

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



GIÁO TRÌNH

THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1. Làm quen	4
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	21
1.3) Trình chỉnh sửa bố cục	50
1.4) Văn bản và các chế độ cuộn	73
1.5) Tài nguyên có sẵn	80
Bài 2) Activities.....	80
2.1) Activity và Intent.....	80
2.2) Vòng đời của Activity và trạng thái	80
2.3) Intent ngầm định	80
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	80
3.1) Trình gỡ lỗi.....	80
3.2) Kiểm thử đơn vị	80
3.3) Thư viện hỗ trợ	80
CHƯƠNG 2. Trải nghiệm người dùng.....	81
Bài 1) Tương tác người dùng	81
1.1) Hình ảnh có thể chọn.....	81
1.2) Các điều khiển nhập liệu	81
1.3) Menu và bộ chọn	81
1.4) Điều hướng người dùng	81
1.5) RecyclerView	81
Bài 2) Trải nghiệm người dùng thú vị.....	81
2.1) Hình vẽ, định kiểu và chủ đề	81
2.2) Thẻ và màu sắc	81
2.3) Bố cục thích ứng.....	81
Bài 3) Kiểm thử giao diện người dùng	81

3.1) Espresso cho việc kiểm tra UI	81
CHƯƠNG 3. Làm việc trong nền	81
Bài 1) Các tác vụ nền.....	81
1.1) AsyncTask.....	81
1.2) AsyncTask và AsyncTaskLoader	81
1.3) Broadcast receivers	81
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	81
2.1) Thông báo.....	81
2.2) Trình quản lý cảnh báo	81
2.3) JobScheduler	81
CHƯƠNG 4. Lưu dữ liệu người dùng	82
Bài 1) Tùy chọn và cài đặt.....	82
1.1) Shared preferences	82
1.2) Cài đặt ứng dụng.....	82
Bài 2) Lưu trữ dữ liệu với Room	82
2.1) Room, LiveData và ViewModel.....	82
2.2) Room, LiveData và ViewModel.....	82

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

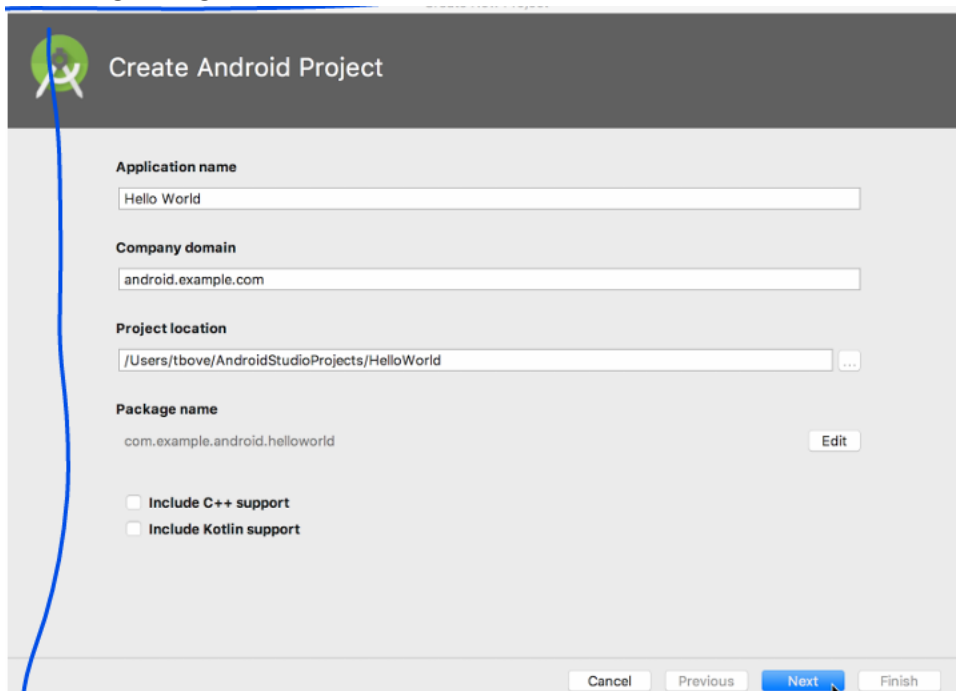
Giới thiệu

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 một dự án mới cho ứng dụng Hello World từ một mẫu. Ứng dụng đơn giản này hiển thị chuỗi “Hello World” trên màn hình của thiết bị ảo hoặc vật lý.

Ứng dụng hoàn thành sẽ trông như thế này:



Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp (IDE) hoàn chỉnh bao gồm trình chỉnh sửa mã nâng cao và một bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ để phát triển, gỡ lỗi thử nghiệm và hiệu suất giúp phát triển ứng dụng nhanh hơn và dễ dàng hơn. Bạn có thể kiểm tra ứng dụng của mình bằng nhiều trình mô phỏng được cấu hình sẵn hoặc trên thiết bị di động của riêng mình, tạo ứng dụng chính thức trên cửa hàng Google Play.

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

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

Để thiết lập và chạy Android Studio, trước tiên hãy kiểm tra **system requirements** để đảm bảo hệ thống của bạn đáp ứng các yêu cầu đó. Việc cài đặt tương tự cho tất cả các nền tảng. Bất kỳ sự khác biệt được lưu ý bên dưới

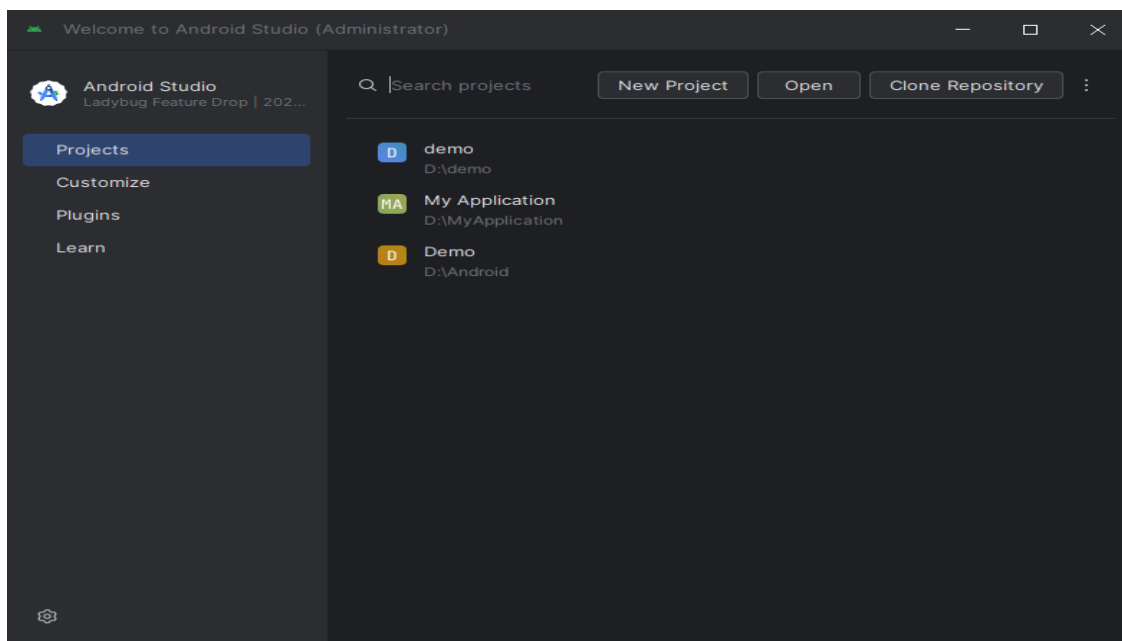
1. Truy cập đến **Android developers site** và làm theo hướng dẫn để tải xuống và cài đặt **Android Studio**
2. Chấp nhận cấu hình mặc định cho tất cả các bước và đảm bảo rằng tất cả các thành phần được chọn để cài đặt
3. Sau khi hoàn tất cài đặt, trình hướng dẫn cài đặt sẽ tải xuống và cài đặt một số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

Xử lý sự cố: Nếu bạn gặp sự cố khi cài đặt, hãy kiểm tra **Android Studio release notes** hoặc nhờ người hướng dẫn trợ giúp

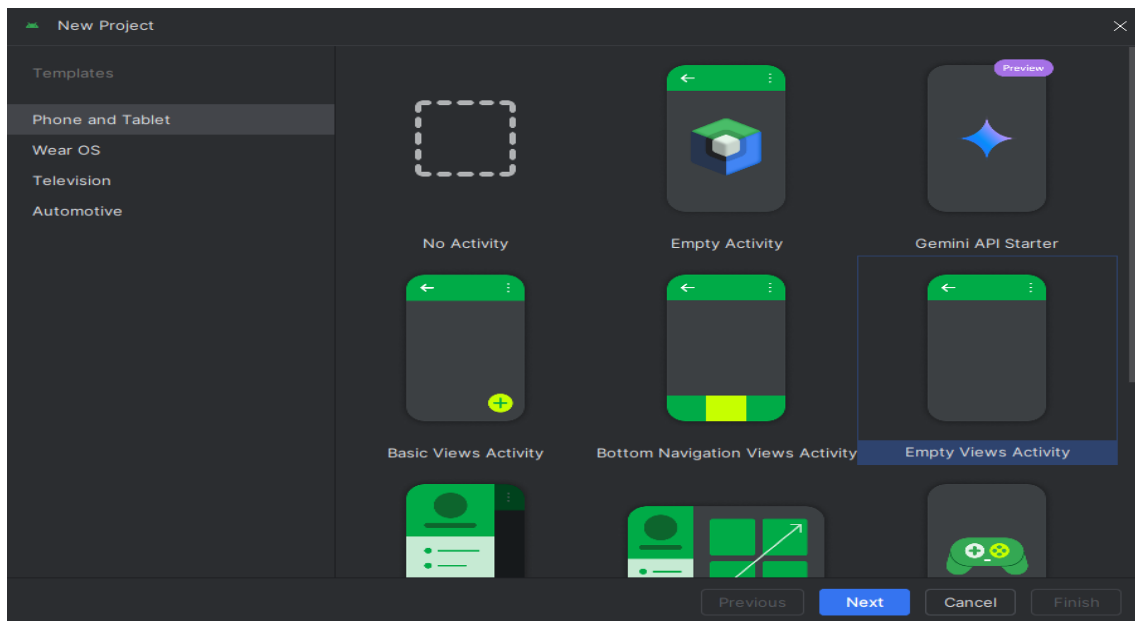
Nhiệm vụ 2: Tạo ứng dụng Hello World

2.1. Tạo dự án ứng dụng

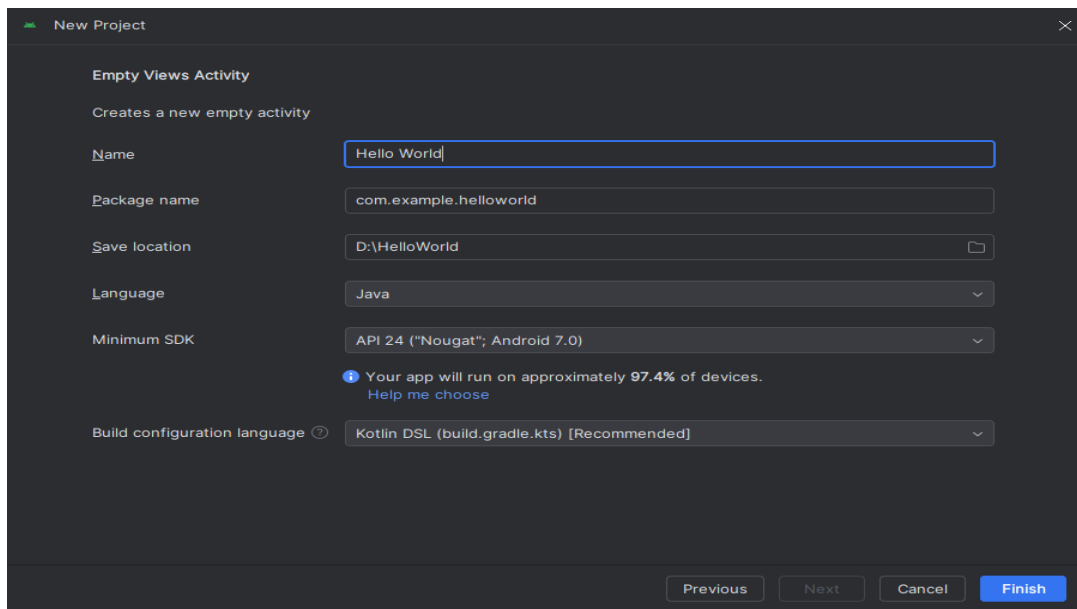
1. Mở Android Studio nếu nó chưa được mở
2. Trong cửa sổ chính **Welcome to Android Studio**, nhấp vào **New Project**



