

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



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

Hà Nội, 2.2025

MỤC LỤC

Contents

CHƯƠNG 1. LÀM QUEN	3
Bài 1) Tạo ứng dụng đầu tiên.....	3
1.1) Android Studio và Hello World	3
1.2) Giao diện người dùng và chỉnh sửa bố cục.....	17
1.3) Trình chỉnh sửa bố cục	67
1.4) Văn bản và các chế độ cuộn.....	67
1.5) Tài nguyên có sẵn	67
Bài 2) Activities	67
2.1) Activity và Intent	67
2.2) Vòng đời của Activity và trạng thái	67
2.3) Intent ngầm định	67
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ.....	67
3.1) Trình gỡ lỗi.....	67
3.2) Kiểm thử đơn vị	67
3.3) Thư viện hỗ trợ.....	67
CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG	68
Bài 1) Tương tác người dùng.....	68
1.1) Hình ảnh có thể chọn	68
1.2) Các điều khiển nhập liệu.....	68
1.3) Menu và bộ chọn.....	68
1.4) Điều hướng người dùng	68
1.5) RecyclerView	68
Bài 2) Trải nghiệm người dùng thú vị	68
2.1) Hình vẽ, định kiểu và chủ đề.....	68
2.2) Thẻ và màu sắc.....	68
2.3) Bố cục thích ứng	68
Bài 3) Kiểm thử giao diện người dùng	68

3.1) Espresso cho việc kiểm tra UI.....	68
CHƯƠNG 3. LÀM VIỆC TRONG NỀN	68
Bài 1) Các tác vụ nền	68
1.1) AsyncTask	68
1.2) AsyncTask và AsyncTaskLoader	68
1.3) Broadcast receivers	68
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền	68
2.1) Thông báo.....	68
2.2) Trình quản lý cảnh báo.....	68
2.3) JobScheduler	68
CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG	69
Bài 1) Tùy chọn và cài đặt	69
1.1) Shared preferences	69
1.2) Cài đặt ứng dụng	69
Bài 2) Lưu trữ dữ liệu với Room	69
2.1) Room, LiveData và ViewModel	69
2.2) Room, LiveData và ViewModel	69

3.1) Trình gowx lỗi

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

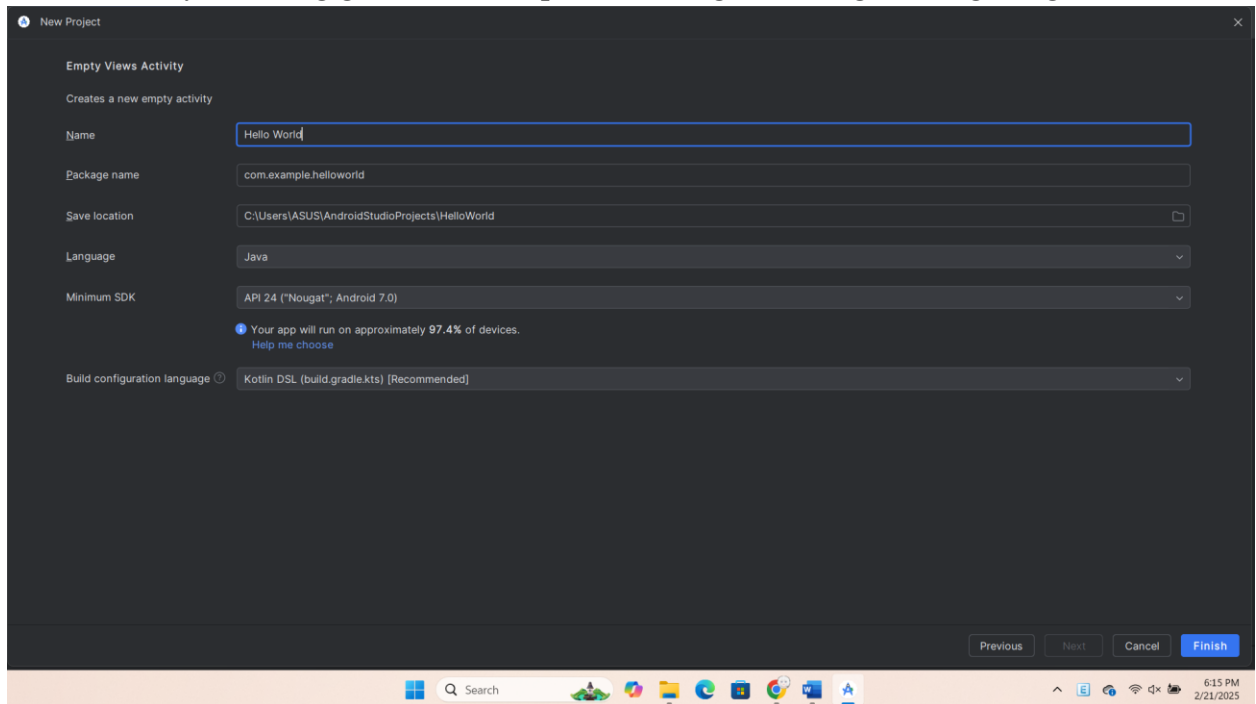
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

- Cài đặt môi trường phát triển **Android Studio**.

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

Tổng quan về ứng dụng

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

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

Task 1: Cài đặt Android Studio

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

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

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

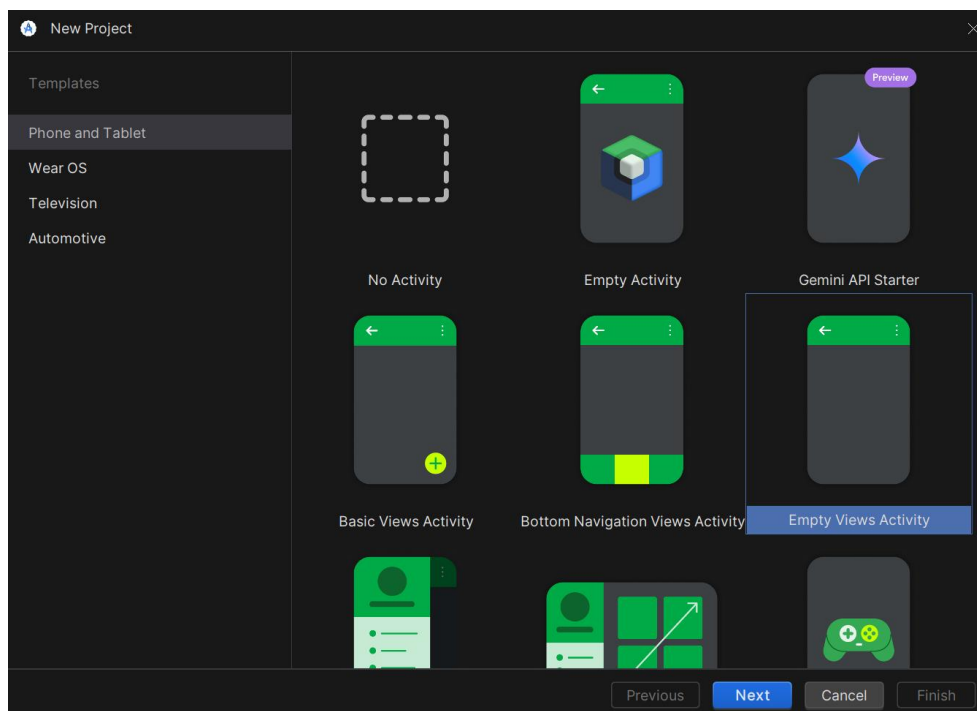
1. Điều hướng đến trang web dành cho nhà phát triển Android và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
2. Chấp nhận cấu hình mặc định cho tất cả các bước và đảm bảo rằng tất cả các thành phần được chọn để cài đặt.
3. Sau khi hoàn tất cài đặt, Trình hướng dẫn thiết lập sẽ tải xuống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

Task 2: Tạo ứng dụng Hello World

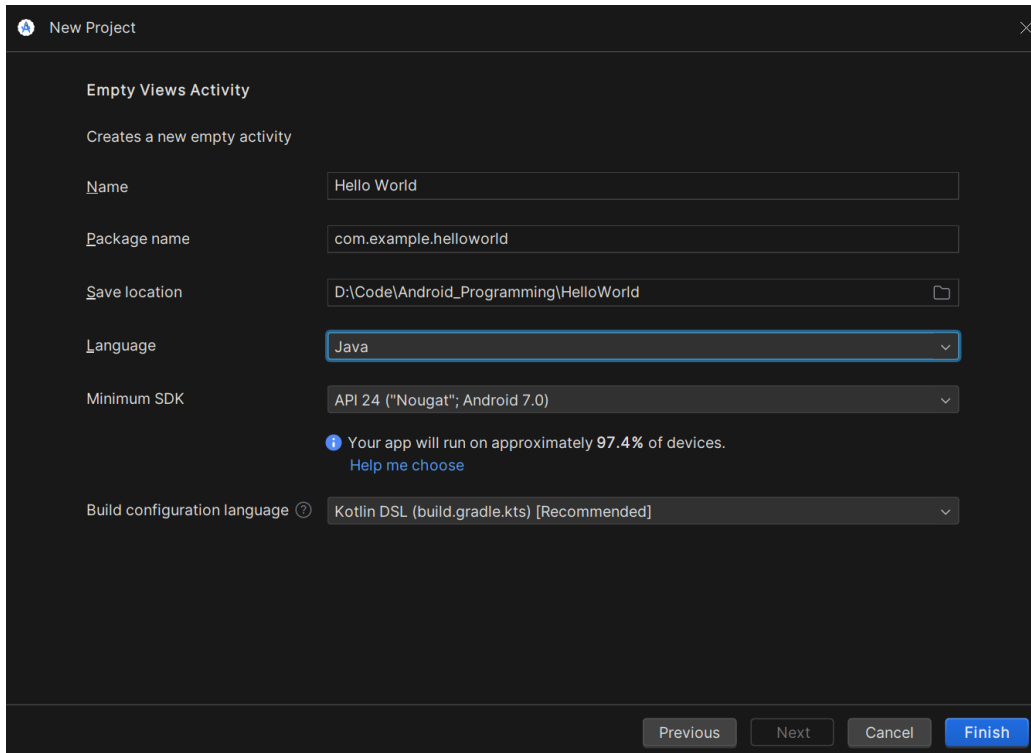
Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị "Hello World" để xác minh rằng Android Studio đã được cài đặt đúng cách và tìm hiểu những điều cơ bản về phát triển bằng Android Studio.

2.1 Tạo dự án

1. Mở **Android Studio**.
2. Trong cửa sổ chính **Welcom to Android Studio**, hãy nhấp vào **New Project**.
3. Một cửa sổ mới xuất hiện. Một **Activity** là một thành phần đơn lẻ, tập trung vào một tác vụ mà người dùng có thể thực hiện. Đây là một phần quan trọng trong bất kỳ ứng dụng Android nào. Một **Activity** thường có một **layout** đi kèm, xác định cách các thành phần giao diện người dùng (UI) hiển thị trên màn hình. **Android Studio** cung cấp các mẫu **Activity** để giúp bạn bắt đầu nhanh chóng. Đối với dự án Hello World, hãy chọn **Empty Views Activity** như hình bên dưới và nhấn **Next**.



4. Sau đó nhập **Hello World** vào **Application name**, chọn **Language** là **Java**.



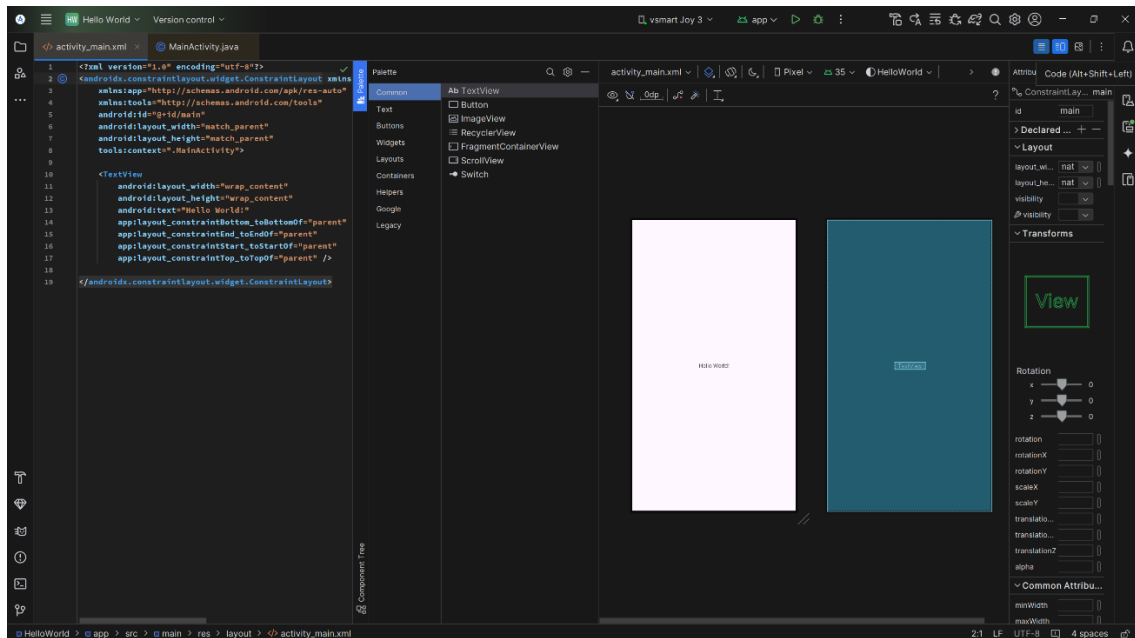
5. Xác định **Save location** là nơi bạn muốn lưu trữ ứng dụng **Hello World** và các dự án **Android Studio** khác hoặc thay đổi thành thư mục bạn thích.
6. Chấp nhận mặc định **android.example.helloworld** cho **Company Domain** hoặc tạo một tên miền công ty duy nhất.
Nếu bạn không có kế hoạch xuất bản ứng dụng của mình, bạn có thể chấp nhận mặc định. Lưu ý rằng việc thay đổi tên gói ứng dụng của bạn sau này là công việc bổ sung.
Sau đó ấn **Finish**.

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

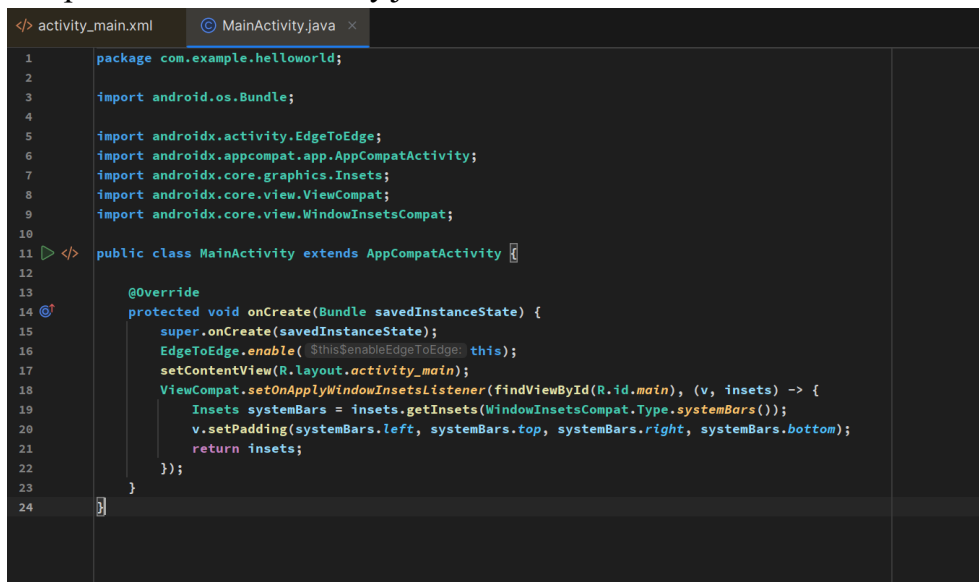
Trình chỉnh sửa Android Studio xuất hiện. Thực hiện theo các bước sau:

1. Nhấp vào tab **activity_main.xml** để xem trình chỉnh sửa bố cục.

2. Nhấp vào phần góc trên bên phải, bạn có thể chọn chế độ xem khác nhau



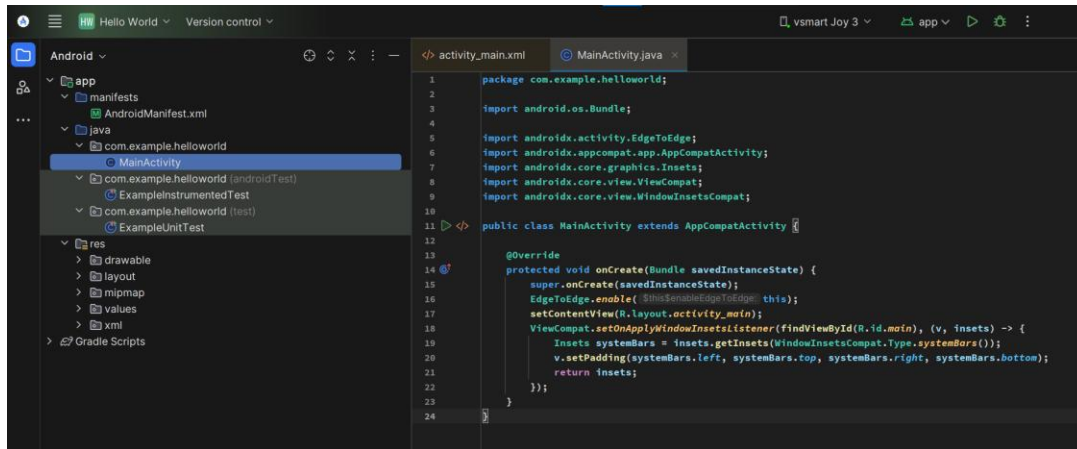
3. Nhấp vào tab MainActivity.java để xem code



2.2 Khám phá Project > Android Pane

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

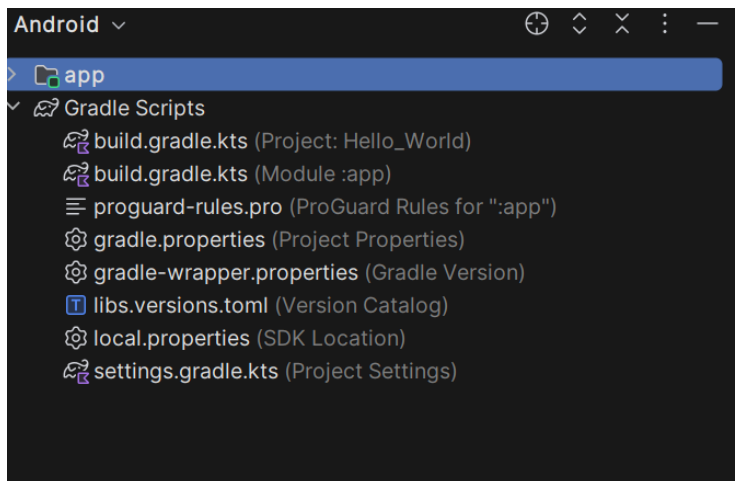
1. Nếu chưa chọn, hãy nhấp vào tab **Project** trong cột tab dọc ở phía bên trái của cửa sổ Android Studio.



