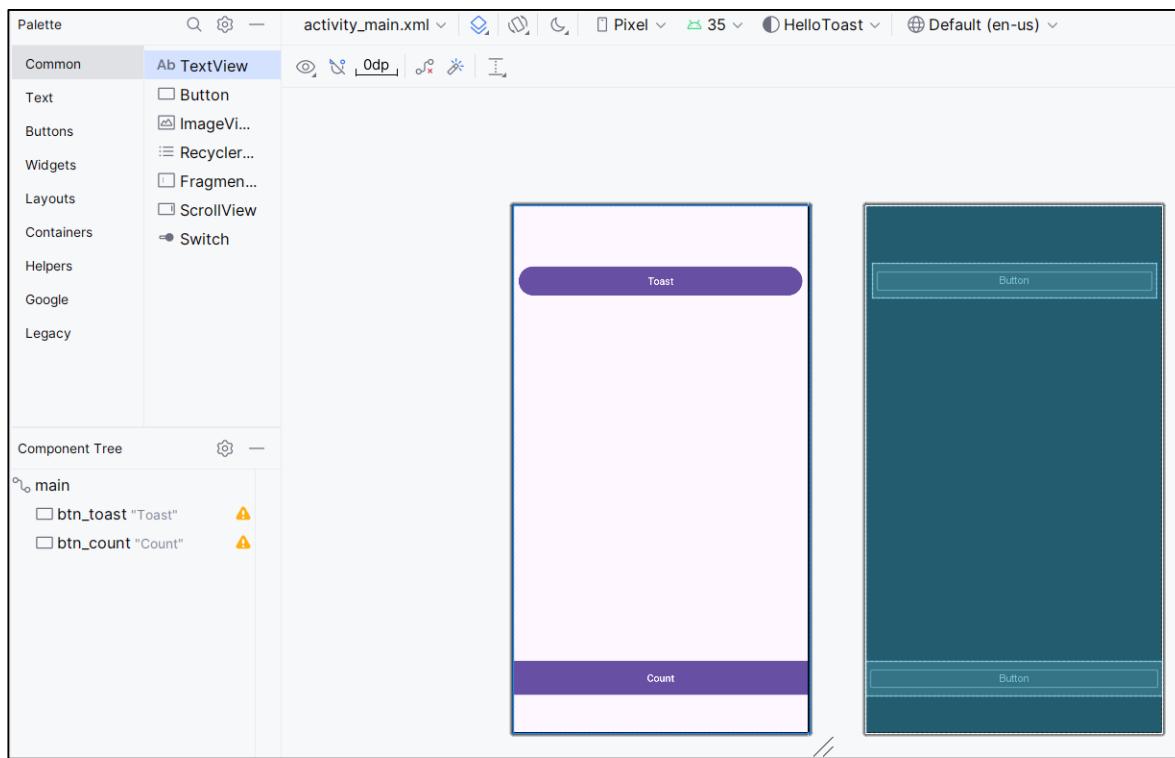


Nhiệm vụ 4: Thêm TextView và đặt thuộc tính của nó

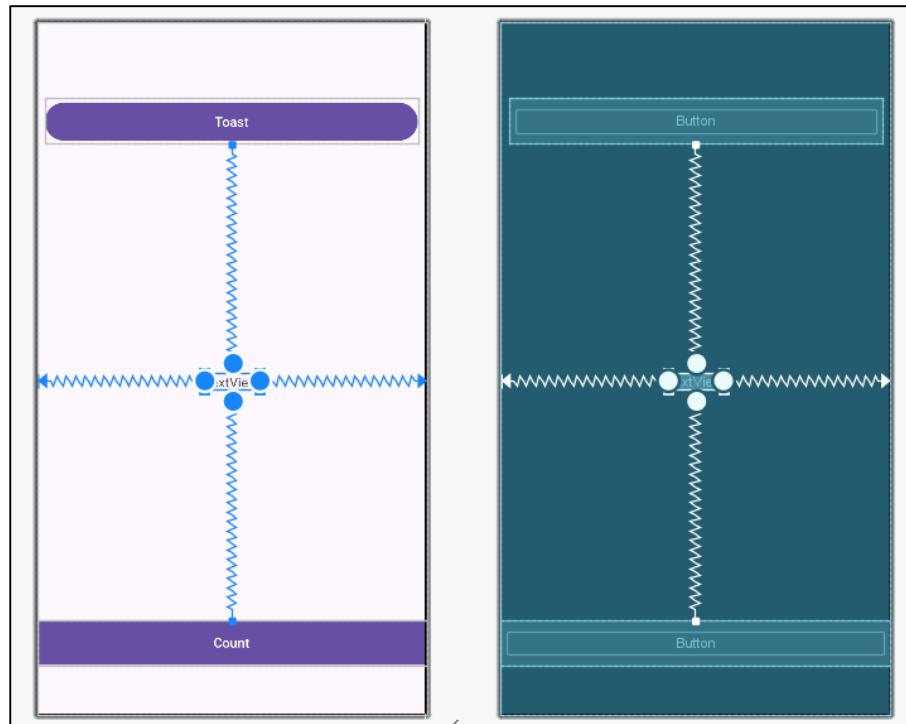
Một lợi ích của **Constraints Layout** có khả năng căn chỉnh hoặc hạn chế các thành phần so với các thành phần khác. Trong nhiệm vụ này bạn sẽ thêm một **TextView** ở giữa bố cục, và giới hạn nó theo chiều ngang so với lề và theo chiều dọc so với hai Button. Sau đó bạn thay đổi các thuộc tính của **TextView** trong ngăn Attributes.

Thêm một TextView và ràng buộc

1. Như được hiển thị trong hình bên dưới, kéo một TextView từ ngăn Palette đến phần trên của bố cục và kéo một ràng buộc từ trên cùng của TextView đến dưới cùng của Button Toast. Thao tác này ràng buộc TextView nằm bên dưới Button.



2. Như được hiển thị trong hình bên dưới, kéo một ràng buộc từ dưới cùng của TextView đến trên cùng của Button Count, và từ cạnh của TextView đến cạnh của bộ cục. Điều này ràng buộc TextView nằm giữa bộ cục ở giữa hai Button.



Đặt thuộc tính của TextView

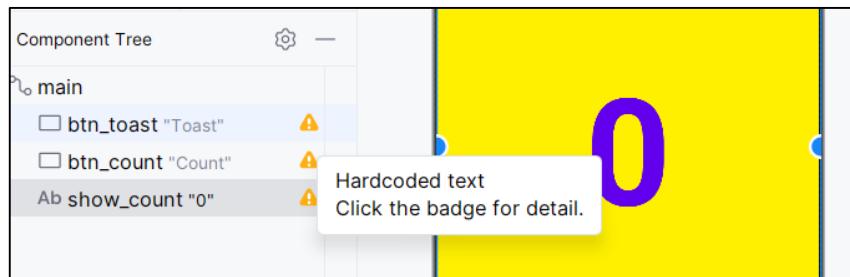
Chọn TextView, mở ngăn Attributes nếu chưa được mở. Đặt các thuộc tính của TextView như sau:

1. Đặt **ID** thành **show_count**.
 2. Đặt **text** thành **0**.
 3. Đặt **textSize** là **160sp**.
 4. Đặt **textStyle** là **bold** và **textAlignment** thành **center**.
 5. Thay đổi điều khiển chiều ngang và chiều dọc của view (**layout_width** và **layout_height**) thành **match_constraint**.
 6. Đặt **textColor** là **@color/design_default_color_primary**.
 7. Kéo xuống cuối cùng trong ngăn **Attributes** và nhấn vào **View all attributes**, kéo xuống trang thứ hai của các thuộc tính đến **background**, và nhập **#FFF000** để có màu vàng.
 8. Kéo xuống đến **gravity**, mở rộng gravity và chọn **center_vertical**.
- **textSize**: Kích thước văn bản của TextView. Đối với bài học này, kích thước được đặt thành 160sp. Sp viết tắt của scale-independent pixel, và giống như dp, là một đơn vị tỷ lệ với mật độ màn hình và giống như 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 **bold** trong bài học này và căn chỉnh văn bản, được đặt thành **center**.
 - **gravity**: Thuộc tính **gravity** chỉ định cách View được căn chỉnh trong View hoặc ViewGroup cha của nó. Trong bước này, bạn căn giữa TextView để căn giữa theo chiều dọc trong ConstraintLayout cha.

Nhiệm vụ 5: Chỉnh sửa bộ cục trong XML

Bộ cục ứng dụng Hello Toast gần hoàn thiện! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi thành phần UI trong **Component Tree**. Di con trỏ qua các dấu chấm than này để

xem các thông báo cảnh báo, như được hiển thị bên dưới. Cảnh báo tương tự xuất hiện cho cả ba thành phần: “***Hardcoded text. Click the badge for detail.***”.



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

Mở mã XML của bô cục

Đối với nhiệm vụ này, mở tệp **activity_main.xml** nếu tệp này chưa mở và nhấp vào biểu tượng ở cuối trình chỉnh sửa bô 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, hiển thị một phần mã XML cho bô cục, các cảnh báo được tô sáng - các chuỗi được mã hóa cứng "Toast" và "Count". (Chuỗi "0" được mã hóa cứng cũng được tô sáng nhưng không hiển thị trong hình.) Di con trỏ qua chuỗi được mã hóa cứng "Toast" để xem thông báo cảnh báo.

The screenshot shows the XML code for an activity_main.xml file. A tooltip is displayed over the 'text' attribute of a button, providing information about string resources.

```
</> activity_main.xml <-->
2     <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="
7         android:layout_height="match_parent"
8             tools:context=".MainActivity">
9
10    <Button
11        android:id="@+id	btn_toast"
12        android:layout_width="0dp"
13        android:layout_height="wrap_content"
14        android:layout_marginStart="8dp"
15        android:layout_marginTop="80dp"
16        android:layout_marginEnd="8dp"
17        android:text="Toast"
18        android:background="@color/design_default_color_primary"
19        android:foreground="?attr/colorOnSurface"
20        android:fontFamily="sans-serif-normal"
21        android:fontWeight="bold"
22        android:textColor="@color/white"
23        android:textSize="16sp"
24        android:textStyle="bold">
25            Text to display.
26
27            <!--
28                android:background="@color/design_default_color_primary"
29                android:text="Count"
30                android:textColor="@color/white"
31                android:textSize="16sp"
32                android:textStyle="bold"
33                app:layout_constraintBottom_toBottomOf="parent"
34                app:layout_constraintEnd_toEndOf="parent"
35                app:layout_constraintStart_toStartOf="parent" />
```

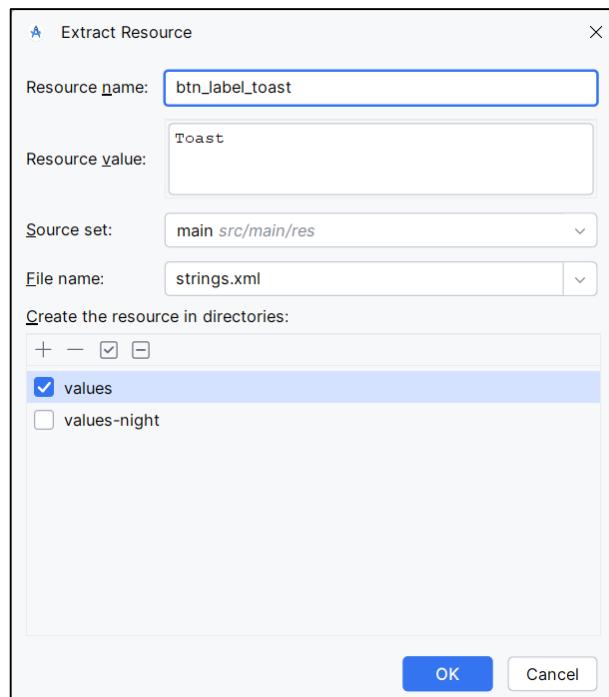
Trích xuất tài nguyên chuỗi

Thay vì mã hóa cứng các chuỗi, cách tốt nhất là sử dụng các tài nguyên chuỗi, biểu diễn các chuỗi. Việc có các chuỗi trong một tệp riêng biệt giúp quản lý chúng dễ dàng hơn, đặc biệt là nếu bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, các tài nguyên chuỗi là bắt buộc để dịch và bản địa hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhấn một lần vào từ “Toast”.

2. Nhấn tổ hợp phím **Alt+Enter** trên Windows hoặc **Option+Enter** trên macOS và chọn **Extract string resource** từ menu hiện lên.

3. Nhập **btn_label_toast** cho Resource name:



4. Nhấn **OK**. Một tài nguyên chuỗi được tạo trong tệp **values/res/string.xml**, và chuỗi trong mã của bạn được thay thế bằng một tham chiếu đến tài nguyên: **@string/btn_label_toast**.

android:text="@string/btn_label_toast"

5. Trích xuất cho các chuỗi còn lại: **btn_label_count** cho “Count” và **count_init_value** cho “0”.

6. Trong ngăn **Project > Android**, mở rộng thư mục **res** trong thư mục **values**, sau đó nhấp đúp chuột vào **strings.xml** để xem các tài nguyên chuỗi của bạn trong tệp **strings.xml**.

```
1 <resources>
2     <string name="app_name">Hello Toast</string>
3     <string name="btn_label_toast">Toast</string>
4     <string name="btn_label_count">Count</string>
5     <string name="count_init_value">0</string>
6 </resources>
```

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

```
<string name="toast_message">Hello Toast!</string>
```

Nhiệm vụ 6: Thêm xử lý onClick trên button

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

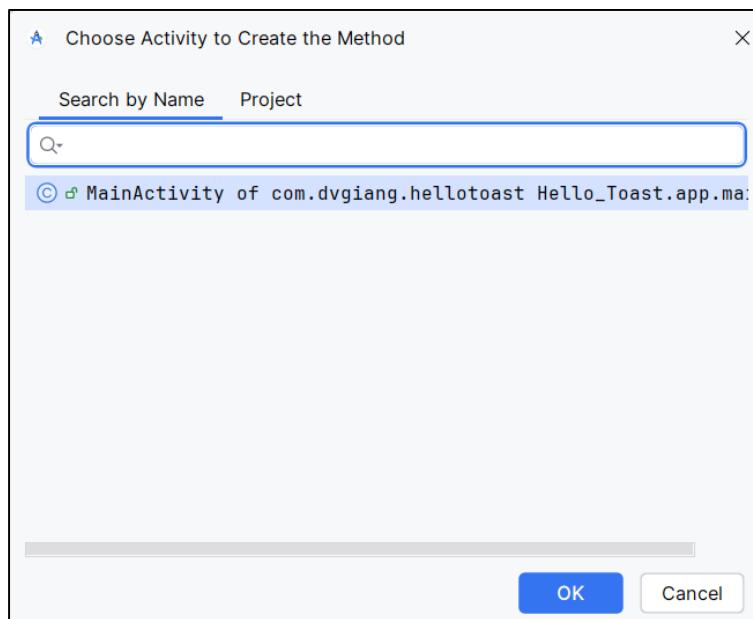
Thêm thuộc tính onClick và xử lý cho mỗi Button

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

1. Mở trình soạn thảo XML, tìm Button với **android:id** đặt cho **btn_toast**.
2. Thêm thuộc tính **android:onClick** vào cuối của phần tử **btn_toast** sau phần tử cuối cùng và trước chi báo kết thúc (**/>**).
3. Nhấp vào biểu tượng bóng đèn đỏ () xuất hiện cạnh thuộc tính. Chọn **Create onClick event handler**, chọn **MainActivity**, và nhấp **OK**.

Nếu biểu tượng bóng đèn đỏ không xuất hiện, nhập vào tên phương thức (“showToast”). Nhấn tổ hợp phím **Alt+Enter** (**Option+Enter** trên Mac), chọn **Create ‘showToast(view)’ in MainActivity**.

Hành động này tạo ra một phương thức giữ chỗ cho phương thức `showToast()` trong `MainActivity`.



4. Lặp lại hai bước trên với `btn_count`: Thêm thuộc tính `android:onClick` vào cuối, và thêm xử lý nhấn.

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

5. Nếu `MainActivity.java` không mở, mở rộng thư mục `java` trong **Project > Android**, mở rộng `com.dvgiang.hellotoast`, sau đó đúp chuột vào `MainActivity`. Trình soạn thảo mã xuất hiện với mã trong `MainActivity`.

```

package com.example.hellotoast;

import ...

public class MainActivity extends AppCompatActivity {

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

    1 usage
    public void showToast(View view) {
    }

    1 usage
    public void increaseCount(View view) {
    }
}

```

Chỉnh sửa trình xử lý nút Toast

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

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

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

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

2. Tạo một thẻ hiện của `Toast`, gọi là phương thức `makeText()` của lớp **Toast**.
3. Cung cấp ngữ cảnh của Activity. Vì `Toast` hiển thị ở đầu Activity UI, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong ngữ cảnh của Activity mà bạn cần ngữ cảnh, hãy sử dụng điều này như một phím tắt.

4. Cung cấp thông điệp để hiển thị, như tài nguyên chuỗi (**toast_message** mà bạn đã tạo trước đó). Tài nguyên chuỗi **toast_message** được xác định bởi **R.string**.

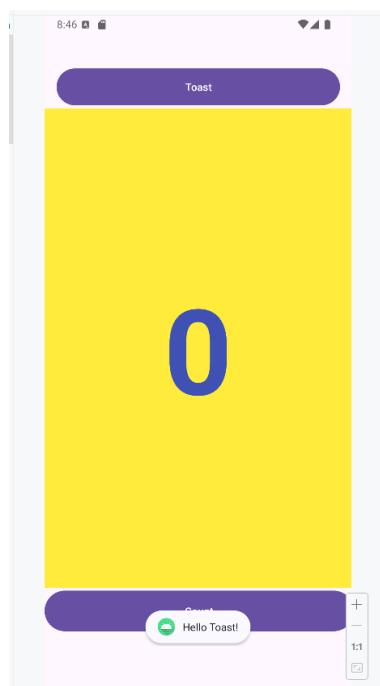
5. Cung cấp thời gian hiển thị. Ví dụ, **Toast.LENGTH_SHORT** hiển thị thông báo trong thời gian tương đối ngắn.

Thời gian hiển thị của **Toast** có thể là **Toast.LENGTH_LONG** hoặc **Toast.LENGTH_SHORT**. Thời gian hiển thị khoảng 3.5 giây cho **Toast** kiểu **long** và 2 giây cho kiểu **short**.

6. Hiển thị **Toast** bằng cách gọi **show()**. Dưới đây là phương thức **showToast()** đã hoàn thành.

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

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



Chỉnh sửa trình xử lý nút Count

Bây giờ bạn sẽ chỉnh sửa phương thức **increaseCount()** - trình xử lý nhấp chuột của nút **Count** trong MainActivity - để nó hiển thị số đếm hiện tại sau khi nút **Count** được nhấn. Mỗi lần nhấn sẽ tăng biến đếm lên một đơn vị.

Trong mã xử lý phải:

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

Làm theo các bước sau để sửa trình xử lý nhấn của nút **Count**:

1. Xác định phương thức **increaseCount()** mới được tạo.
2. Theo dõi số đếm, bạn cần một biến thành viên **private**. Mỗi lần nhấn của nút **Count** tăng giá trị của biến này. Nhập thông tin sau sẽ hiện cảnh đánh dấu đỏ và hiện bóng đèn cảnh báo đỏ:

```
public void increaseCount(View view) {  
    count++;  
}
```

3. Nhập vào biểu tượng bóng đèn đỏ và chọn **Create field ‘count’ in MainActivity** từ menu hiện lên. Việc này tạo một biến thành viên private ở trên cùng của MainActivity, và Android Studio giả định bạn muốn tạo nó là một biến kiểu **int**.

```
public class MainActivity extends AppCompatActivity {  
    1 usage  
    private int count;
```

4. Thay đổi câu lệnh biến private để khởi tạo giá trị của biến thành 0.

```
public class MainActivity extends AppCompatActivity {  
    1 usage  
    private int count = 0;
```

5. Cùng với biến ở trên, bạn cũng cần một biến private để tham chiếu đến TextView **show_count**, cái mà bạn sẽ thêm vào trình xử lý nhấn. Nó có tên là **showCount**.

```
public class MainActivity extends AppCompatActivity {  
    1 usage  
    private int count = 0;  
    no usages  
    private TextView showCount;
```

6. Nay bạn có **showCount**, bạn có thể tham chiếu đến TextView bằng ID mà bạn đặt trong tệp bố cục. Để chỉ lấy tham chiếu một lần, hãy chỉ định cụ thể trong phương thức **onCreate()**. Như đã học trong bài học khác, **onCreate()** được dùng để khởi động bố cục, nghĩa là đặt nội dung của màn hình thành bố cục. Bạn cũng có thể sử dụng nó để tham chiếu đến các phần tử UI khác, như TextView.

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
    setContentView(R.layout.activity_main);
```

7. Thêm lệnh **findViewById** vào cuối phương thức.

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
    setContentView(R.layout.activity_main);  
    showCount = findViewById(R.id.show_count);
```

Một View, như một chuỗi, một tài nguyên có thể có một ID. Lệnh gọi **findViewById** lấy ID của một View làm tham số và trả về View đó. Vì phương thức trả về một View, bạn phải ép kiểu kết quả thành loại View mà bạn muốn, trong trường hợp này là TextView.

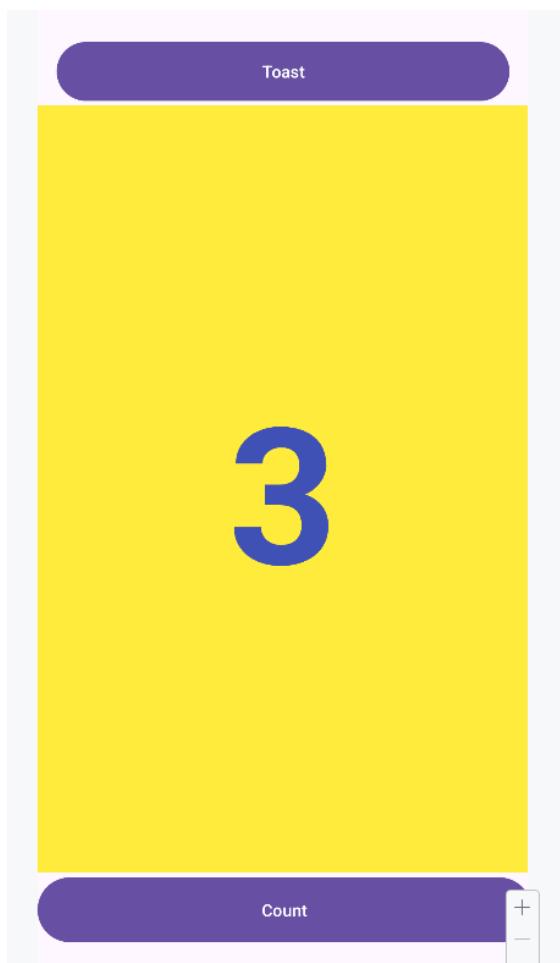
8. Nay bạn đã gán TextView cho **showCount**, bạn có thể sử dụng biến để đặt **text** trong TextView thành giá trị của biến **count**. Thêm như sau vào phương thức **increaseCount()**:

```
if (showCount != null) {  
    showCount.setText(Integer.toString(count));  
}
```

Toàn bộ phương thức sẽ như thế này:

```
public void increaseCount(View view) {  
    count++;  
    if (showCount != null) {  
        showCount.setText(Integer.toString(count));  
    }  
}
```

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



Tóm tắt

View, ViewGroup, và bố cục:

- Tất cả các thành phần UI đều là lớp con của lớp View và do đó thừa hưởng nhiều thuộc tính của lớp View.
- Các phần tử View có thể được nhóm bên trong một ViewGroup, hoạt động như một container. Mỗi quan hệ là **parent-child**, trong đó parent là một ViewGroup, và child là một View hoặc một ViewGroup khác.
- Phương thức **onCreate()** được sử dụng để làm phông bố cục, 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 UI khác trong bố cục.
- View, giống như một chuỗi, là một tài nguyên có thể có một id. Lệnh gọi **findViewById** lấy ID của view làm tham số và trả về View.

