

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



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

SINH VIÊN: HOÀNG TRỌNG VINH - 2251061925 – 64CNTT1

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1. Làm quen	4
Bài 1) Tạo ứng dụng đầu tiên.....	4
1.1) Android Studio và Hello World	4
1.2) Giao diện người dùng tương tác đầu tiên.....	4
1.3) Trình chỉnh sửa bộ cục	107
1.4) Văn bản và các chế độ cuộn.....	Error! Bookmark not defined.
1.5) Tài nguyên có sẵn	Error! Bookmark not defined.
Bài 2) Activities	141
2.1) Activity và Intent	141
2.2) Vòng đời của Activity và trạng thái	174
2.3) Intent ngầm định	190
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	219
3.1) Trình gỡ lỗi.....	219
3.2) Kiểm thử đơn vị	Error! Bookmark not defined.
3.3) Thư viện hỗ trợ.....	251
CHƯƠNG 2. Trải nghiệm người dùng	275
Bài 1) Tương tác người dùng.....	275
1.1) Hình ảnh có thể chọn	275
1.2) Các điều khiển nhập liệu.....	275
1.3) Menu và bộ chọn.....	275
1.4) Điều hướng người dùng	275
1.5) RecycleView	275
Bài 2) Trải nghiệm người dùng thú vị	275
2.1) Hình vẽ, định kiểu và chủ đề.....	275
2.2) Thẻ và màu sắc.....	275
2.3) Bộ cục thích ứng	275
Bài 3) Kiểm thử giao diện người dùng	275

3.1) Espresso cho việc kiểm tra UI.....	275
CHƯƠNG 3. Làm việc trong nền	275
Bài 1) Các tác vụ nền	275
1.1) AsyncTask	275
1.2) AsyncTask và AsyncTaskLoader	275
1.3) Broadcast receivers	275
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền	275
2.1) Thông báo.....	275
2.2) Trình quản lý cảnh báo.....	275
2.3) JobScheduler	275
CHƯƠNG 4. Lưu trữ dữ liệu người dùng	276
Bài 1) Tùy chọn và cài đặt	276
1.1) Shared preferences	276
1.2) Cài đặt ứng dụng	276
Bài 2) Lưu trữ dữ liệu với Room	276
2.1) Room, LiveData và ViewModel	276
2.2) Room, LiveData và ViewModel	276
3.1) Trinhf gowx loi

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

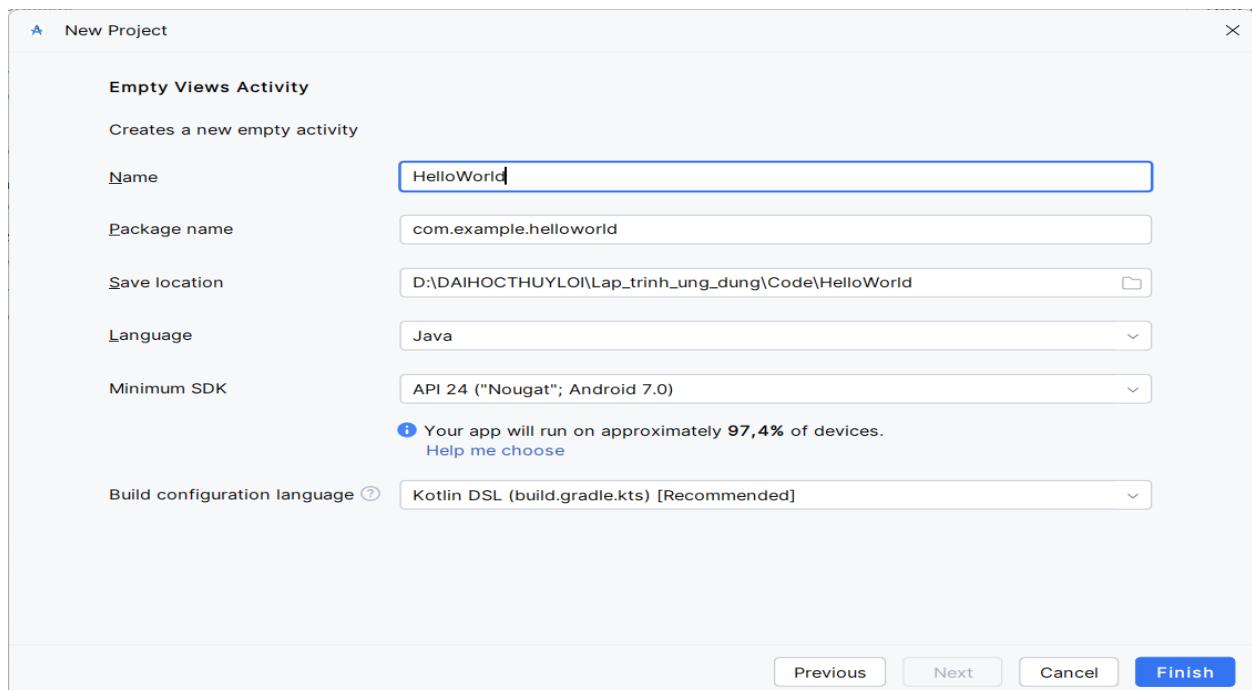
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

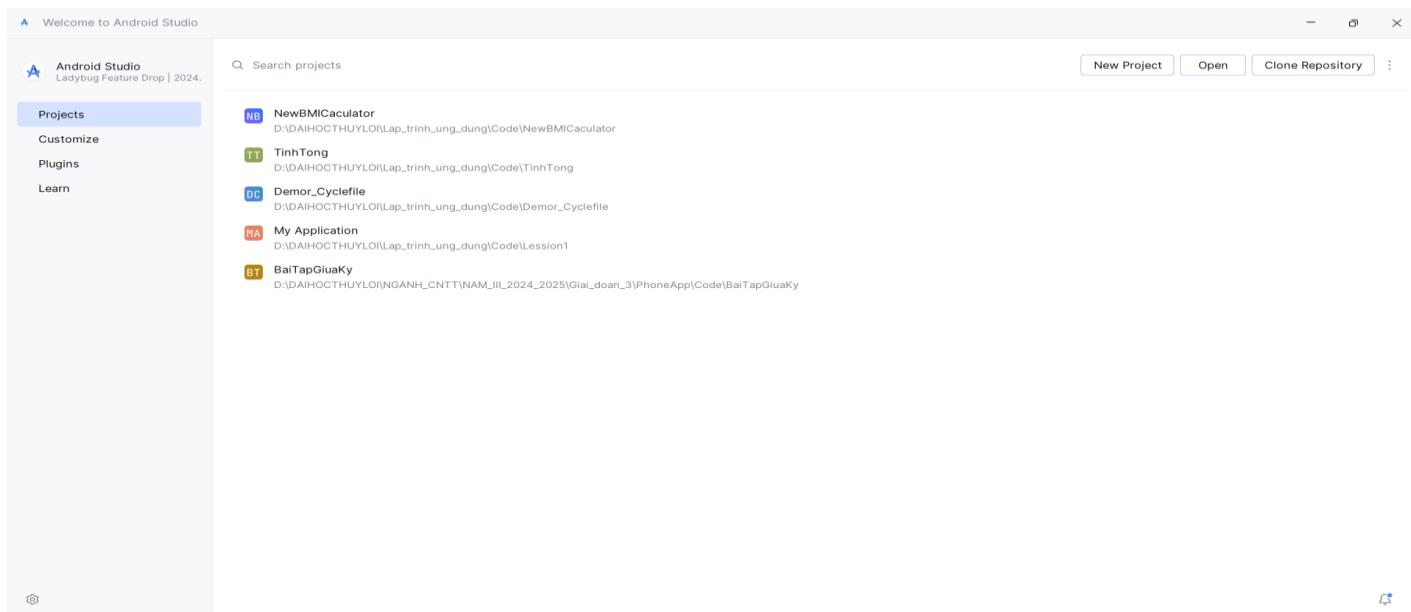
- Cách cài đặt và sử dụng IDE Android Studio.
- Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android.
- Cách tạo một dự án Android từ một mẫu.
- Cách thêm thông điệp ghi lại vào ứng dụng của bạn để phục vụ mục đích gỡ lỗi.

Những gì bạn sẽ làm

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

TASK 1:Cài đặt Android Studio

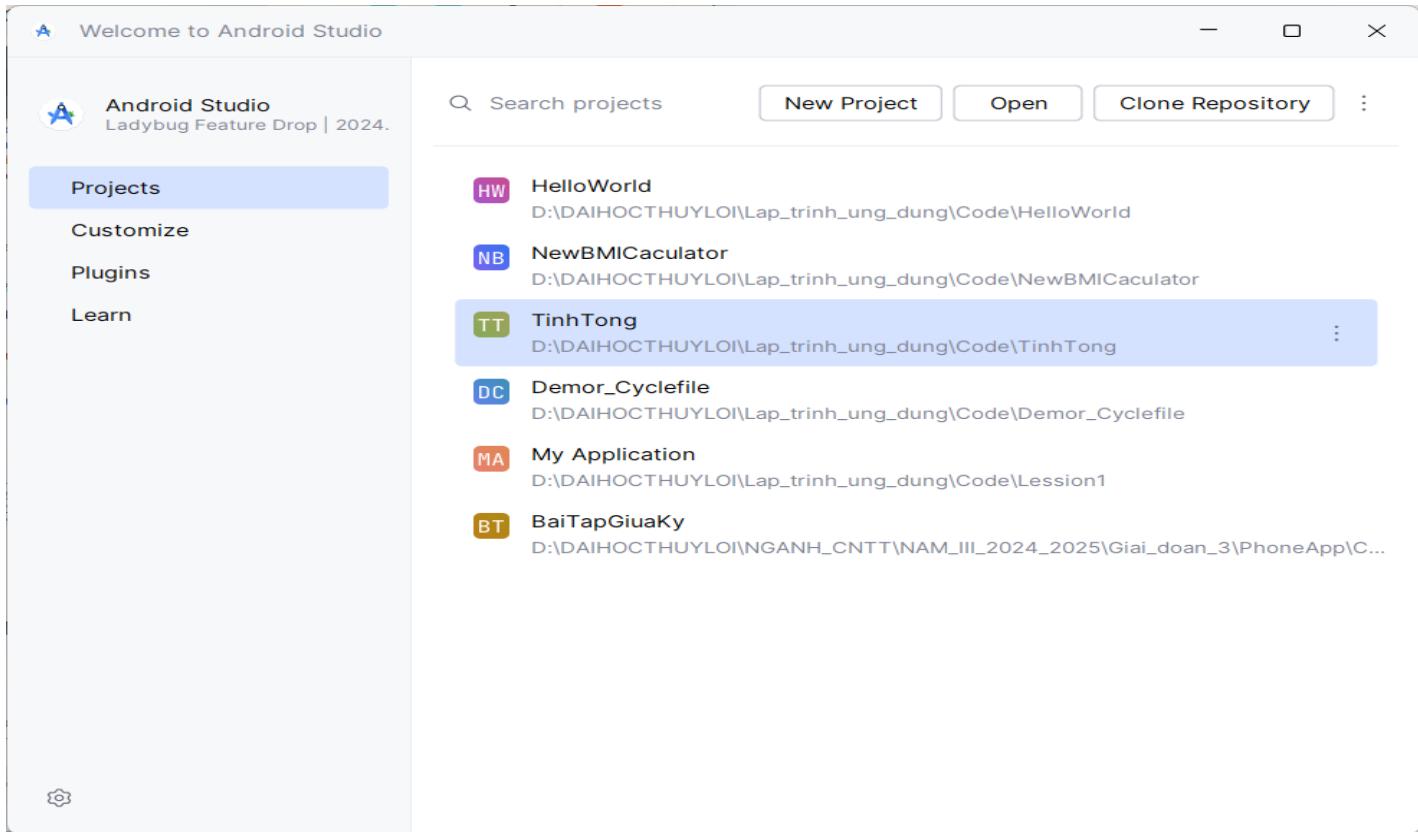
Ứng dụng đã được cài và có giao diện như trên khi đã hoàn tất cài đặt



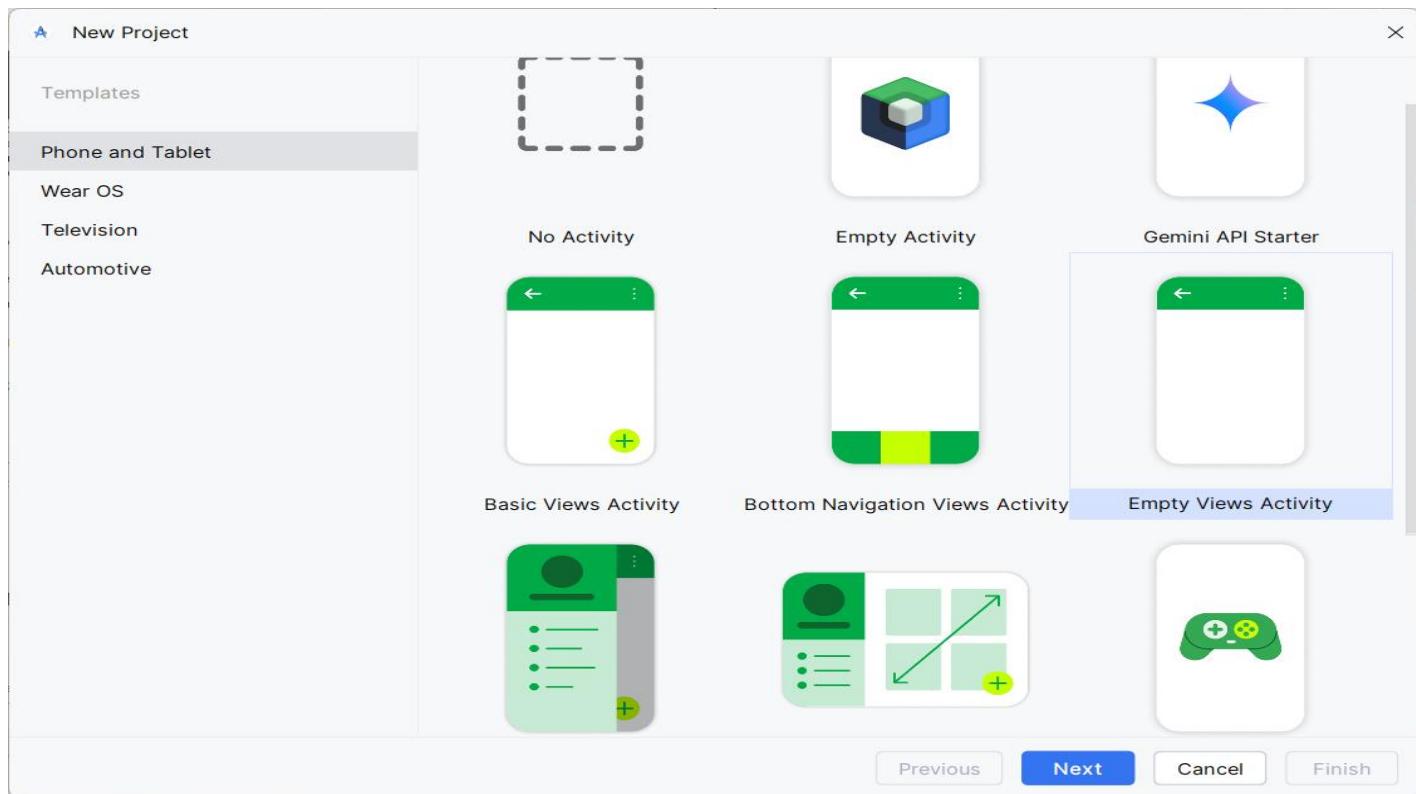
TASK 2:Tạo ứng dụng Hello World

2.1 Tạo một dự án

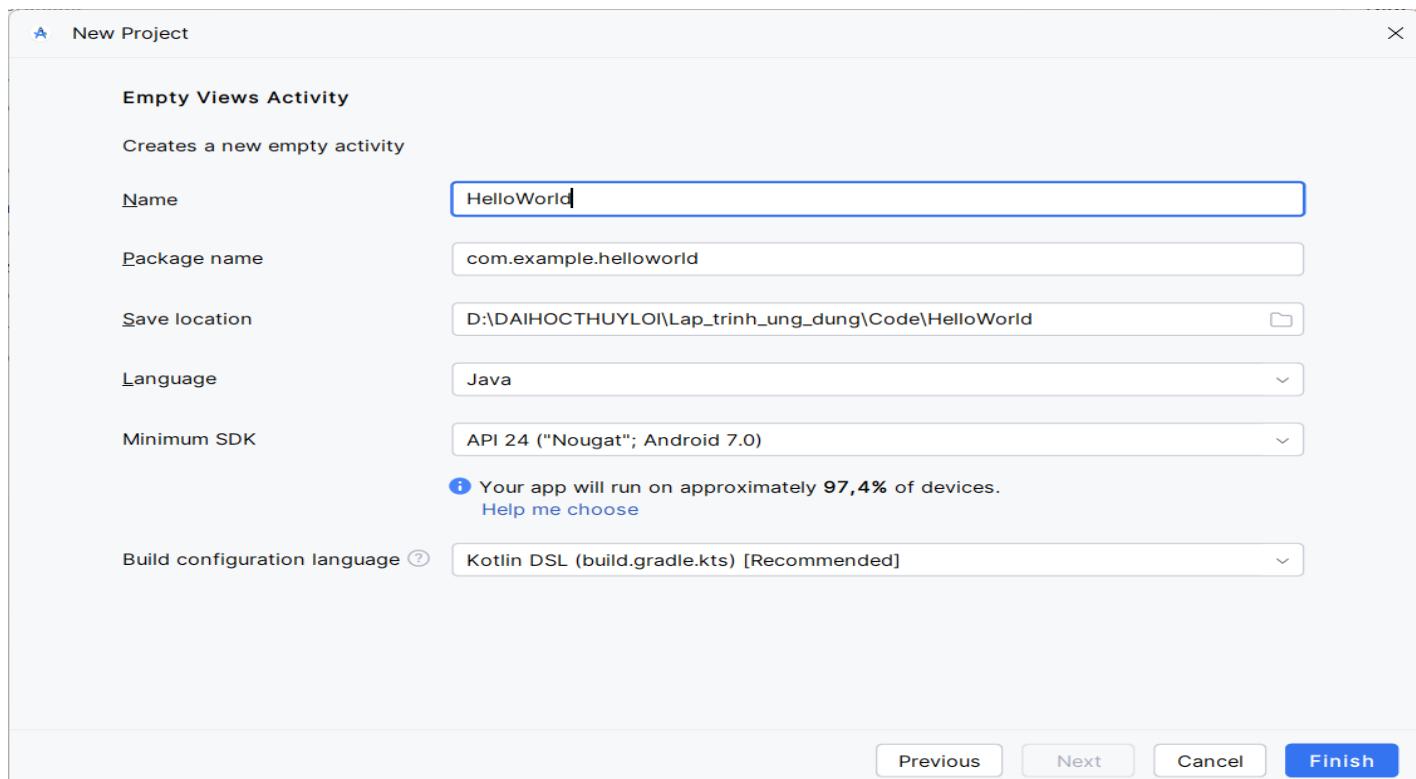
Ở trang giao diện chọn new project



Sau đó chọn Empty Views Activity

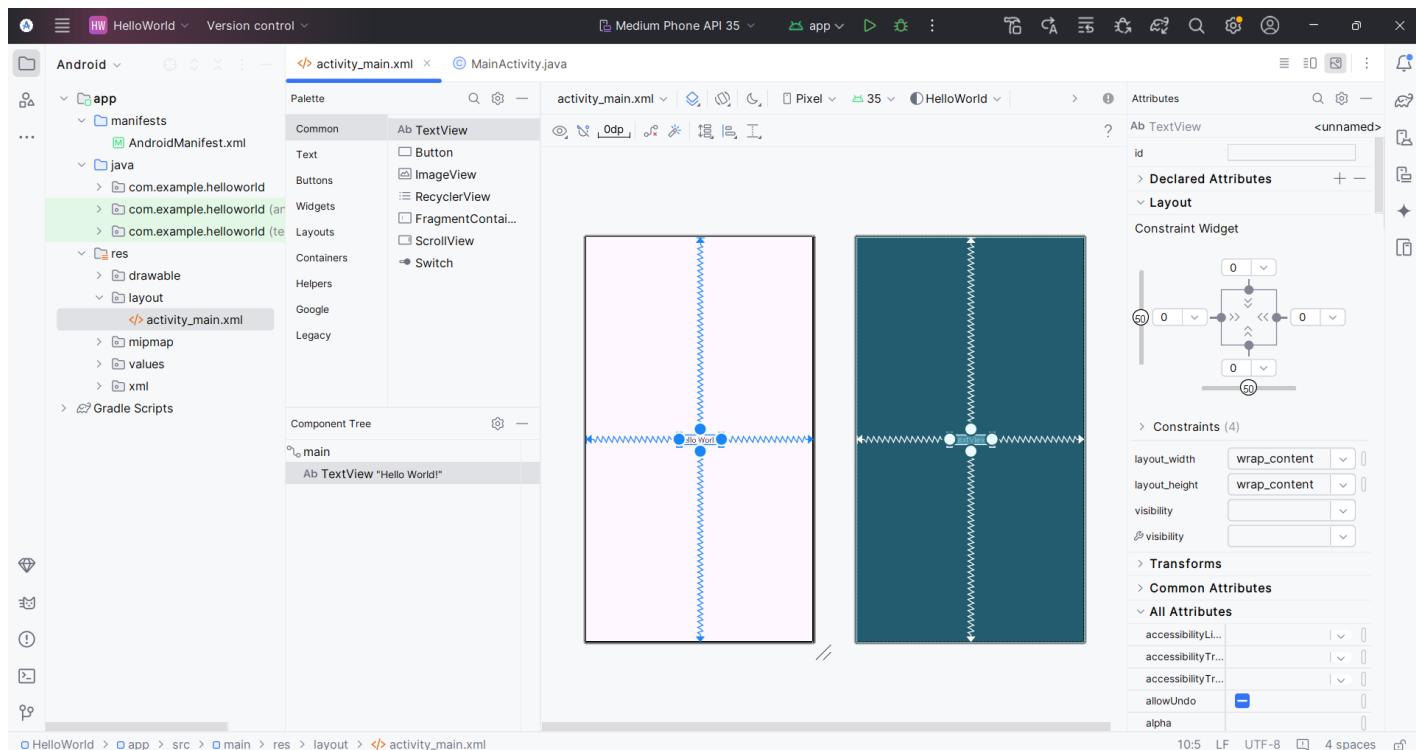


Sau đó ấn **next** rồi sẽ điền tên dự án , chọn ngôn ngữ code là java hoặc kotlin sau đó ấn **Finish**



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

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



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

The screenshot shows the Android Studio interface. On the left, the Project tool window displays the project structure under the 'Android' tab, including the app module with its manifest and Java files. The 'app' file is selected. In the center, the code editor shows the MainActivity.java file with the following code:

```
1 package com.example.helloworld;
2
3 > import ...
4
5 > public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
11        setContentView(R.layout.activity_main);
12        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
13            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
14            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
15        });
16    }
17}
18
19
20
21
22
23
24
```

The code editor has syntax highlighting and code completion suggestions. At the bottom, the status bar shows the project path: HelloWord > app > src > main > java > com > example > helloworld > MainActivity. On the right, there are various toolbars and status indicators.

2.2 Khám phá ngăn Project > Android

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

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

The screenshot shows the Android Studio interface. On the left, the Project Navigators pane is open, showing the project structure under the 'Android' tab. It includes 'Project', 'Packages', 'Production', 'Tests', 'Project Source Files', 'Project Non-Source Files', 'Open Files', 'Scratches and Consoles', and 'Gradle Scripts'. Under 'Android', there are 'layout' (containing 'activity_main.xml'), 'mipmap', 'values', and 'xml'. The main editor window displays the 'MainActivity.java' file, which contains Java code for an AppCompatActivity. The code includes imports, package declarations, and an onCreate method. The status bar at the bottom shows the path 'HelloWorld > app > src > main > java > com > example > helloworld > MainActivity' and settings like '24:2 LF UTF-8 4 spaces'.

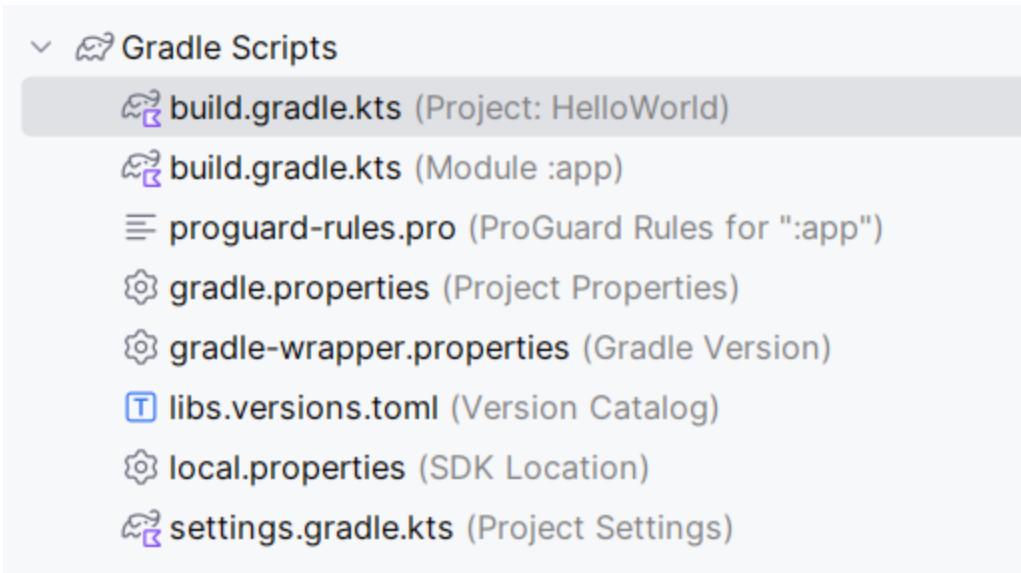
```
1 package com.example.helloworld;
2
3 > import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
11        setContentView(R.layout.activity_main);
12        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
13            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
14            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
15        });
16    }
17
18 }
19
20
21
22
23
24
```

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

2.3 Khám phá thư mục Gradle Scripts

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

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



Hãy làm theo các bước sau để khám phá hệ thống Gradle:

1. Nếu thư mục Tập lệnh Gradle không được mở rộng, hãy nhấp vào hình tam giác để mở rộng.

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

2. Tìm tệp build.gradle(Project: HelloWorld).

Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các module tạo nên dự án của bạn. Mỗi dự án Android Studio đều chứa một tệp Gradle build cấp cao nhất. Trong hầu hết các trường hợp, bạn sẽ không cần thay đổi gì trong tệp này, nhưng việc hiểu nội dung của nó vẫn rất hữu ích.

Theo mặc định, tệp build cấp cao nhất sử dụng khôi buildscript để định nghĩa các kho lưu trữ (repositories) và các phụ thuộc (dependencies) của Gradle chung cho tất cả các module trong dự án. Khi phụ thuộc của bạn không phải là một thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm kiếm các tệp trong các kho lưu trữ trực tuyến được chỉ định trong khôi repositories của tệp này. Theo mặc định, các dự án Android Studio mới khai báo JCenter và Google (bao gồm cả kho lưu trữ Maven của Google) làm các vị trí kho lưu trữ.

The screenshot shows the Android Studio interface with the project 'HelloWorld' selected. The left sidebar displays the project structure under 'Android'. The main editor area shows the 'build.gradle.kts (HelloWorld)' file, which contains the following code:

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
    alias(libs.plugins.android.application) apply false
}
```

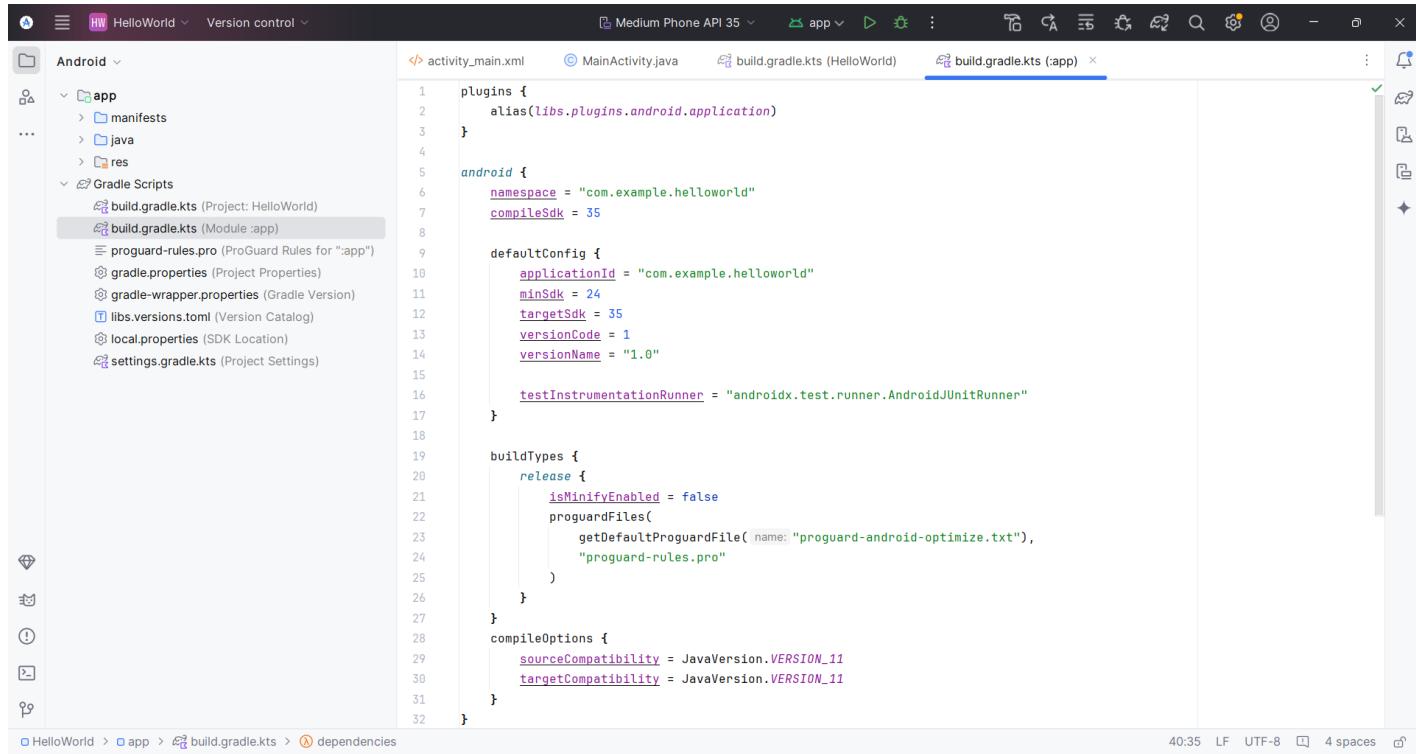
The status bar at the bottom indicates the file is 4:2 LF, UTF-8, 4 spaces.

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

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

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

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



The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the file tree on the left, showing the project structure with an expanded 'app' module. The main area contains the code editor with the 'build.gradle.kts (app)' tab selected. The code is as follows:

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

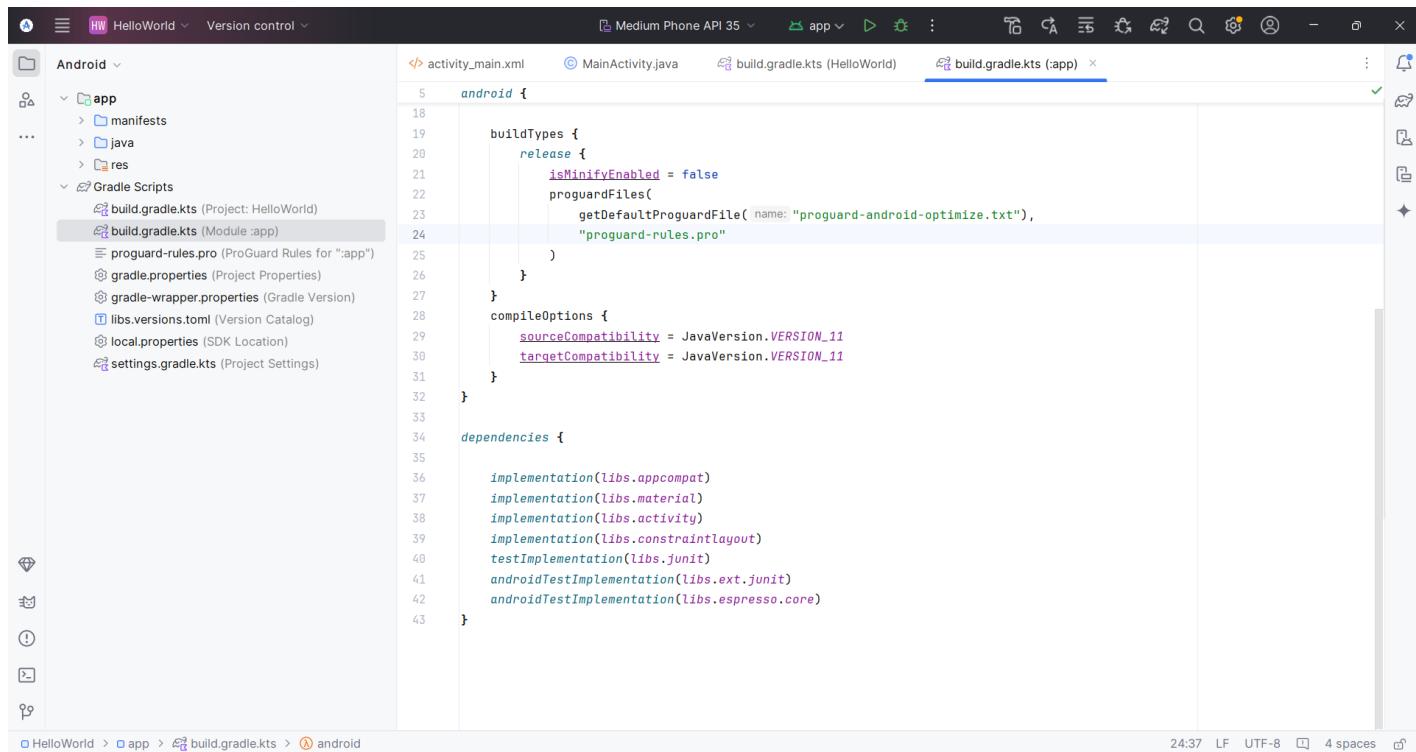
android {
    namespace = "com.example.helloworld"
    compileSdk = 35

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

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

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}
```

The status bar at the bottom indicates the file is 4035 lines long, uses LF line endings, is in UTF-8 encoding, and has 4 spaces per indentation.



This screenshot is similar to the previous one, but the 'android' block in the code editor is currently selected and highlighted. The code remains the same as above.

```
5 android {
18
19     buildTypes {
20         release {
21             isMinifyEnabled = false
22             proguardFiles(
23                 getDefaultProguardFile("proguard-android-optimize.txt"),
24                 "proguard-rules.pro"
25             )
26         }
27     }
28     compileOptions {
29         sourceCompatibility = JavaVersion.VERSION_11
30         targetCompatibility = JavaVersion.VERSION_11
31     }
32 }

33 dependencies {
34
35     implementation(libs.appcompat)
36     implementation(libs.material)
37     implementation(libs.activity)
38     implementation(libs.constraintlayout)
39     testImplementation(libs.junit)
40     androidTestImplementation(libs.ext.junit)
41     androidTestImplementation(libs.espresso.core)
42
43 }
```

The status bar at the bottom indicates the file is 2437 lines long, uses LF line endings, is in UTF-8 encoding, and has 4 spaces per indentation.

4. Nhập vào hình tam giác để đóng Tập lệnh Gradle.

2.4 Khám phá thư mục app và res

Tất cả mã nguồn và tài nguyên của ứng dụng đều nằm trong các thư mục **app** và **res**.

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

The screenshot shows the Android Studio interface. On the left, the project navigation pane displays the directory structure:

- Android
- app
 - manifests
 - java
 - com.example.helloworld (selected)
 - MainActivity
 - com.example.helloworld (androidTest)
 - com.example.helloworld (test)
 - res
- Gradle Scripts

The right side of the screen shows the code editor for **MainActivity.java**:

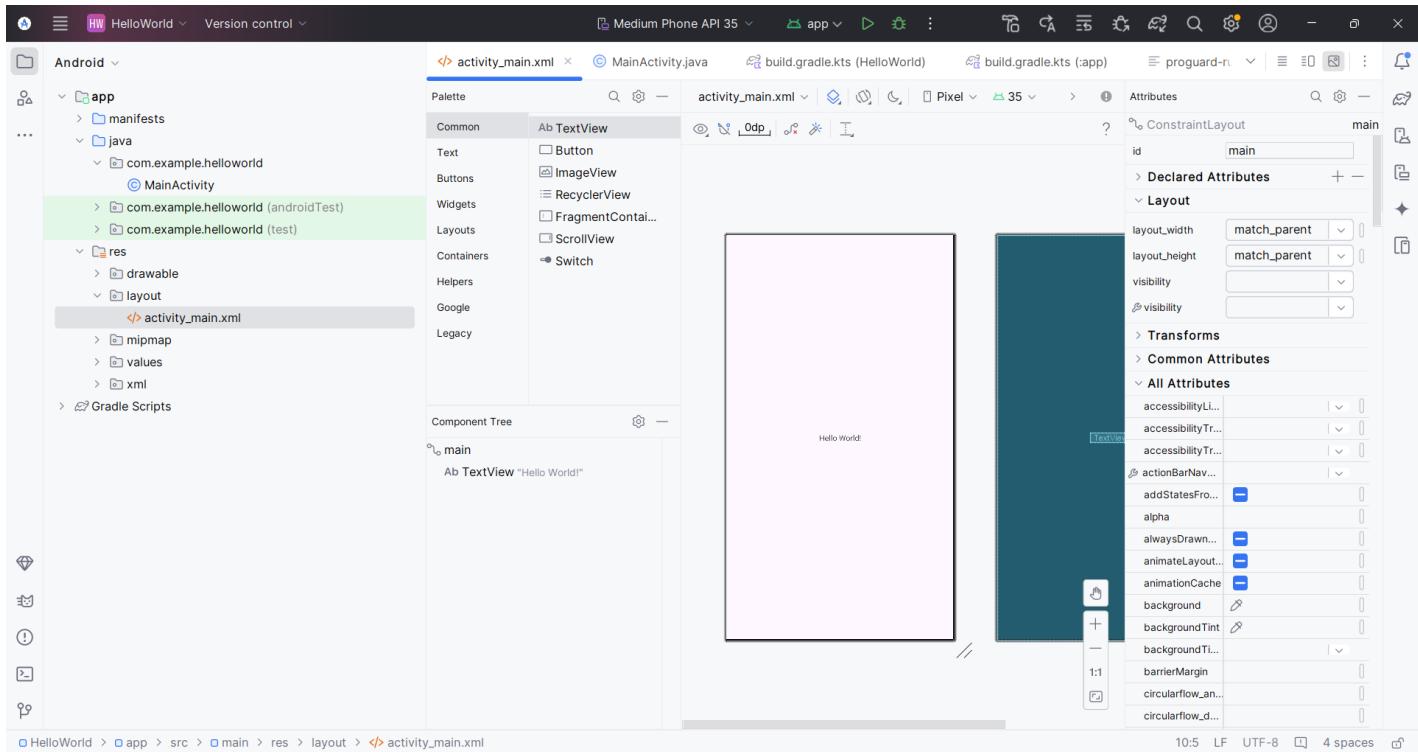
```
1 package com.example.helloworld;
2
3 > import ...
4
5 >< public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        EdgeToEdge.enable( $this$enableEdgeToEdge: this );
11        setContentView(R.layout.activity_main);
12        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
13            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
14            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
15        });
16    }
17
18 }
19
20
21
22
23 }
```

The code implements a custom window inset listener for the main activity's view to handle notch padding.

Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như được hiển thị trong hình trên. Thư mục **com.example.hello.helloworld** (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục còn lại được sử dụng cho mục đích kiểm thử và sẽ được mô tả trong một bài học khác. Đối với ứng dụng Hello World, chỉ có một gói duy nhất và nó chứa tệp **MainActivity.java**. Tên của **Activity** (màn hình) đầu tiên mà người dùng nhìn thấy, đồng thời cũng là nơi

khởi tạo các tài nguyên toàn ứng dụng, thường được gọi là **MainActivity** (phần mở rộng tệp được bỏ qua trong khung **Project > Android**).

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



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

2.5 Khám phá thư mục **manifests**

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

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

Tệp **AndroidManifest.xml** mô tả tất cả các thành phần của ứng dụng Android của bạn. Tất cả các thành phần của ứng dụng, chẳng hạn như mỗi **Activity**, đều phải

được khai báo trong tệp XML này. Trong các bài học khác, bạn sẽ chỉnh sửa tệp này để thêm các tính năng và quyền truy cập tính năng. Để biết thêm thông tin giới thiệu, hãy xem **Tổng quan về App Manifest**.

TASK 3: Sử dụng thiết bị ảo (emulator)

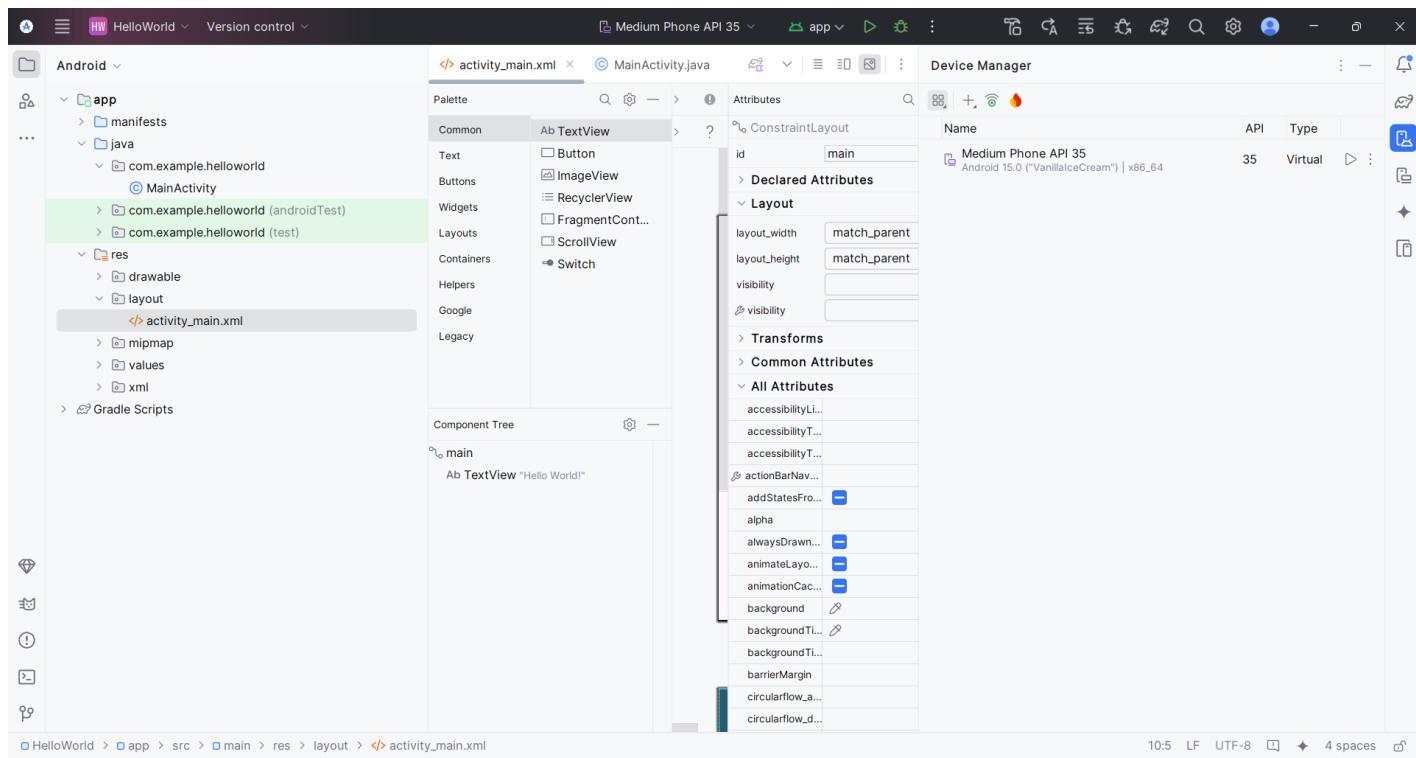
Trong nhiệm vụ này, bạn sẽ sử dụng trình quản lý Android Virtual Device (AVD) để tạo một thiết bị ảo (còn được gọi là emulator) mô phỏng cấu hình cho một loại thiết bị Android cụ thể, và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng Android Emulator có các yêu cầu bổ sung ngoài các yêu cầu hệ thống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

Bằng cách sử dụng trình quản lý AVD, bạn có thể định nghĩa các đặc điểm phần cứng của thiết bị, mức API, bộ nhớ, giao diện (skin) và các thuộc tính khác, sau đó lưu nó dưới dạng một thiết bị ảo. Với các thiết bị ảo, bạn có thể kiểm thử ứng dụng trên các cấu hình thiết bị khác nhau (chẳng hạn như máy tính bảng và điện thoại) với các mức API khác nhau mà không cần sử dụng thiết bị vật lý.

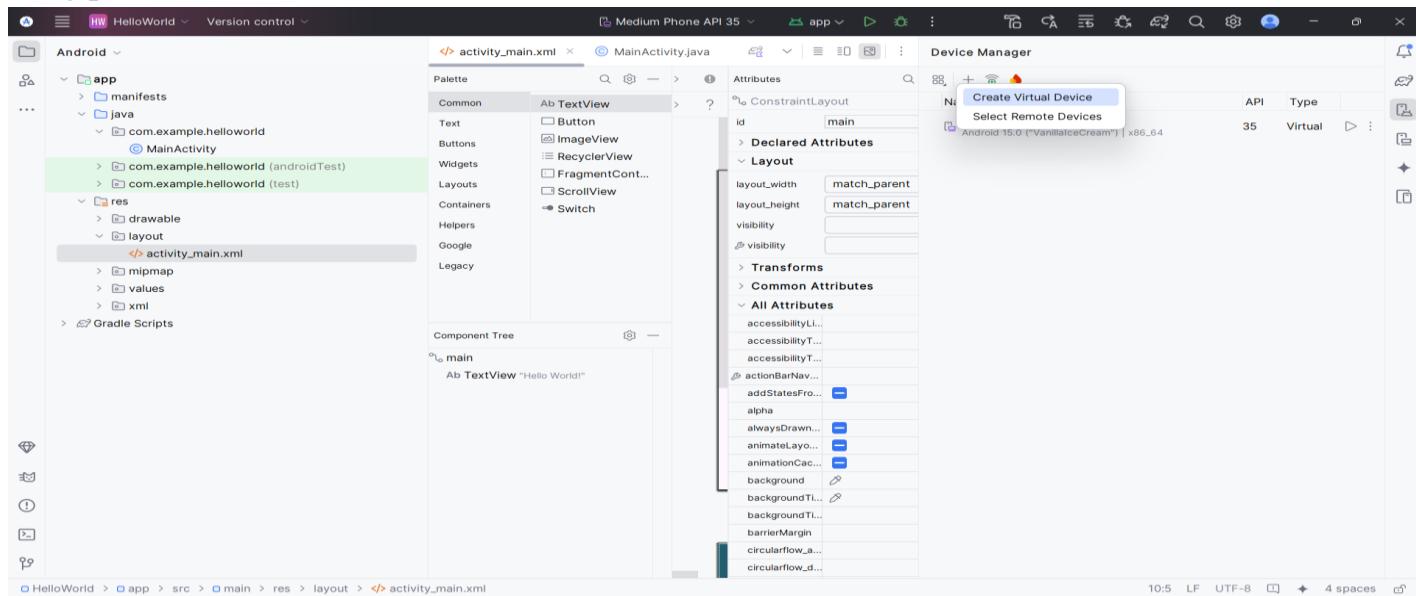
3.1 Tạo một thiết bị ảo Android (AVD)

Để chạy một emulator trên máy tính của bạn, bạn cần tạo một cấu hình mô tả thiết bị ảo.

Bước 1 : Vào Device Manager phía thanh dọc bên phải màn hình



Nhập vào +Create Virtual Device. Cửa sổ Select Hardware sẽ xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng cung cấp một cột checkbox thước màn hình chéo (Size), độ phân giải màn hình tính bằng pixel.



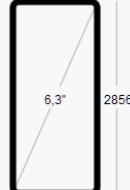
Virtual Device Configuration

Select Hardware

Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
Phone	Small Phone	▶	4,65"	720x1280	xhdpi
Tablet	Medium Phone	▶	6,4"	1080x2400	420dpi
Wear OS	Pixel 9 Pro XL	▶	6,8"	1344x2992	xxhdpi
Desktop	Pixel 9 Pro Fold	▶	8,0"	2076x2152	390dpi
TV	Pixel 9 Pro	▶	6,3"	1280x2856	xxhdpi
Automotive	Pixel 9	▶	6,24"	1080x2424	420dpi
Legacy	Pixel 8a	▶	6,1"	1080x2400	420dpi

Pixel 9 Pro



Size: large
Ratio: long
Density: xxhdpi

New Hardware Profile Import Hardware Profiles Refresh Clone Device... ? Previous Next Cancel Finish

Chọn một thiết bị như Nexus 5x hoặc Pixel 9 Pro và nhấp vào Tiếp theo. Màn hình Ảnh hệ thống xuất hiện.

Nhấp vào tab Đè xuất nếu chưa chọn và chọn phiên bản hệ thống Android nào để chạy trên thiết bị ảo (như Oreo).

Virtual Device Configuration

System Image

Select a system image

Document was last saved: Just now

Recommended x86 Images Other Images

Release Name	API	ABI	xABI	Target
Baklava ↴	Baklava	x86_64		Android API Baklava (Google Play)
Baklava ↴	Baklava	x86_64		Android API Baklava (16 KB Page)
VanillalceCream	35	x86_64	arm64-v8a	Android 15.0 (Google Play)
VanillalceCream ↴	35	x86_64		Android 15.0 (16 KB Page)
UpsideDownCake ↴	34	x86_64		Android 14.0 (Google Play)
Tiramisu ↴	33	x86_64		Android 13.0 (Google Play)
Sv2 ↴	32	x86_64		Android 12L (Google Play)
S ↴	31	x86_64		Android 12.0 (Google Play)
R ↴	30	x86		Android 11.0 (Google Play)

VanillalceCream

API Level: 35

Type: Google Play

Android: 15.0

Google Inc.

System Image: x86_64 (translated: arm64-v8a)

We recommend these Google Play images because this device is compatible with Google Play.

?

Previous Next Cancel Finish

Virtual Device Configuration

Android Virtual Device (AVD)

Verify Configuration

AVD name: Pixel 9 Pro API 35

Pixel 9 Pro 6.3 1280x2856 xxhdpi Change...

VanillalceCream Android 15.0 x86_64 Change...

Preferred ABI: Optimal

Startup orientation: Portrait Landscape

Show Advanced Settings

Default Orientation

Sets the initial orientation of the device. During AVD emulation you can also rotate the device screen.

?

Previous Next Cancel Finish

Có nhiều phiên bản khả dụng hơn so với những phiên bản được hiển thị trong tab Đè xuất. Hãy xem các tab x86 Hình ảnh và Hình ảnh khác để xem chúng.

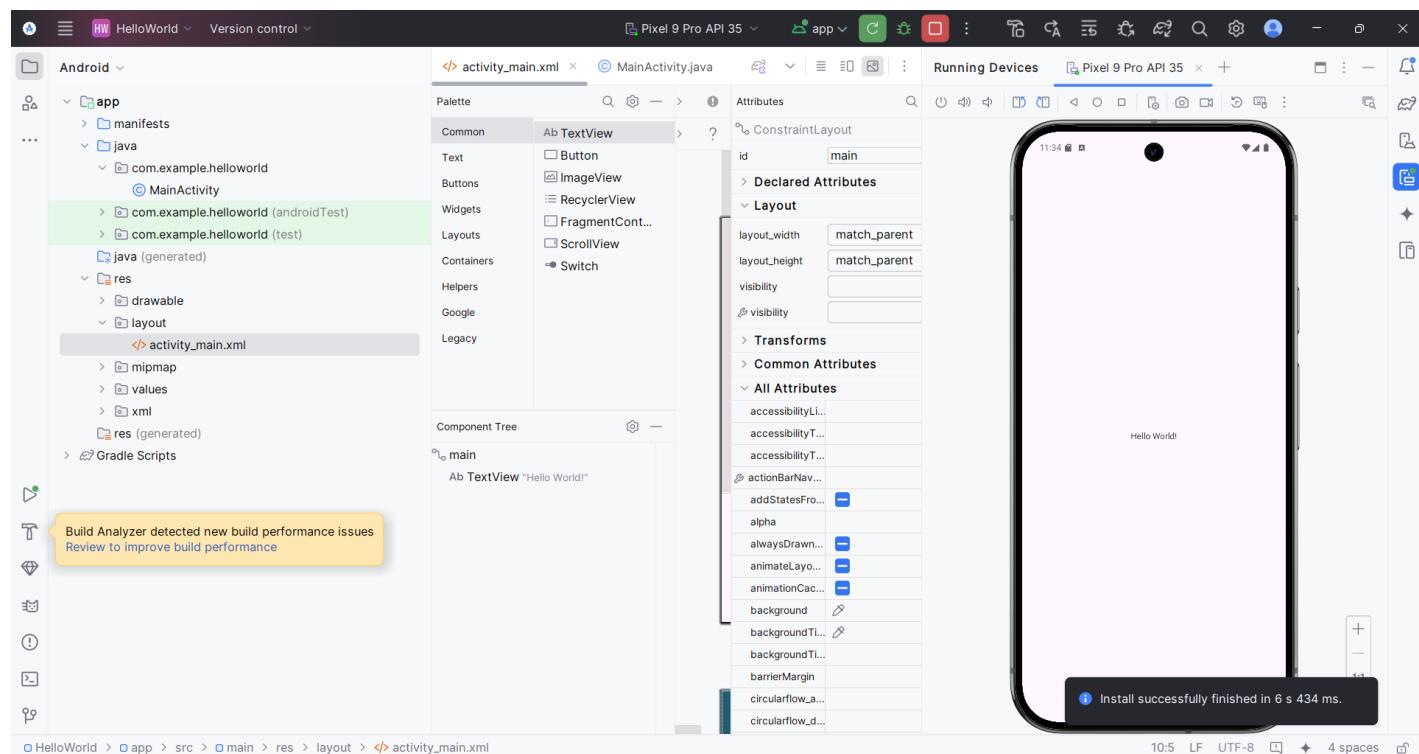
Nếu liên kết Tải xuống hiển thị bên cạnh hình ảnh hệ thống mà bạn muốn sử dụng, thì hình ảnh đó vẫn chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống và nhấp vào Hoàn tất khi hoàn tất.

5. Sau khi chọn hình ảnh hệ thống, hãy nhấp vào Tiếp theo. Cửa sổ Thiết bị ảo Android (AVD) sẽ xuất hiện.

Bạn cũng có thể thay đổi tên của AVD. Kiểm tra cấu hình của bạn và nhấp vào Hoàn tất.

3.2 Chạy ứng dụng trên thiết bị ảo

Sau khi tạo thành công và chạy với dự án hello world với máy ảo Pixel 9 Pro vừa tạo được giao diện như sau:



4.1 Bật chế độ USB Debugging

Để cho phép Android Studio giao tiếp với thiết bị của bạn, bạn phải bật **USB Debugging** trên thiết bị Android của mình. Tính năng này được kích hoạt trong phần cài đặt **Developer options** của thiết bị.

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

1. Trên thiết bị của bạn, mở **Settings**, tìm kiếm **About phone**, nhấp vào **About phone**, và nhấn vào **Build number** bảy lần.
2. Quay lại màn hình trước đó (**Settings / System**). **Developer options** sẽ xuất hiện trong danh sách. Nhấn vào **Developer options**.
3. Chọn **USB Debugging**.

4.2 Chạy ứng dụng của bạn trên thiết bị

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

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

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

Khắc phục sự cố

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

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

Nếu máy tính của bạn vẫn không tìm thấy thiết bị hoặc thông báo thiết bị "không được ủy quyền", hãy làm theo các bước sau:

1. Rút thiết bị ra.
2. Trên thiết bị, mở **Developer Options** trong ứng dụng **Settings**.
3. Nhấn vào **Revoke USB Debugging authorizations** (Thu hồi quyền ủy quyền USB Debugging).

4. Kết nối lại thiết bị với máy tính của bạn.

5. Khi được nhắc, hãy cấp quyền ủy quyền.

Bạn có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Hãy xem tài liệu **Using Hardware Devices**.

TASK 5: Thay đổi cấu hình Gradle của ứng dụng

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

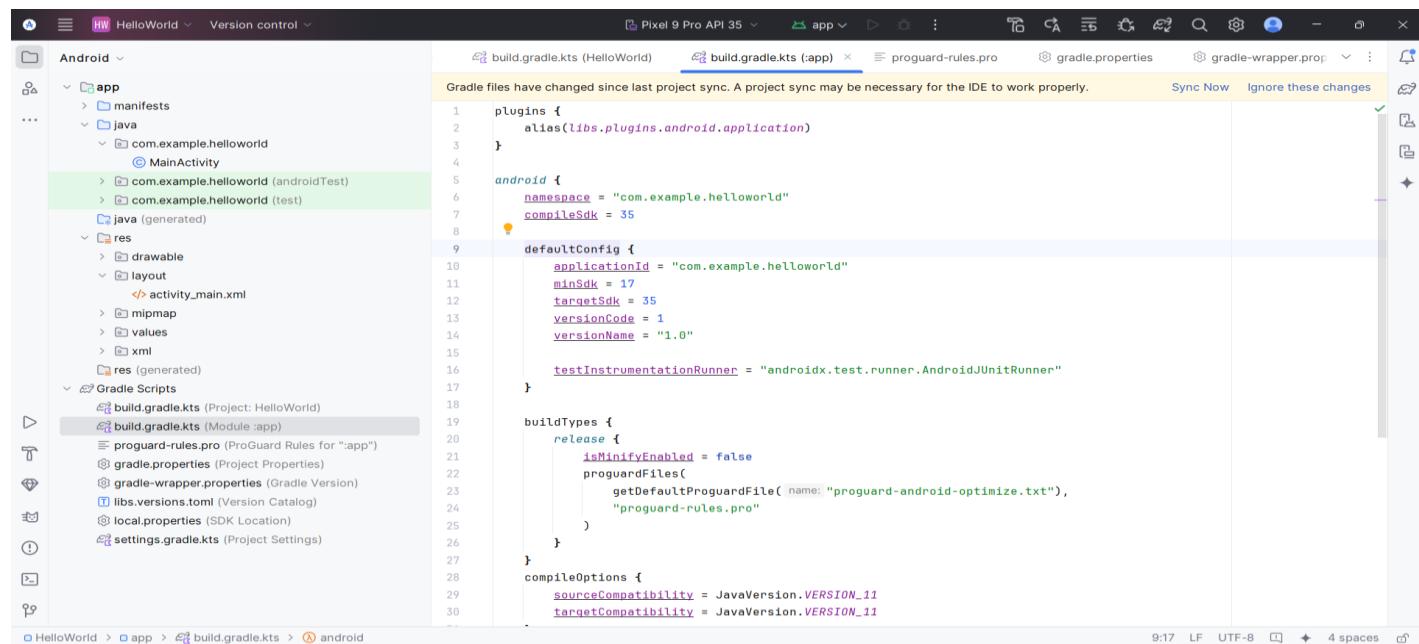
5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Làm theo các bước sau:

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

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

2. Trong khối **defaultConfig**, thay đổi giá trị của **minSdkVersion** thành **17** như hình dưới đây (ban đầu nó được đặt là **24**).