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 phụ thuộc.

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



Thực hiện theo các bước sau để khám phá hệ thống Gradle:

1. Nếu thư mục **Gradle Scripts** 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 mô-đun tạo nên dự án của bạn. Mỗi dự án Android Studio đều chứa một tệp dựng Gradle cấp cao nhất. Hầu hết thời gian, bạn sẽ không cần thực hiện bất kỳ thay đổi nào đối với

tệp này, nhưng vẫn hữu ích khi hiểu nội dung của nó.

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

```
allprojects {
    repositories {
        google()
        jcenter()
    }
}
```

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

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

Tệp này thường là tệp cần chỉnh sửa khi thay đổi cấu hình cấp ứng dụng, chẳng hạn như khai báo các phụ thuộc trong phần phụ thuộc. Bạn có thể khai báo phụ thuộc thư viện bằng một trong một số cấu hình phụ thuộc khác nhau. Mỗi cấu hình phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện. Ví dụ, câu lệnh

implementation(libs.appcompat)

thêm thư viện AndroidX AppCompat, giúp duy trì khả năng tương thích với các phiên bản Android cũ hơn.

```

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

```

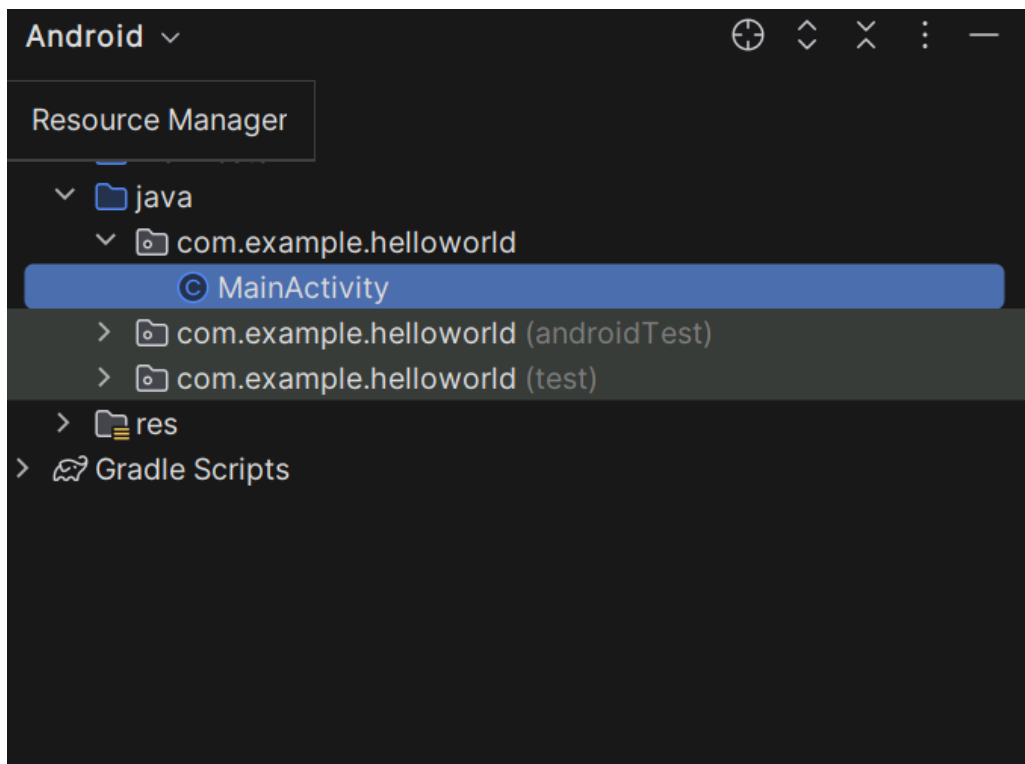
4. Nhấp vào hình tam giác để đóng Gradle Scripts.

2.4 Khám phá thư mục app và thư mục res

Tất cả code và tài nguyên (resource) cho ứng dụng đều nằm trong thư mục app và res.

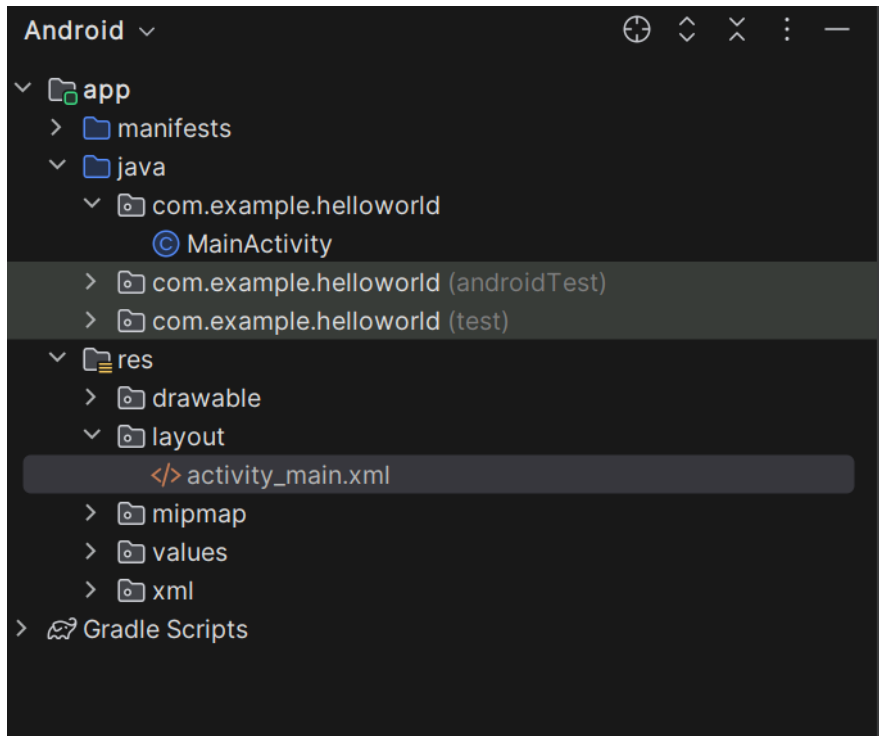
1. Mở rộng thư mục **app**, thư mục **java** và thư mục **com.example.android.helloworld** để xem tệp java **MainActivity**. Nhấp đúp vào tệp để mở tệp trong trình soạn thảo

mã.



Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như thể hiện trong hình trên. Thư mục **com.example.hello.helloworld** (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục khác được sử dụng để thử nghiệm và được mô tả trong bài học khác. Đối với ứng dụng Hello World, chỉ có một gói và nó chứa **MainActivity.java**. Tên của Activity đầu tiên (như trên hình) mà người dùng nhìn thấy, cũng khởi tạo các tài nguyên trên toàn ứng dụng, thường được gọi là MainActivity (phần mở rộng tệp bị bỏ qua trong **Dự án > Android pane**).

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



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

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, hệ thống phải có thông tin này trước khi có thể chạy bất kỳ mã nào của ứng dụng.

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

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

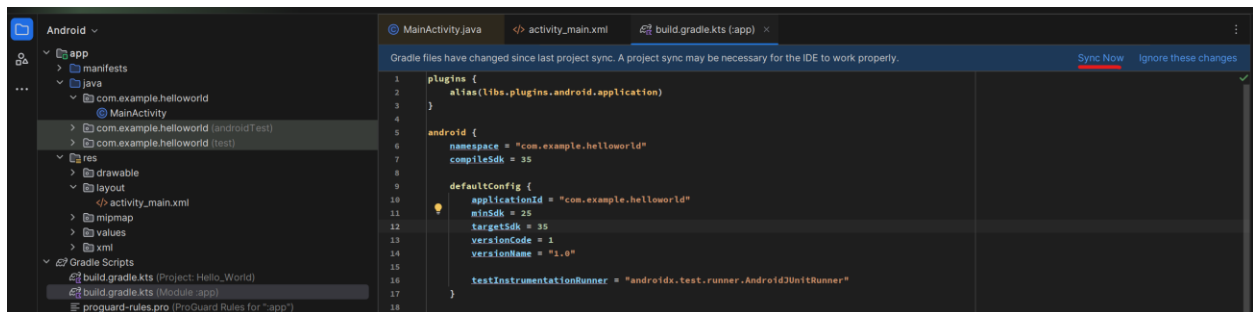
Task 5: Thay đổi cấu hình Gradle của ứng dụng

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

5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Thực hiện các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu nó chưa được mở và nhấp đúp vào tệp **build.gradle(Module:app)**.
2. Trong khối defaultConfig, hãy thay đổi giá trị của minSdkVersion thành 25 như hiển thị bên dưới (ban đầu giá trị này là 24).

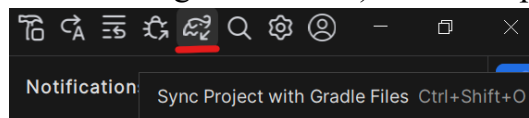


Trình chỉnh sửa code hiển thị thanh thông báo ở trên cùng với liên kết **Sync Now**.

5.2 Đồng bộ hóa với cấu hình Gradle mới

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

Để đồng bộ các tệp dự án, hãy nhấp vào Đồng bộ ngay trên thanh thông báo xuất hiện khi thực hiện thay đổi (như được hiển thị trong hình trước) hoặc nhấp vào biểu tượng



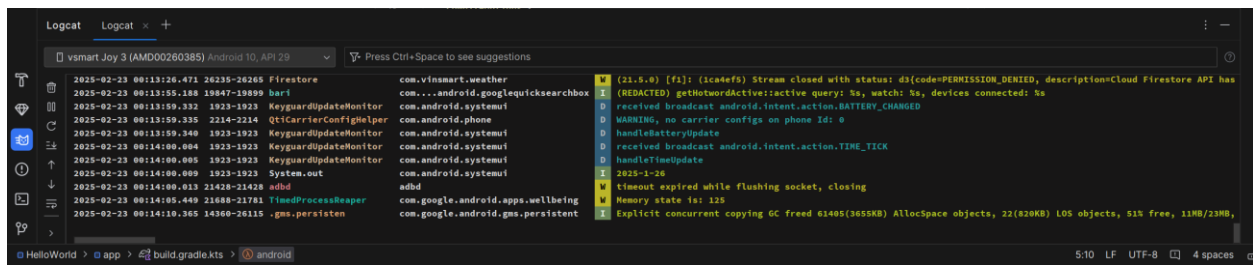
Sync Project with Gradle Files cụ.

trên thanh công

Khi quá trình đồng bộ hóa Gradle hoàn tất, thông báo Gradle build finished sẽ xuất hiện ở góc dưới bên trái của cửa sổ Android Studio. Để tìm hiểu sâu hơn về Gradle, hãy xem tài liệu Tổng quan về hệ thống bản dựng và Cấu hình bản dựng Gradle.

Task 6: Thêm các câu lệnh nhật ký (ghi log) vào ứng dụng của bạn

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



Như trên hình đang hiển thị thông tin Log trên thiết bị vật lí vsmart joy 3

6.2 Thêm câu lệnh log vào ứng dụng của bạn

Các câu lệnh Log trong code của bạn hiển thị thông báo trong ngăn Logcat. Ví dụ:

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

Các phần của tin nhắn bao gồm

- Log: Lóp Log để gửi tin nhắn nhật ký đến ngăn Logcat.
- d: Thiết lập mức Debug Log để lọc hiển thị tin nhắn nhật ký trong ngăn Logcat. Các mức nhật ký khác là e cho **Error**, w cho **Warn** và i cho **Info**.
- "MainActivity": Đối số đầu tiên là một thẻ có thể được sử dụng để lọc tin nhắn trong ngăn Logcat. Đây thường là tên của Activity mà tin nhắn bắt nguồn. Tuy nhiên, bạn có thể biến nó 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 Activity:

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

- "Hello World": Đối số thứ hai là thông điệp thực tế

Thực hiện theo các bước sau:

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

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

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

```

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

```

6. Nếu ngăn Logcat chưa mở, hãy nhấp vào tab **Logcat** ở cuối Android Studio để mở.
7. Kiểm tra xem tên mục tiêu và tên gói của ứng dụng đã đúng chưa.
8. Thay đổi mức Log trong ngăn **Logcat** thành **Debug**
9. Chạy ứng dụng của bạn.

Thông báo sau sẽ xuất hiện trong ngăn Logcat:

```

2025-02-23 00:57:24.452 29550-29550 MainActivity com.example.helloworld D Hello World

```

Coding challenge

Thử thách: Bây giờ bạn đã thiết lập và quen thuộc với quy trình phát triển cơ bản, hãy thực hiện các bước sau:

1. Tạo một dự án mới trong Android Studio.
2. Thay đổi lời chào "Hello World" thành "Happy Birthday to " và tên của một người có ngày sinh nhật gần đây.
3. (Tùy chọn) Chụp ảnh màn hình ứng dụng đã hoàn thành của bạn và gửi email cho một người có ngày sinh nhật mà bạn đã quên.
4. Một cách sử dụng phổ biến của lớp Log là ghi lại các ngoại lệ Java khi chúng xảy ra trong chương trình của bạn. Có một số phương thức hữu ích, chẳng hạn như Log.e(), mà bạn có thể sử dụng cho mục đích này. Khám phá phương pháp bạn có thể sử dụng để bao gồm một ngoại lệ với một thông báo Nhật ký. Sau đó, viết code trong ứng dụng của bạn để kích hoạt và ghi lại một ngoại lệ.

Tóm tắt

- Để cài đặt Android Studio, hãy truy cập Android Studio và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo ứng dụng mới, hãy đảm bảo rằng API 24: Android 4.0.3 IceCreamSandwich được đặt làm SDK tối thiểu.

- Để xem phân cấp Android của ứng dụng trong ngăn Dự án, hãy nhấp vào tab Dự án trong cột tab dọc, sau đó chọn Android trong menu bật lên ở trên cùng.
- Chỉnh sửa tệp build.gradle(Module:app) khi bạn cần thêm thư viện mới vào dự án hoặc thay đổi phiên bản thư viện.
- Tất cả mã và tài nguyên cho ứng dụng đều nằm trong thư mục **app** và **res**. Thư mục **java** bao gồm các hoạt động, bài kiểm tra và các thành phần khác trong mã nguồn Java. Thư mục **res** chứa các tài nguyên, chẳng hạn như bố cục, chuỗi và hình ảnh.
- Chỉnh sửa tệp AndroidManifest.xml để thêm các thành phần tính năng và quyền vào ứng dụng Android của bạn. Tất cả các thành phần cho một ứng dụng, chẳng hạn như nhiều hoạt động, phải được khai báo trong tệp XML này.
- Sử dụng trình quản lý Thiết bị ảo Android (AVD) để tạo một thiết bị ảo (còn được gọi là trình giả lập) để chạy ứng dụng của bạn.
- Thêm các câu lệnh Nhật ký vào ứng dụng của bạn, hiển thị các thông báo trong ngăn Logcat như một công cụ cơ bản để gỡ lỗi.
- Để chạy ứng dụng của bạn trên thiết bị Android vật lý bằng Android Studio, hãy bật Gỡ lỗi USB trên thiết bị.