- Trong cửa sổ **New Project**, chọn Activity cho dự án. Activity là một việc người dùng có thể làm. Nó là một thành phần quan trọng của bất kỳ ứng dụng Android nào. Activity thường có bố cục được liên kết với nó xác định cách các thành phần UI xuất hiện trên màn hình. Android Studio cung cấp các mẫu Activity để giúp bạn bắt đầu. Đối với dự án Hello World thì chọn **Empty Views Activity** và sau đó ấn **Next**



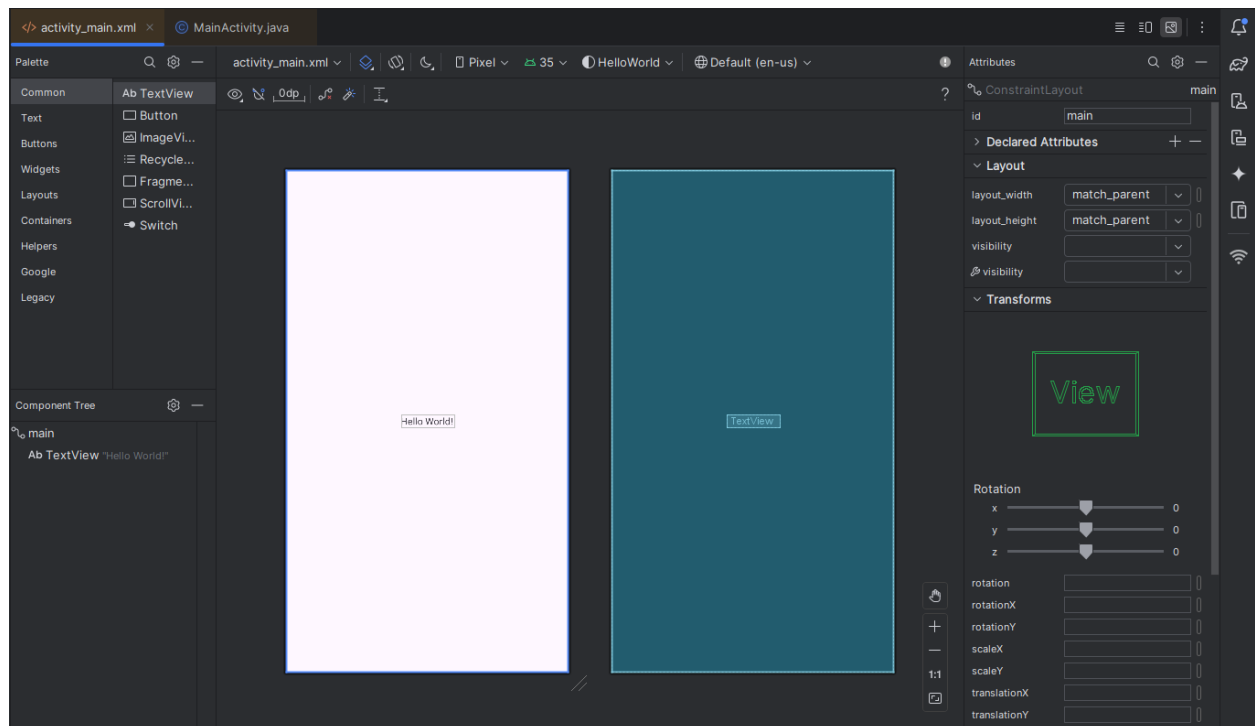
- Trong cửa sổ **New Project**, nhập **Hello World** cho **Name**.
- Chấp nhận tên miền mặc định của công ty **com.example.helloworld** hoặc tạo một tên miền công ty độc quyền
Nếu bạn không có kế hoạch xuất bản ứng dụng của mình, bạn có thể để mặc định. Lưu ý rằng việc thay đổi **Package name** của bạn sau này sẽ tốn thêm công sức
- Xác minh vị trí dự án mặc định **Save location** là nơi bạn muốn lưu trữ ứng dụng Hello World và các dự án Android Studio khác, hoặc thay đổi vị trí đó thành thư mục ưa thích của bạn
- Chọn **Language** là **java**
- Trong **Minimum SDK** được để mặc định là **API 24 ("Nougat", Android 7.0)**, thiết lập này làm cho ứng dụng Hello World của bạn tương thích với 97.4% thiết bị Android đang hoạt động trên Google Play Store. Nếu bạn không muốn hãy bật danh sách Minimum SDK lên để chọn
- Trong **Build configuration language** chọn **Kotlin DSL (build.gradle.kts)** và cuối cùng 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 sẽ xuất hiện. Làm theo các bước sau:

1. Nhấp vào tab **activity_main.xml** để xem trình chỉnh sửa bố cục
2. Nhấp vào tab **Design**, nếu chưa chọn, biểu tượng đồ họa của bố cục hiển thị như hình dưới



3. Nhấp vào tab **MainActivity.java** để xem trình sửa code như hình dưới

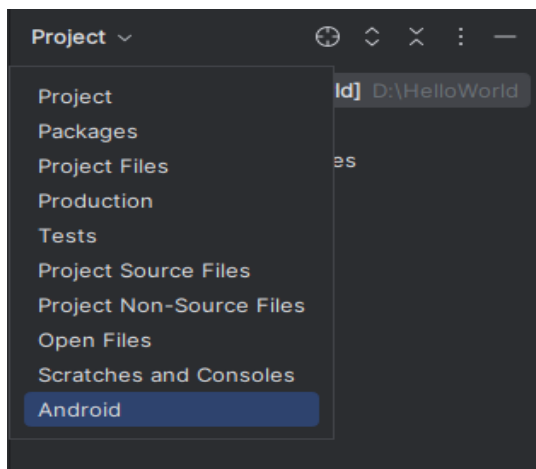


```
1 package com.example.helloworld;
2
3 > import --
10
11 public class MainActivity extends AppCompatActivity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         EdgeToEdge.enable(this);
17         setContentView(R.layout.activity_main);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

2.2. Khám phá dự án > Bảng điều khiển Android

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

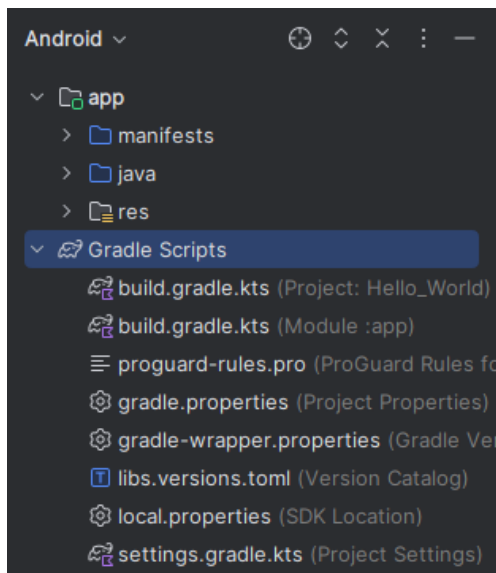
1. Nếu chưa chọn, hãy nhấp vào tab **Project** trong cột tab dọc ở phía bên trái của cửa sổ Android Studio. Ngăn Project xuất hiện
2. Để xem dự án trong hệ thống phân cấp dự án Android tiêu chuẩn, hãy chọn Android từ menu bật lên ở trong ngăn Project, như hình bên dưới



2.3. Khám phá thư mục Gradle Scripts

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

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



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

1. Nếu thư mục **Gradle Scripts** chưa được mở rộng, hãy nhấp vào hình tam giác để mở rộng. Thư mục này chứa tất cả các tệp cần thiết cho hệ thống xây dựng

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

Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô-đun tạo nên dự án của bạn. Mỗi dự án Android Studio đều chứa một tệp bản dựng Gradle cấp cao nhất. Hầu hết thời gian, bạn sẽ không cần thực hiện bất kỳ thay đổi nào đối với tệp này, nhưng nó vẫn hữu ích để hiểu nội dung của nó

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

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

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

Ngoài tệp build.gradle.kts cấp độ dự án, mỗi mô-đun cũng có một build.gradle.kts riêng, cho phép bạn cấu hình các cài đặt build cho từng mô-

đơn cụ thể (trong ứng dụng HelloWorld, chỉ có một mô-đun duy nhất). Việc cấu hình các cài đặt build này giúp bạn tùy chỉnh các tùy chọn đóng gói, chẳng hạn như bổ sung kiểu build và biến thể sản phẩm. Ngoài ra bạn cũng có thể ghi đè các thiết lập trong tệp `AndroidManifest.xml` hoặc tệp `build.gradle` ở cấp dự án

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

Sau đây là tệp **`build.gradle.kts(Module:app)`**:

```

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("name: "proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}
dependencies {
    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
}