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

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

defaultConfig {
    applicationId = "com.example.helloworld"
    minSdk = 17
    targetSdk = 35
    versionCode = 1
    versionName = "1.0"

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

buildTypes {
    release {
        isMinifyEnabled = false
        proguardFiles(
            getDefaultProguardFile("proguard-android-optimize.txt"),
            "proguard-rules.pro"
        )
    }
}

compileOptions {
    sourceCompatibility = JavaVersion.VERSION_11
    targetCompatibility = JavaVersion.VERSION_11
}
```

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

5.2 Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi trong các tệp cấu hình build của dự án, Android Studio yêu cầu bạn đồng bộ các tệp dự án để nó có thể nhập các thay đổi cấu hình build và chạy một số kiểm tra để đảm bảo rằng cấu hình sẽ không gây ra lỗi build.

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

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

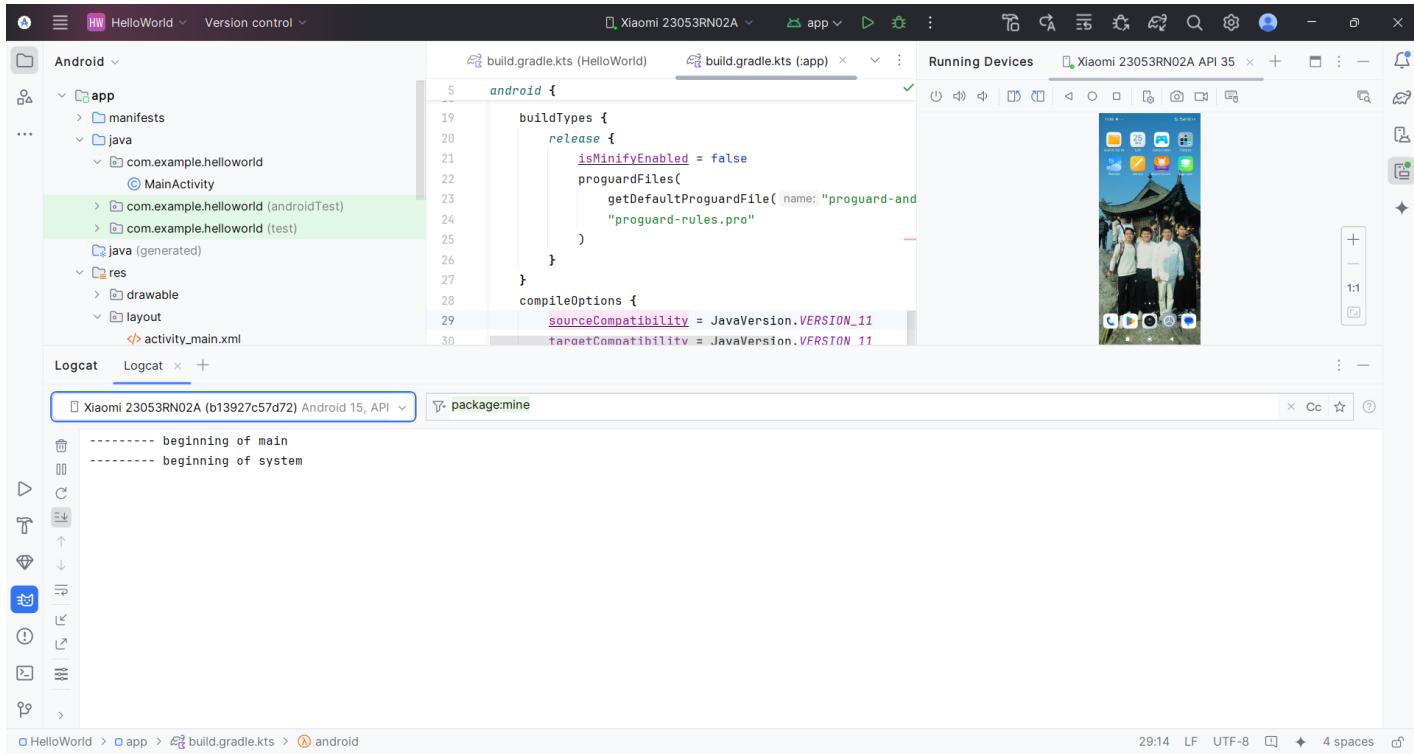
Để tìm hiểu sâu hơn về Gradle, hãy xem tài liệu **Build System Overview** và **Configuring Gradle Builds**.

Nhiệm vụ 6: Thêm các câu lệnh Log vào ứng dụng

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

6.1 Xem khung Logcat

Để xem khung **Logcat**, nhấp vào tab **Logcat** ở cuối cửa sổ Android Studio như hình dưới đây.



Trong hình trên:

1. Tab Logcat để mở và đóng ngắt Logcat, hiển thị thông tin về ứng dụng của bạn khi ứng dụng đang chạy. Nếu bạn thêm câu lệnh Log vào ứng dụng, thông báo Log sẽ xuất hiện ở đây.
2. Menu cấp độ Log được đặt thành Verbose (mặc định), hiển thị tất cả thông báo Log.

Các thiết lập khác bao gồm Debug, Error, Info và Warn.

6.2 Thêm câu lệnh log vào ứng dụng của bạn

Câu lệnh log trong mã ứng dụng của bạn sẽ hiển thị thông báo trong ngăn Logcat. Ví dụ:

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

Các phần của thông báo là:

- Nhật ký: Lớp Nhật ký để gửi thông báo nhật ký đến ngăn Logcat.

- d: Cài đặt mức Nhật ký gỡ lỗi để lọc hiển thị thông báo nhật ký trong ngắn Logcat. Các mức nhật ký khác là e cho Lỗi, w cho Cảnh báo và i cho Thông tin.
- "MainActivity": Đối số đầu tiên là một thẻ có thể được sử dụng để lọc thông báo trong ngắn

Logcat. Đây thường là tên của Hoạt động mà thông báo bắt nguồn. Tuy nhiên, bạn có thể biến điều này thành bất kỳ thứ gì hữu ích cho bạn để gỡ lỗi.

Theo quy ước, thẻ nhật ký được định nghĩa là hằng số cho Hoạt động

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

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

Thực hiện theo các bước sau:

1. Mở ứng dụng Hello World của bạn trong Android studio và mở MainActivity.

2. Để tự động thêm các mục nhập rõ ràng vào dự án của bạn (chẳng hạn như android.util.Log

cần thiết để sử dụng Log), hãy chọn File > Settings trong Windows hoặc Android Studio >

Preferences trong macOS.

3. Chọn Editor > General > Auto Import. Chọn tất cả các hộp kiểm và đặt Insert imports on

paste thành All .

4. Nhập vào Apply rồi nhập vào OK.

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

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

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

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

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

6.Nếu ngăn Logcat chưa mở, hãy nhập vào tab Logcat ở cuối Android Studio để mở nó.

7. Kiểm tra xem tên mục tiêu và tên gói của ứng dụng có đúng không.

8. Thay đổi mức Nhật ký trong ngăn Logcat thành Gõ lỗi (hoặc để nguyên là Chi tiết vì có quá ít thông báo nhật ký).

9. Chạy ứng dụng của bạn.

Thông báo sau sẽ xuất hiện trong ngăn Logcat:

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của một thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là View — mọi thành phần trên màn hình đều là một View. Lớp View đại diện cho khối xây dựng cơ bản cho tất cả các thành phần UI và là lớp cơ sở cho các lớp cung cấp các thành phần UI tương tác như nút bấm, hộp kiểm và trường nhập văn bản. Các lớp con của View thường được sử dụng, sẽ được mô tả trong các bài học tiếp theo, bao gồm:

- TextView: Hiển thị văn bản.
- EditText: Cho phép người dùng nhập và chỉnh sửa văn bản.
- Button và các phần tử có thể nhập khác (như RadioButton, CheckBox, và Spinner): Cung cấp hành vi tương tác.
- ScrollView và RecyclerView: Hiển thị các mục có thể cuộn.
- ImageView: Hiển thị hình ảnh.
- ConstraintLayout và LinearLayout: Chứa các phần tử View khác và định vị chúng.

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

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

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

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

Những gì bạn cần biết trước

Bạn nên quen thuộc với:

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

Những gì bạn sẽ học

- Cách tạo một ứng dụng có hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế bố cục.
- Cách chỉnh sửa bố cục trong XML.
- Rất nhiều thuật ngữ mới. Hãy xem **Từ vựng và khái niệm** để biết các định nghĩa dễ hiểu.

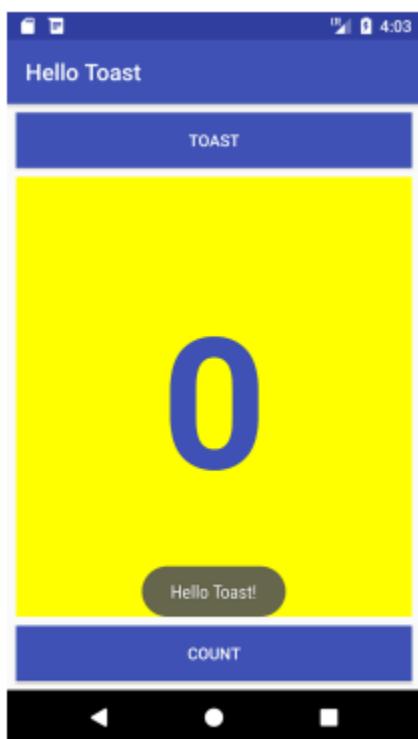
Những gì bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử **Button** và một **TextView** vào bố cục.
- Thao tác với từng phần tử trong **ConstraintLayout** để ràng buộc chúng với lề và các phần tử khác.
- Thay đổi các thuộc tính của phần tử UI.
- Chỉnh sửa bố cục của ứng dụng trong XML.
- Trích xuất các chuỗi mã cứng (hardcoded strings) thành tài nguyên chuỗi (string resources).
- Triển khai các phương thức xử lý sự kiện nhấp (click-handler methods) để hiển thị thông báo trên màn hình khi người dùng nhấp vào mỗi **Button**.

Tổng quan về ứng dụng

Ứng dụng **HelloToast** bao gồm hai phần tử **Button** và một **TextView**. Khi người dùng nhấn vào **Button** đầu tiên, một thông báo ngắn (Toast) sẽ xuất hiện trên màn hình. Nhấn vào **Button** thứ hai sẽ tăng một bộ đếm "click" hiển thị trong **TextView**, bắt đầu từ số 0.

Đây là giao diện của ứng dụng hoàn thiện:



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

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

1.1 Tạo dự án Android Studio

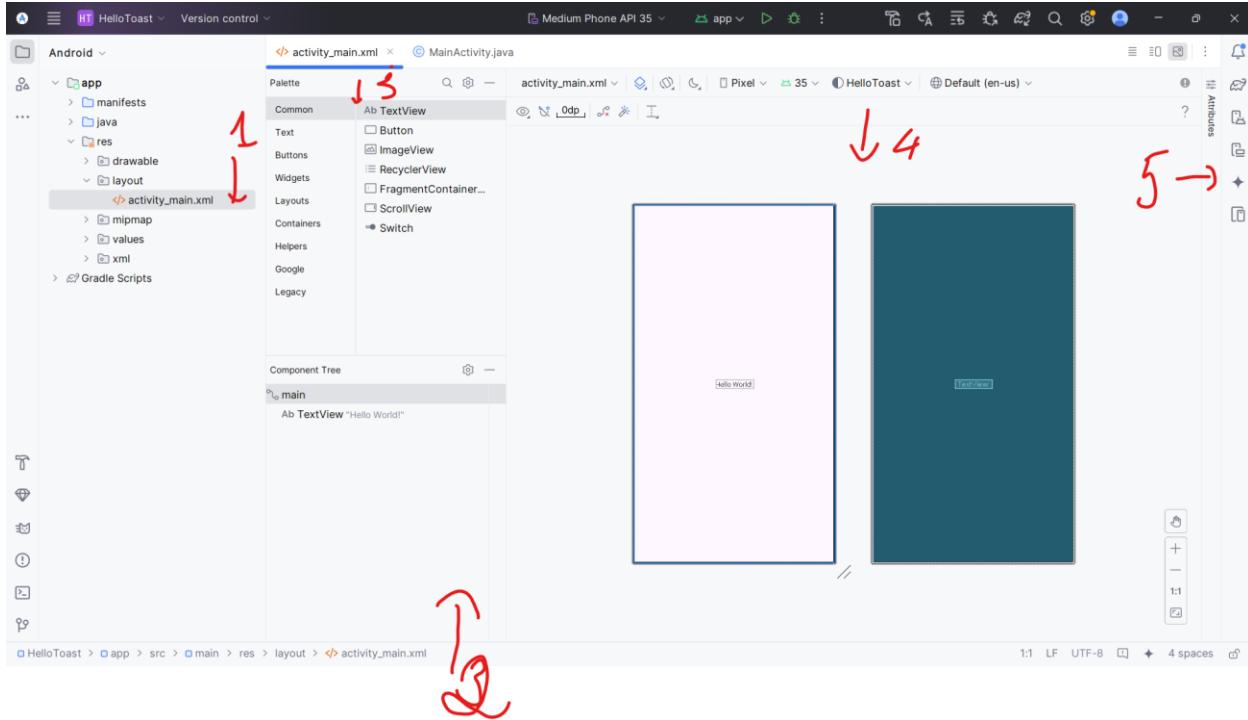
14.Khởi động Android Studio và tạo một dự án mới với các thông số sau:

Attribute	Value
Application Name	Hello Toast
Company Name	com.example.android (or your own domain)

Phone and Tablet Minimum SDK	API24
Template	Empty View Activity
Generate Layout file box	Selected
Backwards Compatibility box	Selected

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

1.2 Khám phá trình chỉnh sửa bố cục



1. Trong thư mục **app > res > layout** trong bảng **Project > Android**, nhấp đúp vào tệp **activity_main.xml** để mở nó, nếu chưa được mở.

2. Bảng **Component tree** hiển thị cấu trúc phân cấp của các phần tử UI. Các phần tử view được tổ chức theo một cấu trúc cây, trong đó một phần tử con kế thừa các thuộc tính từ phần tử cha. Trong hình trên, **TextView** là phần tử con của **ConstraintLayout**. Bạn sẽ tìm hiểu về các phần tử này sau trong bài học.

3. Bảng **Palettes** hiển thị các phần tử giao diện người dùng (UI) mà bạn có thể sử dụng trong bố cục của ứng dụng.

4.Các bảng thiết kế và bản vẽ của trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình trên, bố cục chỉ hiển thị một phần tử: **TextView** với nội dung "Hello World".

5. Tab **Attributes** hiển thị bảng thuộc tính để thiết lập các thuộc tính cho một phần tử UI. **Mẹo:** Xem phần **Building a UI with Layout Editor** để biết chi tiết về cách sử dụng trình chỉnh sửa bố cục, và **Meet Android Studio** để xem toàn bộ tài liệu của Android Studio.

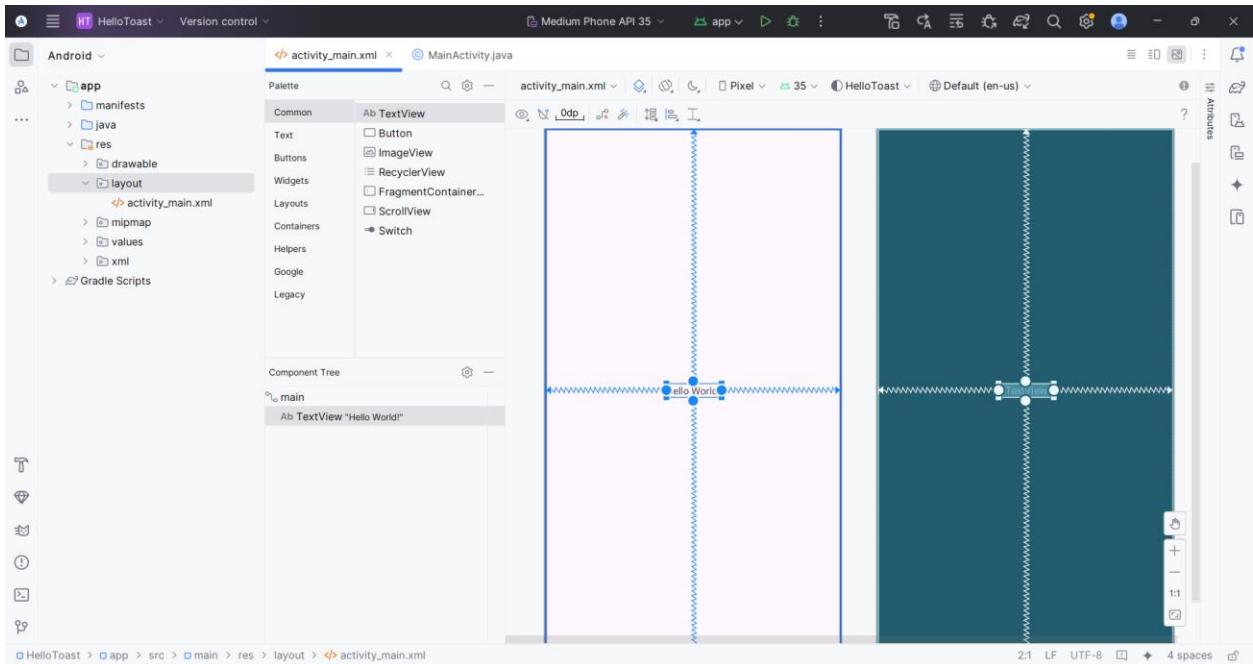
Nhiệm vụ 2: Thêm các phần tử View trong trình chỉnh sửa bố cục

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

2.1 Kiểm tra các ràng buộc phần tử

Thực hiện theo các bước sau:

1. Mở tệp **activity_main.xml** từ bảng **Project > Android** nếu nó chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào nó.
Nếu không có chế độ bản vẽ, hãy nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**.
2. Công cụ **Autoconnect** cũng nằm trên thanh công cụ. Mặc định, công cụ này được kích hoạt. Đối với bước này, đảm bảo rằng công cụ không bị tắt.
3. Nhấp vào nút **zoom in** để phóng to các bảng thiết kế và bản vẽ để xem chi tiết.
4. Chọn **TextView** trong bảng **Component Tree**. **TextView** với nội dung "Hello World" sẽ được tô sáng trong các bảng thiết kế và bản vẽ, và các ràng buộc cho phần tử sẽ hiển thị.
5. Tham khảo hình ảnh động dưới đây cho bước này. Nhấp vào tay cầm hình tròn ở phía bên phải của **TextView** để xóa ràng buộc ngang liên kết view với phía bên phải của bố cục. **TextView** sẽ nhảy sang phía bên trái vì nó không còn bị ràng buộc vào bên phải nữa. Để thêm lại ràng buộc ngang, nhấp vào tay cầm đó và kéo một đường tới phía bên phải của bố cục.



Trong các bảng thiết kế hoặc bản vẽ, các tay cầm sau xuất hiện trên phần tử **TextView**:

- **Constraint handle:** Để tạo một ràng buộc như đã trình bày trong hình ảnh động ở trên, nhấp vào tay cầm ràng buộc, được hiển thị dưới dạng một vòng tròn ở cạnh của phần tử. Sau đó kéo tay cầm đến một tay cầm ràng buộc khác hoặc đến ranh giới của phần tử cha. Một đường zigzag sẽ đại diện cho ràng buộc đó.



- **Resizing handle:** Để thay đổi kích thước phần tử, hãy kéo các tay cầm vuông để thay đổi kích thước. Khi bạn kéo, tay cầm sẽ chuyển thành góc nghiêng.

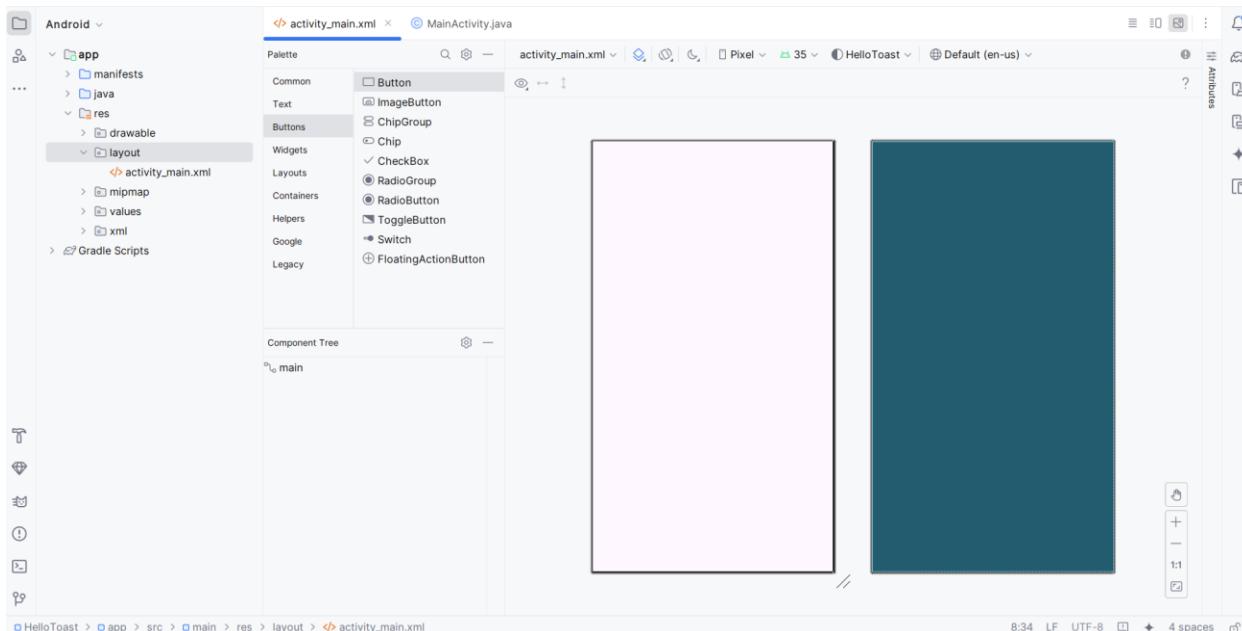


2.2 Thêm một Button vào bố cục

Khi được kích hoạt, công cụ **Autoconnect** sẽ tự động tạo hai hoặc nhiều ràng buộc cho một phần tử UI vào bố cục cha. Sau khi bạn kéo phần tử vào bố cục, nó sẽ tạo các ràng buộc dựa trên vị trí của phần tử.

Thực hiện theo các bước sau để thêm một **Button**:

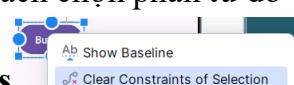
1. Bắt đầu với một bố cục trống. Phần tử **TextView** không cần thiết, vì vậy khi nó vẫn đang được chọn, nhấn phím **Delete** hoặc chọn **Edit > Delete**. Nay giờ bạn có một bố cục hoàn toàn trống.
2. Kéo một **Button** từ bảng **Palette** vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả **Button** ở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

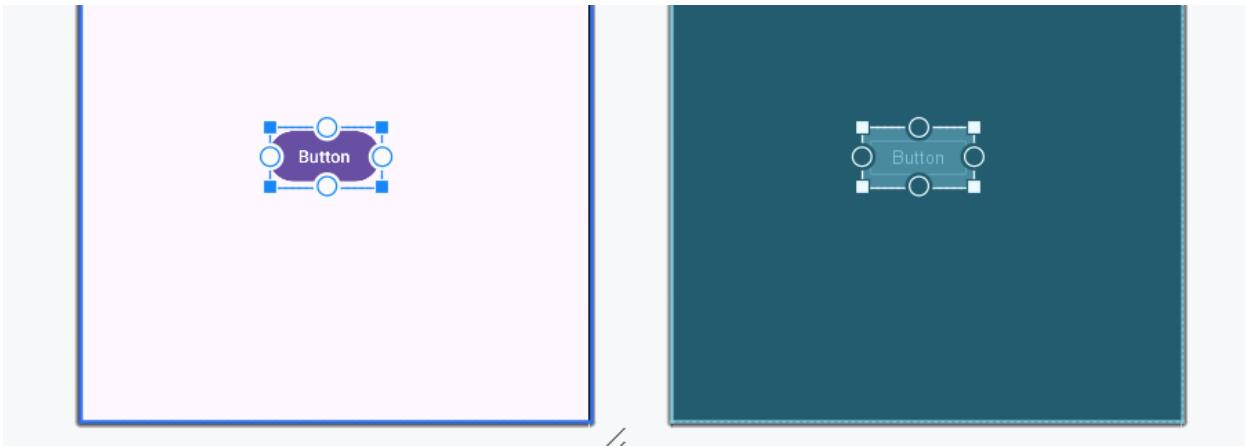


2.3 Thêm một Button thứ hai vào bố cục

1. Kéo một **Button** khác từ bảng **Palette** vào giữa bố cục như minh họa trong hình động dưới đây. **Autoconnect** có thể sẽ tự động cung cấp các ràng buộc ngang cho bạn (nếu không, bạn có thể tự kéo chúng).
2. Kéo một ràng buộc dọc đến phía dưới của bố cục (tham khảo hình dưới).

Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di chuyển

con trỏ chuột qua nó để hiển thị nút **Clear Constraints** . Nhấp vào nút này để xóa tất cả các ràng buộc trên phần tử được chọn. Để xóa một ràng buộc cụ thể, nhấp vào tay cầm cụ thể đặt ra ràng buộc đó. Để xóa tất cả các ràng buộc trong toàn bộ bố cục, nhấp vào công cụ **Clear All Constraints** trên thanh công cụ. Công cụ này rất hữu ích nếu bạn muốn làm lại tất cả các ràng buộc trong bố cục của mình.



Nhiệm vụ 3: **Thay đổi các thuộc tính của phần tử giao diện người dùng (UI)**

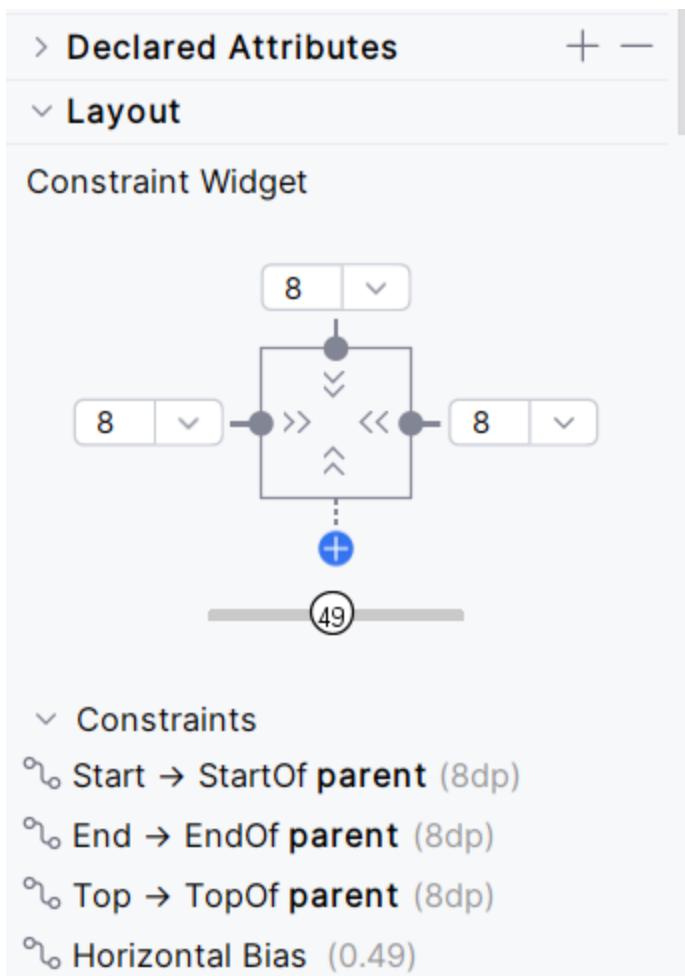
Bảng **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Bạn có thể tìm các thuộc tính (còn gọi là thuộc tính properties) chung cho tất cả các view trong tài liệu của lớp **View**.

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

3.1 Thay đổi kích thước của Button

Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước ở bốn góc của một View để bạn có thể nhanh chóng thay đổi kích thước View. Bạn có thể kéo các tay cầm ở mỗi góc của View để thay đổi kích thước, nhưng làm như vậy sẽ cố định kích thước chiều rộng và chiều cao. Tránh việc cố định kích thước cho hầu hết các phần tử View vì các kích thước cố định không thể thích ứng với các nội dung và kích thước màn hình khác nhau.

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

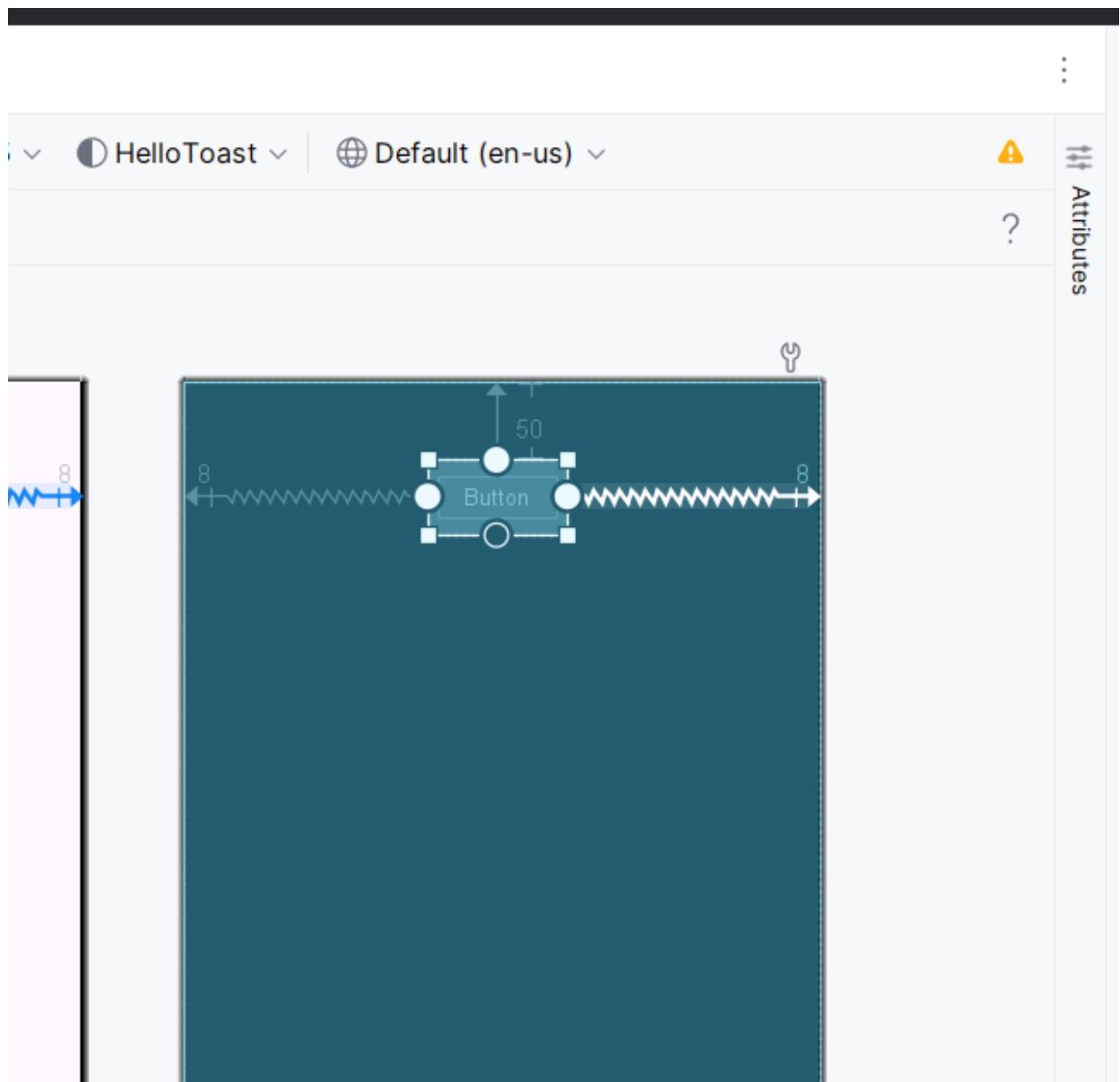


Trong hình trên:

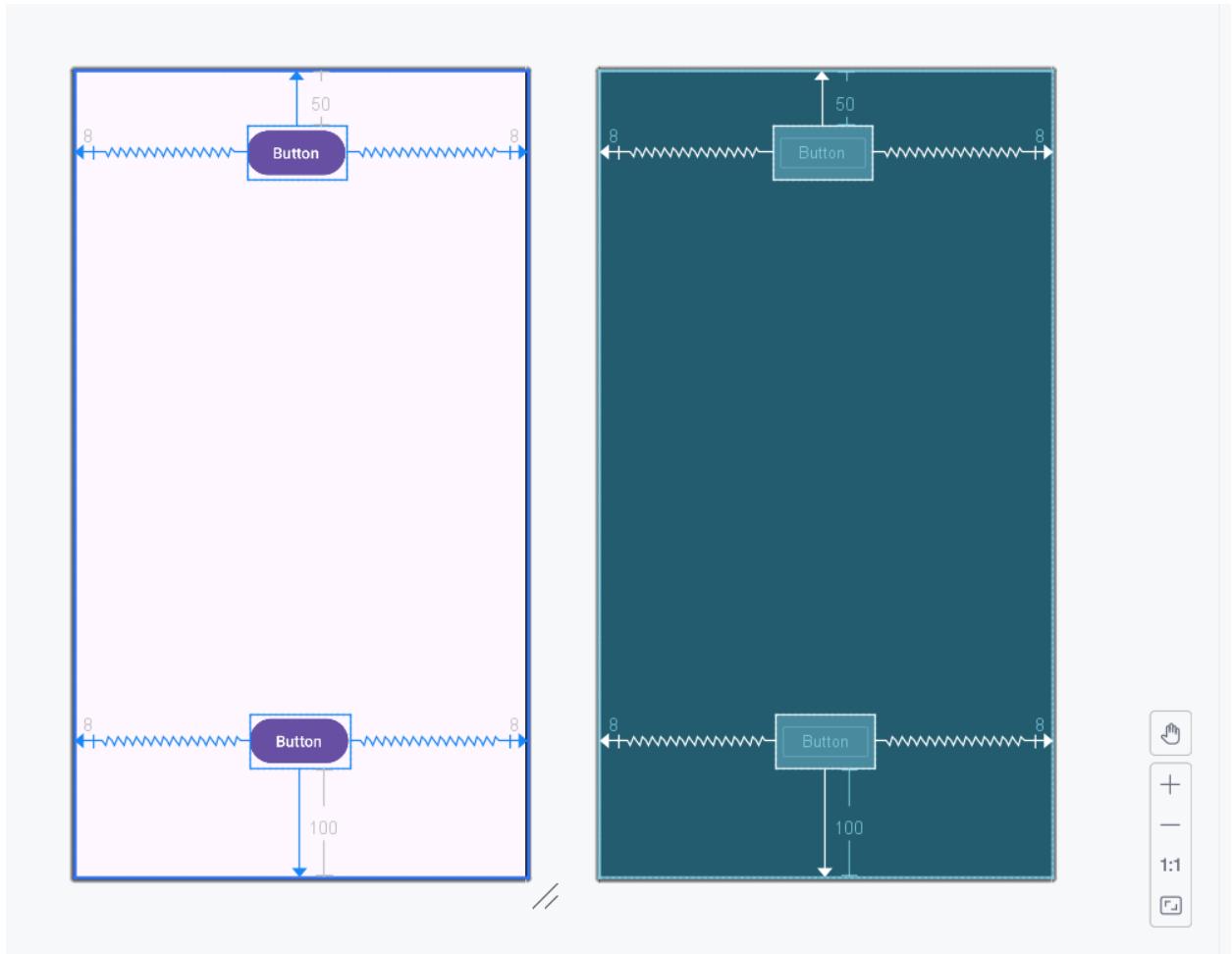
1. **Điều khiển chiều cao.** Điều khiển này chỉ định thuộc tính **layout_height** và xuất hiện ở hai đoạn trên và dưới của hình vuông. Các góc cho biết rằng điều khiển này được đặt thành **wrap_content**, nghĩa là View sẽ mở rộng theo chiều dọc tùy theo nhu cầu để phù hợp với nội dung của nó. Số "8" biểu thị một khoảng cách lề tiêu chuẩn được đặt là 8dp.
2. **Điều khiển chiều rộng.** Điều khiển này chỉ định thuộc tính **layout_width** và xuất hiện ở hai đoạn bên trái và bên phải của hình vuông. Các góc cho biết rằng điều khiển này được đặt thành **wrap_content**, nghĩa là View sẽ mở rộng theo chiều ngang tùy theo nhu cầu để phù hợp với nội dung của nó, với lề tối đa là 8dp.

Thực hiện theo các bước sau:

1. Chọn **Button** trên cùng trong bảng **Component Tree**.
2. Nhấp vào tab **Attributes** ở phía bên phải cửa sổ trình chỉnh sửa bộ cục.



3. Nhấp vào điều khiển chiều rộng hai lần — lần nhấp đầu tiên thay đổi nó thành **Fixed** với các đường thẳng, và lần nhấp thứ hai thay đổi nó thành **Match Constraints** với các cuộn lò xo, như đã hiển thị trong hình động dưới đây.



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

Như đã chỉ ra trong các bước trước, các thuộc tính **layout_width** và **layout_height** trong bảng **Attributes** thay đổi khi bạn thay đổi các điều khiển chiều cao và chiều rộng trong **inspector**. Các thuộc tính này có thể có một trong ba giá trị cho bố cục, là một **ConstraintLayout**:

- Cài đặt **match_constraint** mở rộng phần tử **View** để lấp đầy bố cục cha theo chiều rộng hoặc chiều cao — đến lề, nếu có đặt lề. **Bố cục cha** trong trường hợp này là **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.
- Cài đặt **wrap_content** thu nhỏ kích thước phần tử **View** sao cho đủ lớn để chứa nội dung của nó. Nếu không có nội dung, phần tử **View** sẽ trở nên vô hình.
- Để chỉ định một kích thước cố định có thể điều chỉnh theo kích thước màn hình

của thiết bị, hãy sử dụng một số pixel độc lập với mật độ (đơn vị dp). Ví dụ, **16dp** có nghĩa là 16 pixel độc lập với mật độ.

Mẹo: Nếu bạn thay đổi thuộc tính **layout_width** bằng cách sử dụng menu bật lên của nó, thuộc tính **layout_width** sẽ được đặt thành **zero** vì không có kích thước được đặt. Cài đặt này giống như **match_constraint**— phần tử có thể mở rộng tối đa có thể để đáp ứng các ràng buộc và cài đặt lè.

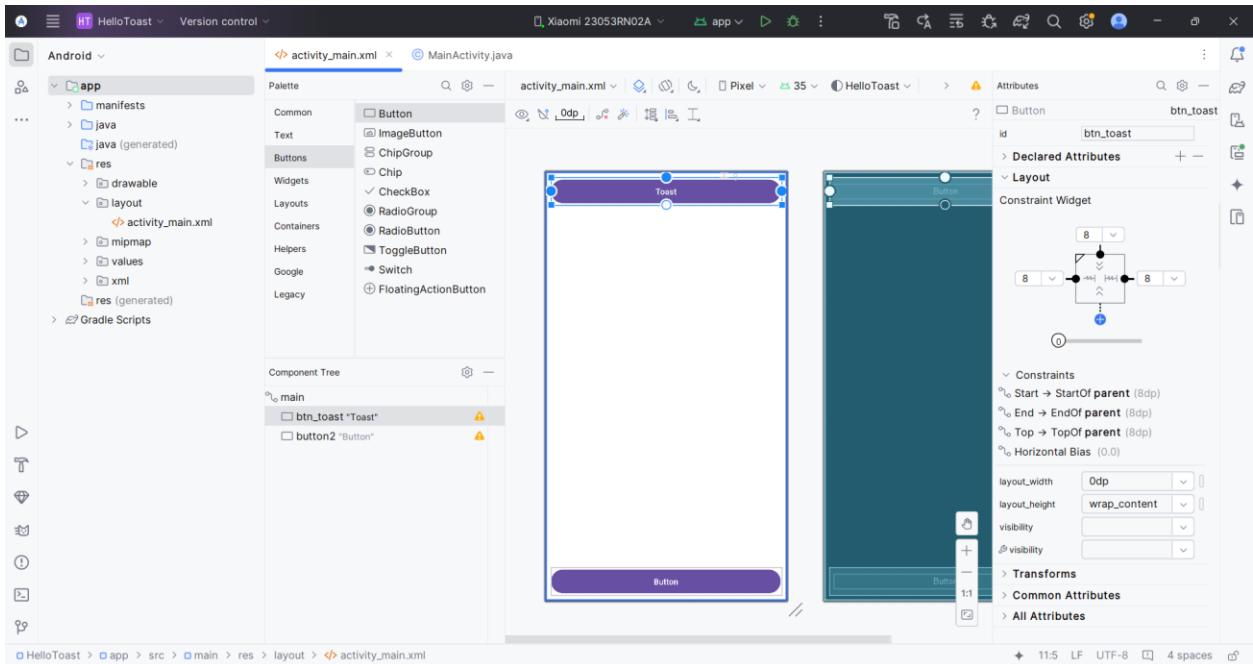
3.2 Thay đổi các thuộc tính của Button

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

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

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

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



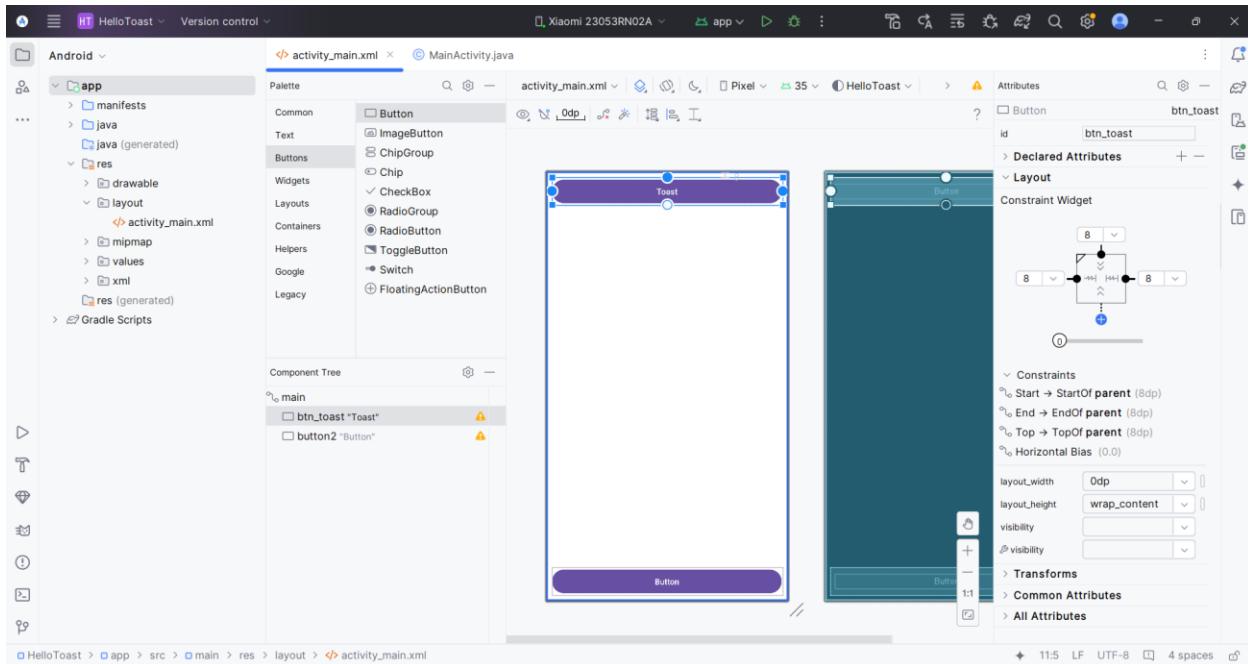
5. Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng **button_count** làm ID, **Count** cho thuộc tính **text**, và các màu tương tự cho nền và văn bản như các bước trước.
- colorPrimary** là màu chính của chủ đề, là một trong các màu cơ bản của chủ đề được xác định trong tệp tài nguyên **colors.xml**. Nó được sử dụng cho thanh ứng dụng. Sử dụng các màu cơ bản cho các phần tử giao diện người dùng khác giúp tạo ra một giao diện thống nhất. Bạn sẽ tìm hiểu thêm về chủ đề ứng dụng và **Material Design** trong bài học khác.

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

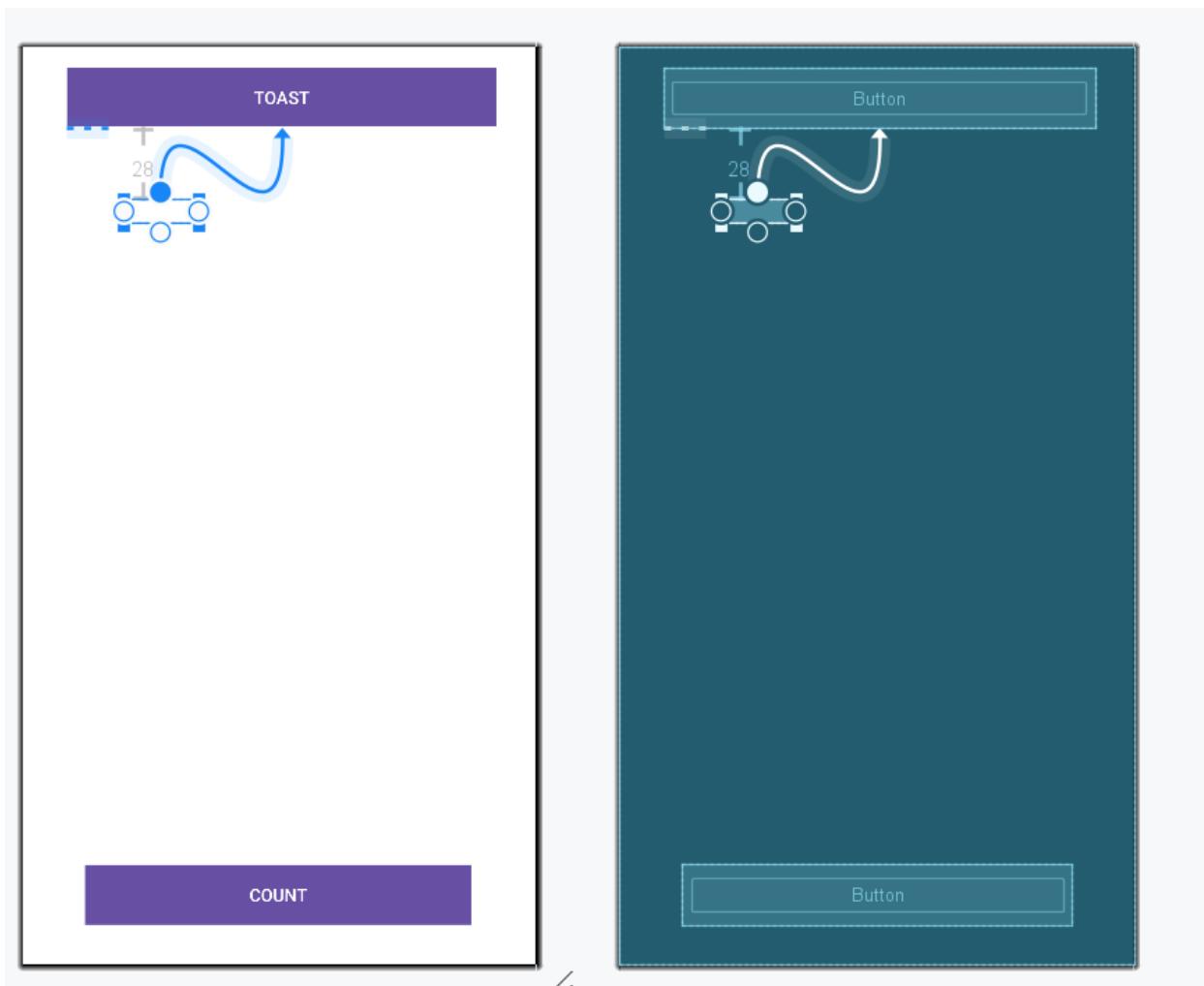
Một trong những lợi ích của **ConstraintLayout** là khả năng căn chỉnh hoặc ràng buộc các phần tử liên quan đến các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một **TextView** vào giữa bố cục và ràng buộc nó theo chiều ngang với các lề và theo chiều dọc với hai nút **Button**. Sau đó, bạn sẽ thay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

4.1 Thêm một TextView và các ràng buộc

- Như được chỉ ra trong hình động dưới đây, kéo một **TextView** từ bảng **Palette** đến phần trên của bố cục, và kéo một ràng buộc từ đỉnh của **TextView** đến tay cầm của nút **Toast Button**. Điều này sẽ ràng buộc **TextView** nằm bên dưới nút **Button**.



1. Như minh họa trong hình động dưới đây, kéo một ràng buộc từ phía dưới của **TextView** đến chốt ở phía trên của nút **Count**. Sau đó, kéo các ràng buộc từ hai bên của **TextView** đến hai cạnh của bố cục. Việc này ràng buộc **TextView** nằm giữa bố cục, ở giữa hai nút **Button**.



4.2 Đặt các thuộc tính cho TextView

Với **TextView** được chọn, hãy mở bảng thuộc tính (**Attributes pane**) nếu nó chưa mở. Đặt các thuộc tính cho **TextView** như hướng dẫn dưới đây. Những thuộc tính mới sẽ được giải thích sau:

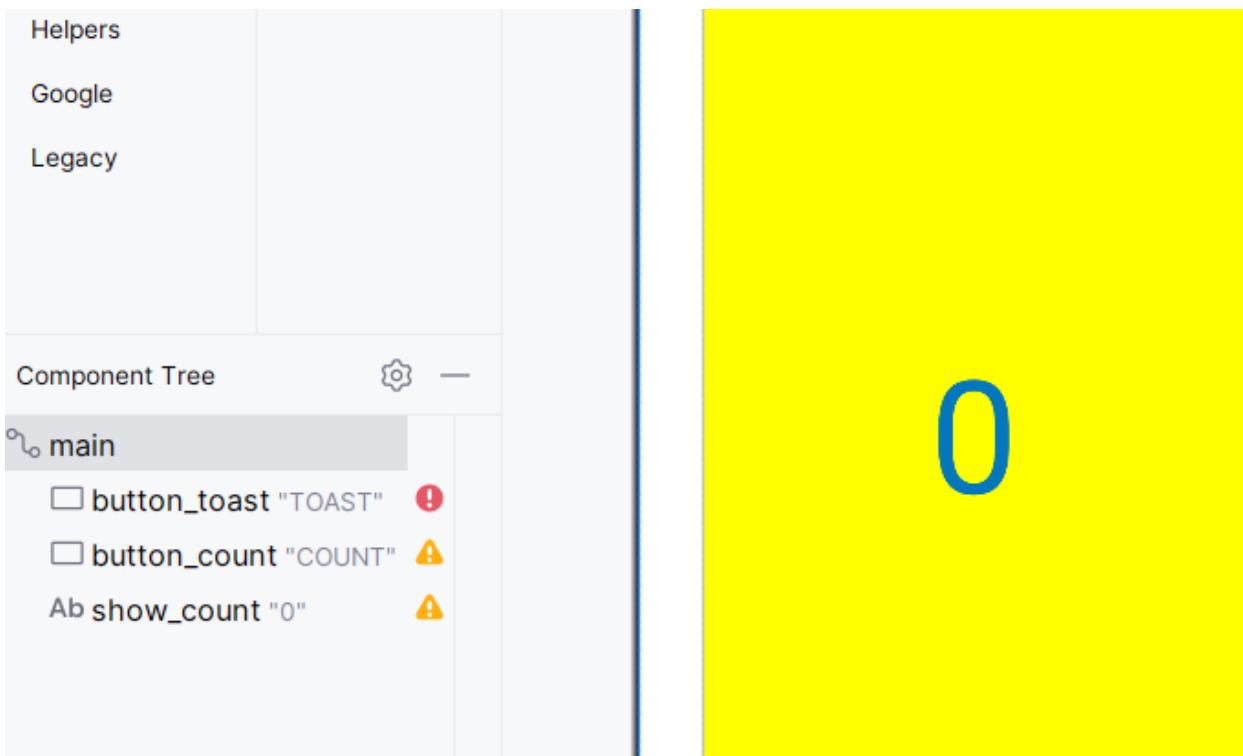
1. Đặt **ID** thành `show_count`.
2. Đặt **text** thành `0`.
3. Đặt **textSize** thành `160sp`.
4. Đặt **textStyle** thành **B** (in đậm) và **textAlignment** thành **ALIGNCENTER** (căn giữa đoạn văn bản).
5. Thay đổi chế độ kích thước ngang và dọc (**layout_width** và **layout_height**) thành **match_constraint**.

6. Đặt **textColor** thành @color/colorPrimary.
 7. Cuộn xuống bảng thuộc tính và nhấp vào "View all attributes", cuộn đến trang thứ hai của các thuộc tính, tìm thuộc tính "background" và nhập giá trị #FFF00 để chọn một sắc thái màu vàng.
 8. Cuộn xuống thuộc tính "gravity", mở rộng mục "gravity" và chọn "center_ver" (để căn giữa theo chiều dọc).
- **textSize**: Kích thước văn bản của TextView. Trong bài học này, kích thước được đặt là **160sp**. sp là viết tắt của **scale-independent pixel** (pixel không phụ thuộc tỷ lệ) và giống như **dp**, là đơn vị điều chỉnh theo mật độ màn hình và tùy chọn kích thước phông chữ của người dùng. Hãy sử dụng đơn vị **dp** khi bạn xác định kích thước phông chữ để đảm bảo 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 là **B (bold)** (in đậm) trong bài học này. Căn chỉnh văn bản, được đặt là **ALIGNCENTER** (căn giữa đoạn văn bản).
 - **gravity**: Thuộc tính *gravity* xác định cách một *View* được căn chỉnh trong *View* hoặc *ViewGroup* cha của nó. Trong bước này, bạn căn chỉnh *TextView* theo chiều dọc ở giữa *ConstraintLayout* cha.

Bạn có thể nhận thấy rằng thuộc tính *background* nằm ở trang đầu tiên của bảng *Attributes* (Thuộc tính) đối với một *Button*, nhưng lại nằm ở trang thứ hai đối với một *TextView*. Bảng *Attributes* thay đổi tùy thuộc vào từng loại *View*: các thuộc tính phổ biến nhất của loại *View* sẽ xuất hiện trên trang đầu tiên, và các thuộc tính khác được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của bảng *Attributes*, nhấn vào biểu tượng trên thanh công cụ ở phía trên cùng của bảng.

Task 5: Chính sửa bộ cục trong XML

Bộ cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, bên cạnh mỗi phần tử giao diện người dùng trong bảng Component Tree lại xuất hiện một dấu chấm than. Di chuyển con trỏ chuột qua các dấu chấm than này để xem thông báo cảnh báo, như hình minh họa bên dưới. Cùng một cảnh báo xuất hiện cho cả ba phần tử: chuỗi được mã hóa cứng nên được thay thế bằng tài nguyên.



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

5.1 Mở mã XML của bố cục

Trong nhiệm vụ này, hãy mở tệp activity_main.xml nếu nó chưa được mở, và nhấp vào tab Text ở dưới cùng của trình chỉnh sửa bố cục.

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

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

```
tools:context=".MainActivity" >

<Button
    android:id="@+id/btnToast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:background="#3F4BA7"
    android:text="@string/button_label_toast"
    android:textColor="@color/white"
    android:textSize="20dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textToast"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="#FFEB3B"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAllCaps="true"
    android:textColor="#3F51B5"
    android:textSize="200dp"
    app:layout_constraintBottom_toTopOf="@+id/btnCount"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/btnToast"
    app:layout_constraintVertical_bias="0.0" />

<Button
```

```
    android:id="@+id/btnCount"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="#3F51B5"
    android:text="@string/button_label_count"
    android:textColor="@color/white"
    android:textSize="20dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

5.2 Tách các chuỗi thành tài nguyên

Thay vì mã hóa cứng các chuỗi, việc sử dụng tài nguyên chuỗi là một thực hành tốt, vì các chuỗi được lưu trữ trong một tệp riêng giúp quản lý dễ dàng hơn, đặc biệt khi bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và địa phương hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho mỗi ngôn ngữ.

Các bước thực hiện:

1. Nhập một lần vào từ "Toast" (chuỗi cảnh báo đầu tiên được đánh dấu).
2. Nhấn Alt-Enter trên Windows hoặc Option-Enter trên macOS và chọn Extract string resource từ menu bật lên.
3. Nhập "button_label_toast" cho tên tài nguyên.
4. Nhấp OK. Một tài nguyên chuỗi sẽ được tạo trong tệp values/res/values.xml, và chuỗi trong mã của bạn sẽ được thay thế bằng tham chiếu tới tài nguyên: @string/button_label_toast
5. Tách các chuỗi còn lại: sử dụng "button_label_count" cho "Count" và "count_initial_value" cho "0".
6. Trong ngăn Project > Android, mở rộng mục values trong thư mục res, sau đó nhấp đúp vào tệp strings.xml để xem các tài nguyên chuỗi của bạn trong tệp strings.xml.

```

<resources>
    <string name="app_name">HelloToast1</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
</resources>

```

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

```

<resources>
    <string name="app_name">HelloToast1</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
    A
</resources>

```

Mẹo: Các tài nguyên chuỗi bao gồm tên ứng dụng, xuất hiện trên thanh ứng dụng ở đầu màn hình nếu bạn khởi tạo dự án bằng Mẫu Trống. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa tài nguyên app_name.

Nhiệm vụ 6: Thêm xử lý sự kiện onClick cho các nút

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

6.1 Thêm thuộc tính onClick và phương thức xử lý cho mỗi nút

Một click handler là phương thức được gọi khi người dùng nhấn hoặc chạm vào một phần tử UI có thể nhấn được. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick ở ngăn Attributes của tab Design. Bạn cũng có thể chỉ định tên của phương thức xử lý trong trình chỉnh sửa XML bằng cách thêm thuộc tính android:onClick vào nút. Ở đây, bạn sẽ sử dụng phương pháp thứ hai vì bạn chưa tạo các phương thức xử lý, và trình chỉnh sửa XML cung cấp cách tự động tạo các phương thức đó.

- Với trình chỉnh sửa XML (tab Text) đang mở, tìm nút có android:id được đặt là button_toast.

```

<Button
    android:id="@+id/button_toast"
    android:layout_width="345dp"
    android:layout_height="47dp"
    android:background="#1565C0"
    android:backgroundTint="#1565C0"
    android:gravity="center"
    android:text="TOAST"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.454"
    app:layout_constraintStart_toStartOf="parent"/>

```

- ```
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintVertical_bias="0.036"
/>
2. Thêm thuộc tính android:onClick vào cuối phần tử button_toast, sau thuộc
tính cuối cùng và trước dấu kết thúc ">"
```
- ```
    android:onClick="showToast"/>
```
3. Nhập vào biểu tượng bóng đèn đỏ xuất hiện bên cạnh thuộc tính. Chọn Create click handler, chọn MainActivity, và nhấp OK. Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy nhập vào tên phương thức ("showToast"). Nhấn Alt-Enter (trên Windows/Linux) hoặc Option-Enter (trên Mac), chọn Create 'showToast(view)' in MainActivity, và nhấp OK. Thao tác này sẽ tạo ra một phương thức mẫu (placeholder method stub) cho phương thức showToast() trong MainActivity, như được hiển thị ở cuối các bước này.
 4. Lặp lại hai bước cuối với nút button_count: Thêm thuộc tính android:onClick vào cuối phần tử, và tạo click handler.

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

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

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

    <Button
        android:id="@+id/btnToast"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:background="#3F4BA7" />
```

```
        android:text="@string/button_label_toast"
        android:textColor="@color/white"
        android:textSize="20dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textToast"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="#FFEB3B"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAllCaps="true"
    android:textColor="#3F51B5"
    android:textSize="200dp"
    app:layout_constraintBottom_toTopOf="@+id(btnCount)"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id	btnToast"
    app:layout_constraintVertical_bias="0.0" />

<Button
    android:id="@+id	btnCount"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="#3F51B5"
    android:text="@string/button_label_count"
    android:textColor="@color/white"
    android:textSize="20dp"
```

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

5. Nếu MainActivity.java chưa được mở, mở rộng mục java trong ngăn Project > Android, mở rộng com.example.android.hellotoast, sau đó nhấp đúp vào MainActivity. Trình soạn thảo mã sẽ xuất hiện với mã của MainActivity.

```
package com.example.hellotoast1;

> import ...