Homework

Xây dựng và chạy ứng dụng

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

1.2) Giao diện người dùng và chỉnh sửa bố cục

a) Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là chế độ xem — mọi thành phần của màn hình là một View. Lớp View biểu thị khối xây dựng cơ bản cho tất cả các thành phần UI và là lớp

cơ sở cho các lớp cung cấp các thành phần UI tương tác như buttons, checkboxes và text entry fields. Các lớp con View thường được sử dụng được mô tả trong nhiều bài học bao gồm:

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

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

Ví dụ, code MainActivity trong ứng dụng Hello World hiển thị một bố cục được xác định trong tệp bố cục activity_main.xml, bao gồm một TextView có nội dung "Hello World".

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

Trong phần 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 ứng dụng bằng mẫu Empty Activity. Bạn cũng sẽ học cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục và cách chỉnh sửa bố cục trong XML. Bạn cần phát triển các kỹ năng này để có thể hoàn thành các phần thực hành khác trong khóa học này.

Những điều bạn nên biết

Bạn nên biết về:

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

Những gì bạn sẽ học được

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

Bạn sẽ làm gì

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

Tổng quan về ứng dụng

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

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

Task 1: Tạo và khám phá một dự án mới

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

1.1 Tạo dự án

1.2 Khám phá trình chỉnh sửa bố cục

Android Studio cung cấp trình chỉnh sửa bố cục để nhanh chóng xây dựng bố cục của các thành phần giao diện người dùng (UI) của ứng dụng. Nó cho phép bạn kéo các thành phần vào chế độ xem thiết kế trực quan và bản thiết kế, định vị chúng trong bố cục, thêm ràng buộc và đặt thuộc tính. Ràng buộc xác định vị trí của thành phần UI trong bố cục. Ràng

buộc thể hiện kết nối hoặc căn chỉnh với chế độ xem khác, bố cục cha hoặc hướng dẫn vô hình.

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

1. Trong thư mục **app > res > layout** trong ngăn **Project > Android**, hãy nhấp đúp vào tệp **activity_main.xml** để mở tệp đó nếu tệp đó chưa được mở.
2. Bạn sử dụng tab **Design** để thao tác các thành phần và bố cục, và tab **Code** để chỉnh sửa mã XML cho bố cục.
3. Ngăn **Palettes** hiển thị các thành phần UI mà bạn có thể sử dụng trong bố cục của ứng dụng.
4. Ngăn **Component tree** hiển thị hệ thống phân cấp chế độ xem của các thành phần UI. Các thành phần View được sắp xếp thành hệ thống phân cấp cây gồm các thành phần cha và con, trong đó thành phần con kế thừa các thuộc tính của thành phần cha. Trong hình trên, TextView là thành phần con của ConstraintLayout. Bạn sẽ tìm hiểu về các thành phần này sau trong bài học này.
5. Ngăn design và blueprint của trình chỉnh sửa bố cục hiển thị các thành phần UI trong bố cục. Trong hình trên, bố cục chỉ hiển thị một thành phần: TextView hiển thị "Hello World".
6. Tab **Attributes** hiển thị ngăn **Attributes** để thiết lập các thuộc tính cho thành phần UI.

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

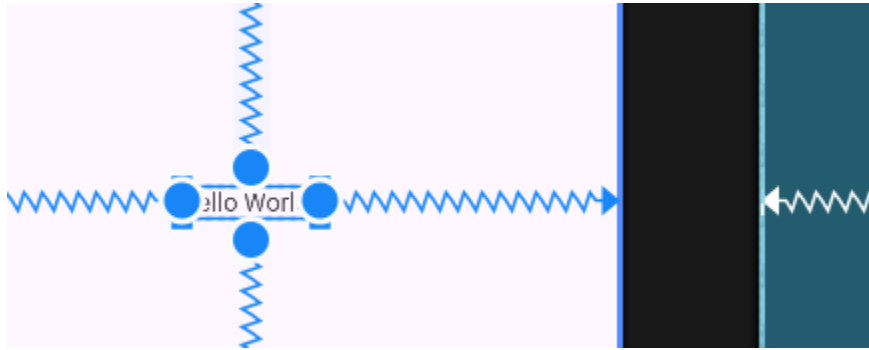
Trong tác vụ này, bạn tạo bố cục UI cho ứng dụng HelloToast trong trình chỉnh sửa bố cục bằng các tính năng ConstraintLayout. Bạn có thể tạo các ràng buộc theo cách thủ công, như được hiển thị sau, hoặc tự động bằng công cụ Autoconnect.

2.1Kiểm tra các ràng buộc phần tử

Thực hiện theo các bước sau:

1. Mở **activity_main.xml** từ ngăn **Project > Android** nếu nó chưa mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào tab đó.
2. Công cụ **Autoconnect** cũng nằm trên thanh công cụ. Theo mặc định, công cụ này được bật. Đối với bước này, hãy đảm bảo rằng công cụ không bị tắt.
3. Nhấp vào nút phóng to để phóng to các ngăn thiết kế và bản thiết kế để xem cận cảnh.
4. Chọn TextView trong ngăn Component Tree. TextView "Hello World" được tô sáng trong các ngăn thiết kế và bản thiết kế và các ràng buộc cho phần tử sẽ hiển thị.

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



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

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



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

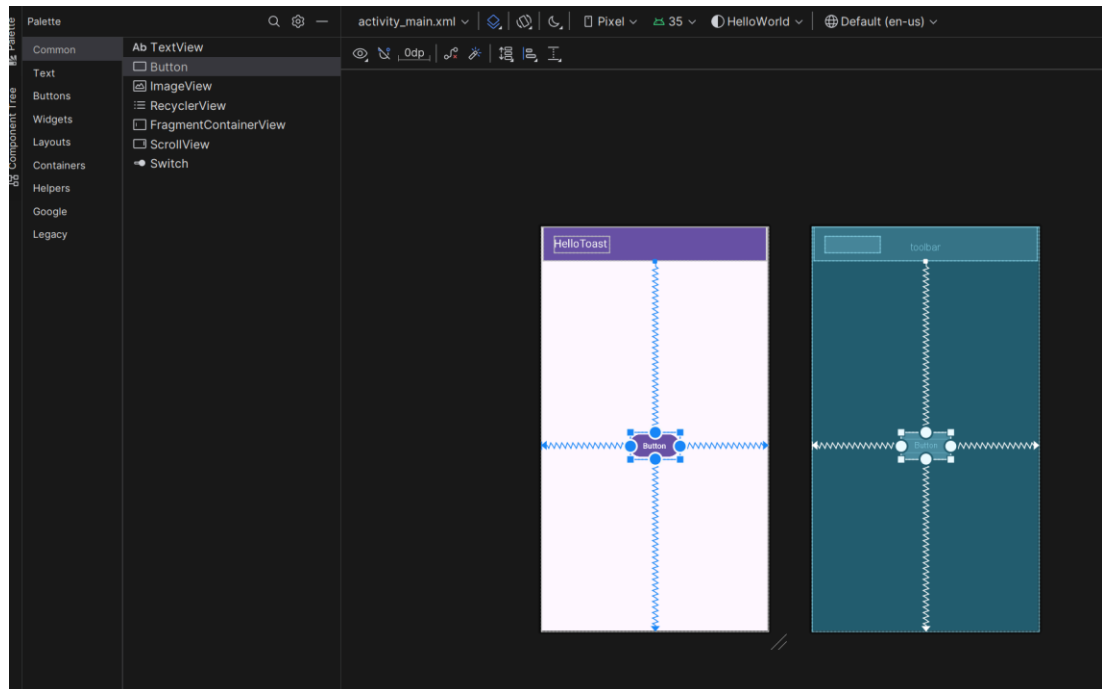


2.2 Thêm một nút vào trong layout.

Thực hiện theo các bước sau để thêm Nút:


1. Bắt đầu với một bảng trắng. Phần tử TextView không cần thiết, vì vậy khi nó vẫn được chọn, hãy nhấn phím Delete hoặc chọn Edit > Delete. Bây giờ bạn có một bố cục hoàn toàn trống.

2. Kéo một Button từ ngăn Palette đến bất kỳ vị trí nào trong bố cục. Nếu bạn thả Button vào vùng giữa trên cùng của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc lên trên cùng, bên trái và bên phải của bố cục như minh họa trong hình động bên dưới



2.3 Thêm nút thứ hai vào bố cục

1. Kéo một Button khác từ ngăn **Palette** màu vào giữa bố cục như minh họa trong hình động bên dưới. Autoconnect có thể cung cấp các ràng buộc theo chiều ngang cho bạn (nếu không, bạn có thể tự kéo chúng).
2. Kéo một ràng buộc theo chiều dọc vào cuối bố cục.

Bạn có thể xóa ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di con trỏ qua nó để hiển thị nút **Clear Constraints** . Nhấp vào nút này để xóa tất cả các ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc duy nhất, hãy nhấp vào tay cầm cụ thể đặt ràng buộc đó. Để xóa tất cả các ràng buộc trong toàn bộ bố cục, hãy nhấp vào công cụ **Clear All Constraints** trên thanh công cụ. Công cụ này hữu ích nếu bạn muốn làm lại tất cả các ràng buộc trong bố cục của mình.

Task 3: Thay đổi thuộc tính thành phần UI

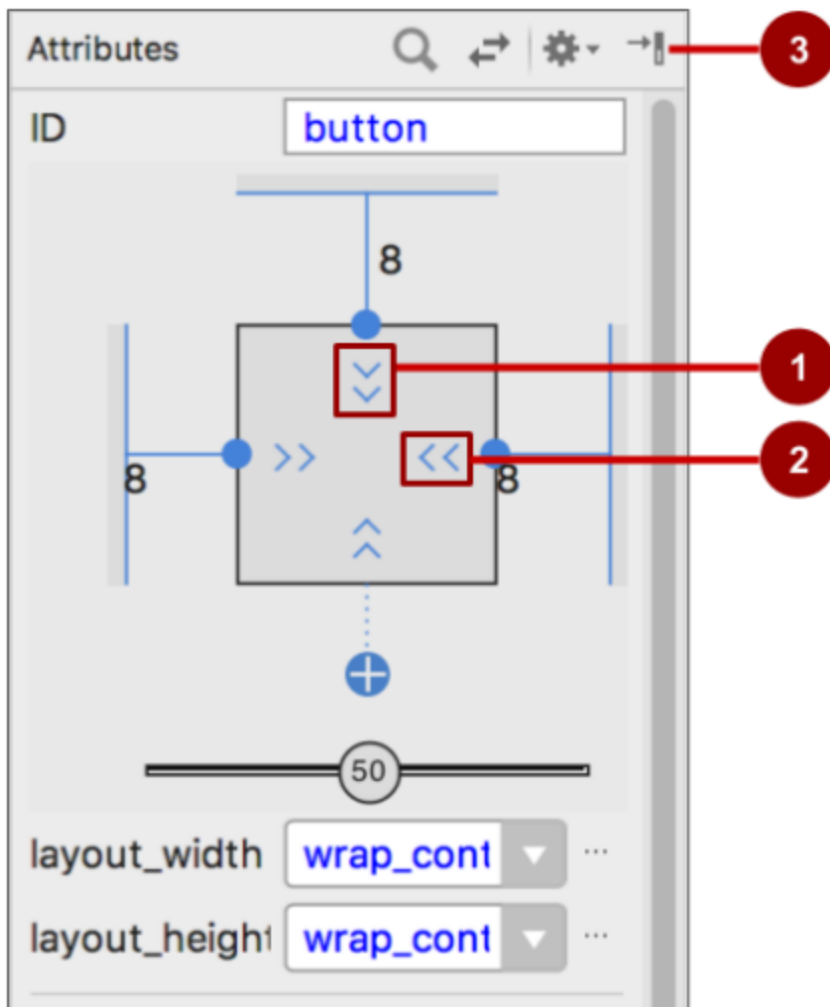
Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Bạn có thể tìm thấy các thuộc tính (được gọi là thuộc tính) chung cho tất cả các chế độ xem trong tài liệu lớp Chế độ xem.

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

3.1 Thay đổi kích thước Button

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

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



Trong hình trên:

1. **Height control.** Control này chỉ định thuộc tính `layout_height` và xuất hiện trong hai đoạn ở phía trên và phía dưới của hình vuông. Các góc cho biết control này được đặt thành `wrap_content`, nghĩa là View sẽ mở rộng theo chiều dọc khi cần để vừa với nội dung của nó. "8" cho biết lề chuẩn được đặt thành 8dp.
2. **Width control.** Control này chỉ định `layout_width` và xuất hiện trong hai đoạn ở phía trái và phía phải của hình vuông. Các góc cho biết control này được đặt thành `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần để vừa với nội dung của nó, lên đến lề 8dp.
3. Nút đóng ngăn Thuộc tính. Nhấp để đóng ngăn.

Thực hiện các bước sau:

1. Chọn nút trên cùng trong ngăn **Component Tree**.

2. Nhấp vào tab **Attributes** ở phía bên phải của cửa sổ trình chỉnh sửa bố cục.
3. Nhấp vào nút điều khiển chiều rộng hai lần—nhấp chuột đầu tiên sẽ thay đổi thành **Fixed** với các đường thẳng và nhấp chuột thứ hai sẽ thay đổi thành **Match Constraints** có cuộn lò xo, như thể hiện trong hình động bên dưới.

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

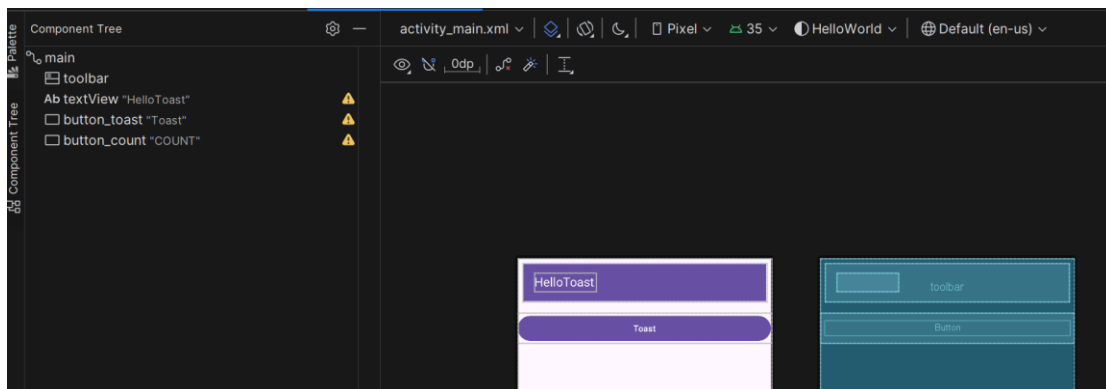
3.2 Thay đổi thuộc tính của Button

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

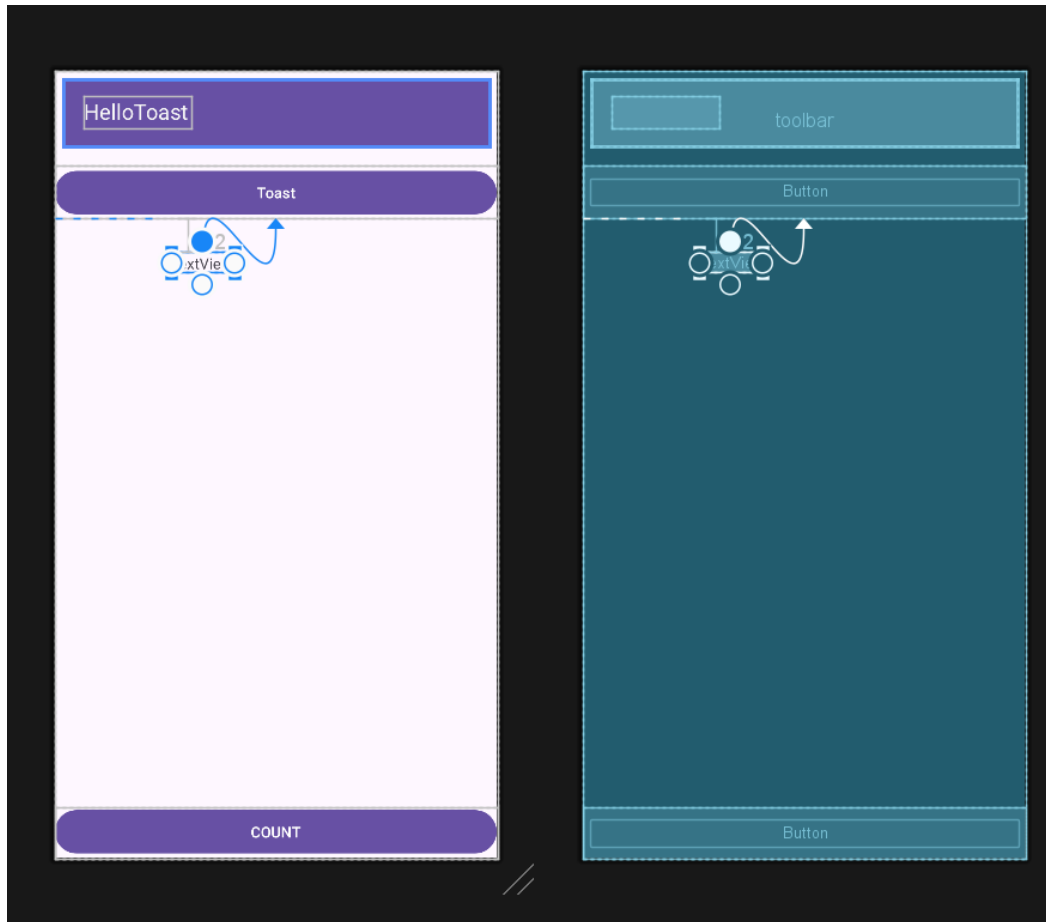
Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính mà bạn có thể gán cho một phần tử View. Bạn có thể nhập giá trị cho từng thuộc tính, chẳng hạn như các thuộc tính `android:id`, `background`, `textColor` và `text`.