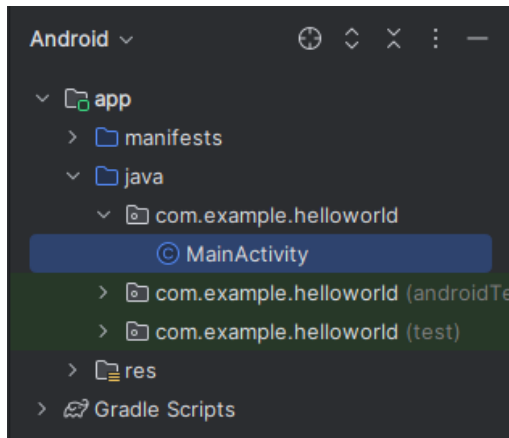
```

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

2.4. Khám phá ứng dụng và thư mục res

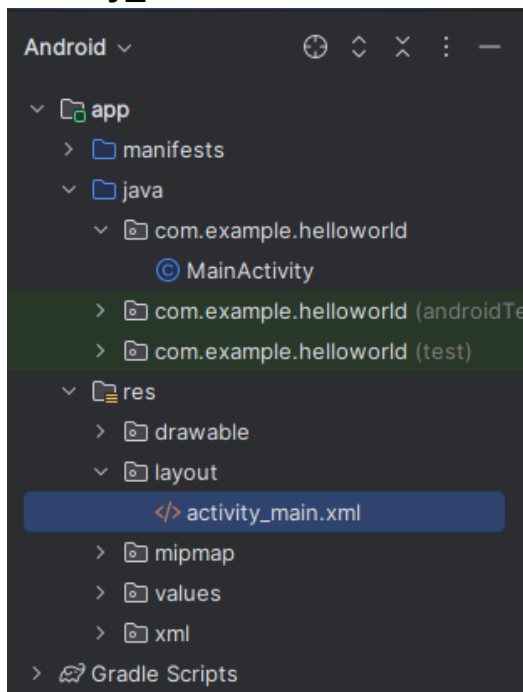
Tất cả mã nguồn và tài nguyên cho ứng dụng đều nằm trong thư mục app và res

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 nó trong trình chỉnh sửa mã nguồn



Thư mục **java** bao gồm các tệp lớp java trong ba thư mục con, như hình trên. Thư mục **com.example.android.helloworld** (hoặc tên miền bạn chỉ định) chứa tất cả các tệp cho gói ứng dụng. Hai thư mục còn lại được sử dụng để kiểm tra và được đề cập trong một bài học khác. Đối với ứng dụng Hello World, chỉ có một gói duy nhất, trong đó chứa MainActivity.java. Tên của Activity đầu tiên (màn hình đầu tiên mà người dùng nhìn thấy), đồng thời khởi tạo các tài nguyên dùng chung cho toàn bộ ứng dụng, theo thông lệ thường được đặt là **MainActivity** (phần mở rộng tệp được ẩn trong phần **Project > Android**)

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

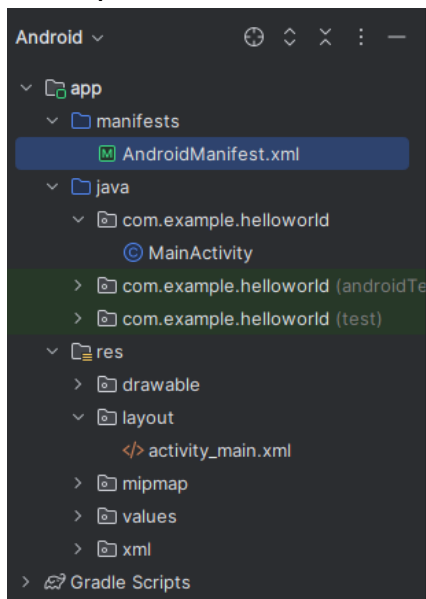


Thư mục **res** chứa các tài nguyên, chẳng hạn như bố cục, chuỗi kí tự và hình ảnh. Một Activity thường được liên kết với bố cục giao diện người dùng, được định nghĩa trong một tệp XML. Tệp này thường được đặt tên theo Activity tương ứng

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, hệ thống phải có thông tin này trước khi có thể chạy bất kỳ mã nguồn 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 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 chỉnh sửa tệp này để thêm tính năng và quyền truy cập tính năng. Để tìm hiểu tổng quan, hãy xem **App Manifest Overview**

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


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

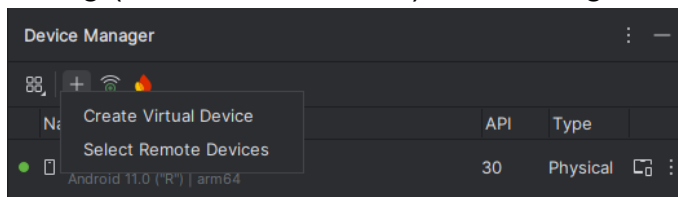
Khi sử dụng Trình quản lý AVD, bạn xác định các đặc điểm phần cứng của thiết bị, cấp độ API, bộ nhớ, giao diện và các thuộc tính khác, lưu lại dưới dạng một thiết bị ảo. Với thiết bị ảo, bạn có thể kiểm thử ứng dụng trên nhiều cấu hình thiết bị khác nhau (chẳng hạn như máy tính bảng và điện thoại) với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

3.1. Tạo thiết bị ảo Android (AVD)

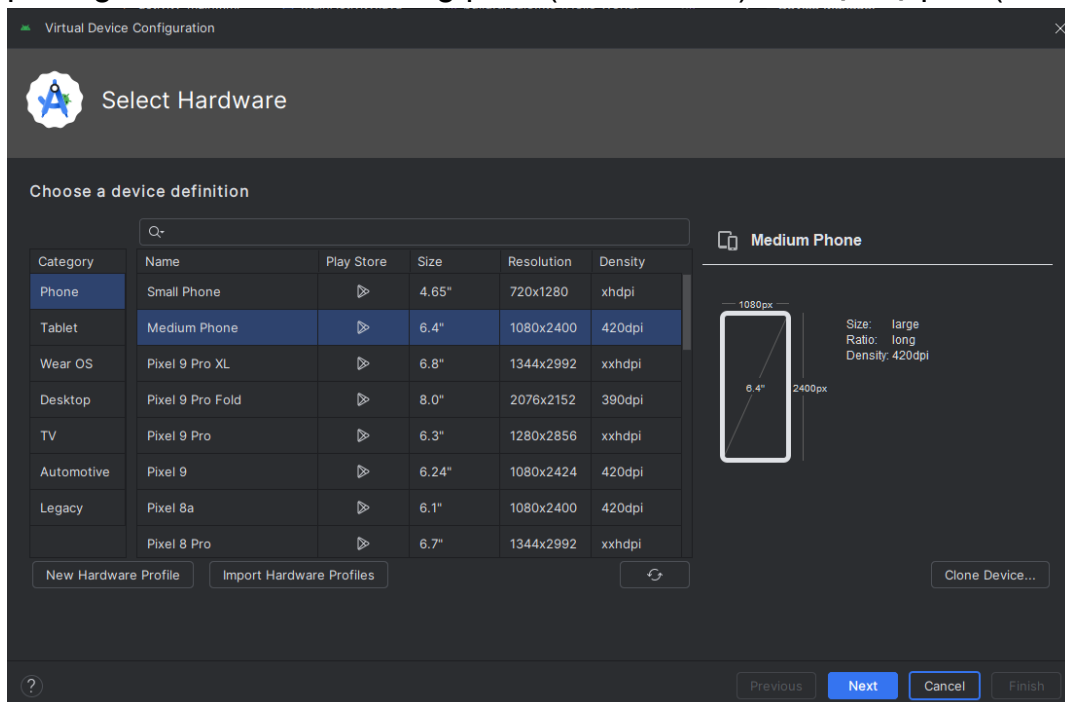
Để chạy trình mô phỏng trên máy tính, bạn phải tạo một cấu hình mô tả thiết bị ảo

1. Trong Android Studio, chọn **Tools > Android > AVD Manager**, hoặc nhấp

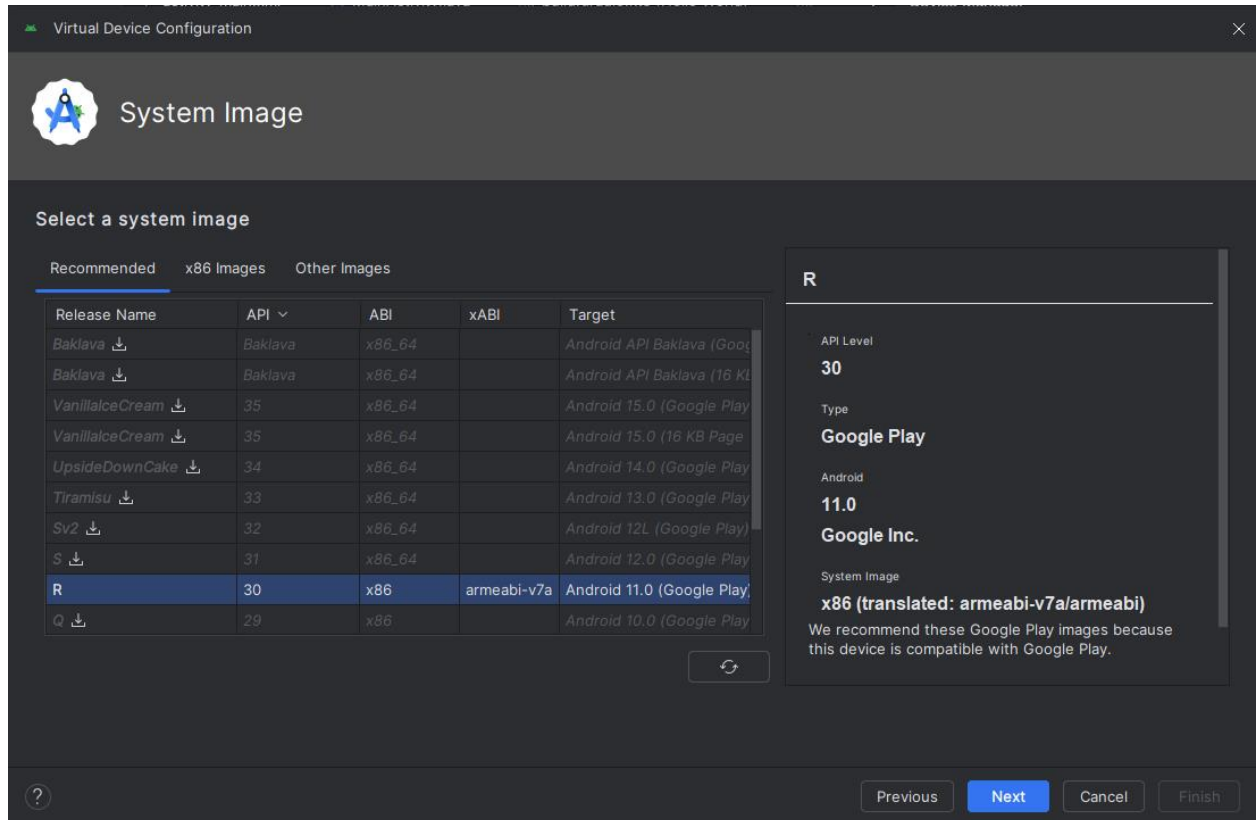
vào biểu tượng Trình quản lý AVD  trên thanh công cụ. Màn hình Your Virtual Devices xuất hiện. Nếu bạn đã tạo thiết bị ảo, màn hình sẽ hiển thị chúng (như hình bên dưới), nếu không bạn sẽ thấy một danh sách trống



2. Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** 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 cho kích thước màn hình chéo (**Size**), độ phân giải màn hình tính bằng pixel (**Resolution**) và mật độ pixel (**Density**)



3. Chọn một thiết bị như **Medium Phone** rồi nhấp vào Next. Màn hình **System Image** sẽ xuất hiện
4. Nhấp vào tab **Recommended** nếu chưa được chọn và chọn phiên bản hệ thống Android để chạy thiết bị ảo (**R**)



Có nhiều phiên bản hơn được hiển thị trong tab **Recommended**. Nhấp vào tab **x86 Images** và **Other Images** để xem thêm

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

5. Sau khi chọn hình ảnh hệ thống, nhấp **Next**. 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 **Finish**

3.2. Chạy ứng dụng trên thiết bị ảo

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

1. Trong Android Studio, chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run**



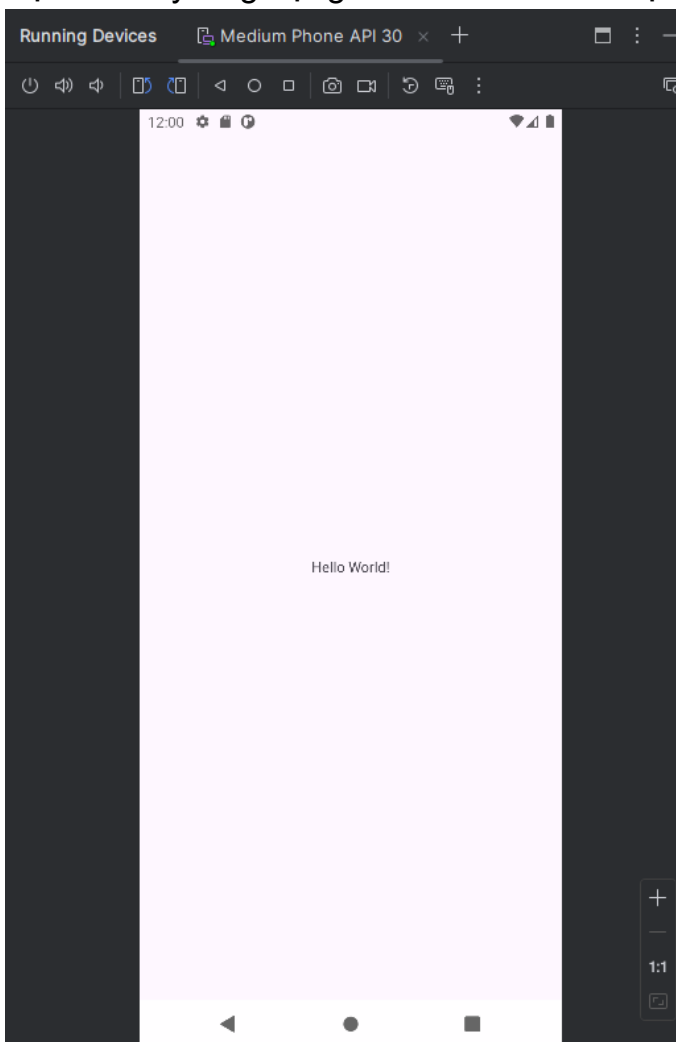
trên thanh công cụ

2. Trong cửa sổ **Select Deployment Target**, dưới mục **Available Virtual Devices**, chọn thiết bị ảo mà bạn vừa tạo



Trình giả lập sẽ khởi động như một thiết bị vật lý Tùy vào tốc độ của máy tính, quá trình này có thể mất một chút thời gian. Ứng dụng của bạn sẽ được biên dịch và khi trình giả lập sẵn sàng, Android Studio sẽ tải ứng dụng lên trình giả lập và chạy nó.

Bạn sẽ thấy ứng dụng Hello World hiển thị như trong hình sau



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

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

Những gì bạn cần:

- Thiết bị android như điện thoại hoặc máy tính bảng
- Cáp dữ liệu để kết nối thiết bị android với máy tính qua cổng USB
- Nếu bạn đang sử dụng hệ thống Linux hoặc Windows, bạn có thể cần thực hiện các bước bổ sung để chạy trên thiết bị phần cứng. Kiểm tra tài liệu **Using Hardware Devices**
- Bạn cũng có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Đối với trình điều khiển USB trên Windows, xem **OEM USB Drivers**

4.1. Bật gỡ lỗi USB


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

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

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

4.2. Chạy ứng dụng của bạn trên thiết bị

Giờ bạn có thể kết nối thiết bị và chạy ứng dụng từ Android Studio

1. Kết nối thiết bị của bạn với máy tính 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 trình giả lập có sẵn và thiết bị được kết nối
3. Chọn thiết bị của bạn, nhấp **OK**

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

Xử lý sự cố

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

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

Nếu máy tính vẫn không tìm thấy thiết bị hoặc hiển thị trạng thái “unauthorized”, hãy làm theo các bước sau:

1. Rút cáp khỏi thiết bị
2. Trên thiết bị, mở **Developer options in Settings app**
3. Nhấn vào thu hồi quyền **USB Debugging**
4. Kết nối lại thiết bị với máy tính
5. Khi được nhắc cấp ủy quyền

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

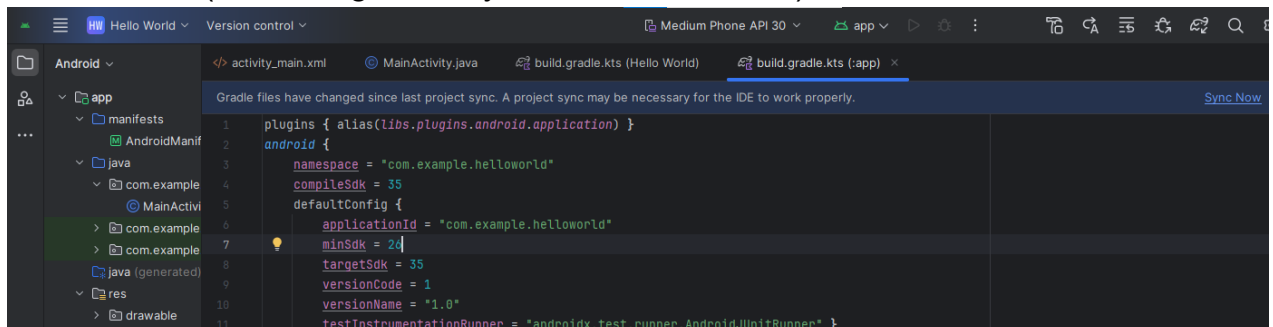
Nhiệm vụ 5: Thay đổi cấu hình Gradle của ứng dụng

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

5.1. Thay đổi phiên bản minimum SDK cho ứng dụng

Làm theo các bước sau:

1. Mở thư mục **Gradle Scripts** nếu thư mục chưa mở và nhấp vào tệp **build.gradle.kts(Module:app)**
Nội dung của tệp tin xuất hiện trong trình soạn thảo mã
2. Trong khối `defaultConfig`, hãy thay đổi giá trị của `minSdk` thành 26 như hiển thị bên dưới (ban đầu giá trị này được đặt thành 24)



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

5.2. Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi đối với các tệp cấu hình xây dựng trong một dự án, Android Studio yêu cầu bạn *đồng bộ hóa* 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 **Sync Now** trên thanh thông báo xuất hiện khi thực hiện thay đổi (như thể hiện trong hình trước) hoặc nhấn vào biểu

tượng **Sync Project with Gradle Files**  trong thanh công cụ


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

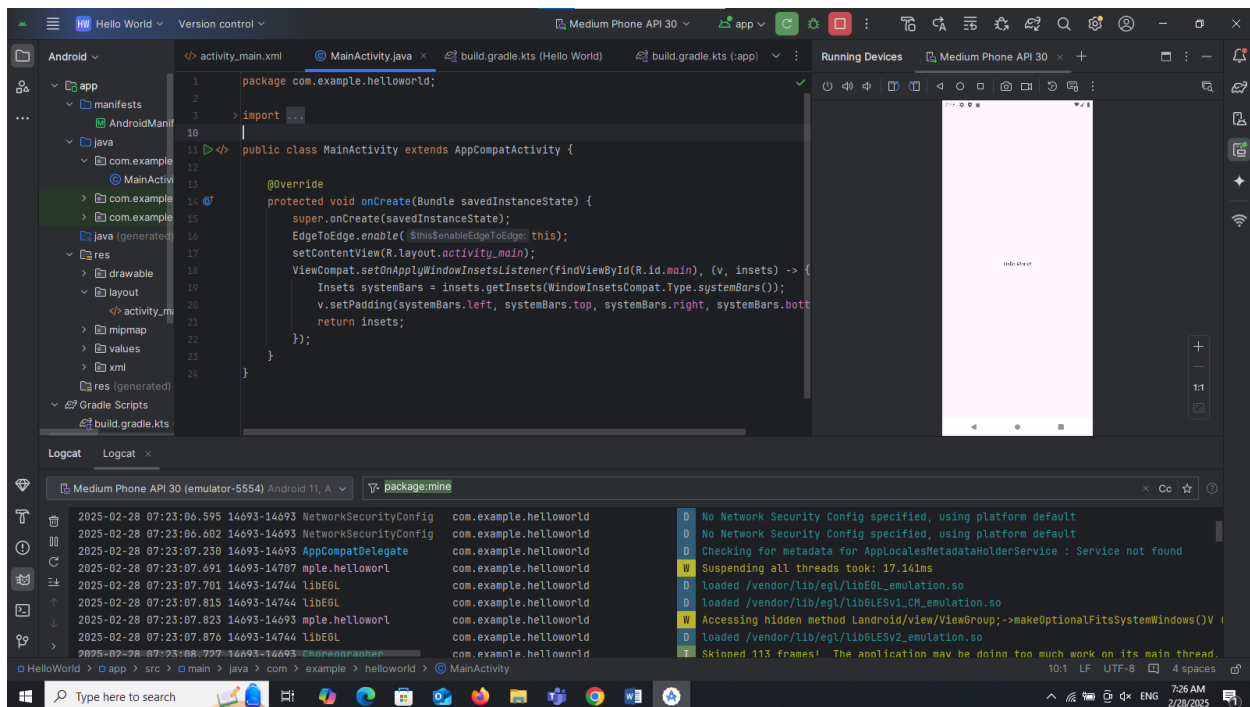
Để hiểu sâu hơn về Gradle, hãy tham khảo tài liệu **Build System Overview** và **Configuring Gradle Builds**

Nhiệm vụ 6: Thêm log statement 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 gỡ lỗi mạnh mẽ mà bạn có thể sử dụng để kiểm tra các giá trị, đường dẫn thực thi và báo cáo các ngoại lệ

6.1. Xem ngăn Logcat

Để xem ngăn **Logcat**, hãy nhấp vào biểu tượng **Logcat**  ở thanh công cụ bên trái màn hình Android Studio như minh họa trong hình bên dưới



Trong hình trên:

1. Tab **Logcat** để mở và đóng ngăn **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 của Log được để mặc định, hiển thị tất cả các thông báo Log. Các thiết lập bao gồm **Debug**, **Error**, **Info** và **Warn**

6.2 Thêm câu lệnh Log vào ứng dụng của bạn

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

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

Các phần của tin nhắn bao gồm:

- Log: Lớp **Log** để gửi tin nhắn log đến ngăn Logcat
- d: Cài đặt mức **Debug** Log để lọc hiển thị thông báo log trong ngăn Logcat. Các mức log 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 log được định nghĩa là hằng số cho Activity:

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

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

Làm theo các bước sau:

1. Mở ứng dụng Hello World của bạn trong Android studio và mở MainActivity.
2. Để tự động thêm các lệnh 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 thiết lập **Insert imports on paste to All**
4. Chọn **Apply** và sau đó nhấn **OK**
5. Trong phương thức onCreate() của MainActivity, thêm câu lệnh sau:

```
Log.d( tag: "MainActivity", msg: "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( $this$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;
    });
    Log.d( tag: "MainActivity", msg: "Hello World");
}
```

6. Nếu ngăn Logcat chưa mở, hãy nhấp vào biểu tượng Logcat ở thanh công cụ bên trái Android Studio để mở
7. Kiểm tra xem tên mục tiêu và tên gói của ứng dụng có đúng không
8. Thay đổi mức Log trong ngăn **Logcat** thành **Debug** (hoặc giữ nguyên vì có rất ít thông báo log)
9. Chạy ứng dụng của bạn

Thông báo sau sẽ xuất hiện trong ngăn Logcat:

```
2025-02-28 07:43:40.972 15720-15720 MainActivity com.example.helloworld D Hello World
```

Câu hỏi 1

Tên của tệp bố cục cho main activity là gì?

- MainActivity.java
- AndroidManifest.xml
- **activity_main.xml**
- build.gradle

Câu hỏi 2

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

- **app_name**
- xmlns:app
- android:name
- applicationId

Câu hỏi 3

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

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

Câu hỏi 4

Giả sử ứng dụng của bạn bao gồm câu lệnh log này:

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

Bạn thấy câu lệnh "MainActivity layout is complete" trong ngăn **Logcat** nếu menu cấp độ Log được đặt thành tùy chọn nào sau đây? (Gợi ý: trả lời nhiều câu hỏi là được.)

- **Verbose**
- **Debug**

- Info
- Warn
- Error
- Assert

1.2) 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 *ché 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

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

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị một bố cục được định nghĩa trong tệp bố cục activity_main.xml, trong đó bao gồm một TextView với 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 một 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 tương tác với người dùng. Bạn sẽ tạo ứng dụng bằng mẫu Empty Activity. Đồng thời, bạn cũng học cách sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế bố cục và chỉnh sửa bố cục trong XML. Việc phát triển những kỹ năng này là cần thiết để bạn hoàn thành các bài thực hành khác trong khóa học này

Những điều bạn nên biết

Bạn cần làm quen với:

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

Những gì bạn sẽ học

- Cách tạo một ứng dụng với hành vi tương tác
- Cách sử dụng layout editor để thiết kế bố cục
- Cách chỉnh sửa bố cục trong XML
- Rất nhiều thuật ngữ mới. Hãy tham khảo **Vocabulary words and concepts glossary** để có các định nghĩa dễ hiểu

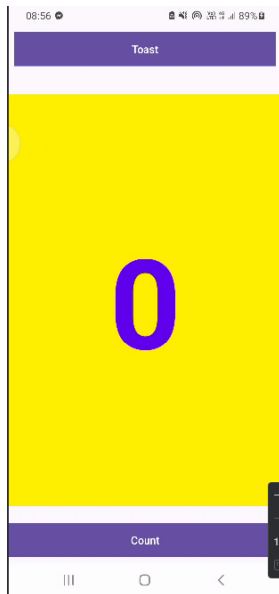
Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

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

Sau đây là hình ảnh ứng dụng đã hoàn thành:

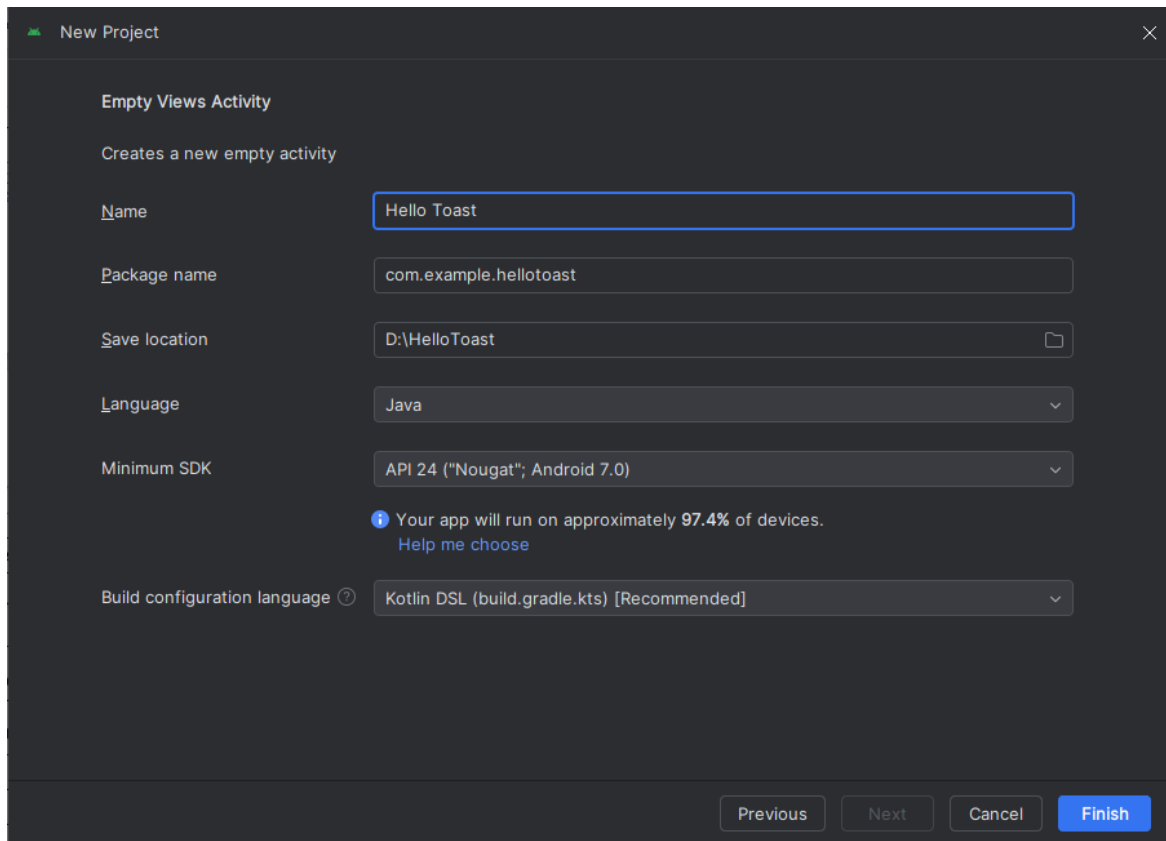



Nhiệm vụ 1: Tạo và khám phá một dự án mới

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

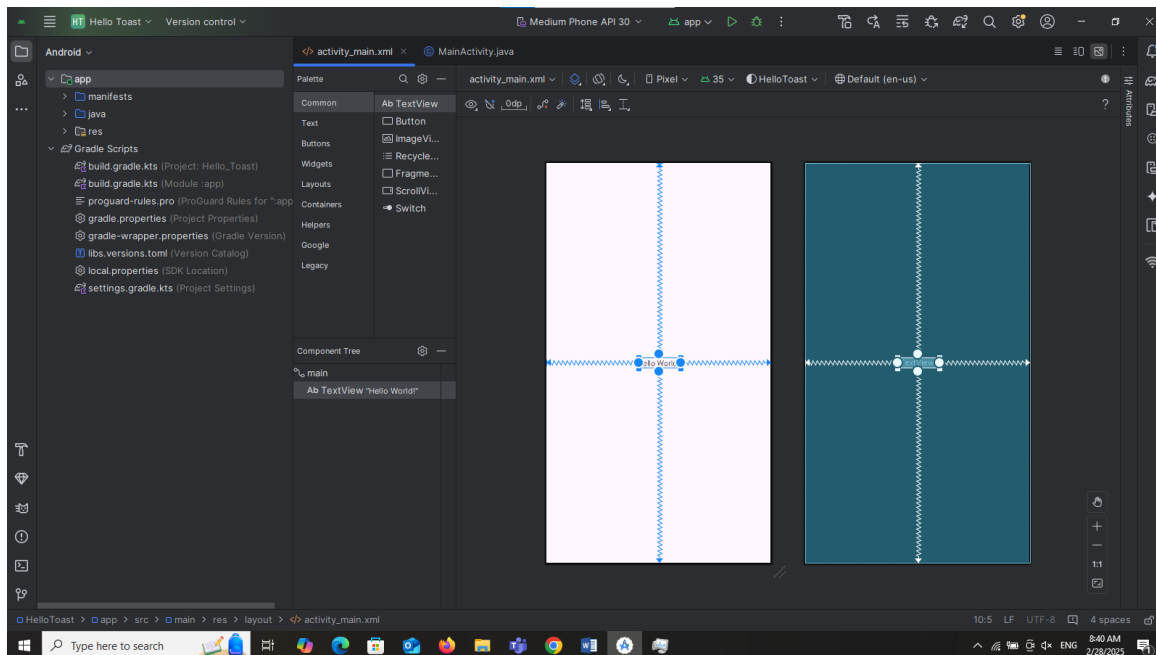
1.1 Tạo dự án Android Studio

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



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

1.2 Khám phá trình chỉnh sửa bố cục



1. Trong thư mục **app > res > layout** của **Project > Android** pane, nhấn đúp vào tệp **activity_main.xml** để mở nó nếu tệp chưa được mở
2. Nhấp vào tab **Design** nếu tab này chưa được chọn. Bạn sử dụng tab **Design** để thao tác với các phần tử và bố cục, và sử dụng tab **Text** để chỉnh sửa mã XML cho bố cục
3. Ngăn **Palettes** hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục của ứng dụng
4. Ngăn **Component tree** hiển thị cấu trúc phân cấp của các phần tử UI. Các phần tử View được tổ chức thành một cây phân cấp gồm các phần tử cha và con, trong đó phần tử con kế thừa các thuộc tính của phần tử cha. Trong hình minh họa, TextView là một phần tử con của **ConstraintLayout**. Bạn sẽ học thêm về các phần tử này trong bài học sau
5. Các ngăn design và blueprint của layout editor hiển thị các phần tử UI trong bố cục. Trong hình minh họa, bố cục chỉ hiển thị một phần tử: một TextView hiển thị "Hello World"
6. Tab **Attributes** hiển thị ngăn **Attributes**, nơi bạn có thể thiết lập các thuộc tính cho một phần tử UI