`<> public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            actionBar.setBackgroundDrawable(new ColorDrawable(Color.BLUE));
        }
        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 countUp(View view) {
    }
}
```

6.2 Chính sửa xử lý sự kiện của nút Toast

Bây giờ, bạn sẽ chỉnh sửa phương thức showToast() – xử lý sự kiện click của nút Toast trong MainActivity – để hiển thị một thông báo. Một Toast cung cấp cách hiển thị thông báo đơn giản trong một cửa sổ popup nhỏ. Toast chỉ chiếm

không cần thiết để hiển thị thông báo, và Activity hiện tại vẫn hiển thị và có thể tương tác. Toast có thể hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn – hãy thêm một thông báo Toast để hiển thị kết quả của thao tác nhấn nút hoặc thực hiện một hành động.

Thực hiện theo các bước sau để chỉnh sửa xử lý sự kiện nút Toast:

1. Tìm vị trí của phương thức showToast() mới được tạo trong MainActivity.

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

2. Để tạo một đối tượng Toast, hãy gọi phương thức makeText (factory method) trên lớp Toast.

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

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

3. Cung cấp context của Activity của ứng dụng. Vì một Toast được hiển thị phía trên giao diện người dùng của Activity, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong context của Activity mà bạn cần, hãy sử dụng từ khóa "this" như một cách rút gọn.

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

4. Cung cấp thông báo để hiển thị, chẳng hạn như một tài nguyên chuỗi (ví dụ: toast_message mà bạn đã tạo ở bước trước). Tài nguyên chuỗi toast_message được nhận diện bằng R.string.toast_message.

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

5. Cung cấp thời lượng hiển thị. Ví dụ, Toast.LENGTH_SHORT hiển thị Toast trong một khoảng thời gian tương đối ngắn.

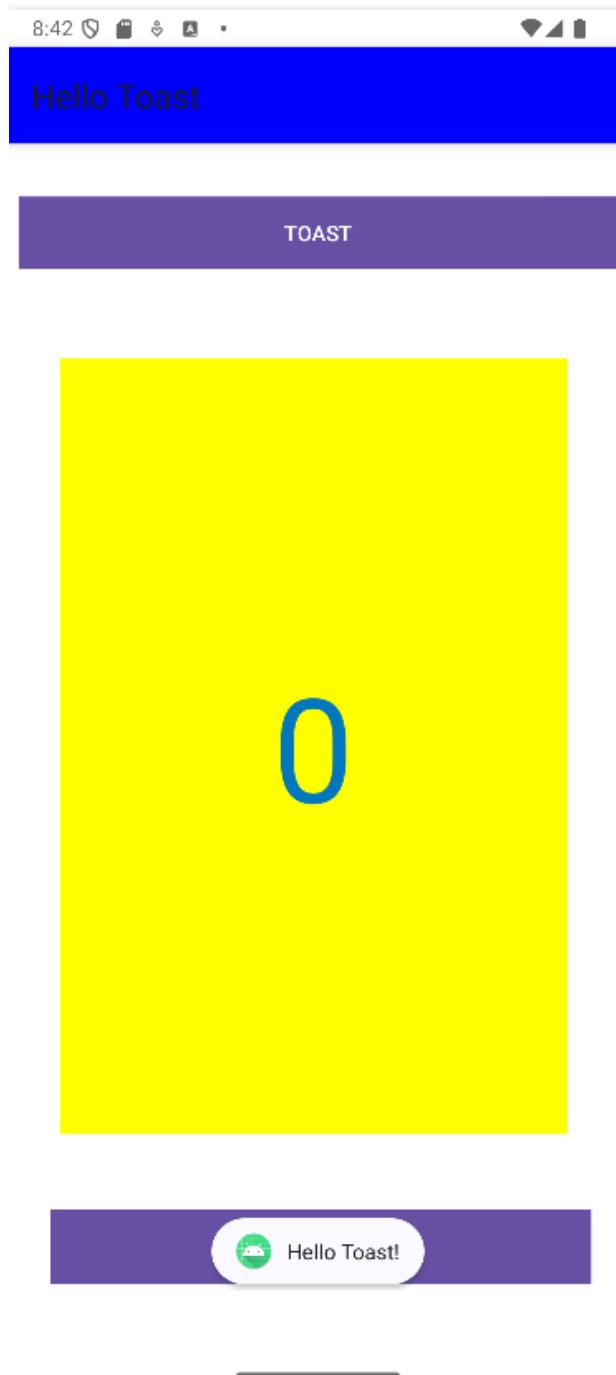
```
Toast toast = Toast.makeText(this,  
R.string.toast_message,Toast.LENGTH_SHORT);
```

Thời lượng hiển thị của Toast có thể là Toast.LENGTH_LONG hoặc Toast.LENGTH_SHORT. Thời gian thực tế là khoảng 3,5 giây cho Toast dài và 2 giây cho Toast ngắn.

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

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

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



6.3 Chính sửa xử lý sự kiện của nút Count

Bây giờ, bạn sẽ chỉnh sửa phương thức countUp() – xử lý sự kiện click của nút Count trong MainActivity – sao cho nó hiển thị giá trị đếm hiện tại sau mỗi lần nhấn nút Count. Mỗi lần nhấn nút sẽ tăng giá trị đếm lên một đơn vị.

Mã xử lý sự kiện cần thực hiện các điều sau:

- Theo dõi giá trị đếm khi nó thay đổi.

- Gửi giá trị đếm được cập nhật đến TextView để hiển thị.

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

1. Tìm vị trí của phương thức countUp() mới được tạo trong MainActivity, có dạng:

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

2. Để theo dõi giá trị đếm, bạn cần một biến thành viên riêng (private member variable). Mỗi lần nhấn nút Count sẽ tăng giá trị của biến này. Nhập đoạn mã sau (đoạn mã này sẽ được đánh dấu màu đỏ và xuất hiện biểu tượng bóng đèn đỏ):

```
public void countUp(View view) {  
    mCount++;  
}
```

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

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn Create field 'mCount' từ menu bật lên. Thao tác này tạo ra một biến thành viên riêng tại đầu lớp MainActivity, và Android Studio mặc định kiểu của biến này là integer (int):
4. Thay đổi câu lệnh khai báo biến thành viên thành khởi tạo giá trị bằng 0:

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

5. Ngoài biến mCount, bạn cũng cần một biến thành viên riêng để lưu tham chiếu đến TextView hiển thị số đếm, gọi biến này là mShowCount:

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

.....}

6. Bây giờ, đã có mShowCount, bạn có thể lấy tham chiếu đến TextView thông qua ID đã đặt trong tệp bố cục. Để lấy tham chiếu này chỉ một lần, hãy thực hiện trong phương thức onCreate(). Như bạn đã học trong một bài học khác, phương thức onCreate() được dùng để inflate bố cục, tức là đặt content view của màn hình bằng tệp XML bố cục. Bạn cũng có thể sử dụng onCreate() để lấy tham chiếu đến các phần tử UI khác trong bố cục, chẳng hạn như TextView. Tìm phương thức onCreate() trong MainActivity:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable(this);  
    ActionBar actionBar = getSupportActionBar();  
    if (actionBar != null) {  
        actionBar.setBackgroundDrawable(new ColorDrawable(Color.BLUE));  
    }  
    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;  
    });  
}
```

7. Thêm câu lệnh findViewById vào cuối phương thức onCreate() để gán giá trị cho mShowCount:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable(this);  
    ActionBar actionBar = getSupportActionBar();  
    if (actionBar != null) {  
        actionBar.setBackgroundDrawable(new ColorDrawable(Color.BLUE));  
    }  
    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;  
    });  
    mShowCount = (TextView) findViewById(R.id.show_count);  
}
```

Một View, giống như một chuỗi, là một tài nguyên có thể có một ID. Gọi `findViewById` với ID của một view sẽ trả về đối tượng View. Vì phương thức này trả về một View, bạn cần ép kiểu kết quả về loại view mà bạn mong đợi, trong trường hợp này là (`TextView`).

8. Sau khi đã gán tham chiếu cho `mShowCount`, bạn có thể sử dụng biến này để cập nhật văn bản của `TextView` với giá trị của biến `mCount`. Thêm đoạn mã sau vào phương thức `countUp()`:

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

Toàn bộ phương thức `countUp()` bây giờ trông như sau:

```
public void countUp(View view) {
    mCount++;
    if (mShowCount != null)
        mShowCount.setText(Integer.toString(mCount));
}
```

9. Chạy ứng dụng để xác minh rằng giá trị đếm tăng lên mỗi khi bạn nhấn nút Count.

8:37 5G 🔋 ⚡ 🔋



Hello Toast

TOAST

1

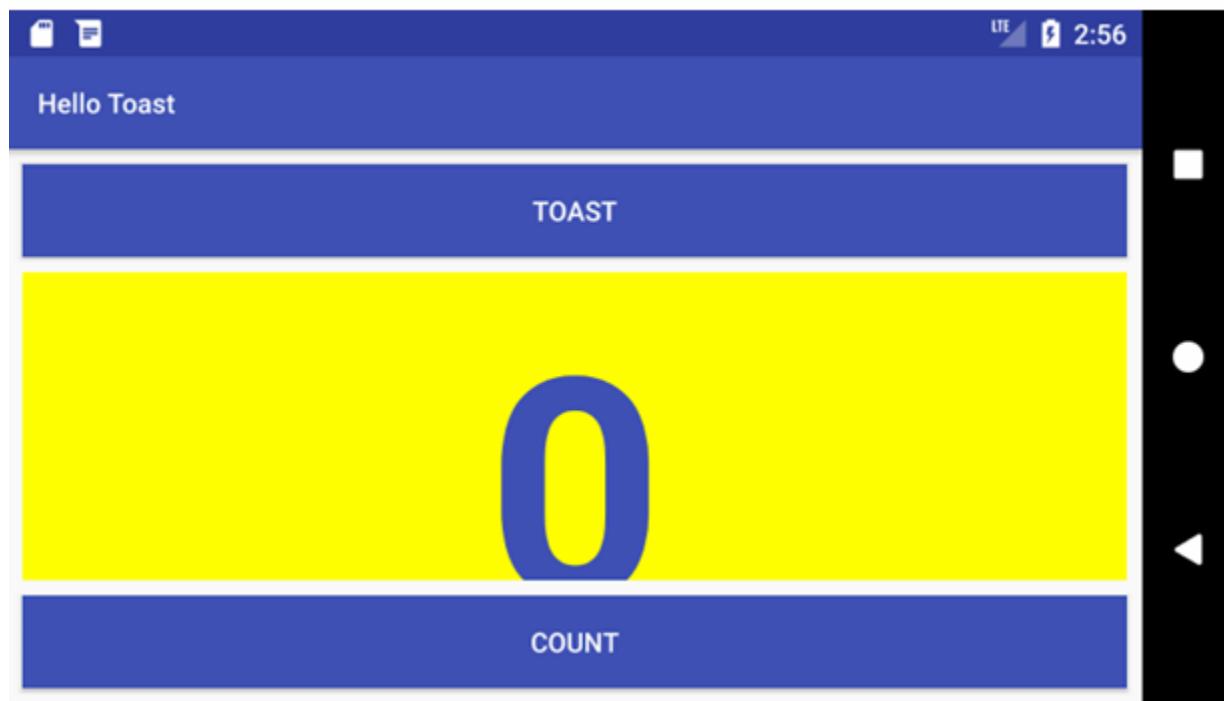
COUNT

Mẹo: Để tìm hiểu chi tiết về cách sử dụng ConstraintLayout, hãy xem Codelab "Using ConstraintLayout to design your views".

Thách thức lập trình

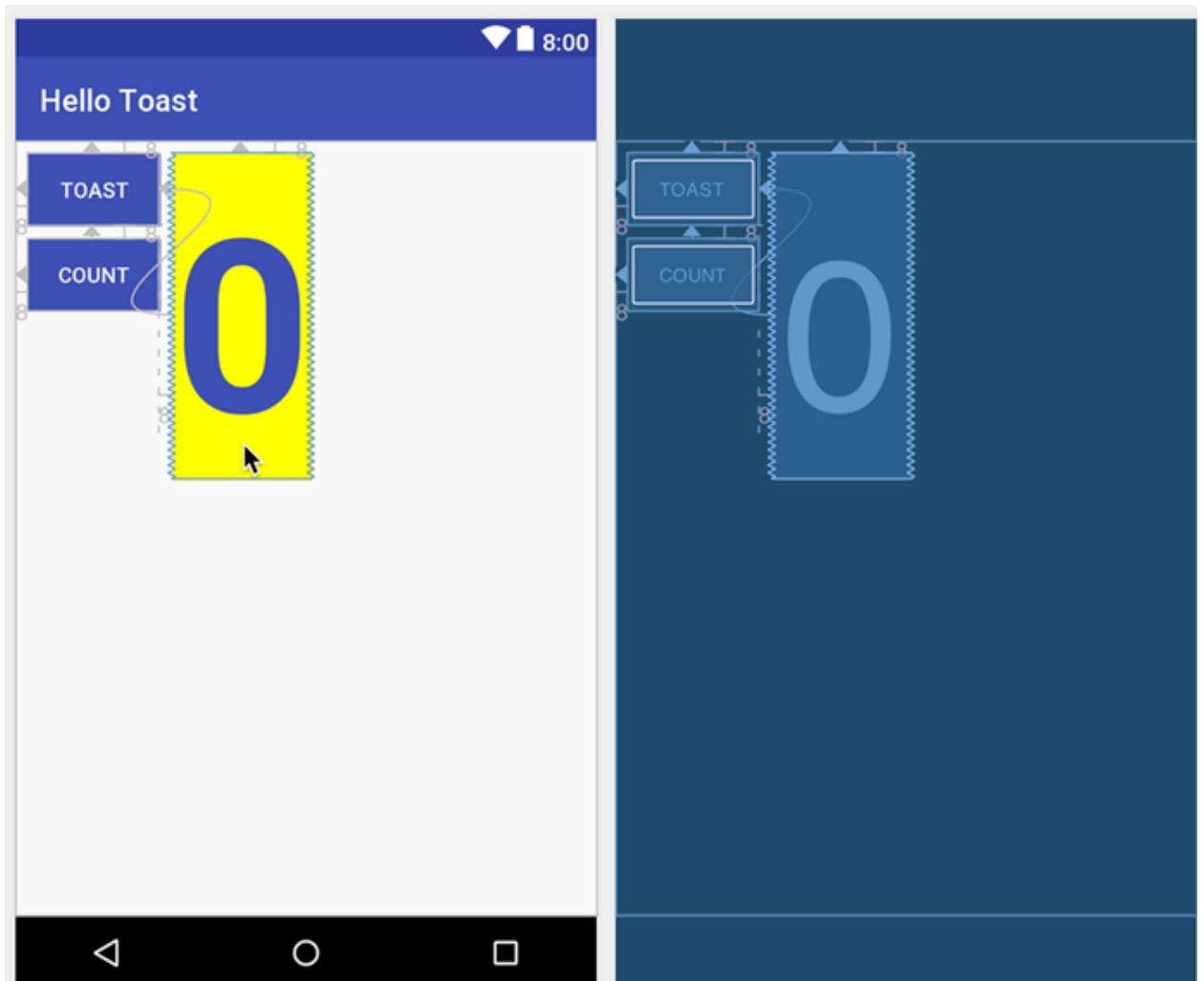
Lưu ý: Tất cả các thách thức lập trình đều không bắt buộc và không phải là điều kiện tiên quyết cho các bài học sau.

Ứng dụng HelloToast trông ổn khi thiết bị hoặc trình giả lập được đặt theo chiều dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang chiều ngang, nút **Count** có thể chồng lên **TextView** ở phía dưới như trong hình minh họa bên dưới.



Thử thách: Thay đổi bố cục để nó trông đẹp cả khi ở chế độ ngang và dọc:

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





Tóm tắt:

View, ViewGroup và layouts:

- Tất cả các phần tử giao diện người dùng (UI) đều là các lớp con của lớp **View**, do đó kế thừa nhiều thuộc tính từ lớp **View**.
- Các phần tử **View** có thể được nhóm lại bên trong một **ViewGroup**, hoạt động như một container. Mỗi quan hệ này là mối quan hệ cha-con, trong đó **parent** là một **ViewGroup**, còn **child** là một **View** hoặc một **ViewGroup** khác.
- Phương thức **onCreate()** được sử dụng để **inflate layout**, nghĩa là thiết lập nội dung hiển thị của màn hình từ tệp XML layout. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện người dùng (UI) khác trong layout.
- Một **View**, giống như một chuỗi ký tự (string), là một tài nguyên có thể có ID. Phương thức **findViewById** nhận ID của một View làm tham số và trả về View đó.

Sử dụng trình chỉnh sửa bố cục:

- Nhấp vào tab **Design** để thao tác các phần tử và bố cục, hoặc tab **Text** để chỉnh sửa mã XML của bố cục.

- Trong tab **Design**, bảng **Palettes** hiển thị các phần tử giao diện người dùng (UI) có thể sử dụng trong bố cục ứng dụng, và bảng **Component tree** hiển thị cây phân cấp của các phần tử UI.
- Các phần tử UI trong bố cục được hiển thị trong các bảng **design** và **blueprint**. Tab **Attributes** hiển thị bảng **Attributes** để cài đặt các thuộc tính cho phần tử UI.
- Constraint handle: Nhấp vào một constraint handle (vòng tròn ở mỗi cạnh của phần tử), kéo đến một constraint khác hoặc đường viền cha để tạo ràng buộc. Ràng buộc được hiển thị bằng một đường gấp khúc.
- Resizing handle: Kéo tay nắm (hình vuông) để thay đổi kích thước phần tử. Trong khi kéo, tay nắm sẽ chuyển thành góc xiên.
- Autoconnect tool: Khi được bật, công cụ này tự động tạo hai hoặc nhiều ràng buộc cho phần tử UI với bố cục cha, dựa trên vị trí của phần tử.
- Bạn có thể xóa ràng buộc của một phần tử bằng cách chọn phần tử, di chuột qua để hiển thị nút **Clear Constraints**, và nhấp để xóa tất cả các ràng buộc. Để xóa một ràng buộc cụ thể, nhấp vào constraint handle tương ứng.
- Bảng **Attributes** cung cấp truy cập vào tất cả các thuộc tính XML có thể gán cho một phần tử UI. Nó cũng bao gồm một bảng định cỡ hình vuông gọi là **view inspector** ở trên cùng. Các biểu tượng trong hình vuông đại diện cho cài đặt chiều cao và chiều rộng.

Thuộc tính **layout_width** và **layout_height** thay đổi khi bạn thay đổi chiều cao và chiều rộng trong bảng điều khiển. Các giá trị có thể là:

1. **match_constraint**: Phần tử mở rộng để lấp đầy không gian bố cục cha (có tính đến lề nếu được đặt).
2. **wrap_content**: Phần tử co lại vừa với nội dung của nó. Nếu không có nội dung, phần tử sẽ trở nên vô hình.
3. **dp cố định**: Chỉ định kích thước cố định, phù hợp với kích thước màn hình của thiết bị.

Trích xuất tài nguyên chuỗi (String Resources):

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

1. Nhấp vào chuỗi mã hóa cứng để trích xuất, nhấn **Alt-Enter** (hoặc **Option-Enter** trên Mac) và chọn **Extract string resources** từ menu bật lên.
2. Đặt tên cho **Resource name**.

3. Nhấn **OK**. Điều này sẽ tạo một tài nguyên chuỗi trong tệp `values/res/string.xml`, và chuỗi trong mã sẽ được thay thế bằng tham chiếu tới tài nguyên: `@string/button_label_toast`.

Xử lý sự kiện nhấp (Handling Clicks):

- **Click handler** là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI.
- Để chỉ định một click handler cho một phần tử UI như **Button**, nhập tên phương thức trong trường **onClick** của bảng **Attributes** ở tab **Design**, hoặc trong trình chỉnh sửa XML bằng cách thêm thuộc tính `android:onClick` vào phần tử **Button**.
- Tạo click handler trong Activity chính bằng cách sử dụng tham số **View**. Ví dụ:

```
public void showToast(View view) {  
    // Xử lý khi nhấp  
}
```

- Có thể tìm thấy thông tin về tất cả các thuộc tính của **Button** trong tài liệu lớp **Button**, và tất cả các thuộc tính của **TextView** trong tài liệu lớp **TextView**.

Toast cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ popup nhỏ. Nó chỉ chiếm không gian đủ để chứa thông báo. Để tạo một instance của Toast, thực hiện các bước sau:

1. Gọi phương thức **makeText()** từ lớp **Toast**.
2. Cung cấp **context** của **Activity** ứng dụng và thông báo cần hiển thị (chẳng hạn như một tài nguyên chuỗi).
3. Cung cấp thời lượng hiển thị, ví dụ: **Toast.LENGTH_SHORT** cho thời gian ngắn. Thời lượng có thể là **Toast.LENGTH_LONG** hoặc **Toast.LENGTH_SHORT**.
4. Hiển thị Toast bằng cách gọi phương thức **show()**.

Bài học 1.2 Phần B: Trình chỉnh sửa bố cục

Giới thiệu

Như đã học trong **1.2 Phần A: Giao diện tương tác đầu tiên**, bạn có thể xây dựng giao diện người dùng (UI) bằng **ConstraintLayout** trong trình chỉnh sửa bố cục. ConstraintLayout sắp xếp các phần tử UI bằng cách kết nối ràng buộc với các phần tử khác hoặc với các cạnh của bố cục. ConstraintLayout được thiết kế để dễ dàng kéo thả các phần tử UI vào trình chỉnh sửa bố cục.

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

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

- **LinearLayout**: Một nhóm căn chỉnh các phần tử **View** con bên trong theo chiều ngang hoặc dọc.
- **RelativeLayout**: Một nhóm các phần tử **View** con, trong đó mỗi phần tử **View** được định vị và căn chỉnh tương đối với các phần tử **View** khác bên trong **ViewGroup**. Vị trí của các phần tử **View** con được mô tả dựa trên mối quan hệ giữa chúng với nhau hoặc với **ViewGroup** cha.

Những gì bạn cần biết trước:

Bạn cần có khả năng:

- Tạo một ứng dụng **Hello World** với Android Studio.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị thực.
- Tạo một bố cục đơn giản cho ứng dụng bằng **ConstraintLayout**.
- Trích xuất và sử dụng tài nguyên chuỗi (string resources).

Những gì bạn sẽ học:

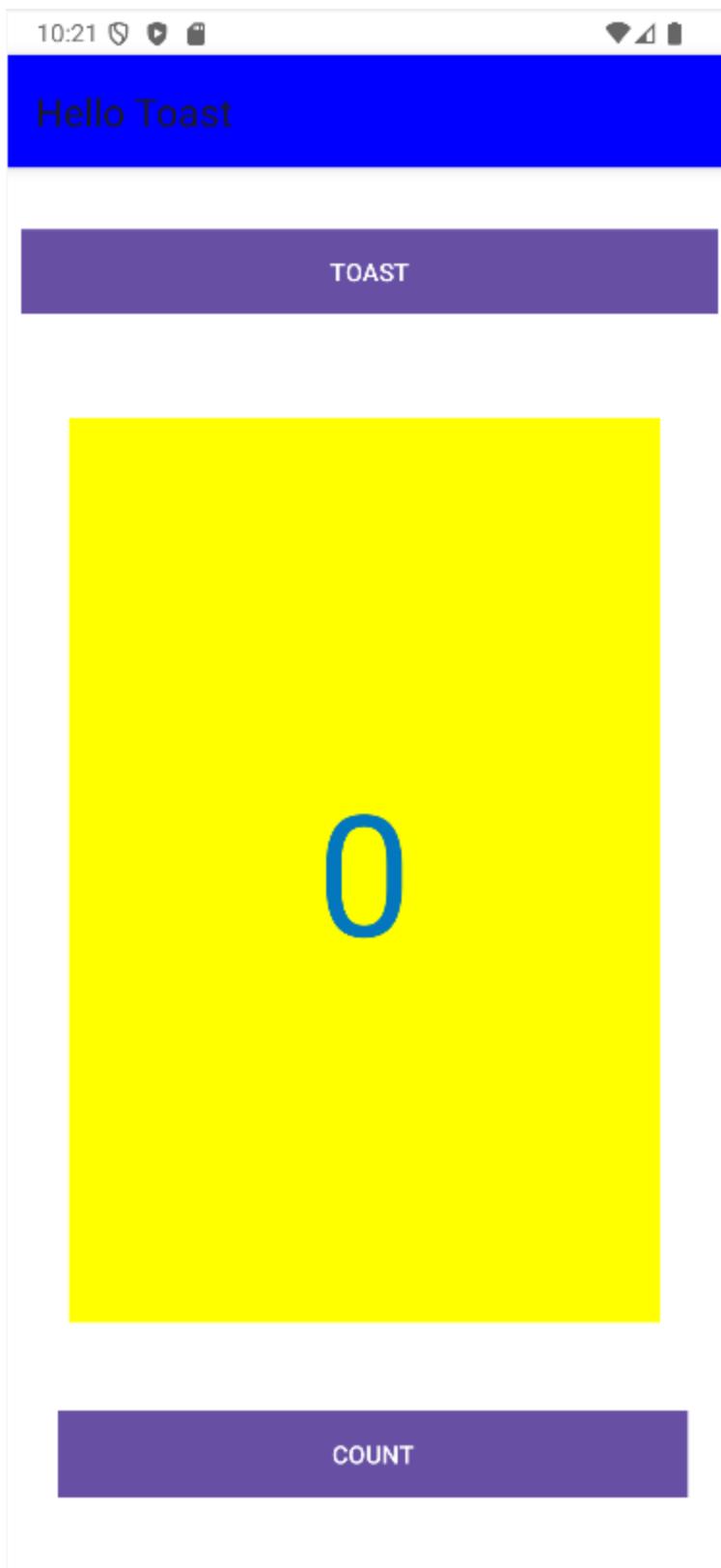
- Cách tạo một biến thể bố cục cho màn hình ngang (landscape).
- Cách tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Cách sử dụng ràng buộc đường cơ sở (baseline constraint) để căn chỉnh các phần tử UI với văn bản.
- Cách sử dụng các nút gói (pack) và căn chỉnh (align) để căn chỉnh các phần tử trong bố cục.
- Cách định vị các view trong **LinearLayout**.
- Cách định vị các view trong **RelativeLayout**.

Những gì bạn sẽ làm:

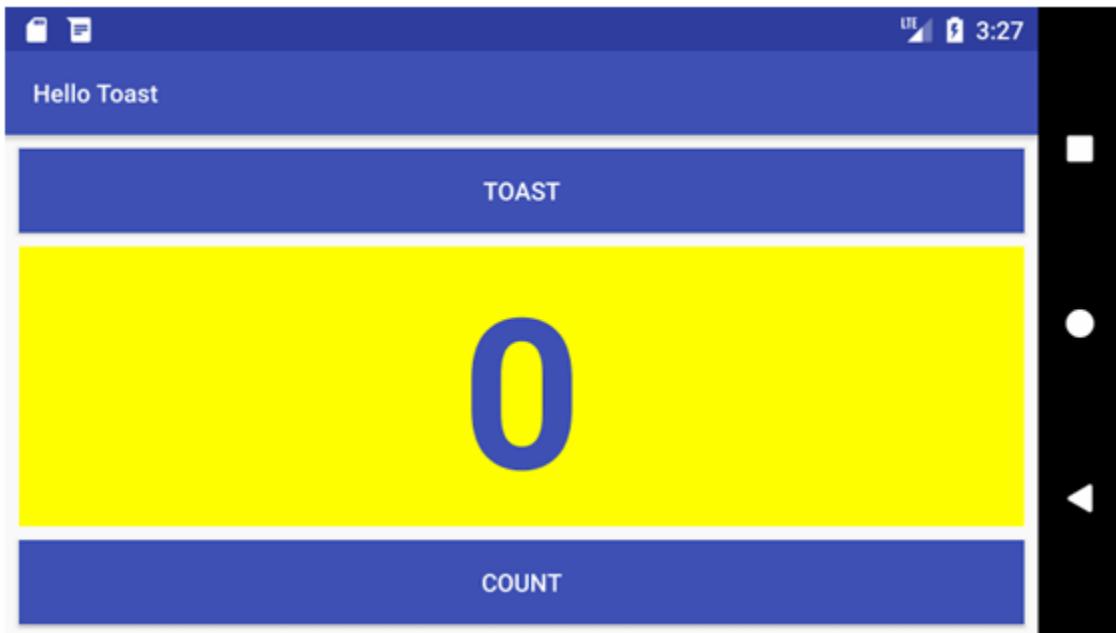
- Tạo một biến thể bố cục cho màn hình ngang.
- Tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Sửa đổi bố cục để thêm các ràng buộc vào các phần tử UI.
- Sử dụng ràng buộc đường cơ sở của **ConstraintLayout** để căn chỉnh các phần tử với văn bản.
- Sử dụng các nút gói và căn chỉnh của **ConstraintLayout** để căn chỉnh các phần tử.
- Thay đổi bố cục để sử dụng **LinearLayout**.
- Định vị các phần tử trong **LinearLayout**.
- Thay đổi bố cục để sử dụng **RelativeLayout**.
- Sắp xếp lại các view trong bố cục chính để liên kết tương đối với nhau.

Tổng quan về ứng dụng:

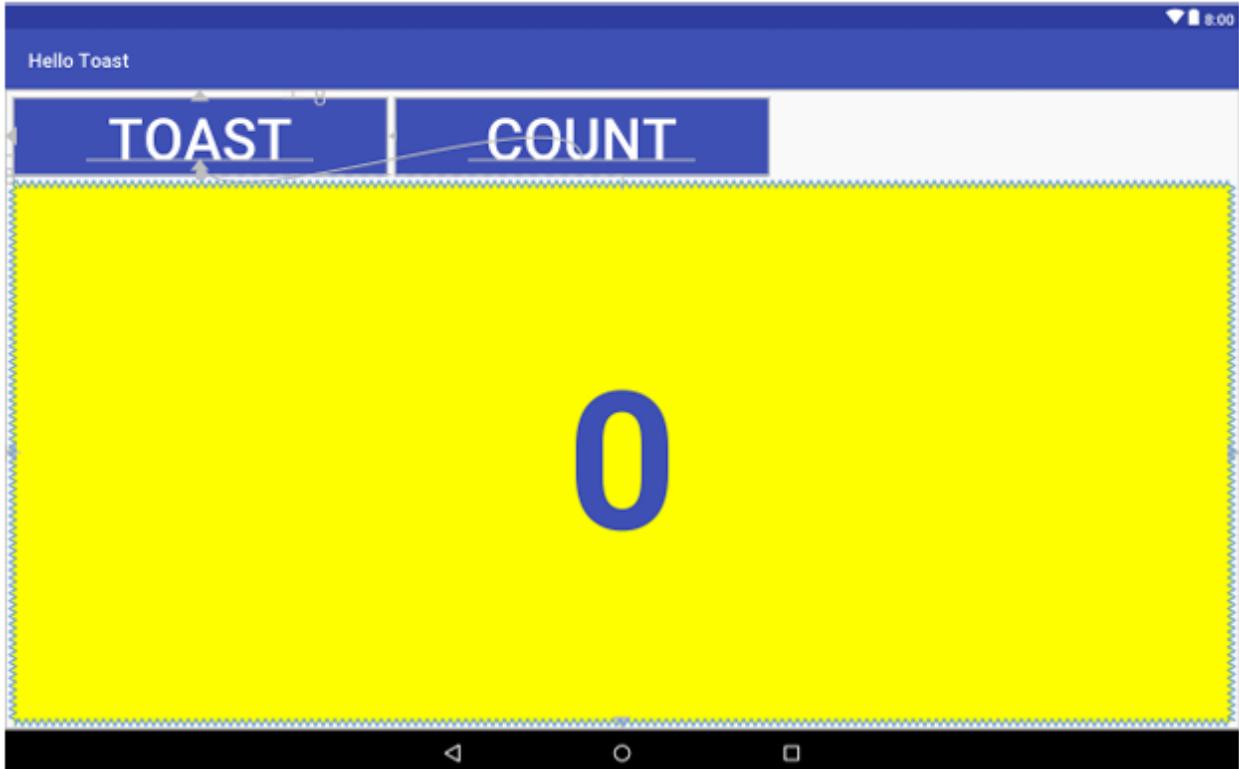
Ứng dụng **Hello Toast** từ bài học trước sử dụng **ConstraintLayout** để sắp xếp các phần tử UI trong bố cục của Activity, như minh họa trong hình dưới đây.



Để thực hành thêm với **ConstraintLayout**, bạn sẽ tạo một biến thể của bố cục này dành cho màn hình ngang (horizontal orientation), như minh họa trong hình dưới đây.



Bạn cũng sẽ học cách sử dụng ràng buộc đường cơ sở (baseline constraints) và một số tính năng căn chỉnh của **ConstraintLayout** bằng cách tạo một biến thể bố cục khác dành cho màn hình máy tính bảng.



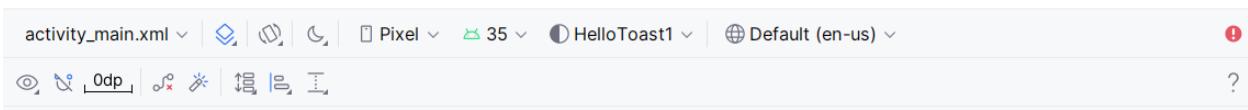
Bạn cũng sẽ tìm hiểu về các lớp con khác của **ViewGroup** như **LinearLayout** và **RelativeLayout**, đồng thời thay đổi bố cục của ứng dụng **Hello Toast** để sử dụng chúng.

Nhiệm vụ 1: Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu bạn thay đổi bố cục của ứng dụng **Hello Toast** để phù hợp với cả hai chế độ **dọc** và **ngang**. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo các biến thể bố cục cho các thiết bị di động ở chế độ **dọc** hoặc **ngang**, cũng như cho các thiết bị màn hình lớn như máy tính bảng.

Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trên **hai thanh công cụ trên cùng** trong trình chỉnh sửa bố cục (**Layout Editor**).

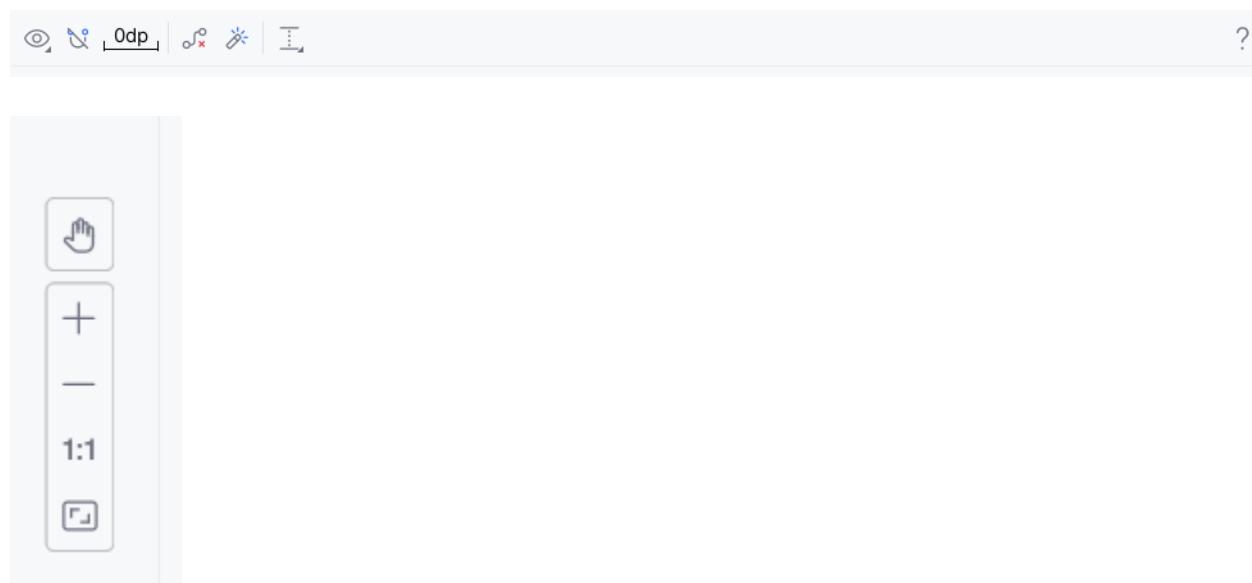
Thanh công cụ trên cùng giúp bạn cấu hình cách hiển thị bản xem trước của bố cục trong trình chỉnh sửa bố cục:



Các bước chính:

1. **Chọn bề mặt thiết kế (Design Surface):** Chọn **Design** để hiển thị bản xem trước màu sắc của bố cục, hoặc **Blueprint** để chỉ hiển thị các đường viền của từng phần tử giao diện người dùng (UI). Để xem cả hai chế độ cạnh nhau, chọn **Design + Blueprint**.
2. **Thay đổi hướng hiển thị trong trình chỉnh sửa (Orientation in Editor):** Chọn **Portrait** (chế độ dọc) hoặc **Landscape** (chế độ ngang) để xem trước bố cục theo chiều dọc hoặc ngang. Điều này rất hữu ích để xem trước bố cục mà không cần chạy ứng dụng trên trình giả lập hoặc thiết bị. Để tạo bố cục thay thế cho từng hướng, chọn **Create Landscape Variation** hoặc các biến thể khác.
3. **Chọn thiết bị trong trình chỉnh sửa (Device in Editor):** Chọn loại thiết bị (ví dụ: điện thoại/máy tính bảng, Android TV hoặc Android Wear).
4. **Chọn phiên bản API trong trình chỉnh sửa (API Version in Editor):** Chọn phiên bản Android để hiển thị bản xem trước.
5. **Chọn chủ đề trong trình chỉnh sửa (Theme in Editor):** Chọn một chủ đề (ví dụ: **AppTheme**) để áp dụng cho bản xem trước.
6. **Chọn ngôn ngữ và khu vực (Locale in Editor):** Chọn ngôn ngữ và khu vực cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có trong tài nguyên chuỗi văn bản (**string resources**). Bạn cũng có thể chọn **Preview as Right To Left** để xem bố cục như khi chọn một ngôn ngữ đọc từ phải sang trái (**RTL**).

Thanh công cụ thứ hai giúp bạn cấu hình cách hiển thị các phần tử giao diện người dùng trong **ConstraintLayout**, cũng như phóng to hoặc cuộn bản xem trước.



Trong hình trên:

1. **Hiển thị (Show):** Chọn **Show Constraints** và **Show Margins** để hiển thị hoặc ẩn các ràng buộc và khoảng cách trong bản xem trước.
2. **Autoconnect:** Bật hoặc tắt tính năng Autoconnect. Khi tính năng này được bật, bạn có thể kéo bất kỳ phần tử nào (như Button) đến bất kỳ phần nào trong bố cục để tự động tạo ràng buộc với bố cục cha.
3. **Xóa tất cả ràng buộc (Clear All Constraints):** Xóa toàn bộ ràng buộc trong bố cục.
4. **Suy luận ràng buộc (Infer Constraints):** Tạo ràng buộc bằng cách suy luận.
5. **Khoảng cách mặc định (Default Margins):** Thiết lập khoảng cách mặc định.
6. **Gói (Pack):** Gom nhóm hoặc mở rộng các phần tử được chọn.
7. **Căn chỉnh (Align):** Căn chỉnh các phần tử được chọn.
8. **Đường dẫn hướng (Guidelines):** Thêm đường dẫn hướng dọc hoặc ngang.
9. **Điều khiển phóng to/thu nhỏ (Zoom/pan controls):** Phóng to hoặc thu nhỏ bản xem trước.

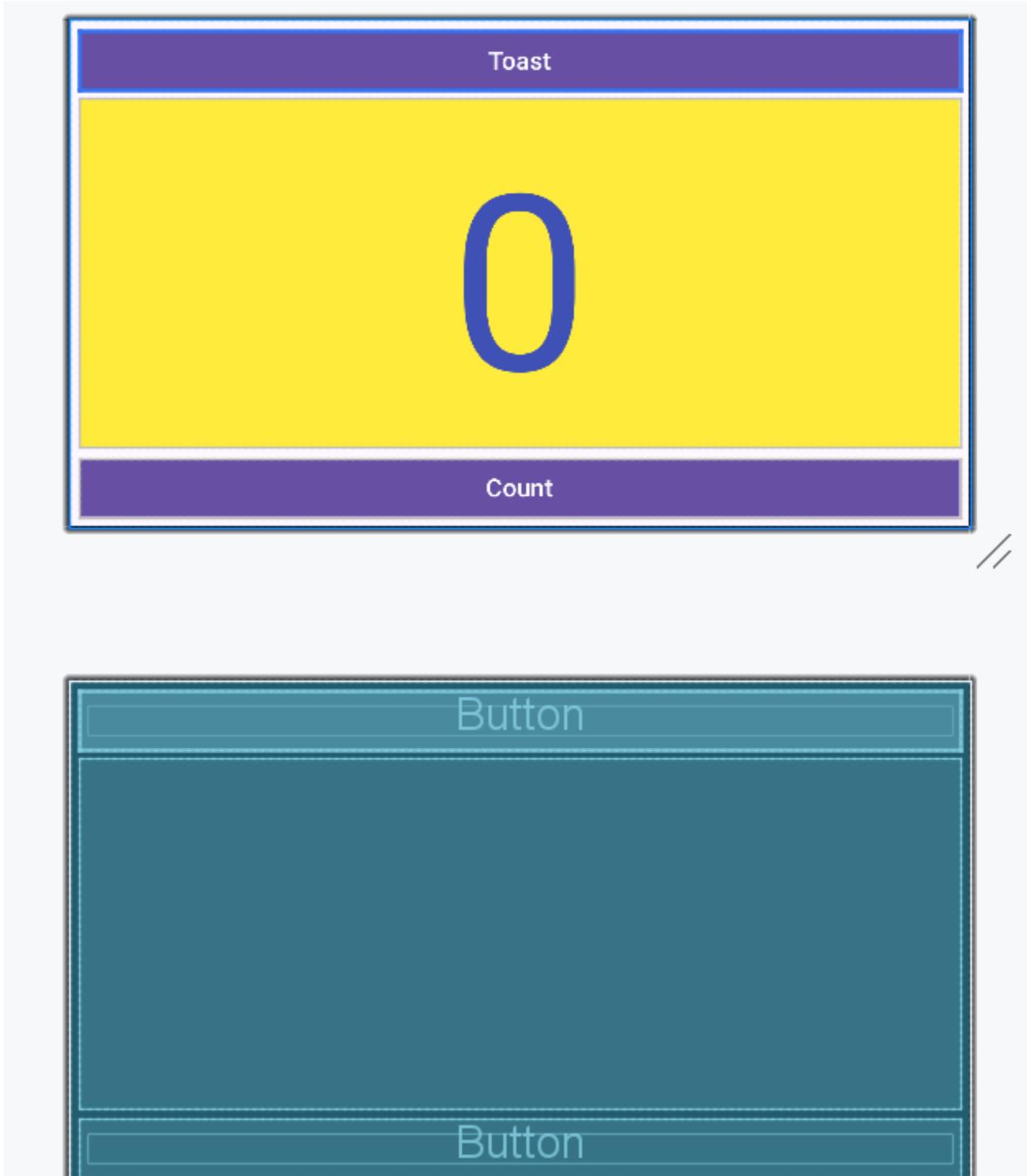
Mẹo: Để tìm hiểu thêm về cách sử dụng trình chỉnh sửa bố cục, xem **Build a UI with Layout Editor**. Để học cách xây dựng bố cục với **ConstraintLayout**, xem **Build a Responsive UI with ConstraintLayout**.

1.1 Xem trước bố cục ở chế độ ngang

Để xem trước bố cục ứng dụng **Hello Toast** với chế độ ngang, hãy làm theo các bước sau:

1. Mở ứng dụng **Hello Toast** từ bài học trước.
 - **Lưu ý:** Nếu bạn đã tải xuống mã hoàn chỉnh cho HelloToast, hãy xóa các bố cục chế độ ngang (**landscape**) và màn hình cực lớn (**extra-large**) đã hoàn thành mà bạn sẽ tạo trong nhiệm vụ này.
 - Chuyển từ chế độ hiển thị **Project > Android** sang **Project > Project Files** trong bảng **Project**. Sau đó, mở rộng đường dẫn **app > src/main > res**, chọn cả hai thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**.
 - Sau khi hoàn tất, chuyển lại bảng **Project** về **Project > Android**.
2. Mở tệp **activity_main.xml** trong thư mục bố cục (**layout**). Nhấp vào tab **Design** nếu nó chưa được chọn.

3. Nhấp vào nút **Orientation in Editor** trên thanh công cụ ở phía trên để thay đổi hướng hiển thị.
4. Trong menu thả xuống, chọn **Switch to Landscape**. Bộ cục sẽ xuất hiện ở chế độ ngang, như minh họa bên dưới. Để quay lại chế độ dọc, chọn **Switch to Portrait**.



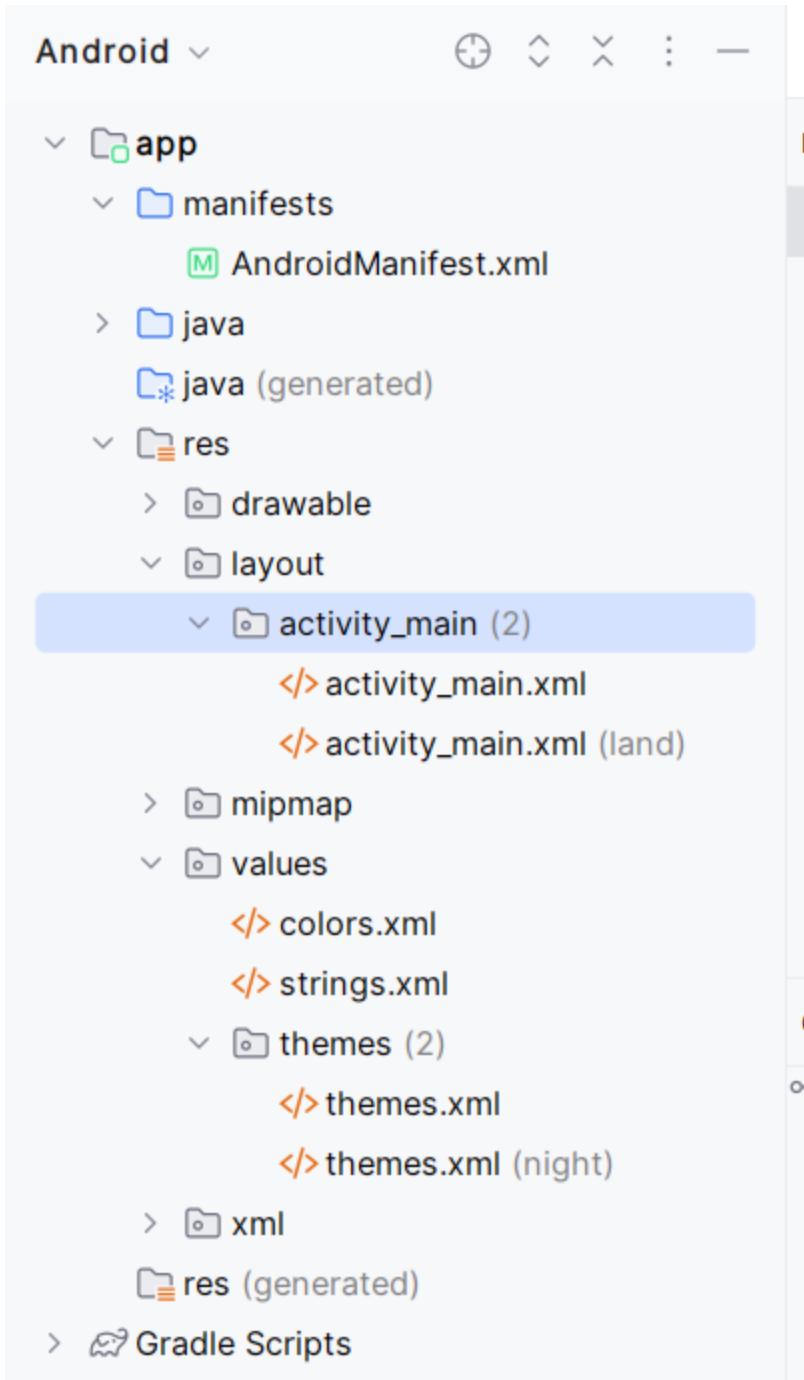
1.2 Tạo biến thể bố cục cho chế độ ngang

Sự khác biệt về mặt hình ảnh giữa chế độ dọc và ngang trong bố cục này là chữ số **0** trong phần tử **show_count** (**TextView**) nằm quá thấp đối với chế độ ngang, quá gần với nút **Count** và kích thước cố định của **TextView** (160sp) có thể khiến bố cục trong chế độ ngang trông không cân đối, với chữ số **0** quá lớn hoặc không được căn giữa. Để giải quyết vấn đề này trong chế độ ngang mà vẫn giữ nguyên bố cục cho chế độ dọc, bạn có thể tạo một biến thể bố cục cho ứng dụng **Hello Toast**. Thực hiện theo các bước sau:

1. Nhấp vào nút **Orientation in Editor** trên thanh công cụ phía trên.
2. Chọn **Create Landscape Variation**.

Khi đó, một cửa sổ chỉnh sửa mới sẽ mở ra với tab **land/activity_main.xml**, hiển thị bố cục dành riêng cho chế độ ngang. Bạn có thể thay đổi bố cục này mà không ảnh hưởng đến bố cục gốc cho chế độ dọc.

3. Trong bảng **Project > Android**, mở thư mục **res > layout**, bạn sẽ thấy Android Studio đã tự động tạo biến thể mới có tên **activity_main.xml (land)**.



1.3 Xem trước bố cục trên các thiết bị khác nhau

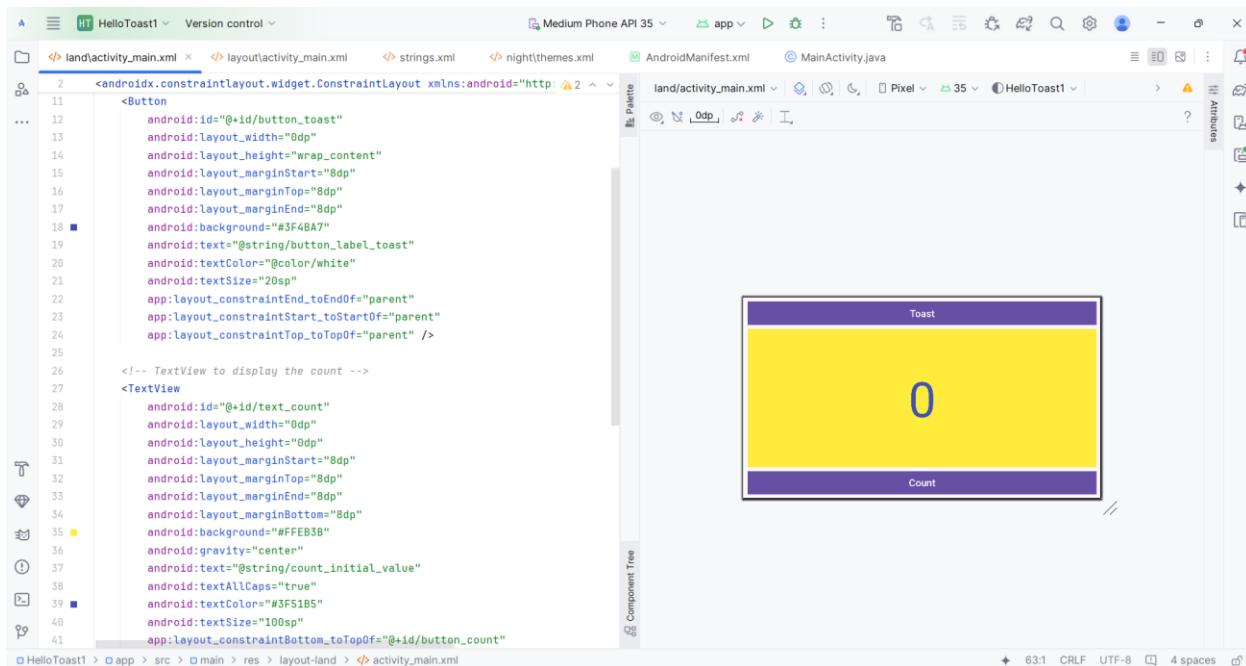
Bạn có thể xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thực hoặc trình giả lập. Thực hiện các bước sau:

1. Tab **land/activity_main.xml** vẫn đang mở trong trình chỉnh sửa bố cục. Nếu không, hãy nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục **layout**.

- Nhập vào nút **Device in Editor** trên thanh công cụ phía trên.
- Trong menu thả xuống, chọn một thiết bị khác. Ví dụ, chọn **Nexus 4, Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong các bản xem trước. Những khác biệt này xảy ra do kích thước văn bản cố định của **TextView**.

1.4 Thay đổi bố cục cho chế độ ngang

Bạn có thể sử dụng bảng **Attributes** trong tab **Design** để thiết lập hoặc thay đổi các thuộc tính. Tuy nhiên, đôi khi việc chỉnh sửa trực tiếp mã XML trong tab **Text** sẽ nhanh hơn. Tab **Text** hiển thị mã XML và cung cấp một tab **Preview** ở phía bên phải cửa sổ để hiển thị bản xem trước bố cục, như minh họa trong hình bên dưới.



Thay đổi bố cục với các bước chi tiết

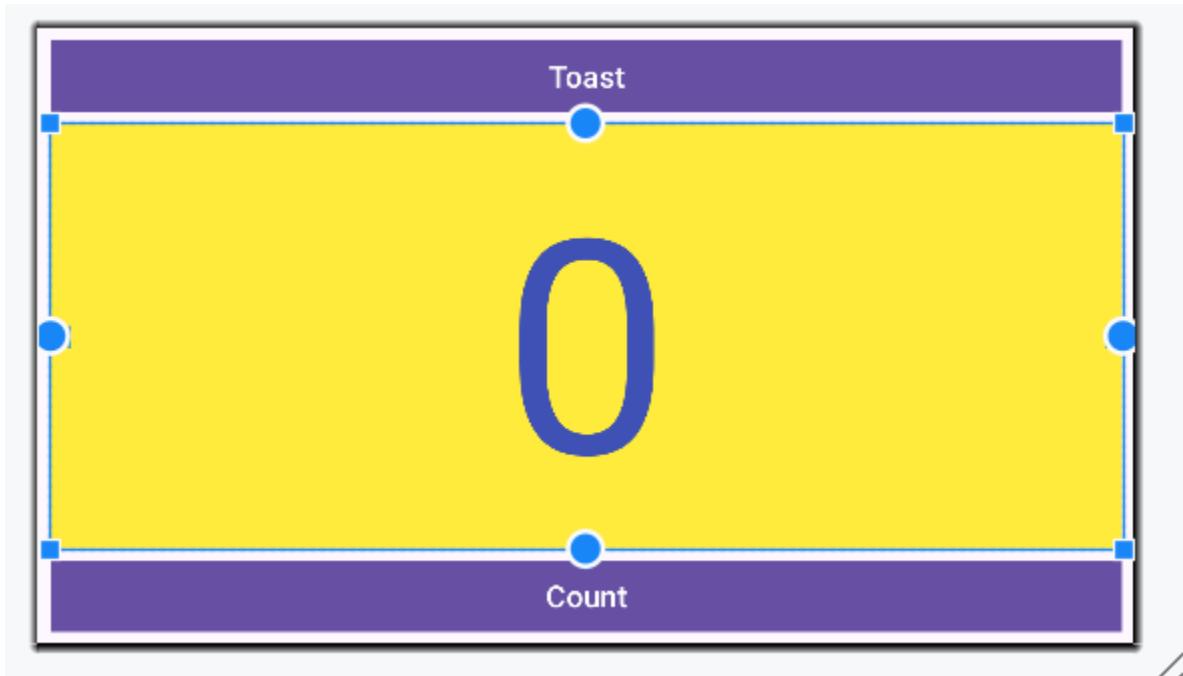
Hình minh họa:

- Tab Preview:** Sử dụng để hiển thị bảng xem trước bố cục.
- Preview Pane:** Hiển thị cách bố cục trông trên thiết bị.
- XML Code:** Phần mã nguồn của bố cục trong tab **Text**.

Các bước thay đổi bố cục:

- Đảm bảo tab **land/activity_main.xml** vẫn mở trong trình chỉnh sửa bố cục. Nếu không, nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục **layout**.

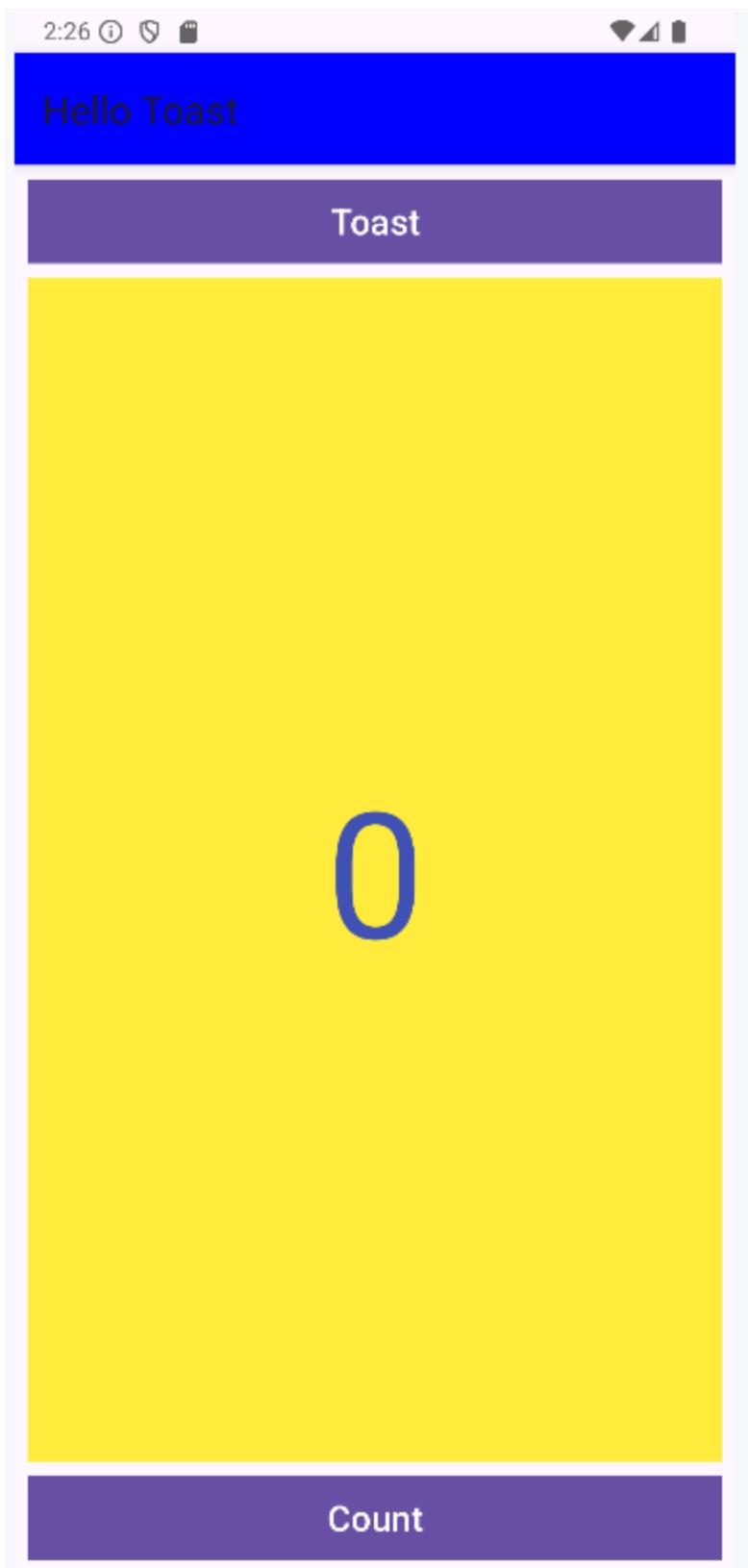
2. Chuyển sang tab **Text** và bật tab **Preview** (nếu chưa được chọn).
3. Trong mã XML, tìm phần tử **TextView**.
4. Thay đổi thuộc tính `android:textSize="100sp"` thành `android:textSize="120sp"`. Kết quả thay đổi sẽ hiển thị ngay lập tức trong bản xem trước.

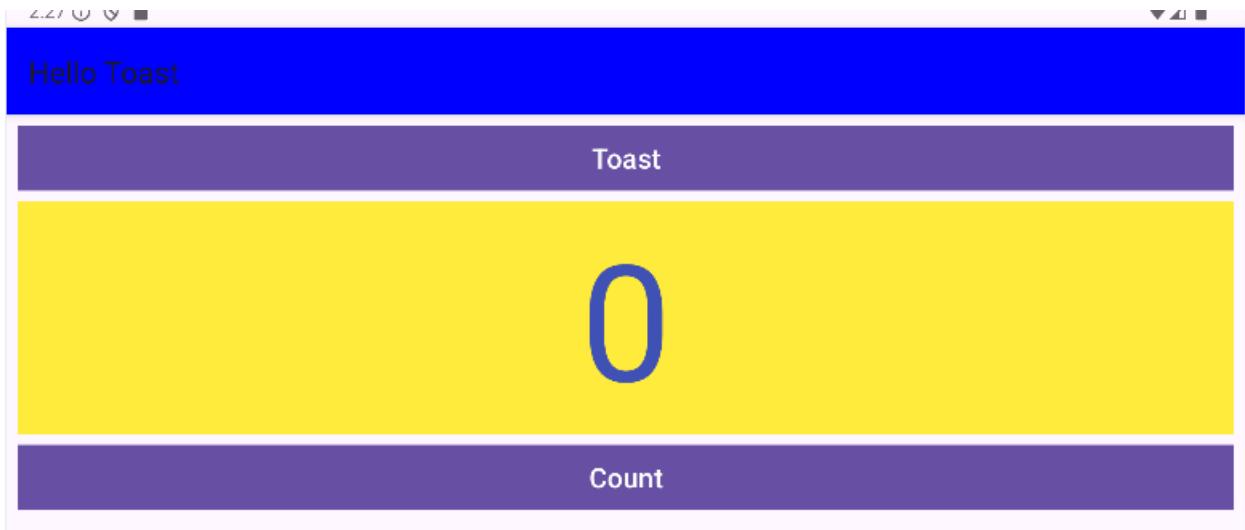


5. Sử dụng menu thả xuống **Device in Editor** để chọn các thiết bị khác nhau và xem cách bố cục hiển thị ở chế độ ngang trên các thiết bị đó.

Lưu ý trong trình chỉnh sửa:

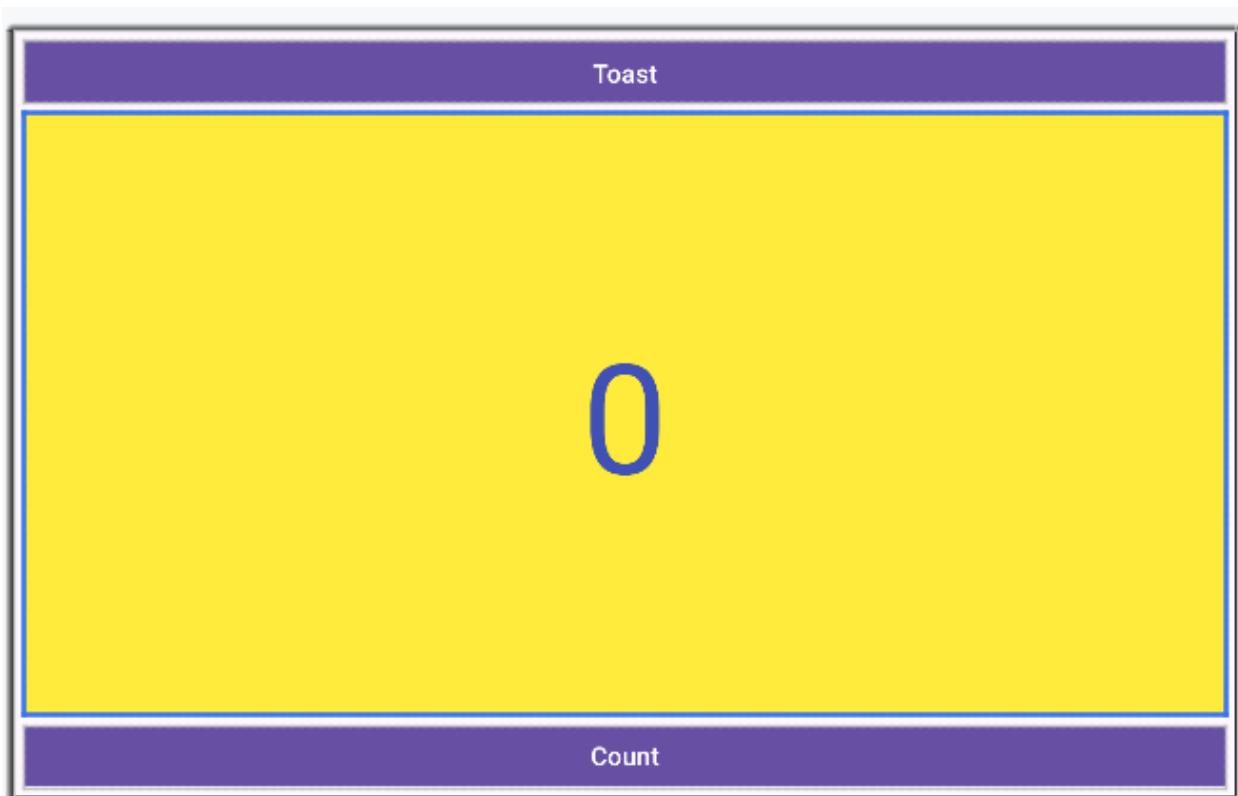
- Tab **land/activity_main.xml** hiển thị bố cục cho chế độ ngang.
 - Tab **activity_main.xml** hiển thị bố cục không thay đổi cho chế độ dọc. Bạn có thể chuyển đổi qua lại giữa hai tab để so sánh.
6. Chạy ứng dụng trên trình giả lập hoặc thiết bị thực. Chuyển hướng hiển thị từ dọc sang ngang để xem cách cả hai bố cục hoạt động.





1.5 Tạo một biến thể bố cục cho máy tính bảng

Như bạn đã học trước đây, bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ phía trên. Nếu bạn chọn một thiết bị như **Nexus 10** (máy tính bảng) từ menu, bạn sẽ nhận thấy rằng bố cục không phù hợp cho màn hình máy tính bảng lớn - văn bản của mỗi nút (**Button**) quá nhỏ và cách sắp xếp các nút (**Button**) ở trên và dưới không lý tưởng cho màn hình lớn của máy tính bảng.



Để điều chỉnh bố cục phù hợp với máy tính bảng mà không ảnh hưởng đến bố cục cho các thiết bị điện thoại ở chế độ ngang hoặc dọc, hãy thực hiện các bước sau:

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị các bảng thiết kế và bản vẽ.
2. Nhấp vào nút **Orientation in Editor** trên thanh công cụ phía trên.
3. Chọn **Create layout x-large Variation**.

Sau khi thực hiện:

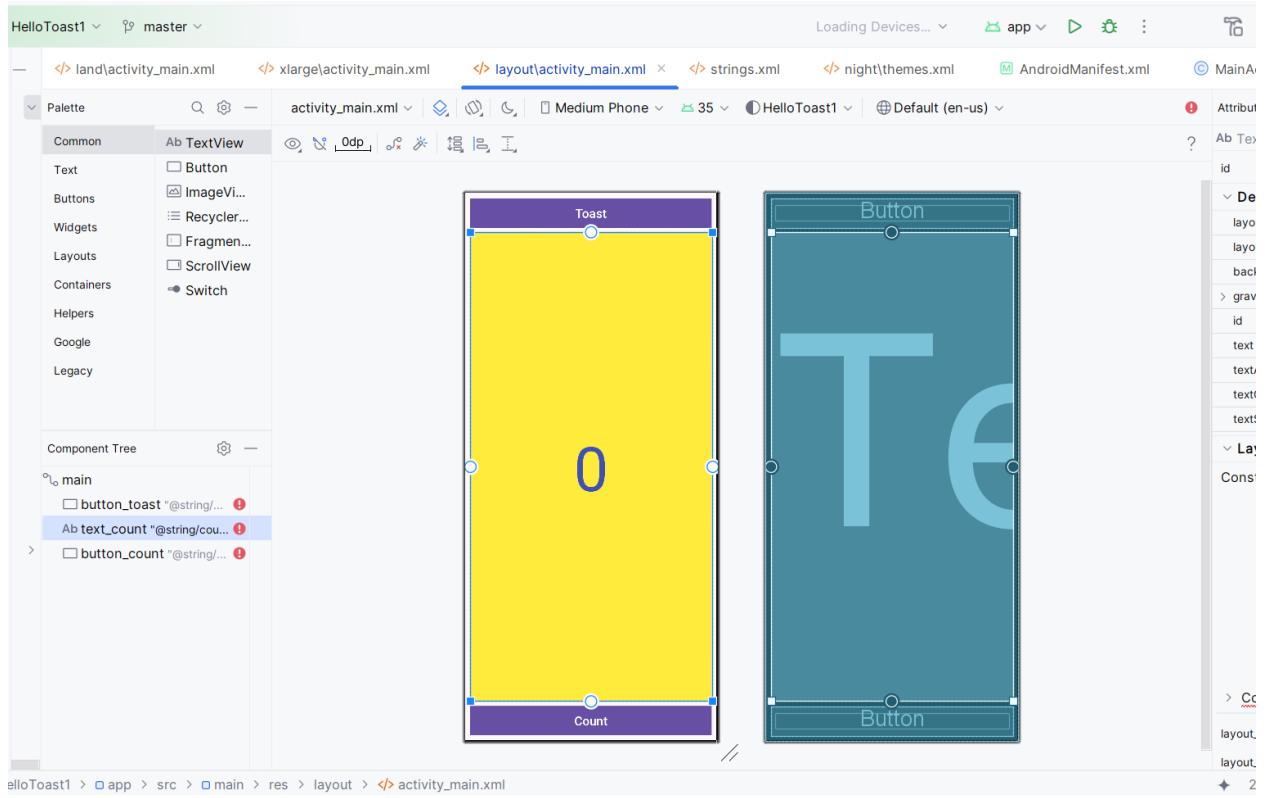
- Một cửa sổ chỉnh sửa mới sẽ mở ra với tab **xlarge/activity_main.xml**, hiển thị bố cục dành riêng cho thiết bị có kích thước màn hình máy tính bảng.
- Trình chỉnh sửa cũng tự động chọn một thiết bị máy tính bảng, chẳng hạn như **Nexus 9** hoặc **Nexus 10**, để hiển thị bản xem trước.

Bạn có thể thay đổi bố cục này, dành riêng cho máy tính bảng, mà không làm thay đổi các bố cục khác.

1.6 Thay đổi biến thể bố cục cho máy tính bảng

Bạn có thể sử dụng bảng thuộc tính (Attributes pane) trong tab **Design** để thay đổi các thuộc tính cho bố cục này.

1. **Tắt công cụ Autoconnect** trên thanh công cụ. Ở bước này, hãy đảm bảo rằng công cụ đã bị vô hiệu hóa.
2. Xóa tất cả các ràng buộc (constraints) trong bố cục bằng cách nhấp vào nút **Clear All Constraints** trên thanh công cụ.
 - Khi các ràng buộc đã bị xóa, bạn có thể tự do di chuyển và thay đổi kích thước các phần tử trên bố cục.
3. Trình chỉnh sửa bố cục cung cấp các tay cầm (handles) ở bốn góc của mỗi phần tử để thay đổi kích thước chúng. Trong **Component Tree**, chọn **TextView** có tên **show_count**. Để dời **TextView** ra ngoài đường đi, giúp bạn dễ dàng kéo các phần tử **Button**, hãy kéo một góc của nó để thay đổi kích thước, như được minh họa trong hình động dưới đây.

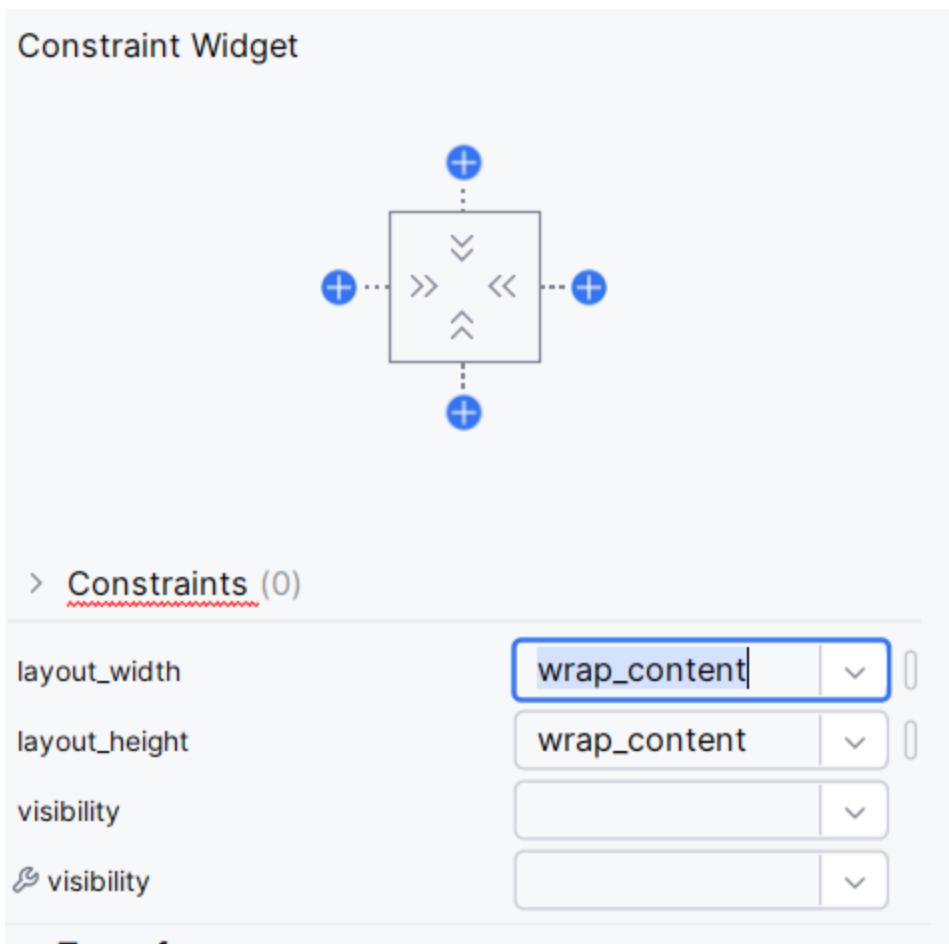


Khi thay đổi kích thước một phần tử, nó sẽ cố định các giá trị chiều rộng và chiều cao. Tránh cố định kích thước cho hầu hết các phần tử, vì điều này có thể dẫn đến giao diện không nhất quán trên các màn hình có kích thước và mật độ khác nhau. Trong bước này, việc thay đổi kích thước chỉ được thực hiện để di chuyển phần tử tạm thời, và các kích thước sẽ được điều chỉnh lại trong bước khác.

Bước 4:

Chọn nút **button_toast** trong **Component Tree**, nhấp vào tab **Attributes** để mở bảng thuộc tính, và thay đổi **textSize** thành **60sp** (#1 trong hình bên dưới), **layout_width** thành **wrap_content** (#2 trong hình bên dưới)

Button		button_toa
id	button_toast	
Declared Attributes		+ -
layout_width	wrap_content	▾
layout_height	wrap_content	▾
background	#3F4BA7	
id	button_toast	
text	@string/button_label_...	
textColor	@color/white	
textSize	60sp	▾
Layout		



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

Bạn sử dụng **wrap_content** để nếu văn bản của **Button** được bắn địa hóa sang một ngôn ngữ khác, nút **Button** sẽ hiển thị rộng hơn hoặc hẹp hơn để phù hợp với từ trong ngôn ngữ khác.

- Chọn nút **button_count** trong **Component Tree**, thay đổi **textSize** thành **60sp** và **layout_width** thành **wrap_content**, sau đó kéo nút **Button** lên trên **TextView** vào một không gian trống trong bố cục.

1.7 Sử dụng ràng buộc đường cơ sở (baseline constraint)

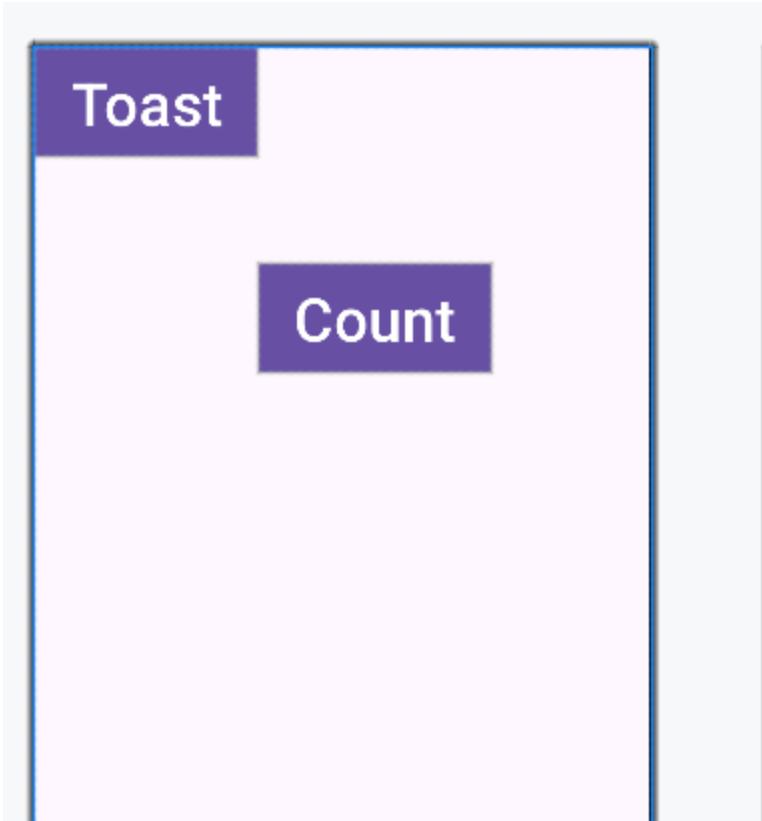
Bạn có thể căn chỉnh một phần tử giao diện người dùng (UI) chưa văn bản, như **TextView** hoặc **Button**, với một phần tử giao diện khác cũng chưa văn bản.

Ràng buộc đường cơ sở (**baseline constraint**) cho phép bạn ràng buộc các phần tử sao cho các đường cơ sở của văn bản được căn thẳng hàng.

1. Ràng buộc nút **button_toast** vào phía trên và bên trái của bố cục. Kéo nút **button_count** đến một vị trí gần nút **button_toast**. Sau đó, ràng buộc nút **button_count** vào phía bên trái của nút **button_toast**, như minh họa trong hình động.



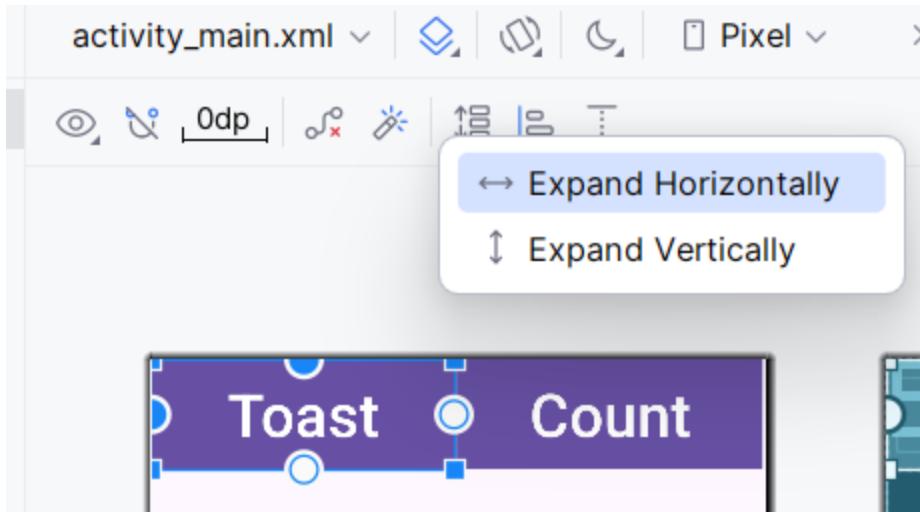
2. **Sử dụng ràng buộc đường cơ sở (baseline constraint)**, bạn có thể ràng buộc nút **button_count** sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của nút **button_toast**. Chọn phần tử **button_count**, sau đó di chuyển con trỏ chuột qua phần tử cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.
3. Nhấp vào nút ràng buộc đường cơ sở. Tay cầm ràng buộc đường cơ sở sẽ xuất hiện, nhấp nháy màu xanh lá. Nhấp và kéo một đường ràng buộc đường cơ sở đến đường cơ sở của phần tử **button_toast** để hoàn tất.



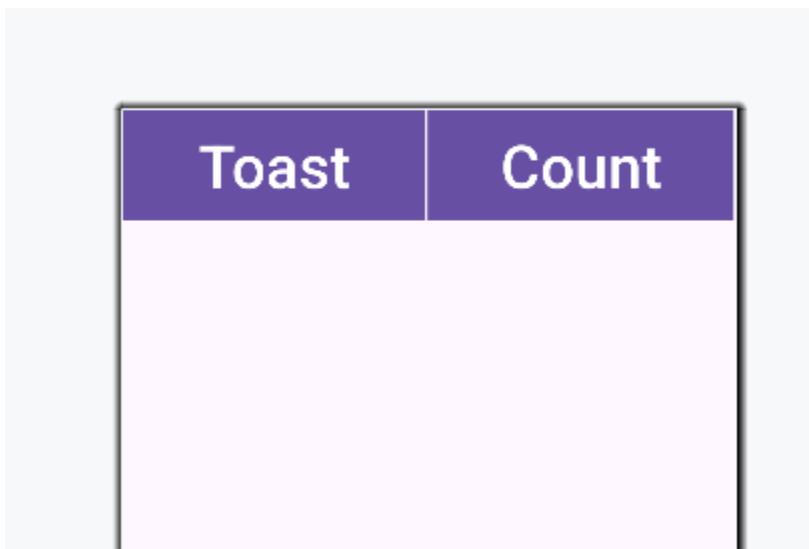
1.8 Mở rộng các nút theo chiều ngang

Nút **pack** trên thanh công cụ cung cấp các tùy chọn để sắp xếp hoặc mở rộng các phần tử giao diện người dùng đã chọn. Bạn có thể sử dụng nó để sắp xếp đều các nút **Button** theo chiều ngang trên giao diện.

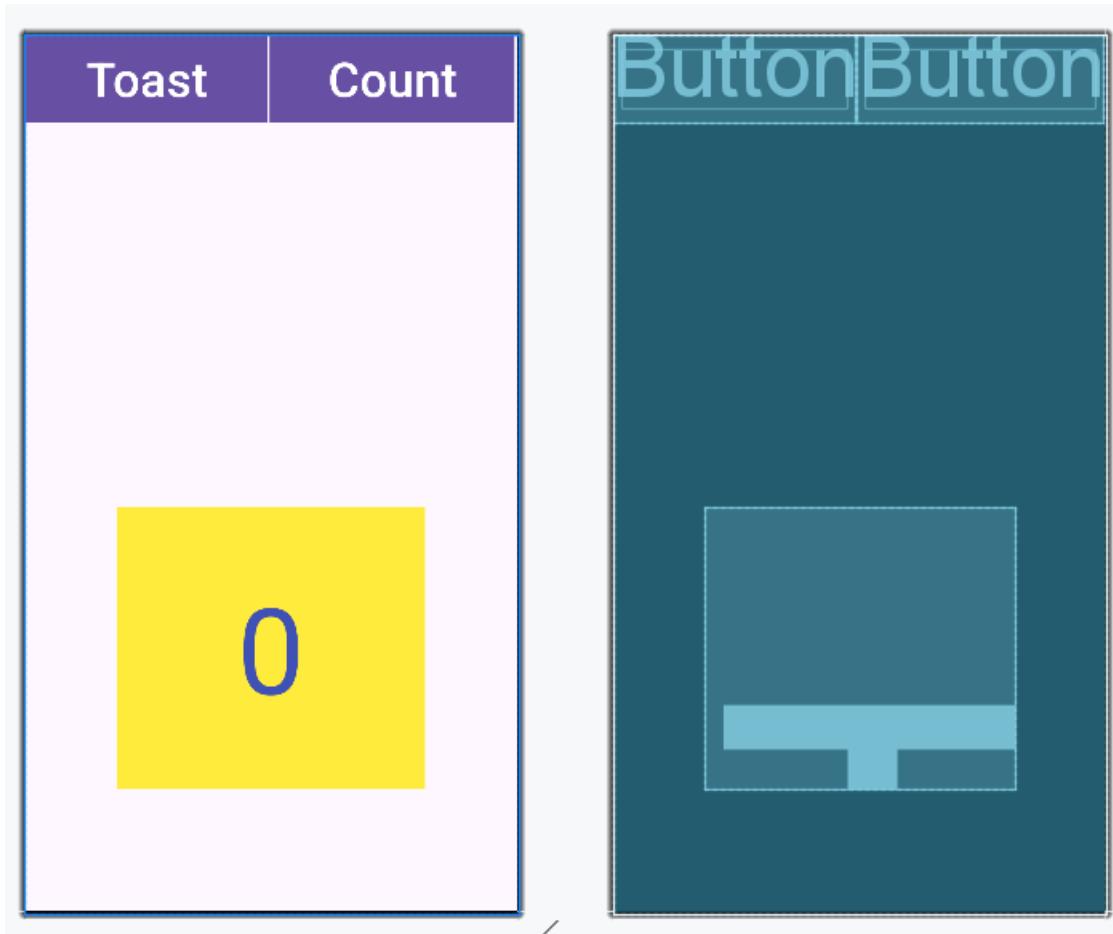
1. Chọn nút **button_count** trong **Component Tree**, sau đó nhấn giữ **Shift** và chọn thêm nút **button_toast** để cả hai nút đều được chọn.
2. Nhấp vào nút **pack** trên thanh công cụ, sau đó chọn tùy chọn **Expand Horizontally** như minh họa trong hình.



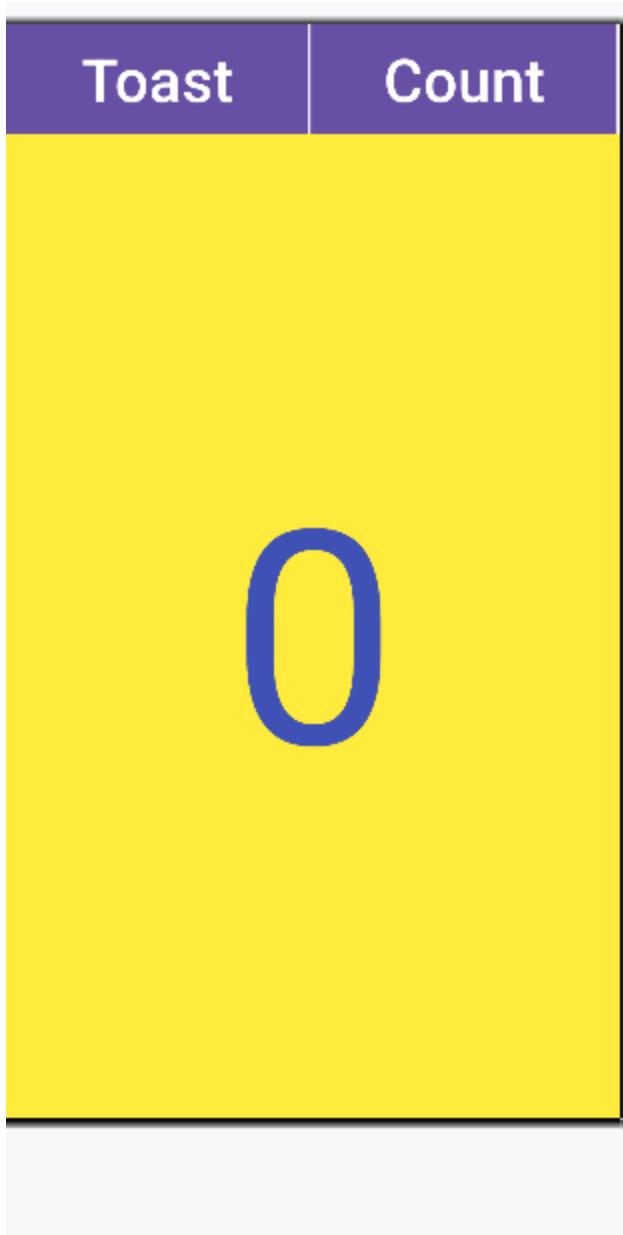
Các phần tử **Button** sẽ mở rộng theo chiều ngang để lấp đầy toàn bộ giao diện như minh họa bên dưới.



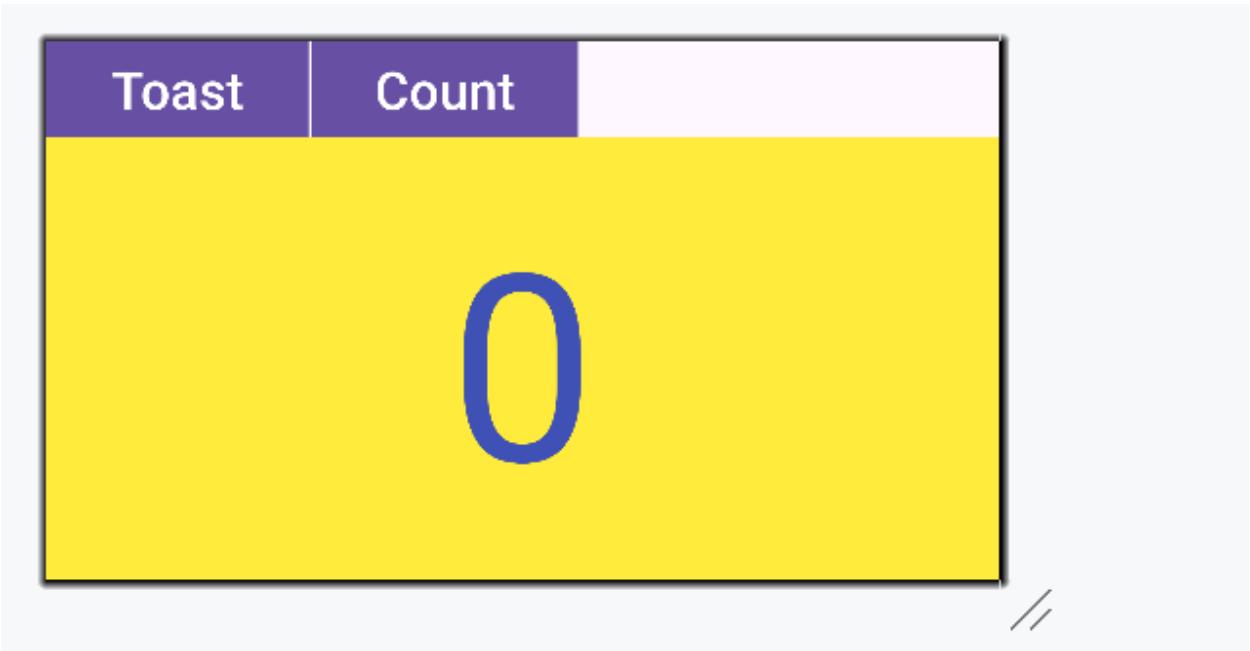
3. Để hoàn thiện bố cục, hãy ràng buộc **show_count TextView** vào phía dưới của nút **button_toast** và ràng buộc nó với các cạnh bên cũng như cạnh dưới của bố cục, như được minh họa trong hình động bên dưới.



4. Các bước cuối cùng là thay đổi **layout_width** và **layout_height** của **show_count TextView** thành **Match Constraints** và đặt **textSize** thành **200sp**. Bố cục cuối cùng sẽ trông như hình minh họa bên dưới.



5. Nhấn vào nút **Orientation in Editor** trên thanh công cụ ở phía trên và chọn **Switch to Landscape** (Chuyển sang chế độ ngang). Giao diện của bố cục trên máy tính bảng sẽ xuất hiện với hướng nằm ngang như hình dưới đây. (Bạn có thể chọn **Switch to Portrait** để quay lại chế độ dọc).

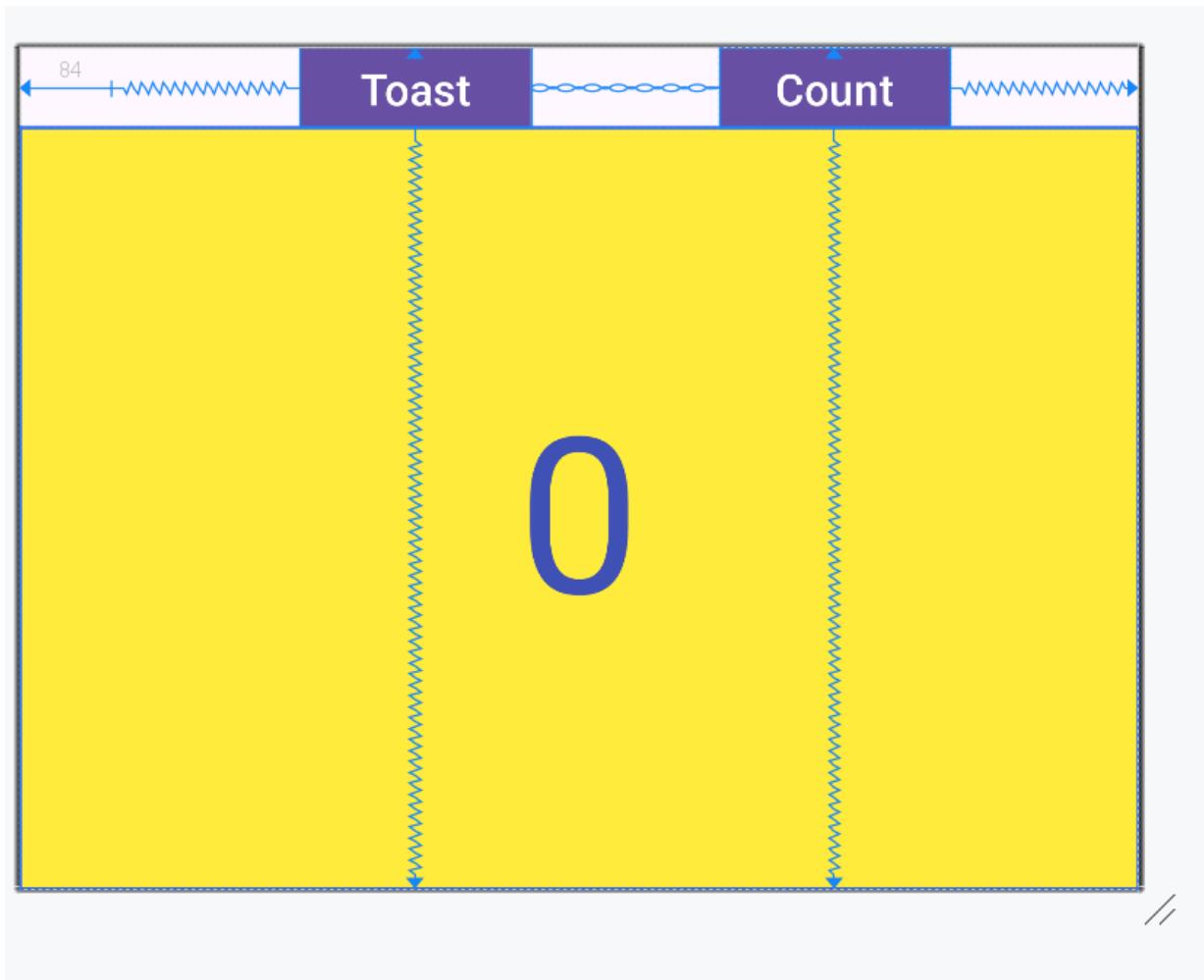


6. Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng màn hình sau khi chạy ứng dụng để xem giao diện trông như thế nào trên các loại thiết bị khác nhau. Bạn đã thành công trong việc tạo một ứng dụng có giao diện người dùng (UI) hoạt động đúng trên điện thoại và máy tính bảng với các kích thước và mật độ màn hình khác nhau.

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

Thử thách: Để hỗ trợ giao diện ngang (landscape) cho máy tính bảng, bạn có thể căn giữa các nút (Button) trong tệp activity_main.xml (xlarge) để chúng xuất hiện như hình minh họa bên dưới.

Gợi ý: Chọn các phần tử, nhấp vào nút căn chỉnh trong thanh công cụ, và chọn **Căn giữa theo chiều ngang (Center Horizontally)**.



Bài 2: Thay đổi bố cục thành LinearLayout

LinearLayout là một **ViewGroup** sắp xếp tập hợp các **view** theo hàng ngang hoặc hàng dọc.

LinearLayout là một trong những bố cục phổ biến nhất vì nó đơn giản và nhanh. Nó thường được sử dụng trong một **view group** khác để sắp xếp các phần tử giao diện người dùng theo chiều ngang hoặc dọc.

LinearLayout yêu cầu có các thuộc tính sau:

- **layout_width**
- **layout_height**
- **orientation**

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

- **match_parent**: Mở rộng view để lấp đầy **parent** theo chiều rộng hoặc chiều cao. Khi **LinearLayout** là **root view**, nó sẽ mở rộng theo kích thước của màn hình (**parent view**).
- **wrap_content**: Thu nhỏ kích thước view sao cho nó vừa đủ chứa nội dung. Nếu không có nội dung, view sẽ trở nên vô hình.
- **Số dp cố định (density-independent pixels)**: Xác định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ, **16dp** có nghĩa là 16 pixel độc lập với mật độ.

orientation có thể là:

- **horizontal**: Các **view** được sắp xếp từ trái sang phải.
- **vertical**: Các **view** được sắp xếp từ trên xuống dưới.

2.1 Thay đổi root view group thành **LinearLayout**

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), và nhấp vào tab **Text** ở cuối khung chỉnh sửa để xem mã XML. Ở dòng đầu tiên của mã XML, bạn sẽ thấy dòng sau:

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Thay đổi thẻ **<android.support.constraint.ConstraintLayout** thành **<LinearLayout** để mã XML trông như sau:

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

4. Đảm bảo rằng thẻ đóng ở cuối mã đã được thay đổi thành **</LinearLayout>** (Android Studio sẽ tự động thay đổi thẻ đóng nếu bạn thay đổi thẻ mở). Nếu nó không thay đổi tự động, hãy chỉnh sửa thủ công.
5. Ngay dưới dòng thẻ **<LinearLayout**, thêm thuộc tính sau vào sau thuộc tính **android:layout_height**:

```
    android:layout_height="match_parent"
```

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

Sau khi thực hiện các thay đổi này, một số thuộc tính XML của các phần tử khác sẽ bị gạch chân màu đỏ vì chúng được sử dụng với **ConstraintLayout** và không phù hợp với **LinearLayout**.

2.2 Thay đổi thuộc tính của các phần tử để phù hợp với LinearLayout

Thực hiện các bước sau để thay đổi thuộc tính của các phần tử giao diện người dùng sao cho chúng hoạt động với **LinearLayout**:

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), và nhập vào tab **Text**.
3. Tìm phần tử **Button** có **id** là **button_toast**, và thay đổi thuộc tính sau:

```
    android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3CB371"
```

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

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

5. Tìm phần tử **Button** có **id** là **button_count**, và thay đổi thuộc tính sau.

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

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

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

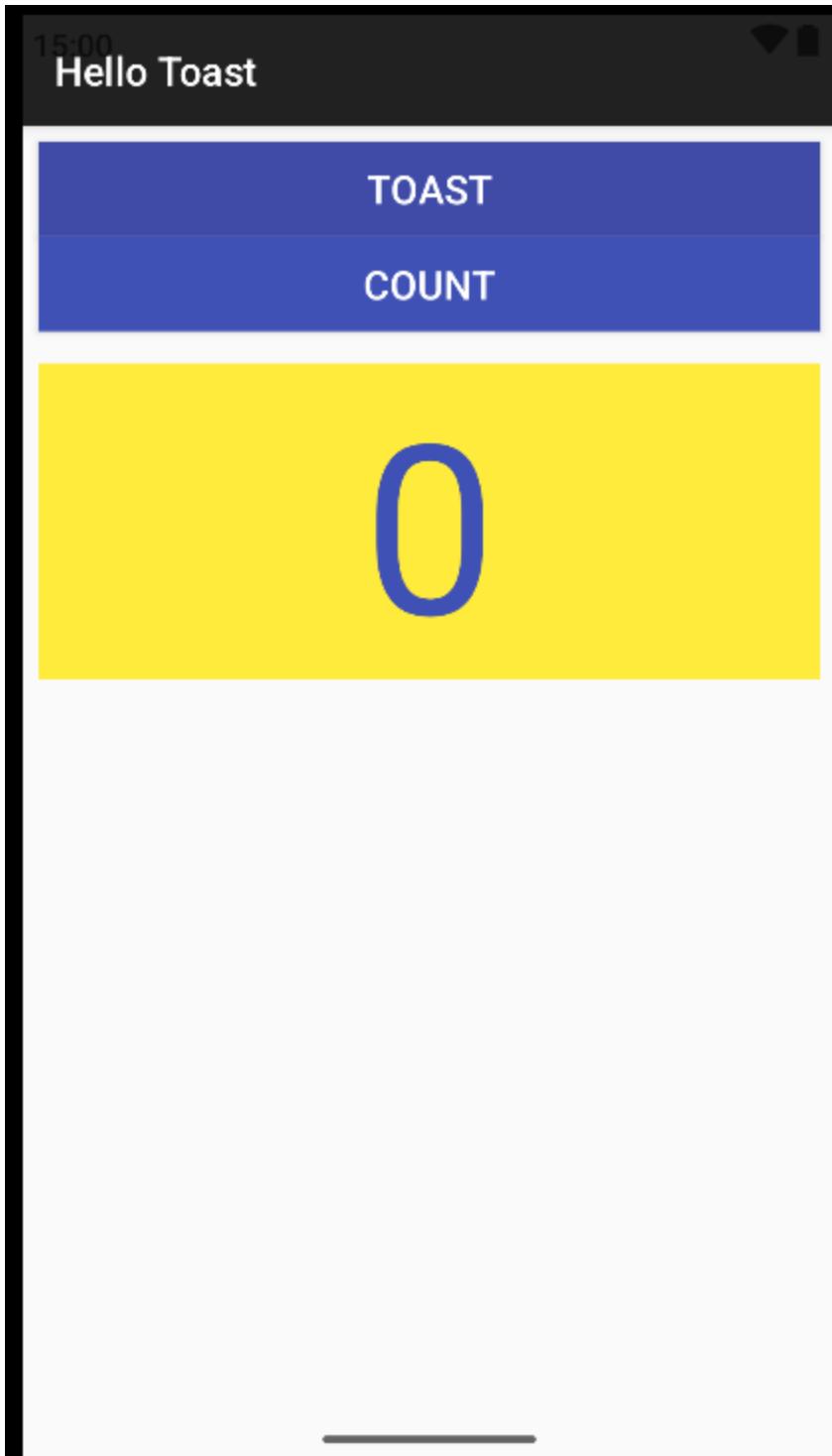
7. Tìm phần tử **TextView** có **id** là **show_count**, và thay đổi các thuộc tính sau.

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

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

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

9. Nhấp vào tab **Preview** ở phía bên phải cửa sổ Android Studio (nếu chưa được chọn) để xem trước bố cục hiện tại.



2.3 Thay đổi vị trí của các phần tử trong LinearLayout

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

Để thay đổi vị trí của chúng sao cho nút **Count** nằm ở phía dưới, hãy làm theo các bước sau:

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp bố cục **activity_main.xml** (nếu chưa mở), sau đó nhấp vào tab **Text**.
3. Chọn nút **button_count** và tất cả các thuộc tính của nó, từ thẻ `<Button` đến hết dấu đóng `>`, sau đó chọn **Edit > Cut (Cắt)**.
4. Nhấp chuột sau thẻ đóng `>` của phần tử **TextView**, nhưng trước thẻ đóng `</LinearLayout>`, sau đó chọn **Edit > Paste (Dán)**.
5. (Tùy chọn) Để chỉnh sửa lại khoảng cách hoặc thụt dòng cho đẹp mắt, chọn **Code > Reformat Code** để định dạng lại mã XML với khoảng cách và thụt dòng phù hợp.

Bây giờ, mã XML cho các phần tử giao diện sẽ trông giống như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<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 for showing toast -->

    <!-- TextView to display the count -->

    <!-- Button for incrementing the count -->

    <Button
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3F4BA7"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
```

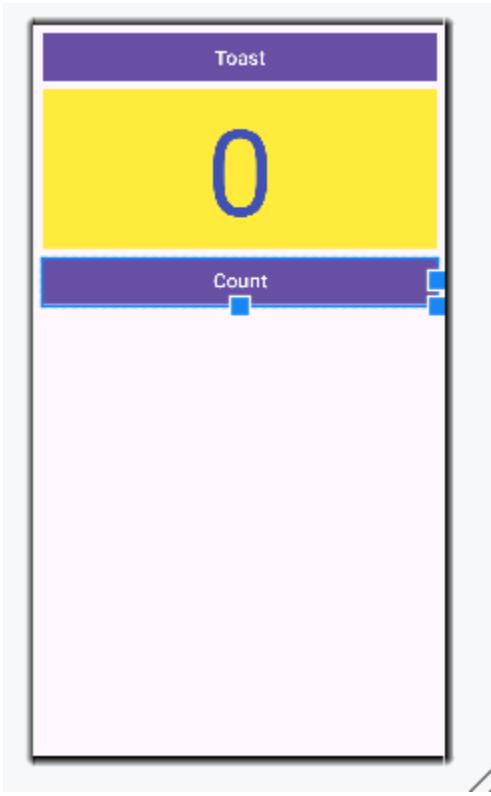
```
    android:layout_marginEnd="8dp"
    android:text="@string/button_label_toast"
    android:textColor="@color/white"
    android:textSize="20sp" />
```

```
<TextView
    android:id="@+id/text_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FFEB3B"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAllCaps="true"
    android:textColor="#3F51B5"
    android:textSize="120dp" />
```

```
<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#3F51B5"
    android:text="@string/button_label_count"
    android:layout_marginStart="8dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:textColor="@color/white"
    android:textSize="20dp" />
```

```
</LinearLayout>
```

Bằng cách di chuyển nút **button_count** xuống dưới **TextView**, bố cục giờ đây gần giống với trước đó, với nút **Count** nằm ở phía dưới. Bản xem trước của bố cục bây giờ trông như sau:



2.4 Thêm thuộc tính weight cho phần tử TextView

Việc xác định các thuộc tính **gravity** và **weight** giúp bạn kiểm soát tốt hơn cách sắp xếp các **View** và nội dung trong **LinearLayout**.

- Thuộc tính `android:gravity` xác định cách căn chỉnh nội dung bên trong một **View**. Ở bài trước, bạn đã đặt thuộc tính này cho **show_count** (**TextView**) để căn giữa nội dung (chữ số **0**) trong khung **TextView**.

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

- Thuộc tính `android:layout_weight` xác định lượng không gian thừa trong **LinearLayout** mà **View** đó sẽ chiếm. Nếu chỉ có một **View** có thuộc tính

này, nó sẽ chiếm toàn bộ không gian trống. Nếu có nhiều View có **weight**, không gian sẽ được chia theo tỷ lệ.

- Ví dụ: Nếu mỗi nút **Button** có `weight="1"` và **TextView** có `weight="2"`, tổng cộng là 4, thì mỗi nút **Button** sẽ nhận $\frac{1}{4}$ không gian, còn **TextView** sẽ nhận $\frac{1}{2}$ không gian.

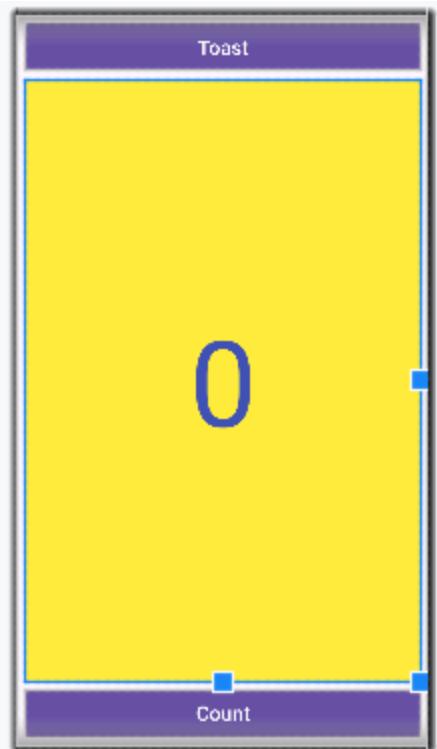
Trên các thiết bị khác nhau, **show_count** (**TextView**) có thể chiếm một phần hoặc phần lớn không gian giữa hai nút **Toast** và **Count**. Để đảm bảo **TextView** mở rộng và lấp đầy không gian trống trên mọi thiết bị, hãy thêm thuộc tính `android:layout_weight`.

Các bước thực hiện

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp bố cục **activity_main.xml** (nếu chưa mở), sau đó nhấp vào tab **Text**.
3. Tìm phần tử **show_count** (**TextView**) và thêm thuộc tính sau:

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

Bản xem trước bây giờ trông như hình sau.



Bây giờ, phần tử **show_count** (**TextView**) sẽ chiếm toàn bộ không gian giữa các nút bấm. Bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ của cửa sổ xem trước và chọn một thiết bị khác.

Dù chọn thiết bị nào, **show_count** (**TextView**) vẫn sẽ lấp đầy không gian giữa các nút.

Mã giải pháp cho **Task 2**

Mã XML trong **activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<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 for showing toast -->

    <!-- TextView to display the count -->

    <!-- Button for incrementing the count -->

    <Button
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3F4BA7"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="@string/button_label_toast"
        android:textColor="@color/white"
        android:textSize="20sp" />
```

```
<TextView  
    android:id="@+id/text_count"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:background="#FFEB3B"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="8dp"  
    android:layout_marginEnd="8dp"  
    android:layout_marginBottom="8dp"  
    android:gravity="center"  
    android:text="@string/count_initial_value"  
    android:textAllCaps="true"  
    android:textColor="#3F51B5"  
    android:textSize="120dp" />
```

```
<Button  
    android:id="@+id/button_count"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="#3F51B5"  
    android:text="@string/button_label_count"  
    android:layout_marginStart="8dp"  
    android:layout_marginBottom="8dp"  
    android:layout_marginEnd="8dp"  
    android:textColor="@color/white"  
    android:textSize="20dp" />
```

```
</LinearLayout>
```

Task 3: Thay đổi bố cục sang RelativeLayout

RelativeLayout là một nhóm View trong đó mỗi View được định vị và căn chỉnh dựa trên các View khác trong cùng nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục bằng **RelativeLayout**.

3.1 Thay đổi LinearLayout thành RelativeLayout

Một cách dễ dàng để thay đổi **LinearLayout** thành **RelativeLayout** là chỉnh sửa trực tiếp trong tab **Text** của tệp XML.

1. Mở tệp **activity_main.xml**, sau đó nhấp vào tab **Text** ở cuối trình chỉnh sửa để xem mã XML.
2. Thay đổi `<LinearLayout` ở đầu tệp thành `<RelativeLayout`, để câu lệnh trông như sau:

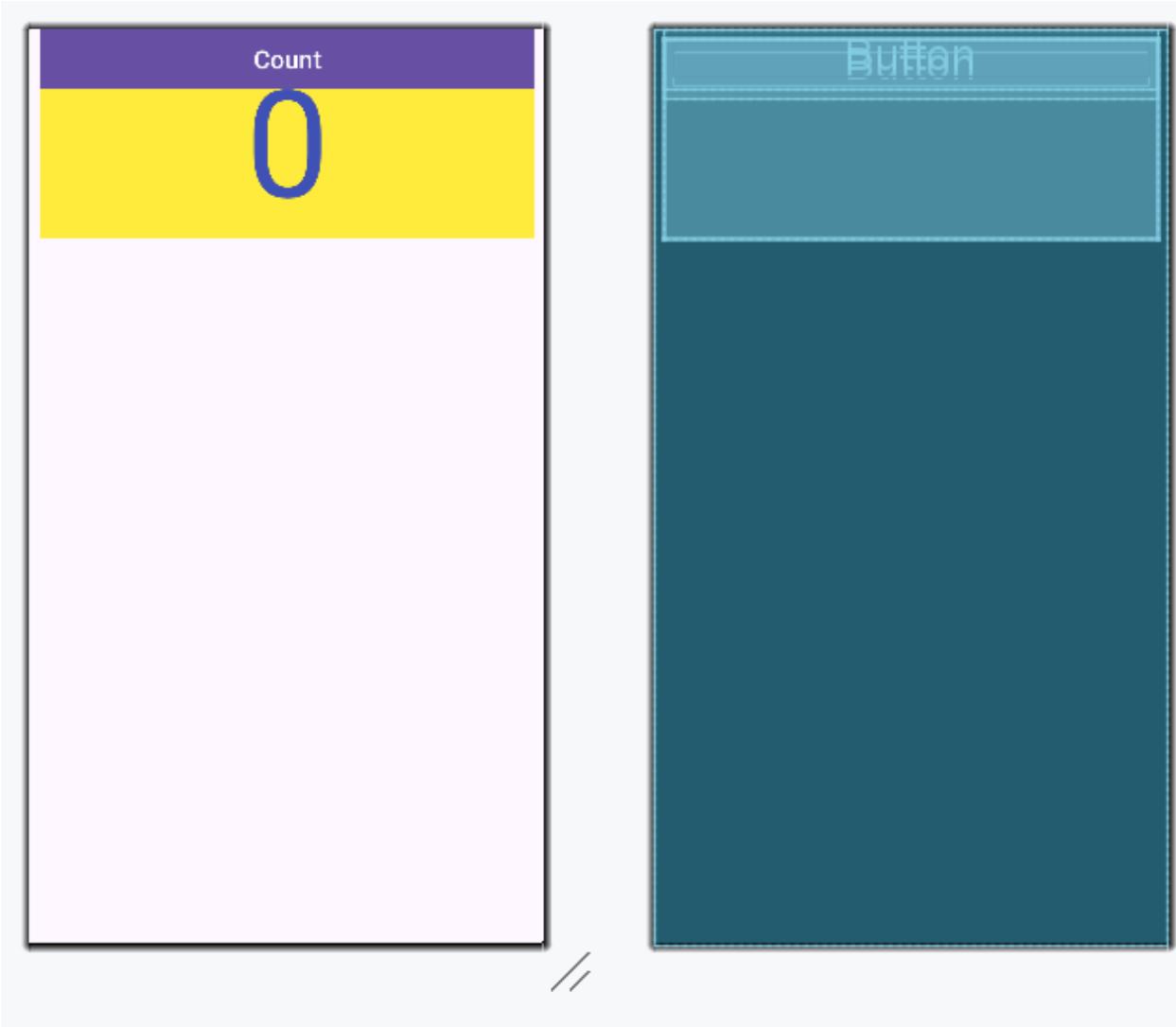
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Cuộn xuống để đảm bảo thẻ kết thúc `</LinearLayout>` cũng được thay đổi thành `</RelativeLayout>`; nếu chưa, hãy chỉnh sửa thủ công.

3.2 Sắp xếp lại các View trong RelativeLayout

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

1. Nhấp vào tab **Preview** bên cạnh trình chỉnh sửa (nếu chưa được chọn) để xem bản xem trước bối cảnh, bây giờ trông giống như hình bên dưới.



Lưu ý: Khi thay đổi sang **RelativeLayout**, trình chỉnh sửa bố cục cũng thay đổi một số thuộc tính của View. Ví dụ:

- Nút **Count** (button_count) **đè lên** nút **Toast** (button_toast), khiến bạn không nhìn thấy nút **Toast**.
 - Phần trên của **TextView** (show_count) **đè lên** các nút **Button**.
2. Thêm thuộc tính `android:layout_below` vào nút **button_count** để đặt nút này ngay bên dưới **TextView** (show_count). Đây là một trong nhiều thuộc tính giúp định vị View trong **RelativeLayout**, cho phép bạn đặt View dựa trên View khác.

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

3. Thêm thuộc tính android:layout_centerHorizontal vào **button_count** để căn giữa nút này theo chiều ngang trong **RelativeLayout** (là nhóm cha của View này).

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

→ Mã XML đầy đủ cho nút **button_count** như sau:

```
<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#3F51B5"
    android:text="Count"
    android:layout_below="@+id/show_count"
    android:layout_marginStart="8dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:textColor="@color/white"
    android:textSize="20dp"
    android:layout_centerHorizontal="true"/>
```

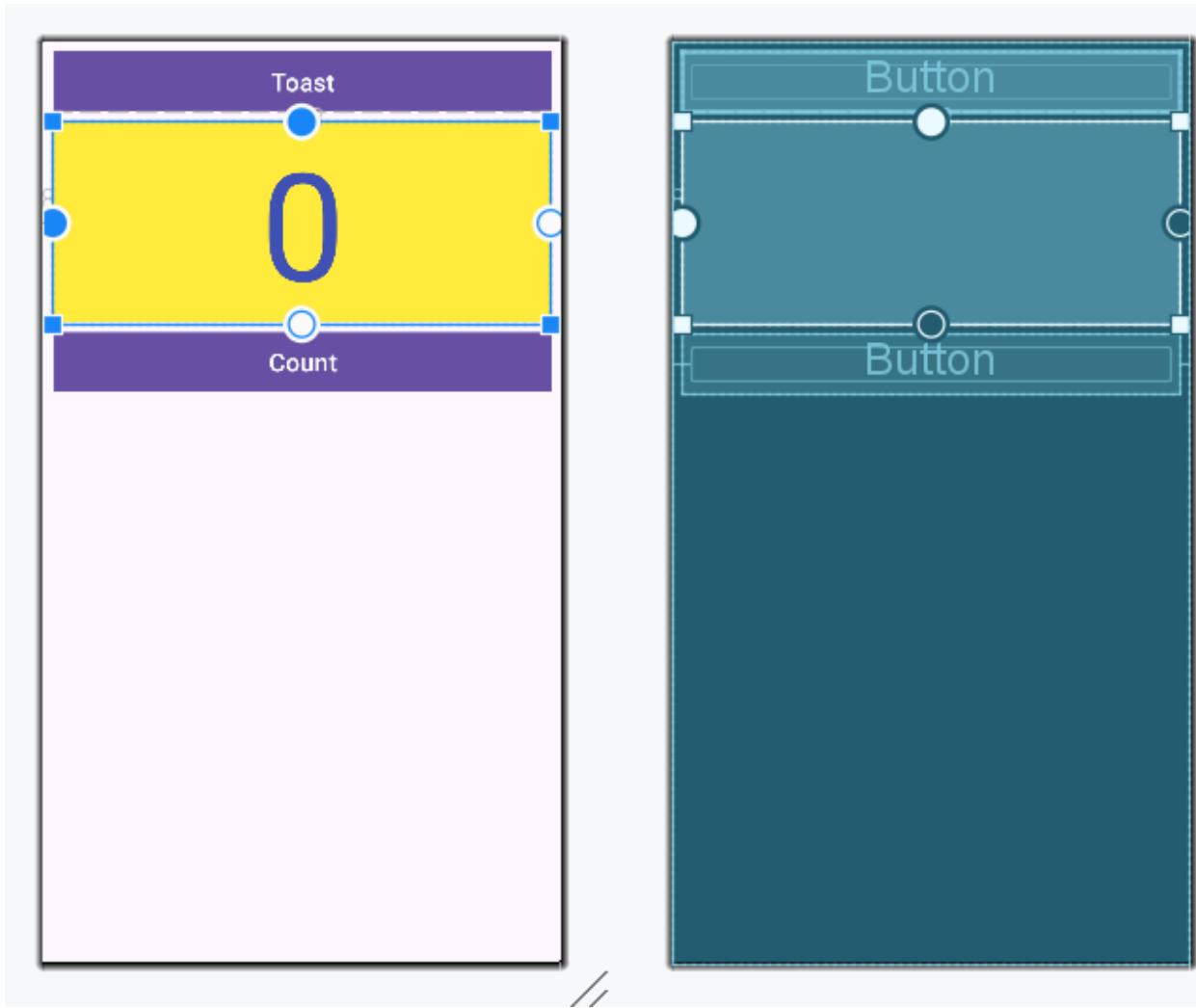
4. Thêm các thuộc tính sau vào **show_count** (TextView):

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

- android:layout_alignParentLeft → Căn View về phía **trái** của **RelativeLayout** (nhóm cha).
- android:layout_alignParentStart → Căn View theo **cạnh bắt đầu** của nhóm cha.
 - Thuộc tính này giúp hỗ trợ các thiết bị sử dụng ngôn ngữ **phải sang trái (RTL)**.
 - Nếu ứng dụng chạy trên thiết bị có ngôn ngữ **trái sang phải (LTR)**, nó sẽ căn về bên **trái**.

- Nếu chạy trên thiết bị có ngôn ngữ **phải sang trái (RTL)**, nó sẽ căn về bên **phải**.
5. Xóa thuộc tính `android:layout_weight="1"` của **show_count** (`TextView`), vì thuộc tính này không có tác dụng trong **RelativeLayout**.

Bản xem trước bố cục bây giờ trông giống như hình bên dưới.



Mẹo:

RelativeLayout giúp bạn dễ dàng sắp xếp nhanh chóng các phần tử UI trong bố cục. Để tìm hiểu thêm về cách định vị **View** trong **RelativeLayout**, hãy tham khảo tài liệu "**Positioning Views**" trong chủ đề "**RelativeLayout**" của **API Guide**.

Mã giải pháp cho Task 3

Mã XML trong **activity_main.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<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 for showing toast -->

    <!-- TextView to display the count -->

    <!-- Button for incrementing the count -->

    <Button
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3F4BA7"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="@string/button_label_toast"
        android:textColor="@color/white"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/show_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="#FFEB3B"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
```

```
    android:layout_marginBottom="8dp"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:layout_below="@+id/button_toast"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:textAllCaps="true"
    android:textColor="#3F51B5"
    android:textSize="120dp" />
```

```
<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#3F51B5"
    android:text="@string/button_label_count"
    android:layout_below="@+id/show_count"
    android:layout_marginStart="8dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:textColor="@color/white"
    android:textSize="20dp"
    android:layout_centerHorizontal="true"/>

</RelativeLayout>
```

Tóm tắt

Sử dụng trình chỉnh sửa bố cục để xem trước và tạo các biến thể

- Để xem trước bố cục ứng dụng theo hướng **ngang**, nhấp vào nút **Orientation in Editor** trên thanh công cụ và chọn **Switch to Landscape**.
→ Chọn **Switch to Portrait** để quay lại hướng dọc.
- Để tạo một biến thể bố cục khác cho **hướng ngang**, nhấp vào nút **Orientation in Editor** và chọn **Create Landscape Variation**.
→ Một cửa sổ chỉnh sửa mới sẽ mở với tab **land/activity_main.xml**, hiển thị bố cục theo hướng ngang.

- Để xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thật hoặc trình giả lập, nhấp vào nút **Device in Editor** trên thanh công cụ và chọn một thiết bị.
- Để tạo một biến thể bố cục khác dành cho **máy tính bảng (màn hình lớn hơn)**, nhấp vào nút **Orientation in Editor** và chọn **Create layout x-large Variation**.
→ Một cửa sổ chỉnh sửa mới sẽ mở với tab **xlarge/activity_main.xml**, hiển thị bố cục cho thiết bị có màn hình lớn.

Sử dụng ConstraintLayout

- Để xóa tất cả ràng buộc trong **ConstraintLayout**, nhấp vào nút **Clear All Constraints** trên thanh công cụ.
- Bạn có thể căn chỉnh một phần tử giao diện người dùng chứa văn bản (chẳng hạn như **TextView** hoặc **Button**) với một phần tử khác bằng **ràng buộc đường cơ sở (baseline constraint)**.
- Để tạo ràng buộc đường cơ sở, di chuột qua phần tử giao diện người dùng cho đến khi nút **baseline constraint** xuất hiện bên dưới phần tử.
- Nút **pack** trên thanh công cụ cung cấp tùy chọn để sắp xếp hoặc mở rộng các phần tử UI đã chọn.
→ Bạn có thể sử dụng nó để **căn đều các nút (Button) theo chiều ngang trong bố cục**.

Sử dụng LinearLayout

- LinearLayout** là một **ViewGroup** sắp xếp các **View** theo hàng **ngang** hoặc **dọc**.
- Một **LinearLayout** cần có các thuộc tính sau:
 - layout_width
 - layout_height
 - orientation
- match_parent** cho layout_width hoặc layout_height:
 - Mở rộng View để lấp đầy phần tử cha.
 - Nếu **LinearLayout** là phần tử gốc, nó sẽ mở rộng theo kích thước màn hình.
- wrap_content** cho layout_width hoặc layout_height:
 - Thu nhỏ View để vừa với nội dung của nó.
 - Nếu không có nội dung, View sẽ trở nên vô hình.

- **Số dp (density-independent pixels) cố định** cho layout_width hoặc layout_height:
 - Xác định kích thước cố định, được điều chỉnh theo mật độ màn hình.
 - Ví dụ: 16dp có nghĩa là 16 pixel độc lập với mật độ.
- **Hướng (orientation) của LinearLayout:**
 - horizontal → Sắp xếp phần tử từ **trái sang phải**.
 - vertical → Sắp xếp phần tử từ **trên xuống dưới**.
- **Sử dụng gravity và weight để kiểm soát bố cục:**
 - **android:gravity** → Căn chỉnh nội dung bên trong View.
 - **android:layout_weight** → Xác định tỷ lệ không gian bổ sung được phân bổ cho View.
 - Nếu chỉ có một View có thuộc tính này, nó sẽ nhận toàn bộ không gian trống.
 - Nếu có nhiều View, không gian sẽ được chia theo tỷ lệ.
 - Ví dụ: Nếu hai **Button** có weight="1" và một **TextView** có weight="2" (tổng cộng 4), mỗi **Button** sẽ nhận $\frac{1}{4}$ không gian và **TextView** sẽ nhận $\frac{1}{2}$.

Sử dụng RelativeLayout

- **RelativeLayout** là một **ViewGroup**, trong đó mỗi **View** được căn chỉnh tương đối với các **View** khác trong cùng nhóm.
- Các thuộc tính quan trọng:
 - **android:layout_alignParentTop** → Căn View lên **đỉnh** của phần tử cha.
 - **android:layout_alignParentLeft** → Căn View về **bên trái** của phần tử cha.
 - **android:layout_alignParentStart** → Căn View theo **cạnh bắt đầu** của phần tử cha.
 - **Start** là **bên trái** nếu ngôn ngữ từ **trái sang phải (LTR)**.
 - **Start** là **bên phải** nếu ngôn ngữ từ **phải sang trái (RTL)**.

Tài liệu liên quan

Tài liệu về các khái niệm liên quan có trong:

1.2: Bố cục và tài nguyên cho UI (Layouts and resources for the UI).

Tìm hiểu thêm:

Tài liệu Android Studio:

- Giới thiệu về Android Studio
- Tạo biểu tượng ứng dụng với Image Asset Studio

Tài liệu Android Developer:

- Layouts
- Xây dựng UI với Layout Editor
- Xây dựng giao diện đáp ứng với ConstraintLayout
- LinearLayout
- RelativeLayout
- View
- Button
- TextView
- Hỗ trợ nhiều mật độ pixel khác nhau

Khác:

- Codelabs: **Sử dụng ConstraintLayout để thiết kế giao diện Android**
- **Từ vựng và khái niệm quan trọng**

Bài tập về nhà

Thay đổi một ứng dụng

Mở ứng dụng **HelloToast**.

1. **Đổi tên** dự án thành **HelloConstraint**, sau đó **refactor** toàn bộ dự án thành **HelloConstraint**.
→ (Hướng dẫn về cách sao chép và refactor một dự án có trong **Phụ lục: Tiện ích**.)
2. **Chỉnh sửa bố cục** trong **activity_main.xml** để căn chỉnh các nút **Toast** và **Count** dọc theo bên trái của **TextView show_count** (hiển thị số "0").
→ Xem hình minh họa về bố cục.
3. **Thêm một nút thứ ba** có tên **Zero**, xuất hiện **giữa** các nút **Toast** và **Count**.
4. **Sắp xếp** các nút theo chiều dọc, phân bổ khoảng cách đều từ trên xuống dưới của **TextView show_count**.
5. **Đặt nền ban đầu** của nút **Zero** thành **màu xám**.
6. **Đảm bảo rằng** nút **Zero** xuất hiện **trong các bố cục khác**, bao gồm:
 - **activity_main.xml (land)** (giao diện ngang).
 - **activity_main.xml (xlarge)** (giao diện trên màn hình máy tính bảng).

7. **Cập nhật xử lý sự kiện của nút Zero** để khi nhấn vào, nó sẽ đặt giá trị trong **show_count** về **0**.
8. **Cập nhật xử lý sự kiện của nút Count:**
 - Khi nhấn, **nút Count** sẽ **đổi màu nền** tùy thuộc vào số hiện tại là **chẵn hay lẻ**.
 - **Gợi ý:** Không sử dụng **findViewById** để tìm nút Count. Hãy tìm cách khác!
 - Có thể sử dụng các hằng số trong lớp **Color** để đặt màu nền.
9. **Cập nhật xử lý sự kiện của nút Count** để thay đổi màu nền của nút **Zero** thành một màu khác ngoài **xám**, nhằm thể hiện rằng nút Zero đã được kích hoạt.
 - **Gợi ý:** Lần này, có thể sử dụng **findViewById** để tìm nút Zero.
10. **Cập nhật xử lý sự kiện của nút Zero** để **đặt lại màu nền** của chính nó về **xám** khi số đếm bằng 0.

Trả lời các câu hỏi

Câu hỏi 1:

Hai thuộc tính **ràng buộc bố cục (layout constraint)** nào trên **nút Zero** giúp nó được **căn giữa theo chiều dọc**, nằm cách đều hai nút còn lại? (Chọn 2 đáp án)

app:layout_constraintBottom_toTopOf="@+id/button_count"
app:layout_constraintTop_toBottomOf="@+id/button_toast"

Câu hỏi 2:

Thuộc tính ràng buộc bố cục nào trên **nút Zero** giúp nó **căn chỉnh ngang hàng** với hai nút còn lại?

app:layout_constraintLeft_toLeftOf="parent"

Câu hỏi 3:

Đâu là chữ ký (signature) đúng của một phương thức được dùng với thuộc tính **android:onClick** trong XML?

public void callMethod(View view)

Câu hỏi 4:

Trong trình xử lý sự kiện của nút **Count**, phương pháp nào **hiệu quả hơn** để thay đổi màu nền của nút?

Sử dụng tham số view được truyền vào trình xử lý sự kiện, với setBackgroundColor():

1.3) Trình chỉnh sửa bộ cục

Giới thiệu

Lớp **TextView** là một lớp con của lớp **View**, dùng để hiển thị văn bản trên màn hình. Bạn có thể kiểm soát cách văn bản xuất hiện bằng cách sử dụng các thuộc tính của **TextView** trong tệp bố cục XML. Bài thực hành này hướng dẫn cách làm việc với nhiều phần tử **TextView**, bao gồm cả một phần tử mà người dùng có thể cuộn nội dung của nó theo chiều dọc.

Nếu có nhiều thông tin hơn so với khả năng hiển thị của màn hình thiết bị, bạn có thể tạo một **chế độ xem cuộn** để người dùng có thể cuộn theo chiều dọc bằng cách vuốt lên hoặc xuống, hoặc cuộn ngang bằng cách vuốt sang phải hoặc trái.

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

Lớp **ScrollView** cung cấp bộ cục cho chế độ xem cuộn. **ScrollView** là một lớp con của **FrameLayout**. Hãy chỉ đặt **một** phần tử con bên trong nó—phần tử con này chứa toàn bộ nội dung cần cuộn. Phần tử con này có thể là một **ViewGroup** (chẳng hạn như **LinearLayout**) chứa các thành phần giao diện người dùng.

Các bộ cục phức tạp có thể gấp ván để về hiệu suất với các phần tử con như hình ảnh. Một lựa chọn tốt cho một **View** bên trong **ScrollView** là **LinearLayout** được sắp xếp theo chiều dọc, hiển thị các mục mà người dùng có thể cuộn qua (chẳng hạn như các phần tử **TextView**).

Với **ScrollView**, tất cả các thành phần giao diện người dùng đều được tải vào bộ nhớ và có trong cây cấu trúc xem ngay cả khi chúng không hiển thị trên màn hình. Điều này làm cho **ScrollView** lý tưởng để cuộn qua các trang văn bản tự do một cách mượt mà, vì văn bản đã được tải sẵn vào bộ nhớ. Tuy nhiên, **ScrollView** có thể tiêu tốn nhiều bộ nhớ, ảnh hưởng đến hiệu suất của phần còn lại của ứng dụng. Để hiển thị danh sách dài các mục mà người dùng có thể

thêm, xóa hoặc chỉnh sửa, hãy cân nhắc sử dụng **RecyclerView**, được mô tả trong bài học khác.

Những gì bạn cần biết trước

Bạn cần có khả năng:

- Tạo ứng dụng Hello World bằng Android Studio.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị.
- Triển khai **TextView** trong bố cục của ứng dụng.
- Tạo và sử dụng tài nguyên chuỗi văn bản.

Những gì bạn sẽ học

- Cách sử dụng mã XML để thêm nhiều phần tử **TextView**.
- Cách sử dụng mã XML để định nghĩa 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 thêm liên kết web vào văn bản.

Những gì bạn sẽ làm

- Tạo ứng dụng ScrollingText.
- Thay đổi **ConstraintLayout ViewGroup** thành **RelativeLayout**.
- Thêm hai phần tử **TextView** cho tiêu đề bài viết và tiêu đề phụ.
- Sử dụng kiểu dáng và màu sắc **TextAppearance** cho tiêu đề bài viết và tiêu đề phụ.

Sử dụng thẻ HTML trong chuỗi văn bản để kiểm soát định dạng:

- Sử dụng thuộc tính **lineSpacingExtra** để thêm khoảng cách giữa các dòng, cải thiện khả năng đọc.
- Thêm **ScrollView** vào bố cục để cho phép cuộn qua phần tử **TextView**.
- Sử dụng thuộc tính **autoLink** để các URL trong văn bản trở nên hoạt động và có thể nhấp vào.

Tổng quan về ứng dụng

Ứng dụng **Scrolling Text** minh họa cách sử dụng thành phần giao diện người dùng **ScrollView**. **ScrollView** là một **ViewGroup**, 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 hoặc xuống. Thanh cuộn xuất hiện ở lề bên phải. Ứng dụng này cho thấy cách bạn có thể sử dụng văn bản được định dạng với các thẻ HTML tối thiểu để thiết lập văn bản in đậm hoặc in nghiêng, và sử dụng ký tự xuống dòng để ngăn cách các đoạn văn. Bạn cũng có thể thêm các liên kết web hoạt động trong văn bản.

Trong hình trên, các yếu tố sau xuất hiện:

1. Một liên kết web hoạt động được nhúng trong văn bản tự do.
2. Thanh cuộn xuất hiện khi cuộn 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 để hiển thị tiêu đề và tiêu đề phụ của bài viết, đồng thời thay đổi phần tử **TextView** "Hello World" hiện tại để hiển thị một bài viết dài. Hình minh họa bên dưới là sơ đồ của bố cục.

Bạn sẽ thực hiện tất cả các thay đổi này trong mã XML và tệp strings.xml. Bạn sẽ chỉnh sửa mã XML cho bố cục trong tab **Text**, mà bạn có thể hiển thị bằng cách nhấp vào tab **Text**, thay vì nhấp vào tab **Design** để mở giao diện thiết kế. Một số thay đổi đối với các phần tử giao diện người dùng (UI) và thuộc tính sẽ dễ dàng thực hiện hơn trực tiếp trong tab **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 thêm các phần tử **TextView**, đồng thời sử dụng các thuộc tính **TextView** để tạo kiểu cho văn bản và nền.