Task 4: Thêm TextEdit và thiết lập thuộc tính cho nó

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



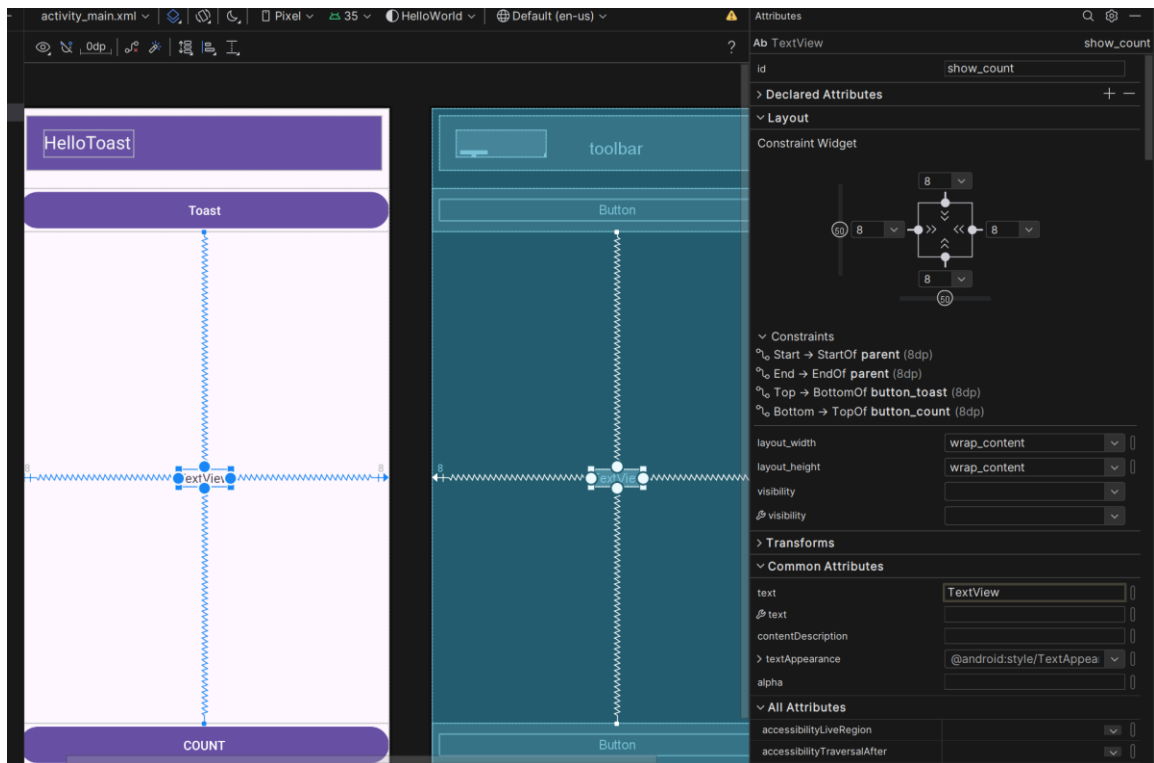
2. Như được hiển thị trong hình động bên dưới, hãy kéo một ràng buộc từ dưới cùng của `TextView` đến tay cầm ở trên cùng của Nút đếm và từ các cạnh của `TextView` đến các cạnh của bố cục. Điều này ràng buộc `TextView` ở giữa bố cục giữa hai phần tử Nút.



4.2 Thiết lập thuộc tính của TextView

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

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).
5. Thay đổi các điều khiển kích thước chế độ xem theo chiều ngang và chiều dọc (layout_width và layout_height) thành **match_constraint**.
6. Đặt textColor thành **@color/colorPrimary**.
7. Cuộn xuống ngăn và nhấp vào **View all attributes**, cuộn xuống trang thuộc tính thứ hai đến phần nền, sau đó nhập **#FFF00** để có màu vàng.
8. Cuộn xuống phần trọng lực, mở rộng trọng lực và chọn **center_ver** (để căn giữa theo chiều dọc).



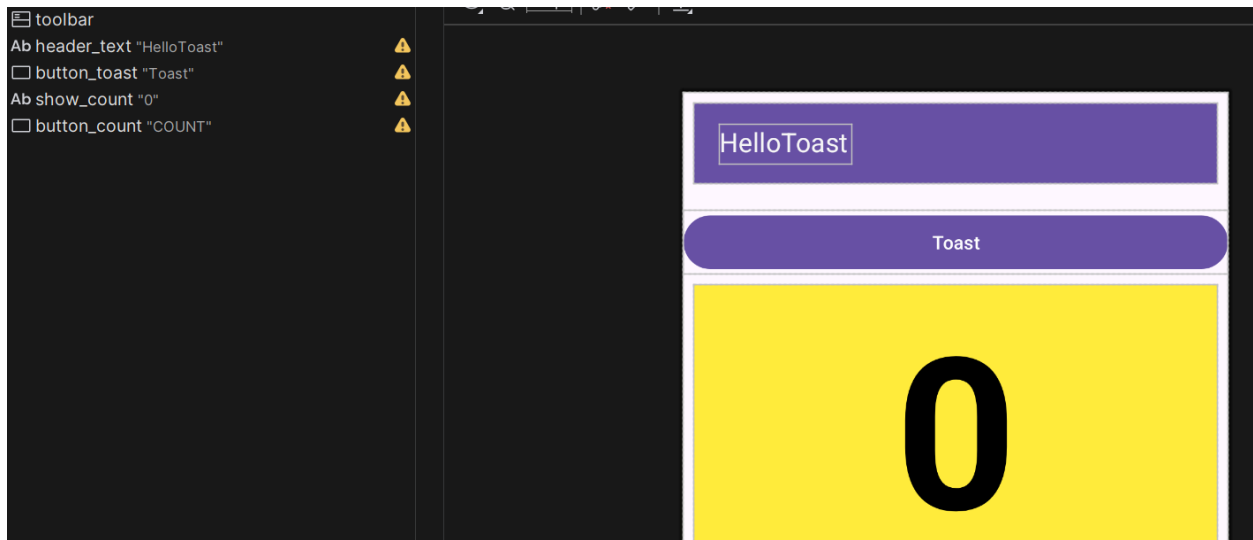
- **textSize:** Kích thước văn bản của TextView. Đối với bài học này, kích thước được đặt thành 160sp. Sp là viết tắt của pixel không phụ thuộc tỷ lệ và giống như dp, là đơn vị tỷ lệ với mật độ màn hình và sở thích về kích thước phông chữ của người dùng. Sử dụng đơn vị dp khi bạn chỉ định kích thước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

Bạn có thể nhận thấy rằng thuộc tính nền nằm trên trang đầu tiên của ngăn **Attributes** cho Button, nhưng lại nằm trên trang thứ hai của ngăn **Attributes** cho TextView. Ngăn **Attributes** thay đổi cho từng loại Chế độ xem: Các thuộc tính phổ biến nhất cho loại Chế độ xem xuất hiện trên trang đầu tiên và các thuộc tính còn lại được liệt kê trên trang thứ

hai. Để quay lại trang đầu tiên của ngăn **Attributes**, hãy nhấp vào biểu tượng  trên thanh công cụ ở đầu ngăn.

Task 5: Chỉnh sửa bố cục trong XML

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



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

5.1 Mở code XML cho bố cục

Đối với tác vụ này, hãy mở tệp **activity_main.xml** nếu tệp này chưa mở và nhấp vào tab Code bên góc trên phải.

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

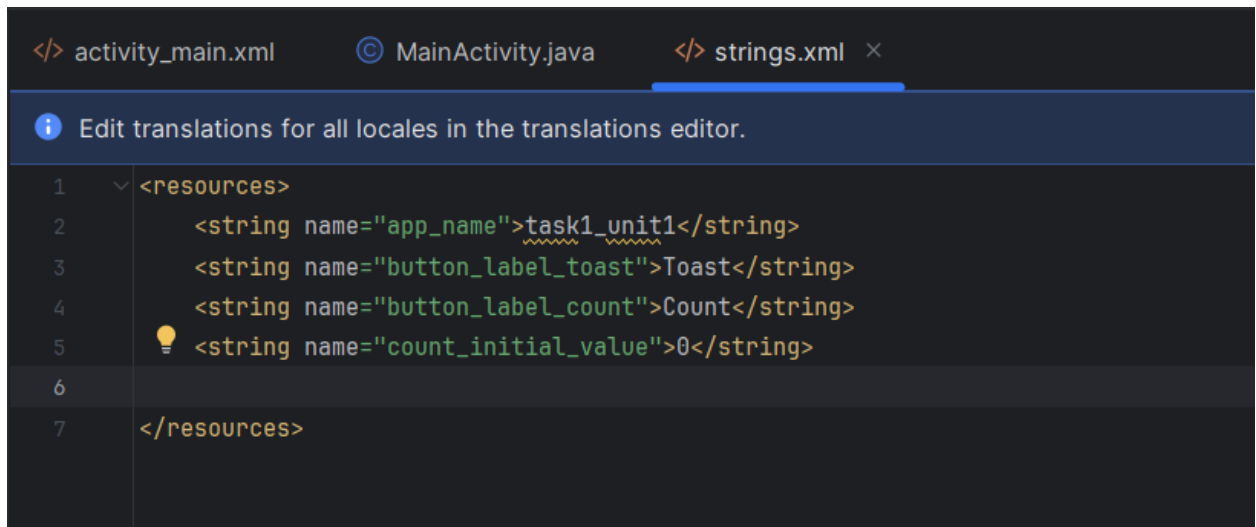
```
activity_main.xml x MainActivity.java strings.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:background="#FFFFFF"
8     tools:context=".MainActivity">
9
10     <!-- Tiêu đề -->
11     <TextView
12         android:id="@+id/title_text"
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:background="#563D9C"
16         android:padding="16dp"
17         android:text="Hello Toast"
18         android:textAlignment="textStart"
19         android:textColor="#FFFFFF"
20         android:textSize="24sp"
21         android:textStyle="bold"
22         app:layout_constraintTop_toTopOf="parent" />
23
24     <!-- Nút Toast -->
25     <Button
26         android:id="@+id/button_toast"
27         android:layout_width="0dp"
28         android:layout_height="wrap_content"
29         android:layout_marginTop="16dp"
30         android:background="#563D9C"
31         android:text="TOAST"
32         android:textColor="#FFFFFF"
33         android:onClick="showToast"
34         app:layout_constraintTop_toBottomOf="@id/title_text"
35     />
36 </androidx.constraintlayout.widget.ConstraintLayout>
37 </android.support.design.widget.CoordinatorLayout>
38 </android.support.design.widget.CoordinatorLayout>
39 </android.support.design.widget.CoordinatorLayout>
40 </android.support.design.widget.CoordinatorLayout>
41 </android.support.design.widget.CoordinatorLayout>
42 </android.support.design.widget.CoordinatorLayout>
43 </android.support.design.widget.CoordinatorLayout>
44 </android.support.design.widget.CoordinatorLayout>
45 </android.support.design.widget.CoordinatorLayout>
46 </android.support.design.widget.CoordinatorLayout>
47 </android.support.design.widget.CoordinatorLayout>
48 </android.support.design.widget.CoordinatorLayout>
49 </android.support.design.widget.CoordinatorLayout>
50 </android.support.design.widget.CoordinatorLayout>
51 </android.support.design.widget.CoordinatorLayout>
52 </android.support.design.widget.CoordinatorLayout>
53 </android.support.design.widget.CoordinatorLayout>
54 </android.support.design.widget.CoordinatorLayout>
55 </android.support.design.widget.CoordinatorLayout>
56 </android.support.design.widget.CoordinatorLayout>
57 </android.support.design.widget.CoordinatorLayout>
58 </android.support.design.widget.CoordinatorLayout>
59 </android.support.design.widget.CoordinatorLayout>
60 </android.support.design.widget.CoordinatorLayout>
61 </android.support.design.widget.CoordinatorLayout>
62 </android.support.design.widget.CoordinatorLayout>
63 </android.support.design.widget.CoordinatorLayout>
64 </android.support.design.widget.CoordinatorLayout>
65 </android.support.design.widget.CoordinatorLayout>
66 </android.support.design.widget.CoordinatorLayout>
67 </android.support.design.widget.CoordinatorLayout>
68 </android.support.design.widget.CoordinatorLayout>
69 </android.support.design.widget.CoordinatorLayout>
70 </android.support.design.widget.CoordinatorLayout>
71 </android.support.design.widget.CoordinatorLayout>
72 </android.support.design.widget.CoordinatorLayout>
73 </android.support.design.widget.CoordinatorLayout>
74 </android.support.design.widget.CoordinatorLayout>
75 </android.support.design.widget.CoordinatorLayout>
76 </android.support.design.widget.CoordinatorLayout>
77 </android.support.design.widget.CoordinatorLayout>
78 </android.support.design.widget.CoordinatorLayout>
79 </android.support.design.widget.CoordinatorLayout>
80 </android.support.design.widget.CoordinatorLayout>
81 </android.support.design.widget.CoordinatorLayout>
82 </android.support.design.widget.CoordinatorLayout>
83 </android.support.design.widget.CoordinatorLayout>
84 </android.support.design.widget.CoordinatorLayout>
85 </android.support.design.widget.CoordinatorLayout>
86 </android.support.design.widget.CoordinatorLayout>
87 </android.support.design.widget.CoordinatorLayout>
88 </android.support.design.widget.CoordinatorLayout>
89 </android.support.design.widget.CoordinatorLayout>
90 </android.support.design.widget.CoordinatorLayout>
91 </android.support.design.widget.CoordinatorLayout>
92 </android.support.design.widget.CoordinatorLayout>
93 </android.support.design.widget.CoordinatorLayout>
94 </android.support.design.widget.CoordinatorLayout>
95 </android.support.design.widget.CoordinatorLayout>
96 </android.support.design.widget.CoordinatorLayout>
97 </android.support.design.widget.CoordinatorLayout>
98 </android.support.design.widget.CoordinatorLayout>
99 </android.support.design.widget.CoordinatorLayout>
100 </android.support.design.widget.CoordinatorLayout>
```

5.2 Trích xuất tài nguyên string

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

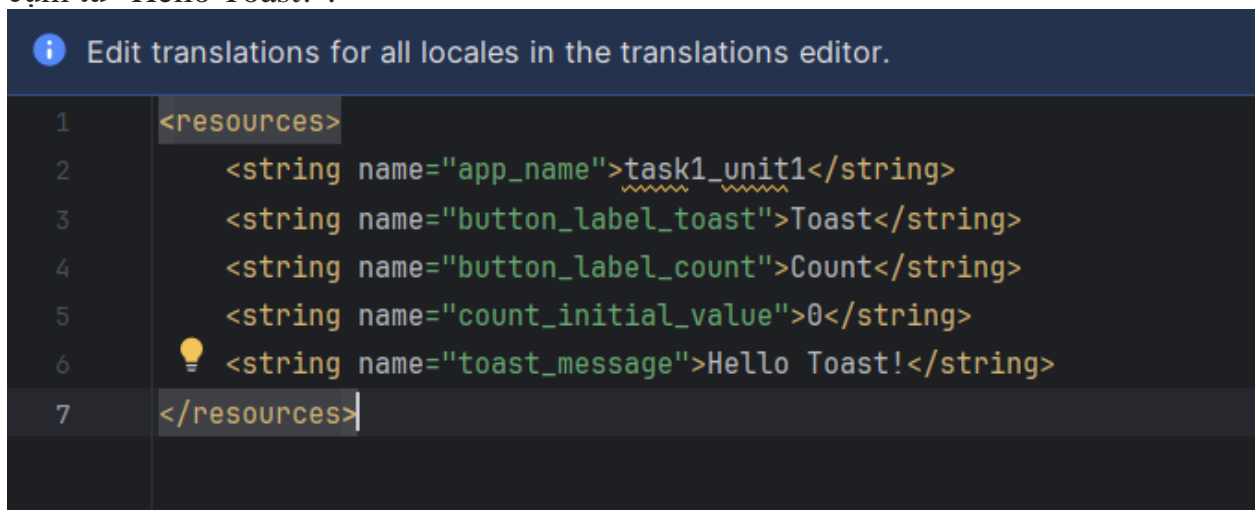
1. Nhấp một lần vào từ "Toast" (cảnh báo đầu tiên được tô sáng).
2. Nhấn **Alt-Enter** trong Windows hoặc **Option-Enter** trong macOS và chọn **Extract string resource** từ menu bật lên.
3. Nhập **button_label_toast** cho tên Tài nguyên.
4. Nhấp vào OK. Một tài nguyên chuỗi được tạo trong tệp values/res/string.xml và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: `@string/button_label_toast`
5. Trích xuất các chuỗi còn lại: **button_label_count** cho "Count" và **count_initial_value** cho "0".