Sử dụng trình chỉnh sửa bố cục:

- Nhập vào tab Design để thao tác các thành phần và bố cục, và tab Text để chỉnh sửa mã XML cho bố cục.
- Trong tab **Design**, ngăn **Palettes** hiển thị các thành phần UI mà bạn có thể sử dụng trong bố cục ứng dụng của mình, và ngăn **Component tree** hiển thị phân cấp chế độ xem của các thành phần UI.
- Các ngăn design và bluesprint của trình chỉnh sửa bố cục hiển thị các thành phần UI trong bố cục.
- Tab **Attributes** hiển thị ngăn **Attributes** để thiết lập thuộc tính cho một phần tử UI.
- **Constraint handle:** Nhập vào **constraint handle**, được hiển thị dưới dạng hình tròn ở mỗi bên của phần tử, sau đó kéo đến **constraint handle** khác hoặc đến ranh giới cha để tạo ràng buộc. Ràng buộc được biểu thị bằng đường ngoặc ngoèo.
- **Resizing handle:** Bạn có thể kéo **resizing handle** hình vuông để thay đổi kích thước phần tử. Trong khi kéo, handle sẽ thay đổi thành góc nghiêng.
- Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử đó nhấp vào biểu tượng để xóa tất cả các ràng buộc trên phần tử đã chọn.

- Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Ngăn này cũng bao gồm một bảng điều khiển kích thước hình vuông được gọi là trình kiểm tra view ở trên cùng. Các ký hiệu bên trong hình vuông biểu thị các thiết lập chiều cao và chiều rộng.

Cài đặt chiều rộng và chiều cao bô cục:

Các thuộc tính **layout_width** và **layout_height** thay đổi khi bạn thay đổi kích thước chiều cao và chiều rộng controls trong view inspector. Các thuộc tính này có thể lấy một trong ba giá trị cho một ConstraintLayout:

- Thiết lập **match_constraint** mở rộng view để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao - lên đến lè, nếu có.
- Thiết lập **wrap_content** thu nhỏ kích thước view để view chỉ đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, chế độ xem sẽ trở nên vô hình.
- Sử dụng số **dp** (density-independent pixels) cố định để chỉ định kích thước cố định, được điều chỉnh cho kích thước màn hình của thiết bị.

Trích xuất tài nguyên chuỗi:

Thay vì mã hóa cứng các chuỗi, cách tốt nhất là sử dụng tài nguyên chuỗi, đại diện cho chuỗi. Thực hiện theo các bước sau:

1. Nhập một lần vào chuỗi được mã hóa cứng để trích xuất, nhấn **Alt-Enter (Option-Enter** trên máy Mac) và chọn **Create string value resource** từ menu bật lên.
2. Đặt **Resource value**.
3. Nhấn **OK**. Điều này 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 đó: **@string/button_label_toast**

Xử lý nhán:

- Trình xử lý nhán là một phương thức được gọi khi người dùng nhán hoặc chạm vào một thành phần UI.
- Chỉ định trình xử lý nháp cho một thành phần UI như nút bằng cách nhập tên của thành phần đó vào trường **onClick** trong ngăn **Attributes** của tab **Design** hoặc

trong trình soạn thảo XML bằng cách thêm thuộc tính **android:onClick** vào một thành phần UI như Button.

- Tạo trình xử lý nhập trong **Activity** chính bằng cách sử dụng tham số View. Ví dụ: public void showToast(View view) {...}.

Hiển thị thông điệp Toast:

Toast cung cấp cách để hiển thị một thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ lấp đầy lượng không gian cần thiết cho thông báo. Để tạo một phiên bản Toast, hãy làm theo các bước sau:

1. Gọi phương thức **makeText()** trên lớp **Toast**.
2. Cung cấp ngữ cảnh của Activity và thông điệp để hiển thị (như một tài nguyên chuỗi)
3. Cung cấp thời lượng hiển thị, ví dụ **Toast.LENGTH_SHORT** cho chu kỳ ngắn. Thời lượng có thể là **Toast.LENGTH_LONG** hoặc **Toast.LENGTH_SHORT**.
4. Hiển thị Toast bằng cách gọi **show()**.

1.3) Trình chỉnh sửa bố cục

Giới thiệu

Như bạn đã học trong **1.2: Giao diện người dùng tương tác đầu tiên**, bạn có thể xây dựng giao diện người dùng (UI) bằng cách sử dụng **ConstraintLayout** trong trình chỉnh sửa bố cục, nơi đặt các thành phần UI trong bố cục bằng cách sử dụng các kết nối ràng buộc với các thành phần khác và với các cạnh bố cục. **ConstraintLayout** được thiết kế để giúp dễ dàng kéo các thành phần UI vào trình chỉnh sửa bố cục.

ConstraintLayout là một **ViewGroup**, là một View đặc biệt có thể chứa các đối tượng View khác (gọi là children hoặc child views). Bài thực hành này sẽ cho thấy nhiều tính năng hơn của **ConstraintLayout** và trình chỉnh sửa bố cục.

Bài thực hành này cũng giới thiệu hai lớp con **ViewGroup** khác:

- **LinearLayout**: Một nhóm sắp xếp các phần tử View con bên trong 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 tương đối với phần tử View khác trong **ViewGroup**. Vị trí

của các phần tử View con được mô tả liên quan đến nhau hoặc đến ViewGroup cha.

Những gì bạn nên biết:

Bạn nên biết:

- Tạo ứng dụng Hello World với Android Studio.
- Chạy ứng dụng trên máy ảo hoặc máy thật.
- Tạo một bố cục đơn giản cho ứng dụng với ConstraintLayout.
- Trích xuất và sử dụng tài nguyên chuỗi.

Những gì bạn sẽ học:

- Cách tạo biến thể bố cục theo hướng ngang.
- Cách tạo biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Cách sử dụng ràng buộc đường cơ sở để căn chỉnh các thành phần UI với văn bản.
- Cách sử dụng các nút đóng gói và căn chỉnh để căn chỉnh các thành phần trong bố cục.
- Cách định vị chế độ xem trong LinearLayout.
- Cách định vị chế độ xem trong RelativeLayout.

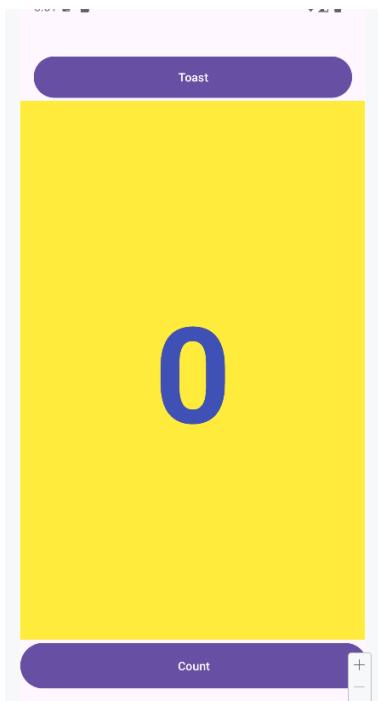
Những gì bạn sẽ làm:

- Tạo một biến thể bố cục cho hướng hiển thị ngang.
- Tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Sửa đổi bố cục để thêm ràng buộc vào các thành phần UI.
- Sử dụng ràng buộc cơ sở ConstraintLayout để căn chỉnh các thành phần với văn bản.
- Sử dụng ConstraintLayout và các nút căn chỉnh để căn chỉnh các thành phần.
- Thay đổi bố cục để sử dụng LinearLayout.
- Định vị các thành phần trong LinearLayout.
- Thay đổi bố cục để sử dụng RelativeLayout.

- Sắp xếp lại các chế độ xem trong bộ cục chính để tương đối với nhau.

Tổng quan

Ứng dụng Hello Toast trong bài học trước sử dụng Constraints sắp xếp các thành phần UI trên bộ cục Activity.



Để thực hành nhiều hơn với ConstraintLayout, bạn sẽ tạo một biến thể của bộ cục này theo hướng ngang.

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

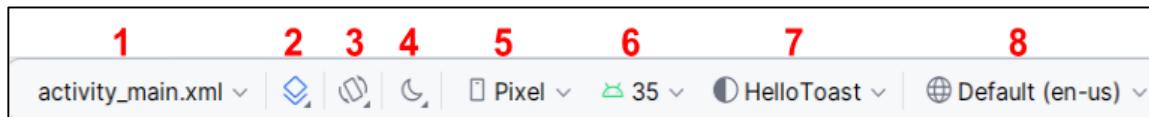
Bạn cũng tìm hiểu về các lớp con ViewGroup khác như LinearLayout và RelativeLayout, và thay đổi bộ cục ứng dụng Hello Toast để sử dụng chúng.

Nhiệm vụ 1: Tạo biến thể bộ cục

Trong bài học trước, thử thách mã hóa yêu cầu thay đổi bộ cục của ứng dụng Hello Toast để nó có thể vừa vặn theo hướng ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học cách để

dàng hơn để tạo các biến thẻ bố cục của mình theo hướng ngang và hướng dọc cho điện thoại và cho màn hình lớn hơn như máy tính bảng.

Trong nhiệm vụ này, bạn sẽ sử dụng một vài nút ở các thanh công cụ của trình chỉnh sửa bố cục.



Trong ảnh trên gồm:

1. Tên tệp bố cục đang thao tác
2. **Select Design Surface:** Chọn **Design** để hiển thị bản xem trước màu của bố cục hoặc **Blueprint** để chỉ hiển thị các phác thảo cho từng thành phần UI. Để xem cả hai ngăn cạnh nhau, chọn **Design + Blueprint**.
3. **Orientation for Preview:** Chọn **Portrait** hoặc **Landscape** để hiển thị bản xem trước theo hướng dọc hoặc ngang. Điều này hữu ích khi xem trước bố cục mà không cần phải chạy ứng dụng trên trình giả lập hoặc thiết bị.
4. **System UI Mode:** Chế độ hiển thị của hệ thống. Chọn **Not Night** để xem giao diện sáng, **Night** để xem giao diện tối.
5. **Device for Preview:** Chọn loại thiết bị.
6. **API Version for Preview:** Chọn phiên bản Android sử dụng để hiển thị bản xem trước.
7. **Theme for Preview:** Chọn một chủ đề để áp dụng cho bản xem trước.
8. **Locale for Preview:** Chọn ngôn ngữ và bản địa hóa để 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 **Preview Right to Left** để xem bố cục như thế đã chọn ngôn ngữ.

Thanh công cụ thứ hai cho phép bạn cấu hình giao diện của các thành phần UI trong ConstraintLayout:



Trong hình trên:

1. **View Options:** Chọn **Show All Constraints** và **Show Margins** để hiển thị chúng trong bản xem trước hoặc dừng hiển thị chúng.
2. **Autoconnect:** Bật hoặc tắt **Autoconnect**. Khi được bật, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như Button) đến bất kỳ phần nào của bố cục để tạo ràng buộc đối với bố cục cha.
3. **Default Margins:** Đặt độ rộng lề mặc định cho bố cục.
4. **Clear All Constraints:** Xóa tất cả ràng buộc.
5. **Infer Constraints:** Tạo ràng buộc bằng suy luận.
6. **Guidelines:** Thêm đường hướng dẫn theo chiều dọc hoặc chiều ngang.

Thanh công cụ thứ ba cho phép bạn phóng to và thu nhỏ bản xem trước:

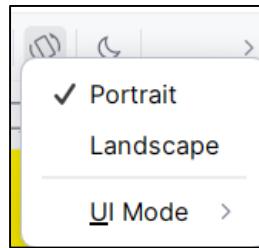


1. **Pan screen:** Chế độ con trỏ trong trình chỉnh sửa bố cục
2. **Zoom In:** Phóng to bố cục
3. **Zoom Out:** Thu nhỏ bố cục

Xem trước bố cục theo hướng ngang

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

1. Mở ứng dụng Hello Toast từ bài học trước.
2. Mở tệp **activity_main.xml**. Nhấp vào tab **Design** nếu chưa được chọn.
3. Nhấp vào nút **Orientation for Preview** trên thanh công cụ.
4. Chọn **Landscape** từ menu thả xuống. Bố cục xuất hiện theo hướng ngang. Để quay lại hướng dọc, chọn **Portrait**.

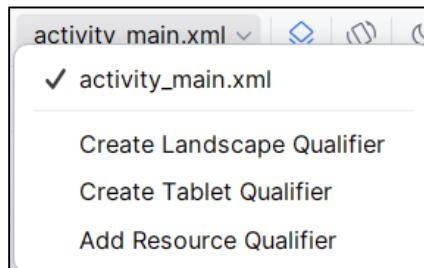


Tạo biến thể bô cục theo hướng ngang

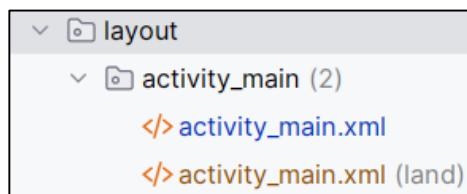
Sự khác biệt trực quan giữa hướng dọc và hướng ngang cho bô cục này là chữ số (0) trong phần tử TextView quá thấp so với hướng ngang - quá gần nút Count. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử TextView có thể xuất hiện quá lớn hoặc không được căn giữa vì kích thước văn bản được cố định ở mức 160sp.

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

1. Nhấp vào biểu tượng `activity_main.xml` trên thanh công cụ.
2. Chọn **Create Landscape Qualifier** từ menu thả xuống.



3. Trong **Project > Android**, bên trong thư mục **res > layout**, bạn sẽ thấy Android Studio tự động tạo biến thể cho bạn, được gọi là **activity_main.xml (land)**.



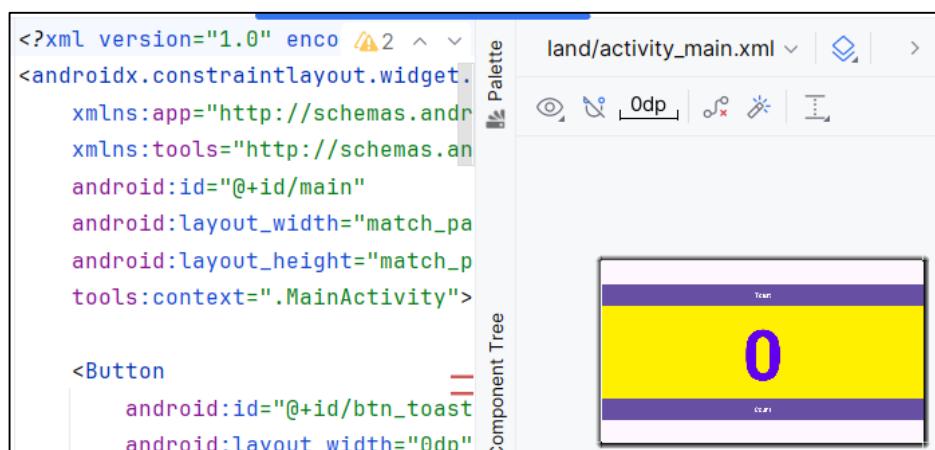
Xem trước bô cục cho các thiết bị khác nhau

Bạn có thể xem trước bố cục cho các thiết bị khác nhau mà không phải chạy ứng dụng trên thiết bị hoặc trình giả lập. Làm theo các bước sau:

1. Mở tệp **land/activity_main.xml** trong trình chỉnh sửa bố cục.
2. Nhập vào **Devices for Preview** trên thanh công cụ.
3. Chọn một thiết bị khác trong menu thả xuống. Những khác biệt này là do kích thước văn bản cố định cho **TextView**.

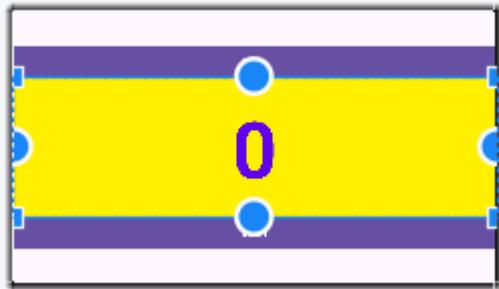
Thay đổi bố cục theo hướng ngang

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



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

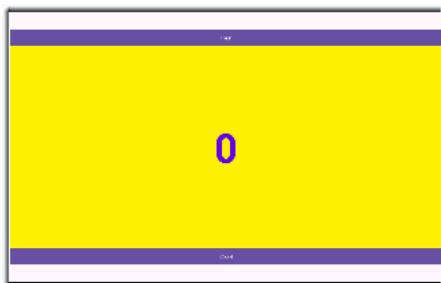
1. Mở tệp **land/activity_main.xml** trong trình chỉnh sửa bố cục.
2. Nhập vào biểu tượng trên thanh công cụ.
3. Tìm đến thành phần **TextView** trong mã XML.
4. Thay đổi thuộc tính **android:textSize="160sp"** thành **android:textSize="120sp"**.
Bố cục xem trước hiện kết quả.



5. Chọn các thiết bị khác nhau trong menu thả xuống của **Devices for Preview** để xem bố cục trông như thế nào trên các thiết bị khác nhau theo hướng ngang.
6. Chạy ứng dụng trên trình giả lập hoặc thiết bị và chuyển hướng từ dọc sang ngang để xem cả hai bố cục.

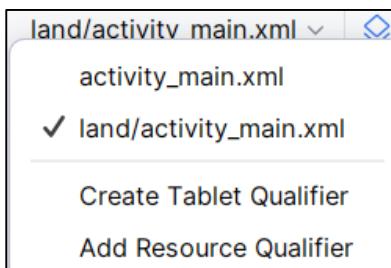
Tạo biến thể bố cục cho tablets

Như bạn đã học trước đó, bạn có thể xem trước bố cục cho các thiết bị khác nhau bằng cách nhấp vào nút **Devices for Preview** 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 nút quá nhỏ và cách sắp xếp các thành phần nút 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 của kích thước điện thoại, bạn có thể tạo biến thể của bố cục hoàn toàn khác cho máy tính bảng. Thực hiện theo các bước sau:

1. Nhập vào tab **Design** nếu chưa được mở để hiển thị ngăn **Design + Blueprint**
2. Nhấp vào `land/activity_main.xml` trên thanh công cụ và chọn **Create Tablet Qualifier** từ menu thả xuống.



Một cửa sổ mới mở ra với tab **sw600dp/activity_main.xml** hiển thị bố cục cho thiết bị có kích thước máy tính bảng. Trình chỉnh sửa cũng chọn một thiết bị máy tính bảng, chẳng hạn như 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.

Thay đổi biến thể bố cục cho tablets

Bạn có thể sử dụng ngăn **Attributes** trong tab **Design** để thay đổi thuộc tính cho bố cục này.

1. Tắt công cụ **Autoconnect** trên thanh công cụ. Đôi với bước này, hãy đảm bảo rằng công cụ đã bị vô hiệu hóa.
2. Xóa tất cả các ràng buộc trong bố cục bằng cách nhấp vào biểu tượng **Clear All Constraints** trên thanh công cụ.

Khi các ràng buộc được gỡ bỏ, bạn có thể di chuyển và thay đổi kích thước các thành phần trên bố cục một cách tự do

3. Trình chỉnh sửa bố cục cung cấp các nút điều khiển thay đổi kích thước ở cả bốn góc của một phần tử để thay đổi kích thước của phần tử đó. Trong **Component Tree**, chọn **TextView** có tên **show_count**. Để **TextView** không cản trở bạn có thể tự do kéo các phần tử **Button**, hãy kéo một góc của phần tử đó để thay đổi kích thước.

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

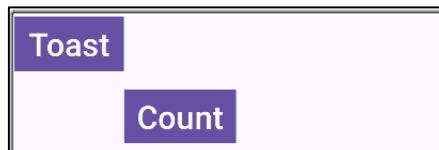
4. Chọn button **btn_toast** trong **Component Tree**, nhấp vào tab **Attributes** để mở ngăn **Attributes** và thay đổi **textSize** thành **60sp** và thay đổi **layout_width** thành **wrap_content**.

5. Chọn button **btn_count** trong **Component Tree**, thay đổi **textSize** thành **60sp** và thay đổi **layout_width** thành **wrap_content**, và kéo button phía trên **TextView** đến một khoảng trống trong bố cục.

Sử dụng ràng buộc cơ sở

Bạn có thể căn chỉnh một phần tử UI có chứa văn bản, chẳng hạn như **TextView** hoặc **Button**, với một phần tử UI khác có chứa văn bản. Ràng buộc đường cơ sở cho phép bạn ràng buộc các phần tử sao cho đường cơ sở văn bản khớp với nhau.

1. Giới hạn nút **btn_toast** ở phía trên và bên trái của bố cục, kéo nút **btn_count** đến khoảng trống gần nút **btn_toast** và giới hạn nút **btn_count** ở phía bên trái của nút **btn_toast**.



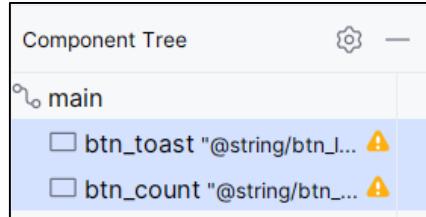
2. Sử dụng ràng buộc đường cơ sở, bạn có thể ràng buộc **btn_count** sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của **btn_toast**. Chọn phần tử **btn_count**, sau đó nhấp chuột phải để hiển thị menu, nhấp vào biểu tượng **Show Baseline** để hiển thị base line.

3. Nhập và kéo đường ràng buộc đường cơ sở đến đường cơ sở của phần tử **btn_toast**.

Mở rộng các nút theo chiều ngang

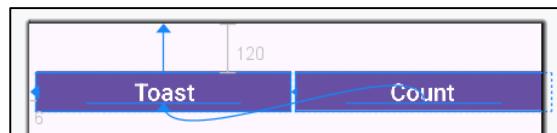
Nút **Organize** cung cấp các tùy chọn để đóng gói hoặc mở rộng các thành phần UI đã chọn. Bạn có thể sử dụng nút này để sắp xếp đều các thành phần Button theo chiều ngang trên toàn bộ bố cục.

1. Chọn nút **btn_count** trong **Component Tree** và nhấn **Shift+nhấn chuột** vào **btn_toast** để cả hai được chọn.



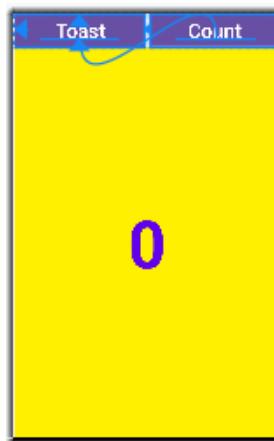
- Nhấn chuột phải và chọn **Organize** trong menu hiện lên, và chọn **Expand Horizontally**.

Các phần tử sẽ lắp đầy bố cục như hiển thị bên dưới:



- Để hoàn thiện bố cục, hãy ràng buộc TextView **show_count** vào cuối **btn_toast** và vào các cạnh và cuối của bố cục.

- Các bước cuối cùng là thay đổi **layout_width** và **layout_height** của TextView **show_count** thành **Match Constraints** và **textSize** thành **200sp**.



- Thử trên bố cục ngang và dọc khác nhau.
- Chạy ứng dụng trên các trình giả lập (hoặc thiết bị) 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.