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


Trong nhiệm vụ này, bạn sẽ tạo bố cục giao diện người dùng cho ứng dụng HelloToast trong layout editor bằng cách sử dụng các tính năng của **ConstraintLayout**. Bạn có thể tạo các ràng buộc thủ công, như được minh họa sau đó, hoặc tự động bằng công cụ **Autoconnect**.


2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện các bước sau:

1. Mở tệp **activity_main.xml** từ **Project > Android** pane nếu tệp này chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào đó

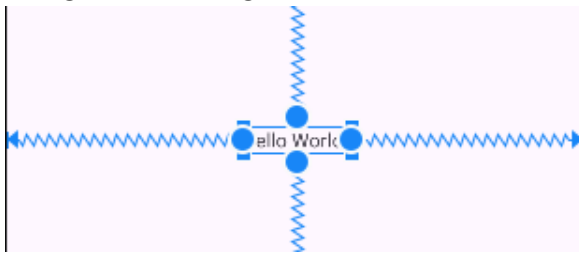
Nếu không có bản thiết kế (**blueprint**), nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**

2. Công cụ **Autoconnect**  cũng nằm trên thanh công cụ và nó được bật theo mặc định. Đối với bước này, đảm bảo rằng công cụ này không tắt

3. Nhấp vào nút Zoom  in để phóng to các ngăn thiết kế và bản thiết kế để quan sát kỹ hơn

4. Chọn **TextView** trong ngăn Component Tree. Phần tử TextView "Hello World" sẽ được làm nổi bật trong cả hai ngăn design và blueprint, và các ràng buộc của phần tử sẽ hiển thị

5. Làm theo hình minh họa động dưới đây: Nhấp vào biểu tượng hình tròn ở phía bên phải của TextView để xóa ràng buộc ngang kết nối phần tử này với cạnh phải của bố cục. TextView sẽ chuyển sang phía bên trái vì nó không còn bị ràng buộc vào cạnh phải. Để thêm lại ràng buộc ngang, nhấp vào cùng biểu tượng hình tròn và kéo một đường tới cạnh phải của bố cục



Trong các ngăn **blueprint** hoặc **design**, các tay nắm (**handles**) sau sẽ xuất hiện trên phần tử **TextView**:

- **Constraint handle:** Để tạo một ràng buộc như minh họa trong hình động ở trên, nhấp vào tay nắm ràng buộc, được hiển thị dưới dạng một hình tròn ở cạnh của một phần tử. Sau đó, kéo tay nắm đó đến một tay nắm ràng buộc khác hoặc đến đường biên của phần tử cha. Một đường gấp khúc sẽ biểu thị ràng buộc được tạo ra.



- **Resizing handle:** Để thay đổi kích thước phần tử, kéo các tay nắm chỉnh kích thước hình vuông. Khi bạn kéo, tay nắm sẽ chuyển thành một góc xiên.

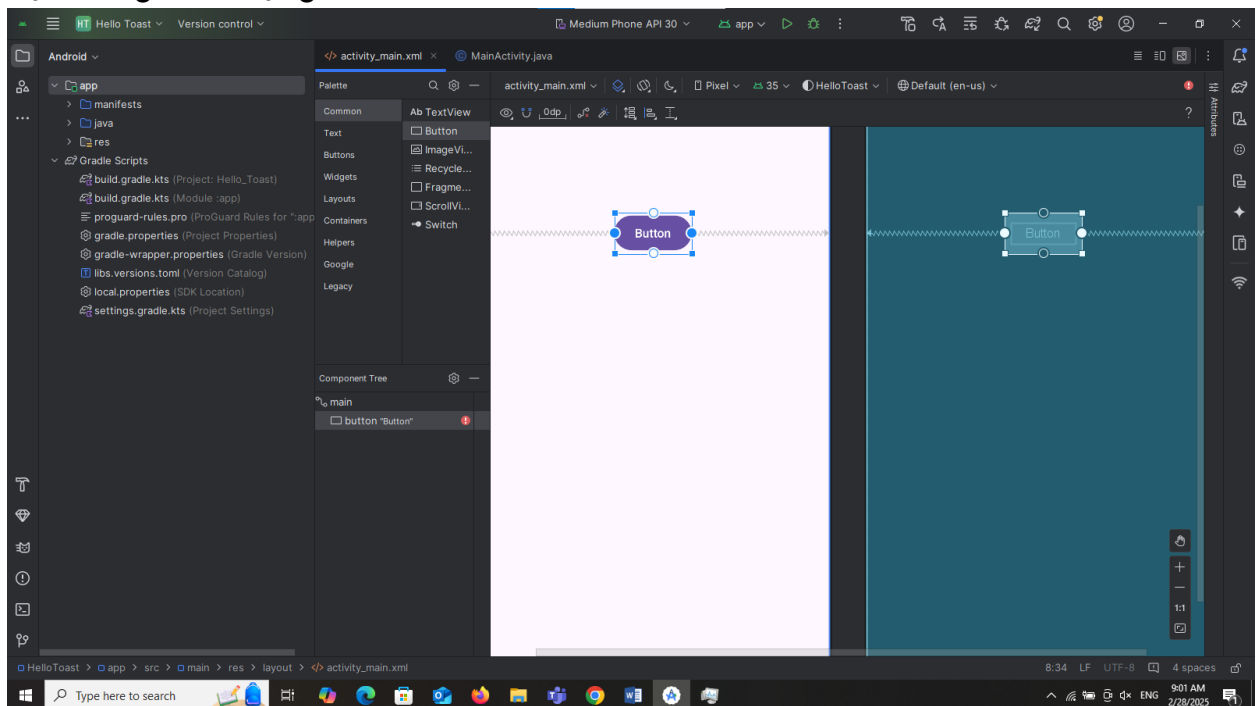


2.2 Thêm một Button vào bố cục

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

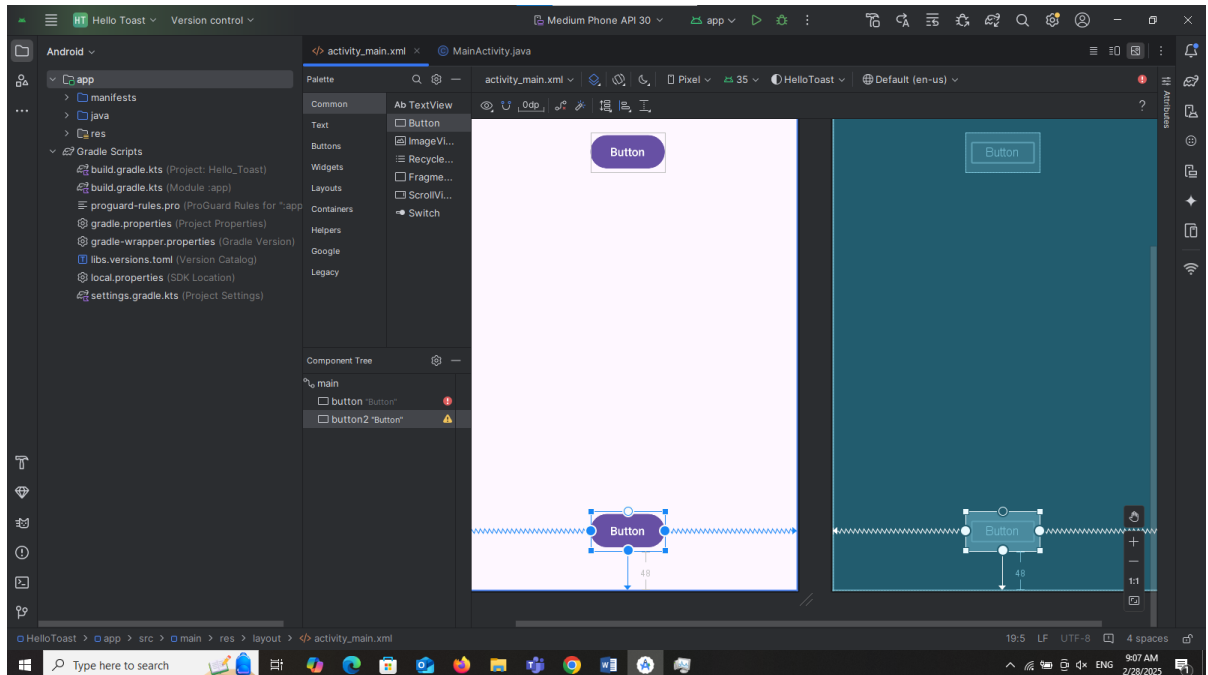
Thực hiện các bước sau để thêm một Button:


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



2.3 Thêm Nút thứ hai vào bố cục

1. Kéo một **Button** khác từ ngăn **Palette** vào giữa bố cục, như được minh họa trong hình động bên dưới. Công cụ **Autoconnect** có thể tự động tạo các ràng buộc ngang cho bạn (nếu không, bạn có thể tự kéo các ràng buộc này)
2. Kéo một ràng buộc dọc từ **Button** xuống cuối của bố cục (như minh họa trong hình bên dưới)



Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di chuột 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 cụ thể, hãy nhấp vào tay cầm đặt 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 thiết lập lại tất cả các ràng buộc trong bố cục của mình

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

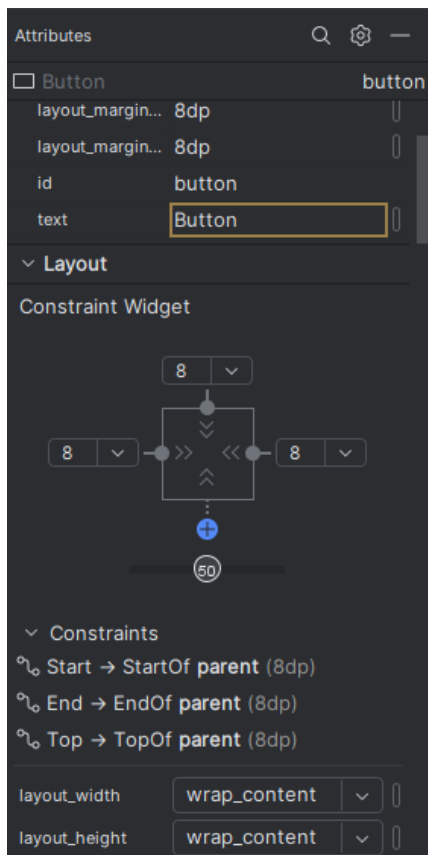
Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Bạn có thể tìm các thuộc tính (được gọi là thuộc tính *properties*) chung cho tất cả các View trong **View class documentation**

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

3.1 Thay đổi kích thước Button

Layout Editor cung cấp các tay cầm thay đổi kích thước ở cả bốn góc của một View, giúp 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 việc này sẽ mã hóa cứng các kích thước chiều rộng và chiều cao. Hạn chế mã hóa cứng kích thước cho hầu hết các phần tử View, vì các kích thước được mã hóa cứng không thể thích ứng với nội dung và kích thước màn hình khác nhau.

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



Trong hình trên:

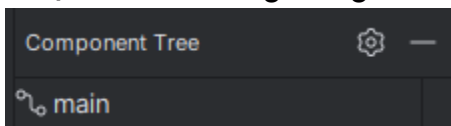
1. **Height control.** Điều khiển này xác định thuộc tính **layout_height** và xuất hiện ở hai đoạn trên và dưới của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành **wrap_content**, nghĩa là View sẽ mở rộng

theo chiều dọc khi cần để phù hợp với nội dung của nó. Số "8" chỉ ra một lề chuẩn được đặt là 8dp