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



```
</> activity_main.xml    MainActivity.java    </> strings.xml x
i Edit translations for all locales in the translations editor.
1  <resources>
2      <string name="app_name">task1_unit1</string>
3      <string name="button_label_toast">Toast</string>
4      <string name="button_label_count">Count</string>
5      <string name="count_initial_value">0</string>
6
7  </resources>
```

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



```
i Edit translations for all locales in the translations editor.
1  <resources>
2      <string name="app_name">task1_unit1</string>
3      <string name="button_label_toast">Toast</string>
4      <string name="button_label_count">Count</string>
5      <string name="count_initial_value">0</string>
6      <string name="toast_message">Hello Toast!</string>
7  </resources>
```

Task 6: Thêm thuộc tính onClick và trình xử lý vào mỗi Button

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

1. Khi trình soạn thảo XML mở (tab Văn bản), hãy tìm Nút có android:id được đặt thành `button_toast`
2. Thêm thuộc tính `android:onClick` vào cuối phần tử `button_toast` sau thuộc tính cuối cùng và trước chỉ báo kết thúc `</>`

```
android:onClick="showToast"
```

3. Nhấp vào biểu tượng bóng đèn màu đỏ xuất hiện bên cạnh thuộc tính. Chọn **Create click handler**, chọn **MainActivity** và nhấp vào **OK**. Nếu biểu tượng bóng đèn màu đỏ không xuất hiện, hãy nhấp vào tên phương thức ("`showToast`"). Nhấn **Alt-Enter** (**Option-Enter** trên máy Mac), chọn **Create 'showToast(view)'** trong **MainActivity** và nhấp vào **OK**. Hành động này tạo một stub phương thức giữ chỗ cho phương thức `showToast()` trong `MainActivity`, như được hiển thị ở cuối các bước này.
4. Lặp lại hai bước cuối cùng với `Button_count`: Thêm thuộc tính `android:onClick` vào cuối và thêm trình xử lý nhấp:

```
android:onClick="countUp"
```

5. Sau các bước trên trong file `MainActivity.java` sẽ hiện như sau:

```
</> activity_main.xml    MainActivity.java    </> strings.xml
1      package com.example.task1_unit1;
2
3      > import ...
9
10     </> public class MainActivity extends AppCompatActivity {
11         2 usages
12         private int mCount = 0;
13         3 usages
14         private TextView mShowCount;
15
16         @Override
17         protected void onCreate(Bundle savedInstanceState) {
18             super.onCreate(savedInstanceState);
19             setContentView(R.layout.activity_main);
20
21             mShowCount = findViewById(R.id.show_count);
22         }
23
24         // Hiển thị Toast khi nhấn nút TOAST
25         1 usage
26         public void showToast(View view) {...}
27
28         // Tăng số đếm khi nhấn nút COUNT
29         1 usage
30         public void countUp(View view) {...}
31
32     }
33
34
35
36
```

6.2 Chỉnh sửa sự kiện của Toast Button

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

Thực hiện theo các bước sau để chỉnh sửa trình xử lý nhấp chuột vào Nút Toast:

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


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

- Để tạo một phiên bản Toast, hãy gọi phương thức `makeText()` trên lớp `Toast`.

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

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

- Cung cấp ngữ cảnh của Activity ứng dụng. Vì **Toast** hiển thị ở đầu Activity UI, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong ngữ cảnh của Activity có ngữ cảnh bạn cần, hãy sử dụng điều này như một phím tắt.

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

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

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

- Cung cấp thời lượng hiển thị. Ví dụ: `Toast.LENGTH_SHORT` hiển thị toast trong thời gian tương đối ngắn.

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

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

- Hiển thị Toast bằng cách gọi `show()`. Sau đây là toàn bộ phương thức `showToast()`:

```
// Hiển thị Toast khi nhấn nút TOAST  
1 usage  
public void showToast(View view) {  
    Toast toast = Toast.makeText(context: this, text: "Hello Toast!", Toast.LENGTH_SHORT);  
    toast.show();  
}
```

Hello Toast

TOAST

10

Hello Toast!

COUNT



6.3 Xử lý sự kiện nút Count

Bây giờ bạn sẽ chỉnh sửa phương thức **countUp()**—trình xử lý nhấp vào Nút Count trong **MainActivity**—để nó hiển thị số đếm hiện tại sau khi Count được chạm vào. Mỗi lần chạm sẽ tăng số đếm lên một.

1. Tạo một biến `int mCount` làm thuộc tính của class `MainActivity` và khởi tạo nó bằng 0.

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

2. Trong phương thức `countUp()` thêm dòng code sau:

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

3. Cùng với biến trên, bạn cũng cần một biến thuộc tính `private` để tham chiếu đến `TextView show_count`, mà bạn sẽ thêm vào trình xử lý nhấp chuột. Gọi biến này là `mShowCount`:

```
public class MainActivity extends AppCompatActivity {  
    2 usages  
    private int mCount = 0;  
    3 usages  
    private TextView mShowCount;  
}
```

4. Thêm code vào hàm `onCreate()`

```

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

    mShowCount = findViewById(R.id.show_count);
}

```

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

```

// Tăng số đếm khi nhấn nút COUNT
1 usage
public void countUp(View view) {
    mCount++;
    if (mShowCount != null) {
        mShowCount.setText(String.valueOf(mCount));
    }
}

```

Và đây là kết quả

Hello Toast

TOAST

10

Hello Toast!

COUNT



Tổng quan

View, ViewGroup và layout:

- Tất cả các thành phần UI đều là lớp con của lớp **View** và do đó kế thừa nhiều thuộc tính của siêu lớp **View**.
- Các thành phần View có thể được nhóm bên trong **ViewGroup**, hoạt động như một vùng chứa. Mối quan hệ là cha-con, trong đó cha là ViewGroup và con là **View** hoặc **ViewGroup** khác.
- Phương thức **onCreate()** được sử dụng để mở rộng bố cục, nghĩa là đặt chế độ xem nội dung của màn hình thành bố cục XML. Bạn cũng có thể sử dụng phương thức này để lấy tham chiếu đến các thành phần UI khác trong bố cục.
- View, giống như chuỗi, là một tài nguyên có thể có id. Lệnh gọi **findViewById** lấy ID của chế độ xem làm tham số và trả về **View**.

Sử dụng trình chỉnh sửa bố cục:

- Nhấp vào tab **Design** để thao tác các thành phần và bố cục, và tab **Code** để chỉnh sửa mã XML cho bố cục.
- Trong tab **Design**, ngăn **Palettes** hiển thị các thành phần UI mà bạn có thể sử dụng trong bố cục của ứng dụng và ngăn **Component Tree** hiển thị thứ bậc chế độ xem của các thành phần UI.
- Ngăn thiết kế và bản thiết kế của trình chỉnh sửa bố cục hiển thị các thành phần UI trong bố cục.
- Tab **Attributes** hiển thị ngăn **Attributes** để thiết lập thuộc tính cho thành phần UI.
- Constraint handle : Nhấp vào Constraint handle, được hiển thị dưới dạng hình tròn ở mỗi bên của thành phần, sau đó kéo đến tay cầm ràng buộc khác hoặc đến ranh giới cha để tạo ràng buộc. Ràng buộc được biểu thị bằng đường ngoằn ngoèo.
- Resizing handle: Bạn có thể kéo tay cầm thay đổi kích thước hình vuông để thay đổi kích thước thành phần. Trong khi kéo, tay cầm sẽ thay đổi thành góc nghiêng.
- Khi được bật, công cụ Autoconnect sẽ tự động tạo hai hoặc nhiều ràng buộc cho thành phần UI với bố cục cha. Sau khi bạn kéo phần tử vào bố cục, nó sẽ tạo ra các ràng buộc dựa trên vị trí của phần tử.
- Bạn có thể xóa các ràng buộc khỏi phần tử bằng cách chọn phần tử và di con trỏ qua phần tử đó để hiển thị nút Xóa ràng buộc. Nhấp vào nút này để xóa tất cả các ràng

buộc trên phần tử đã chọn. Để xóa một ràng buộc duy nhất, hãy nhấp vào tay cầm cụ thể đặt ràng buộc đó.

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

Thiết lập chiều rộng và chiều cao của bố cục:

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

- Thiết lập `match_constraint` mở rộng chế độ xem để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao—lên đến một lề, nếu có.
- Thiết lập `wrap_content` thu nhỏ kích thước chế độ xem để chế độ xem chỉ đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, chế độ xem sẽ trở nên vô hình.
- Sử dụng `Fixed dp` (pixel không phụ thuộc mật độ) cố định để chỉ định kích thước cố định, được điều chỉnh theo kích thước màn hình của thiết bị.

Trích xuất tài nguyên chuỗi:

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

1. Nhấp một lần vào chuỗi được mã hóa cứng để trích xuất, nhấn **Alt-Enter (Option-Enter)** trên máy Mac) và chọn **Extract string resources** chuỗi từ menu bật lên.
2. Đặt tên Tài nguyên.
3. Nhấp vào **OK**. Thao tác này tạo một tài nguyên chuỗi trong tệp **res/values/string.xml** và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: `@string/button_label_toast`

b) Chỉnh sửa bố cục

Giới thiệu

Như bạn đã học trong 1.2 Phần A: Giao diện người dùng tương tác đầu tiên của bạn, bạn có thể xây dựng giao diện người dùng (UI) bằng `ConstraintLayout` trong trình chỉnh sửa

bố cục, nơi đặt các thành phần UI vào một bố cục bằng cách sử dụng các kết nối ràng buộc với các thành phần khác và với các cạnh bố cục.

ConstraintLayout được thiết kế để giúp bạn dễ dàng kéo các thành phần UI vào trình chỉnh sửa bố cục. ConstraintLayout là một ViewGroup, đây là một View đặc biệt có thể chứa các đối tượng View khác (được gọi là các chế độ xem con hoặc chế độ xem con). Bài thực hành này sẽ trình bày thêm các tính năng của ConstraintLayout và trình chỉnh sửa bố cục.

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

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

Những điều bạn nên biết

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

Những gì bạn sẽ học được

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

Bạn sẽ làm gì

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

- Định vị các thành phần trong `LinearLayout`.
- Thay đổi bố cục để sử dụng `RelativeLayout`.
- Sắp xếp lại các chế độ xem trong bố cục chính để tương đối với nhau.


Task 1: Tạo các biến thể bố cục

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

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

1.1 Xem trước bố cục theo hướng ngang

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

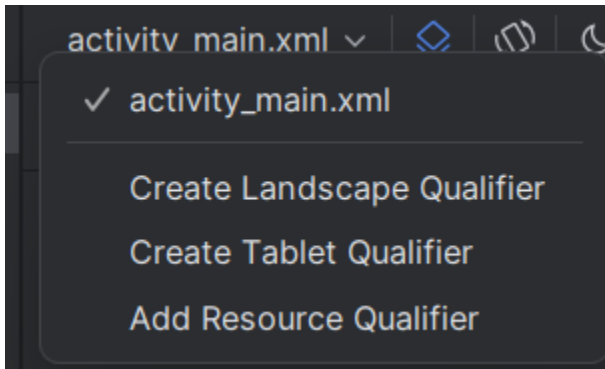
1. Mở ứng dụng Hello Toast từ bài học trước.
2. Mở tệp bố cục **activity_main.xml**. Nhấp vào tab **Design** nếu chưa chọn.
3. Nhấp vào nút **Orientation in Editor**  ở thanh công cụ trên cùng.
4. Chọn **Switch to Landscape** trong menu thả xuống. Bố cục xuất hiện theo hướng ngang như minh họa bên dưới. Để trở về hướng dọc, hãy chọn **Switch to Portrait**.

1.2 Tạo ra một biến thể bố trí cho hướng ngang

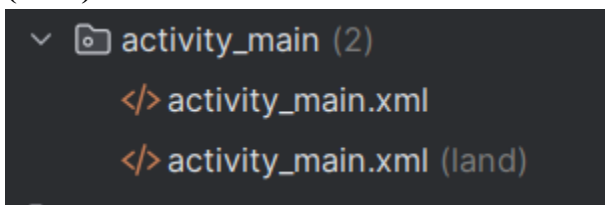
Sự khác biệt trực quan giữa hướng dọc và hướng ngang cho bố cục này là chữ số (0) trong phần tử `TextView` `show_count` quá thấp so với hướng ngang—quá gần **Count** button. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử `TextView` có thể xuất hiện quá lớn hoặc không được căn giữa vì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

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

1. Nhấp vào nút tên file **activity_main.xml** ở thanh công cụ trên cùng.
2. Chọn **Create Landscape Qualifier**. Một cửa sổ trình chỉnh sửa mới mở ra với tab `land/activity_main.xml` hiển thị bố cục cho hướng phong cảnh (ngang). Bạn có thể thay đổi bố cục này, dành riêng cho hướng ngang, mà không thay đổi hướng dọc (dọc) ban đầu.




3. Trong ngăn **Project > Android**, hãy xem bên trong thư mục **res > layout** và bạn sẽ thấy Android Studio tự động tạo biến thể cho bạn, có tên là **activity_main.xml (land)**.



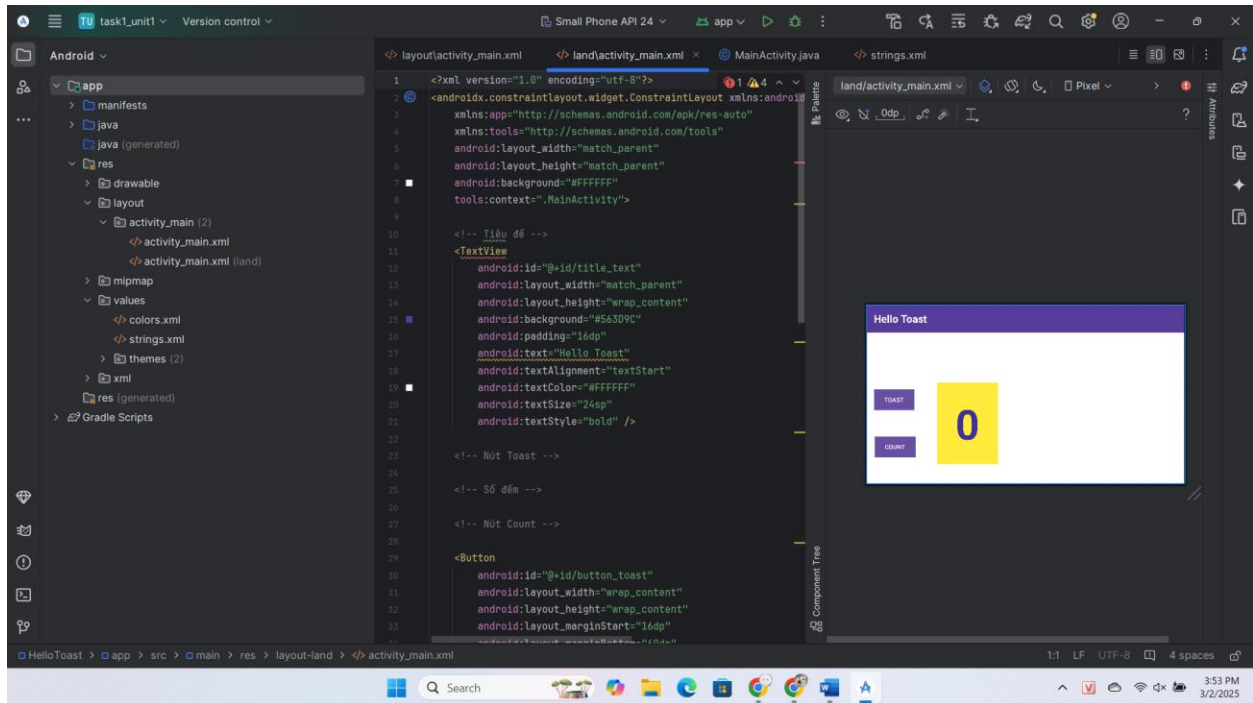
1.3 Xem trước bố cục trên các thiết bị khác nhau

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

1. Tab **land/activity_main.xml** vẫn nên được mở trong trình chỉnh sửa bố cục; nếu chưa mở, hãy nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục bố cục.
2. Nhấp vào nút **Device in Editor**  **Nexus 5** trên thanh công cụ phía trên.
3. Chọn một thiết bị khác trong menu thả xuống. Ví dụ, chọn **Nexus 4**, **Nexus 5** và sau đó **Pixel** để thấy sự khác biệt trong bản xem trước. Những khác biệt này là do kích thước văn bản cố định cho **TextView**.