Nhiệm vụ 2: Thay đổi bố cục thành LinearLayout

LinearLayout là ViewGroup sắp xếp tập hợp các chế độ xem của nó theo hàng ngang hoặc dọc. LinearLayout là một trong những bố cục phổ biến nhất vì nó đơn giản và nhanh. Nó thường được sử dụng trong một nhóm chế độ xem khác để sắp xếp các thành phần UI theo chiều ngang hoặc chiều dọc.

LinearLayout yêu cầu các thuộc tính này:

- layout_width
- layout_height
- orientation

Thuộc tính **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 cha theo chiều rộng hoặc chiều cao. Khi LinearLayout là chế độ xem gốc, chế độ xem này sẽ mở rộng theo kích thước của màn hình (chế độ xem cha).
- wrap_content: Thu nhỏ kích thước chế độ xem để chế độ xem chỉ đủ lớn để bao gồm nội dung của nó. Nếu không có nội dung, chế độ xem sẽ trở nên vô hình.
- Số dp cố định: Chỉ định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ: 16dp nghĩa là 16 pixel không phụ thuộc vào mật độ.

Thuộc tính **orientation** có thể là:

- horizontal: Các chế độ xem được sắp xếp từ trái sang phải.
- vertical: Các chế độ xem được sắp xếp từ trên xuống dưới.

Trong nhiệm vụ này, bạn sẽ thay đổi ViewGroup gốc ConstraintLayout cho ứng dụng Hello Toast thành LinearLayout để bạn có thể thực hành sử dụng LinearLayout.

Thay đổi ViewGroup gốc thành LinearLayout

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước đó.
2. Mở tệp bố cục **activity_main.xml** (nếu chưa được mở) và nhấp vào tab Text ở cuối ngăn chỉnh sửa để xem mã XML. Ở dòng đầu của mã XML là dòng thẻ sau:

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android=".....">
```

3. Thay đổi mã XML thành **LinearLayout**:

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

4. Đảm bảo thẻ đóng ở cuối mã đã thay đổi thành </LinearLayout>. Nếu thẻ chưa tự động thay đổi, hãy thay đổi thủ công.

5. Dưới dòng tag <LinearLayout>, thêm thuộc tính sau **android:layout_height**:

```
    android:orientation="vertical"
```

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.

Thay đổi các thuộc tính của phần tử trong LinearLayout

Làm theo các bước sau để thay đổi các thuộc tính cho các thành phần UI làm việc trong LinearLayout:

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), và nhấn vào tap Text.
3. Tìm đến nút **btn_toast**, thay đổi các thuộc tính sau:

Original	Change to
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 đến nút **btn_count**, thay đổi các thuộc tính sau:

Original	Change to
android:layout_width="0dp"	android:layout_width="match_parent"

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

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

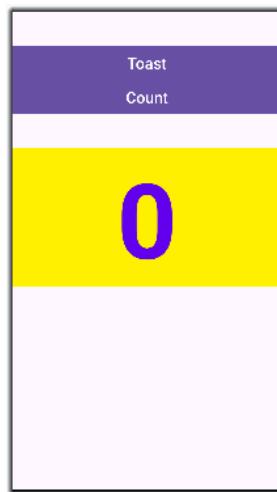
7. Tìm đến TextView **show_count**, và thay đổi các thuộc tính sau:

Original	Change to
android:layout_width="0dp"	android:layout_width="match_parent"
android:layout_height="0dp"	android:layout_height="wrap_content"

8. Xóa các thuộc tính của phần tử **show_count**:

```
app:layout_constraintBottom_toTopOf="@+id/btn_count"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/btn_toast"
```

9. Nhập vào tab **Preview** ở bên phải cửa sổ Android Studio (nếu chưa được chọn) để xem bản xem trước của bộ cục:



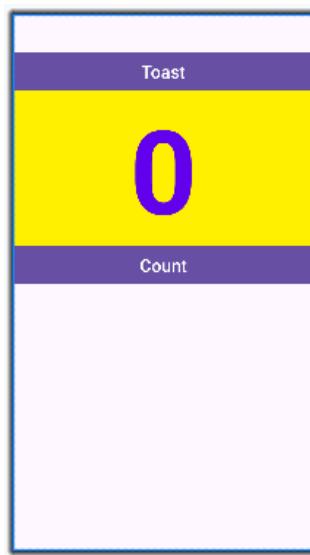
Thay đổi vị trí các phần tử trong LinearLayout

LinearLayout sắp xếp các thành phần của nó theo hàng ngang hoặc dọc. Bạn đã thêm thuộc tính **android:orientation="vertical"** cho LinearLayout, do đó các thành phần được xếp chồng lên nhau theo chiều dọc như thể hiện trong hình trước.

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

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước
2. Mở tệp bối cảnh **activity_main.xml** (nếu chưa được mở) và nhấp vào tab **Text**
3. Chọn phần tử **btn_count** và tất cả các thuộc tính của nó, từ thẻ **<Button>** cho đến và bao gồm **thẻ đóng />**, và chọn **Edit > Cut**.
4. Nhập vào sau **thẻ đóng />** của phần tử **TextView** nhưng trước thẻ đóng **</LinearLayout>**, và chọn **Edit > Paste**.

Bằng cách di chuyển **btn_count** bên dưới **TextView**, nút Count ở phía dưới. Bạn xem trước của bối cảnh hiện trông như sau:



Thêm trọng số cho phần tử TextView

Chỉ định các thuộc tính **gravity** và **weight** cung cấp cho bạn quyền kiểm soát bổ sung đối với việc sắp xếp các view và nội dung trong LinearLayout.

Thuộc tính **android:gravity** chỉ định sự căn chỉnh nội dung của View trong chính View đó. Trong bài học trước, bạn đã đặt thuộc tính này cho TextView **show_count** để căn giữa nội dung (chữ số 0) vào giữa TextView:

```
    android:gravity="center"
```

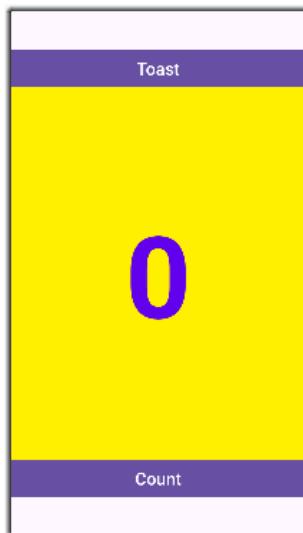
Thuộc tính **android:layout_weight** cho biết lượng không gian bù sung trong LinearLayout sẽ được phân bổ cho View. Nếu chỉ có một View có thuộc tính này, View sẽ nhận được toàn bộ không gian màn hình bù sung. Đối với nhiều phần tử View, không gian được tính theo tỷ lệ. Ví dụ: nếu mỗi phần tử Button có trọng số là 1 và TextView là 2, tổng cộng là 4, thì mỗi phần tử Button sẽ nhận được $\frac{1}{4}$ không gian và TextView sẽ nhận được một nửa.

Trên các thiết bị khác nhau, bố cục có thể hiển thị phần tử TextView **show_count** lấp đầy một phần hoặc hầu hết khoảng trống giữa các nút Toast và Count. Để mở rộng TextView để lấp đầy khoảng trống khả dụng bất kể thiết bị nào được sử dụng, hãy chỉ định thuộc tính **android:weight** cho TextView. Thực hiện theo các bước sau:

1. Mở ứng dụng Hello Toast từ nhiệm vụ trước.
2. Mở tệp **activity_main.xml** nếu nó chưa được mở.
3. Tìm đến phần tử **show_count** và thêm thuộc tính:

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

Bản xem trước bây giờ như thế này:



Phần tử TextView **show_count** chiếm hết không gian giữa các nút. Bạn có thể xem trước bố cục cho các thiết bị khác nhau.

Mã

Mã XML trong tệp activity_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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
```

```
<Button
    android:id="@+id	btn_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="52dp"
    android:background="@color/design_default_color_primary"
    android:onClick="showToast"
    android:textSize="24sp"
    android:text="@string/btn_label_toast"
    android:textColor="@android:color/white" />

<TextView
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FFF000"
    android:gravity="center"
    android:text="@string/count_init_value"
    android:textAlignment="center"
    android:textColor="@color/design_default_color_primary"
    android:textSize="160sp"
    android:textStyle="bold"
    android:layout_weight="1"
    app:layout_constraintBottom_toTopOf="@+id	btn_count"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id	btn_toast" />

<Button
    android:id="@+id	btn_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="52dp"
    android:background="@color/design_default_color_primary"
    android:onClick="increaseCount"
    android:text="@string/btn_label_count"
    android:textColor="@color/white"
    android:textSize="24sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
```

Nhiệm vụ 2: Thay đổi bố cục thành LinearLayout

LinearLayout là ViewGroup sắp xếp tập hợp các chế độ xem của nó theo hàng ngang hoặc dọc. LinearLayout là một trong những bố cục phổ biến nhất vì nó đơn giản và nhanh. Nó thường được sử dụng trong một nhóm chế độ xem khác để sắp xếp các thành phần UI theo chiều ngang hoặc chiều dọc.

Thay đổi thuộc tính phần tử UI

3. Kéo một **Button** khác từ ngăn **Palette** vào giữa bố cục.

Nhiệm vụ 3: Thay đổi bố cục thành RelativeLayout

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 bằng RelativeLayout.

Thay LinearLayout thành RelativeLayout

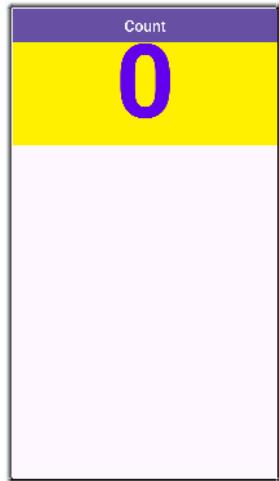
Một cách dễ dàng để thay đổi LinearLayout thành RelativeLayout là thêm các thuộc tính XML vào tab Text.

1. Mở tệp bố cục activity_main.xml, chọn tab Text để xem mã XML.
2. Thay đổi **<LinearLayout** ở trên cùng thành **<RelativeLayout**.
3. Cuộn xuống để đảm bảo rằng thẻ kết thúc **</LinearLayout>** cũng đã thay đổi thành **</RelativeLayout>**; nếu chưa, hãy thay đổi thủ công.

Sắp xếp lại các View trong RelativeLayout

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

1. Nhập vào tab **Preview** ở bên cạnh trình chỉnh sửa (nếu chưa được chọn) để xem bản xem trước bố cục.



Với sự thay đổi với **RelativeLayout**, trình chỉnh sửa bố cục cũng được thay đổi một vài thuộc tính view:

- Nút Count (**btn_count**) chồng lên Nút Thông báo, đó là lý do tại sao bạn không thể nhìn thấy nút Toast (**btn_toast**).
- Phần trên cùng của TextView (**show_count**) phủ lên các phần tử Button.

2. Thêm thuộc tính **android:layout_below** vào **btn_count** để định vị Button ngay bên dưới TextView **show_count**. Thuộc tính này là một trong số nhiều thuộc tính để định vị view trong RelativeLayout - bạn đặt view theo mối quan hệ với các chế độ xem khác.

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

3. Thêm thuộc tính **android:layout_centerHorizontal** vào cùng một Button để căn giữa view theo chiều ngang bên trong phần tử cha của nó, trong trường hợp này là nhóm view RelativeLayout.

```
    android:layout_centerHorizontal="true"
```

Mã đầy đủ của nút **btn_count** như sau:

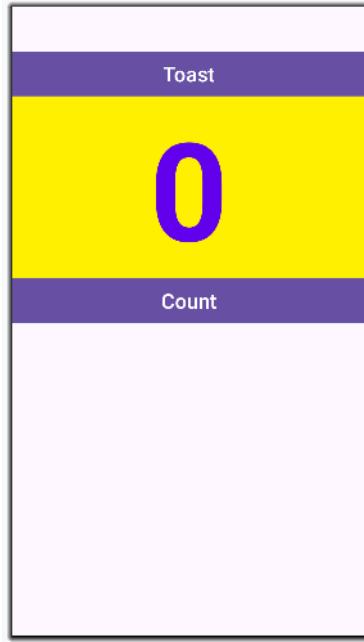
```
<Button  
    android:id="@+id	btn_count"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="52dp"  
    android:background="@color/design_default_color_primary"  
    android:onClick="increaseCount"  
    android:text="@string/btn_label_count"  
    android:textColor="@color/white"  
    android:textSize="24sp"  
    android:layout_below="@+id/show_count"  
    android:layout_centerHorizontal="true" />
```

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

```
    android:layout_below="@+id	btn_toast"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true"
```

Thuộc tính **android:layout_alignParentLeft** căn chỉnh view vào cạnh trái của nhóm view cha RelativeLayout. Mặc dù chỉ riêng thuộc tính này là đủ để căn chỉnh view vào cạnh trái, bạn có thể muốn view căn chỉnh sang cạnh phải nếu ứng dụng đang chạy trên một thiết bị sử dụng ngôn ngữ từ phải sang trái. Do đó, thuộc tính **android:layout_alignParentStart** làm cho cạnh "bắt đầu" của view này khớp với cạnh bắt đầu của phần tử cha. "**Bắt đầu**" là cạnh trái của màn hình nếu tùy chọn là từ trái sang phải, hoặc là cạnh phải của màn hình nếu tùy chọn là từ phải sang trái.

5. Xóa thuộc tính **android:layout_weight="1"** khỏi TextView **show_count**, thuộc tính này không liên quan đến RelativeLayout.



Mã giải pháp

Mã XML trong **activity_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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id	btn_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_marginTop="52dp"
        android:background="@color/design_default_color_primary"
        android:onClick="showToast"
        android:textSize="24sp"
        android:text="@string/btn_label_toast"
        android:textColor="@android:color/white"    />

<TextView
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FFF000"
    android:gravity="center"
    android:text="@string/count_init_value"
    android:textColor="@color/design_default_color_primary"
    android:textSize="160sp"
    android:textStyle="bold"
    android:layout_below="@+id	btn_toast"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<Button
    android:id="@+id	btn_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="52dp"
    android:background="@color/design_default_color_primary"
    android:onClick="increaseCount"
    android:text="@string/btn_label_count"
    android:textColor="@color/white"
    android:textSize="24sp"
    android:layout_below="@+id/show_count"
    android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Tóm tắt

Sử dụng trình chỉnh sửa bộ cục để xem trước và tạo các biến thê:

- Để xem trước bố cục ứng dụng với hướng ngang trong trình chỉnh sửa bố cục, hãy nhấp vào nút **Orientation for Preview**  trên thanh công cụ trên cùng và chọn **Landscape**. Chọn **Portrait** để trở về hướng dọc.
- Để tạo biến thể của bố cục khác cho hướng ngang, hãy nhấp vào biểu tượng `activity_main.xml`  trên thanh công cụ. Chọn **Create Landscape Qualifier** từ menu thả xuống. Một cửa sổ trình chỉnh sửa mới sẽ mở ra với tab `land/activity_main.xml` hiển thị bố cục cho hướng ngang.
- Để xem trước bố cục cho các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị hoặc trình giả lập, hãy nhấp vào nút **Device for Preview** 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, nhấp vào `activity_main.xml` trên thanh công cụ và chọn **Create Tablet Qualifier** từ menu thả xuống. Một cửa sổ mới mở ra với tab `sw600dp/activity_main.xml` hiển thị bố cục cho thiết bị có kích thước máy tính bảng.

Sử dụng ConstraintLayout:

- Để xóa tất cả các ràng buộc trong bố cục với gốc ConstraintLayout, hãy nhấp vào nút **Clear All Constraints** trên thanh công cụ.
- Bạn có thể căn chỉnh một phần tử UI chứa văn bản, chẳng hạn như TextView hoặc Button, với một phần tử UI khác chứa văn bản. Một ràng buộc đường cơ sở cho phép bạn ràng buộc các phần tử sao cho các đường cơ sở văn bản khớp nhau.
- Để tạo ràng buộc đường cơ sở, hãy di con trỏ lên phần tử UI cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.

Sử dụng LinearLayout:

- LinearLayout là một ViewGroup sắp xếp bộ sưu tập các view của nó theo một hàng ngang hoặc dọc.
- LinearLayout phải có thuộc tính **layout_width**, **layout_height** và **orientation**.
- Thuộc tính **match_parent** cho **layout_width** hoặc **layout_height**: Mở rộng View để lấp đầy phần tử cha của nó theo chiều rộng hoặc chiều cao. Khi LinearLayout là View gốc, nó sẽ mở rộng đến kích thước của màn hình (View cha).

- Thuộc tính **wrap_content** cho **layout_width** hoặc **layout_height**: Thu nhỏ kích thước để View vừa đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, View sẽ trở nên vô hình.
- Số dp cố định 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 độc lập với mật độ.
- Thuộc tính **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 các thuộc tính **gravity** và **weight** cho phép bạn kiểm soát thêm việc sắp xếp các view và nội dung trong LinearLayout.
- Thuộc tính android:gravity chỉ định sự căn chỉnh nội dung của View bên trong chính View đó.
- Thuộc tính **android:layout_weight** chỉ ra không gian thừa trong LinearLayout sẽ được phân bổ cho View. Nếu chỉ một View có thuộc tính này, nó sẽ nhận được tất cả không gian màn hình thừa. Đối với nhiều phần tử View, không gian được chia tỷ lệ. Ví dụ: nếu hai phần tử Button mỗi phần tử có trọng số là 1 và TextView là 2, tổng cộng là 4, thì các phần tử Button mỗi phần tử sẽ nhận được $\frac{1}{4}$ không gian và TextView một nửa.

Sử dụng RelativeLayout:

- RelativeLayout là một ViewGroup trong đó mỗi view được định vị và căn chỉnh tương đối so với các view khác trong nhóm.
- Sử dụng **android:layout_alignParentTop** để căn chỉnh View vào đầu phần tử cha.
- Sử dụng **android:layout_alignParentLeft** để căn chỉnh View vào cạnh trái của phần tử cha.
- Sử dụng **android:layout_alignParentStart** để làm cho cạnh bắt đầu của View khớp với cạnh bắt đầu của phần tử cha. Thuộc tính này rất hữu ích nếu bạn muốn ứng dụng của mình hoạt động trên các thiết bị sử dụng các tùy chọn ngôn ngữ hoặc khu vực khác nhau. "Bắt đầu" là cạnh trái của màn hình nếu tùy chọn là từ trái sang phải, hoặc là cạnh phải của màn hình nếu tùy chọn là từ phải sang trái.

1.4) Văn bản và các chế độ 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. Bài thực hành này trình bày cách làm việc với nhiều phần tử TextView, bao gồm một phần tử mà người dùng có thể cuộn nội dung theo chiều dọc.

Nếu bạn có nhiều thông tin hơn mức vừa với màn hình của thiết bị, bạn có thể tạo một view cuộn 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 sử dụng view cuộn cho các tin tức, bài báo hoặc bất kỳ văn bản dài nào không hoàn toàn vừa trên màn hình. Bạn cũng có thể sử dụng view cuộn để cho phép người dùng nhập nhiều dòng văn bản hoặc để kết hợp các phần tử UI (chẳng hạn như một trường văn bản và một nút) bên trong một view cuộn. Lớp **ScrollView** cung cấp bố cục cho view cuộn. **ScrollView** là một lớp con của **FrameLayout**. Chỉ đặt một view làm con bên trong nó - một view con chứa toàn bộ nội dung để cuộn. View con này có thể là một **ViewGroup** (chẳng hạn như **LinearLayout**) chứa các phần tử UI.

Bố cục phức tạp có thể gặp phải các vấn đề về hiệu suất với các view con như hình ảnh. Một lựa chọn tốt cho View bên 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 phần tử UI đều ở trong bộ nhớ và trong cấu trúc view 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 tự do một cách mượt mà, bởi vì văn bản đã ở 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 của bạn. 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 biệt.

Những gì bạn nên biết

Bạn cần có khả năng:

- Tạo một ứng dụng Hello World bằng Android Studio.

- Chạy một ứng dụng trên trình giả lập hoặc thiết bị.
- Triển khai TextView trong bố cục cho một 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 View cuộn.
- Cách hiển thị văn bản tự do với một số thẻ định dạng HTML.
- Cách tạo kiểu cho màu nền và màu văn bản của TextView.
- Cách đưa một liên kết web vào văn bản.

Những gì bạn sẽ làm

- Tạo ứng dụng ScrollingText.
- Thay đổi ViewGroup ConstraintLayout thành RelativeLayout.
- Thêm hai phần tử TextView cho tiêu đề và phụ đề của bài viết.
- Sử dụng kiểu và màu TextAppearance cho tiêu đề và phụ đề 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 để dễ đọc.
- Thêm ScrollView vào bố cục để cho phép cuộn một phần tử TextView.
- Thêm thuộc tính autoLink để kích hoạt các URL trong văn bản và có thể nhấp vào.

Tổng quan ứng dụng

Ứng dụng **Scrolling Text** minh họa thành phần UI 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 một bài 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 bằng các thẻ HTML tối thiểu để đặt văn bản thành đậm hoặc nghiêng, và với các ký tự dòng mới để phân tách các đoạn văn. Bạn cũng có thể bao gồm các liên kết web hoạt động trong văn bản.

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

Trong bài thực hành này, bạn sẽ tạo một dự án Android cho ứng dụng **ScrollingText**, thêm các phần tử TextView vào bố cục cho tiêu đề và phụ đề của bài viết và thay đổi phần tử TextView “**Hello World**” hiện có để hiển thị một bài viết dài. Hình dưới đây là sơ đồ của bố cục. Bạn sẽ thực hiện tất cả những thay đổi này trong mã XML và trong tệp strings.xml. Bạn sẽ chỉnh sửa mã XML cho bố cục trong ngăn Text, mà bạn hiển thị bằng cách nhấp vào tab Text, thay vì nhấp vào tab Design cho ngăn Design. Một số thay đổi đối với các phần tử và thuộc tính UI dễ thực hiện trực tiếp hơn trong ngăn Text bằng cách sử dụng mã nguồn XML.

1.1 Tạo dự án và các phần tử TextView

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

1. Trong Android Studio, tạo dự án mới với các tham số sau:

Thuộc tính	Giá trị
Name	Scroll Text
Package name	com.example.scrolltext
Save application	<Đường dẫn nơi lưu dự án>
Language	Java
Minimum SDK	API 24 (“Mougat”; Android 7.0)

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

Cấu trúc View là ViewGroup ConstraintLayout:

```
<androidx.constraintlayout.widget.ConstraintLayout
```

3. Thay đổi ViewGroup này thành **RelativeLayout**. Dòng mã thứ hai sau khi sửa như sau:

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

RelativeLayout cho phép bạn đặt các phần tử UI tương đối với nhau hoặc tương đối với chính RelativeLayout cha.

Phần tử TextView "Hello World" mặc định được tạo vẫn có các thuộc tính ràng buộc (như **app:layout_constraintBottom_toBottomOf="parent"**).