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

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

Thuộc tính	Giá trị
Application Name	Scrolling Text
Company Name	android.example.com (hoặc tên miền của bạn)
Phone and Tablet Minimum SDK	API15: Android 4.0.3 IceCreamSandwich
Template	Empty Activity
Generate Layout File checkbox	Selected
Backwards Compatibility (AppCompat) checkbox	Selected

2. Trong thư mục **app > res > layout** ở ngăn **Project > Android**, mở tệp **activity_main.xml** và nhấp vào tab **Text** để xem mã XML.
Ở phần trên cùng, hoặc **gốc**, của cấu trúc phân cấp View là **ViewGroup ConstraintLayout**:
`<androidx.constraintlayout.widget.ConstraintLayout>`
3. Thay đổi **ViewGroup** này thành **RelativeLayout**. Dòng mã thứ hai bây giờ sẽ trông giống như sau:

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

RelativeLayout cho phép bạn đặt các phần tử giao diện người dùng (UI) tương đối với nhau hoặc tương đối với chính **RelativeLayout** cha.
Phần tử **TextView** "Hello World" mặc định được tạo bởi mẫu Bố cục Trống (Empty Layout) vẫn có các thuộc tính ràng buộc (chẳng hạn như `app:layout_constraintBottom_toBottomOf="parent"`). Đừng lo lắng—you sẽ xóa chúng trong bước tiếp theo.

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

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

Khôi mã XML ở phần đầu bây giờ trông như thế này:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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 khối **TextView** sẽ xuất hiện, kết thúc bằng `</>`, và hiển

thì các thuộc tính layout_width và layout_height, là những thuộc tính bắt buộc đối với TextView.

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

TextView #1 attribute	Value
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:id	"@+id/article_heading"
android:background	"@color/colorPrimary"
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 thuộc tính android:text với chuỗi cứng "Article Title" trong TextView để tạo một mục trong **strings.xml**.

Đặt con trỏ vào chuỗi cứng, nhấn Alt-Enter (hoặc Option-Enter trên Mac), chọn **Extract string resource**, đảm bảo tùy chọn **Create the resource in directories** được chọn, sau đó chỉnh sửa tên tài nguyên thành **article_title**.

Tài nguyên chuỗi được mô tả chi tiết trong mục **String Resources**.

8. Trích xuất tài nguyên kích thước cho thuộc tính android:padding với chuỗi cứng "10dp" trong TextView để tạo tệp dimens.xml và thêm một mục vào đó.

Đặt con trỏ vào chuỗi cứng, nhấn Alt-Enter (hoặc Option-Enter trên Mac), chọn **Extract dimension resource**, đảm bảo tùy chọn **Create the resource in directories** được chọn, sau đó chỉnh sửa tên tài nguyên 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 đó. Thêm các thuộc tính sau cho TextView.

TextView #2 Attribute	Value
layout_width	"match_parent"

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 được 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 cứng "Article Subtitle" của thuộc tính android:text 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_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"

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

TextView Attribute	Value
android:id	"@+id/article"
android:layout_below	"@+id/article_subheading"
android:lineSpacingExtra	"5sp"
android:padding	"@dimen/padding_regular"
android:text	Change to "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 chọn **Code > Reformat Code**.

Đây là một thực hành tốt để định dạng và căn chỉnh mã của bạn sao cho dễ hiểu hơn đối với bạn và người khác.

1.2 Thêm nội dung bài viết

Trong một ứng dụng thực tế truy cập các bài báo từ tạp chí hoặc báo, các bài viết hiển thị có thể sẽ đến từ một nguồn trực tuyến thông qua nhà cung cấp nội dung hoặc có thể được lưu sẵn trong cơ sở dữ liệu trên thiết bị.

Trong bài thực hành này, bạn sẽ tạo bài viết dưới dạng một chuỗi dài trong tệp 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, hoặc mở tệp **strings.xml** của ứng dụng **ScrollingText** đã hoàn thiện.
3. Nhập giá trị cho các chuỗi article_title và article_subtitle với tiêu đề và phụ đề tùy ý, hoặc sử dụng các giá trị trong tệp strings.xml của ứng dụng **ScrollingText** đã hoàn thiện. Đảm bảo rằng các giá trị chuỗi là văn bản trên một dòng và không có thẻ HTML hoặc nhiều dòng.
4. Nhập hoặc sao chép-dán văn bản cho chuỗi article_text.

Bạn có thể sử dụng văn bản trong tệp văn bản của bạn, hoặc sử dụng văn bản được cung cấp cho chuỗi article_text trong tệp strings.xml của ứng dụng **ScrollingText** đã hoàn thiện. Yêu cầu duy nhất cho nhiệm vụ này là văn bản phải đủ dài để không hiển thị hết trên màn hình.

Lưu ý các điểm sau (tham khảo hình minh họa bên dưới để có ví dụ):

- Khi bạn nhập hoặc dán văn bản vào tệp strings.xml, các dòng văn bản sẽ không tự động xuống dòng mà sẽ kéo dài vượt ra ngoài lề phải. Đây là hành vi đúng—mỗi dòng văn bản mới bắt đầu từ lề trái sẽ đại diện cho toàn bộ một đoạn văn. Nếu bạn muốn văn bản trong strings.xml được xuống dòng, bạn có thể nhấn **Return** để thêm dấu kết thúc dòng cứng, hoặc định dạng văn bản trước trong một trình soạn thảo văn bản với các dấu kết thúc dòng cứng.

- Nhập \n để đại diện cho kết thúc một dòng và thêm một \n khác để đại diện cho một dòng trống. Bạn cần thêm ký tự kết thúc dòng để các đoạn văn không nối liền với nhau.
- Nếu bạn có dấu nháy đơn () trong văn bản, bạn phải thoát nó bằng cách thêm một dấu gạch chéo ngược (\) phía trước. Nếu bạn có dấu nháy kép trong văn bản, bạn cũng phải thoát nó ("). Bạn cũng phải thoát bất kỳ ký tự không thuộc ASCII nào khác. Xem phần **Định dạng và phong cách** của Tài nguyên chuỗi để biết thêm chi tiết.
- Thêm thẻ HTML và xung quanh các từ cần in đậm.
- Thêm thẻ HTML <i> và </i> xung quanh các từ cần in nghiêng. Nếu bạn sử dụng dấu nháy đơn cong trong một cụm từ in nghiêng, hãy thay 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>... từ ...</i>.
- Các thẻ HTML khác sẽ bị bỏ qua.
- Bao toàn bộ văn bản trong <string name="article_text"> </string> trong tệp strings.xml.
- Bao gồm một liên kết web để kiểm tra, chẳng hạn như www.google.com. (Ví dụ dưới đây sử dụng www.rockument.com.) Đừng sử dụng thẻ HTML, vì bất kỳ thẻ HTML nào ngoài thẻ in đậm và in nghiêng sẽ bị bỏ qua và hiển thị dưới dạng văn bản, điều này không phải là điều bạn muốn.



```

<resources>
    <string name="app_name">Scrolling Text</string>
    <string name="article_title">Beatles Anthology Vol. 1</string>
    <string name="article_subtitle">Behind That Locked Door: Beatles Rarities!</string>
    <string name="article_text">
        <![CDATA[
            In a vault deep inside Abbey Road Studios in London – protected by an unmarked, triple-locked door – lies one of the rarest collections of Be
            For more information, see The Beatles Time Capsule at www.rockument.com.<br /><br />
            This volume starts with the first new Beatle song, "Free as a Bird"; (based on a John Lennon demo, found only on <i>The Lost Tapes</i>).
            <b>Highlights include:</b><br />
            <ul>
                <li>&#8226; Cry for a Shadow: – Many a Beatle fanatic started down the outtake road, like I did, with a first listen to this track.</li>
                <li>&#8226; My Bonnie: and #39;t She Sweet: – At the same session, the Beatles played "My Bonnie"; (the first single
                    <li>&#8226; Searching: – A Jerry Leiber &#8211; Mike Stoller comedy song that was a hit for the Coasters in 1957, and a popular part of i
                    <li>&#8226; Love Me Do: – An early version of the song, played a bit slower and with more of a blues feeling, and a cool harmonica part.<
                    <li>&#8226; She Loves You: – Till There Was You – Twist and Shout: – Live at the Princess Wales Theatre by Leicester Square.</li>
                    <li>&#8226; Leave My Kitten Alone: – One of the lost Beatle songs recorded during the "Beatles For Sale"; sessions but never rel
                    <li>&#8226; One After 909: – A song recorded for the <i>Let It Be</i> album was actually worked on way back in the beginning.</li>
            </ul>
        ]]>
    </string>
    <color name="colorPrimary">#6200EE</color>
</resources>

```

1.3 Chạy ứng dụng

Chạy ứng dụng. Bài viết xuất hiện, nhưng người dùng không thể cuộn bài viết vì bạn chưa thêm một ScrollView (việc này bạn sẽ thực hiện trong nhiệm vụ tiếp theo). Lưu ý rằng khi nhấn vào một liên kết web, hiện tại sẽ không có tác dụng gì. Bạn cũng sẽ sửa điều đó trong nhiệm vụ tiếp theo.

Scrolling Text

Beetles Anthology Vol. 1

Behind That Locked Door: Beatles Rarities!

In a vault deep inside Abbey Road Studios in London – protected by an unmarked, triple-locked door – lies one of the rarest collections of Beatles recordings.
For more information, see The Beatles Time Capsule at www.rockument.com.

This volume starts with the first new Beatle song, "Free as a Bird" (based on a John Lennon demo, found only on "The Lost Tapes").
Highlights include:

- "Cry for a Shadow" – Many a Beatle fanatic started down the outtake road, like I did,

Mã giải pháp nhiệm vụ 1

Tệp bô cục activity_main.xml trông như sau:

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

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

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

    <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>
```

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

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

Ngoài ra, TextView tự nó không thể cho phép người dùng cuộn văn bản bài viết để xem toàn bộ nội dung. Bạn sẽ thêm một ViewGroup mới gọi là ScrollView vào bộ cục XML để làm cho TextView có thể cuộn được.

2.1 Thêm thuộc tính autoLink để kích hoạt liên kết web

Thêm thuộc tính android:autoLink="web" vào TextView của bài viết. Mã XML cho TextView này bây giờ sẽ trông 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="@dimen/line_spacing"  
    android:padding="@dimen/padding_regular"  
    android:text="@string/article_text" />
```

2.2 Thêm ScrollView vào bộ cục

Để làm cho Chế độ xem (chẳng hạn như TextView) có thể cuộn được, hãy nhúng Chế độ xem bên trong ScrollView

1. Thêm một ScrollView giữa TextView có ID article_subheading và TextView có ID article. Khi bạn nhập <ScrollView, Android Studio sẽ tự động thêm </ScrollView> ở cuối, và hiển thị các gợi ý cho thuộc tính android:layout_width và android:layout_height.
2. Chọn **wrap_content** từ danh sách gợi ý cho cả hai thuộc tính.
Mã cho hai phần tử TextView và ScrollView bây giờ trông 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="@dimen/padding_regular"
    android:text="@string/article_subtitle"
    android:textAppearance=
        "@android:style/TextAppearance.DeviceDefault" />
<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"
    android:autoLink="web"
    android:layout_below="@+id/article_subheading"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />

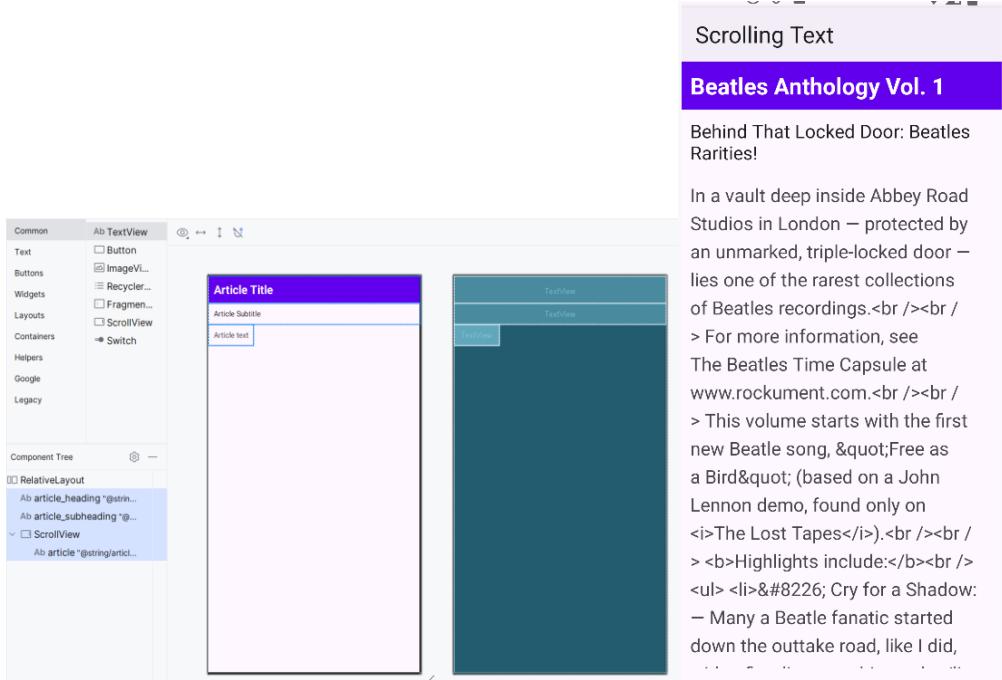
```

3. Di chuyển đoạn mã kết thúc </ScrollView> xuống sau TextView có ID article để các thuộc tính của article nằm hoàn toàn bên trong ScrollView.
4. Xóa thuộc tính sau từ TextView có ID article và thêm nó vào ScrollView: `android:layout_below="@+id/article_subheading"`

Bài viết nằm bên trong phần tử ScrollView.

5. Chọn **Code > Reformat Code** để định dạng lại mã XML, sao cho TextView của bài viết xuất hiện được thụt vào bên trong đoạn mã <ScrollView>.
6. Nhấp vào tab **Preview** ở phía bên phải của trình chỉnh sửa bô cục để xem trước bô cục.

Bô cục giờ đây trông giống như phần bên phải của hình minh họa sau:



2.3 Chạy ứng dụng

Để kiểm tra cách văn bản cuộn:

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

Vuốt lên và xuống để cuộn qua 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 tự động biến bất kỳ URL nào có thể nhận dạng được trong TextView (chẳng hạn như www.rockument.com) thành một liên kết web.

- Xoay thiết bị hoặc trình giả lập khi ứng dụng đang chạy.

Quan sát cách chế độ cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.

- Chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng.

Quan sát cách chế độ cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.

The screenshot shows a smartphone displaying a news article. At the top, there are standard Android status icons: signal strength, battery level, and connectivity. Below the icons, the title "Scrolling Text" is visible. The main content area has a purple header bar containing the text "Beatles Anthology Vol. 1". The main body of the text discusses the "Behind That Locked Door: Beatles Rarities!" section of the anthology. It mentions a vault at Abbey Road Studios containing rare Beatles recordings. It also provides links to "The Beatles Time Capsule" at www.rockument.com and highlights from "The Lost Tapes". A specific track, "Cry for a Shadow", is mentioned as a notable highlight.

Behind That Locked Door: Beatles Rarities!

In a vault deep inside Abbey Road Studios in London – protected by an unmarked, triple-locked door – lies one of the rarest collections of Beatles recordings.
> For more information, see
The Beatles Time Capsule at
www.rockument.com.
> This volume starts with the first new Beatle song, "Free as a Bird" (based on a John Lennon demo, found only on *The Lost Tapes*).
> **Highlights include:**

- > Cry for a Shadow:
– Many a Beatle fanatic started down the outtake road, like I did,
with a first listen to this track

Trong hình trên, các yếu tố sau đây xuất hiện:

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

Mã giải pháp nhiệm vụ 2

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

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.scrollingtext.MainActivity"
    android:id="@+id/main">

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

    <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:text="@string/article_subtitle"
    android:textAppearance=
        "@android:style/TextAppearance.DeviceDefault" />
<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/article_subheading">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/article"
        android:autoLink="web"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_text" />
</ScrollView>

</RelativeLayout>
```

Nhiệm vụ 3: Cuộn nhiều phần tử

Như đã đề cập trước đó, một ScrollView chỉ có thể chứa một View con (ví dụ như TextView bạn đã tạo cho bài viết). 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, chẳng hạn như LinearLayout, vào trong ScrollView, từ đó cho phép cuộn toàn bộ nội dung bên trong LinearLayout.

Ví dụ, nếu bạn muốn phần tiêu đề phụ của bài viết cùng cuộn với nội dung bài viết, bạn cần thêm một LinearLayout vào bên trong ScrollView, sau đó di chuyển tiêu đề phụ và bài viết vào trong LinearLayout.

LinearLayout sẽ trở thành phần tử con duy nhất của ScrollView, như minh họa trong hình dưới đây. Người dùng sẽ có thể cuộn toàn bộ LinearLayout, bao gồm cả tiêu đề phụ và bài viết.

3.1 Thêm một LinearLayout vào ScrollView

1. Mở tệp activity_main.xml trong dự án ứng dụng ScrollingText và chọn tab **Text** để chỉnh sửa mã XML (nếu tab này chưa được chọn).
2. Thêm một LinearLayout phía trên TextView bài viết (article) trong ScrollView. Khi bạn nhập <LinearLayout, Android Studio sẽ tự động thêm </LinearLayout> ở cuối, đồng thời gợi ý các thuộc tính android:layout_width và android:layout_height. Chọn match_parent cho chiều rộng và wrap_content cho chiều cao từ các gợi ý. Mã ở phần đầu của ScrollView bây giờ sẽ trông như sau:

```
    android:layout_width="wrap_content" , -  
<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>
```

Bạn sử dụng match_parent để chiều rộng khớp với ViewGroup cha. Bạn sử dụng wrap_content để thay đổi kích thước LinearLayout sao cho vừa đủ để bao bọc nội dung bên trong.

3. Di chuyển mã kết thúc </LinearLayout> xuống sau TextView bài viết (article) nhưng trước thẻ đóng </ScrollView>. LinearLayout bây giờ sẽ bao gồm TextView bài viết (article) và hoàn toàn nằm trong ScrollView.
4. Thêm thuộc tính android:orientation="vertical" vào LinearLayout để thiết lập hướng của nó thành dọc.
5. Chọn **Code > Reformat Code** để thuần hóa mã chính xác.

LinearLayout bây giờ trông như thế này:

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

3.2 Di chuyển các thành phần giao diện vào trong LinearLayout

Hiện tại, LinearLayout chỉ chứa một thành phần giao diện là TextView bài viết (article). Bạn muốn đưa TextView tiêu đề phụ (article_subheading) vào LinearLayout để cả hai có thể cuộn.

- Để di chuyển TextView tiêu đề phụ (article_subheading), hãy chọn đoạn mã của nó, chọn **Edit > Cut**, nhấp vào phía trên TextView bài viết (article) bên trong LinearLayout, và chọn **Edit > Paste**.
- Xóa thuộc tính android:layout_below="@+id/article_subheading" khỏi TextView tiêu đề phụ (article_subheading). Vì TextView này hiện đang nằm trong

LinearLayout, thuộc tính này sẽ xung đột với các thuộc tính của LinearLayout.

3. Thay đổi thuộc tính bố cục của ScrollView từ android:layout_below="@+id/article_subheading" thành android:layout_below="@+id/article_heading".
Bây giờ, vì 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 sẽ như sau:

```
<ScrollView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/article_heading">  
  
<LinearLayout  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
    <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 hãy chú ý rằng phần tiêu đề phụ giờ đây di chuyển cùng bài viết, trong khi phần tiêu đề chính vẫn cố định tại chỗ.

Thử thách mã hóa

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

1.4) Học cách tự giúp bản thân

Giới thiệu

Những gì bạn nên biết trước

Bạn nên có khả năng:

- Hiểu quy trình làm việc cơ bản của **Android Studio**.
- Tạo một ứng dụng từ đầu bằng mẫu Empty Activity.
- Sử dụng **trình chỉnh sửa bố cục (Layout Editor)**.

Những gì bạn sẽ học

- Nơi tìm thông tin và tài nguyên dành cho nhà phát triển để giải quyết vấn đề hoặc bổ sung tính năng.
- Cách thêm biểu tượng khởi chạy (launcher icon) cho ứng dụng của bạn.
- Cách tìm kiếm sự trợ giúp khi gặp khó khăn trong quá trình phát triển ứng dụng Android.

Những gì bạn sẽ làm

- Khám phá các tài nguyên dành cho nhà phát triển Android ở mọi trình độ, bao gồm tài liệu, hướng dẫn và cộng đồng.
- Thêm biểu tượng khởi chạy (launcher icon) cho ứng dụng HelloToast hoặc bất kỳ ứng dụng nào bạn đã tạo.

Tổng quan về ứng dụng

Bạn sẽ cải tiến ứng dụng của mình bằng cách thêm một **biểu tượng khởi chạy** – biểu tượng nhỏ đại diện cho ứng dụng của bạn trên màn hình chính hoặc trong danh sách ứng dụng. Nhiệm vụ này sẽ dạy bạn cách tùy chỉnh ứng dụng, cải thiện giao diện và tính tiện dụng của nó

Nhiệm vụ 1: Thay đổi biểu tượng khởi chạy

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

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

Thay đổi biểu tượng khởi chạy là một quy trình đơn giản từng bước, giúp bạn làm quen với các tính năng tài sản hình ảnh (image asset) của Android Studio. Trong nhiệm vụ này, bạn cũng sẽ tìm hiểu thêm về cách truy cập tài liệu chính thức của Android.

1.1 Khám phá tài liệu chính thức của Android

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

Tài liệu này chứa rất nhiều thông tin và được Google cập nhật thường xuyên.

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

Phần này nói về Material Design, một triết lý thiết kế mang tính khái niệm, định hướng cách các ứng dụng nên trông như thế nào và hoạt động ra sao trên các thiết bị di động. Hãy điều hướng qua các liên kết để tìm hiểu thêm về Material Design. Ví dụ, truy cập phần Style để tìm hiểu thêm về cách sử dụng màu sắc và các chủ đề khác.

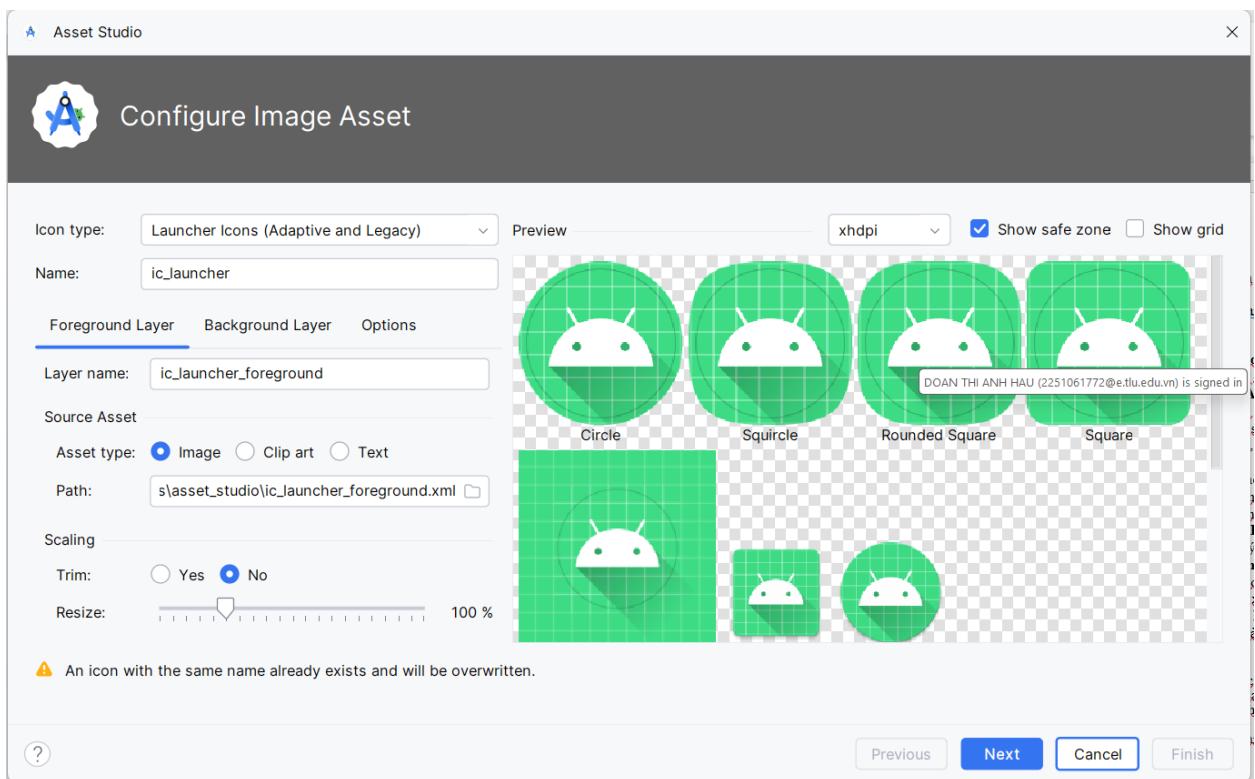
17. Truy cập developer.android.com/docs/ để tìm thông tin về API, tài liệu tham khảo, hướng dẫn, công cụ, và các mẫu mã.

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

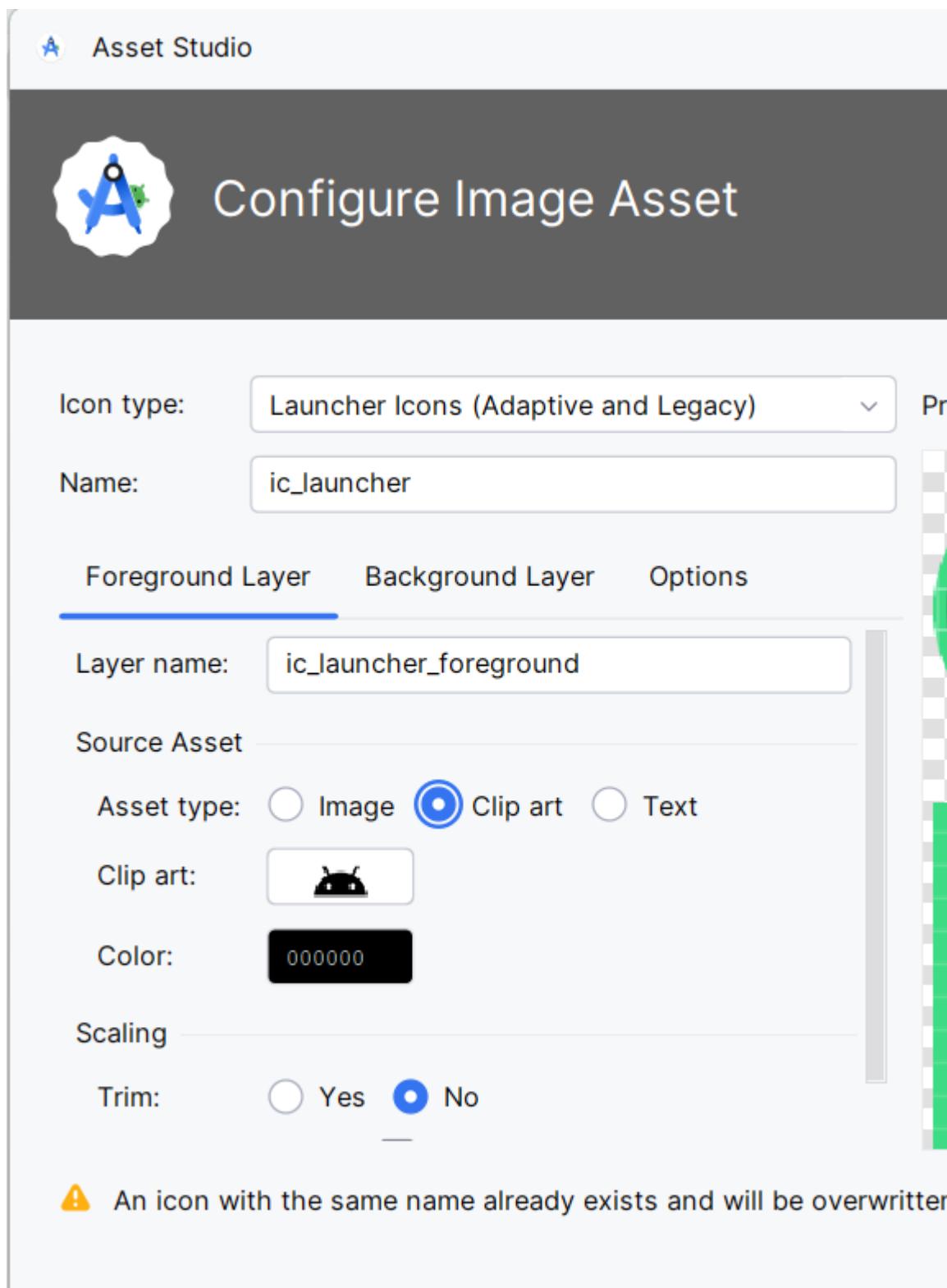
1.2 Thêm một tài nguyên hình ảnh cho biểu tượng khởi động

Để thêm một hình ảnh clip-art làm biểu tượng khởi động, 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 khung **Project > Android**, nhấp chuột phải (hoặc nhấn **Control** và nhấp chuột) vào thư mục **res** và chọn **New > Image Asset**. Cửa sổ **Configure Image Asset** sẽ xuất hiện.

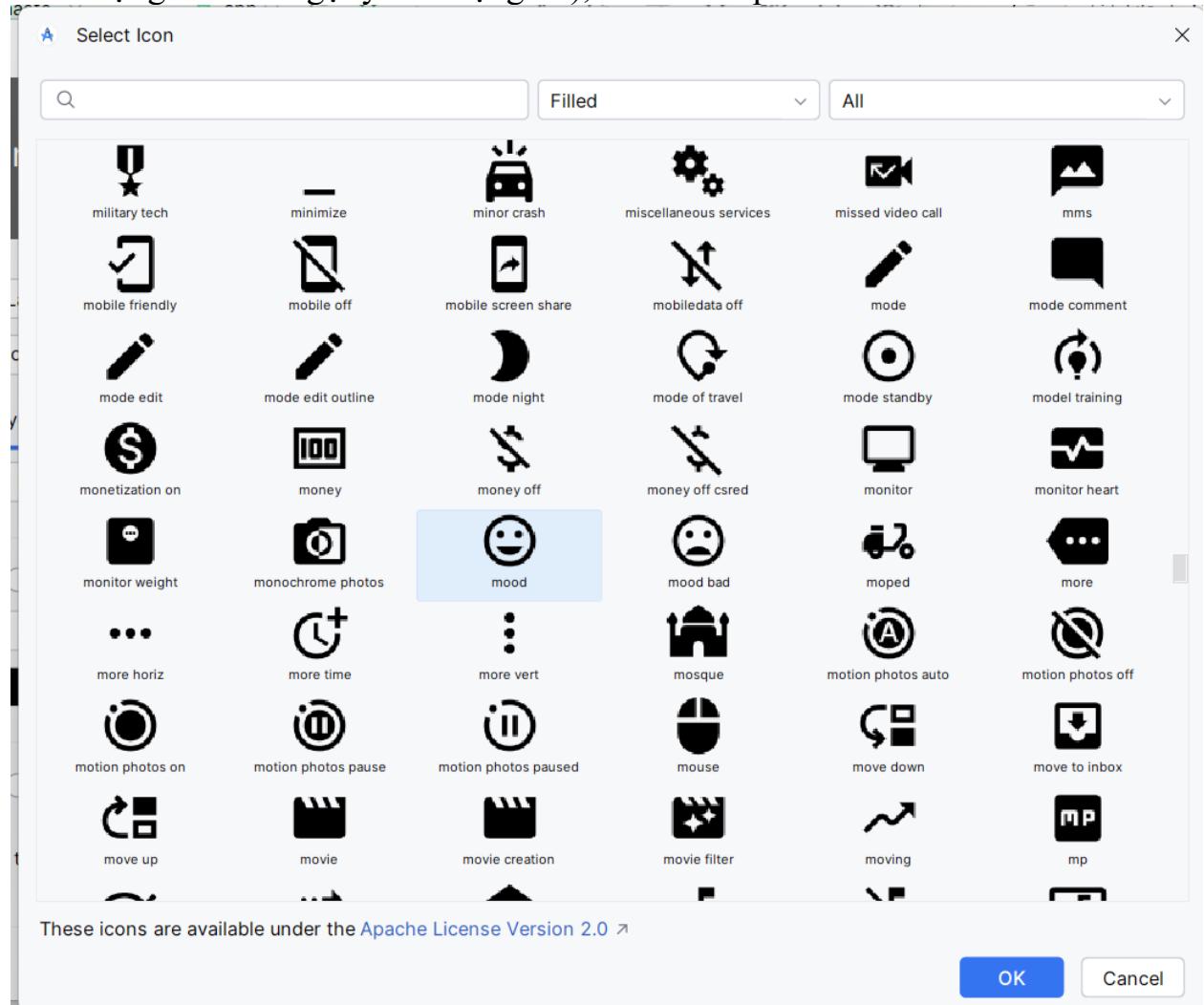


3. Trong trường **Icon Type**, chọn **Launcher Icons (Adaptive & Legacy)** nếu chưa được chọn.
4. Nhấp vào tab **Foreground Layer**, chọn **Clip Art** trong mục **Asset Type**.



- Nhập vào biểu tượng trong trường **Clip Art**. Các biểu tượng từ bộ icon Material Design sẽ xuất hiện.

6. Duyệt qua cửa sổ **Select Icon**, chọn một biểu tượng phù hợp (chẳng hạn như biểu tượng mood để gợi ý tâm trạng tốt), sau đó nhấp vào **OK**.



7. Nhấp vào tab **Background Layer**, chọn **Color** trong mục **Asset Type**, sau đó nhấp vào ô màu để chọn màu sử dụng làm lớp nền.
8. Nhấp vào tab **Legacy** và xem lại các cài đặt mặc định. Xác nhận rằng bạn muốn tạo các biểu tượng **legacy**, **round**, và **biểu tượng cho Google Play Store**. Nhấp **Next** khi hoàn tất.
9. Chạy ứng dụng.

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

Mẹo: Xem **Launcher Icons** để tìm hiểu thêm về cách thiết kế các biểu tượng khởi chạy hiệu quả.

Nhiệm vụ 2: Sử dụng mẫu dự án

Android Studio cung cấp các mẫu (templates) cho các thiết kế ứng dụng và hoạt động (activity) phổ biến cũng như được khuyến nghị. Việc sử dụng các mẫu tích hợp sẵn giúp tiết kiệm thời gian và đảm bảo bạn tuân theo các phương pháp thiết kế tốt nhất.

- Mỗi mẫu bao gồm một **bộ khung hoạt động (skeleton activity)** và giao diện người dùng cơ bản.
- Bạn đã sử dụng mẫu **Empty Activity**.
- Mẫu **Basic Activity** có nhiều tính năng hơn và tích hợp các tính năng ứng dụng được khuyến nghị, chẳng hạn như **menu tùy chọn (options menu)** hiển thị trên **app bar**.

2.1 Khám phá kiến trúc của Basic Activity

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

1. Trong **Android Studio**, tạo một dự án mới bằng mẫu **Basic Activity**.
2. Xây dựng (**Build**) và chạy (**Run**) ứng dụng.
3. Xác định các phần được gắn nhãn trong hình và bảng bên dưới. Tìm các phần tương ứng trên thiết bị hoặc màn hình trình giả lập của bạn. Kiểm tra mã nguồn Java và các tệp XML liên quan được mô tả trong bảng.

Hiểu rõ mã nguồn **Java** và các tệp **XML** sẽ giúp bạn mở rộng và tùy chỉnh mẫu này theo nhu cầu của mình.



First Fragment

:

[Next](#)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam in scelerisque sem. Mauris volutpat, dolor id interdum ullamcorper, risus dolor egestas lectus, sit amet mattis purus dui nec risus. Maecenas non sodales nisi, vel dictum dolor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Suspendisse blandit eleifend diam, vel rutrum tellus vulputate quis. Aliquam eget libero aliquet, imperdiet nisl a, ornare ex. Sed rhoncus est ut libero porta lobortis. Fusce in dictum tellus.

Suspendisse interdum ornare ante. Aliquam nec cursus lorem. Morbi id magna felis. Vivamus egestas, est a condimentum egestas, turpis nisl iaculis ipsum, in dictum tellus dolor sed neq' Morbi tellus erat, dapibus ut s  a, iaculis tincidunt dui. Interdum et malesuada fames ac ante ipsum

#	Mô tả giao diện người dùng	Tham chiếu trong mã
1	Thanh trạng thái Hệ thống Android cung cấp và kiểm soát thanh trạng thái	Không hiển thị trong mã mẫu. Bạn có thể truy cập nó từ Activity của mình. Ví dụ, bạn có thể ẩn thanh trạng thái nếu cần thiết.
2	AppBarLayout > Toolbar Thanh ứng dụng (còn gọi là Action Bar) cung cấp cấu trúc giao diện, các thành phần giao diện tiêu chuẩn và điều hướng. Để đảm bảo tính tương thích ngược, AppBarLayout trong mẫu chứa một Toolbar với chức năng tương tự như ActionBar	Trong tệp activity_main.xml , tìm android.support.v7.widget.Toolbar bên trong android.support.design.widget.AppBarLayout . Thay đổi Toolbar để thay đổi giao diện của thành phần cha, App Bar . Để xem ví dụ, hãy tham khảo App Bar Tutorial .
3	Tên ứng dụng Tên này được lấy từ tên gói của bạn, nhưng có thể tùy chỉnh theo ý muốn	Trong tệp AndroidManifest.xml : <code>android:label="@string/app_name"</code>
4	Nút menu tùy chọn (Options menu overflow button) Chứa các mục menu cho Activity , cũng như các tùy chọn chung như Tìm kiếm (Search) và Cài đặt (Settings) của ứng dụng. Các mục menu của ứng dụng sẽ được thêm vào menu này.	Trong MainActivity.java : <ul style="list-style-type: none"> • onOptionsItemSelected() triển khai hành động xảy ra khi một mục menu được chọn. Trong res > menu > menu_main.xml : <ul style="list-style-type: none"> • Đây là tệp tài nguyên xác định các mục menu cho menu tùy chọn.
5	Bố cục ViewGroup CoordinatorLayout là một ViewGroup giàu tính năng, cung cấp cơ chế để các phần tử View (UI) tương tác. Giao diện	Trong activity_main.xml Không có View nào được chỉ định trực tiếp trong bố cục này; thay vào đó, nó bao gồm một bố cục khác bằng lệnh include layout , để đưa @layout/content_main vào, nơi chứa các View.

	người dùng của ứng dụng được đặt trong tệp content_main.xml , tệp này được bao gồm bên trong ViewGroup này.	Điều này giúp tách biệt các View của hệ thống khỏi các View riêng của ứng dụng.
6	TextView Trong ví dụ, TextView được sử dụng để hiển thị " Hello World ". Hãy thay thế nó bằng các phản tử giao diện người dùng (UI elements) phù hợp với ứng dụng của bạn.	Trong content_main.xml Tất cả các phản tử giao diện người dùng (UI elements) của ứng dụng đều được định nghĩa trong tệp này.
7	Nút hành động nổi (Floating Action Button – FAB)	Trong activity_main.xml được thêm vào dưới dạng một phản tử giao diện người dùng (UI element) sử dụng biểu tượng clip-art. Trong MainActivity.java bao gồm một đoạn mã mẫu (stub) trong <code>onCreate()</code> , thiết lập trình nghe sự kiện (<code>onClick()</code> listener) cho FAB.

2.2 Tùy chỉnh ứng dụng được tạo bởi mẫu

Thay đổi giao diện của ứng dụng được tạo bởi **mẫu Hoạt động Cơ bản**. Ví dụ, bạn có thể thay đổi **màu của thanh ứng dụng** để khớp với **thanh trạng thái** (trên một số thiết bị, thanh trạng thái có màu tối hơn của cùng một màu chính).

Bạn cũng có thể **xóa nút hành động nổi** nếu không sử dụng nó.

1. Thay đổi màu của thanh toolbar(Toolbar) trong activity_main.xml

- o Bằng cách thay đổi giá trị `android:background` thành:

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

- o Điều này sẽ đặt màu của thanh ứng dụng thành màu chính tối hơn, giúp khớp với thanh trạng thái.

2. Để xóa nút hành động nổi, bắt đầu bằng cách xóa đoạn mã trong `onCreate()`

- Thiết lập trình nghe sự kiện (`onClick()` listener) cho nút.
- Mở `MainActivity` và xóa khỏi mã sau.

```
binding.fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
            .setAnchorView(R.id.fab)
            .setAction(text: "Action", listener: null).show();
    }
});
```

3. Để xóa nút hành động nổi khỏi bộ cục, xóa khỏi mã XML sau

- Trong `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"
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="16dp"
    app:srcCompat="@android:drawable/ic_dialog_email" />
```

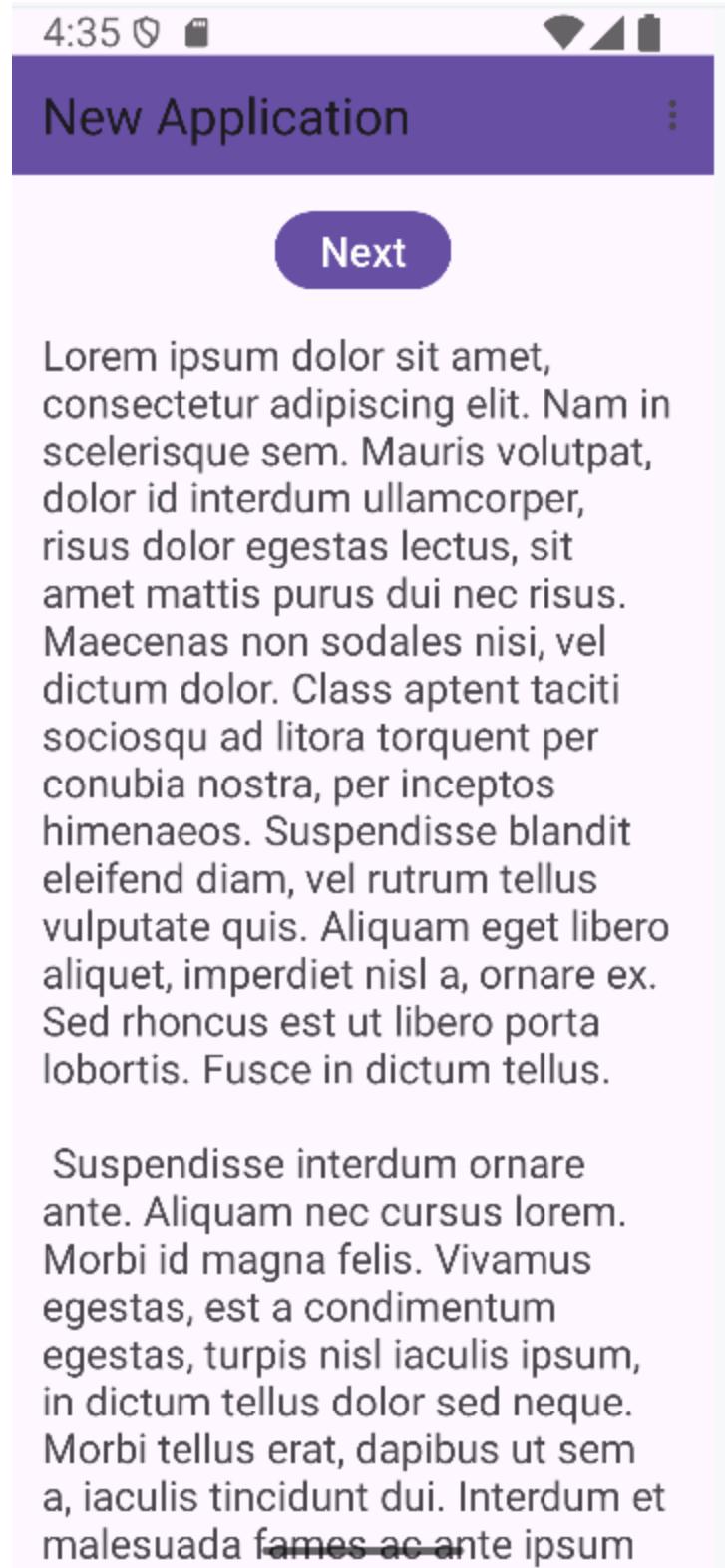
4. Thay đổi tên của ứng dụng hiển thị trên thanh ứng dụng

- Bằng cách thay đổi giá trị chuỗi `app_name` trong `strings.xml` thành nội dung sau.

```
<string name="app_name">New Application</string>
```

5. Chạy ứng dụng

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



Mẹo: Xem Truy cập tài nguyên để biết chi tiết về cú pháp XML khi truy cập tài nguyên.

2.3 Khám phá cách thêm hoạt động bằng mẫu

Trong các bài thực hành trước, bạn đã sử dụng các mẫu **Empty Activity** và **Basic Activity**.

Trong các bài học sau, các mẫu bạn sử dụng sẽ thay đổi tùy theo nhiệm vụ. Các mẫu **Activity** này cũng có sẵn trong dự án của bạn, cho phép bạn thêm **Activity** mới vào ứng dụng ngay cả sau khi thiết lập dự án ban đầu.

(Bạn sẽ tìm hiểu thêm về lớp **Activity** trong một chương khác.)

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

Bài 3: Học từ mã nguồn mẫu

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

3.1 Mã nguồn mẫu Android

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

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

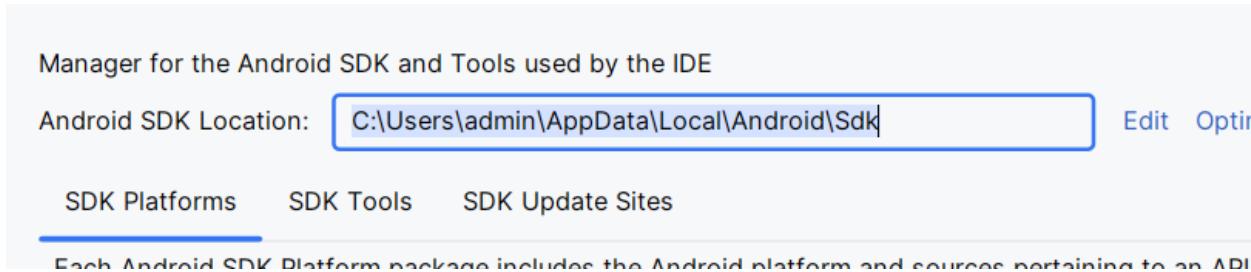
Lưu ý: Các mẫu này chỉ là điểm khởi đầu để phát triển thêm. Chúng tôi khuyến khích bạn tự thiết kế và xây dựng ý tưởng của riêng mình.

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

Khi cài đặt Android Studio, các thành phần thiết yếu của Android SDK (Software Development Kit) cũng được cài đặt. Tuy nhiên, bạn có thể cài đặt thêm thư viện và tài liệu bằng SDK Manager.

1. Chọn **Tools > Android > SDK Manager**.

2. Ở cột bên trái, nhấp vào **Android SDK**.
3. Sao chép đường dẫn trong phần **Android SDK Location** (bạn sẽ cần nó để tìm tài liệu trên máy tính).



4. Nhấp vào tab **SDK Platforms** để cài đặt các phiên bản Android bổ sung.
5. Nhấp vào tab **SDK Update Sites** để xem các trang cập nhật mà Android Studio kiểm tra thường xuyên.
6. Nhấp vào tab **SDK Tools** để cài đặt các công cụ SDK bổ sung và tài liệu nhà phát triển Android ngoại tuyến.
7. Chọn hộp kiểm **Documentation for Android SDK** nếu nó chưa được cài đặt, rồi nhấp vào **Apply**.
8. Khi quá trình cài đặt hoàn tất, nhấp vào **Finish**.
9. Điều hướng đến thư mục **sdk** đã sao chép trước đó, mở thư mục **docs**.
10. Tìm tệp **index.html** và mở nó.

Bài 4: Nhiều nguồn tài nguyên hơn

- Kênh YouTube **Android Developer** là một nguồn tuyệt vời để học các hướng dẫn và mẹo hay.
- Đội ngũ Android thường xuyên đăng tin tức và mẹo trên **blog chính thức của Android**.
- **Stack Overflow** là một cộng đồng lập trình viên hỗ trợ lẫn nhau. Nếu gặp vấn đề, rất có thể ai đó đã đăng câu trả lời. Bạn có thể thử đặt câu hỏi như:
 - "Làm thế nào để thiết lập và sử dụng ADB qua WiFi?"
 - "Những lỗi rò rỉ bộ nhớ phổ biến nhất trong lập trình Android là gì?"
- Cuối cùng nhưng không kém phần quan trọng, hãy tìm kiếm trên Google. Công cụ tìm kiếm sẽ thu thập kết quả từ tất cả các nguồn trên. Ví dụ: "Phiên bản Android OS phổ biến nhất ở Ấn Độ là gì?"

4.1 Tìm kiếm trên Stack Overflow bằng thẻ (tags)

Truy cập **Stack Overflow** và nhập **[android]** vào ô tìm kiếm. Dấu **[]** giúp bạn tìm các bài viết có thẻ liên quan đến Android.

Bạn có thể kết hợp nhiều thẻ và từ khóa để tìm kiếm chính xác hơn. Hãy thử các tìm kiếm sau:

- **[android] and [layout]**
- **[android] "hello world"**

Để tìm hiểu thêm về cách tìm kiếm hiệu quả trên Stack Overflow, hãy xem **trung tâm trợ giúp của Stack Overflow**.

Tóm tắt

- **Tài liệu chính thức về Android Developer:** developer.android.com
- **Material Design** là một triết lý thiết kế giúp ứng dụng hoạt động và hiển thị tốt trên thiết bị di động.
- **Google Play Store** là nền tảng phân phối ứng dụng của Google dành cho các ứng dụng phát triển bằng Android SDK.
- **Android Studio** cung cấp các mẫu thiết kế cho ứng dụng và hoạt động (activity) phổ biến, được khuyến nghị. Những mẫu này bao gồm mã nguồn hoạt động cho các trường hợp sử dụng phổ biến.
- Khi tạo một dự án, bạn có thể chọn một mẫu cho activity đầu tiên của mình. Trong quá trình phát triển ứng dụng, bạn có thể tạo thêm các activity và thành phần khác từ các mẫu có sẵn.
- **Android Studio** chứa nhiều đoạn mã mẫu mà bạn có thể nghiên cứu, sao chép và tích hợp vào dự án của mình.

Khái niệm liên quan

Tài liệu về khái niệm liên quan có trong **Mục 1.4: Các tài nguyên giúp bạn học tập**.

Tìm hiểu thêm

Tài liệu về Android Studio

- **Giới thiệu về Android Studio**
- **Những bước cơ bản trong quy trình phát triển**

Tài liệu dành cho lập trình viên Android

- **Trang web dành cho lập trình viên Android**
- **Khóa đào tạo của Google Developers**
- **Bố cục (Layouts)**
- **Tổng quan về tài nguyên ứng dụng**
- **Bố cục (Layouts)**
- **Menu**
- **TextView**
- **Tài nguyên chuỗi (String resources)**
- **Tệp khai báo ứng dụng (App Manifest)**

Mã nguồn mẫu

- **Mã nguồn bài tập trên GitHub**
- **Mã mẫu dành cho lập trình viên Android**

Video

- **Kênh YouTube Android Developer**
- **Các khóa học trực tuyến trên Udacity**

Khác

- **Blog chính thức của Android**
- **Blog của Android Developers**
- **Google Developers Codelabs**
- **Stack Overflow**
- **Từ vựng Android**

Bài tập về nhà

Tải một ứng dụng mẫu và khám phá tài nguyên

1. Tải một ứng dụng mẫu vào **Android Studio**.
2. Mở một tệp **Java activity** trong ứng dụng. Tìm một lớp, kiểu dữ liệu hoặc thủ tục mà bạn chưa quen thuộc và tra cứu trong tài liệu **Android Developer**.
3. Truy cập **Stack Overflow** và tìm các câu hỏi về chủ đề tương tự.
4. Thay đổi biểu tượng khởi chạy. Sử dụng một biểu tượng có sẵn trong mục **image assets** của **Android Studio**.

Bài 2) Activities

2.1) Activity và Intent

Giới thiệu

Một **Activity** đại diện cho một màn hình đơn lẻ trong ứng dụng của bạn, nơi người dùng có thể thực hiện một tác vụ cụ thể, chẳng hạn như chụp ảnh, gửi email hoặc xem bản đồ. Một hoạt động (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 kết nối lồng léo với nhau. Mỗi màn hình là một activity. Thông thường, một activity trong ứng dụng được chỉ định là "**main activity**" (**MainActivity.java**), và đây là activity được hiển thị khi ứng dụng được khởi chạy. Activity chính có thể khởi động các activity khác để thực hiện các hành động khác nhau.

Mỗi khi một activity mới được khởi động, activity trước đó sẽ dừng lại, nhưng hệ thống sẽ lưu activity đó trong một ngăn xếp (ngăn xếp "back stack"). Khi một activity mới được khởi động, activity mới này sẽ được đẩy vào ngăn xếp và nhận quyền điều khiển từ người dùng. Ngăn xếp "back stack" tuân theo nguyên tắc cơ bản "vào sau, ra trước" (**last in, first out**). Khi người dùng hoàn thành với activity hiện tại và nhấn nút **Back**, activity đó sẽ được xóa khỏi ngăn xếp và bị hủy, và activity trước đó sẽ được tiếp tục.

Một activity được khởi động hoặc kích hoạt bằng cách sử dụng một **intent**. Một **Intent** là một thông điệp 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 khác trong ứng dụng. Bạn sử dụng intent để khởi động một activity từ một activity khác và truyền dữ liệu giữa các activity.

Intent có thể là **explicit** hoặc **implicit**:

- Một **explicit intent** (intent rõ ràng) là intent trong đó bạn biết rõ mục tiêu của intent đó. Nghĩa là, bạn đã biết tên lớp đầy đủ (fully qualified class name) của activity cụ thể.

- Một **implicit intent** (intent không rõ ràng) là intent trong đó 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 **explicit intents**. Bạn sẽ tìm hiểu cách sử dụng **implicit intents** trong bài thực hành sau.

Những gì bạn nên biết trước:

Bạn cần có khả năng:

- Tạo và chạy các ứng dụng trong Android Studio.
- Sử dụng trình chỉnh sửa bố cục (**layout editor**) để tạo bố cục trong một **ConstraintLayout**.
- Chính sửa mã XML của bố cục.
- Thêm chức năng **onClick** cho một **Button**.

Những gì bạn sẽ học được:

- Cách tạo một **Activity** mới trong Android Studio.
- Cách định nghĩa **parent** (cha) và **child activities** (hoạt động con) để sử dụng điều hướng **Up navigation**.
- Cách khởi động một **Activity** bằng một **explicit Intent**.
- Cách truyền dữ liệu giữa các **Activity** bằng một **explicit Intent**.

Những gì bạn sẽ làm:

- Tạo một ứng dụng Android mới với một **main Activity** (activity chính) và một **second Activity** (activity thứ hai).
- Truyền một số dữ liệu (chuỗi) từ **main Activity** sang **second Activity** bằng một **Intent**, và hiển thị dữ liệu đó trong **second Activity**.
- Gửi một dữ liệu khác (dạng chuỗi) ngược lại từ **second Activity** về **main Activity**, cũng bằng một **Intent**.

Tổng quan ứng dụng:

Trong chương này, bạn sẽ tạo và xây dựng một ứng dụng gọi là **Two Activities**, đúng như tên gọi, ứng dụng này chứa hai **Activity**. Bạn sẽ xây dựng ứng dụng qua ba giai đoạn:

- Ở giai đoạn đầu tiên, bạn tạo một ứng dụng với **main activity** chứa một nút **Send**. Khi người dùng nhấn nút này, **main activity** sẽ sử dụng một **intent** để khởi động **second activity**.
- Ở giai đoạn thứ hai, bạn thêm một thành phần **EditText** vào **main activity**. Người dùng nhập một thông điệp và nhấn nút **Send**. **Main activity** sử dụng một **intent** để khởi động **second activity** và gửi thông điệp của người dùng đến **second activity**. **Second activity** hiển thị thông điệp mà nó nhận được.
- Ở giai đoạn cuối cùng của việc tạo ứng dụng **Two Activities**, bạn thêm một thành phần **EditText** và một nút **Reply** vào **second activity**. Nay, người dùng có thể nhập một thông điệp phản hồi và nhấn nút **Reply**, và thông điệp phản hồi sẽ được hiển thị trên **main activity**. Trong giai đoạn này, bạn sử dụng một **intent** để chuyển thông điệp phản hồi từ **second activity** trở lại **main activity**.

Nhiệm vụ 1: Tạo dự án TwoActivities

Trong nhiệm vụ này, bạn thiết lập dự án ban đầu với một **main Activity**, định nghĩa bố cục, và tạo một phương thức cơ bản cho sự kiện nhấn nút **onClick**.

1.1 Tạo dự án TwoActivities

1. Mở **Android Studio** và tạo một **dự án Android Studio mới**.
 - o Đặt tên ứng dụng là **Two Activities** và chọn cài đặt **Phone and Tablet** giống như các bài thực hành trước.
 - o Thư mục dự án sẽ tự động được đặt tên là **TwoActivities**, và tên ứng dụng hiển thị trên **App Bar** sẽ là "**Two Activities**".
2. Chọn **Empty Activity** làm mẫu Activity. Nhấn **Next**.
3. Chấp nhận tên **Activity** mặc định là **MainActivity**.
 - o Đảm bảo rằng **Generate Layout file** và **Backwards Compatibility (AppCompat)** đã được chọn.
4. Nhấn **Finish** để hoàn tất tạo dự án.

1.2 Định nghĩa bố cục cho Main Activity

1. Mở **res > layout > activity_main.xml** trong **Project > Android**. Trình chỉnh sửa bố cục (**Layout Editor**) sẽ xuất hiện.
2. Nhấn vào tab **Design** (nếu chưa được chọn) và xóa **TextView** mặc định (có nội dung "**Hello World**") trong **Component Tree**.
3. Khi chế độ **Autoconnect** đang bật (mặc định), kéo một **Button** từ **Palette** vào góc dưới bên phải của bố cục. **Autoconnect** sẽ tự động tạo các ràng buộc (**constraints**) cho **Button**.

4. Trong **Attributes**, đặt các thuộc tính sau:

- **ID**: button_main
- **layout_width** và **layout_height**: wrap_content
- **Text**: Send

Lúc này, bố cục sẽ trông như sau:



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

 android:onClick="launchSecondActivity"

Giá trị của thuộc tính được gạch dưới màu đỏ vì phương thức launchSecondActivity() chưa được tạo. Bỏ qua lỗi này tạm thời; bạn sẽ sửa trong nhiệm vụ tiếp theo.

6. Trích xuất tài nguyên chuỗi, như đã mô tả trong bài thực hành trước, cho văn bản "Send" và sử dụng tên tài nguyên là button_main.

Mã XML cho nút Button sẽ trông như sau:

```
<Button
    android:id="@+id/button_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp"
    android:onClick="launchSecondActivity"
    android:text="@string/button_main"
    app:layout_constraintBaseline_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    tools:ignore="OnClick" />
```

1.3 Xác định hành động của Button

Trong tác vụ này, bạn sẽ triển khai phương thức launchSecondActivity() mà bạn đã tham chiếu trong bộ cục thông qua thuộc tính android:onClick.

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

```

public void launchSecondActivity(View view) {
    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 tiếp theo

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

```

1 usage
private static final String LOG_TAG = MainActivity.class.getSimpleName();
@Override

```

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

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

Mã cho MainActivity sẽ trông như sau:

```

package com.example.twoactivities;

import ...

public class MainActivity extends AppCompatActivity {
    1 usage
    private static final String LOG_TAG = MainActivity.class.getSimpleName();
    @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 launchSecondActivity(View view) {
        Log.d(LOG_TAG, msg: "Button clicked!");
    }
}

```

Nhiệm vụ 2: Tạo và khởi chạy Activity thứ hai

Mỗi activity mới mà bạn thêm vào dự án sẽ có tệp layout và tệp Java riêng, tách biệt với những tệp của activity chính. Các activity này cũng có các phần tử `<activity>` riêng trong tệp **AndroidManifest.xml**.

Giống như activity chính, các 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 lỏng lẻo với các activity khác. Tuy nhiên, bạn có thể định nghĩa một activity là **parent** (cha) của một activity khác trong tệp **AndroidManifest.xml**. Mỗi quan hệ cha-con này cho phép Android thêm các gợi ý điều hướng, chẳng hạn như mũi tên quay lại hướng trái trong thanh tiêu đề của mỗi activity.

Một activity giao tiếp với các activity khác (trong cùng một ứng dụng hoặc giữa các ứng dụng khác nhau) bằng một **intent**. Một **Intent** có thể là:

- **Intent rõ ràng (explicit intent)**: Là loại mà bạn biết rõ mục tiêu của intent đó; nghĩa là bạn đã biết tên lớp đầy đủ của activity cụ thể đó.
- **Intent không rõ ràng (implicit intent)**: Là loại mà bạn không có tên của thành phần mục tiêu, nhưng có một hành động tổng quát để thực hiện.

Trong nhiệm vụ này, bạn thêm một activity thứ hai vào ứng dụng, với layout riêng của nó. Bạn chỉnh sửa tệp **AndroidManifest.xml** để định nghĩa activity chính là **parent** của activity thứ hai. Sau đó, bạn chỉnh sửa phương thức **launchSecondActivity()** trong **MainActivity** để bao gồm một intent giúp khởi chạy activity thứ hai khi bạn nhấn nút.

2.1 Tạo Activity thứ hai

1. Nhấp vào thư mục **app** của dự án và chọn **File > New > Activity > Empty Activity**.
2. Đặt tên cho Activity mới là **SecondActivity**. Đảm bảo rằng các tùy chọn **Generate Layout File** và **Backwards Compatibility (AppCompat)** được chọn. Tên layout sẽ được tự động điền là **activity_second**.
Không chọn tùy chọn **Launcher Activity**.
3. Nhấn **Finish**. Android Studio sẽ thêm cả một tệp layout mới (**activity_second.xml**) và một tệp Java mới (**SecondActivity.java**) vào dự

án của bạn cho Activity mới. Đồng thời, 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 **SecondActivity**:

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

```
<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:label="Second Activity"
    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 (app bar). VỚI thuộc tính `parentActivityName`, bạn chỉ định rằng **MainActivity** là cha (parent) của **SecondActivity**. Mỗi quan hệ này được sử dụng cho điều hướng **Up navigation** trong ứng dụng của bạn: thanh ứng dụng của **SecondActivity** sẽ có mũi tên quay trái để người dùng có thể điều hướng "lên trên" (quay lại) **MainActivity**.

Với phần tử `<meta-data>`, bạn cung cấp thông tin tùy ý bổ sung về Activity dưới dạng các cặp key-value. Trong trường hợp này, các thuộc tính metadata thực hiện cùng một chức năng như thuộc tính `android:parentActivityName`—chúng định nghĩa một mối quan hệ giữa hai Activity cho điều hướng "lên trên". 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 từ cấp 16 trở lên.

4. Trích xuất tài nguyên chuỗi (string resource) cho "Second Activity" trong đoạn mã trên, và sử dụng tên tài nguyên là activity2_name.