1.4 Thay đổi bố cục cho hướng ngang

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



Hình trên hiển thị các nội dung sau:

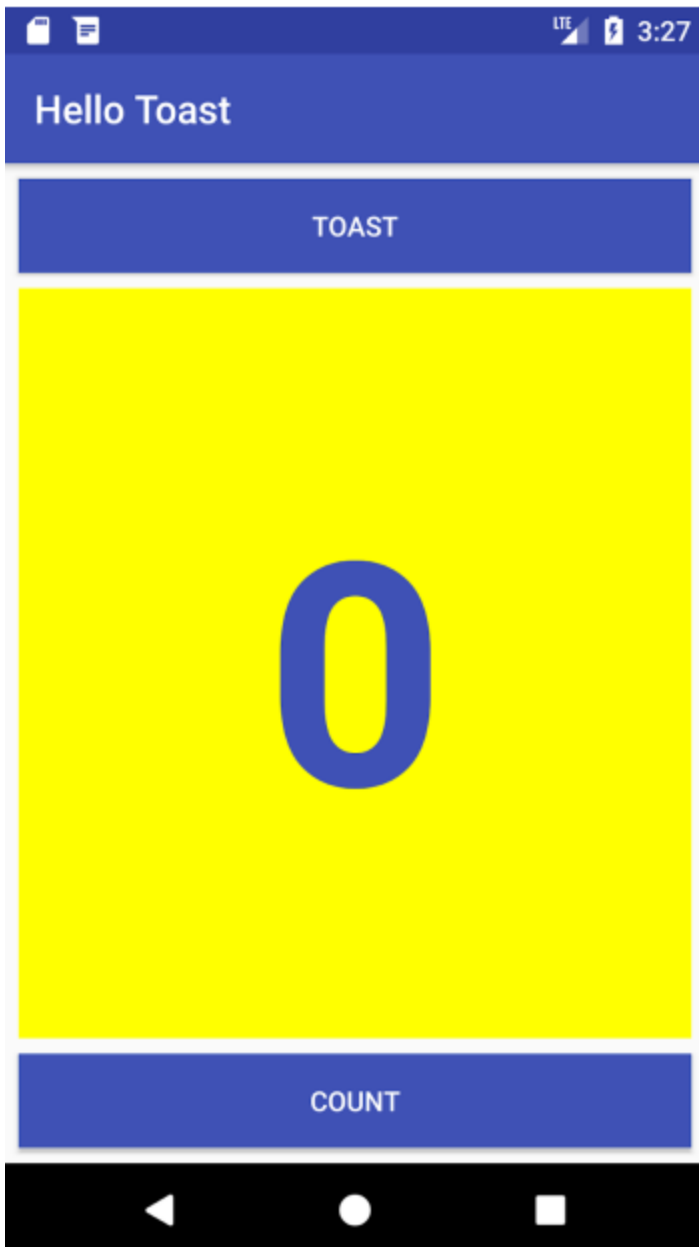
1. **Tab Preview**, được sử dụng để hiển thị khung xem trước.
2. **Khung xem trước**.
3. **Mã XML**.

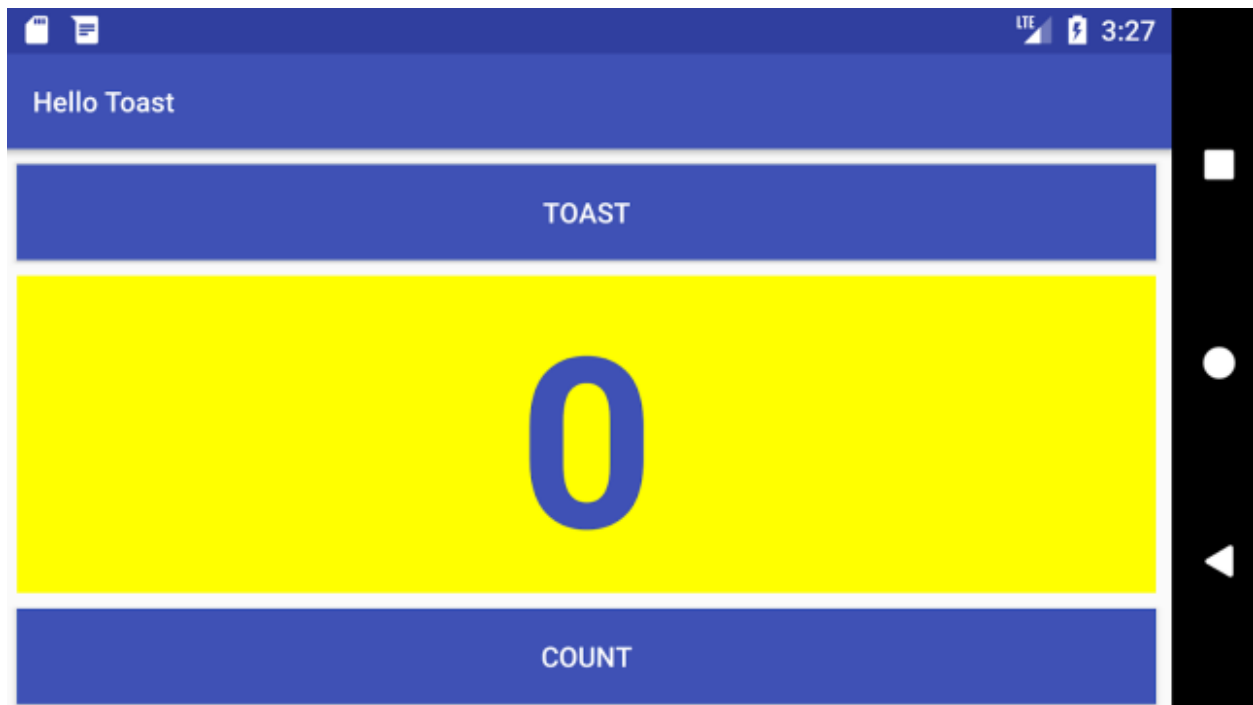
Để thay đổi bố cục, hãy làm theo các bước sau:

1. Tab **land/activity_main.xml** vẫn nên được mở trong trình chỉnh sửa bố cục; nếu chưa mở, hãy nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục bố cục.
2. Nhấp vào tab **Text** và tab **Preview** (nếu chưa được chọn).
3. Tìm phần tử **TextView** trong mã XML.
4. Thay đổi thuộc tính `android:textSize="160sp"` thành `android:textSize="120sp"`.
5. Chọn các thiết bị khác nhau trong menu thả xuống **Device in Editor** để xem cách bố cục hiển thị trên các thiết bị khác nhau ở chế độ ngang.


Trong khung chỉnh sửa, tab **land/activity_main.xml** hiển thị bố cục cho chế độ ngang. Tab **activity_main.xml** hiển thị bố cục chưa thay đổi cho chế độ dọc. Bạn có thể chuyển đổi qua lại bằng cách nhấp vào các tab.

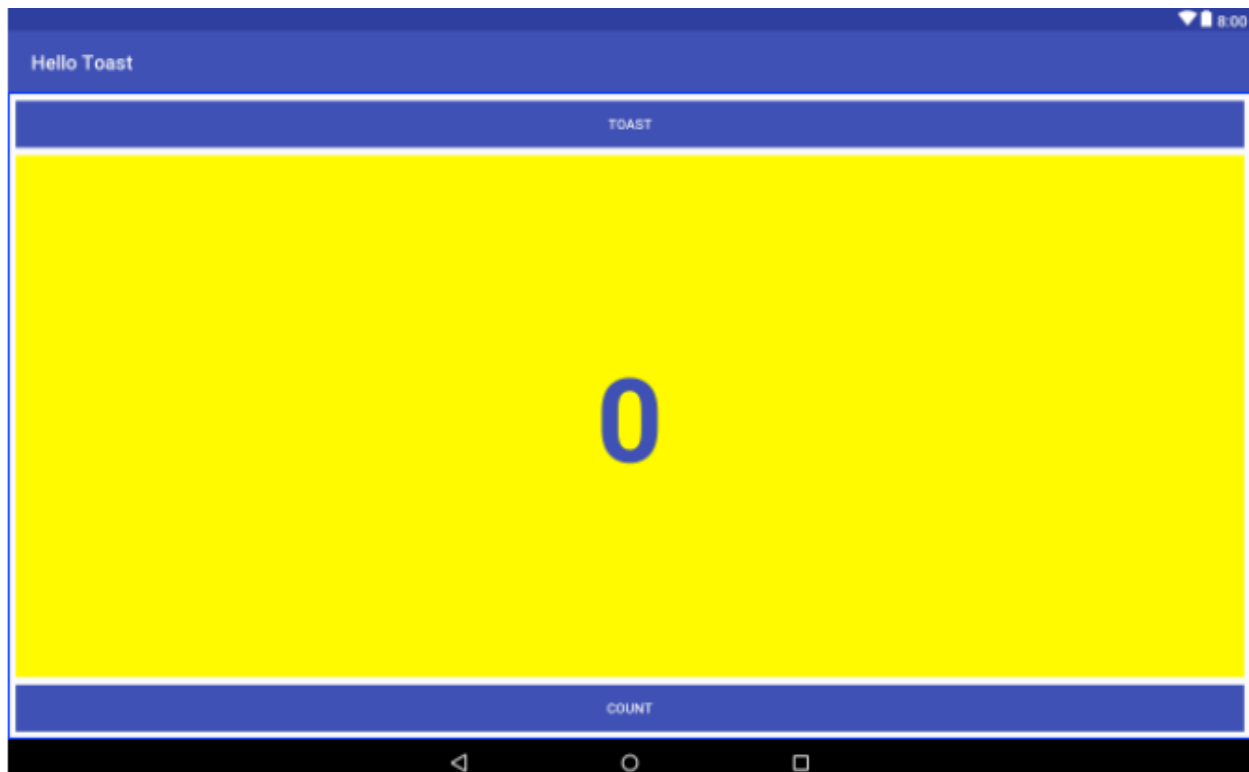
6. Chạy ứng dụng trên trình giả lập hoặc thiết bị thật, sau đó chuyển đổi giữa chế độ dọc và ngang để xem cả hai bố cục.






1.5 Tạo một biến thể bố cục cho máy tính bảng

Như bạn đã học trước đó, bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor**  **Nexus 5** trên thanh công cụ phía trên. Nếu bạn chọn một thiết bị như **Nexus 10** (một máy tính bảng) từ menu, bạn sẽ thấy rằng bố cục không phù hợp với màn hình máy tính bảng—văn bản trên mỗi **Button** quá nhỏ và cách sắp xếp các **Button** ở trên và dưới không tối ưu cho màn hình lớn của máy tính bảng.




Để khắc phục điều này mà không làm thay đổi bố cục theo chiều ngang và dọc trên điện thoại, bạn có thể tạo một biến thể bố cục hoàn toàn khác dành riêng cho máy tính bảng. Thực hiện theo các bước sau:

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị các khung 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**.

Một cửa sổ chỉnh sửa mới sẽ mở ra với tab **xlarge/activity_main.xml** hiển thị bố cục dành cho thiết bị có kích thước máy tính bảng. Trình chỉnh sửa cũng sẽ tự động chọn một thiết bị máy tính bảng, chẳng hạn như **Nexus 9** hoặc **Nexus 10**, để xem trước. Bạn có thể thay đổi bố cục này dành riêng cho máy tính bảng mà không ảnh hưởng đến các bố cục khác.

1.6 Thay đổi biến thể bố cục cho máy tính bảng

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

1. Tắt công cụ **Autoconnect** trên thanh công cụ. Đảm bảo rằng công cụ này đã được vô hiệu hóa 
2. Xóa tất cả các ràng buộc trong bố cục bằng cách nhấp vào nút **Clear All**

Constraints  trên thanh công cụ.

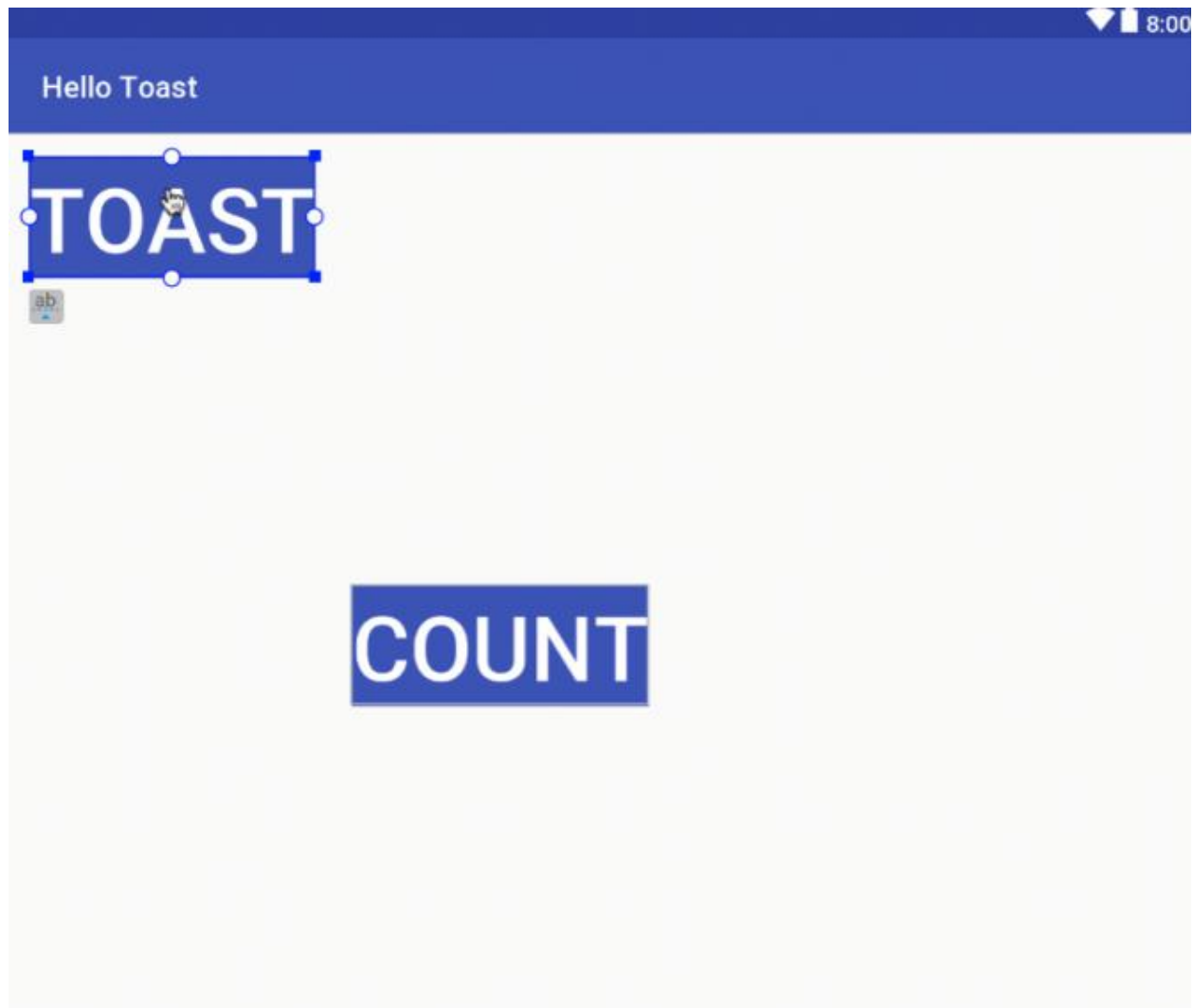
Khi các ràng buộc bị xóa, bạn có thể tự do di chuyển và thay đổi kích thước các phần tử trong bố cục.