4. Xóa dòng mã XML sau, dòng này liên quan đến ConstraintLayout:

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

Đoạn mã XML ở trên cùng bây giờ như sau:

```
<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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

5. Thêm một phần tử TextView phía trên TextView "Hello World" bằng cách nhập **<TextView**. Một khung TextView xuất hiện, kết thúc bằng **/>** và hiển thị các thuộc tính **layout_width** và **layout_height**, là các thuộc tính bắt buộc cho TextView.

6. Nhập các thuộc tính sau cho TextView. Khi nhập từng thuộc tính và giá trị, các đề xuất sẽ xuất hiện để hoàn thành tên hoặc giá trị thuộc tính.

Thuộc tính	Giá trị
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:id	"@+id/article_heading"
android:background	"@color/design_default_color_primary"
android:textColor	"@android:color/white"

android:padding	"10dp"
android:textAppearance	"@android:style/TextAppearance.DeviceDefault.Large"
android:textStyle	"bold"
android:text	"Article Title"

7. Trích xuất tài nguyên chuỗi cho chuỗi mã cứng của thuộc tính **android:text "Article Title"** trong TextView để tạo một mục nhập cho nó trong **strings.xml**.

Đặt con trỏ lên chuỗi mã cứng, nhấn **Alt-Enter (Option-Enter** trên Mac) và chọn **Extract string resource**. Chính sửa tên tài nguyên cho giá trị chuỗi thành **article_title**.

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

Đặt con trỏ lên chuỗi mã cứng, nhấn **Alt-Enter (Option-Enter** trên Mac) và chọn **Extract dimension resource**. Chính sửa Resource name thành **padding_regular**.

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

Thuộc tính	Giá trị
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:textAppearance	"@android:style/TextAppearance.DeviceDefault"
android:text	"Article Subtitle"

Vì bạn đã trích xuất tài nguyên kích thước 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 mã cứng của thuộc tính **android:text** "Article Subtitle" trong TextView thành **article_subtitle**.

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

```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="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	Giá trị
android:id	"@+id/article"
android:layout_below	"@+id/article_subheading"
android:lineSpacingExtra	"5sp"
android:padding	"@dimen/padding_regular"
android:text	"Article text"

13. Trích xuất tài nguyên chuỗi cho "Article text" thành **article_text** và trích xuất tài nguyên kích thước cho "5sp" thành **line_spacing**.

14. Định dạng lại và căn chỉnh mã bằng cách nhấn tổ hợp **Shift+Alt+F**.

1.2 Thêm văn bản cho bài báo

Trong một ứng dụng thực tế truy cập các bài báo tạp chí hoặc báo chí, các bài báo xuất hiện có thể đến từ một nguồn trực tuyến thông qua một **content provider** hoặc có thể được lưu trước trong một cơ sở dữ liệu trên thiết bị.

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

1. Trong thư mục **app > res > values**, mở **strings.xml**.
2. Mở bất kỳ tệp văn bản nào có một lượng lớn văn bản.

3. Nhập các giá trị cho các chuỗi **article_title** và **article_subtitle** với tiêu đề và phụ đề do bạn tự tạo. Hãy tạo các giá trị chuỗi thành văn bản một dòng mà không có thẻ HTML hoặc nhiều dòng.

4. Nhập hoặc sao chép và dán văn bản cho chuỗi **article_text**.

Bạn có thể sử dụng văn bản trong tệp văn bản của mình. Yêu cầu duy nhất cho tác vụ này là văn bản phải đủ dài để nó không vừa trên màn hình.

Hãy nhớ những điều sau:

- Khi bạn nhập hoặc dán văn bản trong tệp **strings.xml**, các dòng văn bản không tự động xuống dòng mà kéo dài ra ngoài lề phải. Đây là hành vi chính xác - mỗi dòng văn bản mới bắt đầu từ lề trái đại diện cho một đoạn văn hoàn chỉnh. Nếu bạn muốn văn bản trong **strings.xml** được xuống dòng, bạn có thể nhấn phím **Enter** để nhập các ký tự kết thúc dòng cứng hoặc định dạng văn bản trước trong một trình soạn thảo văn bản với các ký tự kết thúc dòng cứng.
- Nhập **\n** để biểu thị phần cuối của một dòng và một **\n** khác để biểu thị một dòng trống. Bạn cần thêm các ký tự kết thúc dòng để ngăn các đoạn văn chạy vào nhau.
- Nếu bạn có dấu nháy đơn (‘) trong văn bản, bạn phải thoát nó bằng cách đặt một dấu gạch chéo ngược (\') trước nó. Nếu bạn có dấu ngoặc kép trong văn bản của mình, bạn cũng phải thoát nó (”). Bạn cũng phải thoát bất kỳ ký tự không phải ASCII nào khác.
- Nhập các thẻ HTML **** và **** xung quanh các từ cần được in đậm.
- Nhập các thẻ HTML **<i>** và **</i>** xung quanh các từ cần được in nghiêng. Nếu bạn sử dụng dấu nháy đơn uốn cong trong một cụm từ in 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 quanh toàn bộ văn bản bên trong **<string name="article_text"> </string>** trong tệp **strings.xml**.
- Bao gồm một liên kết web để kiểm tra, chẳng hạn như www.google.com. (Ví dụ: <https://cafef.vn>) Không sử dụng thẻ HTML, vì mọi thẻ HTML ngoại trừ thẻ in đậm và in nghiêng đều bị bỏ qua và được trình bày dưới dạng văn bản, điều này không phải là điều bạn muốn.

```
</> strings.xml × : 

1 <resources>
2     <string name="app_name">Scroll Text</string>
3     <string name="article_title">Article Title</string>
4     <string name="article_subtitle">Article Subtitle</string>
5     <string name="article_text">Cuộc sống là một hành trình dài với n
6 \n
7 <b>Thành công</b> không bao giờ đến dễ dàng. Nó đòi hỏi sự kiên trì,
8 <i>Hãy luôn nhớ rằng, "Không có áp lực, không có kim cương."</i> Nh
9 \n
10 Có người từng nói: "Cuộc sống là một hành trình, không phải đích đến
11 Dù bạn đang ở đâu, dù bạn đang làm gì, đừng bao giờ dừng lại.\n
12 <b><i>Hãy luôn tiến về phía trước</i></b>, bởi vì cơ hội sẽ không tự
13 \n
14 Thế giới đang thay đổi từng ngày. Những gì bạn biết hôm nay có thể tr
15 Chính vì vậy, việc học hỏi và phát triển bản thân là vô cùng quan trọng
16 Người thành công không phải là người biết tất cả, mà là người không n
17 \n
```

1.3 Chạy ứng dụng

Chạy ứng dụng. Bài báo xuất hiện, nhưng người dùng không thể cuộn bài báo vì bạn chưa bao gồm **ScrollView**. Lưu ý rằng việc nhấn vào một liên kết web hiện tại không thực hiện bất cứ điều gì.



Mã giải pháp

Tệp bố cục **activity_main.xml** như sau:

```
<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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```

        android:id="@+id/article_heading"
        android:background="@color/design_default_color_primary"
        android:textColor="@color/white"
        android:padding="@dimen/padding_regular"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
        android:textStyle="bold"
        android:text="@string/article_title" />

<TextView
    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:textAppearance="@android:style/TextAppearance.DeviceDefault"
    android:text="@string/article_subtitle" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/article"
    android:layout_below="@id/article_subheading"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />

</RelativeLayout>

```

Task 2: Thêm ScrollView và một liên kết web

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

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

2.1 Thêm thuộc tính autoLink cho các liên kết web

Thêm thuộc tính `android:autoLink="web"` vào TextView `article`. Mã XML cho TextView này như sau:

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/article"  
    android:autoLink="web"  
    android:layout_below="@+id/article_subheading"  
    android:lineSpacingExtra="5sp"  
    android:padding="10dp"  
    android:text="@string/article_text" />
```

2.2 Thêm ScrollView vào bộ cục

Để làm cho một View (chẳng hạn như TextView) có thể cuộn được, hãy nhúng View đó bên trong một ScrollView.

1. Thêm ScrollView giữa TextView `article_subheading` và TextView `article`. Khi bạn nhập `<ScrollView`, Android Studio đưa ra các thuộc tính `android:layout_width` và `android:layout_height` với các đề xuất.
2. Chọn `wrap_content` từ các đề xuất cho cả hai thuộc tính. Mã cho hai phần tử TextView và ScrollView bây giờ như sau:

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:id="@+id/article_subheading"  
    android:layout_below="@+id/article_heading"  
    android:padding="10dp"  
    android:textAppearance="@android:style/TextAppearance."  
    android:text="Article Subtitle" />  
  
<ScrollView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" ></ScrollView>  
  
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/article"
```

3. Di chuyển mã kết thúc </ScrollView> sau TextView **article** sao cho các thuộc tính của TextView **article** nằm hoàn toàn bên trong **ScrollView**.

4. Xóa thuộc tính sau khỏi TextView **article** và thêm nó vào ScrollView:

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

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

5. Nhập vào tab **Preview** ở phía bên phải của trình chỉnh sửa bố cục để xem trước bố cục.

2.3 Chạy ứng dụng

Để kiểm tra cách văn bản cuộn:

1. Chạy ứng dụng trên thiết bị hoặc trình giả lập.

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

Nhấn vào liên kết web để truy cập trang web. Thuộc tính **android:autoLink** biến bất kỳ URL nào có thể nhận dạng trong TextView (chẳng hạn như <https://cafef.vn>) thành liên kết web.

2. Xoay thiết bị hoặc trình giả lập 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.

3. 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.

Task 3: Cuộn nhiều phần tử

Như đã lưu ý trước đó, một ScrollView chỉ có thể chứa một View con (chẳng hạn như TextView bài viết 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 lồng một ViewGroup như LinearLayout bên trong 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 một LinearLayout bên trong ScrollView và di chuyển tiêu đề phụ và bài viết vào LinearLayout. LinearLayout trở thành View con duy nhất trong ScrollView như hình dưới đây 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 TextView **article** bên 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** với các gợi ý. Chọn **match_parent** và **wrap_content** từ các gợi ý cho chiều rộng và chiều cao của nó, tương ứng. Mã ở đầu ScrollView bây giờ:

```
<ScrollView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/article_subheading" >  
  
    <LinearLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content" ></LinearLayout>  
  
    <TextView  
        android:id="@+id/article"  
        android:layout_width="wrap_content"
```

Bạn sử dụng **match_parent** để khớp với chiều rộng của ViewGroup cha. Bạn sử dụng **wrap_content** để thay đổi kích thước LinearLayout sao cho nó vừa đủ lớn để bao quanh nội dung của nó.

3. Di chuyển mã kết thúc **</LinearLayout>** sau TextView **article** nhưng trước thẻ đóng **</ScrollView>**.

LinearLayout bây giờ bao gồm TextView bài viết và hoàn toàn nằm bên trong ScrollView.

4. Thêm thuộc tính **android:orientation="vertical"** vào LinearLayout để đặt hướng của nó thành dọc.

LinearLayout như sau:

```

<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/article_subheading">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/article"
            android:layout_width="wrap_content"

```

3.2 Di chuyển các phần tử UI bên trong LinearLayout

LinearLayout bây giờ chỉ có một phần tử UI - TextView **article**. Bạn muốn bao gồm TextView **article_subheading** trong LinearLayout để cả hai sẽ cuộn.

1. Để di chuyển TextView **article_subheading**, hãy chọn mã, chọn **Edit > Cut**, nhấp vào phía trên TextView bài viết bên trong LinearLayout và chọn **Edit > Paste**.
2. Xóa thuộc tính **android:layout_below="@+id/article_heading"** khỏi TextView **article_subheading**. Vì TextView này hiện nằm trong LinearLayout, thuộc tính này sẽ xung đột với các thuộc tính LinearLayout.
3. Thay đổi thuộc tính của ScrollView từ **android:layout_below="@+id/article_subheading"** thành **android:layout_below="@+id/article_heading"**. Giờ đây, tiêu đề phụ là một phần của LinearLayout, ScrollView phải được đặt bên dưới tiêu đề, không phải tiêu đề phụ.

Mã XML cho ScrollView bây giờ như sau:

```

<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/article_heading">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

```

```

<TextView
    android:id="@+id/article_subheading"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_subtitle"
    android:textAppearance=
        "@android:style/TextAppearance.DeviceDefault" />

<TextView
    android:id="@+id/article"
    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 nhận thấy rằng tiêu đề phụ hiện cuộn cùng với bài viết trong khi tiêu đề vẫn ở đúng vị trí.

Tóm tắt

- Sử dụng ScrollView để cuộn một View con duy nhất (chẳng hạn như TextView). Một ScrollView chỉ có thể chứa một View con hoặc ViewGroup.
- Sử dụng ViewGroup như LinearLayout làm View con bên trong ScrollView để cuộn nhiều hơn một phần tử View. Đặt các phần tử bên trong LinearLayout.
- Hiển thị văn bản dạng tự do trong TextView với các thẻ định dạng HTML cho in đậm và in nghiêng.

- Sử dụng \n làm ký tự kết thúc dòng trong văn bản dạng tự do để ngăn đoạn văn tràn sang đoạn văn tiếp theo.
- Sử dụng thuộc tính android:autoLink="web" để làm cho các liên kết web trong văn bản có thể nhấp được

1.5) Tài nguyên có sẵn

Giới thiệu

Những gì bạn nên biết:

Bạn nên có khả năng:

- Hiểu quy trình làm việc cơ bản của Android Studio.
- Tạo một ứng dụng từ đầu bằng cách sử dụng mẫu Empty Activity.
- Sử dụng trình chỉnh sửa bộ cục.

Những gì bạn sẽ học:

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

Những gì bạn sẽ làm:

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

Tổng quan ứng dụng

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

Task 1: Thay đổi biểu tượng launcher

Mỗi ứng dụng mới bạn tạo bằng Android Studio đều bắt đầu bằng một biểu tượng launcher tiêu chuẩn đại diện cho ứng dụng đó. Biểu tượng launcher 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 launcher sẽ xuất hiện trên thiết bị ở nhiều nơi khác nhau, bao gồm màn hình chính và màn hình Search Apps. Ví dụ: ứng dụng HelloToast xuất hiện trong màn hình Search Apps của trình giả lập với biểu tượng tiêu chuẩn cho các dự án ứng dụng mới, như hình dưới đây.

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

1.1 Khám phá tài liệu Android chính thức

Bạn có thể tìm thấy tài liệu dành cho nhà phát triển Android chính thức tại developer.android.com.

Tài liệu này chứa rất nhiều thông tin được Google cập nhật liên tục.

1. Truy cập developer.android.com/design/

Phần này nói về Material Design, một triết lý thiết kế khái niệm vạch ra cách các ứng dụng nên trông và hoạt động trên các 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.

2. Truy cập developer.android.com/docs/ để tìm thông tin API, tài liệu tham khảo, hướng dẫn, hướng dẫn sử dụng công cụ và các mẫu mã.

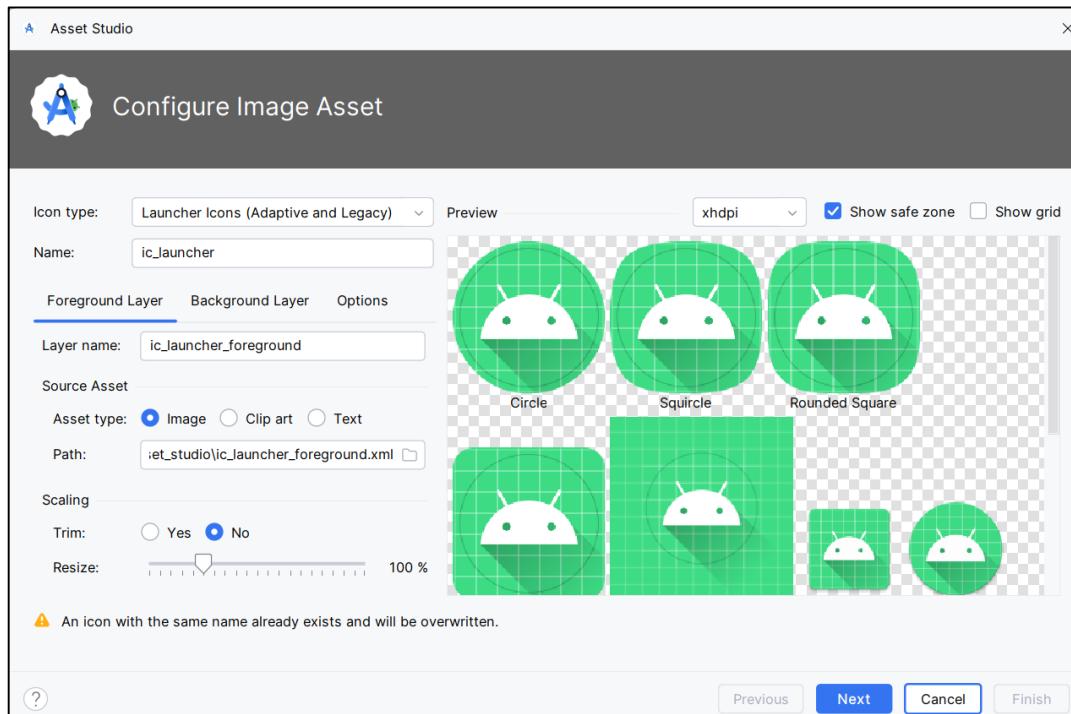
3. Truy cập developer.android.com/distribute/ để tìm thông tin về cách đưa ứng dụng lên Google Play, hệ thống phân phối kỹ thuật số của Google cho các ứng dụng được phát triển bằng Android SDK. Sử dụng Google Play Console để phát triển cơ sở người dùng của bạn và bắt đầu kiếm tiền.

1.2 Thêm một tài nguyên hình ảnh cho biểu tượng launcher

Để thêm một hình ảnh clip-art làm biểu tượng launcher, hãy làm theo các bước sau:

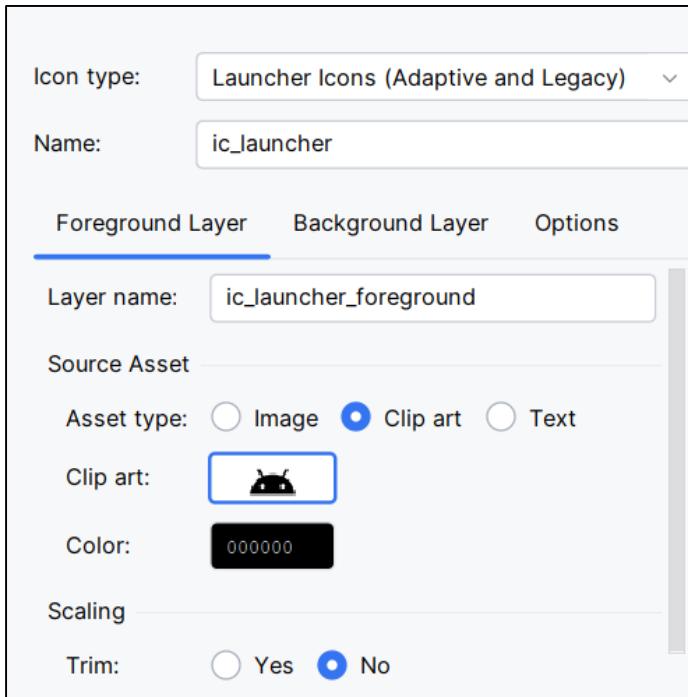
1. Mở dự án ứng dụng HelloToast từ bài học trước về cách sử dụng trình chỉnh sửa bộ cục hoặc tạo một dự án ứng dụng mới.

2. Trong **Project > Android**, nhấp chuột phải vào thư mục **res** và chọn **New > Image Asset**. Cửa sổ **Configure Image Asset** sẽ xuất hiện.



3. Trong trường **Icon Type**, chọn **Launcher Icons (Adaptive & Legacy)** nếu nó chưa được chọn.

4. Nhấp vào tab **Foreground Layer**, chọn **Clip Art**.



5. Nhập vào biểu tượng trong trường **Clip Art**. Các biểu tượng xuất hiện từ bộ biểu tượng **Material Design**.
6. Duyệt cửa sổ Select Icon, chọn một biểu tượng thích hợp, sau đó nhấp vào **OK**.
7. Nhấp vào tab **Background Layer**, chọn **Color** làm **Asset Type**, sau đó nhấp vào ô màu để chọn màu sử dụng làm lớp nền.
8. Nhấp vào tab **Legacy** và xem xét các cài đặt mặc định. Xác nhận rằng bạn muốn tạo các biểu tượng kê thừa, tròn và biểu tượng Google Play Store. Nhấp vào Next khi hoàn tất.
9. Chạy ứng dụng.

Android Studio tự động thêm các hình ảnh launcher vào các thư mục mipmap cho các mật độ khác nhau.

Task 2: Sử dụng các mẫu dự án

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

Mỗi mẫu bao gồm một hoạt động khung và giao diện người dùng. Bạn đã sử dụng mẫu Empty Activity. Mẫu Basic Activity có nhiều tính năng hơn và kết hợp các tính năng ứng dụng được đề xuất, chẳng hạn như menu tùy chọn xuất hiện trên thanh ứng dụng.

2.1 Khám phá kiến trúc Basic Activity

Mẫu **Basic Views Activity** là một mẫu linh hoạt do Android Studio cung cấp để hỗ trợ bạn bắt đầu nhanh chóng việc phát triển ứng dụng.

1. Trong Android Studio, tạo một dự án mới với mẫu Basic Views Activity.
2. Xây dựng và chạy ứng dụng.
3. Xác định các phần được gắn nhãn trong hình và bảng dưới đây. Tìm các phần tương đương trên thiết bị hoặc màn hình trình giả lập của bạn. Kiểm tra mã Java và các tệp XML tương ứng 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.

2.2 Tùy chỉnh ứng dụng được tạo bởi mẫu

Thay đổi giao diện của ứng dụng được tạo bởi mẫu Basic Views Activity. Ví dụ: bạn có thể thay đổi màu của thanh ứng dụng để khớp với thanh trạng thái. Bạn cũng có thể muộn xóa nút hành động nổi nếu bạn không định sử dụng nó.

1. Thay đổi màu của thanh ứng dụng (Toolbar) trong activity_main.xml bằng cách thay đổi android:background thành "?attr/colorPrimaryDark", điều này sẽ đặt màu thanh ứng dụng thành màu chính đậm 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 gốc trong **onCreate()** đặt trình nghe **onClick()** cho nút. Mở MainActivity và xóa khỏi mã sau:

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "Hello", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();
    }
});
```

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

```
<com.google.android.material.floatingactionbutton.FloatingActionButton  
    android:id="@+id/fab"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="bottom|end"  
    app:srcCompat="@android:drawable/ic_dialog_email" />
```

4. Thay đổi tên của ứng dụng được hiển thị trên thanh ứng dụng bằng cách thay đổi tài nguyên chuỗi **app_name** trong **strings.xml**.

5. Chạy ứng dụng.

2.3 Khám phá cách thêm hoạt động bằng cách sử dụng các mẫu

Cho đến nay, đối với các bài thực hành, bạn đã sử dụng các mẫu **Empty Views Activity** và **Basic Views Activity**. Trong các bài học sau, các mẫu bạn sử dụng sẽ khác nhau, tùy thuộc vào task.

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.

1. Tạo một dự án ứng dụng mới hoặc chọn một dự án hiện có.
2. Trong khung **Project > Android**, nhấp chuột phải vào thư mục **java**.
3. Chọn **New > Activity > Gallery**.
4. Thêm một **Activity**. Ví dụ: nhấp vào Navigation Drawer Views Activity để thêm activity có ngăn điều hướng vào ứng dụng của bạn.
5. Nhấp đúp vào các tệp bô cục cho Activity để hiển thị chúng trong trình chỉnh sửa bô cục.

Task 3: Tìm hiểu từ mã ví dụ

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

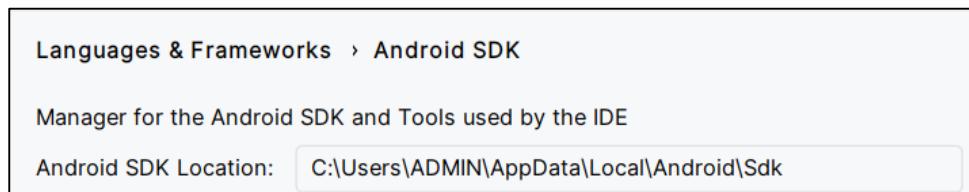
Bạn có thể khám phá hàng trăm mẫu mã trực tiếp từ bên trong Android Studio.