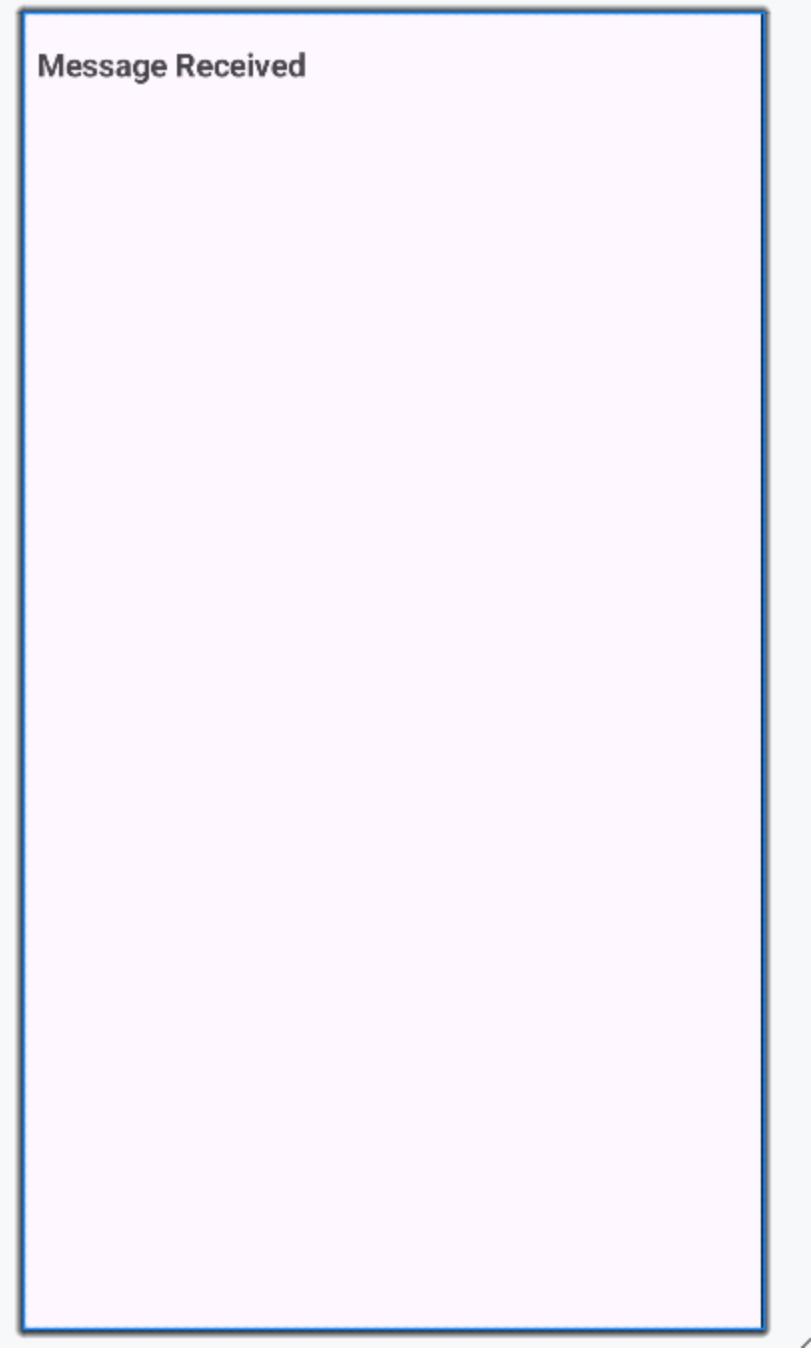
2.3 Định nghĩa bố cục cho Activity thứ hai

1. Mở tệp **activity_second.xml** và nhấp vào tab **Design** nếu nó chưa được chọn.
2. Kéo một **TextView** từ bảng **Palette** vào góc trên bên trái của bố cục và thêm các ràng buộc (**constraints**) vào các cạnh trên và trái của bố cục.
 - o Thiết lập các thuộc tính của nó trong bảng **Attributes** như sau:

Attribute	Value
id	text_header
Top margin	16
Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	Message Received
textAppearance	AppCompat.Medium
textStyle	B (bold)

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

Bố cục bây giờ sẽ trông như thế này:



Message Received

3. Nhập vào tab **Text** để chỉnh sửa mã XML, sau đó 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** dành cho các phiên bản Android cũ hơn.

Mã XML cho tệp **activity_second.xml** nên như sau:

```

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

    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
    />
</androidx.constraintlayout.widget.ConstraintLayout>

```

2.4 Thêm một Intent vào Main Activity

Trong nhiệm vụ này, bạn sẽ thêm một **Intent** tường minh (explicit Intent) vào **MainActivity**. Intent này được sử dụng để kích hoạt **SecondActivity** khi nút **Send** được nhấn.

Các bước thực hiện:

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

Constructor của **Intent** yêu cầu hai đối số cho một **explicit Intent**:

- **Context**: Ngữ cảnh ứng dụng (sử dụng **this** để tham chiếu đến **MainActivity**).
- **Component**: Thành phần cụ thể sẽ nhận Intent (sử dụng **SecondActivity.class**).

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

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

```
startActivity(intent);
```

4. Chạy ứng dụng.

Khi bạn nhấn nút **Send**, **MainActivity** sẽ gửi **Intent** và hệ thống Android sẽ khởi chạy **SecondActivity**, hiển thị màn hình của nó. Để quay lại **MainActivity**, bạn có thể nhấp vào nút **Up** (mũi tên quay lại ở thanh ứng dụng) hoặc nút **Back** ở dưới cùng của màn hình.

Task 3: Gửi dữ liệu từ **MainActivity** sang **SecondActivity**

Trong nhiệm vụ trước, bạn đã thêm một intent rõ ràng vào **MainActivity** để khởi chạy **SecondActivity**. Bạn cũng có thể sử dụng một intent để **gửi dữ liệu** từ một activity này sang một activity khác trong quá trình khởi chạy.

Đối tượng intent có thể truyền dữ liệu tới activity mục tiêu theo hai cách: trong trường **data**, hoặc trong phần **intent extras**.

- Dữ liệu trong intent là một **URI** chỉ định **dữ liệu cụ thể cần xử lý**.
- Nếu thông tin bạn muốn truyền tới activity thông qua intent không phải là một **URI**, hoặc nếu bạn muốn gửi nhiều thông tin, bạn có thể đưa thông tin bổ sung vào phần **extras**.

Extras của intent là các cặp key/value được lưu trữ trong một **Bundle**. **Bundle** là một tập hợp dữ liệu được lưu trữ dưới dạng các cặp key/value.

Để truyền thông tin từ một activity này sang một activity khác, bạn đưa các key và value vào phần **extras** của intent từ activity gửi và sau đó lấy chúng ra trong activity nhận.

Trong nhiệm vụ này, bạn sẽ chỉnh sửa intent rõ ràng trong **MainActivity** để thêm dữ liệu bổ sung (trong trường hợp này là một chuỗi do người dùng nhập vào) vào **extras** của intent. Sau đó, bạn chỉnh sửa **SecondActivity** để lấy dữ liệu này từ **extras** của intent và hiển thị trên màn hình.

3.1 Thêm một **EditText** vào bố cục của **MainActivity**

1. Mở tệp **activity_main.xml**.
2. Kéo một thành phần **Plain Text (EditText)** từ **Palette** vào phía dưới của bố cục:
 - Thêm ràng buộc (constraints) vào cạnh trái của bố cục, cạnh dưới của bố cục, và cạnh trái của nút **Send**.

- Đặt các thuộc tính trong bảng **Attributes** như sau:

Attribute	Value
id	editText_main
Right margin	8

Left margin	8
Bottom margin	16
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 trong tệp activity_main.xml sẽ trông như thế này:

5:58



TwoActivities

Enter Your Message H

Send

- Nhấn vào tab **Text** để chỉnh sửa mã XML và trích xuất chuỗi "**Enter Your Message Here**" vào một tài nguyên (resource) với tên là **editText_main**.

Mã XML cho bố cục sẽ trông giống như sau.

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

    <Button
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="16dp"
        android:onClick="launchSecondActivity"
        android:text="@string/button_main"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

    <EditText
        android:id="@+id/editText_main"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginRight="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginBottom="16dp"
        android:ems="10"
        android:inputType="textLongMessage"
        android:hint="@string/editText_main"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button_main" />
```

```
    app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3.2 Thêm một chuỗi vào Intent extras

Các **Intent extras** là các cặp key/value được lưu trong một **Bundle**. Một **Bundle** là một tập hợp dữ liệu được lưu trữ dưới dạng cặp key/value. Để truyền thông tin từ một **Activity** sang một **Activity** khác, bạn cần đưa các cặp key/value vào **Intent extras** từ **Activity** gửi, và sau đó lấy chúng ra từ **Intent extras** trong **Activity** nhận.

1. Mở **MainActivity**.
2. Thêm một hằng số **public** ở đầu lớp để định nghĩa key cho **Intent extra**:

```
no usages
public static final String EXTRA_MESSAGE = "com.example.android.twoactivities.extra.MESSAGE";
..
```

3. Thêm một biến riêng tư ở đầu lớp để chứa **EditText**.

```
private EditText mMessageEditText;
```

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

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

5. Trong phương thức **launchSecondActivity()**, ngay dưới Intent mới, lấy văn bản từ **EditText** dưới dạng 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);
```

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

```

@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;
    });
    mMessageEditText = findViewById(R.id.editText_main);
}

```

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

```

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

```

3.3 Thêm một TextView vào SecondActivity để hiển thị thông điệp

1. Mở tệp activity_second.xml.
2. Kéo một TextView khác vào giao diện bên dưới TextView có ID là text_header, và thêm các ràng buộc (constraints) vào cạnh trái của giao diện và cạnh dưới của text_header.
3. Đặt các thuộc tính cho TextView mới trong bảng Attributes như sau:

Attribute	Value
id	text_message
Top margin	8
Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	(Delete any text in this field)
textAppearance	AppCompat.Medium

Giao diện mới trông giống như trong bài tập trước, vì TextView mới chưa (chưa có) chứa bất kỳ văn bản nào, vì vậy nó không xuất hiện trên màn hình. Mã XML cho giao diện activity_second.xml sẽ trông giống như sau

```

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

<TextView
    android:id="@+id/text_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    android:textAppearance="AppCompat.Medium"
    app:layout_constraintBottom_toBottomOf="@+id/text_header"
    app:layout_constraintStart_toStartOf="parent" />

```

3.4 Sửa đổi SecondActivity để lấy dữ liệu extras và hiển thị thông điệp

1. Mở SecondActivity để thêm mã vào phương thức onCreate().
2. Lấy Intent kích hoạt Activity này:

```
Intent intent = getIntent();
```

3. Lấy chuỗi chứa thông điệp từ extras của Intent bằng cách sử dụng biến tĩnh MainActivity.EXTRA_MESSAGE làm khóa:

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

4. Sử dụng findViewById() để tham chiếu đến TextView cho thông điệp từ giao diện:

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

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

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

- Chạy ứng dụng. Khi bạn nhập thông điệp trong MainActivity và nhấn Send, SecondActivity sẽ được mở và hiển thị thông điệp.

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

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable($this$enableEdgeToEdge: this);  
    setContentView(R.layout.activity_second);  
    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;  
    });  
    Intent intent = getIntent();  
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);  
    TextView textView = findViewById(R.id.text_message);  
    textView.setText(message);  
}
```

Nhiệm vụ 4: Trả lại dữ liệu cho Activity chính

Bây giờ bạn đã có một ứng dụng khởi chạy một Activity mới và gửi dữ liệu đến đó, bước cuối cùng là trả lại dữ liệu từ Activity thứ hai về Activity chính. Bạn cũng sẽ sử dụng Intent và intent extras cho nhiệm vụ này.

4.1 Thêm một EditText và một Button vào giao diện SecondActivity

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

```
<string name="button_second">Reply</string>  
<string name="editText_second">Enter Your Reply Here</string>
```

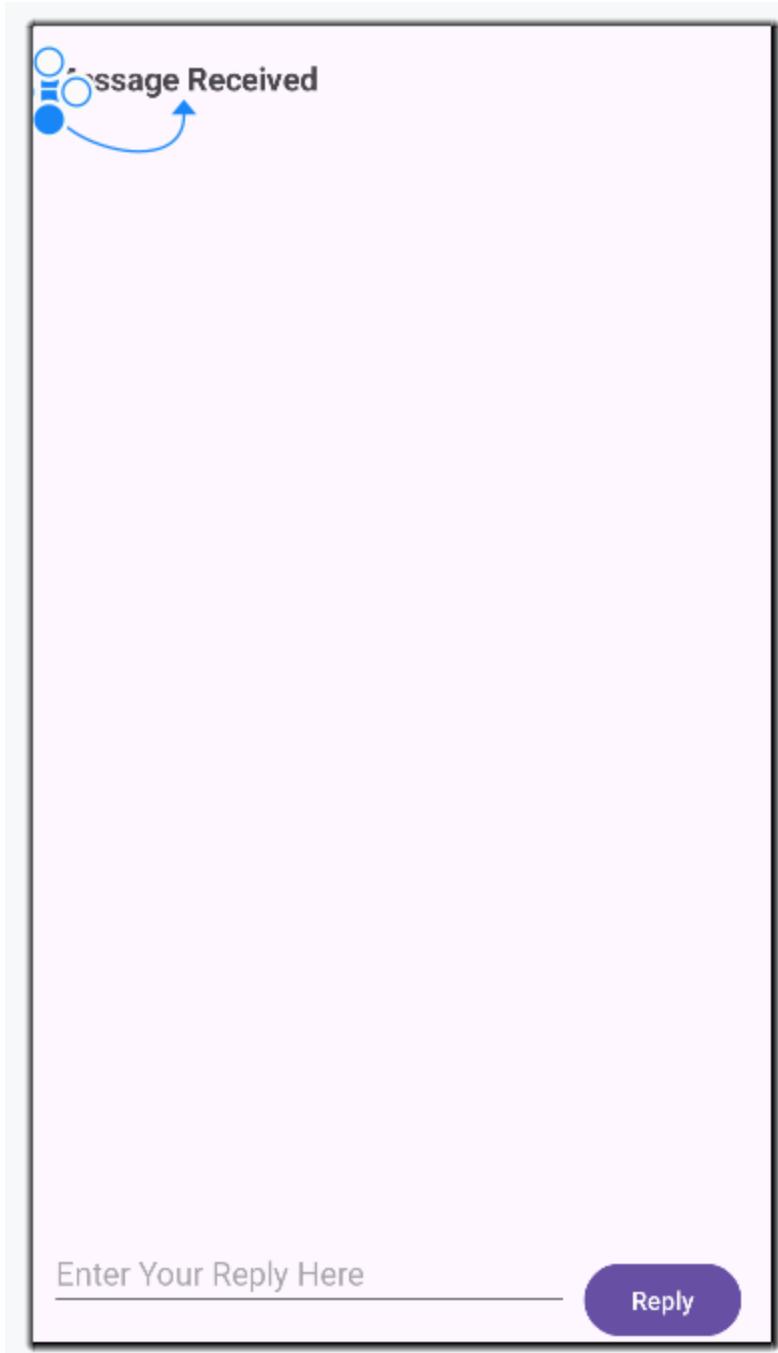
- Mở tệp activity_main.xml và activity_second.xml.
- Sao chép EditText và Button từ tệp layout activity_main.xml và dán chúng vào layout activity_second.xml.
- Trong activity_second.xml, sửa đổi giá trị thuộc tính cho Button như sau:

Old attribute value	New attribute value
android:id="@+id/button_main"	android:id="@+id/button_second"
android:onClick="launchSecondActivity"	android:onClick="returnReply"
android:text="@string/button_main"	android:text="@string/button_second"

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

Old attribute value	New attribute value
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 XML layout, nhấp vào returnReply, nhấn Alt+Enter (Option+Return trên Mac), và chọn Tạo 'returnReply(View)' trong 'SecondActivity'. Android Studio sẽ tạo ra một phương thức khung cho trình xử lý returnReply(). Bạn sẽ triển khai phương thức này trong nhiệm vụ tiếp theo. Giao diện mới cho activity_second.xml trông như sau:



Mã XML cho tệp layout activity_second.xml như sau:

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

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SecondActivity">

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

    <TextView
        android:id="@+id/text_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:textAppearance="App.AppCompat.Medium"
        app:layout_constraintBottom_toBottomOf="@+id/text_header"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/button_second"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="16dp"
        android:onClick="returnReply"
        android:text="@string/button_second"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

    <EditText
        android:id="@+id/editText_second"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginRight="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginBottom="16dp"
        android:ems="10"
        android:inputType="textLongMessage"
        android:hint="@string/editText_second"
```

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button_second"
    app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

4.2 Tạo một Intent phản hồi trong Activity thứ hai

Dữ liệu phản hồi từ Activity thứ hai về Activity chính được gửi trong một Intent extra.

Bạn tạo ra Intent trả lại này và đưa dữ liệu vào đó theo cách tương tự như cách bạn làm với Intent gửi đi.

1. Mở SecondActivity.

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

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

3. Thêm một biến riêng tư ở đầu lớp để chứa EditText.

```
private EditText mReply;
```

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

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

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

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

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

```
Intent replyIntent = new Intent();
```

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

```
replyIntent.putExtra(EXTRA_REPLY, reply);
```

8. Đặt kết quả thành RESULT_OK để chỉ ra rằng phản hồi đã thành công. Lớp Activity định nghĩa các mã kết quả, bao gồm RESULT_OK và RESULT_CANCELLED.

```
setResult(RESULT_OK, replyIntent);
```

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

```
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 nhiệm vụ này, bạn sẽ thêm các phần tử TextView vào giao diện activity_main.xml để hiển thị phản hồi trong MainActivity.

Để làm cho nhiệm vụ này dễ dàng hơn, bạn sao chép các phần tử TextView mà bạn đã sử dụng trong SecondActivity.

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

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

2. Mở tệp activity_main.xml và activity_second.xml.
3. Sao chép hai phần tử TextView từ tệp layout activity_second.xml và dán chúng vào giao diện activity_main.xml phía trên Button.
4. Trong activity_main.xml, sửa đổi giá trị thuộc tính cho TextView đầu tiên như sau:

Old attribute value	New attribute value
android:id="@+id/text_header"	android:id="@+id/text_header_reply"
android:text="@string/text_header"	android:text="@string/text_header_reply"

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

Old attribute value	New attribute value
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 chúng ban đầu vô hình. (Việc để chúng hiển thị trên màn hình mà không có nội dung 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 lại từ Activity thứ hai.

Giao diện activity_main.xml trông giống như trong nhiệm vụ trước — mặc dù bạn đã thêm hai phần tử TextView mới vào giao diện. Vì bạn đã đặt các phần tử này thành vô hình, nên chúng không hiển thị trên màn hình.

Dưới đây là mã XML cho tệp activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/text_header_reply"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:text="@string/text_header_reply"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:visibility="invisible"/>
    <TextView
        android:id="@+id/text_message_reply"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
```

```
        android:layout_marginTop="8dp"

        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        app:layout_constraintTop_toBottomOf="@+id/text_header_reply"
        app:layout_constraintStart_toStartOf="parent"
        android:visibility="invisible"/>

<Button
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="16dp"
        android:onClick="launchSecondActivity"
        android:text="@string/button_main"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

<EditText
        android:id="@+id/editText_main"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginRight="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginBottom="16dp"
        android:ems="10"
        android:inputType="textLongMessage"
        android:hint="@string/editText_main"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button_main"
        app:layout_constraintStart_toStartOf="parent" />

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

4.4 Lấy phản hồi từ Intent extra và hiển thị nó

Khi bạn sử dụng một Intent rõ ràng để bắt đầu một Activity khác, bạn có thể không mong đợi nhận lại bất kỳ dữ liệu nào — bạn chỉ đang kích hoạt Activity đó. Trong trường hợp này, bạn sử dụng startActivityForResult() để bắt đầu Activity mới, như bạn đã làm trước đó trong bài thực hành này. Tuy nhiên, nếu bạn muốn lấy dữ liệu từ Activity đã được kích hoạt, bạn cần bắt đầu nó với startActivityForResult().

Trong nhiệm vụ này, bạn sẽ chỉnh sửa ứng dụng để bắt đầu SecondActivity và mong đợi một kết quả, lấy dữ liệu trả về từ Intent và hiển thị dữ liệu đó trong các phần tử TextView mà bạn đã tạo trong nhiệm vụ trước.

1. Mở MainActivity.
2. Thêm một hằng số public ở đầu lớp để định nghĩa khóa cho một loại phản hồi mà bạn quan tâm:

```
public static final int TEXT_REQUEST = 1;
```

3. Thêm hai biến riêng tư để chứa phần tử tiêu đề phản hồi và phần tử TextView phản hồi:

```
private TextView mReplyHeadTextView;
```

no usages

```
private TextView mReplyTextView;
```

4. Trong phương thức onCreate(), sử dụng findViewById() để lấy tham chiếu từ giao diện đến phần tử tiêu đề phản hồi và phần tử TextView phản hồi. Gán các đối tượng view này cho các biến riêng tư:

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

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

Phương thức onCreate() đây đủ bây giờ sẽ trông như sau:

```

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;
    });
    mMessageEditText = findViewById(R.id.editText_main);
    mReplyHeadTextView = findViewById(R.id.text_header_reply);
    mReplyTextView = findViewById(R.id.text_message_reply);|
}

```

- Trong phương thức launchSecondActivity(), thay đổi lời gọi từ startActivity() thành startActivityForResult(), và bao gồm khóa TEXT_REQUEST làm đối số:

startActivityForResult(intent, *TEXT_REQUEST*);

- Ghi đè phương thức callback onActivityResult() với chữ ký sau:**

```

public void onActivityResult(int requestCode, int resultCode, Intent data)

```

Ba tham số trong onActivityResult() chứa tất cả thông tin mà bạn cần để xử lý dữ liệu trả về: requestCode là mã yêu cầu mà bạn đã đặt khi khởi động Activity với startActivityForResult(), resultCode là mã kết quả được đặt trong Activity đã được khởi động (thường là một trong RESULT_OK hoặc RESULT_CANCELED), và Intent data chứa dữ liệu trả về từ Activity đã được khởi động.

- Bên trong onActivityResult(), gọi super.onActivityResult():**

```

super.onActivityResult(requestCode, resultCode, data);

```

- Thêm mã để kiểm tra TEXT_REQUEST để đảm bảo bạn xử lý đúng kết quả của Intent, trong trường hợp có nhiều kết quả. Cũng kiểm tra RESULT_OK để đảm bảo rằng yêu cầu đã thành công:**

```
if (requestCode == TEXT_REQUEST)
{
    if (resultCode == RESULT_OK)
    {
        ...
    }
}
```

Lớp Activity định nghĩa các mã kết quả. Mã có thể là RESULT_OK (yêu cầu thành công), RESULT_CANCELED (người dùng hủy bỏ thao tác), hoặc RESULT_FIRST_USER (dành cho mã kết quả do người dùng tự định nghĩa).

9. **Bên trong khôi if con, lấy Intent extra từ phản hồi Intent (data):** Ở đây, khóa cho extra là hằng số EXTRA_REPLY từ SecondActivity. Bạn sẽ sử dụng data.getStringExtra() để lấy giá trị trả về:

```
String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);
```

10. **Đặt độ hiển thị của tiêu đề phản hồi (reply header) thành true:**

```
mReplyHeadTextView.setVisibility(View.VISIBLE);
```

11. **Đặt nội dung cho TextView phản hồi (reply TextView) với reply và đặt độ hiển thị của nó thành true:**

```
mReplyTextView.setText(reply);
```

```
mReplyTextView.setVisibility(View.VISIBLE);
```

Phương thức onActivityResult() đầy đủ:

```
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == TEXT_REQUEST)
    {
        if (resultCode == RESULT_OK)
        {
            String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);
            mReplyHeadTextView.setVisibility(View.VISIBLE);
            mReplyTextView.setText(reply);
            mReplyTextview.setVisibility(View.VISIBLE);
        }
    }
}
```

12. Chạy ứng dụng.

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

11:17



TwoActivities

Reply Received

Thauhs sds



Hi

Send

2.2) Vòng đời của Activity và trạng thái

Giới thiệu

Trong bài thực hành này, bạn sẽ tìm hiểu thêm về **vòng đời của Activity**. Vòng đời là tập hợp các trạng thái mà một Activity có thể trải qua trong suốt thời gian tồn tại của nó, từ khi được tạo ra đến khi bị hủy và tài nguyên của nó được hệ thống thu hồi. Khi người dùng điều hướng giữa các Activity trong ứng dụng của bạn (cũng như khi vào hoặc thoát khỏi ứng dụng), các Activity sẽ 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()**, và nhiều hơn nữa. Khi một Activity thay đổi trạng thái, phương thức callback tương ứng sẽ được gọi. Bạn đã từng thấy một trong những phương thức này: **onCreate()**. Bằng cách ghi đè bất kỳ phương thức callback nào của vòng đời trong các lớp **Activity** của bạn, bạn có thể thay đổi hành vi mặc định của Activity để phản hồi các hành động của người dùng hoặc hệ thống.

Trạng thái của Activity cũng có thể thay đổi do các thay đổi cấu hình của thiết bị, ví dụ như khi người dùng xoay thiết bị từ chế độ dọc sang ngang. Khi những thay đổi cấu hình này xảy ra, Activity sẽ bị hủy và được tạo lại trong trạng thái mặc định, và người dùng có thể mất thông tin mà họ đã nhập vào Activity. Để tránh làm người dùng bối rối, điều quan trọng là bạn cần phát triển ứng dụng của mình để ngăn ngừa việc mất dữ liệu ngoài ý muốn. Trong phần sau của bài thực hành này, bạn sẽ thử nghiệm với các thay đổi cấu hình và học cách bảo toàn trạng thái của Activity để phản hồi những thay đổi cấu hình thiết bị và các sự kiện khác trong vòng đời của Activity.

Trong bài thực hành này, bạn sẽ thêm các câu lệnh ghi nhật ký (logging statements) vào ứng dụng **TwoActivities** và quan sát các thay đổi vòng đời của Activity khi bạn sử dụng ứng dụng. Sau đó, bạn sẽ bắt đầu làm việc với những thay đổi này và khám phá cách xử lý đầu vào của người dùng trong những điều kiện đó.

Những gì bạn cần biết trước

Bạn cần có khả năng:

- Tạo và chạy một dự án ứng dụng trong Android Studio.
- Thêm các câu lệnh ghi nhật ký (log statements) vào ứng dụng của bạn và

xem các nhật ký đó trong bảng **Logcat**.

- Hiểu và làm việc với **Activity** và **Intent**, đồng thời 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 lại và bị hủy.
- Các phương thức callback của vòng đời liên quan đến những thay đổi của **Activity**.
- Tác động của các hành động (chẳng hạn như thay đổi cấu hình) có thể dẫn đến các sự kiện trong vòng đời **Activity**.
- Cách giữ trạng thái của **Activity** qua 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 phương thức callback vòng đời khác nhau của **Activity**, bao gồm cả các câu lệnh ghi nhật ký.
- Quan sát các thay đổi trạng thái khi ứng dụng của bạn chạy và khi bạn tương tác với từng **Activity** trong ứng dụng.
- Sửa đổi ứng dụng của bạn để giữ trạng thái phiên bản của một **Activity** khi nó được tạo lại một cách không mong muốn do hành vi của người dùng hoặc thay đổi cấu hình trên thiết bị.

Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ bổ sung vào ứng dụng **TwoActivities**. Ứng dụng sẽ trông và hoạt động gần giống như trong bài codelab trước. Nó chứa hai triển khai **Activity** và cung cấp cho người dùng khả năng gửi dữ liệu giữa chúng. Những thay đổi bạn thực hiện trong bài thực hành này sẽ không ảnh hưởng đến hành vi giao diện người dùng có thể thấy được của ứng dụng.

Nhiệm vụ 1: Thêm các callback vòng đời vào TwoActivities

Trong nhiệm vụ này, bạn sẽ triển khai tất cả các phương thức callback vòng đời của **Activity** để in thông báo vào **logcat** khi các phương thức này được gọi. Các thông báo log này sẽ cho phép bạn thấy khi nào vòng đời **Activity** thay đổi trạng thái và cách những thay đổi trạng thái vòng đời này ảnh hưởng đến ứng dụng khi nó chạy.

1.1 (Tùy chọn) Sao chép dự án TwoActivities

Đối với các nhiệm vụ 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 một bản sao của dự án.

1.2 Thêm các callback vào MainActivity

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

```
Log.d(LOG_TAG, " ");
Log.d(LOG_TAG, "onCreate");
```

3. Thêm một phương thức ghi đè cho callback **onStart()**, với một câu lệnh ghi nhật ký cho sự kiện đó:

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

Để tiết kiệm thời gian, chọn **Code > Override Methods** trong Android Studio. Một hộp thoại sẽ xuất hiện với tất cả các phương thức có thể ghi đè trong lớp của bạn. Chọn một hoặc nhiều phương thức callback từ danh sách sẽ chèn một mẫu hoàn chỉnh cho các phương thức đó, bao gồm cả lời gọi bắt buộc đến lớp cha (superclass).

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 chữ ký (signature) giống nhau (ngoại trừ tên). Nếu bạn Copy và Paste phương thức **onStart()** để tạo các phương thức callback khác, đừng quên cập nhật nội dung để gọi đúng phương thức trong lớp cha, và ghi nhật ký đúng phương thức.

1. Chạy ứng dụng của bạn.
2. Nhấp vào tab **Logcat** ở dưới cùng của Android Studio để hiển thị bảng

Logcat. Bạn sẽ thấy ba thông báo nhật ký hiển thị ba trạng thái vòng đời mà Activity đã chuyển qua khi nó bắt đầu:

```
D onCreat  
D onStart  
D onResume
```

1.3 Triển khai các phương thức callback vòng đời trong 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 phương thức callback vòng đời và câu lệnh log vào SecondActivity.** (Bạn có thể sao chép và dán các phương thức callback từ MainActivity).
4. **Thêm một câu lệnh log vào phương thức returnReply() ngay trước khi gọi finish().**

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

1.4 Quan sát log khi ứng dụng chạy

1. Chạy ứng dụng của bạn.
2. Nhấp vào tab **Logcat** ở dưới cùng trong Android Studio để hiển thị cửa sổ Logcat.
3. Nhập **Activity** vào ô tìm kiếm.

Logcat của Android có thể rất dài và lộn xộn. Vì biến **LOG_TAG** trong mỗi lớp chứa từ **MainActivity** hoặc **SecondActivity**, từ khóa này giúp bạn lọc log chỉ hiển thị những thông tin bạn quan tâm.

Time	Process	Method	Caller	State
2025-03-03 23:54:13.174	4240-4240	CompatChangeReporter	com.example.twoactivities	D Compat change id reported: 63938206; UID 10219; state: ENABLED
2025-03-03 23:54:13.272	4240-4240	CompatChangeReporter	com.example.twoactivities	D Compat change id reported: 303326708; UID 10219; state: ENABLED
2025-03-03 23:54:13.278	4240-4240	MainActivity	com.example.twoactivities	D
2025-03-03 23:54:13.278	4240-4240	MainActivity	com.example.twoactivities	D onCreat
2025-03-03 23:54:13.278	4240-4240	MainActivity	com.example.twoactivities	D onStart
2025-03-03 23:54:13.278	4240-4240	MainActivity	com.example.twoactivities	D onResume
2025-03-03 23:54:13.278	4240-4240	MainActivity	com.example.twoactivities	D onPause
2025-03-03 23:54:13.278	4240-4240	MainActivity	com.example.twoactivities	D onStop
2025-03-03 23:54:13.278	4240-4240	MainActivity	com.example.twoactivities	D onDestroy
2025-03-03 23:54:13.278	4240-4240	MainActivity	com.example.twoactivities	D onRestart
2025-03-03 23:54:13.278	4240-4240	MainActivity	com.example.twoactivities	D onCreate
2025-03-03 23:54:13.304	4240-4245	e.twoactivities	com.example.twoactivities	I Compiler allocated 4219KB to compile void android.widget.TextView.<init>(anc
2025-03-03 23:54:13.314	4240-4240	MainActivity	com.example.twoactivities	D onStart
2025-03-03 23:54:13.317	4240-4240	MainActivity	com.example.twoactivities	D onResume
2025-03-03 23:54:13.328	4240-4240	HWUI	com.example.twoactivities	W Unknown dataspace 0

Thử nghiệm sử dụng ứng dụng của bạn và ghi chú lại các sự kiện vòng đời xảy ra khi thực hiện các hành động khác nhau. Cụ thể, thử các điều sau:

- Sử dụng ứng dụng bình thường (gửi tin nhắn, trả lời bằng tin nhắn khác).
 - Dùng nút **Back** để quay lại từ **SecondActivity** về **MainActivity**.
 - Dùng nút **Up arrow** trong thanh công cụ ứng dụng để quay lại từ **SecondActivity** về **MainActivity**.
 - Xoay thiết bị trên cả **MainActivity** và **SecondActivity** vào các thời điểm khác nhau trong ứng dụng và quan sát những gì xảy ra trong log và trên màn hình.
 - Nhấn nút **Overview** (nút vuông bên phải nút Home) và đóng ứng dụng (chạm vào nút **X**).
 - Quay lại màn hình chính và khởi động lại ứng dụng của bạn.

MEO: Nếu bạn đang chạy ứng dụng trong giả lập, bạn có thể mở phông việc xoay màn hình bằng cách nhấn **Control+F11** hoặc **Control+Function+F11**.

Mã giải pháp cho tác vụ 1

Dưới đây là đoạn mã thêm vào **MainActivity**, nhưng không phải là toàn bộ lớp.

Phuong thức onCreate():

```
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;
    });
    mMessageEditText = findViewById(R.id.editText_main);
    mReplyHeadTextView = findViewById(R.id.text_header_reply);
    mReplyTextView = findViewById(R.id.text_message_reply);
    Log.d(LOG_TAG, msg: "");
    Log.d(LOG_TAG, msg: "onCreate");
    Log.d(LOG_TAG, msg: "onStart");
    Log.d(LOG_TAG, msg: "onResume");
    Log.d(LOG_TAG, msg: "onPause");
    Log.d(LOG_TAG, msg: "onStop");
    Log.d(LOG_TAG, msg: "onDestroy");
    Log.d(LOG_TAG, msg: "onRestart");
    Log.d(LOG_TAG, msg: "onCreate");
```

Các phương thức vòng đời khác:

```

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

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d(LOG_TAG, msg: "onDestroy");

}

@Override
protected void onStop() {
    super.onStop();
    Log.d(LOG_TAG, msg: "onStop");
}

1 usage
@Override
protected void onRestart() {
    super.onRestart();
    Log.d(LOG_TAG, msg: "onRestart");
}

```

SecondActivity

Dưới đây là đoạn mã thêm vào **SecondActivity**, nhưng không phải là toàn bộ lớp.

Ở đầu lớp **SecondActivity**:

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

Phương thức **returnReply()**:

```
public void returnReply(View view) {  
    String reply = mReply.getText().toString();  
    Intent replyIntent = new Intent();  
    replyIntent.putExtra(EXTRA_REPLY, reply);  
    setResult(RESULT_OK, replyIntent);  
    Log.d(LOG_TAG, msg: "End SecondActivity");  
    finish();  
}
```

Các phương thức vòng đời khác:
Giống như đối với **MainActivity**, ở trên.

Tác vụ 2: Lưu và phục hồi trạng thái của Activity

Tùy thuộc vào tài nguyên hệ thống và hành vi người dùng, mỗi **Activity** trong ứng dụng của bạn có thể bị hủy và tái tạo lại nhiều hơn bạn nghĩ.

Bạn có thể đã nhận thấy hành vi này trong phần trước khi bạn xoay thiết bị hoặc giả lập. Xoay thiết bị là một ví dụ của thay đổi cấu hình thiết bị. Mặc dù xoay màn hình là thay đổi cấu hình phổ biến nhất, tất cả các thay đổi cấu hình đều dẫn đến việc **Activity** hiện tại bị hủy và tái tạo lại như thể nó là mới. Nếu bạn không xử lý hành vi này trong mã của mình, khi xảy ra thay đổi cấu hình, giao diện **Activity** có thể trở lại giao diện mặc định và các giá trị ban đầu, và người dùng có thể mất vị trí, dữ liệu hoặc trạng thái tiến trình trong ứng dụng của bạn.

Trạng thái của mỗi **Activity** được lưu dưới dạng một tập hợp các cặp khóa/giá trị trong một đối tượng **Bundle** gọi là trạng thái phiên bản **Activity**. Hệ thống lưu thông tin trạng thái mặc định vào **Bundle** của trạng thái phiên bản ngay trước khi **Activity** bị dừng và chuyển **Bundle** đó đến phiên bản **Activity** mới để phục hồi.

Để tránh mất dữ liệu trong **Activity** khi nó bị hủy và tái tạo lại một cách bất ngờ, bạn cần triển khai phương thức **onSaveInstanceState()**. Hệ thống gọi phương thức này trên **Activity** của bạn (giữa **onPause()** và **onStop()**) khi có khả năng **Activity** có thể bị hủy và tái tạo lại.

Dữ liệu bạn lưu trong trạng thái phiên bản chỉ áp dụng cho chính phiên bản **Activity** này trong phiên làm việc hiện tại của ứng dụng. Khi bạn dừng và khởi động lại một phiên làm việc mới, trạng thái phiên bản **Activity** bị mất và **Activity** trở lại giao diện mặc định. Nếu bạn cần lưu trữ dữ liệu người dùng giữa các phiên làm việc của ứng dụng, hãy sử dụng **shared preferences** hoặc cơ sở dữ liệu. Bạn sẽ học về cả hai trong một bài thực hành sau.

2.1 Lưu trạng thái phiên bản Activity với onSaveInstanceState()

Bạn có thể đã nhận thấy rằng việc xoay thiết bị không ảnh hưởng đến trạng thái của **SecondActivity** chút nào. Điều này là vì giao diện và trạng thái của **SecondActivity** được tạo ra từ giao diện và **Intent** kích hoạt nó. Ngay cả khi **Activity** bị tái tạo, **Intent** vẫn còn và dữ liệu trong **Intent** đó vẫn được sử dụng mỗi khi phương thức **onCreate()** trong **SecondActivity** được gọi.

Ngoài ra, bạn có thể nhận thấy rằng trong mỗi **Activity**, bất kỳ văn bản nào bạn đã nhập vào các phần tử **EditText** của tin nhắn hoặc trả lời đều được giữ lại ngay cả khi thiết bị bị xoay. Điều này là vì thông tin trạng thái của một số phần tử **View** trong giao diện của bạn được tự động lưu trữ qua các thay đổi cấu hình, và giá trị hiện tại của **EditText** là một trong những trường hợp đó.

Vì vậy, trạng thái **Activity** mà bạn quan tâm chỉ là các phần tử **TextView** cho tiêu đề trả lời và văn bản trả lời trong **MainActivity**. Cả hai phần tử **TextView** này đều vô hình mặc định; chúng chỉ hiển thị khi bạn gửi tin nhắn trở lại **MainActivity** từ **SecondActivity**.

Trong tác vụ này, bạn sẽ thêm mã để bảo vệ trạng thái phiên bản của hai phần tử **TextView** này bằng cách sử dụng **onSaveInstanceState()**.

1. Mở **MainActivity**.
2. Thêm cấu trúc triển khai của **onSaveInstanceState()** vào **Activity**, hoặc sử dụng **Code > Override Methods** để chèn một phương thức ghi đè cấu trúc.

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

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

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

Hãy nhớ rằng tiêu đề và văn bản trả lời sẽ được đánh dấu là vô hình cho đến khi có một câu trả lời từ **SecondActivity**. Nếu tiêu đề đang hiển thị, điều này có nghĩa là có dữ liệu trả lời cần được lưu lại.

Lưu ý rằng chúng ta chỉ quan tâm đến trạng thái hiển thị đó — văn bản thực tế của tiêu đề không cần phải lưu lại, vì văn bản đó không bao giờ thay đổi.

4. Trong cùng một kiểm tra đó, thêm văn bản trả lời vào trong **Bundle**:

```
protected void onSaveInstanceState(@NotNull Bundle outState) {
    super.onSaveInstanceState(outState);
    if(mReplyHeadTextView.getVisibility() == View.VISIBLE)
    {
        outState.putBoolean("reply_visible", true);
        outState.putString("reply_text", mReplyTextView.getText().toString());
    }
}
```

Ở đây, **mReplyTextView** là phần tử **TextView** chứa văn bản trả lời. Phương thức **putString()** sẽ lưu lại văn bản trả lời vào trong **Bundle**.

Nếu tiêu đề đang hiển thị, bạn có thể giả định rằng tin nhắn trả lời cũng đang hiển thị. Bạn không cần phải kiểm tra hay lưu trạng thái hiển thị hiện tại của tin nhắn trả lời. Chỉ cần lưu lại văn bản thực tế của tin nhắn vào trong **Bundle** trạng thái với khóa "**reply_text**".

Bạn chỉ lưu trạng thái của những phần tử **View** có thể thay đổi sau khi **Activity** được tạo. Các phần tử **View** khác trong ứng dụng của bạn (như **EditText**, **Button**) có thể được tái tạo từ giao diện mặc định bất kỳ lúc nào.

Lưu ý rằng hệ thống sẽ tự động 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 phải phục hồi nó khi **Activity** được tái tạo. Bạn có thể làm điều này trong **onCreate()**, hoặc bằng cách triển khai phương thức **onRestoreInstanceState()**, phương thức này sẽ được gọi sau **onStart()** sau khi **Activity** được tạo.

Hầu hết thời gian, việc phục hồi trạng thái **Activity** trong **onCreate()** là lựa chọn tốt hơn, để đảm bảo rằng giao diện người dùng, bao gồm cả trạng thái, có sẵn càng sớm càng tốt. Tuy nhiên, đôi khi cũng tiện lợi khi làm điều này trong **onRestoreInstanceState()** sau khi tất cả việc khởi tạo đã hoàn tất, hoặc để cho các lớp con quyết định có sử dụng triển khai mặc định của bạn hay không.

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

```
mMessageEditText = findViewById(R.id.editText_main);
mReplyHeadTextView = findViewById(R.id.text_header_reply);
mReplyTextView = findViewById(R.id.text_message_reply);
if(savedInstanceState != null){
    ...
}
```

Khi **Activity** của bạn được tạo, hệ thống sẽ truyền **Bundle** trạng thái vào **onCreate()** như là đối số duy nhất. Lần đầu tiên **onCreate()** được gọi và ứng dụng của bạn khởi động, **Bundle** sẽ là **null** — không có trạng thái tồn tại khi ứng dụng của bạn khởi động lần đầu tiên. Những lần gọi sau của **onCreate()** sẽ có một **Bundle** được điền dữ liệu từ những gì bạn đã lưu trong **onSaveInstanceState()**.

2. Bên trong kiểm tra đó, lấy trạng thái hiển thị hiện tại (đúng hoặc sai) từ **Bundle** với khóa "reply_visible".

```
if(savedInstanceState != null){  
    boolean isVisible = savedInstanceState.getBoolean( key: "reply_visible");  
}
```

3. Thêm một kiểm tra bên dưới dòng trước đó cho biến **isVisible**.

```
if (isVisible){  
}  
}
```

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

4. Bên trong kiểm tra **isVisible**, làm cho tiêu đề hiển thị.

```
mReplyHeadTextView.setVisibility(View.VISIBLE);
```

5. Lấy văn bản trả lời từ **Bundle** với khóa "reply_text", và đặt **TextView** trả lời để hiển thị chuỗi đó.

```
mReplyTextView.setText(savedInstanceState.getString( key: "reply_text"));
```

6. Làm cho **TextView** trả lời hiển thị nữa:

```
mReplyTextView.setVisibility(View.VISIBLE);
```

7. Chạy ứng dụng. Thủ xoay thiết bị hoặc trình giả lập để đảm bảo rằng tin nhắn trả lời (nếu có) vẫn hiển thị trên màn hình sau khi **Activity** được tái tạo.

Mã giải pháp cho nhiệm vụ 2

MainActivity

Các đoạn mã sau đây hiển thị mã đã thêm vào **MainActivity**, nhưng không phải là toàn bộ lớp.

Phương thức **onSaveInstanceState()**:

```

@Override
protected void onSaveInstanceState(@NonNull Bundle outState) {
    super.onSaveInstanceState(outState);
    if(mReplyHeadTextView.getVisibility() == View.VISIBLE)
    {

        outState.putBoolean("reply_visible", true);
        outState.putString("reply_text", mReplyTextView.getText().toString());
    }
}

```

Phuong thuc onCreate():

```

protected void onSaveInstanceState(@NonNull Bundle outState) {
    super.onSaveInstanceState(outState);
    if(mReplyHeadTextView.getVisibility() == View.VISIBLE)
    {

        outState.putBoolean("reply_visible", true);
        outState.putString("reply_text", mReplyTextView.getText().toString());
    }
}

```

@Override

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_main);
}

```

```

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
    Insets systemBars =
    insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top, systemBars.right,
    systemBars.bottom);
    return insets;
});

```

```

mMessageEditText = findViewById(R.id.editText_main);
mReplyHeadTextView = findViewById(R.id.text_header_reply);
mReplyTextView = findViewById(R.id.text_message_reply);
if(savedInstanceState != null){
    boolean isVisible = savedInstanceState.getBoolean("reply_visible");
    if(isVisible){
        mReplyHeadTextView.setVisibility(View.VISIBLE);

mReplyTextView.setText(savedInstanceState.getString("reply_text"));
        mReplyTextView.setVisibility(View.VISIBLE);
    }
}

```

Dự án hoàn chỉnh:

Dự án Android Studio: TwoActivitiesLifecycle

Thách thức lập trình Lưu ý: Tất cả các thách thức lập trình là tùy chọn và không phải là yêu cầu bắt buộc cho các bài học sau.

Thách thức: Tạo một ứng dụng danh sách mua sắm đơn giản với một hoạt động chính cho danh sách mà người dùng đang xây dựng, và một hoạt động thứ hai cho danh sách các mặt hàng mua sắm thông thường.

- Hoạt động chính nên chứa danh sách để xây dựng, bao gồm mười phần tử TextView trống.
- Một nút **Thêm mục** trên hoạt động chính sẽ mở một hoạt động thứ hai chứa danh sách các mặt hàng mua sắm thông thường (Phô mai, Gạo, Táo, v.v.). Sử dụng các phần tử Button để hiển thị các mục.
- Việc chọn một mục sẽ trả người dùng về hoạt động chính và cập nhật một TextView trống để bao gồm mục đã chọn.

Sử dụng một Intent để truyền thông tin từ một hoạt động sang hoạt động khác. Đảm bảo rằng trạng thái hiện tại của danh sách mua sắm được lưu lại khi người dùng xoay thiết bị.

Tóm tắt

- Vòng đời của Activity là một tập hợp các trạng thái mà một Activity chuyển qua, bắt đầu khi nó được tạo ra và kết thúc khi hệ thống Android thu hồi tài nguyên cho Activity đó.
- Khi người dùng điều hướng từ một Activity sang một Activity khác, và trong và 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 bao gồm: onCreate(), onStart(), onPause(), onRestart(), onResume(), onStop(), onDestroy().
- Việc ghi đè một phương thức callback của vòng đời cho phép bạn thêm hành vi xảy ra khi Activity của bạn chuyển sang trạng thái đó.
- Bạn có thể thêm các phương thức ghi đè khung vào các lớp của mình trong Android Studio bằng cách vào **Code > Override**.

Dưới đây là bản dịch tiếng Việt của đoạn mô tả:

- Các thay đổi cấu hình thiết bị như xoay màn hình dẫn đến Activity bị hủy và tái tạo lại như thế nó là mới.
- Một phần trạng thái của Activity được lưu lại khi có sự thay đổi cấu hình, bao gồm các giá trị hiện tại của các phần tử EditText. Đối với tất cả các dữ liệu khác, bạn phải tự lưu trữ dữ liệu đó.
- Lưu trạng thái của Activity trong phương thức onSaveInstanceState().
- Dữ liệu trạng thái của instance được lưu trữ dưới dạng các cặp khóa/giá trị đơn giản trong một Bundle. Sử dụng các phương thức của Bundle để đưa dữ liệu vào và lấy dữ liệu ra khỏi Bundle.
- Khôi phục trạng thái của instance trong onCreate(), đó là cách được ưu tiên, hoặc onRestoreInstanceState().

Khái niệm liên quan

Tài liệu về khái niệm liên quan có trong mục 2.2: Vòng đời và trạng thái của Activity.

Tìm hiểu thêm

Tài liệu của Android Studio:

- Làm quen với Android Studio

Tài liệu của nhà phát triển Android:

- Các nguyên lý cơ bản của ứng dụng
- Hoạt động (Activities)
- Hiểu về vòng đời của Activity
- Intents và Bộ lọc Intent
- Xử lý các thay đổi cấu hình

- Activity
- Intent

Bài tập về nhà

Xây dựng và chạy một ứng dụng

1. Tạo một ứng dụng với một layout chứa một TextView để đếm số, một nút Button để tăng giá trị của bộ đếm, và một EditText. Xem ảnh chụp màn hình dưới đây làm ví dụ. Bạn không cần phải sao chép chính xác layout.
2. Thêm một trình xử lý sự kiện khi nhấn nút Button để tăng bộ đếm.
3. Chạy ứng dụng và tăng bộ đếm. Nhập một số văn bản vào EditText.
4. Xoay thiết bị. Lưu ý rằng bộ đếm bị đặt lại, nhưng EditText thì không.
5. Triển khai phương thức onSaveInstanceState() để lưu lại trạng thái hiện tại của ứng dụng.
6. Cập nhật phương thức onCreate() để khôi phục trạng thái của ứng dụng.
7. Đảm bảo rằng khi bạn xoay thiết bị, trạng thái của ứng dụng vẫn được bảo lưu.

Câu hỏi 1

Nếu bạn chạy ứng dụng bài tập trước khi triển khai phương thức onSaveInstanceState(), điều gì sẽ xảy ra khi bạn xoay thiết bị? Chọn một đáp án:

- EditText không còn chứa văn bản bạn đã nhập, nhưng bộ đếm vẫn được bảo lưu.
- Bộ đếm bị đặt lại về 0, và EditText không còn chứa văn bản bạn đã nhập.
- Bộ đếm bị đặt lại về 0, nhưng nội dung của EditText được bảo lưu.
- Bộ đếm và nội dung của EditText đều được bảo lưu.

Câu hỏi 2

Các phương thức vòng đời của Activity nào được gọi khi có thay đổi cấu hình thiết bị (chẳng hạn như xoay màn hình)? Chọn một đáp án:

- Android ngay lập tức tắt Activity của bạn bằng cách gọi onStop(). Mã của bạn phải khởi động lại Activity.
- Android tắt Activity của bạn bằng cách gọi onPause(), onStop(), và onDestroy(). Mã của bạn phải khởi động lại Activity.
- Android tắt Activity của bạn bằng cách gọi onPause(), onStop(), và onDestroy(), sau đó khởi động lại Activity và gọi onCreate(), onStart(), và onResume().
- Android ngay lập tức gọi onResume().

Câu hỏi 3

Khi nào phương thức onSaveInstanceState() được gọi trong vòng đời của Activity? Chọn một đáp án:

- onSaveInstanceState() được gọi trước phương thức onStop().
- onSaveInstanceState() được gọi trước phương thức onResume().
- onSaveInstanceState() được gọi trước phương thức onCreate().
- onSaveInstanceState() được gọi trước phương thức onDestroy().

Câu hỏi 4

Các phương thức vòng đời của Activity nào là tốt nhất để lưu dữ liệu trước khi Activity kết thúc hoặc bị hủy? Chọn một đáp án:

- onPause() hoặc onStop()
- onResume() hoặc onCreate()
- onDestroy()
- onStart() hoặc onRestart()

2.3) Intent gián tiếp

Giới thiệu

Trong một phần trước, bạn đã học về **explicit intents** (intents rõ ràng). Trong một explicit intent, bạn thực hiện một hoạt động trong ứng dụng của bạn, hoặc trong một ứng dụng khác, bằng cách gửi một intent với tên lớp đầy đủ của hoạt động. Trong bài học này, bạn sẽ tìm hiểu thêm về **implicit intents** (intents gián tiếp) và cách sử dụng chúng để thực hiện các hoạt động.

Với một implicit intent, bạn khởi tạo một hoạt động mà không biết ứng dụng hoặc hoạt động nào sẽ xử lý tác vụ đó. Ví dụ, nếu bạn muốn ứng dụng của mình chụp một bức ảnh, gửi email, hoặc hiển thị một vị trí trên bản đồ, bạn thường không quan tâm ứng dụng hoặc hoạt động nào sẽ thực hiện tác vụ đó.

Ngược lại, hoạt động của bạn có thể khai báo một hoặc nhiều **intent filters** trong tệp AndroidManifest.xml để thông báo rằng hoạt động đó có thể chấp nhận implicit intents và định nghĩa các loại intents mà hoạt động sẽ chấp nhận.

Để khớp yêu cầu của bạn với một ứng dụng đã cài đặt trên thiết bị, hệ thống Android sẽ so khớp implicit intent của bạn với một hoạt động có intent filters chỉ ra rằng chúng có thể thực hiện hành động đó. Nếu có nhiều ứng dụng khớp,

người dùng sẽ được hiển thị một trình chọn ứng dụng cho phép họ chọn ứng dụng mà họ muốn sử dụng để xử lý intent.

Trong bài thực hành này, bạn sẽ xây dựng một ứng dụng gửi một implicit intent để thực hiện các tác vụ sau:

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

Chia sẻ — gửi một mảnh thông tin cho những người khác qua email hoặc mạng xã hội — là một tính năng phổ biến trong nhiều ứng dụng. Để thực hiện hành động chia sẻ, bạn sử dụng lớp ShareCompat.IntentBuilder, giúp tạo một implicit intent để chia sẻ dữ liệu một cách dễ dàng.

Cuối cùng, bạn sẽ tạo một intent-receiver đơn giản để chấp nhận một implicit intent cho một hành động cụ thể.

Những gì bạn đã biết

Bạn sẽ có khả năng:

- Sử dụng trình chỉnh sửa layout để thay đổi một layout.
- Chính sửa mã XML của một layout.
- Thêm một Button và một trình xử lý sự kiện nhấn nút.
- Tạo và sử dụng một Activity.
- Tạo và gửi một Intent giữa hai Activity.

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

- Tạo một ứng dụng mới để thử nghiệm với Implicit Intent.
- Triển khai một Implicit Intent để mở một trang web, và một Implicit Intent khác để 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 Implicit Intent để mở một trang web.

Tổng quan về ứng dụng

Trong phần này, bạn sẽ tạo một ứng dụng mới với một Activity và ba tùy chọn 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 chưa có giá trị mặc định.

Nhiệm vụ 1: Tạo dự án và layout

Trong bài tập này, bạn sẽ tạo một dự án và ứng dụng mới có tên là **Implicit Intents**, với một layout mới.

1.1 Tạo dự án

1. Mở Android Studio và tạo một dự án Android Studio mới. Đặt tên ứng dụng của bạn là **Implicit Intents**.
2. Chọn **Empty Activity** làm mẫu cho dự án. Nhấn **Next**.
3. Chấp nhận tên **Activity** mặc định là **MainActivity**. Đảm bảo ô **Generate Layout file** được chọn. Nhấn **Finish**.

1.2 Tạo layout

Trong nhiệm vụ này, bạn sẽ tạo layout cho ứng dụng. Sử dụng một LinearLayout, ba phần tử Button, và ba phần tử EditText, như sau:

1. Mở ứng dụng > res > values > strings.xml trong bảng **Project > Android**, và thêm các tài nguyên chuỗi sau:

```
<string name="button_uri">Open Website</string>
<string name="edittex_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>
```

2. Mở res > layout > activity_main.xml trong bảng **Project > Android**. Nhập vào tab **Text** để chuyển sang mã XML.
3. Thay đổi android.support.constraint.ConstraintLayout thành LinearLayout, như bạn đã học trong bài thực hành trước.
4. Thêm thuộc tính android:orientation với giá trị "vertical". Thêm thuộc tính android:padding với giá trị "16dp".

```

<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"
    android:padding="16dp"
    tools:context=".MainActivity">

```

5. Xóa TextView hiển thị "Hello World".
6. Thêm một bộ các phần tử giao diện người dùng cho nút **Open Website**. Bạn cần một phần tử EditText và một phần tử Button. Sử dụng các giá trị thuộc tính sau:

EditText attribute	Value
android:id	"@+id/website_edittext"
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:text	"@string/edittext_uri"
Button attribute	Value
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 đỏ cho đến khi bạn định nghĩa phương thức callback trong một nhiệm vụ sau.

7. **Thêm một bộ các phần tử giao diện người dùng** (EditText và Button) vào layout cho nút **Open Location**. Sử dụng các thuộc tính giống như trong bước trước, nhưng sửa đổi chúng như sau (Bạn có thể sao chép các giá trị từ nút **Open Website** và chỉnh sửa chúng)

EditText attribute	Value
android:id	"@+id/location_edittext"
android:text	"@string/edittext_loc"
Button attribute	Value
android:id	"@+id/open_location_button"
android:text	"@string/button_loc"
android:onClick	"openLocation"

Giá trị cho thuộc tính android:onClick sẽ vẫn được gạch chân đỏ cho đến khi bạn định nghĩa phương thức callback trong một nhiệm vụ sau.

8. **Thêm một bộ các phần tử giao diện người dùng** (EditText và Button) vào layout cho nút **Share This**. Sử dụng các thuộc tính như dưới đây. (Bạn có thể sao chép các giá trị từ nút **Open Website** và chỉnh sửa chúng.)

EditText attribute	Value
android:id	"@+id/share_edittext"
android:text	"@string/edittext_share"
Button attribute	Value
android:id	"@+id/share_text_button"
android:text	"@string/button_share"
android:onClick	"shareText"

Tùy thuộc vào phiên bản Android Studio của bạn, mã activity_main.xml của bạn sẽ trông giống như sau. Các giá trị cho thuộc tính android:onClick sẽ vẫn được gạch chân đỏ cho đến khi bạn định nghĩa các phương thức callback trong một nhiệm vụ sau:

```
<?xml version="1.0" encoding="utf-8"?>
<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"
    android:padding="16dp"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/website_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/edittext_uri" />

    <Button
        android:id="@+id/open_website_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        android:text="@string/button_uri"
        android:onClick="openWebsite"/>

    <EditText
        android:id="@+id/location_edittext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/edittex_loc" />

    <Button
        android:id="@+id/open_location_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        android:text="@string/button_loc"
        android:onClick="openLocation"/>

    <EditText
```

```
    android:id="@+id/share_edittext"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/edittext_share" />

<Button
    android:id="@+id/share_text_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="@string/button_share"
    android:onClick="shareText"/>

</LinearLayout>
```

Nhiệm vụ 2: Triển khai nút Open Website

Trong nhiệm vụ này, bạn triển khai phương thức xử lý sự kiện nhấn nút cho nút đầu tiên trong layout, **Open Website**. Hành động này sử dụng một **Implicit Intent** để gửi URI đã cho đến một **Activity** có thể xử lý **Implicit Intent** đó (chẳng hạn như trình duyệt web).

2.1 Định nghĩa openWebsite()

1. Nhập vào "openWebsite" trong mã XML activity_main.xml.
2. Nhấn **Alt+Enter** (hoặc **Option+Enter** trên 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()**.

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

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

```
private EditText mWebsiteEditText;
```

- Trong phương thức **onCreate()** của **MainActivity**, sử dụng **findViewById()** để lấy tham chiếu đến đối tượng **EditText** và gán nó cho biến private đó:

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

2.2 Thêm mã cho **openWebsite()**

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

```
String url = mWebsiteEditText.getText().toString();
```

2. Mã hóa và phân tích chuỗi đó thành đối tượng Uri:

```
Uri webpage = Uri.parse(url);
```

- Tạo một **Intent** mới với **Intent.ACTION_VIEW** làm hành động và URI làm dữ liệu:

```
Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
```

Trình tạo **Intent** này khác với cái bạn đã sử dụng để tạo một **explicit Intent**. Trong trình tạo trước, bạn chỉ định ngữ cảnh hiện tại và một thành phần cụ thể (lớp **Activity**) để gửi **Intent**. Trong trình tạo này, bạn chỉ định một hành động và dữ liệu cho hành động đó. Các hành động được định nghĩa 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 gọi một số điện thoại). Trong trường hợp này, hành động là **ACTION_VIEW** vì bạn muốn hiển thị trang web được chỉ định bởi URI trong biến **webpage**.

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

```
if (intent.resolveActivity(getApplicationContext()) != null) {
```

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

5. Bên trong câu lệnh **if**, gọi **startActivity()** để gửi **Intent**:

```
startActivity(intent);
```

6. Thêm một khôi else để in một thông báo Log nếu Intent không thể được giải quyết.

```
else {
    Log.d( tag: "ImplicitIntents", msg: "Can't handle this!");
}
```

Phương thức **openWebsite()** bây giờ nên trông như sau. (Các chú thích đã được thêm vào để làm rõ.)

```
public void openWebsite(View view) {
    String url = mWebsiteEditText.getText().toString();
    Uri webpage = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
    if (intent.resolveActivity(getApplicationContext()) != null) {
        startActivity(intent);
    } else {
        Log.d( tag: "ImplicitIntents", msg: "Can't handle this!");
    }
}
```

Nhiệm vụ 3: Triển khai nút Mở Vị trí

Trong nhiệm vụ này, bạn sẽ triển khai phương thức xử lý sự kiện khi nhấp vào nút thứ hai trong giao diện người dùng, **Mở Vị trí**. Phương thức này gần như giống hệt với phương thức **openWebsite()**. Điểm khác biệt là việc sử dụng URI dạng geo để chỉ định một vị trí bản đồ. Bạn có thể sử dụng URI dạng geo với vĩ độ và kinh độ, hoặc sử dụng chuỗi truy vấn để chỉ định một vị trí chung. Trong ví dụ này, chúng tôi đã sử dụng cách thứ hai.

3.1 Định nghĩa openLocation()

1. Nhập vào "openLocation" trong mã XML của **activity_main.xml**.
2. Nhấn **Alt+Enter** (hoặc **Option+Enter** trên Mac) và chọn **Create 'openLocation(View)' in MainActivity.**
 - o Android Studio sẽ tự động tạo một phương thức khung xương trong **MainActivity** dành cho trình xử lý openLocation().