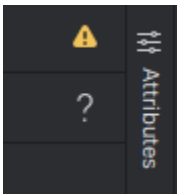
2. **Width control.** Điều khiển này xác định thuộc tính **layout_width** và xuất hiện ở hai đoạn bên trái và phải của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành **wrap_content**, nghĩa là View sẽ mở rộng theo chiều ngang khi cần để phù hợp với nội dung của nó, tối đa đến một lề là 8dp
3. **Nút đóng Attributes pane.** Nhấp để đóng bảng điều khiển

Làm theo các bước sau:

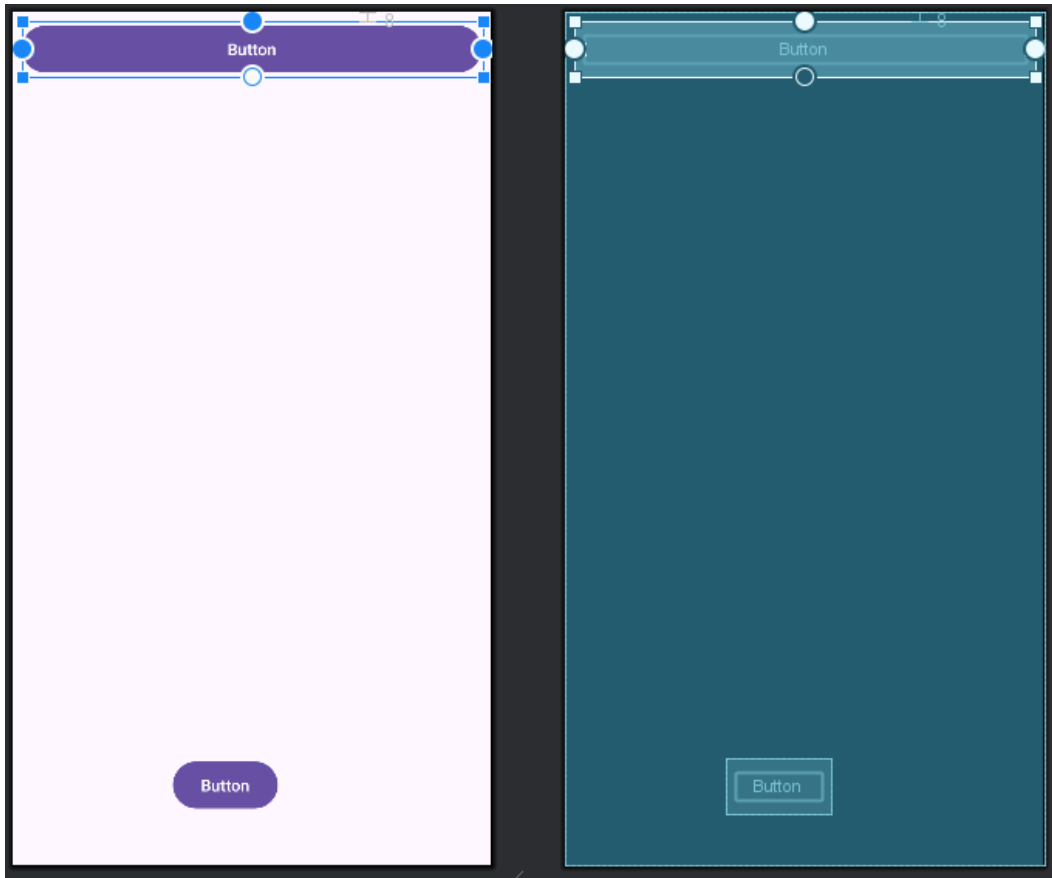
1. Chọn button trong bảng **Component Tree**



2. Nhấp vào tab **Attributes** ở phía bên phải của cửa sổ trình chỉnh sửa bố cục

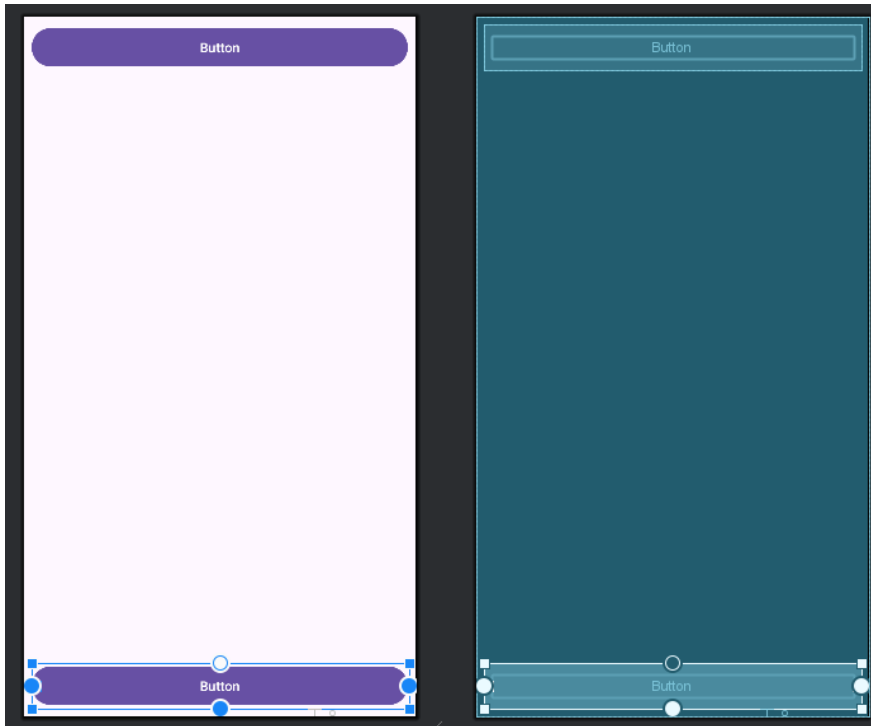


3. Nhấp vào điều khiển độ rộng hai lần - lần nhấp đầu tiên thay đổi thành **Fixed** với các đường thẳng, và lần nhấp thứ hai thay đổi thành **0dp (Match Constraints)** với các cuộn lò xo, như minh họa trong hình động dưới đây



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

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



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

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

3.2. Thay đổi thuộc tính của Button

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

Ngăn **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`

Các bước thực hiện:

1. Sau khi chọn Button đầu tiên, hãy chỉnh sửa trường ID ở đầu ngăn 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/design_default_color_primary**. (Khi bạn nhập @c, các lựa chọn sẽ xuất hiện để dễ dàng lựa chọn)
3. Đặt thuộc tính **textColor** thành **@android:color/white**
4. Sửa thuộc tính **text** thành **Toast**
5. Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng **button_count** làm ID, **Count** cho thuộc tính **text** và cùng màu cho nền và văn bản như các bước trước

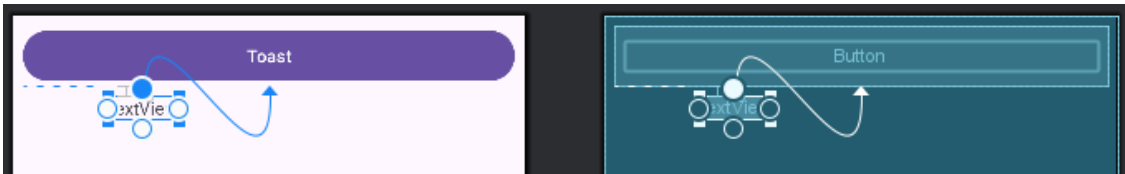


Nhiệm vụ 4: Thêm TextEdit và đặt thuộc tính của nó

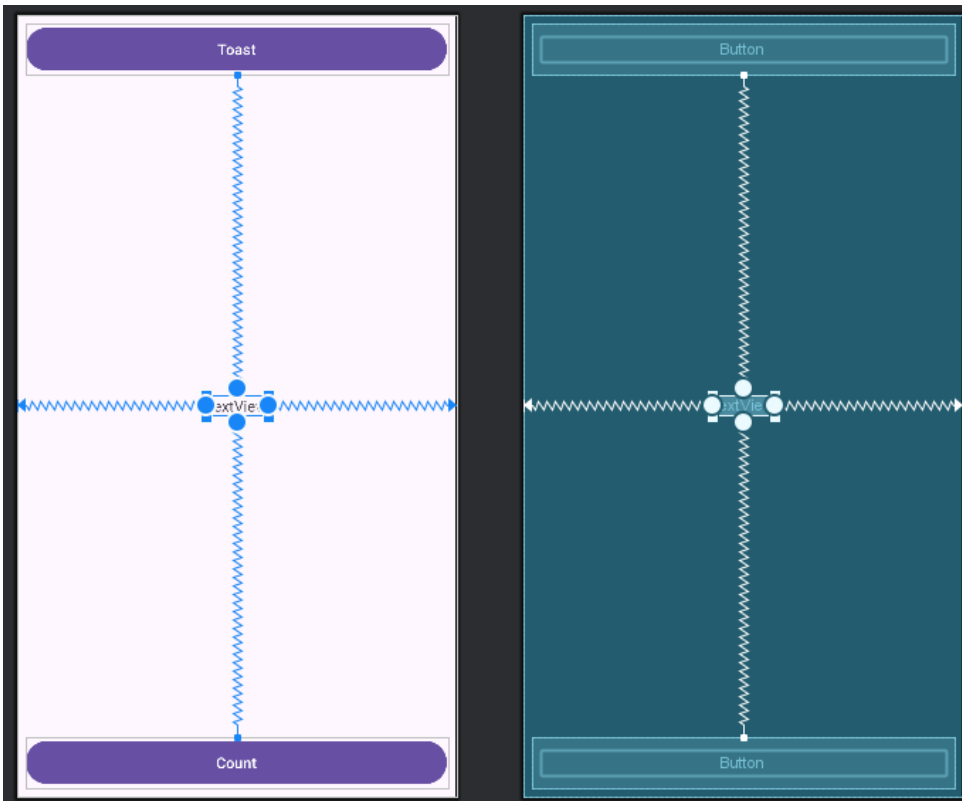
Một lợi ích của **Constraints Layout** có khả năng căn chỉnh hoặc hạn chế các thành phần so với các thành phần khác. Trong nhiệm vụ này bạn sẽ thêm một **TextView** ở giữa bố cục, và giới hạn nó theo chiều ngang so với lề và theo chiều dọc so với hai Button. Sau đó bạn thay đổi các thuộc tính của **TextView**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

4.1. Thêm một TextView và ràng buộc

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



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



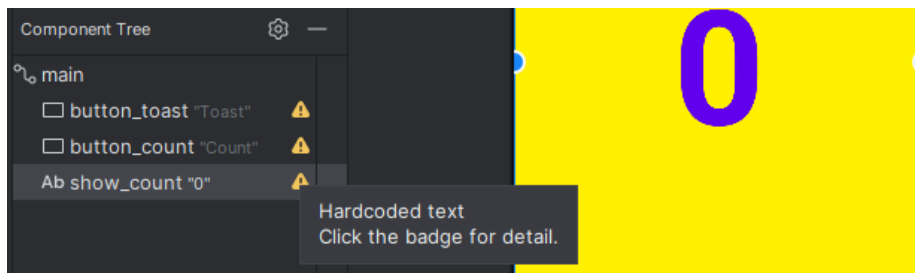
4.2. Đặt thuộc tính của TextView

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

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


Nhiệm vụ 5: Chỉnh sửa 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 được hiển thị bên dưới. Cảnh báo tương tự xuất hiện cho cả ba thành phần: ***“Hardcoded text. Click the badge for detail.”***

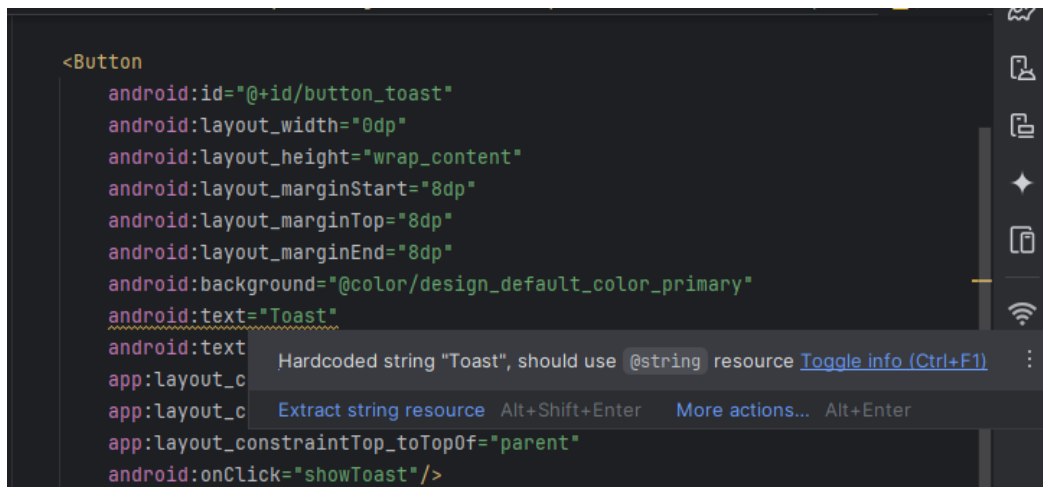


Cách dễ nhất để khắc phục sự cố bố cục là chỉnh sửa bố cục trong XML. Mặc dù trình chỉnh sửa bố cục là một công cụ mạnh mẽ, một số thay đổi dễ thực hiện trực tiếp trong mã nguồn XML hơn

5.1. Mở mã XML của bố cục

Đối với nhiệm vụ này, mở tệp **activity_main.xml** nếu tệp này chưa mở và nhấp vào biểu tượng  ở cuối trình chỉnh sửa bố cục.

Trình soạn thảo XML xuất hiện, thay thế các ngăn thiết kế và bản thiết kế. Như bạn có thể thấy trong hình bên dưới, 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

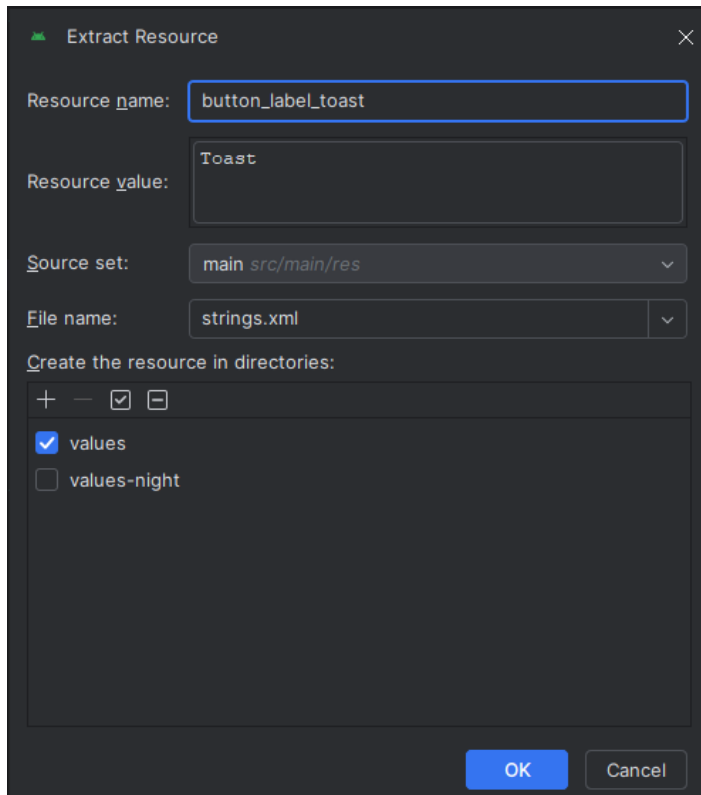


5.2. Trích xuất tài nguyên chuỗi

Thay vì mã hóa cứng các chuỗi, cách tốt nhất là sử dụng các tài nguyên chuỗi, biểu diễn các chuỗi. Việc có các chuỗi trong một tệp riêng biệt giúp quản lý chúng


dễ dàng hơn, đặc biệt là nếu bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, các tài nguyên chuỗi là bắt buộc để dịch và bản địa hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho từng ngôn ngữ

1. Nhấn một lần vào từ **“Toast”**
2. Nhấn tổ hợp phím **Alt+Enter** trên Windows hoặc **Option+Enter** trên macOS và chọn **Extract string resource** từ menu hiện lên
3. Nhập **button_label_toast** cho Resource name:



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