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

3.2 Sử dụng SDK Manager để cài đặt tài liệu ngoại tuyến

Cài đặt Android Studio cũng cài đặt các yêu tố cần thiết của Android SDK. Tuy nhiên, có sẵn các thư viện và tài liệu bổ sung và bạn có thể cài đặt chúng bằng SDK Manager.

1. Chọn **Tools > SDK Manager**.
2. Trong cột bên trái, nhấp vào **Android SDK**.
3. Chọn và sao chép đường dẫn cho **Android SDK Location** ở đầu màn hình, vì bạn sẽ cần nó để định vị tài liệu trên máy tính của mình:



4. Nhấp vào tab **SDK Platforms**. Bạn có thể cài đặt các phiên bản bổ sung của hệ thống Android từ đây.
5. Nhấp vào tab **SDK Update Sites**. Android Studio thường xuyên kiểm tra các trang web được liệt kê và chọn để tìm các bản cập nhật.
6. Nhấp vào tab **SDK Tools**. 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. Khi quá trình cài đặt hoàn tất, hãy nhấp vào **Finish**.
8. Điều hướng đến thư mục sdk bạn đã sao chép ở trên và mở thư mục docs.
9. Tìm index.html và mở nó.

Task 4: Nhiều tài nguyên hơn

- Kênh **Android Developer** trên YouTube là một nguồn tuyệt vời về hướng dẫn và mẹo
- **Nhóm Android** đăng tin tức và mẹo trên blog Android chính thức.
- **Stack Overflow** là một cộng đồng các lập trình viên giúp đỡ lẫn nhau. Nếu bạn gặp phải vấn đề, rất có thể 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 "Các lỗi rò rỉ bộ nhớ phổ biến nhất trong 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 liên quan từ tất cả các tài nguyên này. Ví dụ: "Phiên bản Android OS phổ biến nhất ở Ấn Độ là gì?"

4.1 Tìm kiếm trên Stack Overflow bằng cách sử dụng các thẻ

Truy cập **Stack Overflow** và nhập [android] vào hộp tìm kiếm. Dấu ngoặc vuông [] cho biết rằng bạn muốn tìm kiếm các bài đăng đã được gắn thẻ là về Android.

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

- [android] và [layout]
- [android] "hello world"

Để tìm hiểu thêm về nhiều cách bạn có thể tìm kiếm trên Stack Overflow, hãy xem trung tâm trợ giúp **Stack Overflow**.

Tóm tắt

- Tài liệu Nhà phát triển Android chính thức: developer.android.com
- **Material Design** là một triết lý thiết kế khái niệm vạch ra cách các ứng dụng nên trông và hoạt động trên các 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 Android SDK

- Android Studio cung cấp các mẫu cho các thiết kế ứng dụng và hoạt động phổ biến và được đề xuất. Các mẫu này cung cấp mã hoạt động cho các trường hợp sử dụng phổ biến.
- Khi bạn tạo một dự án, bạn có thể chọn một mẫu cho hoạt động đầu tiên của mình.
- Android Studio chứa nhiều mẫu mã mà bạn có thể nghiên cứu, sao chép và kết hợp với các dự án của mình.

Bài 2) Activities

2.1) Activity và Intent

Giới thiệu

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

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

Mỗi khi một activity mới bắt đầu, activity trước đó sẽ bị dừng lại, nhưng hệ thống vẫn giữ activity đó trong một ngăn xếp. Khi một activity mới bắt đầu, activity mới đó sẽ được đẩy vào ngăn xếp lịch sử và chiếm quyền kiểm soát người dùng. Ngăn xếp lịch sử tuân theo logic ngăn xếp cơ bản "**vào sau, ra trước**". Khi người dùng hoàn thành activity hiện tại và nhấn nút **Back**, activity đó sẽ bị lấy ra khỏi ngăn xếp và bị hủy, đồng thời activity trước đó tiếp tục hoạt động.

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

Một Intent có thể là tường minh hoặc ngầm định:

- Một intent tường minh là intent mà bạn biết mục tiêu của intent đó. Tức là bạn đã biết tên lớp đủ điều kiện của activity cụ thể đó.

- Một intent ngầm định là intent mà bạn không có tên của thành phần mục tiêu, nhưng bạn có một hành động chung để thực hiện.

Trong bài thực hành này, bạn sẽ tạo các intent tường minh. Bạn sẽ tìm hiểu cách sử dụng các intent ngầm định trong một bài thực hành sau.

Những gì bạn nên biết

Bạn nên biết:

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

Những gì bạn sẽ học

- Cách tạo một Activity mới trong Android Studio.
- Cách xác định các activity cha và con để điều hướng Up.
- Cách khởi động một Activity bằng một Intent tường minh.
- Cách truyền dữ liệu giữa mỗi Activity bằng một Intent tường minh.

Những gì bạn sẽ làm

- Tạo một ứng dụng Android mới với một Activity chính và một Activity thứ hai.
- Truyền một số dữ liệu (một chuỗi) từ Activity chính đến Activity thứ hai bằng một Intent và hiển thị dữ liệu đó trong Activity thứ hai.
- Gửi một phần dữ liệu khác trở lại Activity chính, cũng bằng một Intent.

Tổng quan ứng dụng

Trong chương này, bạn sẽ tạo và xây dựng một ứng dụng có tên **Two Activities**, một cách không ngạc nhiên, chứa hai triển khai Activity. Bạn xây dựng ứng dụng trong ba giai đoạn.

Trong giai đoạn đầu tiên, bạn tạo một ứng dụng có activity chính chứa một nút **Send**. Khi người dùng nhấp vào nút này, activity chính của bạn sẽ sử dụng một intent để khởi động activity thứ hai.

Trong giai đoạn thứ hai, bạn thêm một view **EditText** vào activity chính. Người dùng nhập một tin nhắn và nhấp vào **Send**. Activity chính sử dụng một intent để khởi động activity thứ hai và gửi tin nhắn của người dùng đến activity thứ hai. Activity thứ hai hiển thị tin nhắn mà nó nhận được.

Trong giai đoạn cuối cùng của việc tạo ứng dụng **Two Activities**, bạn thêm một **EditText** và một nút **Reply** vào activity thứ hai. Giờ đây, người dùng có thể nhập tin nhắn trả lời và nhấn **Reply**, và câu trả lời sẽ được hiển thị trên activity chính. Tại thời điểm này, bạn sử dụng một intent để truyền câu trả lời từ activity thứ hai trở lại activity chính.

Task 1: Tạo dự án TwoActivities

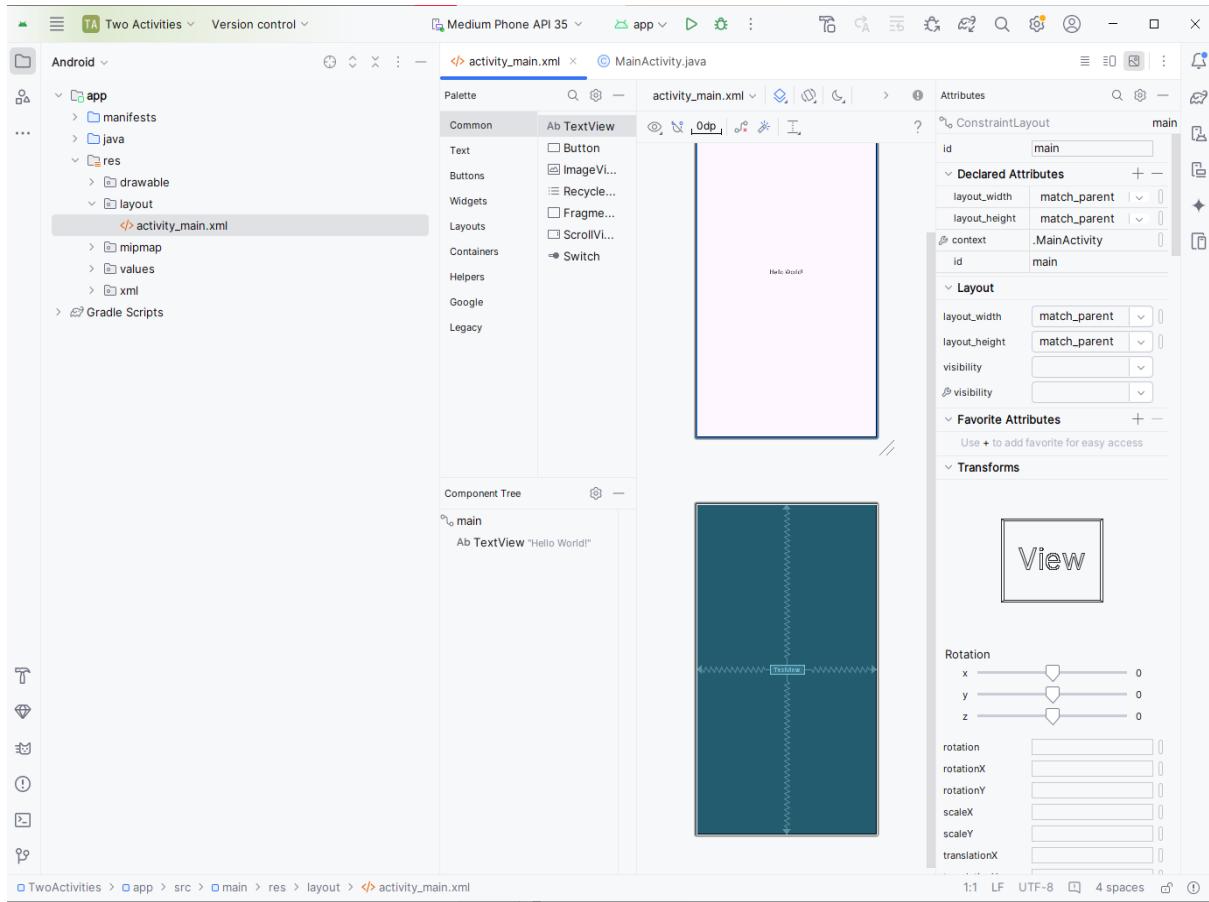
Trong task này, bạn thiết lập dự án ban đầu với một Activity chính, xác định bố cục và xác định một phương thức khung cho sự kiện nút **onClick**.

1.1 Tạo dự án TwoActivities

1. Khởi động Android Studio và tạo một dự án Android Studio mới.
2. Chọn **Empty Views Activity** cho mẫu Activity. Nhấp vào **Next**.
3. Đặt tên cho ứng dụng của bạn là **Two Activities** và chọn cùng các cài đặt Phone and Tablet mà bạn đã sử dụng trong các bài thực hành trước. Thư mục dự án tự động có tên là TwoActivities, và tên ứng dụng xuất hiện trong thanh ứng dụng sẽ là "**Two Activities**".
4. Nhấp vào **Finish**.

1.2 Xác định bố cục cho Activity chính

1. Mở **res > layout > activity_main.xml** trong **Project > Android**. Trình chỉnh sửa bố cục sẽ xuất hiện.
2. Nhấp vào tab **Design** nếu nó chưa được chọn và xóa **TextView** trong ngăn **Component Tree**.



3. Với Autoconnect được bật, hãy kéo một **Button** từ ngăn **Palette** đến góc dưới bên phải của bố cục. Tự động kết nối sẽ tạo các ràng buộc cho Button.
4. Trong ngăn **Attributes**, đặt ID thành **btn_main**, **layout_width** và **layout_height** thành **wrap_content** và nhập **Send** cho trường **Text**. Bố cục hiện tại sẽ như thế này:



5. Nhập vào tab Text để chỉnh sửa mã XML. Thêm thuộc tính sau vào Button:

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

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

6. Trích xuất tài nguyên chuỗi, như đã mô tả trong một bài thực hành trước, cho "Send" và sử dụng tên **btn_main** cho tài nguyên.

Mã XML cho Button phải như sau:

```
<Button
    android:id="@+id	btn_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="16dp"
    android:text="@string/btn_main"
    android:onClick="launchSecondActivity"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent" />
```

1.3 Xác định hành động Button

Trong task này, bạn triển khai phương thức **launchSecondActivity()** mà bạn đã đề cập trong bước 5 cho thuộc tính **android:onClick**.

1. Nhập vào "launchSecondActivity" trong mã XML **activity_main.xml**.
 2. Nhấn **Alt+Enter** trên Windows (**Option+Enter** trên máy Mac) và chọn **Create 'launchSecondActivity(View)' in 'MainActivity'**.
- Tệp MainActivity sẽ mở ra và Android Studio sẽ tạo một phương thức khung cho trình xử lý **launchSecondActivity()**.
3. Bên trong **launchSecondActivity()**, thêm một câu lệnh Log "Button Clicked!".

```
Log.d(LOG_TAG, msg: "Button clicked!");
```

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

4. Ở đầu lớp MainActivity, hãy thêm một hằng số cho biến **LOG_TAG**:

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

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

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

Task 2: Tạo và khởi chạy Activity thứ hai

Mỗi activity mới bạn thêm vào dự án của mình đều có tệp bố cục và tệp Java riêng, tách biệt với activity chính. Chúng cũng có các phần tử `<activity>` riêng trong tệp **AndroidManifest.xml**.

Như activity chính, các triển khai activity mới mà bạn tạo trong Android Studio cũng mở rộng từ lớp **AppCompatActivity**.

Mỗi activity trong ứng dụng của bạn chỉ được kết nối lồng lěu với các activity khác. Tuy nhiên, bạn có thể xác định một activity là cha của một activity khác trong tệp **AndroidManifest.xml**. Mỗi quan hệ cha-con này cho phép Android thêm các gợi ý điều hướng như mũi tên hướng sang trái trong thanh tiêu đề cho mỗi activity.

Một activity giao tiếp với các activity khác (trong cùng một ứng dụng và trên các ứng dụng khác nhau) bằng một intent. Một Intent có thể là tường minh hoặc ngầm định:

- Một intent tường minh là intent mà bạn biết mục tiêu của intent đó; tức là bạn đã biết tên lớp đủ điều kiện của activity cụ thể đó.
- Một intent ngầm định là intent mà bạn không có tên của thành phần mục tiêu, nhưng có một hành động chung để thực hiện.

Trong task này, bạn sẽ thêm một activity thứ hai vào ứng dụng của chúng ta, với bố cục riêng của nó. Bạn sửa đổi tệp **AndroidManifest.xml** để xác định activity chính là cha của activity thứ hai. Sau đó, bạn sửa đổi phương thức **launchSecondActivity()** trong **MainActivity** để bao gồm một intent khởi động activity thứ hai khi bạn nhấp vào nút.

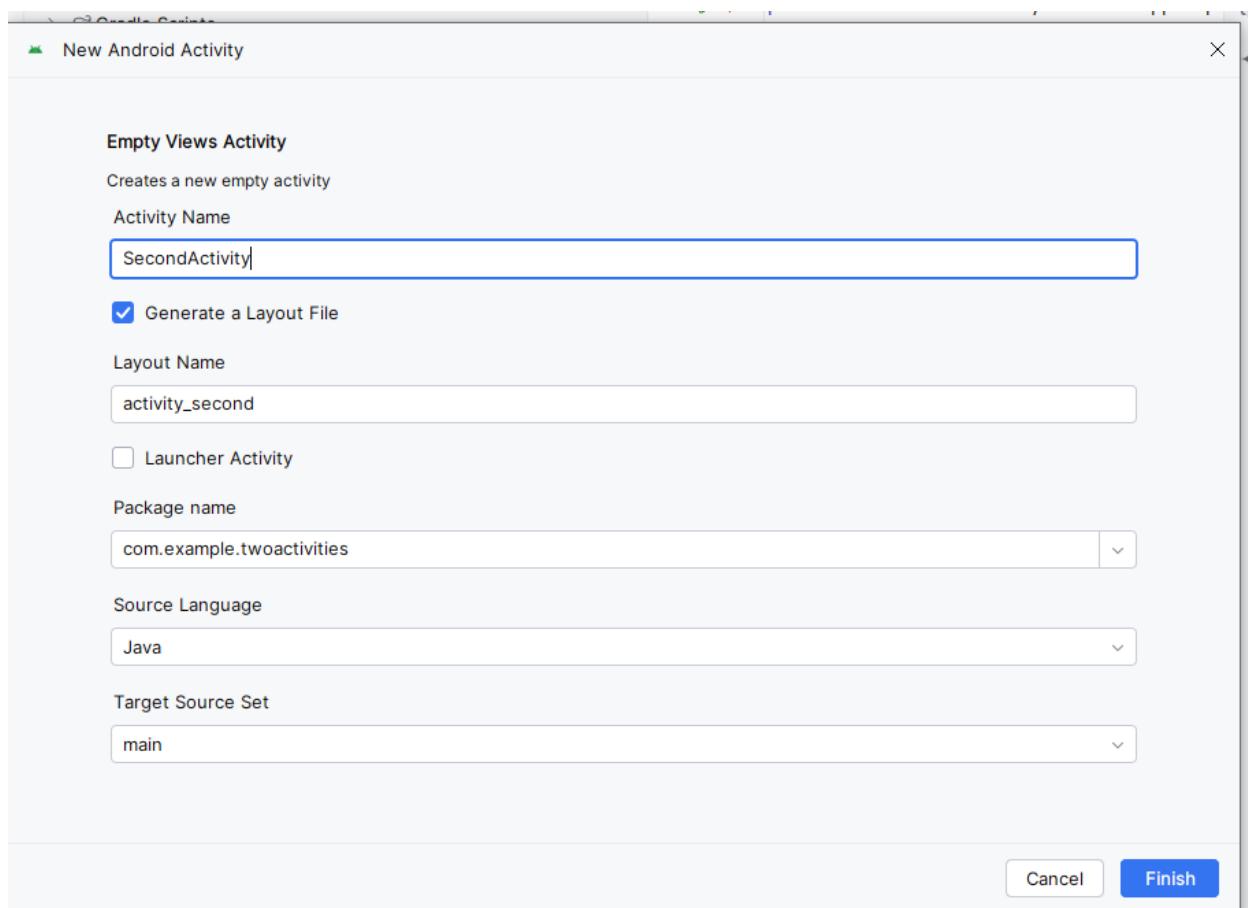
2.1 Tạo Activity thứ hai

1. Nhập vào thư mục **app** cho dự án của bạn và chọn **File > New > Activity > Empty Views Activity**.

2. Đặt tên cho Activity mới là **SecondActivity**. Tên bối cục được tự động điền là **activity_second**.

Không chọn tùy chọn **Launcher Activity**.

3. Nhập vào **Finish**. Android Studio tự động thêm cả bối cục Activity mới (**activity_second.xml**) và 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 tệp AndroidManifest.xml

1. Mở **manifests > AndroidManifest.xml**.

2. Tìm phần tử **<activity>** mà Android Studio đã tạo cho Activity thứ hai.

```

<activity
    android:name=".SecondActivity"
    android:exported="false" />
<activity>
```

3. Thay thế toàn bộ phần tử <activity> bằng nội dung sau:

```

<activity
    android:name=".SecondActivity"
    android:exported="false"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.example.twoactivities.MainActivity"/>
</activity>
    . . .
```

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

Với thuộc tính **parentActivityName**, bạn chỉ ra rằng activity chính là cha của activity thứ hai. Mỗi quan hệ này được sử dụng để điều hướng Up trong ứng dụng của bạn: thanh ứng dụng cho activity thứ hai sẽ có một mũi tên hướng sang trái để người dùng có thể điều hướng đến activity chính.

Với phần tử <meta-data>, bạn cung cấp thông tin tùy ý bổ sung về activity dưới dạng các cặp **khóa-giá trị**. Trong trường hợp này, các thuộc tính metadata thực hiện cùng một việc với thuộc tính **android:parentActivityName** - chúng xác định mối quan hệ giữa hai activity để điều hướng. Các thuộc tính **metadata** này là bắt buộc đối với các phiên bản Android cũ hơn, vì thuộc tính **android:parentActivityName** chỉ khả dụng cho API cấp 16 trở lên.

4. Trích xuất một tài nguyên chuỗi cho "**Second Activity**" trong mã trên và sử dụng **activity2_name** làm tên tài nguyên.

2.3 Xác định bối cảnh cho Activity thứ hai

1. Mở **activity_second.xml** và nhấp vào tab **Design** nếu nó chưa được chọn.
2. Kéo một **TextView** từ ngăn **Palette** đến góc trên bên trái của bối cảnh và thêm các ràng buộc vào cạnh trên và cạnh trái của bối cảnh. Đặt các thuộc tính của nó trong ngăn Attributes như sau:

Thuộc tính	Giá trị
id	text_header
Top margin	16dp
Left margin	16dp
layout_width	wrap_content
layout_height	wrap_content
text	Message Received
textAppearance	AppCompat.Medium
textStyle	bold

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

Bố cục bây giờ sẽ như thế này:



3. Nhấp vào tab **Text** để chỉnh sửa mã XML và trích xuất chuỗi "**Message Received**" thành một tài nguyên có tên là **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.

2.4 Thêm Intent vào Activity chính

Trong task này, bạn thêm một Intent tường minh vào Activity chính. Intent này được sử dụng để kích hoạt Activity thứ hai khi nút **Send** được nhấp.

1. Mở **MainActivity**.

2. Tạo một Intent mới trong phương thức **launchSecondActivity()**.

```
Intent intent =  
    new Intent( packageContext: this, SecondActivity.class);
```

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

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

```
startActivity(intent);
```

4. Chạy ứng dụng.

Khi bạn nhấp vào nút **Send**, MainActivity sẽ gửi Intent và hệ thống Android sẽ khởi chạy SecondActivity, xuất hiện trên màn hình. Để quay lại MainActivity, hãy nhấp vào nút **Back** ở cuối màn hình.

Task 3: Gửi dữ liệu từ Activity chính đến Activity thứ hai

Trong task trước, bạn đã thêm một intent tường minh vào MainActivity để khởi chạy SecondActivity. Bạn cũng có thể sử dụng intent để gửi dữ liệu từ một activity sang một activity khác trong khi khởi chạy nó.

Đối tượng intent của bạn có thể truyền dữ liệu đến activity đích theo hai cách: trong trường dữ liệu hoặc trong intent extras. Dữ liệu intent là một URI cho biết dữ liệu cụ thể cần được thực hiện. Nếu thông tin bạn muốn truyền đến một activity thông qua một intent không phải là URI hoặc bạn có nhiều hơn một mảng thông tin bạn muốn gửi, bạn có thể đặt thông tin bổ sung đó vào intent extras thay thế.

Intent extras là các cặp khóa/giá trị trong một Bundle. Bundle là một tập hợp dữ liệu được lưu trữ dưới dạng các cặp khóa/giá trị. Để truyền thông tin từ một activity này sang activity

khác, bạn đặt các khóa và giá trị vào Intent extra Bundle từ activity gửi, và sau đó lấy chúng ra lại trong activity nhận.

Trong task này, bạn sẽ sửa đổi intent tường minh trong MainActivity để bao gồm dữ liệu bổ sung (trong trường hợp này là một chuỗi do người dùng nhập) trong Intent extra Bundle. Sau đó, bạn sửa đổi SecondActivity để lấy dữ liệu đó ra khỏi Intent extra Bundle và hiển thị nó trên màn hình.

3.1 Thêm một EditText vào bố cục MainActivity

1. Mở **activity_main.xml**.

2. Kéo một phần tử **Plain Text** từ ngăn Palette đến cuối bố cục và thêm các ràng buộc vào cạnh trái của bố cục, cuối bố cục và cạnh trái của nút Send. Đặt các thuộc tính của nó trong ngăn Attributes như sau:

Thuộc tính	Giá trị
id	editText_main
Right margin	8dp
Left margin	8dp
Bottom margin	16dp
layout_width	match_constraint
layout_height	wrap_content
inputType	textLongMessage
hint	Enter your message here
text	(Delete any text in this field)

Bố cục mới:



3. Nhập vào tab **Text** để chỉnh sửa mã XML và trích xuất chuỗi "Enter your message here" thành một tài nguyên có tên là **editText_main**.