```
public void openLocation(View view) {  
    |  
}  
}
```

3. Thêm một biến riêng (private variable) ở đầu lớp **MainActivity** để lưu đối tượng **EditText** cho URI vị trí:

```
private EditText mLocationEditText;
```

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

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

3.2 Thêm mã vào openLocation()

1. Trong phương thức mới **openLocation()**, thêm một lệnh để lấy giá trị chuỗi từ đối tượng **mLocationEditText**:

```
String loc = mLocationEditText.getText().toString();
```

2. Phân tích chuỗi đó thành một đối tượng **Uri** với một truy vấn tìm kiếm geo:

```
Uri addressUri = Uri.parse(uriString: "geo: 0, 0?q=" + loc);
```

3. Tạo một đối tượng **Intent** mới với **Intent.ACTION_VIEW** làm hành động và **addressUri** làm dữ liệu:

```
Intent intent = new Intent(Intent.ACTION_VIEW, addressUri);
```

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

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

Khi hoàn thành, phương thức **openLocation()** đây sẽ trông như sau:

```
public void openLocation(View view) {
    String loc = mLocationEditText.getText().toString();
    Uri addressUri = Uri.parse( uriString: "geo: 0, 0?q=" + loc);
    Intent intent = new Intent(Intent.ACTION_VIEW, addressUri);
    if (intent.resolveActivity(getApplicationContext()) != null)
        startActivity(intent);
    else
        Log.d( tag: "ImplicitIntents", msg: "Can't handle this intent!");
}
```

Nhiệm vụ 4: Triển khai nút Chia sẻ Văn bản này

Hành động chia sẻ là một cách đơn giản để người dùng chia sẻ nội dung trong ứng dụng của bạn lên mạng xã hội và các ứng dụng khác. Mặc dù bạn có thể tự xây dựng một hành động chia sẻ trong ứng dụng bằng cách sử dụng một Intent ngầm định, Android cung cấp lớp trợ giúp **ShareCompat.IntentBuilder** để việc triển khai chia sẻ trở nên dễ dàng hơn.

Bạn có thể sử dụng **ShareCompat.IntentBuilder** để tạo một Intent và khởi chạy trình chọn (chooser), cho phép người dùng chọn ứng dụng đích để chia sẻ.

Trong nhiệm vụ này, bạn sẽ triển khai chức năng chia sẻ một đoạn văn bản trong ô nhập văn bản (text edit), bằng cách sử dụng lớp **ShareCompat.IntentBuilder**.

4.1 Định nghĩa shareText()

1. Nhập vào "shareText" trong mã XML của **activity_main.xml**.

2. Nhấn **Alt+Enter** (hoặc **Option+Enter** trên Mac) và chọn **Create 'shareText(View)' in MainActivity**.
 - o Android Studio sẽ tự động tạo một phương thức khung xương trong **MainActivity** cho trình xử lý **shareText()**.

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

3. Thêm một biến riêng (private variable) ở đầu lớp **MainActivity** để lưu đối tượng **EditText**:

```
private EditText mShareEditText;
```

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

```
mShareEditText = findViewById(R.id.share_edittext);
```

4.2 Thêm mã vào shareText()

1. Trong phương thức **shareText()**, thêm một lệnh để lấy giá trị chuỗi từ đối tượng **mShareEditText**:

```
String txt = mShareEditText.getText().toString();
```

2. Xác định loại MIME của văn bản cần chia sẻ:

```
String mimeType = "text/plain";
```

3. Gọi **ShareCompat.IntentBuilder** với các phương thức sau:

```

public void shareText(View view) {
    String txt = mShareEditText.getText().toString();
    String mimeType = "text/plain";
    ShareCompat.IntentBuilder
        .from( launchingActivity: this)
        .setType(mimeType)
        .setChooserTitle("Share this text with:")
        .setText(txt)
        .startChooser();
}

```

4. Trích xuất giá trị của .setChooserTitle thành một tài nguyên chuỗi (string resource).

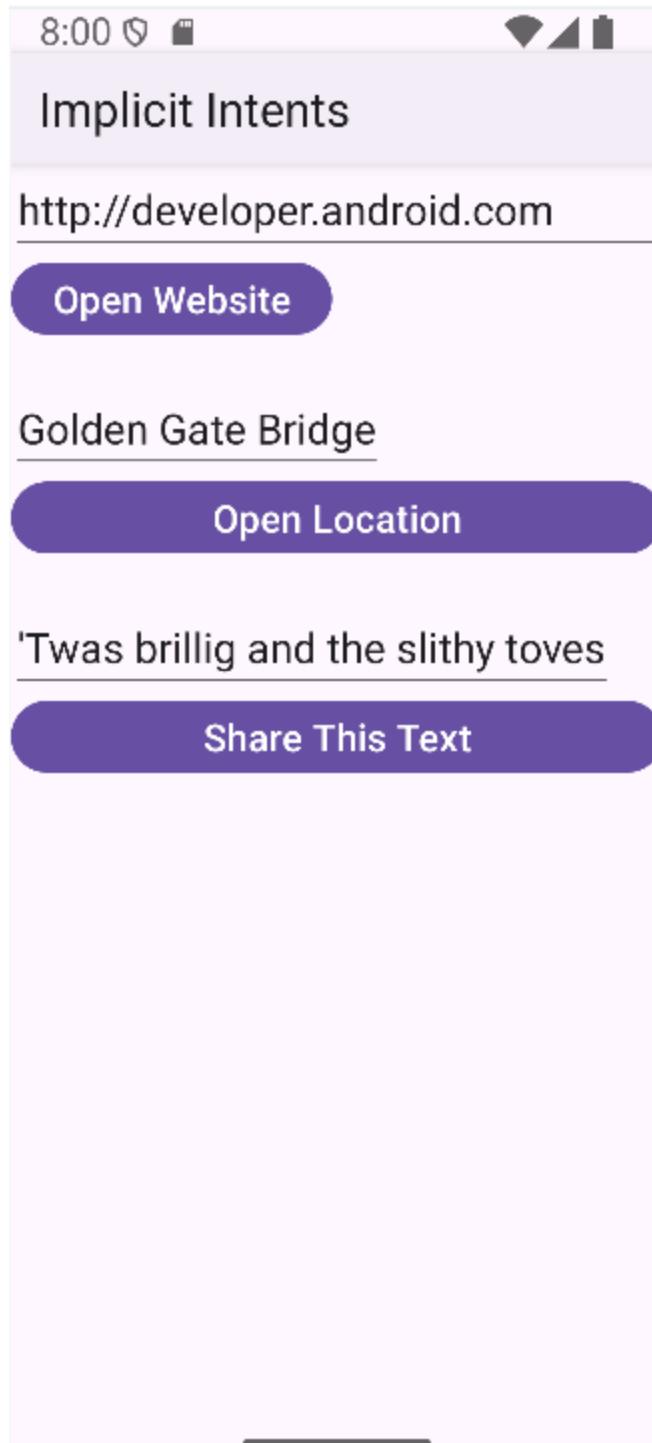
Lời gọi đến ShareCompat.IntentBuilder sử dụng các phương thức sau:

Method	Description
from()	The Activity that launches this share Intent (this).
setType()	The MIME type of the item to be shared.
setChooserTitle()	The title that appears on the system app chooser.
setText()	The actual text to be shared
startChooser()	Show the system app chooser and send the Intent.

Định dạng này, với tất cả các phương thức setter của builder được kết nối với nhau trong một câu lệnh, là một cách viết tắt dễ dàng để tạo và khởi chạy Intent. Bạn có thể thêm bất kỳ phương thức bổ sung nào vào danh sách này.

Phương thức shareText() bây giờ nên trông như sau:

```
public void shareText(View view) {  
    String txt = mShareEditText.getText().toString();  
    String mimeType = "text/plain";  
    ShareCompat.IntentBuilder  
        .from(launchingActivity: this)  
        .setType(mimeType)  
        .setChooserTitle("Share this text with:")  
        .setText(txt)  
        .startChooser();  
}
```



4.3 Chạy ứng dụng

1. Chạy ứng dụng.

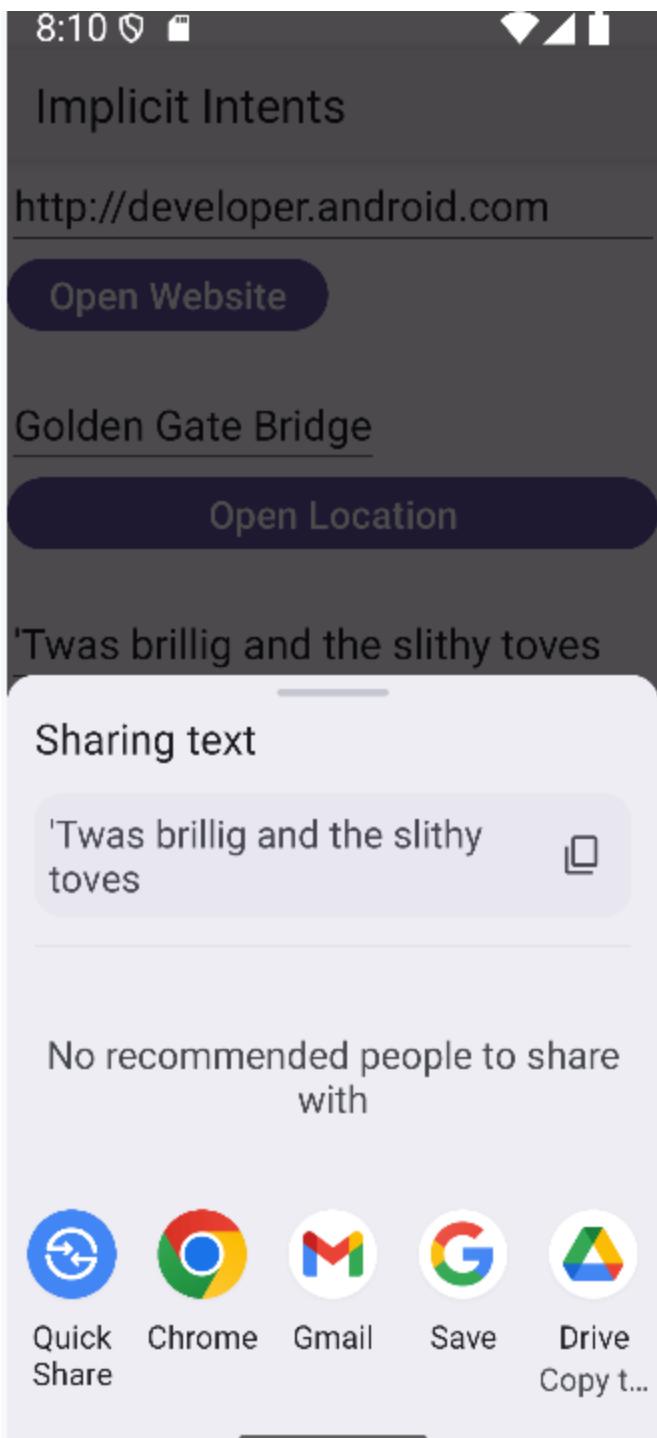
- Nhập vào nút **Open Website** để mở trình duyệt với URL trang web được nhập trong ô **EditText** phía trên nút **Button**. Trình duyệt và trang web sẽ hiển thị như hình minh họa bên dưới.



- Nhấn nút **Open Location** để mở bản đồ với vị trí được nhập trong **EditText** phía trên nút. Bản đồ hiển thị vị trí sẽ xuất hiện như hình dưới đây.



4. Nhấn nút **Share This Text** để mở hộp thoại với các tùy chọn chia sẻ văn bản. Hộp thoại hiển thị các tùy chọn sẽ xuất hiện như hình dưới đây.



Nhiệm vụ 5: Nhận một Intent ngầm định

Đến nay, bạn đã tạo một ứng dụng sử dụng một **Intent ngầm định** để khởi chạy **Activity** của ứng dụng khác. Trong nhiệm vụ này, bạn sẽ xem xét vấn đề từ góc độ ngược lại: cho phép một **Activity** trong ứng dụng của bạn phản hồi một **Intent ngầm định** được gửi từ ứng dụng khác.

Một Activity trong ứng dụng của bạn luôn có thể được kích hoạt từ bên trong hoặc bên ngoài ứng dụng bằng một Intent tường minh (explicit Intent).

Để cho phép một Activity nhận một Intent ngầm định (implicit Intent), bạn cần định nghĩa một **Intent filter** trong tệp **AndroidManifest.xml** của ứng dụng để chỉ ra loại Intent ngầm định nào mà Activity của bạn quan tâm xử lý.

Để kết nối yêu cầu của bạn với một ứng dụng cụ thể đã cài đặt trên thiết bị, hệ thống Android sẽ khớp Intent ngầm định của bạn với một Activity có Intent filter cho biết rằng Activity đó có thể thực hiện hành động đó. Nếu có nhiều ứng dụng cài đặt phù hợp, hệ thống sẽ hiển thị một bộ chọn ứng dụng (app chooser) để người dùng chọn ứng dụng 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 (action) và dữ liệu (data) của Intent đó với bất kỳ Activity nào có Intent filter phù hợp. Khi các Intent filter của một Activity khớp với Intent:

- Nếu chỉ có một Activity phù hợp, Android sẽ để Activity đó xử lý Intent.
- Nếu có nhiều Activity phù hợp, Android sẽ hiển thị một bộ chọn ứng dụng để người dùng chọn ứng dụng họ muốn thực thi hành động đó.

Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng rất đơn giản để nhận một Intent ngầm định mở URI của một trang web. Khi được kích hoạt bởi Intent ngầm định, ứng dụng đó sẽ hiển thị URI được yêu cầu dưới dạng một chuỗi trong một **TextView**.

5.1 Tạo dự án và bối cảnh

1. Tạo một dự án Android Studio mới với tên ứng dụng là **Implicit Intents Receiver** và chọn mẫu dự án **Empty Activity**.
2. Chấp nhận tên **Activity** mặc định (**MainActivity**). Nhấn **Next**.
3. Đảm bảo hộp kiểm **Generate Layout file** đã được chọn. Nhấn **Finish**.
4. Mở tệp **activity_main.xml**.
5. Trong **TextView** hiện có (hiển thị "Hello World"), xóa thuộc tính `android:text`. Mặc định **TextView** này sẽ không có văn bản, nhưng bạn sẽ thêm URI từ **Intent** trong phương thức `onCreate()`.
6. Giữ nguyên các thuộc tính `layout_constraint`, nhưng thêm các thuộc tính sau:

Attribute	Value
android:id	"@+id/text_uri_message"
android:textSize	"18sp"
android:textStyle	"bold"

5.2 Sửa đổi AndroidManifest.xml để thêm một Intent filter

1. Mở tệp **AndroidManifest.xml**.
2. Lưu ý rằng **MainActivity** đã có Intent filter sau:

```

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

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

```

Intent filter này, là một phần của manifest mặc định trong dự án, cho biết rằng Activity này là điểm khởi đầu chính của ứng dụng (với Intent action là "android.intent.action.MAIN") và rằng Activity này sẽ xuất hiện dưới dạng một mục cấp cao nhất trong trình khởi chạy (category là "android.intent.category.LAUNCHER").

3. Thêm một thẻ `<intent-filter>` thứ hai bên trong thẻ `<activity>`, và bao gồm các phần tử sau:

```

</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:host="developer.android.com"
    android:scheme="http" />

<intent-filter>

```

Các dòng này định nghĩa một Intent filter cho Activity, tức là loại Intent mà Activity có thể xử lý. Intent filter này khai báo các phần tử sau:

Loại (Type)	Giá trị (Value)	Khớp với (Matches)
action	"android.intent.action.VIEW"	Bất kỳ Intent nào có hành động là "view" (xem).
category	"android.intent.category.DEFAULT"	Bất kỳ Intent ngầm định nào. Category này phải được bao gồm để Activity của bạn nhận bất kỳ Intent ngầm định nào.
category	"android.intent.category.BROWSABLE"	Yêu cầu cho các liên kết duyệt web từ các trang web, email, hoặc nguồn khác.
data	android:scheme="http" android:host="developer.android.com"	Các URI chứa giao thức là http và tên máy chủ là developer.android.com .

Lưu ý rằng bộ lọc **data** có một giới hạn đối với cả loại liên kết mà nó sẽ chấp nhận và tên máy chủ cho các URI đó. Nếu bạn muốn bộ nhận (receiver) của mình có thể chấp nhận bất kỳ liên kết nào, bạn có thể bỏ qua phần tử <data>.

Phần **application** trong tệp **AndroidManifest.xml** bây giờ sẽ trông như sau:

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ImplicitIntentsReceiver"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </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:host="developer.android.com"
                android:scheme="http" />

            <intent-filter>
                </intent-filter>
        </activity>
    </application>

</manifest>
```

5.3 Xử lý Intent

Trong phương thức onCreate() của Activity, xử lý Intent đến để lấy bất kỳ dữ liệu hoặc thông tin bổ sung (extras) nào mà nó chứa. Trong trường hợp này, Intent ngầm định đến sẽ có URI được lưu trữ trong dữ liệu của Intent.

1. Mở tệp **MainActivity**.
2. Trong phương thức onCreate(), lấy Intent đến được sử dụng để kích hoạt Activity:

```
Intent intent = getIntent();
```

3. Lấy dữ liệu từ Intent. Dữ liệu trong Intent luôn là một đối tượng URI:

```
Uri data = intent.getData();
```

4. Kiểm tra để đảm bảo rằng biến uri không phải là null. Nếu kiểm tra đó thành công, tạo một chuỗi từ đối tượng URI:

```
if (data != null) {  
    String uri_string = "URI: " + data.toString();  
}
```

5. Trích xuất phần "URI: " vào một tài nguyên chuỗi (string resource) với tên **uri_label**.
6. Trong cùng khái if, lấy TextView để hiển thị thông báo:

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

7. Cũng trong khái if, đặt nội dung văn bản cho TextView thành URI:

```
textView.setText(uri_string);
```

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

```
package com.example.implicitintentsreceiver;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v,
        insets) -> {
            Insets systemBars =
            insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
            systemBars.bottom);
            return insets;
        });
        Intent intent = getIntent();
        Uri uri = intent.getData();
        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 ứng dụng **Implicit Intents Receiver** và **Implicit Intents** trên trình giả lập hoặc thiết bị của mình.

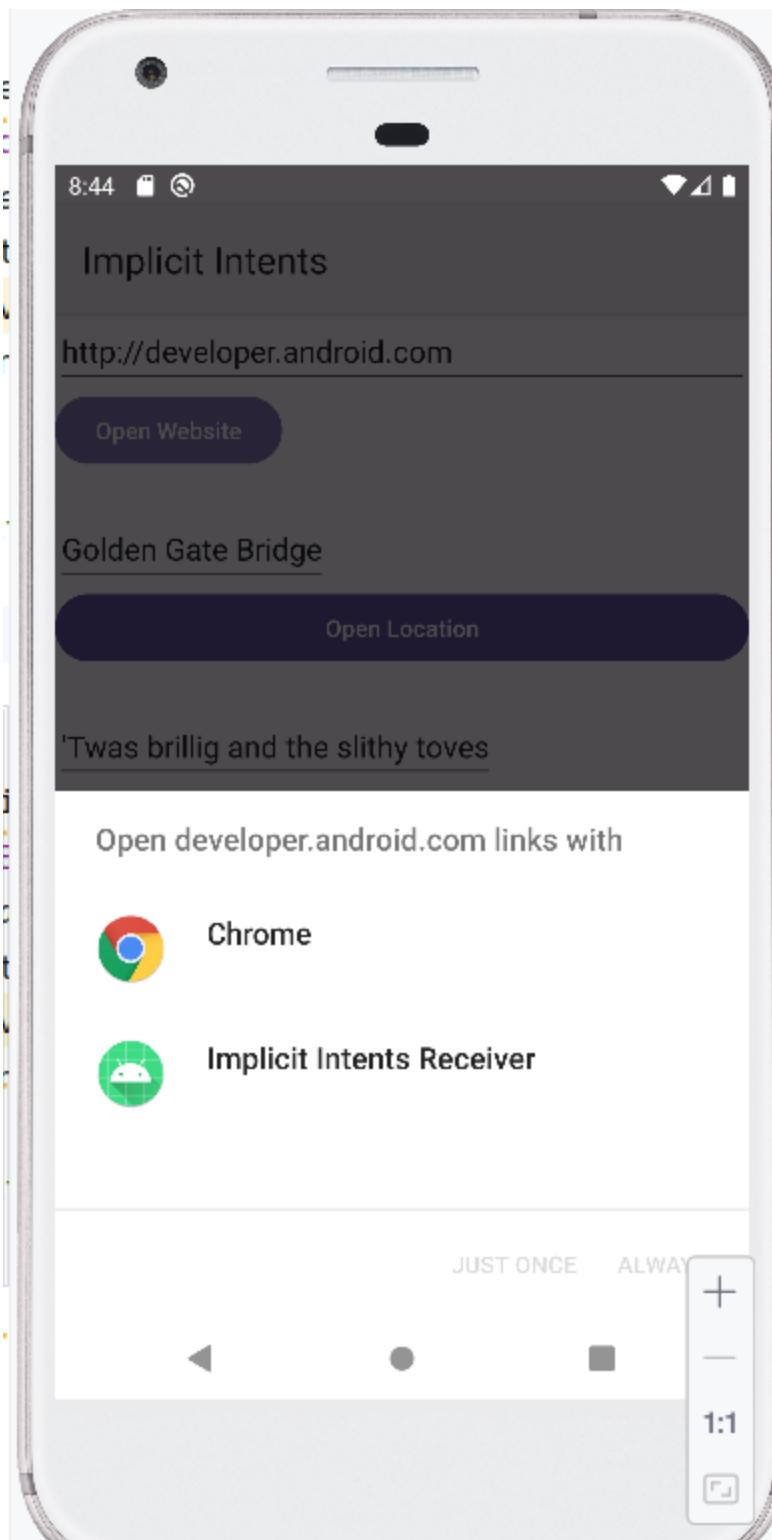
1. Chạy ứng dụng Implicit Intents Receiver.

Khi chạy ứng dụng này riêng lẻ, nó sẽ hiển thị một Activity trống mà không có văn bản. Điều này là do Activity được kích hoạt từ trình khởi chạy hệ thống, không phải bằng một Intent từ ứng dụng khác.

2. Chạy ứng dụng Implicit Intents và nhấn nút Open Website với URI mặc định.

Một hộp thoại chọn ứng dụng xuất hiện, hỏi bạn có muốn sử dụng trình duyệt mặc định (Chrome trong hình minh họa) hay ứng dụng **Implicit Intents Receiver**.

- Chọn **Implicit Intents Receiver**, sau đó nhấn **Just Once**.
- Ứng dụng **Implicit Intents Receiver** sẽ được khởi chạy, và thông báo sẽ hiển thị URI từ yêu cầu ban đầu



3. Nhấn nút Back và nhập một URI khác. Nhấn nút Open Website.

Ứng dụng receiver có bộ lọc Intent rất hạn chế, chỉ khớp với giao thức URI chính xác (<http://>) và host (developer.android.com). Bất kỳ URI nào khác sẽ được mở bằng trình duyệt web mặc định.

Mã giải pháp cho Task 5

Dự án Android Studio: **ImplicitIntentsReceiver**

Thử thách lập trình

Ghi chú: Tất cả các thử thách lập trình đều tùy chọn và không phải là điều kiện tiên quyết cho các bài học sau.

Thử thách:

Trong một thử thách thực hành trước, bạn đã tạo một ứng dụng danh sách mua sắm với một Activity để hiển thị danh sách và một Activity khác để chọn một mục.

Hãy thêm một **EditText** và một **Button** vào Activity danh sách mua sắm để định vị một cửa hàng cụ thể trên bản đồ.

Tóm tắt

- **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 Intent ngầm định phải định nghĩa bộ lọc Intent trong tệp **AndroidManifest.xml** để khớp với một hoặc nhiều hành động (action) và danh mục (category) của Intent.
- Hệ thống Android sẽ so khớp nội dung của Intent ngầm định và bộ lọc Intent của bất kỳ Activity có sẵn nào để xác định Activity nào sẽ được kích hoạt. Nếu có nhiều hơn một Activity có sẵn, hệ thống sẽ cung cấp một hộp chọn để người dùng chọn một.
- Lớp **ShareCompat.IntentBuilder** giúp việc tạo một Intent ngầm định để chia sẻ dữ liệu lên mạng xã hội hoặc email trở nên dễ dàng.

Liên quan đến khái niệm

Tài liệu về khái niệm liên quan nằm trong **2.3: Implicit intents (Intent ngầm định)**.

Tìm hiểu thêm

Tài liệu phát triển Android:

- **Application Fundamentals** (**Nguyên tắc cơ bản của ứng dụng**)
- **Activities** (**Hoạt động**)
- **Understand the Activity Lifecycle** (**Hiểu vòng đời của Activity**)
- **Intents and Intent Filters** (**Intent và Bộ lọc Intent**)
- **Allowing Other Apps to Start Your Activity** (**Cho phép ứng dụng khác khởi chạy Activity của bạn**)
- **Google Maps Intents for Android** (**Intent Google Maps cho Android**)
- **Activity**
- **Intent**
- **<intent-filter>**
- **<activity>**
- **Uri**
- **ShareCompat.IntentBuilder**

Bài tập về nhà

Xây dựng và chạy một ứng dụng

Mở ứng dụng **ImplicitIntents** mà bạn đã tạo.

1. Thêm một nút khác ở cuối màn hình.
2. Khi nhấn vào nút này, hãy khởi chạy một ứng dụng camera để chụp ảnh.
(Bạn không cần trả lại ảnh về ứng dụng ban đầu).

Lưu ý:

Nếu bạn sử dụng trình giả lập Android để kiểm tra camera, hãy mở cấu hình trình giả lập trong **Android AVD Manager**, chọn **Advanced Settings** và sau đó chọn **Emulated** cho cả camera trước và sau. Khởi động lại trình giả lập nếu cần thiết.

Trả lời các câu hỏi

Câu hỏi 1

Phương thức constructor nào bạn sử dụng để tạo một implicit Intent để mở ứng dụng camera?

- new Intent()
- new Intent(Context context, Class<?> class)
- new Intent(String action, Uri uri)
- new Intent(String action)

Câu hỏi 2

Khi bạn tạo một đối tượng implicit Intent, điều nào sau đây là đúng?

- Không cần chỉ định Activity hoặc thành phần cụ thể nào để mở.
 - Thêm một hành động Intent hoặc các danh mục Intent (hoặc cả hai).
 - Giải quyết Intent với hệ thống trước khi gọi startActivity() hoặc startActivityForResult().
 - Tất cả các đáp án trên.
-

Câu hỏi 3

Hành động Intent nào bạn sử dụng để chụp ảnh bằng ứng dụng camera?

- Intent takePicture = new Intent(Intent.ACTION_VIEW);
 - Intent takePicture = new Intent(Intent.ACTION_MAIN);
 - Intent takePicture = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
 - Intent takePicture = new Intent(Intent.ACTION_GET_CONTENT);
-

Nộp ứng dụng của bạn để được chấm điểm

Hướng dẫn cho người chấm điểm:

Kiểm tra rằng ứng dụng có các tính năng sau:

- Ứng dụng hiển thị nút **Take a Picture** ở cuối màn hình.
- Khi được nhấn, nút này mở ứng dụng camera trên thiết bị.
- **Trước khi gửi intent**, phương thức onClick() dành cho nút **Take a Picture** đảm bảo rằng trên thiết bị có sẵn một ứng dụng có thể xử lý intent này. Điều này được thực hiện bằng cách sử dụng các phương thức resolveActivity() và getPackageManager().

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 ra nhật ký hệ thống, thông tin này sẽ xuất hiện ở bảng Logcat trong Android Studio khi ứng dụng của bạn chạy. Thêm các câu lệnh ghi log vào ứng dụng là một cách để tìm lỗi và cải thiện hoạt động của ứng dụng. Một cách khác là sử dụng trình gỡ lỗi (debugger) được tích hợp trong Android Studio.

Trong bài thực hành này, bạn sẽ học cách gỡ lỗi ứng dụng trên trình giả lập hoặc thiết bị thật, đặt và xem điểm dừng (breakpoints), từng bước thực hiện mã lệnh và kiểm tra các biến.

Những gì bạn cần biết trước

Bạn cần có khả năng:

- Tạo một dự án Android Studio.
 - Sử dụng trình chỉnh sửa giao diện (layout editor) để làm việc với các thành phần EditText và Button.
 - Xây dựng và chạy ứng dụng của bạn trong Android Studio, trên trình giả lập và thiết bị thật.
 - Đọc và phân tích dấu vết ngăn xếp (stack trace), bao gồm cách hiểu thứ tự **last on, first off**.
 - Thêm các câu lệnh ghi log và xem nhật ký hệ thống (Logcat) trong Android Studio.
-

Những gì bạn sẽ học

- Cách chạy ứng dụng ở chế độ gỡ lỗi trên trình giả lập hoặc thiết bị thật.
- Cách thực hiện từng bước mã lệnh khi ứng dụng chạy.
- Cách đặt và tổ chức các điểm dừng (breakpoints).
- Cách kiểm tra và thay đổi giá trị của các biến trong trình gỡ lỗi.

Những gì bạn sẽ làm

- Xây dựng ứng dụng SimpleCalc.
 - Đặt và xem các điểm dừng trong mã nguồn của ứng dụng SimpleCalc.
 - Từng bước thực hiện mã lệnh khi ứng dụng chạy.
 - Kiểm tra các biến và đánh giá các biểu thức.
 - Xác định và sửa lỗi trong ứng dụng mẫu.
-

Tổng quan ứng dụng

Ứng dụng SimpleCalc có hai thành phần EditText và bốn nút Button. Khi bạn nhập hai số và nhấn một nút, ứng dụng sẽ thực hiện phép tính tương ứng với nút đó và hiển thị kết quả.

Nhiệm vụ 1: Khám phá dự án và ứng dụng SimpleCalc

Trong thực hành này, bạn sẽ không tự xây dựng ứng dụng SimpleCalc. Dự án hoàn chỉnh đã sẵn sàng tại **SimpleCalc**. Trong nhiệm vụ 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 file.
2. Mở **Android Studio** và chọn **File > Open**.
3. Điều hướng đến thư mục chứa SimpleCalc, chọn thư mục đó và nhấn **OK**.
Dự án SimpleCalc sẽ được xây dựng.
4. Mở **Project > Android** nếu nó chưa được mở.

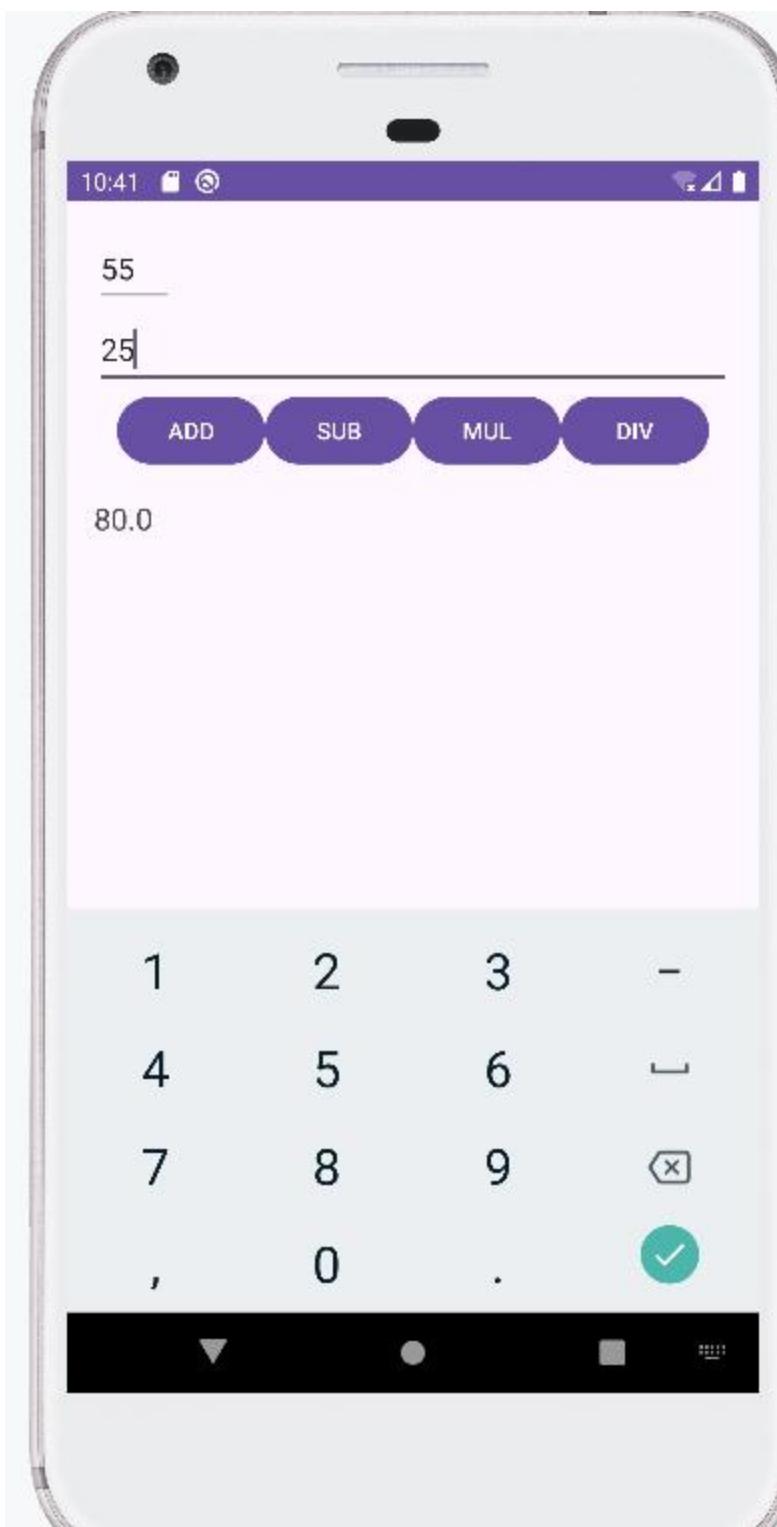
Cảnh báo: Ứng dụng này có chứa lỗi mà bạn sẽ tìm và sửa chữa. Nếu bạn chạy ứng dụng trên thiết bị hoặc trình giả lập, có thể gặp phải hành vi bất thường, bao gồm cả sự cố ứng dụng.

1.2 Khám phá bố cục (Layout)

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

Khám phá mã XML bố cục và thiết kế, lưu ý các điểm sau:

- Bố cục chứa hai phần tử **EditText** để nhập liệu, bốn nút **Button** cho các phép tính và một **TextView** để hiển thị kết quả.
- Mỗi nút tính toán (**Button**) có trình xử lý sự kiện nhấp riêng thông qua thuộc tính **android:onClick** (như `onAdd`, `onSub`, v.v.).
- **TextView** dùng để hiển thị kết quả không có bất kỳ văn bản nào theo mặc định.
- Hai phần tử **EditText** có thuộc tính **android:inputType** với giá trị "`numberDecimal`". Thuộc tính này cho biết **EditText** chỉ chấp nhận đầu vào là số. Bàn phím hiển thị trên màn hình sẽ chỉ chứa các số. Bạn sẽ tìm hiểu thêm về kiểu nhập liệu (**input types**) cho phần tử **EditText** trong các thực hành sau.



1.3 Khám phá mã nguồn ứng dụng

1. Mở rộng thư mục **app > java** trong **Project > Android**. Ngoài lớp **MainActivity**, dự án này còn bao gồm lớp tiện ích **Calculator**.

2. Mở **Calculator** và xem xét mã nguồn. Lưu ý rằng các phép toán mà máy tính có thể thực hiện được định nghĩa bởi **enum Operator**, và tất cả các phương thức thực hiện phép toán đều có phạm vi **public**.
 3. Mở **MainActivity** và xem xét mã nguồn cùng các chú thích.
Lưu ý những điểm sau:
 - Tất cả các trình xử lý sự kiện android:onClick được định nghĩa đều gọi phương thức riêng tư compute(), với tên phép toán là một trong các giá trị của Calculator.Operator enumeration.
 - Phương thức compute() gọi phương thức riêng tư getOperand() (mà tiếp tục gọi getOperandText()) để lấy giá trị số từ các thành phần EditText.
 - Phương thức compute() sau đó sử dụng câu lệnh switch trên tên phép toán để gọi phương thức thích hợp trong đối tượng Calculator (được biểu diễn bởi mCalculator).
 - Các phương thức tính toán trong lớp Calculator thực hiện các phép toán 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

1. Chạy ứng dụng và làm theo các bước sau:
 - Nhập cả giá trị số nguyên và số thực cho phép tính.
 - Nhập các giá trị số thực với các phần thập phân lớn (ví dụ: 1.6753456).
 - Thực hiện phép chia một số cho 0.
 - Để trống một hoặc cả hai thành phần EditText và thử bất kỳ phép tính nào.
2. Nhấp vào tab **Logcat** ở cuối cửa sổ Android Studio để mở khung **Logcat** (nếu chưa được mở).
3. Kiểm tra dấu vết ngăn xếp (stack trace) tại điểm ứng dụng báo lỗi.
 - Nếu một hoặc cả hai thành phần EditText trong SimpleCalc để trống, ứng dụng sẽ báo lỗi **exception**, như minh họa trong hình dưới đây.
 - Nhật ký hệ thống (log) sẽ hiển thị trạng thái ngăn xếp thực thi (execution stack) tại thời điểm ứng dụng tạo ra lỗi.

- Stack trace thường cung cấp thông tin quan trọng về lý do xảy ra lỗi.

2.1 Bắt đầu và chạy ứng dụng trong chế độ gỡ lỗi

1. Trong Android Studio, chọn **Run > Debug app** hoặc nhấp vào biểu tượng **Debug** trên thanh công cụ.
 2. Nếu ứng dụng của bạn đang chạy, hệ thống sẽ hỏi bạn có muốn khởi động lại ứng dụng trong chế độ gỡ lỗi hay không. Nhấp vào **Restart app** (Khởi động lại ứng dụng).
- Android Studio sẽ biên dịch và chạy ứng dụng của bạn trên trình giả lập hoặc trên thiết bị.
- Gỡ lỗi (debugging) đều hoạt động giống nhau trên cả hai trường hợp.
 - Trong khi Android Studio đang khởi tạo debugger, bạn có thể thấy một thông báo trên thiết bị hoặc trình giả lập có nội dung: "**Waiting for debugger**" trước khi bạn có thể sử dụng ứng dụng.
3. Nhấp vào tab **Debug** ở cuối cửa sổ Android Studio để mở bảng điều khiển **Debug** (hoặc chọn **View > Tool Windows > Debug**). Tab **Debugger** sẽ tự động được chọn và hiển thị bảng điều khiển Debugger.

2.2 Đặt một điểm dừng (breakpoint)

Một điểm dừng (breakpoint) là một vị trí trong mã nguồn nơi bạn muốn tạm dừng việc thực thi bình thường của ứng dụng để thực hiện các hành động khác, chẳng hạn như: kiểm tra giá trị của các biến, đánh giá các biểu thức hoặc thực thi mã theo từng dòng để xác định nguyên nhân của các lỗi trong quá trình chạy. Bạn có thể đặt điểm dừng trên bất kỳ dòng mã nào có thể thực thi.

1. Mở tệp **MainActivity**, nhấp vào dòng thứ tư của phương thức `compute()` (ngay sau câu lệnh `try`).
2. Nhấp vào phần lề bên trái cạnh số dòng. Một chấm đỏ xuất hiện, biểu thị rằng một breakpoint đã được đặt. Nếu ứng dụng đang chạy ở chế độ debug, chấm đỏ sẽ có thêm dấu kiểm ().

Ngoài ra, bạn có thể chọn **Run > Toggle Line Breakpoint** hoặc nhấn **Control-F8** (hoặc **Command-F8** trên Mac) để đặt hoặc xóa breakpoint.

- Đặt nhầm có thể được sửa bằng cách nhập lại vào chấm đỏ.
 - Nếu dòng mã không thể thực thi, chấm đỏ sẽ có thêm dấu "x" và hiển thị cảnh báo.
3. Trong ứng dụng **SimpleCalc**, nhập số vào các ô **EditText** và nhấn một nút tính toán.

Việc thực thi ứng dụng của bạn sẽ dừng lại khi đạt đến điểm dừng (breakpoint) mà bạn đã đặt, và trình gỡ lỗi (debugger) sẽ hiển thị trạng thái hiện tại của ứng dụng tại điểm dừng đó, như được minh họa trong hình dưới đây.

Hình minh họa hiển thị **ngăn Debug** với các tab **Debugger** và **Console**. Tab **Debugger** được chọn, hiển thị ngăn Debugger với các tính năng sau:

1. **Tab Frames:** Nhấp để hiển thị ngăn **Frames** với các khung ngăn xếp thực thi (execution stack frames) hiện tại cho một luồng (thread) cụ thể. Ngăn xếp thực thi hiển thị từng lớp (class) và phương thức (method) đã được gọi trong ứng dụng của bạn và trong thời gian chạy Android (Android runtime), với phương thức được gọi gần nhất ở trên cùng.
 - Nhấp vào tab **Threads** để thay thế ngăn **Frames** bằng ngăn **Threads**. Ứng dụng của bạn hiện đang chạy trong luồng chính (main thread) và đang thực thi phương thức **compute()** trong lớp **MainActivity**.
2. **Nút Watches:** Nhấp để hiển thị ngăn **Watches** bên trong ngăn **Variables**, nơi hiển thị giá trị của bất kỳ biến nào mà bạn đã đặt để theo dõi (watches). Tính năng Watches cho phép bạn theo dõi một biến cụ thể trong chương trình và xem cách biến đó thay đổi khi chương trình chạy.
3. **Ngăn Variables:** Hiển thị các biến trong phạm vi hiện tại và giá trị của chúng. Tại giai đoạn này của việc thực thi ứng dụng, các biến có sẵn bao gồm: **this** (đại diện cho Activity), **operator** (tên toán tử từ **Calculator.Operator** mà phương thức được gọi từ đó), cũng như các biến toàn cục cho các phần tử **EditText** và **TextView**. Mỗi biến trong ngăn này có biểu tượng mở rộng để bạn có thể mở rộng danh sách các thuộc tính của đối tượng cho biến đó. Hãy thử mở rộng một biến để khám phá các thuộc tính của nó.

2.3 Tiếp tục thực thi ứng dụng của bạn

Tiếp tục thực thi ứng dụng của bạn bằng cách chọn **Run > Resume Program**, hoặc nhập vào biểu tượng **Resume** ở phía bên trái của cửa sổ trình gỡ lỗi.

Ứng dụng **SimpleCalc** sẽ tiếp tục chạy, và bạn có thể tương tác với ứng dụng cho đến khi quá trình thực thi mã tiếp theo đạt đến điểm dừng (breakpoint).

2.4 Gỡ lỗi một ứng dụng đang chạy

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

1. Chạy ứng dụng **SimpleCalc** bình thường bằng cách nhấp vào biểu tượng **Run**.
2. Chọn **Run > Attach debugger to Android process**, hoặc nhấp vào biểu tượng **Attach** trên thanh công cụ.
3. Chọn tiến trình (process) của ứng dụng bạn từ hộp thoại xuất hiện (như hình minh họa bên dưới). Nhấp vào **OK**.

Cửa sổ Debug xuất hiện với phần Debugger được mở, và bây giờ bạn có thể gỡ lỗi ứng dụng của mình như thể bạn đã khởi chạy nó trong chế độ debug.

Lưu ý: Nếu cửa sổ Debug không tự động xuất hiện, nhấp vào tab **Debug** ở cuối màn hình. Nếu tab **Debugger** chưa được chọn, nhấp vào tab **Debugger** trong cửa sổ **Debug** để hiển thị phần Debugger.

Nhiệm vụ 3: Khám phá các tính năng của trình gỡ lỗi

Trong nhiệm vụ này, chúng ta sẽ khám phá các tính năng khác nhau trong trình gỡ lỗi của Android Studio, bao gồm thực thi ứng dụng từng dòng, làm việc với các điểm dừng (breakpoints), và kiểm tra các biến.

3.1 Thực hiện từng bước khi ứng dụng chạy

Sau khi dừng tại điểm dừng (breakpoint), bạn có thể sử dụng trình gỡ lỗi để thực thi từng dòng mã trong ứng dụng và kiểm tra trạng thái của các biến khi ứng dụng chạy.

1. Gỡ lỗi ứng dụng trong Android Studio với điểm dừng mà bạn đã đặt ở nhiệm vụ trước.
2. Trong ứng dụng, nhập số vào cả hai trường **EditText**, sau đó nhấp vào nút **Add**.
 - Việc thực thi ứng dụng sẽ dừng lại tại điểm dừng mà bạn đã đặt trước đó, và cửa sổ **Debugger** sẽ hiển thị trạng thái hiện tại của ứng dụng. Dòng mã hiện tại sẽ được đánh dấu nổi bật trong mã nguồn của bạn.
3. Nhấp vào nút **Step Over** ở đầu cửa sổ trình gỡ lỗi.
 - Trình gỡ lỗi sẽ thực thi dòng mã hiện tại trong phương thức **compute()** (nơi bạn đặt điểm dừng, dòng gán giá trị cho biến **operandOne**), và điểm đánh dấu sẽ chuyển sang dòng tiếp theo trong mã nguồn (dòng gán giá trị cho biến **operandTwo**).

- Cửa sổ **Variables** sẽ cập nhật để phản ánh trạng thái thực thi mới, và các giá trị hiện tại của biến sẽ xuất hiện dưới dạng chữ nghiêng sau mỗi dòng mã trong mã nguồn.
 - Bạn cũng có thể sử dụng **Run > Step Over** hoặc nhấn **F8** để thực hiện lệnh này.
4. Tại dòng tiếp theo (dòng gán giá trị cho `operandTwo`), nhấp vào biểu tượng **Step Into**.
- **Step Into** nhảy vào việc thực thi một lời gọi phương thức trong dòng hiện tại (so với việc chỉ thực thi phương thức đó và giữ nguyên tại dòng hiện tại). Trong trường hợp này, do dòng lệnh bao gồm lời gọi tới phương thức `getOperand()`, trình gõ lỗi sẽ cuộn tới định nghĩa của phương thức `getOperand()` trong mã nguồn của **MainActivity**.
 - Khi bạn nhảy vào một phương thức, cửa sổ **Frames** sẽ cập nhật để hiển thị khung mới trong ngăn xếp lệnh gọi (ở đây là `getOperand()`), và cửa sổ **Variables** hiển thị các biến có sẵn trong phạm vi phương thức mới. Bạn có thể nhấp vào bất kỳ dòng nào trong cửa sổ **Frames** để xem điểm trong khung lệnh gọi trước đó nơi phương thức được gọi.

Bạn cũng có thể sử dụng **Run > Step Into** hoặc phím **F7** để nhảy vào một phương thức.

1. Nhấp vào **Step Over** để thực thi từng dòng trong phương thức `getOperand()`.
 - Lưu ý rằng khi phương thức hoàn thành, trình gõ lỗi sẽ đưa bạn quay lại điểm mà bạn đã nhảy vào phương thức ban đầu, và tất cả các bảng điều khiển sẽ được 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 phù hợp được định nghĩa trong lớp **Calculator**, mở tệp **Calculator.java**, và cuộn đến điểm thực thi trong lớp đó. Một lần nữa, các bảng điều khiển khác nhau sẽ được 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 quay trở lại phương thức `compute()` trong **MainActivity**. Bạn có thể tiếp tục gõ lỗi phương thức `compute()` từ điểm mà bạn đã dừng trước đó.
 - Bạn cũng có thể sử dụng **Run > Step Out** hoặc nhấn **Shift-F8** để thoát khỏi việc thực thi một phương thức.

3.2 Làm việc với Breakpoint

Sử dụng breakpoint để chỉ định vị trí trong mã mà bạn muốn tạm dừng thực thi ứng dụng để gỡ lỗi phần đó của ứng dụng.

1. Tìm breakpoint mà bạn đã đặt trong tác vụ trước—ở phần bắt đầu của phương thức **compute()** trong **MainActivity**.
2. Thêm một breakpoint vào dòng bắt đầu của câu lệnh **switch**.
3. Nhấp chuột phải vào breakpoint mới đó để nhập một điều kiện, như trong hình minh họa bên dưới, và nhập thử nghiệm sau vào trường **Condition**:

CopyEdit

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

4. Nhấn **Done** (Hoàn tất).

Breakpoint thứ hai này là một **breakpoint có điều kiện** (*conditional breakpoint*). Việc thực thi ứng dụng của bạn chỉ dừng lại tại breakpoint này nếu kiểm tra điều kiện trong biểu thức là **true**. Trong trường hợp này, biểu thức chỉ đúng nếu một trong hai toán hạng bạn nhập là **42**. Bạn có thể nhập bất kỳ biểu thức Java nào làm điều kiện miễn là nó trả về một giá trị **boolean**.

5. Chạy ứng dụng của bạn ở chế độ gỡ lỗi (**Run > Debug**) hoặc nhấn **Resume** nếu ứng dụng đã chạy. Trong ứng dụng, nhập hai số khác **42** và nhấp vào nút **Add**. Việc thực thi sẽ dừng lại tại breakpoint đầu tiên trong phương thức **compute()**.
 6. Nhấn **Resume** để tiếp tục gỡ lỗi ứng dụng. Quan sát rằng việc thực thi không dừng lại ở breakpoint thứ hai của bạn vì điều kiện không được đáp ứng.
 7. Trong ứng dụng, nhập **42** vào ô **EditText** đầu tiên và nhấn bất kỳ nút nào. Nhấn **Resume** để tiếp tục thực thi sau breakpoint đầu tiên. Quan sát rằng breakpoint thứ hai tại câu lệnh **switch**—breakpoint có điều kiện—dừng việc thực thi vì điều kiện đã được đáp ứng.
 8. Nhấp chuột phải (hoặc **Control-click**) vào breakpoint đầu tiên trong phương thức **compute()** và bỏ chọn **Enabled**. Nhấn **Done**. Quan sát rằng biểu tượng breakpoint giờ có một chấm xanh với viền đỏ.
- Việc vô hiệu hóa một breakpoint cho phép bạn tạm thời "tắt tiếng" breakpoint đó mà không thực sự xóa nó khỏi mã của bạn. Nếu bạn xóa hoàn toàn một breakpoint, bạn cũng sẽ mất bất kỳ điều kiện nào đã tạo cho breakpoint đó, vì vậy việc vô hiệu hóa thường là lựa chọn tốt hơn.

Bạn cũng có thể tắt tất cả các breakpoint trong ứng dụng cùng lúc bằng biểu tượng **Mute Breakpoints**.

9. Nhấp vào **View Breakpoints** ở cạnh trái của cửa sổ debugger. Cửa sổ **Breakpoints** xuất hiện.

Cửa sổ **Breakpoints** cho phép bạn xem tất cả các breakpoint trong ứng dụng, bật hoặc tắt từng breakpoint, và thêm các tính năng bổ sung cho breakpoint bao gồm các điều kiện, phụ thuộc vào các breakpoint khác và ghi nhật ký (*logging*).

Để đóng cửa sổ **Breakpoints**, nhấn **Done**.

3.3 Kiểm tra và thay đổi biến

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

1. Chạy ứng dụng **SimpleCalc** ở chế độ debug nếu nó chưa chạy.
2. Trong ứng dụng, nhập hai số, một trong số đó là **42**, sau đó nhấn nút **Add**. Breakpoint đầu tiên trong phương thức **compute()** hiện đang bị tắt. Việc thực thi dừng lại tại breakpoint thứ hai (breakpoint có điều kiện ở câu lệnh **switch**), và trình gõ lỗi sẽ xuất hiện.
3. Quan sát trong bảng **Variables** rằng các biến **operandOne** và **operandTwo** đã nhận giá trị mà bạn đã nhập vào ứng dụng.
4. Biến **this** là một đối tượng thuộc lớp **MainActivity**. Nhấp vào biểu tượng mở rộng để xem danh sách các biến thành viên của đối tượng đó. Sau đó, nhấp lại vào biểu tượng mở rộng để đóng danh sách.
5. Nhấp chuột phải (hoặc **Control-click**) vào biến **operandOne** trong bảng **Variables**, và chọn **Set Value**.
6. Thay đổi giá trị của biến **operandOne** thành **10** và nhấn **Return**.
7. Thay đổi giá trị của biến **operandTwo** thành **10** theo cách tương tự và nhấn **Return**.
8. Quan sát rằng kết quả trong ứng dụng bây giờ dựa trên các giá trị biến mà bạn đã thay đổi trong trình gõ lỗi; ví dụ, vì bạn đã nhấn nút **Add** ở Bước 2, kết quả trong ứng dụng 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**) vẫn được giữ nguyên trong các trường **EditText**. (Các giá trị này chỉ được thay đổi trong trình gõ

lỗi.) Nhấn lại nút **Add**. Quá trình thực thi sẽ dừng lại tại điểm breakpoint một lần nữa.

11.Nhấn 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 đoạn mã nào vào cửa sổ này.

12.Nhập câu lệnh **mOperandOneEditText.getHint()**; vào trường phía trên của cửa sổ **Evaluate Code Fragment** (như minh họa trong hình trên) và nhấn **Evaluate**.

13.Trường **Result** hiển thị kết quả của biểu thức đó. Gợi ý (hint) cho trường **EditText** này là chuỗi "**Type Operand 1**", như đã được định nghĩa ban đầu trong tệp XML cho trường **EditText** này.

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

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

14. Nhấn **Close** để đóng cửa sổ **Evaluate Code Fragment**.

Thách thức lập trình

Lưu ý: Tất cả các thách thức lập trình đều tùy chọn và không phải là điều kiện tiên quyết cho các bài học sau.

Thách thức: Ở cuối Bài 1, bạn đã thử chạy ứng dụng **SimpleCalc** mà không nhập giá trị vào một trong các trường **EditText**, dẫn đến lỗi. Sử dụng trình gõ lỗi để từng bước thực thi mã và xác định chính xác lý do gây ra lỗi này. Khắc phục lỗi gây ra sự cố này.

Tóm tắt:

- Xem thông tin ghi log trong Android Studio bằng cách nhấp vào tab **Logcat**.
- Chạy ứng dụng của bạn trong chế độ gõ lỗi bằng cách nhấp vào biểu tượng **Debug** hoặc chọn **Run > Debug app**.

- Nhấp vào tab **Debug** để hiển thị khung **Debug**. Nhấp vào tab **Debugger** trong khung **Debug** để hiển thị khung **Debugger** (nếu nó chưa được chọn).
- Khung **Debugger** hiển thị các **Frames** (ngăn xếp), **Variables** trong một frame cụ thể và **Watches** (theo dõi hoạt động của một biến khi chương trình chạy).
- **Breakpoint** là một điểm 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 một điểm dừng gỡ lỗi bằng cách nhấp vào lề bên trái của cửa sổ trình soạn thảo ngay cạnh dòng mục tiêu.

Khái niệm liên quan:

Tài liệu khái niệm liên quan nằm trong mục **3.1: Trình gỡ lỗi của Android Studio.**

Tìm hiểu thêm

Tài liệu Android Studio:

- Hướng dẫn sử dụng Android Studio
- Gỡ lỗi ứng dụng của bạn
- Ghi và xem log
- Phân tích Stack Trace
- Cầu nối gỡ lỗi Android (ADB)
- Công cụ phân tích Android Profiler
- Trình phân tích mạng (Network Profiler)
- Trình phân tích CPU (CPU Profiler)
- Traceview

Khác:

- Video: Gỡ lỗi và kiểm thử trong Android Studio

Bài tập về nhà

Xây dựng và chạy ứng dụng

Mở ứng dụng SimpleCalc.

1. Trong **MainActivity**, đặt một breakpoint ở dòng đầu tiên của phương thức **onAdd()**.
2. Chạy ứng dụng trong chế độ gỡ lỗi. Thực hiện một phép tính cộng trong ứng dụng. Quá trình thực thi dừng lại tại breakpoint.

3. Nhấp vào **Step Into** để theo dõi quá trình thực thi của ứng dụng từng bước. Lưu ý rằng **Step Into** sẽ mở và thực thi các tệp từ framework Android, cho phép bạn xem cách Android hoạt động với mã của bạn.
4. Quan sát cách khung **Debug** thay đổi khi bạn bước qua mã cho frame ngắn xếp hiện tại và các biến cục bộ.
5. Quan sát cách mã trong khung trình soạn thảo được chú thích khi mỗi dòng được thực thi.
6. Nhấp vào **Step Out** để quay lại ứng dụng của bạn nếu ngăn xếp thực thi trở nên quá sâu để hiểu.

Câu hỏi 1

Chạy ứng dụng **SimpleCalc** mà không sử dụng trình gõ lỗi. Để trả lời một hoặc cả hai trường **EditText**, sau đó thử thực hiện phép tính bất kỳ. Tại sao lỗi xảy ra?

- **java.lang.NumberFormatException: empty String**
- **W/OpenGLO Renderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED**
- **Ứng dụng có thể đang thực hiện quá nhiều công việc trên luồng chính.**
- **Dung lượng bộ nhớ cache của mã đã được tăng lên 128KB.**

Câu hỏi 2

Chức năng nào bạn thực hiện trong bảng **Debug** để thực thi dòng hiện tại nơi đặt breakpoint, sau đó dừng ở dòng tiếp theo trong mã? Hãy chọn một:

- **Step Into**
- **Step Over**
- **Step Out**
- **Resume**

Câu hỏi 3

Chức năng nào bạn thực hiện trong bảng **Debug** để nhảy tới thực thi một phương thức được gọi từ dòng hiện tại nơi đặt breakpoint? Hãy chọn một:

- **Step Into**
- **Step Over**
- **Step Out**
- **Resume**

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Không có ứng dụng nào cần gửi cho bài tập về nhà này.

3.2) Kiểm tra đơn vị

Giới thiệu

Kiểm thử mã của bạn có thể giúp bạn phát hiện lỗi sớm trong quá trình phát triển, khi chi phí xử lý lỗi là thấp nhất. Khi ứng dụng của bạn trở nên lớn hơn và phức tạp hơn, việc kiểm thử cải thiện độ tin cậy của mã.

Với các bài kiểm thử trong mã, bạn có thể kiểm tra từng phần nhỏ trong ứng dụng một cách độc lập, và bạn có thể kiểm thử theo cách tự động hóa và lặp lại được.

Android Studio và **Android Testing Support Library** hỗ trợ nhiều loại kiểm thử và khung kiểm thử khác nhau. Trong bài thực hành này, bạn sẽ khám phá chức năng kiểm thử tích hợp sẵn của Android Studio và học cách viết và chạy các bài kiểm thử đơn vị cục bộ.

Kiểm thử đơn vị cục bộ là các bài kiểm thử được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với **Java Virtual Machine (JVM)**. Bạn sử dụng kiểm thử đơn vị cục bộ để kiểm tra các phần của ứng dụng không cần truy cập vào **Android framework** hoặc thiết bị/emulator Android, chẳng hạn như logic nội bộ.

Bạn cũng có thể sử dụng kiểm thử đơn vị cục bộ để kiểm tra các phần của ứng dụng mà bạn có thể tạo ra các đối tượng giả ("mock" hoặc "stub") để bắt chước hành vi của các thành phần tương đương trong framework.

Kiểm thử đơn vị được viết bằng **JUnit**, một khung kiểm thử đơn vị phổ biến dành cho Java.

Những gì bạn cần biết trước

Bạn cần phải:

- Tạo một dự án Android Studio.
- Xây dựng và chạy ứng dụng của bạn trong Android Studio, trên cả trình giả lập và thiết bị thực.
- Điều hướng trong **Project > Android pane** của Android Studio.

- **Tìm các thành phần chính của một dự án Android Studio, bao gồm:**

- **AndroidManifest.xml**,
 - Các tài nguyên (resources),
 - Các tệp Java,
 - Các tệp Gradle.
-

Bạn sẽ học được gì?

- Cách tổ chức và chạy các bài kiểm thử trong Android Studio.
 - Hiểu kiểm thử đơn vị là gì.
 - Viết các bài kiểm thử đơn vị cho mã của bạn.
-

Bạn sẽ làm gì?

- Chạy các bài kiểm thử ban đầu trong ứng dụng **SimpleCalc**.
 - Thêm các bài kiểm thử mới vào ứng dụng **SimpleCalc**.
 - Chạy các bài kiểm thử đơn vị để xem kết quả.
-

Tổng quan về ứng dụng

Bài thực hành này sử dụng ứng dụng **SimpleCalc** từ bài codelab thực hành trước (**Android fundamentals 3.1: The debugger**). Bạn có thể chỉnh sửa ứng dụng này tại chỗ hoặc tạo một bản sao thư mục dự án trước khi tiếp tục.

Nhiệm vụ 1: Khám phá và chạy CalculatorTest

Bạn sẽ viết và chạy các bài kiểm thử (bao gồm cả kiểm thử đơn vị và kiểm thử có công cụ) trong **Android Studio**, cùng với mã của ứng dụng.

Mỗi dự án Android mới đều bao gồm các lớp kiểm thử mẫu cơ bản mà bạn có thể mở rộng hoặc thay thế để sử dụng cho mục đích riêng của mình.

Trong nhiệm vụ này, bạn sẽ quay lại ứng dụng **SimpleCalc**, ứng dụng này đã bao gồm một lớp kiểm thử đơn vị cơ bản.

1.1 Khám phá bộ mã nguồn (source sets) và CalculatorTest

Source sets là các tập hợp mã trong dự án của bạn dùng cho các mục tiêu build khác nhau hoặc các "phiên bản" khác nhau của ứng dụng. Khi Android Studio tạo dự án của bạn, nó sẽ tạo ra ba bộ mã nguồn:

- **Bộ mã nguồn chính (main source set)**: Chứa mã và tài nguyên chính của ứng dụng.
- **Bộ mã nguồn (test)**: Chứa các bài kiểm thử đơn vị cục bộ của ứng dụng. Bộ mã nguồn này hiển thị (**test**) sau tên gói.
- **Bộ mã nguồn (androidTest)**: Chứa các bài kiểm thử có công cụ (instrumented tests) dành cho Android. Bộ mã nguồn này hiển thị (**androidTest**) sau tên gói.

Trong nhiệm vụ này, bạn sẽ khám phá cách các bộ mã nguồn được hiển thị trong Android Studio, kiểm tra cấu hình Gradle dành cho việc kiểm thử, và chạy các bài kiểm thử đơn vị cho ứng dụng SimpleCalc.

Lưu ý: Bộ mã nguồn (**androidTest**) đã được loại bỏ khỏi ví dụ này để đơn giản hóa. Nó sẽ được giải thích chi tiết hơn trong một bài học khác.

Các bước thực hiện

1. Mở dự án **SimpleCalc** trong Android Studio nếu bạn chưa làm điều này.
2. Mở bảng điều hướng **Project > Android**, sau đó mở rộng các thư mục **app** và **java**.
 - Thư mục **java** trong chế độ xem Android hiển thị tất cả các bộ mã nguồn trong ứng dụng theo tên gói.
 - Trong trường hợp này (như minh họa bên dưới), mã của ứng dụng nằm trong bộ mã nguồn **com.android.example.SimpleCalc**.
 - Mã kiểm thử nằm trong bộ mã nguồn có chữ **test** xuất hiện trong ngoặc đơn sau tên gói: **com.android.example.SimpleCalc (test)**.

2. Mở rộng thư mục **com.android.example.SimpleCalc (test)**

Thư mục này là nơi bạn đặt các bài kiểm thử đơn vị cục bộ của ứng dụng. Android Studio tạo sẵn một lớp kiểm thử mẫu trong thư mục này cho các dự án mới, nhưng với ứng dụng **SimpleCalc**, lớp kiểm thử được gọi là **CalculatorTest**.

1. Mở tệp CalculatorTest

Hãy kiểm tra mã nguồn và lưu ý những điểm sau:

- Các thư viện được nhập:**
Chỉ có các thư viện từ các gói **org.junit**, **org.hamcrest**, và **android.test** được nhập. Không có sự phụ thuộc nào vào các lớp của framework Android.
- Chú thích @RunWith(Unit4.class):**
Chú thích này chỉ định **runner** sẽ được sử dụng để chạy các bài kiểm thử trong lớp này. **Test runner** là một thư viện hoặc bộ công cụ cho phép thực hiện kiểm thử và in kết quả ra log. Với các bài kiểm thử cần thiết lập hoặc hàn tầng phức tạp hơn (như Espresso), bạn sẽ sử dụng các test runner khác. Trong ví dụ này, chúng ta sử dụng **JUnit4 test runner** cơ bản.
- Chú thích @SmallTest:**
Chú thích này cho biết tất cả các bài kiểm thử trong lớp là kiểm thử đơn vị, không có phụ thuộc nào và chạy trong thời gian rất ngắn (tính bằng mili giây). Các chú thích **@SmallTest**, **@MediumTest**, và **@LargeTest** là các quy ước giúp dễ dàng nhóm các bài kiểm thử thành các bộ với chức năng tương tự nhau.
- Phương thức setUp():**
Phương thức này được dùng để thiết lập môi trường trước khi kiểm thử và có chú thích **@Before**. Trong trường hợp này, phương thức **setUp()** tạo một instance mới của lớp **Calculator** và gán nó vào biến thành viên **mCalculator**.
- Phương thức addTwoNumbers():**
Đây là một bài kiểm thử thực tế và được chú thích bằng **@Test**. Chỉ những phương thức trong lớp kiểm thử có chú thích **@Test** mới được trình kiểm thử (test runner) xem là bài kiểm thử. Theo quy ước, tên của phương thức kiểm thử không bao gồm từ "test."
- Dòng lệnh đầu tiên của addTwoNumbers():**
Dòng lệnh này gọi phương thức **add()** từ lớp **Calculator**. Bạn chỉ có thể kiểm thử các phương thức có phạm vi truy cập là **public** hoặc **package-protected**. Trong trường hợp này, lớp **Calculator** là lớp **public** với các phương thức **public**, nên không có vấn đề gì.

- **Dòng lệnh thứ hai:**

Đây là một biểu thức khẳng định (assertion) cho bài kiểm thử. **Assertion** là các biểu thức phải đánh giá và trả về **true** để bài kiểm thử được thông qua. Trong trường hợp này, **assertion** kiểm tra xem kết quả nhận được từ phương thức **add()** ($1 + 1$) có khớp với số được chỉ định là 2 hay không. Bạn sẽ tìm hiểu thêm về cách tạo các **assertion** sau trong bài thực hành này.

1.2 Chạy kiểm thử trong Android Studio

Trong phần này, bạn sẽ chạy các bài kiểm thử đơn vị trong thư mục kiểm thử và xem kết quả cho cả kiểm thử thành công và thất bại.

Các bước thực hiện:

1. Trong ngăn Project > **Android**, nhấp chuột phải (hoặc Control-click) vào **CalculatorTest** và chọn **Run 'CalculatorTest'**.
 - o Dự án sẽ được xây dựng (nếu cần thiết), và ngăn **CalculatorTest** sẽ xuất hiện ở dưới cùng của màn hình.
 - o Ở đầu ngăn này, danh sách thả xuống cho các cấu hình thực thi khả dụng cũng thay đổi thành **CalculatorTest**.
2. Tất cả các bài kiểm thử trong lớp **CalculatorTest** sẽ được chạy.
 - o Nếu các bài kiểm thử thành công, thanh tiến trình ở đầu khung nhìn sẽ chuyển sang màu **xanh lá cây**.
 - o Một thông báo trạng thái ở cuối màn hình sẽ báo "**Tests Passed**".
3. Mở **CalculatorTest** nếu nó chưa được mở, và thay đổi biểu thức khẳng định trong phương thức **addTwoNumbers()** thành:

java

CopyEdit