```
android:text="@string/button_label_toast"
```
5. Trích xuất cho các chuỗi còn lại: **button_label_count** cho **“Count”** và **count_init_value** cho **“0”**.
6. Trong ngăn **Project > Android**, mở rộng thư mục **res** trong thư mục **values**, sau đó nhấn đúp chuột vào **strings.xml** để xem các tài nguyên chuỗi của bạn trong tệp **strings.xml**

```
<resources>
    <string name="app_name">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
     <string name="count_init_value">0</string>
</resources>
```

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

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

Nhiệm vụ 6: Thêm xử lý onClick trên button


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

6.1. Thêm thuộc tính onClick và xử lý cho mỗi Button

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

1. Mở trình soạn thảo XML, tìm Button với **android:id** đặt cho **button_toast**.
2. Thêm thuộc tính **android:onClick** vào cuối của phần tử **button_toast** sau phần tử cuối cùng và trước chỉ báo kết thúc (**/>**)

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

3. Nhấp vào biểu tượng bóng đèn đỏ () xuất hiện cạnh thuộc tính. Chọn **Create onClick event handler**, chọn **MainActivity**, và nhấp **OK**.

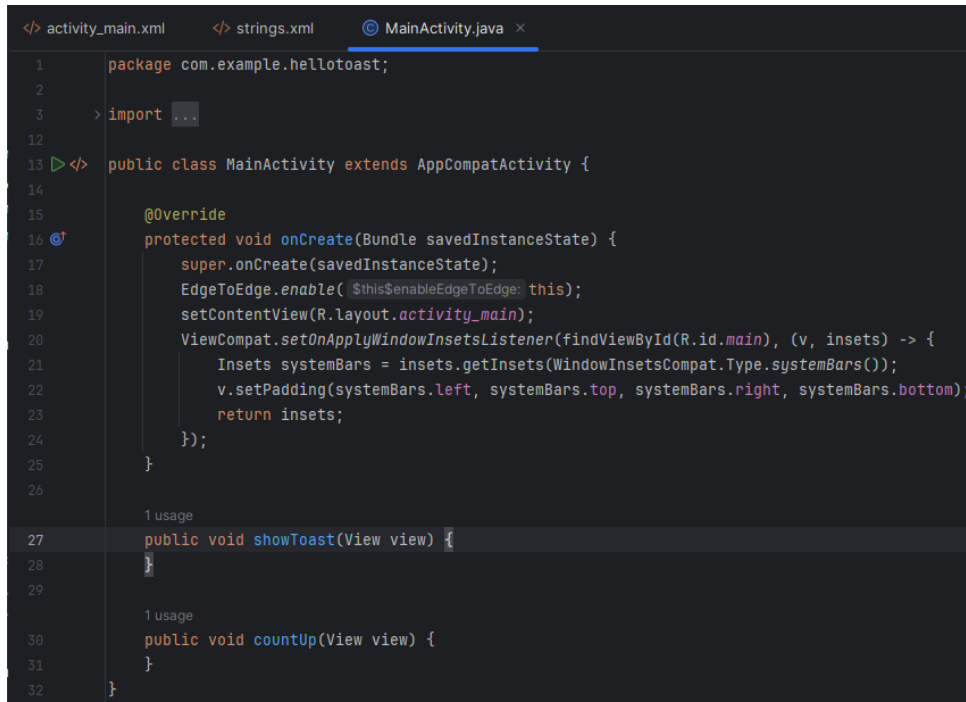
Nếu biểu tượng bóng đèn đỏ không xuất hiện, nhấp vào tên phương thức ("showToast"). Nhấn tổ hợp phím **Alt+Enter** (**Option+Enter** trên Mac), chọn **Create 'showToast(view)' in MainActivity**

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

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

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

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