Mã XML của bộ cục:

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

    <Button
        android:id="@+id/btn_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:text="@string/btn_main"
```

```

        android:onClick="launchSecondActivity"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

<EditText
    android:id="@+id/editText_main"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginRight="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginBottom="16dp"
    android:ems="10"
    android:inputType="textLongMessage"
    android:hint="@string/editText_main"
    android:text=""
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id	btn_main"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3.2 Thêm một chuỗi vào Intent extras

Intent extras là các cặp khóa/giá trị trong một Bundle. Bundle là một tập hợp dữ liệu, được lưu trữ dưới dạng các cặp khóa/giá trị. Để truyền thông tin từ một Activity này sang Activity khác, bạn đặt các khóa và giá trị vào Intent extra Bundle từ Activity gửi, và sau đó lấy chúng ra lại trong Activity nhận.

1. Mở **MainActivity**.

2. Thêm một hằng số **public** ở đầu lớp để xác định khóa cho Intent extra:

```

private static final String EXTRA_MESSAGE = "com.example.twoactivities.extra.MESSAGE";

```

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

```

private EditText mMessageEditText;
```

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

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

5. Trong phương thức **launchSecondActivity()**, ngay dưới **new Intent**, lấy văn bản từ EditText dưới dạng một chuỗi:

```
String message = mMessageEditText.getText().toString();
```

6. Thêm chuỗi đó vào Intent dưới dạng một extra với hằng số **EXTRA_MESSAGE** làm khóa và chuỗi làm giá trị:

```
intent.putExtra(EXTRA_MESSAGE, message);
```

Mã của phương thức **onCreate()**:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
    setContentView(R.layout.activity_main);  
  
    mMessageEditText = findViewById(R.id.editText_main);
```

Mã của phương thức **launchSecondActivity()**:

```
public void launchSecondActivity(View view) {  
    Log.d(LOG_TAG, msg: "Button clicked!");  
  
    Intent intent = new Intent( packageContext: this, SecondActivity.class);  
  
    String message = mMessageEditText.getText().toString();  
    intent.putExtra(EXTRA_MESSAGE, message);  
  
    startActivity(intent);  
}
```

3.2 Thêm một TextView vào SecondActivity cho tin nhắn

1. Mở **activity_second.xml**.
2. Kéo một TextView khác vào bố cục bên dưới TextView **text_header** và thêm các ràng buộc vào cạnh trái của bố cục và vào cuối **text_header**.
3. Đặt thuộc tính TextView mới trong ngăn **Attributes** như sau:

Thuộc tính	Giá trị
id	text_message
Top margin	8dp
Left margin	8dp
layout_width	wrap_content
layout_height	wrap_content
text	(Delete any text in this field)
textAppearance	AppCompat.Medium

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

3.3 Sửa đổi SecondActivity để lấy extras và hiển thị tin nhắn

- Mở **SecondActivity** để thêm mã vào phương thức **onCreate()**.
- Lấy Intent đã kích hoạt Activity này: `Intent intent = getIntent();`
- Lấy chuỗi chứa tin nhắn từ **Intent extras** bằng cách sử dụng biến tĩnh **MainActivity.EXTRA_MESSAGE** làm khóa:

```
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

- Sử dụng **findViewById()** để lấy một tham chiếu đến TextView cho tin nhắn từ bố cục:

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

- Đặt văn bản của TextView thành chuỗi từ Intent extra:

```
textView.setText(message);
```

- Chạy ứng dụng. Khi bạn nhập một tin nhắn trong MainActivity và nhập vào Send, SecondActivity sẽ khởi chạy và hiển thị tin nhắn.

Phương thức **onCreate()** của SecondActivity như sau:

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);

    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

    TextView textView = findViewById(R.id.text_message);
    textView.setText(message);

    setContentView(R.layout.activity_second);
}

```

Task 4: Trả dũ liệu trả lại Activity chính

Bây giờ bạn đã có một ứng dụng khởi chạy một activity mới và gửi dũ liệu đến nó, bước cuối cùng là trả dũ liệu từ activity thứ hai trả lại activity chính. Bạn cũng sử dụng một intent và intent extras cho task này.

4.1 Thêm một EditText và một Button vào bố cục SecondActivity

- Mở **strings.xml** và thêm các tài nguyên chuỗi cho văn bản Button và gợi ý cho EditText mà bạn sẽ thêm vào SecondActivity:

```

<string name="button_second">Reply</string>
<string name="editText_second">Enter your reply here</string>

```

- Mở **activity_main.xml** và **activity_second.xml**.
- Sao chép EditText và Button từ tệp bố cục **activity_main.xml** và dán chúng vào bố cục **activity_second.xml**.
- Trong **activity_second.xml**, sửa đổi các giá trị thuộc tính cho Button như sau:

Giá trị cũ	Giá trị mới
android:id="@+id/button_main"	android:id="@+id/btn_second"
android:onClick="launchSecondActivity"	android:onClick="returnReply"

android:text="@string/button_main"	android:text="@string/button_second"
------------------------------------	--------------------------------------

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

Giá trị cũ	Giá trị mới
android:id="@+id/editText_main"	android:id="@+id/editText_second"
app:layout_constraintEnd_toStartOf="@+id/button"	app:layout_constraintEnd_toStartOf="@+id/button_second"
android:hint="@string/editText_main"	android:hint="@string/editText_second"

6. Trong trình chỉnh sửa bộ cục XML, nhấp vào returnReply, nhấn Alt+Enter (Option+Return trên Mac) và chọn Create 'returnReply(View)' in 'SecondActivity'.

Android Studio tạo một phương thức khung cho trình xử lý **returnReply()**. Bạn triển khai phương thức này trong task tiếp theo.

Bộ cục mới cho activity_second.xml trông như thế này:



4.2 Tạo một Intent phản hồi trong activity thứ hai

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

1. Mở **SecondActivity**.

2. Ở đầu lớp, thêm một hằng số public để xác định khóa cho Intent extra:

```
no usages
public static final String EXTRA_REPLY = "com.example.twoactivities.extra.REPLY";
```

3. Thêm một biến private ở đầu lớp để giữ EditText: **private EditText mReply;**

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

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

5. Trong phương thức **returnReply()**, lấy văn bản của EditText làm một chuỗi:

```
String replyMessage = mReply.getText().toString();
```

6. Trong phương thức **returnReply()**, tạo một intent mới cho phản hồi - không sử dụng lại đối tượng Intent mà bạn nhận được từ yêu cầu ban đầu.

7. Thêm chuỗi phản hồi từ EditText vào intent mới dưới dạng một Intent extra. Vì extras là các cặp khóa/giá trị, ở đây khóa là EXTRA_REPLY và giá trị là phản hồi:

```
Intent intent = new Intent();
intent.putExtra(EXTRA_REPLY, replyMessage);
```

8. Đặt kết quả thành RESULT_OK để cho biết rằng phản hồi đã thành công. Lớp Activity định nghĩa các mã kết quả, bao gồm RESULT_OK và RESULT_CANCELLED: **setResult(RESULT_OK);**

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

1 usage

```
public void returnReply(View view) {
    Intent intent = new Intent();
    String replymessage = mReply.getText().toString();
    intent.putExtra(EXTRA_REPLY, replymessage);
    setResult(RESULT_OK);
    finish();
}
```

4.3 Thêm các phần tử TextView để hiển thị phản hồi

MainActivity cần một cách để hiển thị phản hồi mà SecondActivity gửi. Trong task này, bạn thêm các phần tử TextView vào bố cục activity_main.xml để hiển thị phản hồi trong MainActivity.

Để giúp task này dễ dàng hơn, bạn sao chép các phần tử TextView bạn đã sử dụng trong SecondActivity.

1. Mở **strings.xml** và thêm một tài nguyên chuỗi cho tiêu đề phản hồi:

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

2. Mở **activity_main.xml** và **activity_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 cũ	Giá trị mới
android:id="@+id/txt_header"	android:id="@+id/text_header_reply"
android:text="@string/txt_header"	android:text="@string/text_header_reply"

5. Trong **activity_main.xml**, sửa đổi các giá trị thuộc tính cho TextView thứ hai như sau:

Giá trị cũ	Giá trị mới
android:id="@+id/text_message"	android:id="@+id/text_message_reply"
app:layout_constraintTop_toBottomOf="@+id/text_header"	app:layout_constraintTop_toBottomOf="@+id/text_header_reply"

6. Thêm thuộc tính **android:visibility** vào mỗi TextView để làm cho chúng ban đầu không hiển thị. (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="invisible"
```

Bạn sẽ làm cho các phần tử TextView này hiển thị sau khi dữ liệu phản hồi được truyền trở lại từ activity thứ hai

Bộ cục activity_main.xml trông giống như trong task trước - mặc dù bạn đã thêm hai phần tử TextView mới vào bộ cục. Bởi vì bạn đặt các phần tử này thành không hiển thị, chúng không xuất hiện trên màn hình.

4.4 Lấy phản hồi từ Intent extra và hiển thị nó

Khi bạn sử dụng một Intent tường minh để khởi động một Activity khác, bạn có thể không mong đợi nhận được bất kỳ dữ liệu nào trả lại - bạn chỉ đang kích hoạt Activity đó. Trong trường hợp đó, bạn sử dụng **startActivity()** để khởi động Activity mới, như bạn đã làm trước đó trong bài thực hành này. Tuy nhiên, nếu bạn muốn nhận dữ liệu trả lại từ Activity đã kích hoạt, bạn cần khởi động nó bằng **startActivityForResult()**.

Trong task này, bạn sửa đổi ứng dụng để khởi động SecondActivity và mong đợi một kết quả, để trích xuất dữ liệu trả về đó từ Intent và để hiển thị dữ liệu đó trong các phần tử TextView mà bạn đã tạo trong task trước đó.

1. Mở **MainActivity**.

2. Thêm một hằng số public 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 private để giữ các phần tử tiêu đề và phản hồi TextView:

```
private TextView mReplyHeadTextView;  
1 usage  
private TextView mReplyTextView;
```

4. Khai báo biến private để đăng ký nhận kết quả trả về:

```
private ActivityResultLauncher<Intent> activityResultLauncher;
```

5. Trong phương thức **onCreate()**, sử dụng **findViewById()** để lấy các tham chiếu từ bộ cục đến các phần tử tiêu đề và phản hồi TextView. Gán các phiên bản view đó cho các biến private:

```
mReplyHeadTextView = findViewById(R.id.text_header_reply);
mReplyTextView = findViewById(R.id.text_message_reply);
```

6. Đăng ký nhận kết quả trả về với registerForActivityResult():

```
activityResultLauncher = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        // code
    }
);
```

7. Trong phương thức **launchSecondActivity()**, thay đổi lệnh gọi **startActivity()** thành **activityResultLauncher.launch()**:

```
activityResultLauncher.launch(intent);
```

8. Trong khối đăng ký nhận kết quả trả về, kiểm tra mã kết quả và lấy dữ liệu trả về.

9. Bên trong khối if bên trong, lấy Intent extra từ Intent phản hồi . Ở đây, khóa cho extra là hằng số EXTRA_REPLY từ SecondActivity:

10. Đặt khả năng hiển thị của tiêu đề phản hồi thành true:

11. Đặt văn bản TextView phản hồi thành phản hồi và đặt khả năng hiển thị của nó thành true:

```
result -> {
    if (result.getResultCode() == RESULT_OK) {
        Intent data = result.getData();
        if (data != null) {
            String message = data.getStringExtra(SecondActivity.EXTRA_REPLY);
            mReplyHeadTextView.setVisibility(View.VISIBLE);
            mReplyTextView.setText(message);
            mReplyTextView.setVisibility(View.VISIBLE);
        }
    }
}
```

12. Chạy ứng dụng.

Bây giờ, khi bạn gửi một tin nhắn đến activity thứ hai và nhận được phản hồi, MainActivity sẽ cập nhật để hiển thị phản hồi.

Message Received

Hello



huhuu

Reply

Tóm tắt

Tổng quan:

- Một Activity là một thành phần ứng dụng cung cấp một màn hình duy nhất tập trung vào một tác vụ người dùng duy nhất.
- Mỗi Activity có tệp bố cục giao diện người dùng riêng.
- Bạn có thể gán cho các triển khai Activity của mình một mối quan hệ cha/con để cho phép điều hướng Up trong ứng dụng của bạn.
- Một View có thể được hiển thị hoặc không hiển thị với thuộc tính android:visibility.

Để triển khai một Activity:

- Chọn **File > New > Activity** để bắt đầu từ một mẫu và thực hiện các bước sau tự động.
- Nếu không bắt đầu từ một mẫu, hãy tạo một lớp Java Activity, triển khai một UI cơ bản cho Activity trong một tệp bố cục XML được liên kết và khai báo Activity mới trong AndroidManifest.xml.

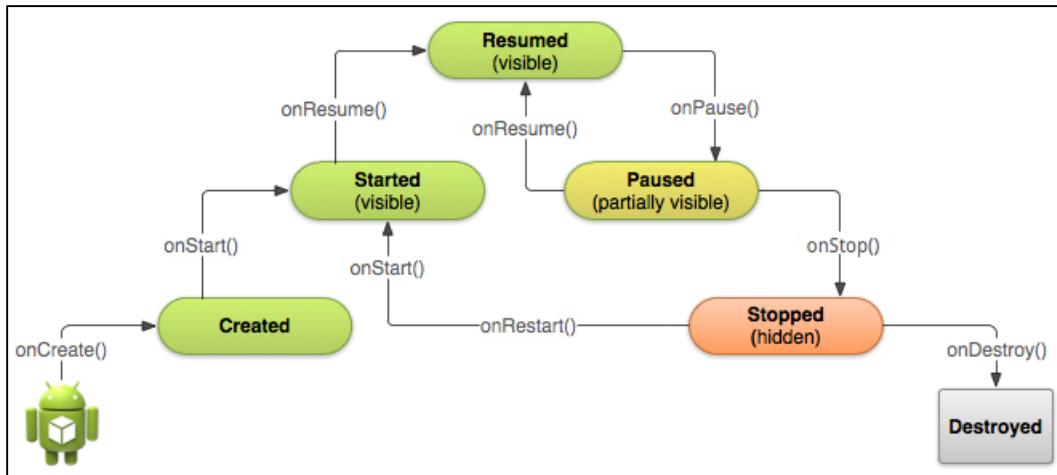
Intent:

- Một Intent cho phép bạn yêu cầu một hành động từ một thành phần khác trong ứng dụng của bạn, ví dụ: để khởi động một Activity này từ một Activity khác. Một Intent có thể là tường minh hoặc ngầm định.
- Với một Intent tường minh, bạn chỉ định thành phần mục tiêu cụ thể để nhận dữ liệu.
- Với một Intent ngầm định, 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.
- Intent extras là các cặp khóa/giá trị trong một Bundle được gửi cùng với Intent

2.2)Vòng đời của Activity và trạng thái

Giới thiệu

Trong bài thực hành này, bạn sẽ tìm hiểu thêm về vòng đời activity. Vòng đời là tập hợp các trạng thái mà một activity có thể ở trong suốt thời gian tồn tại của nó, từ khi nó được tạo cho đến khi nó bị hủy và hệ thống thu hồi lại các tài nguyên của nó. Khi người dùng điều hướng giữa các activity trong ứng dụng của bạn (cũng như vào và ra khỏi ứng dụng của bạn), các activity chuyển đổi giữa các trạng thái khác nhau trong vòng đời của chúng.



Mỗi giai đoạn trong vòng đời của một activity có một phương thức callback tương ứng: `onCreate()`, `onStart()`, `onPause()`, v.v. Khi một activity thay đổi trạng thái, phương thức callback liên quan sẽ được gọi. Bạn đã thấy một trong những phương thức này: `onCreate()`. Bằng cách ghi đè bất kỳ phương thức callback vòng đời nào trong các lớp Activity của bạn, bạn có thể thay đổi hành vi mặc định của activity để đáp ứng các hành động của người dùng hoặc hệ thống.

Trạng thái activity cũng có thể thay đổi để đáp ứng các thay đổi cấu hình thiết bị, ví dụ: khi người dùng xoay thiết bị từ hướng dọc sang hướng ngang. Khi những thay đổi cấu hình này xảy ra, activity sẽ bị hủy và tạo lại ở trạng thái mặc định của nó, và người dùng có thể mất thông tin mà họ đã nhập vào activity. Để tránh gây nhầm lẫn cho người dùng, điều quan trọng là bạn phải phát triển ứng dụng của mình để ngăn chặn mất dữ liệu không mong muốn. Cuối bài thực hành này, bạn sẽ thử nghiệm với các thay đổi cấu hình và tìm hiểu cách duy trì trạng thái của một activity để đáp ứng các thay đổi cấu hình thiết bị và các sự kiện vòng đời activity khác.

Trong bài thực hành này, bạn sẽ thêm các câu lệnh ghi nhật ký vào ứng dụng TwoActivities và quan sát các thay đổi vòng đời activity 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 các điều kiện này.

Những gì bạn nên biết

Bạn nên biết:

- Tạo và chạy một dự án ứng dụng trong Android Studio.
- Thêm các câu lệnh ghi nhật ký vào ứng dụng của bạn và xem các nhật ký đó trong ngăn Logcat.
- Hiểu và làm việc với một Activity và một Intent, và cảm thấy thoải mái khi tương tác với chúng.

Những gì bạn sẽ học

- Cách hoạt động của vòng đời Activity.
- Khi nào một Activity bắt đầu, tạm dừng, dừng và bị hủy.
- Về các phương thức callback vòng đời liên quan đến các thay đổi Activity.
- 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 trong các sự kiện vòng đời.

Những gì bạn sẽ làm

- Thêm mã vào ứng dụng TwoActivities từ bài thực hành trước để triển khai các callback vòng đời Activity 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 mỗi Activity trong ứng dụng của bạn.
- Sửa đổi ứng dụng của bạn để giữ lại trạng thái phiên bản của một Activity bị tạo lại ngoài ý muốn để đáp ứng hành vi của người dùng hoặc thay đổi cấu hình trên thiết bị.

Tổng quan ứng dụng

Trong bài thực hành này, bạn sẽ thêm vào ứng dụng **TwoActivities**. Ứng dụng trông và hoạt động gần giống như trong codelab cuối cùng. Nó chứa hai triển khai Activity và cho phép người dùng gửi giữa chúng. Các thay đổi bạn thực hiện đối với ứng dụng trong bài thực hành này sẽ không ảnh hưởng đến hành vi người dùng có thể nhìn thấy của nó.

Task 1: Thêm các callback vòng đời vào TwoActivities

Trong task này, bạn sẽ triển khai tất cả các phương thức callback vòng đời Activity để in các thông báo ra logcat khi các phương thức đó được gọi. Các thông báo nhật ký này sẽ cho phép bạn xem khi nào vòng đời Activity thay đổi trạng thái và những thay đổi trạng thái vòng đời đó ảnh hưởng đến ứng dụng của bạn như thế nào khi nó chạy.

1.1 (Tùy chọn) Sao chép dự án TwoActivities

Đối với các task trong bài thực hành này, bạn sẽ sửa đổi dự án TwoActivities hiện có mà bạn đã xây dựng trong bài thực hành trước. Nếu bạn muốn giữ nguyên dự án TwoActivities trước đó, hãy làm theo các bước trong Phụ lục: Tiện ích để tạo bản sao của dự án.

1.2 Triển khai các callback vào MainActivity

1. Mở dự án TwoActivities trong Android Studio và mở MainActivity trong ngăn Project > Android.
2. Trong phương thức onCreate(), thêm các câu lệnh log sau:

```
Log.d(LOG_TAG, msg: "----");
Log.d(LOG_TAG, msg: "onCreate");
```

3. Thêm một override cho callback onStart(), với một câu lệnh cho log cho sự kiện đó:

```
@Override
public void onStart() {
    super.onStart();
    Log.d(LOG_TAG, msg: "onStart");
}
```

Để có một phím 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 callback 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 callback vòng đời onPause(), onRestart(), onResume(), onStop() và onDestroy().

Tất cả các phương thức callback có cùng chữ ký (ngoại trừ tên). Nếu bạn Sao chép và dán **onStart()** để tạo các phương thức callback khác này, đừng quên cập nhật nội dung để gọi đúng phương thức trong siêu lớp và để ghi nhật ký đúng phương thức.

5. Chạy ứng dụng của bạn.
6. 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 đổi qua khi nó bắt đầu.

```
10600-10600 MainActivity      com.example.twoactivities
10600-10600 MainActivity      com.example.twoactivities
10600-10600 MainActivity      com.example.twoactivities
10600-10600 MainActivity      com.example.twoactivities
```



1.3 Triển khai các callback vòng đời trong SecondActivity

Bây giờ bạn đã triển khai các phương thức callback vòng đời cho MainActivity, hãy làm tương tự cho SecondActivity.

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 các callback vòng đời và câu lệnh nhật ký vào Activity thứ hai. (Bạn có thể sao chép và dán các phương thức callback từ MainActivity)
4. Thêm một câu lệnh nhật ký vào phương thức `returnReply()` ngay trước phương thức `finish()`:

```
Log.d(LOG_TAG, "End SecondActivity");
```

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.

Nhật ký 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.

```
26 MainActivity      com.example.twoactivities
26 MainActivity      com.example.twoactivities
26 MainActivity      com.example.twoactivities
3  WindowManagerShell pid-964
26 MainActivity      com.example.twoactivities
26 MainActivity      com.example.twoactivities
26 MainActivity      com.example.twoactivities
controlTarget == DisplayContent.controlT
D  onStop
D  onDestroy
D  Button clicked!
V  Transition requested (#66): android.os.B:
D  onPause
D  onStart
D  onResume
```

Thử nghiệm bằng cách sử dụng ứng dụng của bạn và lưu ý các sự kiện vòng đời xảy ra để đáp ứng các hành động khác nhau. Đặc biệt, hãy thử những điều này:

- Sử dụng ứng dụng bình thường (gửi tin nhắn, trả lời bằng một tin nhắn khác).
- Sử dụng nút Back để quay lại từ Activity thứ hai sang Activity chính.
- Sử dụng mũi tên Up trong thanh ứng dụng để quay lại từ Activity thứ hai sang Activity chính.
- Xoay thiết bị trên cả Activity chính và Activity 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 hình vuông ở bên phải Home) và đóng ứng dụng (chạm vào X).
- Quay lại màn hình chính và khởi động lại ứng dụng của bạn.

Task 2: Lưu và khôi phục trạng thái phiên bản Activity

Tùy thuộc vào tài nguyên hệ thống và hành vi của người dùng, mỗi Activity trong ứng dụng của bạn có thể bị hủy và xây dựng lại thường xuyên hơn bạn nghĩ.

Bạn có thể đã nhận thấy hành vi này trong phần trước khi bạn xoay thiết bị hoặc trình giả lập. Xoay thiết bị là một ví dụ về thay đổi cấu hình thiết bị. Mặc dù xoay là phổ biến nhất, tất cả các thay đổi cấu hình đều dẫn đến việc Activity hiện tại bị hủy và tạo lại như thế nó là mới. Nếu bạn không tính đến hành vi này trong mã của mình, khi một thay đổi cấu hình xảy ra, bố cục Activity của bạn có thể trở lại giao diện mặc định và các giá trị ban đầu, và người dùng của bạn có thể mất vị trí, dữ liệu của họ hoặc trạng thái tiến trình của họ 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 phiên bản Activity. Hệ thống lưu thông tin trạng thái mặc định vào Bundle trạng thái phiên bản ngay trước khi Activity bị dừng và truyền Bundle đó đến phiên bản Activity mới để khôi phục.