```
assertThat(resultAdd, is(equalTo(3d)));
```

2. Chạy lại bài kiểm thử với cấu hình CalculatorTest

1. Trong danh sách thả xuống cấu hình chạy (run configurations) ở phía trên màn hình, chọn **CalculatorTest** (nếu nó chưa được chọn) và nhấn **Run**.
 - o Bài kiểm thử sẽ chạy lại như trước, nhưng lần này biểu thức khẳng định thất bại (**3 không bằng 1 + 1**).

- Thanh tiến trình trong khung chạy kiểm thử chuyển sang màu **đỏ**, và nhật ký kiểm thử chỉ ra vị trí bài kiểm thử (biểu thức khăng định) thất bại cũng như lý do tại sao.
2. Thay đổi biểu thức khăng định trong phương thức **addTwoNumbers()** quay lại bài kiểm thử đúng, và chạy lại kiểm thử để đảm bảo tất cả bài kiểm thử đều thành công.
 3. Trong danh sách thả xuống cấu hình chạy, chọn **app** để chạy ứng dụng bình thường.
-

Nhiệm vụ 2: Thêm các kiểm thử đơn vị khác vào CalculatorTest

Kiểm thử đơn vị là kiểm tra một phần nhỏ trong mã nguồn của bạn (chẳng hạn một phương thức hoặc một lớp), và tách phần đó khỏi phần còn lại của ứng dụng. Điều này giúp đảm bảo rằng phần mã nhỏ đó hoạt động đúng như mong đợi.

Thông thường, một kiểm thử đơn vị gọi một phương thức với nhiều đầu vào khác nhau, và kiểm tra rằng phương thức hoạt động như mong đợi và trả về kết quả đúng.

Trong nhiệm vụ này, bạn sẽ viết thêm các kiểm thử đơn vị cho các phương thức tiện ích của lớp **Calculator** trong ứng dụng **SimpleCalc**, và chạy các bài kiểm thử để đảm bảo rằng chúng tạo ra đầu ra như bạn mong đợi.

2.1 Thêm các bài kiểm thử khác cho phương thức add()

Mặc dù không thể kiểm thử mọi giá trị đầu vào mà phương thức **add()** có thể nhận, nhưng việc kiểm tra các trường hợp đầu vào đặc biệt là rất quan trọng. Ví dụ:

- Đầu vào có chứa **toán hạng âm**.
- Đầu vào có chứa **số thực (floating-point numbers)**.
- **Số rất lớn (exceptionally large numbers)**
- **Các loại toán hạng khác nhau (float và double)**
- **Toán hạng bằng 0 (zero)**

- **Toán hạng là vô cực (infinity)**

Trong nhiệm vụ này, chúng ta sẽ thêm các bài kiểm thử đơn vị (unit tests) mới cho phương thức add() để kiểm tra các loại đầu vào khác nhau.

1. **Thêm phương thức mới vào CalculatorTest có tên là addTwoNumbersNegative().**

Sử dụng khung mã như sau:

java

CopyEdit

@Test

```
public void addTwoNumbersNegative() {  
}
```

Phương thức kiểm thử này có cấu trúc tương tự như addTwoNumbers():

- Là phương thức **public**, không có tham số, và trả về kiểu **void**.
- Được chú thích bằng **@Test**, cho biết đây là một bài kiểm thử đơn lẻ.

2. **Tại sao không thêm nhiều lệnh kiểm tra (assertions) vào addTwoNumbers()?**

- Gom nhiều lệnh kiểm tra vào một phương thức có thể làm cho bài kiểm thử khó gỡ lỗi nếu chỉ một lệnh kiểm tra thất bại.
- Ngoài ra, việc này cũng làm mờ các bài kiểm thử thành công khác.
- Quy tắc chung cho các bài kiểm thử đơn vị là **tạo một phương thức kiểm thử cho từng lệnh kiểm tra riêng lẻ**.

3. **Chạy tất cả các bài kiểm thử trong CalculatorTest như trước.**

- Trong cửa sổ kiểm thử, cả addTwoNumbers và addTwoNumbersNegative đều được liệt kê dưới dạng các bài kiểm thử khả dụng (và đang thành công) trong bảng điều khiển bên trái.

- Bài kiểm thử addTwoNumbersNegative vẫn thành công ngay cả khi không chứa bất kỳ đoạn mã nào—một bài kiểm thử không làm gì vẫn được xem là thành công.
-

4. Thêm một dòng mã vào addTwoNumbersNegative() để gọi phương thức add() trong lớp Calculator với một toán hạng âm.

java

CopyEdit

```
double resultAdd = mCalculator.add(1d, 2d);
```

Ký hiệu d sau mỗi toán hạng cho biết đây là các số kiểu double.

Do phương thức add() được định nghĩa với các tham số kiểu double, các kiểu dữ liệu khác như float hoặc int cũng có thể hoạt động. Việc chỉ rõ kiểu giúp bạn kiểm tra riêng biệt các kiểu khác nếu cần thiết.

4. Thêm một lệnh kiểm tra (assertion) với assertThat().

java

CopyEdit

```
assertThat(resultAdd, is(equalTo(1d)));
```

- Phương thức assertThat() là một lệnh kiểm tra trong JUnit4, khẳng định biểu thức trong đối số đầu tiên bằng với biểu thức trong đối số thứ hai.
 - Các phiên bản cũ của JUnit sử dụng các phương thức kiểm tra cụ thể hơn như assertEquals(), assertNull(), hoặc assertTrue(), nhưng assertThat() linh hoạt hơn, dễ đọc và dễ gỡ lỗi hơn.
-

assertThat() sử dụng matchers.

- Matchers là các phương thức liên kết được gọi trong đối số thứ hai của lệnh kiểm tra này, chẳng hạn như is(equalTo()).
- Framework Hamcrest định nghĩa các matchers sẵn có để xây dựng các lệnh kiểm tra. (Tên "Hamcrest" là một phép đảo chữ từ "matchers.")
- Hamcrest cung cấp nhiều matchers cơ bản cho hầu hết các lệnh kiểm tra. Bạn cũng có thể tự định nghĩa matchers của riêng mình cho các lệnh kiểm tra phức tạp hơn.

Ví dụ:

Trong trường hợp này, lệnh kiểm tra xác nhận rằng kết quả của phép cộng add() (-1 + 2) bằng với 1.

5. Thêm bài kiểm tra đơn vị mới vào CalculatorTest cho số dấu phẩy động:

java

CopyEdit

@Test

```
public void addTwoNumbersFloats() {  
    double resultAdd = mCalculator.add(1.111f, 1.111d);  
    assertThat(resultAdd, is(equalTo(2.222d)));  
}
```

- Đây là một bài kiểm tra rất giống với phương thức kiểm tra trước, nhưng một đối số của phương thức add() được chỉ rõ là kiểu float thay vì double.
- Phương thức add() được định nghĩa với các tham số kiểu double, vì vậy bạn có thể gọi nó với kiểu float. Khi đó, số kiểu float sẽ được chuyển đổi thành kiểu double.

6. Nhấn vào Run để chạy lại tất cả các bài kiểm tra.

Lần này bài kiểm tra thất bại và thanh tiến trình chuyển sang màu đỏ. Đây là phần quan trọng trong thông báo lỗi:

makefile

CopyEdit

java.lang.AssertionError:

Expected: is <2.222>

but: was <2.2219999418258665>

Tính toán với các số dấu phẩy động không chính xác, và việc chuyển đổi đã gây ra hiệu ứng phụ về độ chính xác.

Lệnh kiểm tra trong bài kiểm tra này về mặt kỹ thuật là sai: giá trị mong đợi không bằng giá trị thực tế.

Câu hỏi đặt ra là:

Khi gặp vấn đề về độ chính xác do việc chuyển đổi tham số kiểu float, đó có phải là vấn đề của mã nguồn hay bài kiểm tra?

- Trong trường hợp này, cả hai đối số đầu vào của phương thức add() từ ứng dụng SimpleCalc đều luôn thuộc kiểu double, do đó đây là một bài kiểm tra tùy ý và không thực tế.
 - Tuy nhiên, nếu ứng dụng của bạn được viết sao cho đầu vào của phương thức add() có thể là kiểu double hoặc float, và bạn chỉ quan tâm đến một mức độ chính xác nhất định, thì bạn cần thêm một biên độ "gần đúng" để bài kiểm tra thành công.
-

7. Thay đổi phương thức assertThat() để sử dụng matcher closeTo():

java

CopyEdit

```
assertThat(resultAdd, is(closeTo(2.222, 0.01)));
```

Hướng dẫn:

- Bạn cần chọn matcher thích hợp. Nhập vào closeTo hai lần (cho đến khi toàn bộ biểu thức được gạch chân) và nhấn **Alt+Enter** (**Option+Return** trên Mac).
 - Chọn **isCloseTo.closeTo (org.hamcrest.number)**.
-

8. Nhấn vào Run để chạy lại tất cả các bài kiểm tra.

Lần này bài kiểm tra đã thành công.

Với matcher closeTo(), thay vì kiểm tra sự bằng nhau tuyệt đối, bạn có thể kiểm tra sự bằng nhau trong một phạm vi delta nhất định.

- Trong trường hợp này, phương thức matcher closeTo() nhận hai tham số: giá trị mong đợi và giá trị delta.
- Trong ví dụ trên, delta là phạm vi gần đúng với độ chính xác hai chữ số thập phân.

2.2 Thêm các bài kiểm tra đơn vị cho các phương thức tính toán khác

Sử dụng những gì bạn đã học trong nhiệm vụ trước để hoàn thiện các bài kiểm tra đơn vị cho lớp Calculator.

1. Thêm một bài kiểm tra đơn vị có tên subTwoNumbers() để kiểm tra phương thức sub().
2. Thêm một bài kiểm tra đơn vị có tên subWorksWithNegativeResults() để kiểm tra phương thức sub() khi phép tính cho ra kết quả âm.
3. Thêm một bài kiểm tra đơn vị có tên mulTwoNumbers() để kiểm tra phương thức mul().
4. Thêm một bài kiểm tra đơn vị có tên mulTwoNumbersZero() để kiểm tra phương thức mul() khi ít nhất một tham số là số 0.
5. Thêm một bài kiểm tra đơn vị có tên divTwoNumbers() để kiểm tra phương thức div() với hai tham số khác 0.
6. Thêm một bài kiểm tra đơn vị có tên divTwoNumbersZero() để kiểm tra phương thức div() với một số chia kiểu double và số chia là 0.

Tất cả các bài kiểm tra trên đều phải thành công, ngoại trừ divTwoNumbersZero(), bài này sẽ gây ra ngoại lệ tham số không hợp lệ khi chia cho 0.

- Nếu bạn chạy ứng dụng, nhập 0 làm Số hạng 2 và nhấn Div để chia, kết quả sẽ là lỗi.

Giải pháp mã cho nhiệm vụ 2:

Dự án Android Studio: SimpleCalcTest

Đoạn mã sau đây minh họa các bài kiểm tra cho nhiệm vụ này:

```

public void addTwoNumbers() {
    double resultAdd = mCalculator.add(1d, 1d);
    assertThat(resultAdd, is(equalTo(2d)));
}

@Test
public void addTwoNumbersNegative() {
    double resultAdd = mCalculator.add(-1d, 2d);
    assertThat(resultAdd, is(equalTo(1d)));
}
@Test
public void addTwoNumbersFloats() {
    double resultAdd = mCalculator.add(1.111f, 1.111d);
    assertThat(resultAdd, is(closeTo(2.222, 0.01)));
}
@Test
public void subTwoNumbers() {
    double resultSub = mCalculator.sub(1d, 1d);
    assertThat(resultSub, is(equalTo(0d)));
}
@Test
public void subWorksWithNegativeResult() {
    double resultSub = mCalculator.sub(1d, 17d);
    assertThat(resultSub, is(equalTo(-16d)));
}
@Test
public void multTwoNumbers() {
    double resultMul = mCalculator.mul(32d, 2d);
    assertThat(resultMul, is(equalTo(64d)));
}
@Test
public void divTwoNumbers() {
    double resultDiv = mCalculator.div(32d, 2d);
    assertThat(resultDiv, is(equalTo(16d)));
}
@Test
public void divTwoNumbersZero() {
    double resultDiv = mCalculator.div(32d, 0);
    assertThat(resultDiv, is(equalTo(Double.POSITIVE_INFINITY)));
}

```

```
@Test
public void subWorksWithNegativeResult() {
    double resultSub = mCalculator.sub(1d, 17d);
    assertThat(resultSub, is(equalTo(-16d)));
}

@Test
public void multTwoNumbers() {
    double resultMul = mCalculator.mul(32d, 2d);
    assertThat(resultMul, is(equalTo(64d)));
}

@Test
public void divTwoNumbers() {
    double resultDiv = mCalculator.div(32d, 2d);
    assertThat(resultDiv, is(equalTo(16d)));
}

@Test
public void divTwoNumbersZero() {
    double resultDiv = mCalculator.div(32d, 0);
    assertThat(resultDiv, is(equalTo(Double.POSITIVE_INFINITY)));
}
```

Thách thức lập trình

Lưu ý: Tất cả các thách thức lập trình đều tùy chọn và không phải là yêu cầu bắt buộc cho các bài học sau.

Thách thức 1:

Chia cho 0 luôn là một trường hợp đặc biệt trong toán học và đáng để kiểm tra. Làm thế nào bạn có thể thay đổi ứng dụng để xử lý chia cho 0 một cách dễ chịu hơn?

Để thực hiện thách thức này, hãy bắt đầu bằng một bài kiểm tra cho thấy hành vi đúng là gì.

- Xóa phương thức divTwoNumbersZero() khỏi lớp CalculatorTest và thêm một bài kiểm tra đơn vị mới có tên là divByZeroThrows() để kiểm tra

phương thức div() với đối số thứ hai bằng 0, với kết quả mong đợi là IllegalArgumentException.class.

- Bài kiểm tra này sẽ thành công và chứng minh rằng bất kỳ phép chia nào với số 0 sẽ dẫn đến ngoại lệ này.

Sau khi bạn học cách viết mã cho một trình xử lý Exception, ứng dụng của bạn có thể xử lý ngoại lệ này một cách dễ chịu, ví dụ: hiển thị một thông báo Toast yêu cầu người dùng thay đổi Operand 2 từ 0 thành một số khác.

Thách thức 2:

Đôi khi rất khó để cô lập một đơn vị mã khỏi tất cả các phụ thuộc bên ngoài của nó. Thay vì tổ chức mã của bạn theo những cách phức tạp chỉ để dễ dàng kiểm tra hơn, bạn có thể sử dụng một khung giả lập để tạo ra các đối tượng giả ("mock") giả vờ là các phụ thuộc.

- Nghiên cứu khung Mockito và tìm hiểu cách thiết lập nó trong Android Studio.
- Viết một lớp kiểm tra cho phương thức calcButton() trong ứng dụng SimpleCalc và sử dụng Mockito để mô phỏng ngữ cảnh Android mà các bài kiểm tra của bạn sẽ chạy.

Tóm tắt

Android Studio có các tính năng tích hợp để chạy các bài kiểm tra đơn vị cục bộ:

- **Bài kiểm tra đơn vị cục bộ** sử dụng JVM trên máy cục bộ của bạn. Chúng không sử dụng khung Android.
- Các bài kiểm tra đơn vị được viết bằng **JUnit**, một khung kiểm tra đơn vị phổ biến cho Java.
- Các bài kiểm tra JUnit được đặt trong thư mục **(test)** trong mục **Project > Android** của Android Studio.
- Các bài kiểm tra đơn vị cục bộ chỉ cần các gói sau: **org.junit**, **org.hamcrest**, và **android.test**.
- Chú thích **@RunWith(JUnit4.class)** báo cho trình chạy kiểm tra thực thi các bài kiểm tra trong lớp này.
- Các chú thích **@SmallTest**, **@MediumTest**, và **@LargeTest** là các quy ước giúp dễ dàng nhóm các bài kiểm tra tương tự lại với nhau.
- Chú thích **@SmallTest** cho biết tất cả các bài kiểm tra trong một lớp là các bài kiểm tra đơn vị không có phụ thuộc và chạy trong vài mili-giây.

- **Bài kiểm tra có công cụ hỗ trợ (Instrumented tests)** là các bài kiểm tra chạy trên một thiết bị hoặc trình giả lập chạy Android. Các bài kiểm tra này có quyền truy cập vào khung Android.
- **Trình chạy kiểm tra (Test runner)** là một thư viện hoặc tập hợp công cụ cho phép thực hiện các bài kiểm tra và in kết quả vào nhật ký (log).

Khái niệm liên quan

Tài liệu khái niệm liên quan nằm trong **3.2: App testing**.

Tìm hiểu thêm

Tài liệu Android Studio:

- [Hướng dẫn sử dụng Android Studio \(Android Studio User Guide\)](#)
- [Ghi và xem nhật ký \(Write and View Logs\)](#)

Tài liệu nhà phát triển Android:

- [Thực hành tốt nhất cho kiểm tra \(Best Practices for Testing\)](#)
- [Bắt đầu với kiểm tra \(Getting Started with Testing\)](#)
- [Xây dựng kiểm tra đơn vị cục bộ \(Building Local Unit Tests\)](#)

Khác:

- [Trang chủ JUnit 4 \(JUnit 4 Home Page\)](#)
- [Tài liệu API JUnit 4 \(JUnit 4 API Reference\)](#)
- [java.lang.Math](#)
- [Java Hamcrest](#)
- [Trang chủ Mockito \(Mockito Home Page\)](#)
- [Video: Android Testing Support - Testing Patterns](#)
- [Hướng dẫn lập trình kiểm tra Android \(Android Testing Codelab\)](#)
- [Protip về chú thích kích thước kiểm tra Android \(Android Tools Protip: Test Size Annotations\)](#)
- [Lợi ích của việc sử dụng assertThat thay vì các phương thức Assert khác trong kiểm tra đơn vị \(The Benefits of Using assertThat over other Assert Methods in Unit Tests\)](#)

Bài tập về nhà

Xây dựng và chạy ứng dụng

Mở ứng dụng **SimpleCalc** từ bài thực hành về sử dụng trình gõ lỗi. Bạn sẽ thêm một nút **POW** vào giao diện. Nút này thực hiện phép tính số mũ: cơ số (operand đầu tiên) được nâng lên lũy thừa bởi số mũ (operand thứ hai). Ví dụ, với các toán hạng là 5 và 4, ứng dụng sẽ tính $5^4 = 625$.

Trước khi thực hiện:

Hãy xem xét các trường hợp kiểm tra bạn muốn thực hiện cho phép tính này. Những giá trị bất thường nào có thể xảy ra?

Các bước thực hiện

1. Cập nhật lớp **Calculator**:

- Thêm phương thức **pow()** vào lớp **Calculator** trong ứng dụng.
- Gợi ý: Tham khảo tài liệu về lớp [java.lang.Math](#).

2. Cập nhật lớp **MainActivity**:

- Kết nối nút **POW** trong giao diện với phép tính trong lớp **Calculator**.
-

Viết và chạy các bài kiểm tra

- Kiểm tra với các toán hạng là số nguyên dương.
- Kiểm tra với toán hạng đầu tiên là số nguyên âm.
- Kiểm tra với toán hạng thứ hai là số nguyên âm.
- Kiểm tra với toán hạng đầu tiên là 0 và toán hạng thứ hai là số nguyên dương.

Chạy toàn bộ bài kiểm tra của bạn mỗi khi viết xong một bài kiểm tra, và sửa phép tính trong ứng dụng nếu cần.

- Một bài kiểm tra với 0 làm toán hạng thứ hai.
- Một bài kiểm tra với 0 làm toán hạng thứ nhất và -1 làm toán hạng thứ hai.

(Gợi ý: tham khảo tài liệu về **Double.POSITIVE_INFINITY**.)

- Một bài kiểm tra với -0 làm toán hạng thứ nhất và bất kỳ số âm nào làm toán hạng thứ hai.

Trả lời các câu hỏi

Câu hỏi 1

Phát biểu nào mô tả đúng nhất về kiểm tra đơn vị cục bộ? Chọn một:

- Các bài kiểm tra chạy trên một thiết bị hoặc trình giả lập Android và có quyền truy cập vào khung làm việc Android.
- Các bài kiểm tra cho phép bạn viết các phương thức kiểm tra giao diện người dùng tự động.
- Các bài kiểm tra được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với Máy ảo Java (JVM).

Trả lời:

- *Các bài kiểm tra được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với Máy ảo Java (JVM).*

Câu hỏi 2

Tập hợp nguồn (source set) là các tập hợp mã liên quan. Trong tập hợp nguồn nào bạn có thể tìm thấy các bài kiểm tra đơn vị? Chọn một:

- app/res
- com.example.android.SimpleCalcTest
- com.example.android.SimpleCalcTest (test)
- com.example.android.SimpleCalcTest (androidTest)

Trả lời:

- *com.example.android.SimpleCalcTest (test)*

Câu hỏi 3

Chú thích nào được sử dụng để đánh dấu một phương thức là một bài kiểm tra thực tế? Chọn một:

- *@RunWith(JUnit4.class)*

- • `@SmallTest`
- • `@Before`
- • `@Test`

Trả lời:

- • `@Test`

Nộp ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kiểm tra rằng ứng dụng có các tính năng sau:

- • Ứng dụng hiển thị nút **POW** để thực hiện phép tính mũ ("power of").
- • Việc triển khai trong **MainActivity** bao gồm trình xử lý sự kiện nhấn nút cho nút **POW**.
- • Việc triển khai trong **Calculator** bao gồm phương thức **pow()** thực hiện phép tính lũy thừa.
- • Phương thức **CalculatorTest()** có các phương thức kiểm tra riêng biệt cho phương thức **pow()** trong lớp **Calculator** để kiểm tra các trường hợp toán hạng âm, toán hạng bằng 0, và trường hợp toán hạng là 0 và -1.

3.3) Thư viện hỗ trợ

Giới thiệu

Android SDK bao gồm Android Support Library, là một tập hợp nhiều thư viện. Các thư viện này cung cấp các tính năng không được tích hợp sẵn trong Android framework, bao gồm:

- Các phiên bản tương thích ngược của các thành phần framework, cho phép các ứng dụng chạy trên các phiên bản Android cũ hơn hỗ trợ các tính năng có trong các phiên bản mới hơn của nền tảng.
- Các thành phần bổ sung cho bộ cục và giao diện người dùng.
- Hỗ trợ cho các thiết bị có hình thức khác nhau, chẳng hạn như thiết bị TV hoặc thiết bị đeo được.
- Các thành phần hỗ trợ các yếu tố của Material Design.
- Các tính năng khác, bao gồm hỗ trợ bảng màu (palette support), chú thích (annotations), kích thước bộ cục dựa trên phần trăm, và tùy chọn (preferences).

Những điều bạn nên biết trước

Bạn cần có khả năng:

- Tạo một dự án trong Android Studio.
- Sử dụng trình chỉnh sửa bộ cục để làm việc với các thành phần **EditText** và **Button**.
- Xây dựng và chạy ứng dụng của bạn trong Android Studio, cả trên trình giả lập và trên thiết bị thực.
- Điều hướng trong **Project > Android** pane trong Android Studio.
- Xác định các thành phần chính của một dự án Android Studio, bao gồm **AndroidManifest.xml**, tài nguyên (resources), tệp Java, và tệp Gradle.

Bạn sẽ học được gì

- Cách kiểm tra xem **Android Support Library** có sẵn trong cài đặt Android Studio của bạn hay không.

- Cách sử dụng các lớp của thư viện hỗ trợ trong ứng dụng của bạn.
- Cách phân biệt các giá trị của **compileSdkVersion**, **targetSdkVersion**, và **minSdkVersion**.
- Cách nhận biết các API đã lỗi thời hoặc không khả dụng trong mã của bạn.
- Hiểu thêm về các thư viện hỗ trợ Android.

Bạn sẽ làm gì

- Tạo một ứng dụng mới với một **TextView** và một **Button**.
- Kiểm tra xem **Android Support Repository** (chứa Android Support Library) có sẵn trong cài đặt Android Studio của bạn hay không.
- Khám phá các tệp **build.gradle** của dự án ứng dụng của bạn.
- Quản lý các lớp hoặc phương thức không khả dụng cho phiên bản Android mà ứng dụng của bạn hỗ trợ.
- Sử dụng một lớp tương thích từ thư viện hỗ trợ để cung cấp khả năng tương thích ngược cho ứng dụng của bạn.

Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ tạo một ứng dụng có tên **HelloCompat** với một **TextView** hiển thị dòng chữ "Hello World" trên màn hình và một **Button** thay đổi màu sắc của văn bản. Có 20 màu khác nhau được định nghĩa dưới dạng tài nguyên trong tệp **color.xml**, và mỗi lần nhấn nút sẽ ngẫu nhiên chọn một trong các màu đó.

Các phương thức để lấy giá trị màu từ tài nguyên của ứng dụng đã thay đổi theo các phiên bản khác nhau của Android framework. Ví dụ này sử dụng lớp **ContextCompat** trong Android Support Library, cho phép bạn sử dụng một phương thức hoạt động trên tất cả các phiên bản.

Nhiệm vụ 1: Thiết lập dự án của bạn để sử dụng các thư viện hỗ trợ

Trong nhiệm vụ này, bạn sẽ thiết lập một dự án mới cho ứng dụng **HelloCompat** và triển khai bộ cục cũng như hành vi cơ bản.

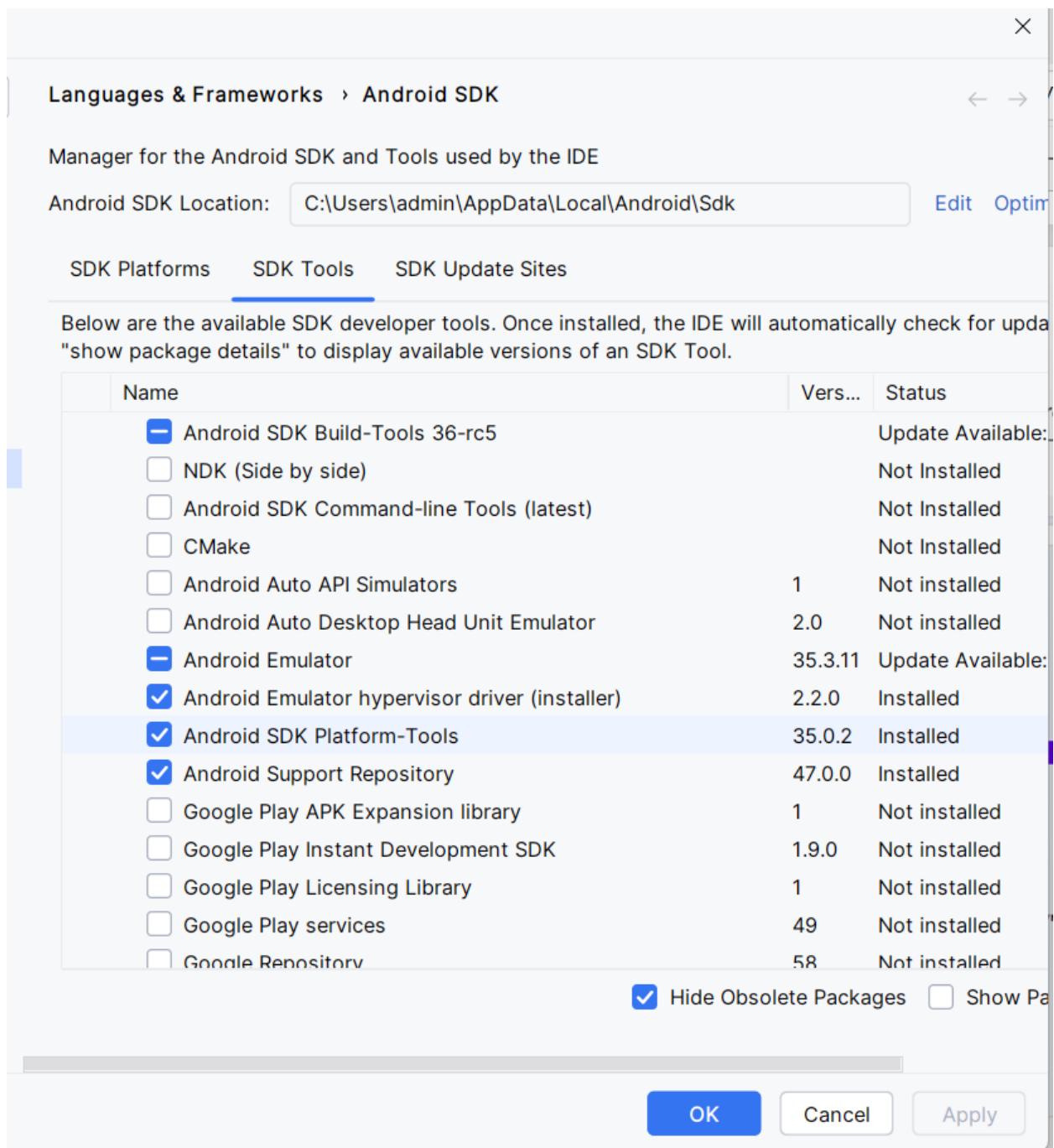
1.1 Xác minh rằng Android Support Repository có sẵn

Android Support Library được tải xuống như một phần của Android SDK và có sẵn trong **Android SDK Manager**. Trong Android Studio, bạn sẽ sử dụng **Android Support Repository**—kho lưu trữ cục bộ cho các thư viện hỗ trợ—để truy cập các thư viện từ bên trong các tệp Gradle build của bạn. Trong nhiệm vụ này, bạn sẽ xác minh rằng **Android Support Repository** đã được tải xuống và sẵn sàng cho các dự án của bạn.

15.Trong Android Studio, chọn **Tools > Android > SDK Manager**, hoặc nhấp vào biểu tượng **SDK Manager**.

Bảng **Android SDK Default Preferences** sẽ xuất hiện.

16.Nhấp vào tab **SDK Tools** và mở rộng **Support Repository**, như hình minh họa bên dưới.



17. Tìm **Android Support Repository** trong danh sách.

- Nếu trạng thái **Installed** xuất hiện trong cột Status, bạn đã sẵn sàng. Nhấp vào **Cancel**.
- Nếu trạng thái **Not installed** hoặc **Update Available** xuất hiện, hãy nhấp vào ô kiểm bên cạnh **Android Support Repository**. Một biểu tượng tải xuống sẽ xuất hiện bên cạnh ô kiểm. Nhấp vào **OK**.

18.Nhấp vào **OK** một lần nữa, sau đó nhấp vào **Finish** khi lưu trữ hỗ trợ đã được cài đặt.

1.2 Thiết lập dự án và kiểm tra build.gradle

1. Tạo một dự án mới có tên **HelloCompat**.

Trên trang **Target Android Devices**, chọn **API 15: Android 4.0.3 (IceCreamSandwich)** làm phiên bản SDK tối thiểu. Như bạn đã học trong các bài học trước, đây là phiên bản Android cũ nhất mà ứng dụng của bạn sẽ hỗ trợ.

1. Nhấp vào **Next**, và chọn mẫu **Empty Activity**.
2. Nhấp vào **Next**, và đảm bảo rằng các tùy chọn **Generate Layout file** và **Backwards Compatibility (App Compat)** được chọn. Tùy chọn thứ hai đảm bảo rằng ứng dụng của bạn sẽ tương thích ngược với các phiên bản Android trước đó.
3. Nhấp vào **Finish**.

Khám phá build.gradle (Module: app)

1. Trong Android Studio, đảm bảo rằng bảng **Project > Android** đang mở.
2. Mở rộng **Gradle Scripts** và mở tệp **build.gradle (Module: app)**.

Lưu ý rằng tệp **build.gradle** cho toàn bộ dự án (**build.gradle (Project: HelloCompat)**) là một tệp khác so với **build.gradle** cho mô-đun ứng dụng. Trong tệp **build.gradle (Module: app)**:

3. Tìm dòng **compileSdkVersion** gần đầu tệp. Ví dụ:

compileSdk = 35

compileSdkVersion là phiên bản Android framework mà ứng dụng của bạn được biên dịch trong Android Studio. Đối với các dự án mới, phiên bản compile thường là tập hợp API framework mới nhất mà bạn đã cài đặt. Giá trị này chỉ ảnh hưởng đến chính Android Studio và các cảnh báo hoặc lỗi mà bạn nhận được trong Android Studio nếu bạn sử dụng các API cũ hơn hoặc mới hơn.

4. Tìm dòng **minSdkVersion** trong phần **defaultConfig** cách vài dòng bên dưới.

minSdk = 24

Phiên bản tối thiểu (minimum) là phiên bản API Android cũ nhất mà ứng dụng của bạn có thể chạy. Đây là số bạn đã chọn ở Bước 1 khi tạo dự án. Cửa hàng Google Play sử dụng số này để đảm bảo rằng ứng dụng của bạn có thể chạy trên thiết bị của người dùng. Android Studio cũng sử dụng số này để cảnh báo bạn về việc sử dụng các API đã lỗi thời.

5. Tìm dòng **targetSdkVersion** trong phần **defaultConfig**. Ví dụ:

targetSdkVersion 26

targetSdk = 35

versionCode = 1

Phiên bản mục tiêu (target) chỉ ra phiên bản API mà ứng dụng của bạn được thiết kế và kiểm tra. Nếu API của nền tảng Android cao hơn số này (tức là ứng dụng của bạn chạy trên một thiết bị mới hơn), nền tảng có thể kích hoạt các hành vi tương thích để đảm bảo rằng ứng dụng của bạn tiếp tục hoạt động theo cách nó được thiết kế.

Ví dụ, Android 6.0 (API 23) cung cấp một mô hình cấp quyền khi chạy (runtime permissions model). Nếu ứng dụng của bạn nhắm mục tiêu đến một cấp API thấp hơn, nền tảng sẽ sử dụng lại mô hình cấp quyền khi cài đặt (install-time permissions model) cũ.

Mặc dù **target SDK** có thể giống với **compile SDK**, nhưng thường là một số thấp hơn để chỉ ra phiên bản API mới nhất mà bạn đã kiểm tra ứng dụng của mình.

6. Tìm phần **dependencies** trong tệp **build.gradle**, gần cuối tệp. Ví dụ:

```
dependencies {  
  
    implementation(libs.appcompat)  
    implementation(libs.material)  
    implementation(libs.activity)  
    implementation(libs.constraintlayout)  
    testImplementation(libs.junit)  
    androidTestImplementation(libs.ext.junit)  
    androidTestImplementation(libs.espresso.core)  
}
```

Phần **dependencies** cho một dự án mới bao gồm nhiều thư viện phụ thuộc để hỗ trợ kiểm thử với Espresso và JUnit, cũng như thư viện hỗ trợ **appcompat v7**. Các số phiên bản của các thư viện này trong dự án của bạn có thể khác so với những số được hiển thị ở đây.

Thư viện hỗ trợ **appcompat v7** cung cấp khả năng tương thích ngược cho các phiên bản Android cũ hơn, từ API 9 trở đi. Nó cũng bao gồm cả thư viện hỗ trợ **compat v4**, vì vậy bạn không cần thêm cả hai thư viện này vào phụ thuộc.

7. Cập nhật số phiên bản, nếu cần.

- Nếu số phiên bản hiện tại của một thư viện thấp hơn so với phiên bản thư viện hiện có, Android Studio sẽ làm nổi bật dòng đó và cảnh báo rằng có phiên bản mới hơn ("A newer version of com.android.support:appcompat-v7 is available"). Hãy chỉnh sửa số phiên bản thành phiên bản mới nhất.
- Mẹo: Bạn cũng có thể nhấp bất kỳ đâu trên dòng được đánh dấu và nhấn **Alt+Enter (Option+Return** trên Mac). Chọn **Change to xx.xx.x** từ menu, trong đó **xx.xx.x** là phiên bản mới nhất hiện có.

8. Cập nhật số **compileSdkVersion**, nếu cần.

- Số phiên bản chính của thư viện hỗ trợ (số đầu tiên) phải khớp với **compileSdkVersion** của bạn. Khi bạn cập nhật phiên bản thư viện hỗ trợ, bạn cũng có thể cần cập nhật **compileSdkVersion** để khớp.

9. Nhấp vào **Sync Now** để đồng bộ các tệp Gradle đã cập nhật với dự án, nếu được nhắc.

10.Cài đặt các tệp nền tảng SDK bị thiếu, nếu cần.

Nếu bạn cập nhật **compileSdkVersion**, bạn có thể cần cài đặt các thành phần nền tảng SDK để khớp. Nhập vào **Install missing platform(s) and sync project** để bắt đầu quá trình này.

Nhiệm vụ 2: Triển khai bố cục và MainActivity

Trong nhiệm vụ này, bạn sẽ triển khai bố cục và hành vi cơ bản cho lớp **MainActivity**.

2.1 Thay đổi bố cục và màu sắc

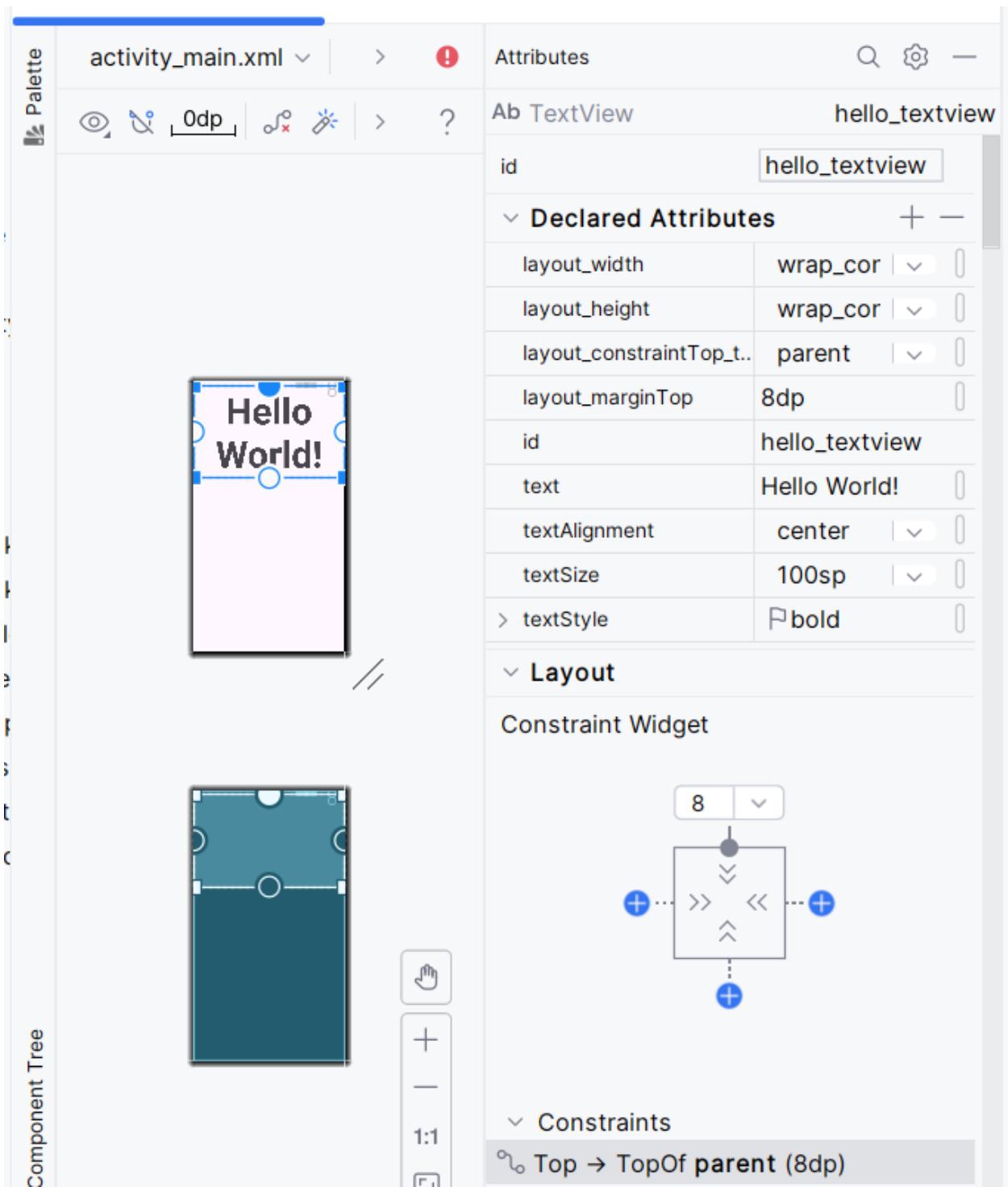
Trong nhiệm vụ này, bạn sẽ chỉnh sửa bố cục **activity_main.xml** cho ứng dụng.

1. Mở tệp **activity_main.xml** trong **Project > Android pane**.
2. Nhập vào tab **Design** (nếu chưa được chọn) để hiển thị trình chỉnh sửa bố cục.
3. Chọn **TextView "Hello World"** trong bố cục và mở bảng **Attributes**.
4. Thay đổi các thuộc tính của **TextView** như sau:

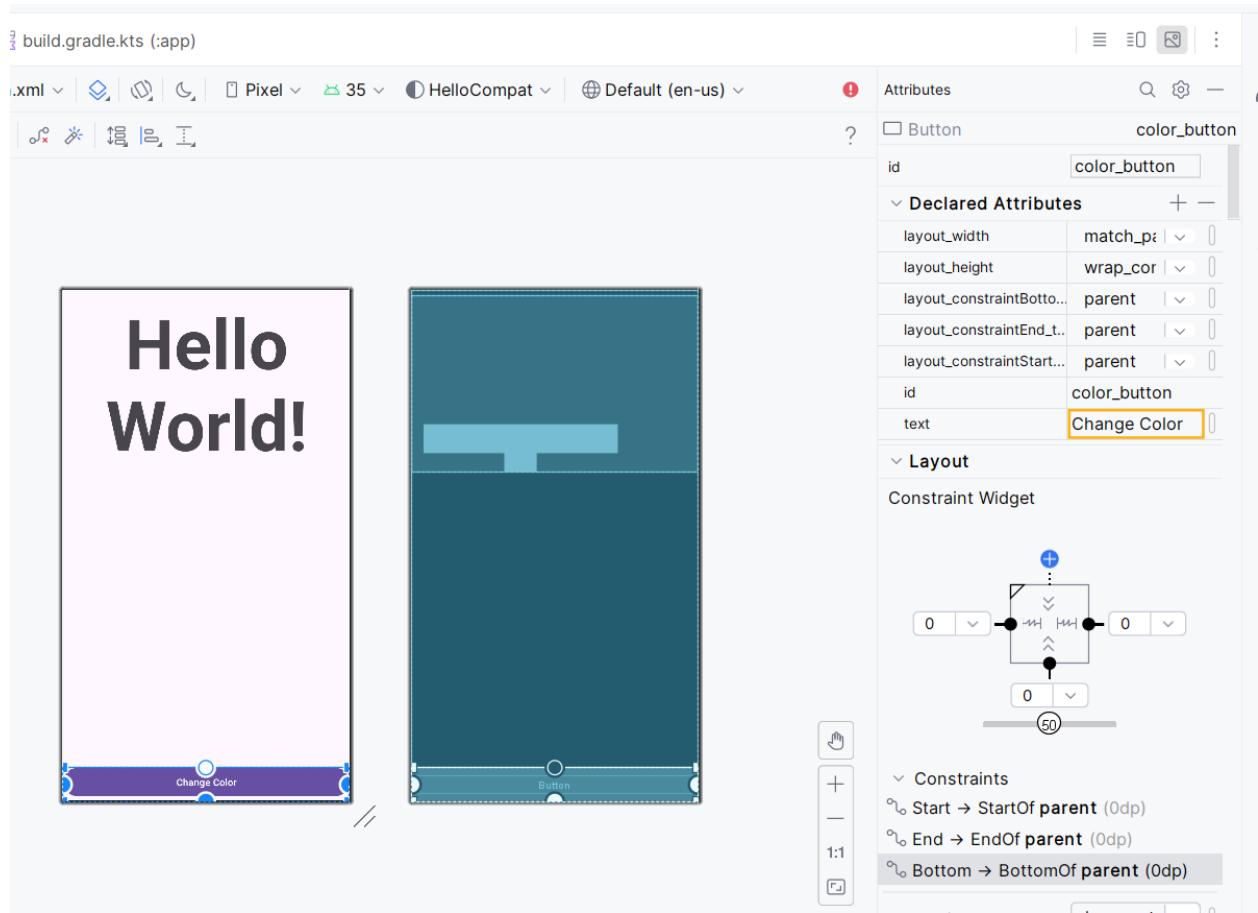
Attribute field	Enter the following:
ID	hello_textview
textStyle	B (bold)
textAlignment	Center the paragraph icon
textSize	100sp

Điều này thêm thuộc tính **android:id** vào **TextView** với **id** được đặt là **hello_textview**, thay đổi căn chỉnh văn bản, làm cho văn bản in đậm, và đặt kích thước văn bản lớn hơn là **100sp**.

5. Xóa ràng buộc (constraint) kéo dài từ phần dưới của **hello_textview** (**TextView**) đến phần dưới của bố cục, để **TextView** gắn lên phần trên của bố cục, và chọn **8** (8dp) cho lề trên (top margin) như hình dưới đây.



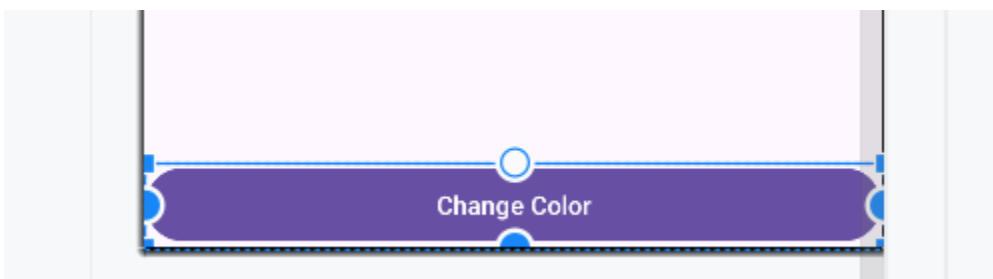
6. Kéo một **Button** vào phía dưới của bố cục, và thêm các ràng buộc (constraints) vào các cạnh trái, phải, và phía dưới của bố cục, như hiển thị trong hình dưới đây.



7. Thay đổi thuộc tính **layout_width** trong bảng **Attributes** cho **Button** thành **match_constraint**.
8. Thay đổi các thuộc tính khác trong bảng **Attributes** cho **Button** như sau:

Attribute field	Enter the following:
ID	color_button
text	"Change Color"

Button bây giờ sẽ hiển thị trong bố cục như minh họa bên dưới.



9. Trong một bài học trước, bạn đã học cách trích xuất một tài nguyên chuỗi từ một chuỗi văn bản cố định. Nhấp vào tab **Text** để chuyển sang mã XML, trích xuất chuỗi "Hello Text!" và "Change Color" trong **TextView** và **Button**, và đặt tên tài nguyên chuỗi (string resource) cho chúng.

10.Thêm thuộc tính sau vào **Button**:

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

11.Để thêm màu sắc, mở rộng **res** và **values** trong **Project > Android** pane, sau đó mở tệp **colors.xml**.

12.Thêm các tài nguyên màu sau vào tệp:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3 ■ <color name="black">#FF000000</color>
4 <color name="white">#FFFFFF</color>
5 ■ <color name="red">#FF0000</color>
6 ■ <color name="blue">#0000FF</color>
7 ■ <color name="green">#008000</color>
8 ■ <color name="yellow">#FFFF00</color>
9 ■ <color name="purple">#800080</color>
0 ■ <color name="pink">#FFC0CB</color>
1 ■ <color name="orange">#FFA500</color>
2 ■ <color name="gray">#808080</color>
3 ■ <color name="cyan">#00FFFF</color>
4 ■ <color name="magenta">#FF00FF</color>
5 ■ <color name="lime">#00FF00</color>
6 ■ <color name="maroon">#800000</color>
7 ■ <color name="navy">#000080</color>
8 ■ <color name="teal">#008080</color>
9 ■ <color name="olive">#808000</color>
0 ■ <color name="brown">#A52A2A</color>
1 ■ <color name="gold">#FFD700</color>
2 ■ <color name="beige">#F5F5DC</color>
3 ■ <span style="color: #C0C0C0;">! <color name="silver">#C0C0C0</color>
4
5
6 </resources>
```

Các giá trị và tên màu này đến từ bảng màu được khuyến nghị cho các ứng dụng Android được định nghĩa tại **Material Design - Style - Color**. Các mã này biểu thị giá trị RGB của màu theo hệ thập lục phân.

2.2 Thêm hành vi vào MainActivity

Trong nhiệm vụ này, bạn sẽ hoàn thành việc thiết lập dự án bằng cách thêm các biến **private** và triển khai các phương thức **onCreate()** và **onSaveInstanceState()**.

1. Mở **MainActivity**.
2. Thêm một biến **private** ở đầu lớp để giữ đối tượng **TextView**:

```
private TextView mHelloTextView;
```

3. Thêm mảng màu sau đây ngay sau biến **private**:

```
private String[] mColorArray = {  
    "red",  
    "pink",  
    "deep_purple",  
    "indigo",  
    "blue",  
    "light_blue",  
    "cyan",  
    "teal",  
    "green",  
    "light_green",  
    "lime",  
    "yellow",  
    "amber",  
    "orange",  
    "deep_orange",  
    "brown",  
    "grey",  
    "blue_grey",  
    "black"  
};|
```

Mỗi tên màu tương ứng với tên của một tài nguyên màu trong **color.xml**.

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

```
mHelloTextView = findViewById(R.id.hello_textview);
```

1. Trong phương thức **onCreate()**, khôi phục trạng thái phiền lưu trữ, nếu có:

```
if (savedInstanceState != null) {  
    mHelloTextView.setTextColor(savedInstanceState.getInt("color"));  
}
```

2. Thêm phương thức **onSaveInstanceState()** vào lớp **MainActivity** để lưu màu của văn bản:

```
@Override  
public void onSaveInstanceState(Bundle outState) {  
    super.onSaveInstanceState(outState);  
    // lưu màu hiện tại của văn bản  
    outState.putInt("color", mHelloTextView.getCurrentTextColor());  
}
```

Mã giải pháp cho Nhiệm vụ 2

Dưới đây là mã giải pháp cho bộ cục XML và một đoạn mã trong lớp **MainActivity** của ứng dụng **HelloCompat** cho đến thời điểm này.

Tệp bộ cục **activity_main.xml** như sau. Trình xử lý **changeColor** cho thuộc tính **android:onClick** của nút **Button** đang được gạch đỏ vì nó chưa được định nghĩa. Bạn sẽ định nghĩa nó trong nhiệm vụ tiếp theo.

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

```
<TextView  
    android:id="@+id/hello_textview"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/hello_world"  
    android:textAlignment="center"  
    android:textSize="100sp"  
    android:textStyle="bold"  
    android:layout_marginTop="8dp"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"/>  
  
<Button  
    android:id="@+id/color_button"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Change Color"  
    android:layout_marginTop="8dp"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    android:onClick="changeColor"/>  
  
</androidx.constraintlayout.widget.ConstraintLayout>
```

Lớp MainActivity

Lớp **MainActivity** bao gồm các biến private sau ở đầu lớp:

```
private TextView mHelloTextView;
no usages
private String[] mColorArray ={
    "red",
    "pink",
    "deep_purple",
    "indigo",
    "blue",
    "light_blue",
    "cyan",
    "teal",
    "green",
    "light_green",
    "lime",
    "yellow",
    "amber",
    "orange",
    "deep_orange",
    "brown",
    "grey",
    "blue_grey",
    "black"
};

};
```

Phương thức onCreate() và onSaveInstanceState() trong MainActivity

```

@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;
    });
    mHelloTextView = findViewById(R.id.hello_textview);
    if (savedInstanceState != null) {
        mHelloTextView.setTextColor(savedInstanceState.getInt( key: "color"));
    }
}

@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    // lưu màu hiện tại của văn bản
    outState.putInt("color", mHelloTextView.getCurrentTextColor());
}

```

Nhiệm vụ 3: Thực hiện hành vi cho Button

Nút **Change Color** trong ứng dụng **HelloCompat** chọn ngẫu nhiên một trong 20 màu từ tệp tài nguyên color.xml và đặt màu của văn bản thành màu đó. Trong nhiệm vụ này, bạn sẽ triển khai hành vi cho trình xử lý sự kiện khi nhấn Button.

2.1 Thêm trình xử lý sự kiện changeButton()

1. Mở tệp activity_main.xml nếu chưa mở. Chuyển sang tab **Text** để hiển thị mã XML.
2. Nhập vào "changeColor" trong thuộc tính android:onClick của phần tử Button.
3. Nhấn **Alt+Enter** (hoặc **Option+Enter** trên Mac) và chọn **Create onClick event handler**.
4. Chọn **MainActivity** và nhập **OK**.

Điều này sẽ tạo một phương thức mẫu cho changeColor() trong MainActivity:

1 usage

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

2.2 Triển khai hành động của Button

1. Chuyển sang tệp MainActivity.
2. Trong phương thức changeColor(), tạo một đối tượng số ngẫu nhiên bằng cách sử dụng lớp Random (một lớp trong Java):

```
Random random = new Random();
```

3. Sử dụng đối tượng random để chọn một màu ngẫu nhiên từ mảng mColorArray:

```
String colorName = mColorArray[random.nextInt( bound: 20)];
```

Phương thức nextInt() với tham số 20 sẽ trả về một số nguyên ngẫu nhiên trong khoảng từ 0 đến 19. Số nguyên này sẽ được dùng làm chỉ số để truy cập vào mảng và lấy tên màu.

4. Lấy mã định danh tài nguyên (một số nguyên) cho tên màu từ tài nguyên:

```
int colorResouceName = getResources().getIdentifier(colorName, "color",  
getApplicationContext().getPackageName());
```

Khi ứng dụng của bạn được biên dịch, hệ thống Android chuyển đổi các định nghĩa trong các tệp XML của bạn thành các tài nguyên với các mã định danh nội bộ dạng số nguyên (ID). Có các ID riêng biệt cho cả tên và giá trị. Dòng mã này ánh xạ các chuỗi màu từ mảng colorName với các ID tên màu tương ứng trong tệp tài nguyên XML. Phương thức getResources() lấy tất cả các tài nguyên của ứng dụng, trong khi phương thức getIdentifier() tìm kiếm tên màu (chuỗi) trong tài nguyên màu ("color") của tên gói hiện tại.

5. **Lấy mã số nguyên cho màu thực tế từ các tài nguyên và gán nó cho biến colorRes, sau đó sử dụng phương thức getTheme() để lấy giao diện chủ đề cho ngữ cảnh ứng dụng hiện tại.**

```
int colorRes = getResources().getColor(colorResouceName, this.getTheme());
```

Phương thức getResources() lấy tập hợp các tài nguyên cho ứng dụng của bạn, và phương thức getColor() truy xuất một màu cụ thể từ các tài nguyên đó thông qua ID của tên màu. Tuy nhiên, getColor() bị gạch chân màu đỏ.

Nếu bạn trỏ chuột vào getColor(), Android Studio sẽ báo lỗi: "Call requires API 23 (current min is 15)". Vì minSdkVersion của bạn là 15, bạn sẽ nhận được thông báo này khi cố gắng sử dụng bất kỳ API nào được giới thiệu sau API 15. Bạn vẫn có thể biên dịch ứng dụng của mình, nhưng vì phiên bản getColor() này không khả dụng trên các thiết bị chạy API trước 23, ứng dụng của bạn sẽ bị lỗi khi người dùng nhấn nút **Change Color**.

Ở giai đoạn này, bạn có thể kiểm tra phiên bản nền tảng và sử dụng phiên bản phù hợp của getColor() tùy thuộc vào nơi ứng dụng đang chạy. Một cách tốt hơn để hỗ trợ cả các API Android cũ hơn và mới hơn mà không gặp cảnh báo là sử dụng một trong các lớp tương thích trong thư viện hỗ trợ.

6. Thay đổi dòng gán colorRes để sử dụng lớp ContextCompat:

```
int colorRes = ContextCompat.getColor( context: this, colorResourceName );
```

Lớp ContextCompat cung cấp nhiều phương thức hỗ trợ tương thích để xử lý sự khác biệt giữa các phiên bản API trong ngữ cảnh ứng dụng và tài nguyên của ứng dụng. Phương thức getColor() trong ContextCompat nhận hai đối số: ngữ cảnh hiện tại (trong trường hợp này là instance của Activity, tức this) và tên của màu sắc.

Việc triển khai phương thức này trong thư viện hỗ trợ ẩn đi sự khác biệt về cách thực hiện giữa các phiên bản API khác nhau. Bạn có thể gọi phương thức này mà không gặp bất kỳ cảnh báo, lỗi, hoặc sự cố nào, bất kể phiên bản SDK biên dịch hoặc phiên bản SDK tối thiểu của bạn là gì.

1. Đặt màu cho TextView bằng ID tài nguyên màu:

```
mHelloTextView.setTextColor(colorRes);
```

2. Chạy ứng dụng trên thiết bị hoặc trình giả lập, sau đó nhấn nút Change Color.

Nút **Change Color** bây giờ sẽ thay đổi màu văn bản trong ứng dụng, như minh họa bên dưới.



cessfully finished in 719 ms

Mã giải pháp

Giải pháp cho MainActivity

Dưới đây là trình xử lý sự kiện **changeColor()** trong **MainActivity**:

```
public void changeColor(View view) {  
    Random random = new Random();  
    String colorName = mColorArray[random.nextInt( bound: 20)];  
    int colorResourceName = getResources().getIdentifier(colorName, defType: "color", getApplicationContext().getPackageName());  
    int colorRes = ContextCompat.getColor( context: this, colorResourceName);  
    mHelloTextView.setTextColor(colorRes);  
}
```

Dự án Android Studio

Dự án Android Studio: HelloCompat

Thử thách lập trình

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

Thử thách: Thay vì sử dụng **ContextCompat** để lấy tài nguyên màu, hãy sử dụng kiểm tra giá trị trong lớp **Build** để thực hiện một thao tác khác nếu ứng dụng đang chạy trên thiết bị hỗ trợ phiên bản Android cũ hơn API 23.

Tóm tắt

Cài đặt Android Support Library:

- Sử dụng **SDK Manager** để cài đặt **Android Support Repository**. Chọn **Tools > Android > SDK Manager**, nhấp vào tab **SDK Tools** và mở rộng **Support Repository**.
- Nếu cột **Status** hiển thị **Installed** cho **Android Support Repository**, nhấp vào **Cancel**; nếu hiển thị **Not installed** hoặc **Update Available**, đánh dấu vào ô kiểm. Một biểu tượng tải xuống sẽ xuất hiện bên cạnh ô kiểm. Nhấp vào **OK**.

Android sử dụng ba chỉ thị để chỉ định cách ứng dụng của bạn hoạt động với các phiên bản API khác nhau:

- minSdkVersion:** phiên bản API tối thiểu mà ứng dụng của bạn hỗ trợ.
- compileSdkVersion:** phiên bản API mà ứng dụng của bạn được biên dịch.
- targetSdkVersion:** phiên bản API mà ứng dụng của bạn được thiết kế để chạy.

Quản lý dependencies trong dự án:

- Mở rộng **Gradle Scripts** trong mục **Project > Android**, và mở tệp **build.gradle (Module: app)**.

- Bạn có thể thêm dependencies trong phần **dependencies**.

Lớp ContextCompat cung cấp các phương thức hỗ trợ tương thích với ngữ cảnh và các phương thức liên quan đến tài nguyên cho cả API cũ và mới.

Khái niệm liên quan

Tài liệu về khái niệm liên quan nằm trong **3.3: The Android Support Library**.

Tìm hiểu thêm

Tài liệu Android Studio:

- **Android Studio User Guide**
Tài liệu nhà phát triển Android:
 - **Android Support Library (introduction)**
 - **Support Library Setup**
 - **Support Library Features**
 - **Supporting Different Platform Versions**
 - **Package Index** (tất cả các gói API bắt đầu bằng android.support).

Khác:

- **Picking your compileSdkVersion, minSdkVersion, and targetSdkVersion**
- **Understanding the Android Support Library**
- **All the Things Compat**

Bài tập về nhà

Chạy ứng dụng

Mở ứng dụng **HelloCompat** mà bạn đã tạo trong bài thực hành về cách sử dụng các thư viện hỗ trợ.

1. Đặt một điểm dừng gỡ lỗi (debugger breakpoint) trên dòng mã trong phương thức changeColor() nơi thực sự thay đổi màu sắc:

java

CopyEdit

```
int colorRes = ContextCompat.getColor(this, colorResourceName);
```

2. Chạy ứng dụng ở chế độ debug trên một thiết bị hoặc trình giả lập (emulator) đang chạy phiên bản API 23 hoặc mới hơn. Nhấp vào **Step Into** để bước vào phương thức getColor() và theo dõi các lời gọi phương thức sâu hơn trong ngăn xếp.
 - Kiểm tra cách lớp ContextCompat xác định phương pháp lấy màu từ tài nguyên và các lớp framework khác mà nó sử dụng.
 - Một số lớp có thể hiển thị cảnh báo rằng "mã nguồn không khớp với bytecode." Nhấp vào **Step Out** để quay lại một tệp mã nguồn đã biết hoặc tiếp tục nhấp vào **Step Into** cho đến khi trình gõ lỗi tự quay trở lại.
3. Lặp lại bước trước đó trên một thiết bị hoặc trình giả lập chạy phiên bản API cũ hơn 23.
 - Ghi nhận các đường dẫn khác nhau mà framework thực hiện để lấy màu.

Câu hỏi 1

Khi bạn **bước vào lần đầu tiên** bằng cách nhấp **Step Into** phương thức ContextCompat.getColor(), lớp nào sẽ xuất hiện? Chọn một:

- **MainActivity**
- **ContextCompat**
- **AppCompatActivity**
- **Context**

Câu hỏi 2

Trong lớp xuất hiện, câu lệnh nào được thực thi nếu phiên bản API là 23 hoặc mới hơn? Chọn một:

- **return context.getColor(id);**
- **return context.getResources().getColor(id);**
- **throw new IllegalArgumentException("permission is null");**
- **return mResources == null ? super.getResources() : mResources;**

Câu hỏi 3

Nếu bạn thay đổi phương thức ContextCompat.getColor() thành phương thức getColor(), điều gì sẽ xảy ra khi bạn chạy ứng dụng? Chọn một:

- Nếu minSdkVersion của bạn là 15, từ getColor sẽ bị gạch chân màu đỏ trong trình chỉnh sửa mã. Khi trỏ chuột vào, Android Studio sẽ báo: "Call requires API 23 (current min is 15)".
- Ứng dụng sẽ chạy mà không gặp lỗi trên các trình giả lập và thiết bị sử dụng API 23 hoặc mới hơn.
- Ứng dụng sẽ bị sập khi người dùng nhấn Change Color nếu trình giả lập hoặc thiết bị đang sử dụng API 17.
- Tất cả các điều trên.

Nộp ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Không có ứng dụng nào cần nộp cho bài tập về nhà này.

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) Thể và màu sắc
- 2.3) Bố cục thích ứng

Bài 3) Kiểm thử giao diện người dùng

- 3.1) Espresso cho việc kiểm tra UI

CHƯƠNG 3. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

- 1.1) AsyncTask
- 1.2) AsyncTask và AsyncTaskLoader
- 1.3) Broadcast receivers

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

- 2.1) Thông báo
- 2.2) Trình quản lý cảnh báo
- 2.3) JobScheduler

CHƯƠNG 4. 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