3. Trình chỉnh sửa bố cục cung cấp các điểm điều chỉnh kích thước ở cả bốn góc của mỗi phần tử.
 - o Trong **Component Tree**, chọn **TextView** có tên là `show_count`.
 - o Để tạo không gian giúp bạn dễ dàng kéo các phần tử **Button**, hãy kéo một góc của **TextView** để thay đổi kích thước của nó, như minh họa trong hình động bên dưới.
4. Chọn nút **button_toast** trong **Component Tree**, nhấp vào tab **Attributes** để mở bảng thuộc tính, sau đó thay đổi **textSize** thành **60sp** (#1 trong hình bên dưới) và **layout_width** thành **wrap_content**.
5. Chọn nút **button_count** trong **Component Tree**, thay đổi **textSize** thành **60sp** và **layout_width** thành **wrap_content**, sau đó kéo nút này lên phía trên **TextView** vào một khoảng 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 chứa văn bản, chẳng hạn như **TextView** hoặc **Button**, với một phần tử khác chứa văn bản. **Ràng buộc đường cơ sở (Baseline Constraint)** cho phép bạn căn chỉnh các phần tử sao cho đường cơ sở của văn bản trong chúng khớp nhau.

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

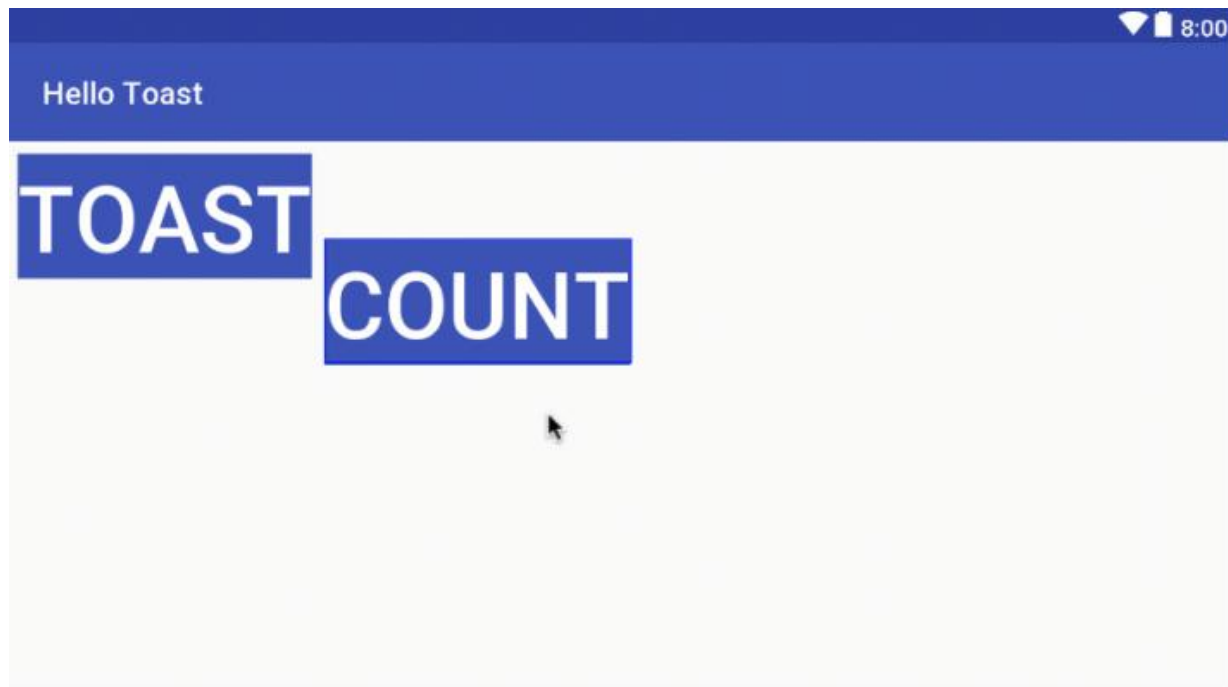


2. Sử dụng **Baseline Constraint**, bạn có thể căn chỉnh nút **button_count** sao cho đường cơ sở của văn bản trong nó khớp với đường cơ sở của văn bản trong **button_toast**. Chọn phần tử **button_count**, sau đó di chuột qua phần tử cho đến khi nút **Baseline Constraint**




xuất hiện bên dưới nó.


3. Nhấp vào nút **Baseline Constraint**. Tay cầm của đường cơ sở sẽ xuất hiện, nhấp nháy màu xanh lá như trong hình minh họa. Nhấp và kéo đường ràng buộc đường cơ sở đến đường cơ sở của phần tử **button_toast**.



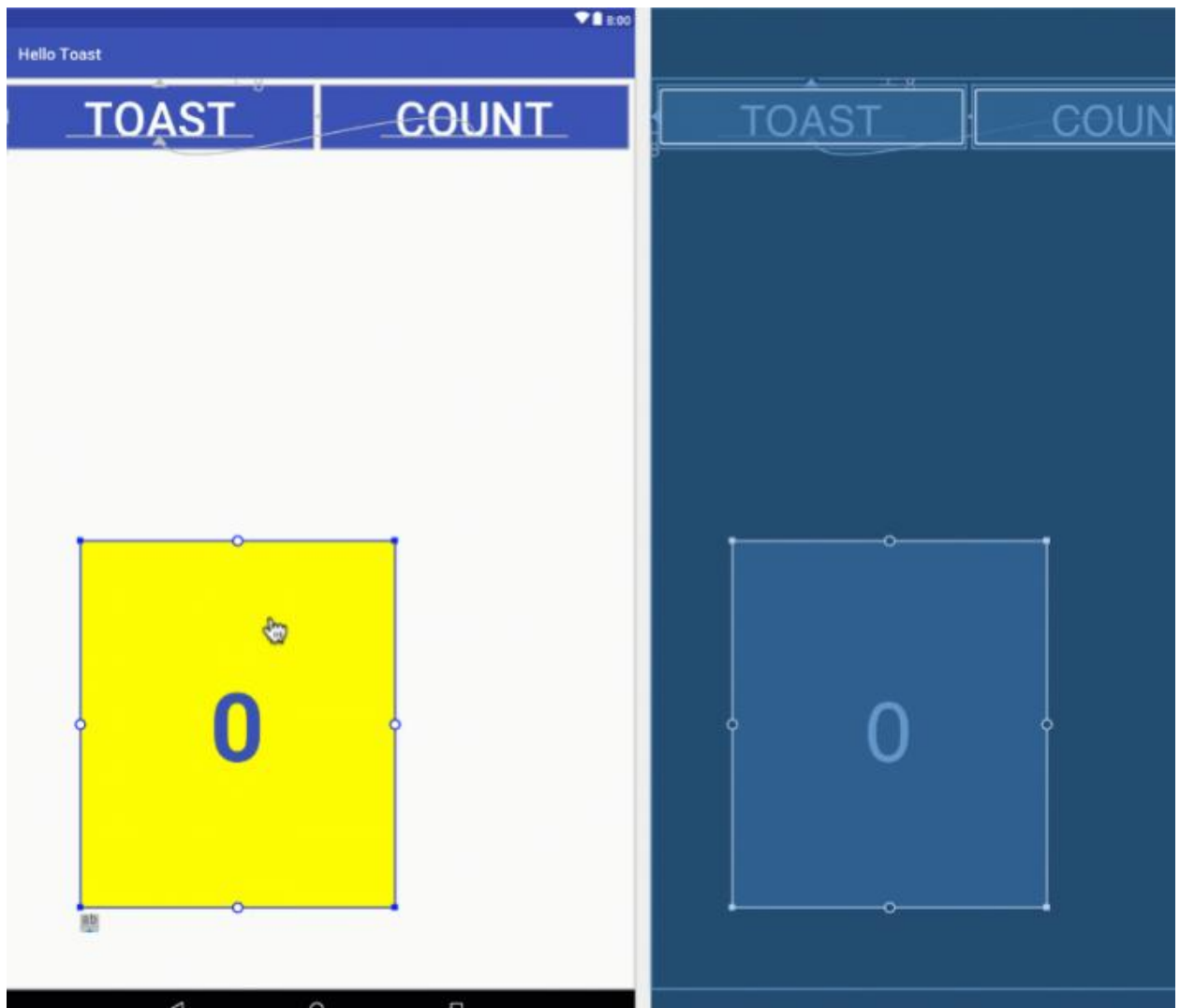
1.8 Mở rộng các nút theo chiều ngang

Nút **Pack**  trên thanh công cụ cung cấp các tùy chọn để gom nhóm 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 trong bố cục.

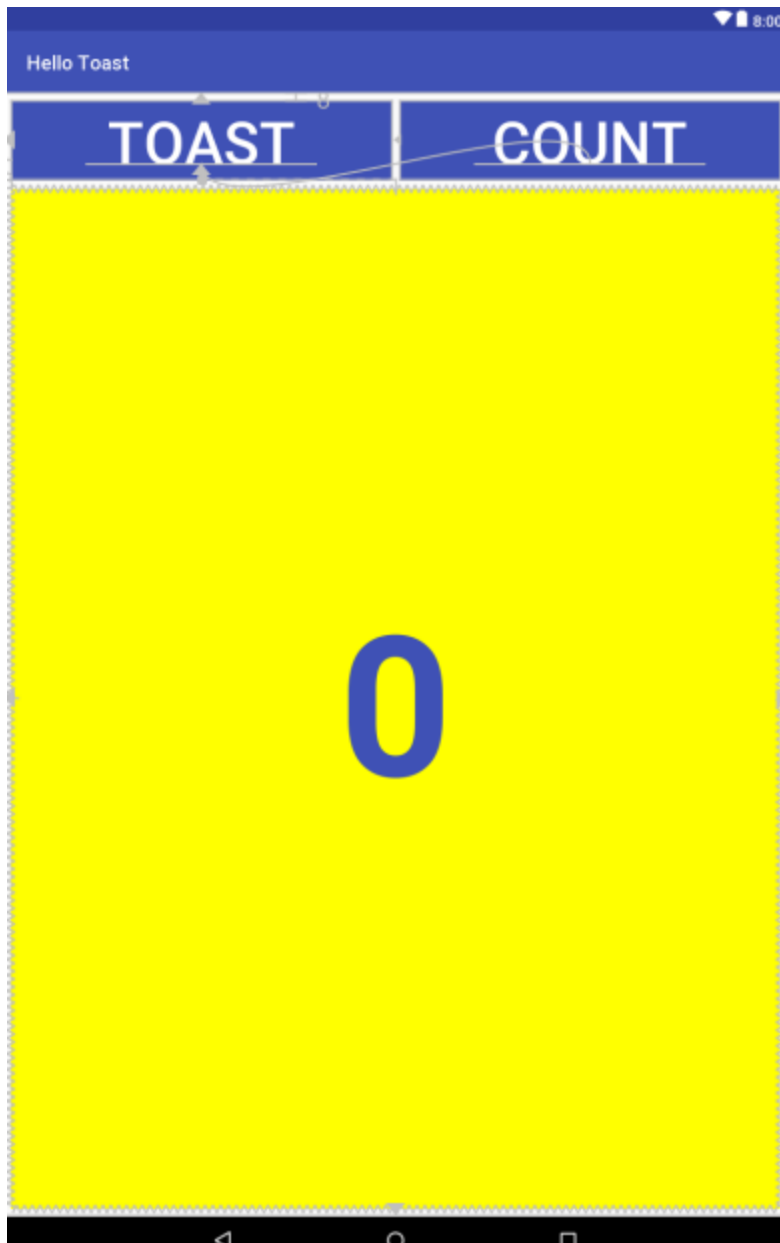
1. Chọn nút **button_count** trong **Component Tree**, sau đó nhấn **Shift** và chọn tiếp nút **button_toast** để chọn cả hai nút.

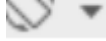
2. Nhấp vào nút **Pack**  trên thanh công cụ và chọn **Expand Horizontally**

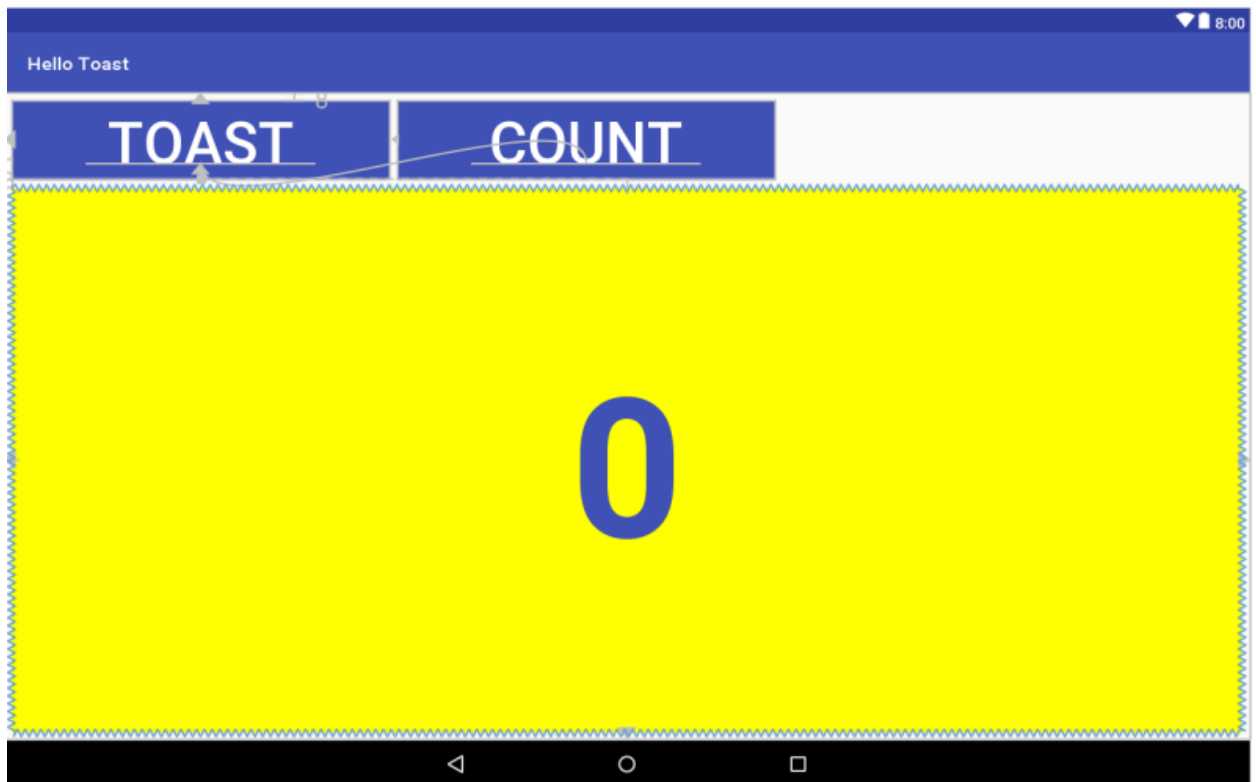
3. Để hoàn tất bố cục, hãy ràng buộc **show_count TextView** vào cạnh dưới của **button_toast**, đồng thời ràng buộc nó vào các cạnh bên và cạnh dưới của bố cục, như trong hình minh họa.



4. Bước cuối cùng là thay đổi **layout_width** và **layout_height** của **show_count TextView** thành **Match Constraints**, đồng thời đặt **textSize** thành **200sp**. Bố cục cuối cùng sẽ trông giống như trong hình minh họa.



5. Nhấp vào nút **Orientation in Editor**  trên thanh công cụ phía trên và chọn **Switch to Landscape**. Giao diện ứng dụng trên máy tính bảng sẽ xuất hiện ở chế độ ngang như hình minh họa. (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 trên các loại thiết bị khác nhau. Bạn đã tạo thành công một ứng dụng có thể chạy với giao diện phù hợp trên điện thoại và máy tính bảng với các kích thước màn hình và mật độ điểm ảnh khác nhau.

Task 2: Thay đổi bố cục thành Linear Layout

LinearLayout là một **ViewGroup** sắp xếp các phần tử con theo hàng ngang hoặc dọc. Đây là một trong những bố cục phổ biến nhất vì nó đơn giản và nhanh chóng. **LinearLayout** thường được sử dụng bên trong một nhóm giao diện khác để sắp xếp các phần tử UI theo chiều ngang hoặc dọc.

Thuộc tính bắt buộc của LinearLayout:

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

Các giá trị có thể sử dụng cho `layout_width` và `layout_height`:

- **match_parent**: Mở rộng kích thước để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao. Khi **LinearLayout** là **root view**, nó sẽ mở rộng theo kích thước màn hình (tức là phần tử cha).
- **wrap_content**: Thu nhỏ kích thước để vừa khít nội dung bên trong. Nếu không có nội dung, phần tử 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** nghĩa là 16 pixel độc lập với mật độ màn hình.