Để tránh mất dữ liệu trong một Activity khi nó bị hủy và tạo lại ngoài ý muốn, 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 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 suốt phiên ứng dụng hiện tại. Khi bạn dừng và khởi động lại một phiên ứng dụng mới, trạng thái phiên bản Activity sẽ bị mất và Activity sẽ trở lại giao diện mặc định của nó. Nếu bạn cần lưu dữ liệu người dùng giữa các phiên ứng dụng, hãy sử dụng shared preferences hoặc cơ sở dữ liệu. Bạn sẽ tìm hiểu về cả hai điều này trong một bài thực hành sau.

2.1 Lưu trạng thái phiên bản Activity bằng **onSaveInstanceState()**

Bạn có thể đã nhận thấy rằng việc xoay thiết bị không ảnh hưởng đến trạng thái của Activity thứ hai. Điều này là do bố cục và trạng thái của Activity thứ hai được tạo từ bố cục và Intent đã kích hoạt nó. Ngay cả khi 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 thứ hai được gọi.

Ngoài ra, bạn có thể nhận thấy rằng trong mỗi Activity, bất kỳ văn bản nào bạn nhập vào các phần tử tin nhắn hoặc phản hồi 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 bố cục của bạn được tự động lưu trên các thay đổi cấu hình và giá trị hiện tại của EditText là một trong những trường hợp đó.

Vì vậy, trạng thái Activity duy nhất bạn quan tâm là các phần tử TextView cho tiêu đề phản hồi và văn bản phản hồi trong Activity chính. Cả hai phần tử TextView đều không hiển thị theo mặc định; chúng chỉ xuất hiện khi bạn gửi tin nhắn trả lại Activity chính từ Activity thứ hai.

Trong task này, bạn sẽ thêm mã để giữ lại trạng thái phiên bản của hai phần tử TextView này bằng cách sử dụng **onSaveInstanceState()**.

1. Mở **MainActivity**.
2. Thêm triển khai của **onSaveInstanceState()** vào Activity.

```
@Override  
public void onSaveInstanceState(Bundle outState) {  
    super.onSaveInstanceState(outState);  
}
```

3. Kiểm tra xem tiêu đề có hiển thị hay không và nếu có, hãy đưa trạng thái hiển thị đó vào Bundle trạng thái bằng phương thức **putBoolean()** và khóa "reply_visible".

```
if (mReplyHeadTextView.getVisibility() == View.VISIBLE) {  
    outState.putBoolean("reply_visible", true);  
}
```

Hãy nhớ rằng tiêu đề và văn bản phản hồi được đánh dấu là không hiển thị cho đến khi có phản hồi từ Activity thứ hai. Nếu tiêu đề hiển thị, thì có dữ liệu phản hồi 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 được lưu, vì văn bản đó không bao giờ thay đổi.

4. Bên trong cùng một kiểm tra đó, thêm văn bản phản hồi vào Bundle.

```
outState.putString("reply_text", mReplyTextView.getText().toString());
```

Nếu tiêu đề hiển thị, bạn có thể giả sử rằng bản thân tin nhắn phản hồi cũng hiển thị. Bạn không cần kiểm tra hoặc lưu trạng thái hiển thị hiện tại của tin nhắn phản hồi. Chỉ văn bản thực tế của tin nhắn được đưa vào Bundle trạng thái với khóa "reply_text".

Bạn chỉ lưu trạng thái của các 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 bố cục mặc định bất kỳ 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()

Sau 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 callback onRestoreInstanceState(), được gọi sau onStart() sau khi Activity được tạo.

Hầu hết, nơi tốt hơn để khôi phục trạng thái Activity là trong onCreate(), để đảm bảo rằng UI, bao gồm cả trạng thái, có sẵn càng sớm càng tốt. Đôi khi, việc thực hiện trong onRestoreInstanceState() sau khi tắt cả 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 là tiện lợi hơn.

- Trong phương thức onCreate(), sau khi các biến View được khởi tạo bằng findViewById(), thêm một thử nghiệm để đảm bảo rằng savedInstanceState không phải là null.

```
if (savedInstanceState != null) {  
}
```

Khi Activity của bạn được tạo, hệ thống sẽ chuyển Bundle trạng thái đến onCreate() làm đối số duy nhất của nó. Lần đầu tiên onCreate() được gọi và ứng dụng của bạn bắt đầu, Bundle là null — không có trạng thái hiện tại nào trong lần đầu tiên ứng dụng của bạn bắt đầu. Các lệnh gọi tiếp theo đến onCreate() có một bundle chứa dữ liệu bạn đã lưu trong onSaveInstanceState().

- Bên trong kiểm tra đó, hãy lấy khả năng hiển thị hiện tại (true hoặc false) ra khỏi Bundle với khóa "reply_visible".

```
boolean isVisible =  
    savedInstanceState.getBoolean(key: "reply_visible");
```

- Thêm một thử nghiệm bên dưới dòng trước đó cho biến isVisible.

Nếu có một khóa reply_visible trong Bundle trạng thái (và do đó isVisible là true), bạn sẽ cần khôi phục trạng thái.

- Bên trong thử nghiệm isVisible, hãy làm cho tiêu đề hiển thị
- Lấy tin nhắn phản hồi văn bản từ Bundle với khóa "reply_text" và đặt TextView phản hồi để hiển thị chuỗi đó.

```
if (isVisible) {  
    mReplyTextView.setText(savedInstanceState.getString(key: "reply_text"));  
}
```

6. Làm cho TextView phản hồi cũng hiển thị
7. Chạy ứng dụng. Thủ xoay thiết bị hoặc trình giả lập để đảm bảo rằng tin nhắn phản hồi (nếu có) vẫn còn trên màn hình sau khi Activity được tạo lại.

Tóm tắt

- Vòng đời Activity là một tập hợp các trạng thái mà một Activity di chuyển qua, bắt đầu khi nó được tạo lần đầu tiên và kết thúc khi hệ thống Android thu hồi các tài nguyên cho Activity đó.
- Khi người dùng điều hướng từ Activity này sang Activity khác, và bên trong và bên ngoài ứng dụng của bạn, mỗi Activity di chuyển giữa các trạng thái trong vòng đời Activity.
- Mỗi trạng thái trong vòng đời Activity có một phương thức callback tương ứng mà bạn có thể ghi đè trong lớp Activity của bạn.
- Các phương thức vòng đời là onCreate(), onStart(), onPause(), onRestart(), onResume(), onStop(), onDestroy().
- Ghi đè một phương thức callback 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 override khung vào các lớp của mình trong Android Studio với Code > Override.
- Các thay đổi cấu hình thiết bị như xoay dẫn đến việc Activity bị hủy và tạo lại như thế nào là mới.
- Một phần của trạng thái Activity được giữ lại trên một thay đổi cấu hình, bao gồm các giá trị hiện tại của các phần tử EditText. Đối với tất cả dữ liệu khác, bạn phải tự mình 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 một Bundle. Sử dụng các phương thức Bundle để đưa dữ liệu vào và lấy dữ liệu ra khỏi Bundle.
- Khôi phục trạng thái phiên bản trong onCreate(), là cách ưu tiên hoặc trong onRestoreInstanceState().

2.3) Intent ngầm định

Giới thiệu

Trong một phần trước, bạn đã tìm hiểu về intent tường minh. Trong một intent tường minh, bạn thực hiện một activity trong ứng dụng của mình hoặc trong một ứng dụng khác, bằng cách gửi một intent với tên lớp đủ điều kiện của activity. Trong phần này, bạn sẽ tìm hiểu thêm về intent ngầm định và cách sử dụng chúng để thực hiện các activity.

Với một intent ngầm định, bạn khởi tạo một activity mà không biết ứng dụng hoặc activity nào sẽ xử lý task. 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 activity nào thực hiện task.

Ngược lại, activity của bạn có thể khai báo một hoặc nhiều bộ lọc intent trong tệp AndroidManifest.xml để quảng cáo rằng activity có thể chấp nhận các intent ngầm định và để xác định các loại intent mà activity sẽ chấp nhận.

Để khớp yêu cầu của bạn với một ứng dụng được cài đặt trên thiết bị, hệ thống Android sẽ khớp intent ngầm định của bạn với một activity có bộ lọc intent cho biết rằng chúng có thể thực hiện hành động. Nếu có nhiều ứng dụng phù hợp, người dùng sẽ được cung cấp một trình chọn ứng dụng cho phép họ chọn ứng dụng nào họ muốn sử dụng để xử lý intent.

Trong bài thực hành này, bạn sẽ xây dựng một ứng dụng gửi một intent ngầm định để thực hiện từng task sau:

- Mở một URL trong trình duyệt web.
- Mở một vị trí trên bản đồ.
- Chia sẻ văn bản.

Chia sẻ - gửi một mẫu thông tin cho những người khác thông 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 ShareCompat.IntentBuilder, giúp dễ dàng xây dựng một intent ngầm định để chia sẻ dữ liệu.

Cuối cùng, bạn tạo một trình nhận intent đơn giản chấp nhận một intent ngầm định cho một hành động cụ thể.

Những gì bạn nên biết

Bạn có thể:

- Sử dụng trình chỉnh sửa bộ cục để sửa đổi bộ cục.
- Chính sửa mã XML của một bộ cục.
- Thêm một Button và một trình xử lý nhập chuột.
- Tạo và sử dụng một Activity.
- Tạo và gửi một Intent giữa Activity này và Activity khác.

Những gì bạn sẽ học

- Cách tạo một Intent ngầm định và sử dụng các hành động và danh mục của nó.
- Cách sử dụng lớp trợ giúp ShareCompat.IntentBuilder để tạo một Intent ngầm định để chia sẻ dữ liệu.
- Cách quảng cáo rằng ứng dụng của bạn có thể chấp nhận một Intent ngầm định bằng cách khai báo các bộ lọc Intent trong tệp AndroidManifest.xml.

Những gì bạn sẽ làm

- Tạo một ứng dụng mới để thử nghiệm với Intent ngầm định.
- Triển khai một Intent ngầm định mở một trang web và một Intent 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 định để mở một trang web.

Tổng quan ứng dụng

Trong phần này, bạn sẽ tạo một ứng dụng mới với một Activity và ba tùy chọn cho các hành động: mở một trang web, mở một vị trí trên bản đồ và chia sẻ một đoạn văn bản. Tất cả các trường văn bản đều có thể chỉnh sửa (EditText), nhưng chứa các giá trị mặc định.

Task 1: Tạo dự án và bố cục

Đối với bài tập này, bạn sẽ tạo một dự án và ứng dụng mới có tên là Implicit Intents, với một bố cục mới.

1.1 Tạo dự án

- Khởi động Android Studio và tạo một dự án Android Studio mới. Đặt tên cho ứng dụng của bạn là **Implicit Intents**.
- Chọn Empty Views Activity cho mẫu dự án. Nhấp vào Next.
- Chấp nhận tên Activity mặc định (MainActivity). Đảm bảo hộp Generate Layout file được chọn. Nhấp vào Finish.

1.2 Tạo bố cục

Trong task này, hãy tạo bố cục cho ứng dụng. Sử dụng LinearLayout, ba phần tử Button và ba phần tử EditText, như sau:

- Mở **app > res > values > strings.xml** trong ngăn Project > Android và thêm các tài nguyên chuỗi sau:

```
<string name="edittext_uri">http://developer.android.com</string>
<string name="button_uri">Open Website</string>
<string name="edittext_loc">Golden Gate Bridge</string>
<string name="button_loc">Open Location</string>
<string name="edittext_share">\'Twas brillig and the slithy toves</string>
<string name="button_share">Share This Text</string>
```

- Mở **res > layout > activity_main.xml** trong ngăn Project > Android. Nhấp vào tab Text để chuyển sang mã XML.
- Thay đổi android.support.constraint.ConstraintLayout thành LinearLayout.

4. Thêm thuộc tính **android:orientation** với giá trị "**vertical**". Thêm thuộc tính **android:padding** với giá trị "**16dp**".
5. Xóa TextView hiển thị "Hello World".
6. Thêm một tập hợp các phần tử UI vào bố cục cho nút Open Website. Bạn cần một phần tử EditText và một phần tử Button. Sử dụng các giá trị thuộc tính sau:

Thuộc tính của EditText	Giá trị
android:id	"@+id/website_edittext"
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:text	"@string/edittext_uri"
Thuộc tính của Button	Giá trị
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"

Giá trị cho thuộc tính **android:onClick** sẽ vẫn được gạch chân màu đỏ cho đến khi bạn xác định phương thức callback trong một task tiếp theo.

7. Thêm một tập hợp các phần tử UI (EditText và Button) vào bố cục cho nút Open Location. Sử dụng các thuộc tính tương tự như trong bước trước, nhưng sửa đổi chúng như được hiển thị bên dưới.

Thuộc tính của EditText	Giá trị
android:id	"@+id/location_edittext"
android:layout_width	"match_parent"
Thuộc tính của Button	Giá trị
android:id	"@+id/open_location_button"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginBottom	"24dp"
android:text	"@string/button_uri"
android:onClick	"openLocation"

android:layout_height	"wrap_content"
android:text	"@string/edittext_loc"
Thuộc tính của Button	Giá trị
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"

Giá trị cho thuộc tính **android:onClick** sẽ vẫn được gạch chân màu đỏ cho đến khi bạn xác định phương thức callback trong một task tiếp theo.

8. Thêm một tập hợp các phần tử UI (EditText và Button) vào bộ cục cho nút Share This. Sử dụng các thuộc tính được hiển thị bên dưới.

Thuộc tính của EditText	Giá trị
android:id	"@+id/share_edittext"
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:text	"@string/edittext_share"
Thuộc tính của Button	Giá trị
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"
-----------------	-------------

Tùy thuộc vào phiên bản Android Studio của bạn, mã activity_main.xml của bạn phải 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 callback trong một task tiếp theo.

Task 2: Triển khai nút Open Website

Trong task này, bạn triển khai phương thức xử lý nhấp chuột cho nút đầu tiên trong bố cục, Open Website. Hành động này sử dụng một Intent ngầm định để gửi URI đã cho đến một Activity có thể xử lý Intent ngầm định đó (chẳng hạn như trình duyệt web).

2.1 Xác định openWebsite()

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

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

- Trong MainActivity, hãy thêm một biến private ở đầu lớp để giữ đối tượng EditText cho URI trang web.

```
private EditText mWebsiteEditText;
```

- Trong phương thức onCreate() cho MainActivity, sử dụng findViewById() để lấy một tham chiếu đến phiên bản EditText và gán nó cho biến private đó:

```
mWebsiteEditText = findViewById(R.id.website_edittext);
```

2.2 Thêm mã vào openWebsite()

- Thêm một câu lệnh vào phương thức **openWebsite()** mới để lấy giá trị chuỗi của EditText

2. Mã hóa và phân tích cú pháp chuỗi đó thành một đối tượng Uri:
3. Tạo một Intent mới với Intent.ACTION_VIEW làm hành động và URI làm dữ liệu:

```
String url = mWebsiteEditText.getText().toString();
Uri webpage = Uri.parse(url);
Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
```

Hàm tạo Intent này khác với hàm bạn đã sử dụng để tạo một Intent tường minh. Trong hàm tạo trước đó, bạn đã chỉ định ngữ cảnh hiện tại và một thành phần cụ thể (lớp Activity) để gửi Intent. Trong hàm 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 trang web.

4. Sử dụng phương thức resolveActivity() và trình quản lý gói Android để tìm một Activity có thể xử lý Intent ngầm định của bạn. Đảm bảo rằng yêu cầu đã được giải quyết thành công.

Yêu cầu này khớp hành động và dữ liệu Intent của bạn với các bộ lọc Intent cho các ứng dụng đã cài đặt trên thiết bị. Bạn sử dụng nó để đảm bảo rằng có ít nhất một Activity có thể xử lý các yêu cầu của bạn.

5. Bên trong câu lệnh if, hãy gọi startActivity() để gửi Intent.
6. Thêm một khôi else để in một thông báo Log nếu Intent không thể được giải quyết.

```
if (intent.resolveActivity(getApplicationContext()) != null) {
    startActivity(intent);
} else {
    Log.d("tag: "ImplicitIntents", msg: "Can't handle this!");
}
```

Task 3: Triển khai nút Open Location

Trong task này, bạn triển khai phương thức xử lý nhập chuột cho nút thứ hai trong UI, Open Location. Phương thức này gần giống với phương thức openWebsite(). Sự khác biệt là việc sử dụng URI địa lý để chỉ định một vị trí bản đồ. Bạn có thể sử dụng URI địa lý với vĩ độ và kinh độ hoặc sử dụng một chuỗi truy vấn cho một vị trí chung. Trong ví dụ này, chúng tôi đã sử dụng cái sau.

3.1 Xác định 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)' in MainActivity.

Android Studio tạo một phương thức khung trong MainActivity cho trình xử lý openLocation().

3. Thêm một biến private ở đầu MainActivity để giữ đối tượng EditText cho URI vị trí.

```
private EditText mLocationEditText;
```

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

```
mLocationEditText = findViewById(R.id.location_edittext);
```

3.2 Thêm mã vào openLocation()

1. Trong phương thức **openLocation()**, thêm một câu lệnh để lấy giá trị chuỗi của EditText mLocationEditText.
2. Phân tích cú pháp chuỗi đó thành một đối tượng Uri với một truy vấn tìm kiếm địa lý:
3. Tạo một Intent mới với Intent.ACTION_VIEW làm hành động và lọc làm dữ liệu.

```
String loc = mLocationEditText.getText().toString();
Uri addressUri = Uri.parse("geo:0,0?q=" + loc);
Intent intent = new Intent(Intent.ACTION_VIEW, addressUri);
```

4. Giải quyết Intent và kiểm tra để đảm bảo rằng Intent đã được giải quyết thành công. Nếu vậy, startActivity(), nếu không ghi một thông báo lỗi.

```
if (intent.resolveActivity(getApplicationContext()) != null) {  
    startActivity(intent);  
} else {  
    Log.d( tag: "ImplicitIntents", msg: "Can't handle this intent!");  
}
```

Task 4: Triển khai nút Share This Text

Hành động chia sẻ là một cách dễ dàng để người dùng chia sẻ các mục trong ứng dụng của bạn với các mạng xã hội và các ứng dụng khác. Mặc dù bạn có thể xây dựng một hành động chia sẻ trong ứng dụng của riêng mình bằng cách sử dụng một Intent ngầm định, 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 xây dựng một Intent và khởi chạy một trình chọn để cho phép người dùng chọn ứng dụng đích để chia sẻ.

Trong task này, bạn triển khai chia sẻ một đoạn văn bản trong một chỉnh sửa văn bản, bằng cách sử dụng lớp ShareCompat.IntentBuilder.

4.1 Xác định 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)' in MainActivity.

Android Studio tạo một phương thức khung trong MainActivity cho trình xử lý shareText()

3. Thêm một biến private ở đầu MainActivity để giữ EditText.
4. Trong onCreate(), sử dụng findViewById() để lấy một tham chiếu đến phiên bản EditText và gán nó cho biến private đó

4.2 Thêm mã vào shareText()

- Trong phương thức **shareText()**, hãy thêm một câu lệnh để lấy giá trị chuỗi của EditText mShareTextEdit.
- Xác định loại mime của văn bản để chia sẻ:

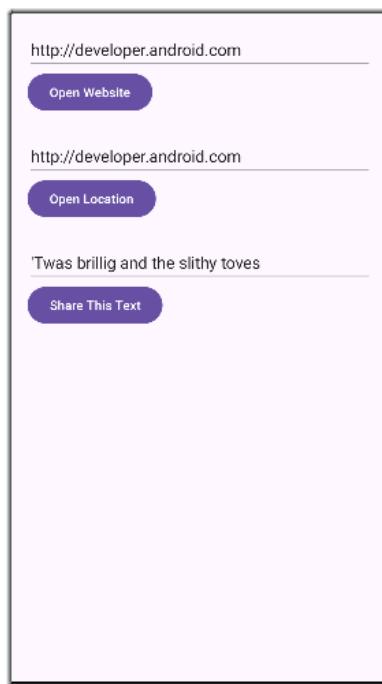
```
String txt = mShareTextEdit.getText().toString();
String mimeType = "text/plain";
```

- Tạo intent cho dữ liệu cần share:

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType(mimeType);
intent.putExtra(Intent.EXTRA_TEXT, txt);
```

- Tạo intent thực hiện shareText, gọi phương thức startActivity trên intent vừa tạo:

```
Intent shareIntent = Intent.createChooser(
    intent, title: "Share this text with:");
startActivity(shareIntent);
```



4.3 Chạy ứng dụng

- Chạy ứng dụng.

2. Nhập vào nút **Open Website** để khởi chạy một trình duyệt với URL trang web trong EditText phía trên Button. Trình duyệt và trang web phải xuất hiện như được hiển thị bên dưới.
3. Nhập vào nút **Open Location** để khởi chạy bản đồ với vị trí trong EditText phía trên Button. Bản đồ với vị trí phải xuất hiện như được hiển thị bên dưới.
4. Nhập vào nút **Share This Text** để khởi chạy một hộp thoại với các lựa chọn để chia sẻ văn bản. Hộp thoại với các lựa chọn phải xuất hiện như được hiển thị bên dưới.

Task 5: Nhận một intent ngầm định

Cho đến nay, bạn đã tạo một ứng dụng sử dụng một Intent ngầm định để khởi chạy Activity của một ứng dụng khác. Trong task này, bạn xem xét vấn đề theo hướng ngược lại: cho phép một Activity trong ứng dụng của bạn phản hồi một Intent ngầm định được gửi từ một ứng dụng khác.

Một Activity trong ứng dụng của bạn luôn có thể được kích hoạt từ bên trong hoặc bên ngoài ứng dụng của bạn bằng một Intent tường minh. Để cho phép một Activity nhận một Intent ngầm định, bạn xác định một bộ lọc Intent trong tệp AndroidManifest.xml của ứng dụng để cho biết những loại Intent ngầm định 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 định của bạn với một Activity có bộ lọc Intent cho biết rằng chúng có thể thực hiện hành động đó. Nếu có nhiều ứng dụng đã cài đặt phù hợp, người dùng sẽ được cung cấp một trình chọn ứng dụng cho phép họ chọn ứng dụng nào họ muốn sử dụng để xử lý Intent.

Khi một ứng dụng trên thiết bị gửi một Intent ngầm định, hệ thống Android sẽ khớp hành động và dữ liệu của Intent đó với bất kỳ Activity có sẵn nào bao gồm các bộ lọc Intent phù hợp. Khi các bộ lọc Intent cho một Activity khớp với Intent:

- Nếu chỉ có một Activity phù hợp, Android cho phép Activity đó tự xử lý Intent.

- Nếu có nhiều kết quả phù hợp, Android sẽ hiển thị một trình chọn ứng dụng để cho phép người dùng chọn ứng dụng nào họ muốn thực thi hành động đó.

Trong task này, bạn tạo một ứng dụng rất đơn giản nhận một Intent ngầm định để mở URI cho một trang web. Khi được kích hoạt bởi một Intent ngầm định, 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 là **Implicit Intents Receiver** và chọn Empty Views Activity cho mẫu dự án. Chấp nhận tên Activity mặc định (MainActivity). Nhập vào Next.
2. Nhập vào Finish.
3. Mở activity_main.xml.
4. Trong TextView hiện có ("Hello World"), xóa thuộc tính android:text. Không có văn bản nào trong TextView này theo mặc định, nhưng bạn sẽ thêm URI từ Intent trong onCreate().
5. Để nguyên các thuộc tính layout_constraint, nhưng thêm các thuộc tính sau:

Thuộc tính	Giá trị
android:id	"@+id/text_uri_message"
android:textSize	"18sp"
android:textStyle	"bold"

5.2 Sửa đổi AndroidManifest.xml để thêm bộ lọc Intent

1. Mở tệp AndroidManifest.xml.
2. Lưu ý rằng MainActivity đã có bộ lọc Intent này:

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Bộ lọc Intent này, là một phần của manifest dự án mặc định, cho biết rằng Activity này là điểm vào chính cho ứng dụng của bạn (nó có một hành động Intent là "android.intent.action.MAIN") và rằng Activity này sẽ xuất hiện dưới dạng một 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 phần tử này:

```
<intent-filter>
    <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" />
</intent-filter>
```

Các dòng này xác định một bộ lọc Intent cho Activity, đó là loại Intent mà Activity có thể xử lý. Bộ lọc Intent này khai báo các phần tử này:

action	"android.intent.action.VIEW"	Bất kỳ intent nào có hành động xem.
category	"android.intent.category.DEFAULT"	Bất kỳ intent ngầm định nào. Danh mục này phải được bao gồm để activity của bạn nhận được bất kỳ intent ngầm định nào.
category	"android.intent.category.BROWSABLE"	Yêu cầu về các liên kết có thể duyệt được từ các trang web, email hoặc các nguồn khác.
data	android:scheme="http" android:host="developer.android.com"	URI chứa lược đồ của http và tên máy chủ của developer.android.com.

Lưu ý rằng bộ lọc dữ liệu có một hạn chế về cả loại liên kết mà nó sẽ chấp nhận và tên máy chủ cho các URI đó. Nếu bạn muốn trình nhận của mình có thể chấp nhận bất kỳ liên kết nào, bạn có thể bỏ qua phần tử <data>.

5.3 Xử lý Intent

Trong phương thức onCreate() cho Activity của bạn, hãy xử lý Intent đến cho bất kỳ dữ liệu hoặc extras nào mà nó bao gồm. Trong trường hợp này, Intent ngầm định đến có URI được lưu trữ trong dữ liệu Intent.

1. Mở MainActivity.
2. Trong phương thức onCreate(), hãy lấy Intent đến đã được sử dụng để kích hoạt Activity
3. Lấy dữ liệu Intent. Dữ liệu Intent luôn là một đối tượng Uri

```
Intent intent = getIntent();
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
5. Trích xuất phần "URI: " ở trên thành một tài nguyên chuỗi (uri_label).
6. Bên trong cùng một khối if đó, hãy lấy TextView cho tin nhắn:
7. Cũng bên trong khối if, hãy đặt văn bản của TextView đó thành URI:

```
if (uri != null) {
    String uri_string = "URI: " + 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 của việc nhận một Intent ngầm định, bạn sẽ chạy cả hai ứng dụng Implicit Intents Receiver và Implicit Intents trên trình giả lập hoặc thiết bị của mình.

1. Chạy ứng dụng Implicit Intents Receiver.

Chạy ứng dụng một mình sẽ hiển thị một Activity trống không có văn bản. Điều này là do Activity được kích hoạt từ trình khởi chạy hệ thống, chứ không phải bằng một Intent từ một ứng dụng khác.

2. Chạy ứng dụng Implicit Intents và nhấp vào Open Website với URI mặc định.

Một trình chọn ứng dụng xuất hiện hỏi bạn có muốn sử dụng trình duyệt mặc định (Chrome trong hình bên dưới) hay ứng dụng Implicit Intents Receiver không. Chọn Implicit Intents Receiver và nhấp vào Just Once. Ứng dụng Implicit Intents Receiver khởi chạy và thông báo hiển thị URI từ yêu cầu ban đầu.

3. Chạm vào nút Back và nhập một URI khác. Nhập vào Open 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.

Tóm tắt

- Một Intent ngầm định cho phép bạn kích hoạt một Activity nếu bạn biết hành động, nhưng không biết ứng dụng hoặc Activity cụ thể nào sẽ xử lý hành động đó.
- Một Activity có thể nhận một Intent ngầm định phải xác định các bộ lọc Intent trong tệp AndroidManifest.xml khớp với một hoặc nhiều hành động và danh mục Intent.
- Hệ thống Android khớp nội dung của một Intent ngầm định và các bộ lọc Intent của bất kỳ Activity có sẵn nào để xác định Activity nào cần kích hoạt. Nếu có nhiều hơn một Activity có sẵn, hệ thống cung cấp một trình chọn để người dùng có thể chọn một cái.
- Lớp ShareCompat.IntentBuilder giúp dễ dàng xây dựng một Intent ngầm định để chia sẻ dữ liệu lên phương tiện truyền thông xã hội hoặc email.

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

3.1) Trình gỡ lỗi

Giới thiệu

Trong các bài thực hành trước, bạn đã sử dụng lớp Log để in thông tin vào nhật ký hệ thống, xuất hiện trong ngăn Logcat trong Android Studio khi ứng dụng của bạn chạy. Thêm các câu lệnh ghi nhật ký vào ứng dụng của bạn là một cách để tìm lỗi và cải thiện hoạt

động của ứng dụng của bạn. Một cách khác là sử dụng trình gỡ lỗi được tích hợp trong Android Studio.

Trong bài thực hành này, bạn sẽ học cách gỡ lỗi ứng dụng của mình trong trình giả lập và trên thiết bị, đặt và xem các điểm ngắt, bước qua mã của bạn và kiểm tra các biến.

Những gì bạn nên biết

Bạn có thể:

- Tạo một dự án Android Studio.
- Sử dụng trình chỉnh sửa bô 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 giả lập và trên thiết bị.
- Đọc và phân tích một dấu vết ngắn xếp, bao gồm "vào sau, ra trước".
- Thêm các câu lệnh nhật ký và xem nhật ký hệ thống (ngăn Logcat) trong Android Studio.

Những gì bạn sẽ học

- Cách chạy ứng dụng của bạn ở chế độ gỡ lỗi trong trình giả lập hoặc trên thiết bị.
- Cách bước qua quá trình thực thi ứng dụng của bạn.
- Cách đặt và sắp xếp các điểm ngắt.
- Cách kiểm tra và sửa đổi các biến trong trình gỡ lỗi.

Những gì bạn sẽ làm

- 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 các biểu thức.
- Xác định và sửa các 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 một Button, ứng dụng sẽ thực hiện phép tính cho Button đó và hiển thị kết quả.

Task 1: Khám phá dự án và ứng dụng SimpleCalc

Đối với bài thực hành này, bạn sẽ không tự xây dựng ứng dụng SimpleCalc. Dự án hoàn chỉnh có sẵn tại SimpleCalc. Trong task này, bạn sẽ 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.
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 ngăn Project > Android nếu nó chưa mở.

1.2 Khám phá bố cục

1. Mở activity_main.xml.
2. Nhấp vào tab Text để xem mã XML.
3. Nhấp vào tab Preview để xem bản xem trước của bố cục.

Kiểm tra mã và thiết kế bố cục XML 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 TextView để hiển thị kết quả.
- Mỗi Button tính toán có trình xử lý nhấp chuột android:onClick riêng (onAdd, OnSub, v.v.).
- TextView cho kết quả không có bất kỳ văn bản nào theo mặc định.

- Hai phần tử EditText có thuộc tính android:inputType và giá trị "numberDecimal". Thuộc tính này cho biết rằng EditText chỉ chấp nhận số làm đầu vào. Bàn phím xuất hiện trên màn hình sẽ chỉ chứa các số. Bạn sẽ tìm hiểu thêm về các loại đầu vào cho các phần tử EditText trong một bài thực hành sau.

1.3 Khám phá mã ứng dụng

- Mở rộng thư mục app > java trong ngăn Project > Android. Ngoài lớp MainActivity, dự án này còn bao gồm một lớp Calculator tiện ích.
 - Mở Calculator và kiểm tra mã. Lưu ý rằng các thao tác mà máy tính có thể thực hiện được xác định bởi Operator enum và tất cả các phương thức thao tác đều là public.
 - Mở MainActivity và kiểm tra mã và nhận xét.
- Tất cả các trình xử lý nhấp chuột android:onClick đã xác định đều gọi phương thức compute() private, với tên thao tác là một trong các giá trị từ liệt kê Calculator.Operator.
 - Phương thức compute() gọi phương thức getOperand() private (đến lượt nó gọi getOperandText()) để truy xuất các giá trị số từ các phần tử EditText.
 - Phương thức compute() sau đó sử dụng một switch trên tên toán hạng để gọi phương thức thích hợp trong phiên bản Calculator (mCalculator).
 - Các phương thức tính toán trong lớp Calculator 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 của phép tính.

1.4 Chạy ứng dụng

Chạy ứng dụng và làm theo các bước sau:

- Nhập cả giá trị số nguyên và dấu phẩy động cho phép tính.
- Nhập các giá trị dấu phẩy động với các phân số thập phân lớn (ví dụ: 1,6753456)
- Chia một số cho không.
- Để trống một hoặc cả hai phần tử EditText và thử bất kỳ phép tính nào.

- Nhập vào tab Logcat ở cuối cửa sổ Android Studio để mở ngăn Logcat (nếu nó 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 được hiển thị trong hình bên dưới và nhật ký hệ thống hiển thị trạng thái của ngăn xếp thực thi vào thời điểm ứng dụng tạo ra lỗi. Dấu vết ngăn xếp thường cung cấp thông tin quan trọng về lý do xảy ra lỗi.

Task 2: Chạy SimpleCalc trong trình gỡ lỗi

Trong task này, bạn sẽ làm quen với trình gỡ lỗi trong Android Studio và học cách đặt điểm ngắt và chạy ứng dụng của bạn ở chế độ gỡ lỗi.

2.1 Bắt đầu và chạy ứng dụng của bạn ở chế độ gỡ lỗi

- Trong Android Studio, chọn Run > Debug app hoặc nhấp vào biểu tượng Debug trên thanh công cụ.
- Nếu ứng dụng của bạn đã chạy, bạn sẽ được hỏi liệu bạn có muốn khởi động lại ứng dụng của mình ở chế độ gỡ lỗi hay không. Nhấp vào Restart app.

Android Studio xây dựng và chạy ứng dụng của bạn trên trình giả lập hoặc trên thiết bị. Gỡ lỗi là như nhau trong cả hai trường hợp. Trong khi Android Studio đang khởi tạo trình gỡ lỗi, bạn có thể thấy một thông báo cho biết "Waiting for debugger" trên thiết bị trước khi bạn có thể sử dụng ứng dụng của mình.

- Nhập vào tab Debug ở cuối cửa sổ Android Studio để hiển thị ngăn Debug (hoặc chọn View > Tool Windows > Debug). Tab Debugger trong ngăn phải được chọn, hiển thị ngăn Debugger.

2.2 Đặt điểm ngắt

Điểm ngắt là một vị trí trong mã của bạn, nơi bạn muốn tạm dừng quá trình thực thi bình thường của ứng dụng để thực hiện các hành động khác, chẳng hạn như kiểm tra các biến hoặc đánh giá các biểu thức hoặc thực thi mã của bạn từng dòng một 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.

1. Mở MainActivity và nhấp vào dòng thứ tư của phương thức compute() (dòng ngay sau câu lệnh try).
2. Nhấp vào máng xối bên trái của ngăn trình chỉnh sửa tại dòng đó, bên cạnh số dòng. Một chấm đỏ xuất hiện tại dòng đó, cho biết một điểm ngắt. Chấm đỏ bao gồm một dấu kiểm nếu ứng dụng đã chạy ở chế độ gỡ lỗi.

Ngoài ra, 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.

Nếu bạn nhấp vào một điểm ngắt do nhầm lẫn, bạn có thể hoàn tác bằng cách nhấp vào điểm ngắt. Nếu bạn nhấp vào một dòng mã không thể thực thi, chấm đỏ sẽ bao gồm một "x" và một cảnh báo sẽ xuất hiện rằng dòng mã không thể thực thi.

3. Trong ứng dụng SimpleCalc, hãy nhập các số vào các phần tử EditText và nhấp vào một trong các phần tử Button tính toán.

Quá trình thực thi ứng dụng của bạn dừng lại khi nó đạt đến điểm ngắt bạn đã đặt và trình gỡ lỗi hiển thị trạng thái hiện tại của ứng dụng của bạn tại điểm ngắt đó như được hiển thị trong hình bên dưới.

2.3 Tiếp tục quá trình thực thi ứng dụng của bạn

Tiếp tục quá trình thực thi ứng dụng của bạn bằng cách chọn Run > Resume Program hoặc nhấp vào biểu tượng Resume ở 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 tiếp theo quá trình thực thi mã đến điểm ngắt.

2.4 Gỡ lỗi một ứng dụng đang chạy

Nếu ứng dụng của bạn đã chạy trên một thiết bị hoặc trình giả lập và bạn quyết định bạn muốn gỡ lỗi ứng dụng đó, bạn có thể chuyển một ứng dụng đang chạy sang chế độ gỡ lỗi.

1. Chạy ứng dụng SimpleCalc bình thường, với biểu tượng Run.
2. Chọn Run > Attach debugger to Android process hoặc nhấp vào biểu tượng Attach trên thanh công cụ.
3. Chọn quy trình của ứng dụng của bạn từ hộp thoại xuất hiện (hiển thị bên dưới). Nhấp vào OK.

Ngăn Debug xuất hiện với ngăn Debugger mở và bây giờ bạn có thể gỡ lỗi ứng dụng của mình như thế bạn đã bắt đầu nó ở chế độ gỡ lỗi.

Task 3: Khám phá các tính năng của trình gỡ lỗi

Trong task 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 một, làm việc với các 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 bạn một lần 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 task cuối cùng.
2. Trong ứng dụng, hãy nhập các số vào cả hai phần tử EditText và nhấp vào nút Add.

Quá trình thực thi ứng dụng của bạn dừng lại tại điểm ngắt mà bạn đã đặt trước đó và ngăn Debugger hiển thị trạng thái hiện tại của ứng dụng. Dòng hiện tại được đánh dấu trong mã của bạn.

3. Nhấp vào nút Step Over ở đầu cửa sổ trình gỡ lỗi.

Trình gỡ lỗi thực thi dòng hiện tại trong phương thức compute() (nơi có điểm ngắt, việc gán cho operandOne) và điểm đánh dấu di chuyển đến dòng tiếp theo trong mã (việc gán cho operandTwo). Ngăn Variables cập nhật để phản ánh trạng thái thực thi mới và các 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 Run > Step Over, hoặc nhấn F8, để bước qua mã của bạn.

4. Tại dòng tiếp theo (việc gán cho operandTwo), nhấp vào biểu tượng Step Into.

Step Into nhảy vào quá trình thực thi một lệnh gọi phương thức trong dòng hiện tại (so với chỉ thực thi phương thức đó và vẫn ở trên cùng một dòng). Trong trường hợp này, vì việc gán đó bao gồm một lệnh gọi đến getOperand(), trình gỡ lỗi cuộn mã MainActivity đến định nghĩa phương thức đó.

Khi bạn bước vào một phương thức, ngăn Frames cập nhật để cho biết khung mới trong ngăn xếp lệnh gọi (ở đây, getOperand()) và ngăn Variables hiển thị các biến có sẵn trong phạm vi phương thức mới. Bạn có thể nhấp vào bất kỳ dòng nào trong ngăn Frames để xem điểm trong khung ngăn xếp trước đó, nơi phương thức được gọi.

Bạn cũng có thể sử dụng Run > Step Into, hoặc F7, để bước vào một phương thức.

1. Nhấp vào Step Over để chạy từng dòng trong getOperand(). Lưu ý rằng khi phương thức hoàn tất, trình gõ lỗi sẽ trả bạn về điểm mà bạn đã bước vào phương thức lần đầu tiên và tắt cả các bảng điều khiển sẽ cập nhật để hiển thị thông tin mới.
2. Nhấp vào Step Over hai lần để di chuyển điểm thực thi đến dòng đầu tiên bên trong câu lệnh case cho ADD.
3. Nhấp 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.

4. Sử dụng biểu tượng Step Out để thực thi phần còn lại của phương thức tính toán đó và bật trở 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 quá trình thực thi phương thức.

3.2 Làm việc với các điểm ngắt

Sử dụng các điểm ngắt để cho biết vị trí trong mã của bạn, nơi bạn muốn làm gián đoạn quá trình thực thi ứng dụng của bạn để gõ lỗi phần đó của ứng dụng đó.

1. Tìm điểm ngắt bạn đã đặt trong task cuối cùng—ở đầu phương thức compute() trong MainActivity.
2. Thêm một điểm ngắt vào đầu câu lệnh switch.
3. Nhấp chuột phải vào điểm ngắt mới đó để nhập một điều kiện, như được hiển thị trong hình bên dưới và nhập bài kiểm tra sau vào trường Condition:

(operandOne == 42)|| (operandTwo == 42)

4. Nhập vào Done.

Điểm ngắt thứ hai này là một điểm ngắt có điều kiện. Quá trình thực thi ứng dụng của bạn sẽ chỉ dừng lại tại điểm ngắt này nếu bài kiểm tra trong điều kiện là true. Trong trường hợp này, biểu thức chỉ đúng nếu một trong các toán hạng bạn đã nhập là 42. Bạn có thể nhập bất kỳ biểu thức Java nào làm điều kiện miễn là nó trả về một boolean.

5. Chạy ứng dụng của bạn ở chế độ gỡ lỗi (Run > Debug) hoặc nhấp vào Resume nếu nó đã chạy. Trong ứng dụng, hãy nhập hai số khác 42 và nhấp vào nút Add. Quá trình thực thi tạm dừng tại điểm ngắt đầu tiên trong phương thức compute().
6. Nhấp vào Resume để tiếp tục gỡ lỗi ứng dụng. Quan sát thấy 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, hãy nhập 42 vào EditText đầu tiên và nhấp vào bất kỳ Button nào. Nhấp vào Resume để tiếp tục thực thi sau điểm ngắt đầu tiên. Quan sát thấy điểm ngắt thứ hai tại câu lệnh switch—điểm ngắt có điều kiện—dừng quá trình thực thi vì điều kiện đã được đáp ứng.
8. Nhấp chuột phải (hoặc Control-click) vào điểm ngắt đầu tiên trong compute() và bỏ chọn Enabled. Nhấp vào Done. Quan sát thấy biểu tượng điểm ngắt bây giờ có một chấm xanh lục với đường viền đỏ.

Tắt một 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 một điểm ngắt hoàn toàn, bạn cũng sẽ mất bất kỳ điều kiện nào bạn đã tạo cho điểm ngắt đó, vì vậy việc tắt nó thường là một lựa chọn tốt hơn.

Bạn cũng có thể tắt tiếng tất cả các điểm ngắt trong ứng dụng của mình cùng một lúc bằng biểu tượng Mute Breakpoints.

9. Nhấp vào View Breakpoints ở cạnh trái của cửa sổ trình gỡ lỗi. Cửa sổ Breakpoints xuất hiện.

Cửa sổ Breakpoints 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 các điểm ngắt, bao gồm các điều kiện, sự phụ thuộc vào các điểm ngắt khác và ghi nhật ký.

Để đóng cửa sổ Breakpoints, nhấp vào Done.

3.3 Kiểm tra và sửa đổi các biến

Trình gỡ lỗi Android Studio cho phép bạn kiểm tra trạng thái của các biến trong ứng dụng của bạn khi ứng dụng đó chạy.

1. Chạy ứng dụng SimpleCalc ở chế độ gỡ lỗi nếu nó 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 compute() vẫn bị tắt tiếng. Quá trình thực thi dừng lại tại điểm ngắt thứ hai (điểm ngắt có điều kiện tại câu lệnh switch) và trình gỡ lỗi xuất hiện.

3. Quan sát thấy trong ngăn Variables, các biến operandOne và operandTwo có các giá trị bạn đã nhập vào ứng dụng.
4. Biến this 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 lại để đóng danh sách.
5. Nhấp chuột phải (hoặc Control-click) vào biến operandOne trong ngăn Variables và chọn Set Value.
6. Thay đổi giá trị của operandOne thành 10 và nhấn Return.
7. Thay đổi giá trị của operandTwo thành 10 theo cùng một cách và nhấn Return.
8. Quan sát thấy kết quả trong ứng dụng bây giờ dựa trên các giá trị biến bạn đã thay đổi trong trình gỡ lỗi; ví dụ: vì bạn đã nhấp vào nút Add trong Bước 2, kết quả trong ứng dụng bây giờ là 20.
9. Nhấp vào biểu tượng Resume để tiếp tục chạy ứng dụng của bạn.
10. Trong ứng dụng, các mục nhập ban đầu (bao gồm 42) được giữ nguyên trong các phần tử EditText. (Giá trị của chúng chỉ được thay đổi trong trình gỡ lỗi.) Nhấp vào nút Add. Quá trình thực thi tạm dừng lại tại điểm ngắt.
11. Nhấp vào biểu tượng Evaluate Expression hoặc chọn Run > Evaluate Expression. Bạn cũng có thể nhấp chuột phải (hoặc Control-click) vào bất kỳ biến nào và chọn Evaluate Expression.

Cửa sổ Evaluate Code Fragment xuất hiện. Sử dụng nó 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.

12. Nhập câu lệnh mOperandOneEditText.getHint(); vào trường trên cùng của cửa sổ Evaluate Code Fragment (như được hiển thị trong hình trên) và nhấp vào Evaluate.
13. Trường Result hiển thị kết quả của biểu thức đó. Gợi ý cho EditText này là chuỗi "Type Operand 1", như ban đầu được xác định trong XML cho EditText đó.

Kết quả bạn nhận được từ việc đánh giá một biểu thức dựa trên trạng thái hiện tại của ứng dụng. Tùy thuộc vào các giá trị của các biến trong ứng dụng của bạn vào thời điểm bạn đánh giá các biểu thức, bạn có thể nhận được các kết quả khác nhau.

Cũng lưu ý rằng nếu bạn sử dụng Evaluate Expression để thay đổi các giá trị của các biến hoặc thuộc tính đối tượng, bạn sẽ thay đổi trạng thái đang chạy của ứng dụng.

14. Nhấp vào Close để đóng cửa sổ Evaluate Code Fragment.

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 gỡ lỗi bằng cách nhấp vào biểu tượng Debug hoặc chọn Run > Debug app.
- Nhấp vào tab Debug để hiển thị ngăn Debug. Nhấp vào tab Debugger trong ngăn Debug để hiển thị ngăn Debugger (nếu nó chưa được chọn).
- Ngăn Debugger hiển thị (ngăn xếp) Frames, Variables trong một khung cụ thể và Watches (theo dõi tích cực một biến trong khi chương trình chạy).
- Điểm ngắt là một vị trí trong mã của bạn, nơi 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 gỡ lỗi bằng cách nhấp vào máng xói bên trái của cửa sổ trình soạn thảo ngay bên cạnh dòng mục tiêu.

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

1.1) Hình ảnh có thể chọn

1.2) Các điều khiển nhập liệu

1.3) Menu và bộ chọn

1.4) Điều hướng người dùng

1.5) RecyclerView

Bài 2) Trải nghiệm người dùng thú vị

2.1) Hình vẽ, định kiểu và chủ đề

2.2) Thẻ và màu sắc

2.3) Bộ cục thích ứng

Bài 3) Kiểm thử giao diện người dùng

3.1) Espresso cho việc kiểm tra UI

CHƯƠNG 3. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

1.1) AsyncTask

1.2) AsyncTask và AsyncTaskLoader

1.3) Broadcast receivers

Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền

2.1) Thông báo

2.2) Trình quản lý cảnh báo

2.3) JobScheduler

CHƯƠNG 4. LUU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

1.1) Shared preferences

1.2) Cài đặt ứng dụng

Bài 2) Lưu trữ dữ liệu với Room

2.1) Room, LiveData và ViewModel

2.2) Room, LiveData và ViewModel