Các giá trị có thể sử dụng cho `orientation`:

- **horizontal**: Sắp xếp các phần tử từ trái sang phải.
- **vertical**: Sắp xếp các phần tử từ trên xuống dưới.

Trong nhiệm vụ này, bạn sẽ thay đổi nhóm View gốc **ConstraintLayout** của ứng dụng **HelloToast** thành **LinearLayout** để thực hành sử dụng **LinearLayout**.

2.1 Thay đổi nhóm View gốc thành **LinearLayout**

1. Mở ứng dụng **Hello Toast** từ task trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), sau đó nhấp vào tab **Text** ở dưới cùng của bảng chỉnh sửa để xem mã XML. Ở đầu mã XML, bạn sẽ thấy dòng thẻ sau:

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

3. Thay đổi thẻ `<android.support.constraint.ConstraintLayout>` thành `<LinearLayout>` để mã trông giống như sau:

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

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

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

2.2 Thay đổi thuộc tính của các phần tử cho LinearLayout

Hãy làm theo các bước sau để thay đổi thuộc tính của các phần tử UI sao cho phù hợp với **LinearLayout**:

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

```
android:layout_width="0dp"
```

```
android:layout_width="match_parent"
```

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

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

5. Tìm phần tử `button_count` và thay đổi các thuộc tính sau.

Original	Change to
<pre>android:layout_width="0dp"</pre>	<pre>android:layout_width="match_parent"</pre>

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

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

7. Tìm phần tử `show_count` (TextView) và thay đổi các thuộc tính sau.

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

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

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

9. 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 hàng ngang hoặc dọc. Bạn đã thêm thuộc tính `android:orientation="vertical"` cho **LinearLayout**, vì vậy các phần tử sẽ được xếp chồng lên nhau theo chiều dọc như trong hình minh họa trước đó.

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

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), sau đó nhấp vào tab **Text**.
3. Chọn toàn bộ phần tử `button_count` cùng với tất cả các thuộc tính của nó, từ thẻ `<Button>` cho đến thẻ đóng `</>`, sau đó chọn **Edit > Cut** (Cắt).
4. Nhấp vào ngay 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 và thụt lề cho đẹp mắt, chọn **Code > Reformat Code** để định dạng lại mã XML với khoảng cách và thụt lề phù hợp.

Hello Toast

TOAST

COUNT

0



2.4 Thêm thuộc tính weight cho phần tử TextView

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

Thuộc tính `android:gravity` xác định cách căn chỉnh nội dung bên trong **View**. Trong bài học trước, bạn đã đặt thuộc tính này cho phần tử **show_count** (TextView) để căn giữa nội dung (số 0) bên trong **TextView**.

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

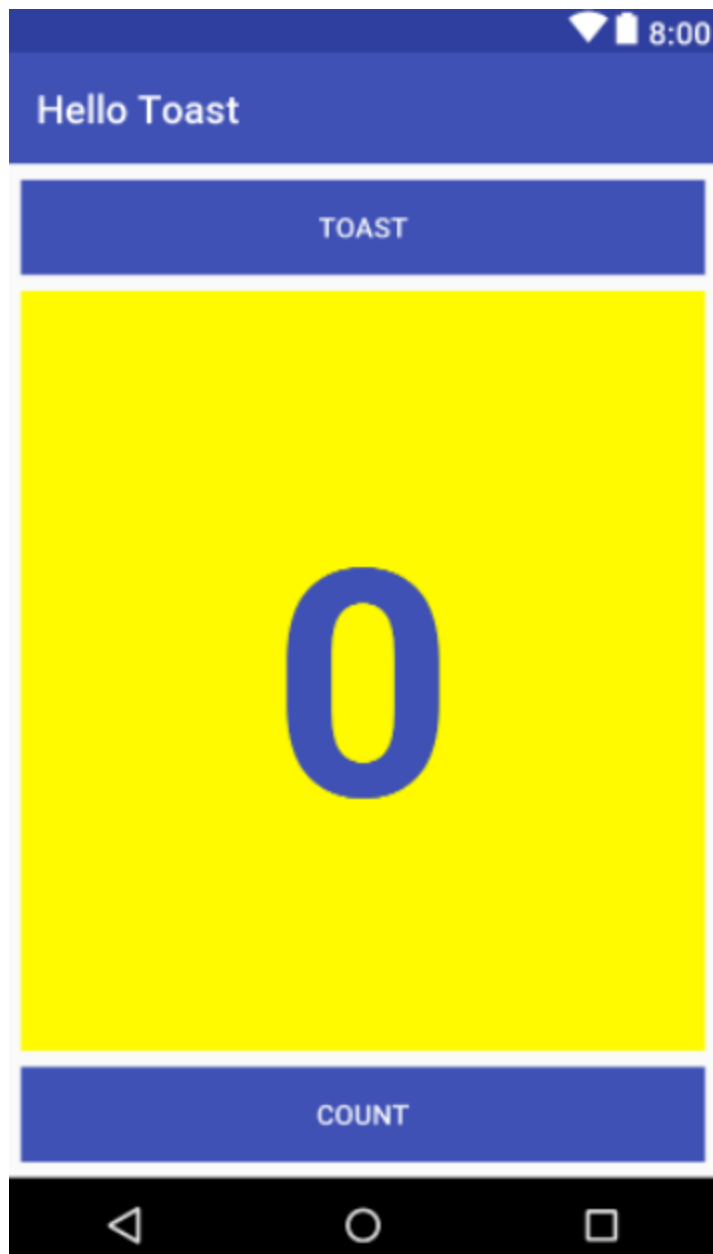
Thuộc tính `android:layout_weight` xác định mức độ phân bổ không gian thừa trong **LinearLayout** cho một **View**. Nếu chỉ có một **View** có thuộc tính này, nó sẽ nhận toàn bộ phần không gian còn lại. Nếu có nhiều **View** được gán trọng số, không gian sẽ được chia theo tỷ lệ.

Trên các thiết bị khác nhau, phần tử **show_count** (TextView) có thể hiển thị với kích thước khác nhau giữa hai nút **Toast** và **Count**. Để đảm bảo **TextView** mở rộng và lấp đầy không gian trống trên mọi thiết bị, bạn cần chỉ định thuộc tính `android:gravity`.

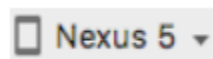
Hãy thực hiện các bước sau:

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước.
2. Mở tệp **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"
```



Phần tử **show_count** (TextView) sẽ chiếm toàn bộ không gian giữa hai nút **Toast** và **Count**. 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 của bảng xem trước, sau đó chọn một thiết bị khác. Dù bạn chọn thiết bị nào để xem trước, phần tử **show_count** (TextView) vẫn sẽ lấp đầy toàn bộ không gian giữa hai nút.

3.1 Thay đổi LinearLayout thành RelativeLayout

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

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

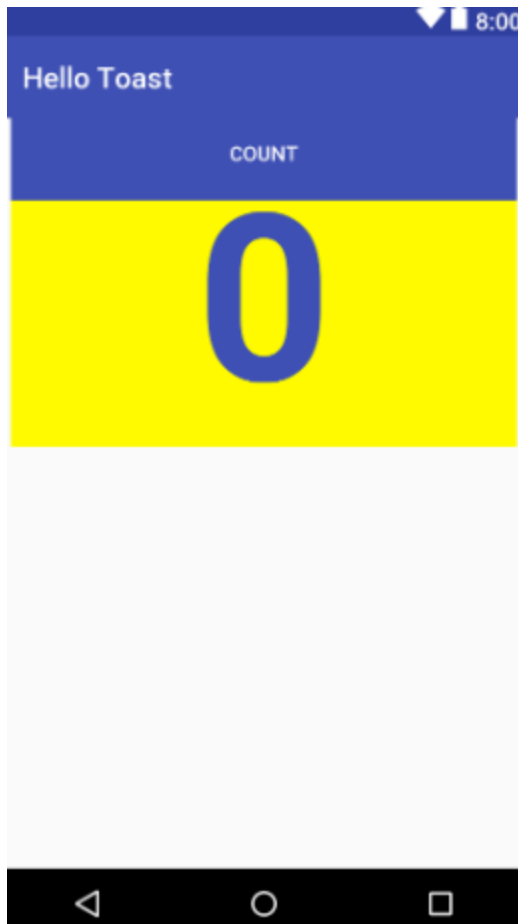
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Cuộn xuống để đảm bảo thẻ kết thúc `</LinearLayout>` cũng đã thay đổi thành `</RelativeLayout>`. Nếu chưa, hãy thay đổi 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** ở bên cạnh trình chỉnh sửa (nếu chưa được chọn) để xem bản xem trước bố cục, hiện trông giống như hình dưới đây.



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

- **Nút Count** (button_count) đang chồng lên **Nút Toast** (button_toast), khiến bạn không thể nhìn thấy button_toast.
- Phần trên của **TextView** (show_count) đang chồng lên các nút Button.

2. Thêm thuộc tính android:layout_below vào **button_count** để đặt Button này ngay bên dưới **show_count TextView**. Đây là một trong những thuộc tính dùng để định vị View trong **RelativeLayout**, cho phép bạn đặt View dựa trên vị trí của View khác.

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

3. Thêm thuộc tính `android:layout_centerHorizontal` vào cùng Button để căn giữa Button theo chiều ngang trong View cha, trong trường hợp này là nhóm View **RelativeLayout**

```
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"
```


5. Xóa thuộc tính `android:layout_weight="1"` khỏi **show_count TextView**, vì thuộc tính này không phù hợp với **RelativeLayout**. Bản xem trước bố cục giờ sẽ trông giống như hình dưới đây.






Tóm tắt

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

- Để xem trước bố cục ứng dụng với **hướng ngang** trong trình chỉnh sửa bố cục, nhấp vào nút **Orientation in Editor**  trên thanh công cụ trên cùng 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 ra với tab **land/activity_main.xml** hiển thị bố cục cho chế độ ngang.

- Để xem trước bố cục trên **các thiết bị khác nhau** mà không cần chạy ứng dụng trên thiết bị hoặc trình giả lập, nhấp vào nút **Device in Editor**  **Nexus 5** trên thanh công cụ trên cùng và chọn một thiết bị.
- Để tạo **biến thể bố cục** khác 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 ra. Cửa sổ mới sẽ mở ra với tab **xlarge/activity_main.xml**, hiển thị bố cục cho thiết bị có kích thước máy tính bảng.

Sử dụng ConstraintLayout:

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

Sử dụng LinearLayout:

- **LinearLayout** là một **ViewGroup** sắp xếp các phần tử con của nó theo hàng ngang hoặc dọc.
- **LinearLayout** bắt buộc phải có các thuộc tính **layout_width**, **layout_height**, và **orientation**.
- **match_parent** cho **layout_width** hoặc **layout_height**: **Mở rộng View** để lấp đầy View cha theo chiều rộng hoặc chiều cao. Khi **LinearLayout** là View gốc, nó sẽ mở rộng theo kích thước màn hình (View cha).
- **wrap_content** cho **layout_width** hoặc **layout_height**: **Thu nhỏ kích thước** để View vừa đủ chứa nội dung. Nếu không có nội dung, View sẽ trở nên vô hình.
- **Kích thước cố định (sử dụng dp - density-independent pixels)** cho **layout_width** hoặc **layout_height**: Xác định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ: 16dp nghĩa là 16 pixel độc lập với mật độ.

- **orientation** trong **LinearLayout** có thể là **horizontal**: Sắp xếp các phần tử từ **trái sang phải**, **vertical**: Sắp xếp các phần tử từ **trên xuống dưới**.

Xác định thuộc tính gravity và weight giúp bạn kiểm soát tốt hơn việc sắp xếp các View và nội dung trong LinearLayout:

- Thuộc tính android:gravity xác định **căn chỉnh nội dung bên trong View**.
- Thuộc tính android:layout_weight xác định **lượng không gian bổ sung** trong **LinearLayout** được phân bổ cho **View**. Nếu chỉ có một View có thuộc tính này, nó sẽ nhận toàn bộ không gian trống. Nếu nhiều View có thuộc tính này, không gian sẽ được phân chia theo tỷ lệ. Ví dụ: nếu hai **Button** có trọng số **1** và một **TextView** có trọng số **2**, tổng cộng là **4**, thì:
 - Mỗi **Button** sẽ nhận $\frac{1}{4}$ không gian.
 - **TextView** sẽ nhận $\frac{1}{2}$ không gian.

Sử dụng RelativeLayout:

- **RelativeLayout** là một **ViewGroup** trong đó mỗi View được định vị và căn chỉnh dựa trên các **View khác** trong nhóm.
- Dùng android:layout_alignParentTop để **căn View với đỉnh của View cha**.
- Dùng android:layout_alignParentLeft để **căn View với mép trái của View cha**.
- Dùng android:layout_alignParentStart để làm cho **cạnh bắt đầu của View khớp với cạnh bắt đầu của View cha**. Thuộc tính này hữu ích nếu bạn muốn ứng dụng **tương thích với các thiết bị sử dụng ngôn ngữ hoặc cài đặt vùng khác nhau**. **Cạnh bắt đầu** là **mép trái** của màn hình nếu ngôn ngữ viết từ **trái sang phải**. **Cạnh bắt đầu** là **mép phải** của màn hình nếu ngôn ngữ viết từ **phải sang trái**.

Bài tập về nhà

Thay đổi một ứng dụng

Mở ứng dụng HelloToast.

1. Đổi tên dự án thành **HelloConstraint** và refactor dự án thành **HelloConstraint**. (Để biết hướng dẫn cách sao chép và refactor dự án, xem phần Phụ lục: Tiện ích.)
2. Chỉnh sửa tệp **activity_main.xml** để căn chỉnh các nút **Toast** và **Count** theo hàng dọc bên trái của **TextView** hiển thị số "0". Tham khảo hình minh họa để bố trí giao diện.
3. Thêm một nút thứ ba có tên **Zero**, đặt giữa hai nút **Toast** và **Count**.
4. Căn chỉnh các nút theo chiều dọc, phân bố đều từ trên xuống dưới của **TextView** hiển thị số.
5. Đặt nền của nút **Zero** thành màu xám ban đầu.
6. Đảm bảo rằng nút **Zero** cũng xuất hiện trong chế độ xoay ngang (**activity_main.xml (land)**) và trong giao diện dành cho màn hình máy tính bảng (**activity_main (xlarge)**).
7. Viết chức năng để khi nhấn nút **Zero**, giá trị của **TextView** hiển thị số sẽ trở về **0**.
8. Cập nhật sự kiện nhấn cho nút **Count**, sao cho khi nhấn, nền của nó thay đổi màu tùy vào số hiện tại là **chẵn** hay **lẻ**.
Gợi ý: Không sử dụng **findViewById** để lấy nút **Count**. Bạn có thể sử dụng phương pháp nào khác?
Bạn có thể sử dụng các hằng số trong lớp **Color** để chọn hai màu nền khác nhau.
9. Cập nhật trình xử lý sự kiện nhấn cho nút **Count**, sao cho khi nhấn, màu nền của nút **Zero** thay đổi thành một màu khác (không phải màu xám) để hiển thị rằng nút **Zero** đã hoạt động.
Gợi ý: Trong trường hợp này, bạn có thể sử dụng **findViewById**.
10. Cập nhật trình xử lý sự kiện nhấn cho nút **Zero**, sao cho khi số hiển thị trở về **0**, màu nền của nó cũng được đặt lại thành **xám**.



1.3) Trình chỉnh sửa bố cục

1.4) Văn bản và các chế độ cuộn

1.5) Tài nguyên có sẵn

Bài 2) Activities

2.1) Activity và Intent

2.2) Vòng đời của Activity và trạng thái

2.3) Intent ngầm định

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

3.1) Trình gỡ lỗi

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

- 1.1) Hình ảnh có thể chọn**
- 1.2) Các điều khiển nhập liệu**
- 1.3) Menu và bộ chọn**
- 1.4) Điều hướng người dùng**
- 1.5) RecyclerView**

Bài 2) Trải nghiệm người dùng thú vị

- 2.1) Hình vẽ, định kiểu và chủ đề**
- 2.2) Thẻ và màu sắc**
- 2.3) Bố cục thích ứng**

Bài 3) Kiểm thử giao diện người dùng

- 3.1) Espresso cho việc kiểm tra UI**

CHƯƠNG 3. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

- 1.1) AsyncTask**
- 1.2) AsyncTask và AsyncTaskLoader**
- 1.3) Broadcast receivers**

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

- 2.1) Thông báo**
- 2.2) Trình quản lý cảnh báo**
- 2.3) JobScheduler**

CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

1.1) Shared preferences

1.2) Cài đặt ứng dụng

Bài 2) Lưu trữ dữ liệu với Room

2.1) Room, LiveData và ViewModel

2.2) Room, LiveData và ViewModel