```
<?xml version="1.0" encoding="utf-8"?>
<activity_main.xml>
<strings.xml>
@ MainActivity.java x
1 package com.example.hellotoast;
2
3 > import ...
12
13 public class MainActivity extends AppCompatActivity {
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         EdgeToEdge.enable(this);
19         setContentView(R.layout.activity_main);
20         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
21             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
22             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
23             return insets;
24         });
25     }
26
27     1 usage
28     public void showToast(View view) {
29
30     1 usage
31     public void countUp(View view) {
32     }
```

6.2. Chỉnh sửa trình xử lý nút Toast

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

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

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

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

2. Tạo một thể hiện của Toast, gọi là phương thức **makeText()** của lớp **Toast**

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

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

```
public void showToast(View view) {
    Toast toast = Toast.makeText(context: this,
}
```

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

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

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

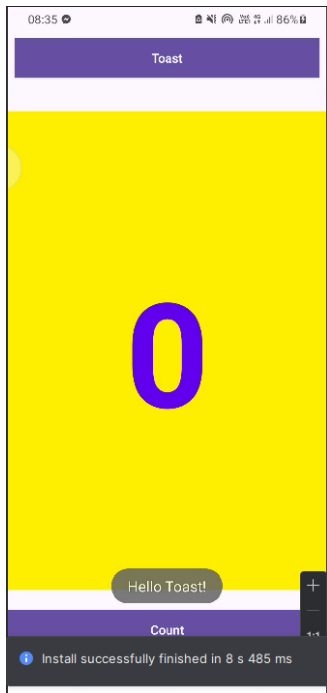
```
public void showToast(View view) {
    Toast toast = Toast.makeText(context: this, R.string.toast_message,
        Toast.LENGTH_LONG);
}
```

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

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

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

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



6.3.Chỉnh sửa trình xử lý nút Count

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

Trong mã xử lý phải:

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

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

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

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

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'count' in MainActivity** từ menu hiện lên. Việc này tạo một biến thành viên private ở

trên cùng của MainActivity, và Android Studio giả định bạn muốn tạo nó là một biến kiểu **int**

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

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

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

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

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

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

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

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

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

Một View, như một chuỗi, một tài nguyên có thể có một ID. Lệnh gọi **findViewById** lấy ID của một View làm tham số và trả về View đó. Vì phương

thức trả về một View, bạn phải ép kiểu kết quả thành loại View mà bạn muốn, trong trường hợp này là TextView.

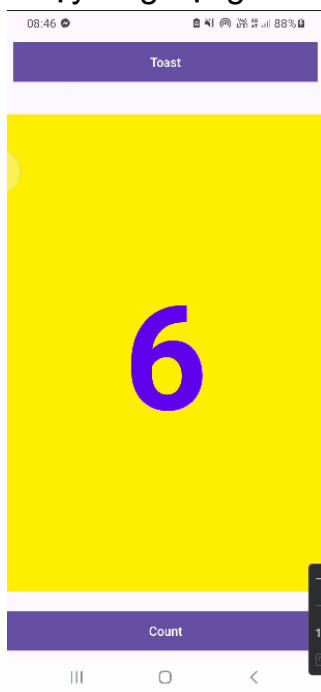
8. Bây giờ bạn đã gán TextView cho showCount, bạn có thể sử dụng biến để đặt **text** trong TextView thành giá trị của biến count. Thêm như sau vào phương thức countUp():

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

Toàn bộ phương thức countUp() sẽ như thế này:

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

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




Tóm tắt

View, ViewGroup, và bố cục:

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

Sử dụng trình chỉnh sửa bố cục:

- Nhấp vào tab Design để thao tác các thành phần và bố cục, và tab Text để chỉnh sửa mã XML cho bố cục
- Trong tab **Design**, ngăn **Palettes** hiển thị các thành phần UI mà bạn có thể sử dụng trong bố cục ứng dụng của mình, và ngăn **Component tree** hiển thị phân cấp chế độ xem của các thành phần UI
- Các ngăn design và blueprint của trình chỉnh sửa bố cục hiển thị các thành phần UI trong bố cục
- Tab **Attributes** hiển thị ngăn **Attributes** để thiết lập thuộc tính cho một phần tử UI
- **Constraint handle**: Nhấp vào **constraint handle**, được hiển thị dưới dạng hình tròn ở mỗi bên của phần tử, sau đó kéo đến **constraint handle** khác hoặc đến ranh giới cha để tạo ràng buộc. Ràng buộc được biểu thị bằng đường ngoằn ngoèo
- **Resizing handle**: Bạn có thể kéo **resizing handle** hình vuông để thay đổi kích thước phần tử. Trong khi kéo, handle sẽ thay đổi thành góc nghiêng
- Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử đó nhấp vào biểu tượng  để xóa tất cả các ràng buộc trên phần tử đã chọn
- Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Ngăn này cũng bao gồm một bảng điều khiển kích thước hình vuông được gọi là trình kiểm tra view ở trên cùng. Các ký hiệu bên trong hình vuông biểu thị các thiết lập chiều cao và chiều rộng

Cài đặt chiều rộng và chiều cao bố cục:

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

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

Trích xuất tài nguyên chuỗi:

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

1. Nhấp một lần vào chuỗi được mã hóa cứng để trích xuất, nhấn **Alt-Enter** (**Option-Enter** trên máy Mac) và chọn **Create string value resource** từ menu bật lên
2. Đặt **Resource value**
3. Nhấn **OK**. Điều này tạo ra một tài nguyên chuỗi trong tệp **values/res/string.xml** và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên đó: **@string/button_label_toast**

Xử lý nhấn:

- Trình xử lý nhấn là một phương thức được gọi khi người dùng nhấn hoặc chạm vào một thành phần UI
- Chỉ định trình xử lý nhấn cho một thành phần UI như nút bằng cách nhập tên của thành phần đó vào trường **onClick** trong ngăn **Attributes** của tab **Design** hoặc trong trình soạn thảo XML bằng cách thêm thuộc tính **android:onClick** vào một thành phần UI như Button
- Tạo trình xử lý nhấn trong **Activity** chính bằng cách sử dụng tham số View. Ví dụ: `public void showToast(View view) {...}`

Hiển thị thông điệp Toast:

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

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

1.3) Trình chỉnh sửa bố cục

Giới thiệu

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

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

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

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

Những gì bạn nên biết:

Bạn nên biết:

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

Những gì bạn sẽ học:

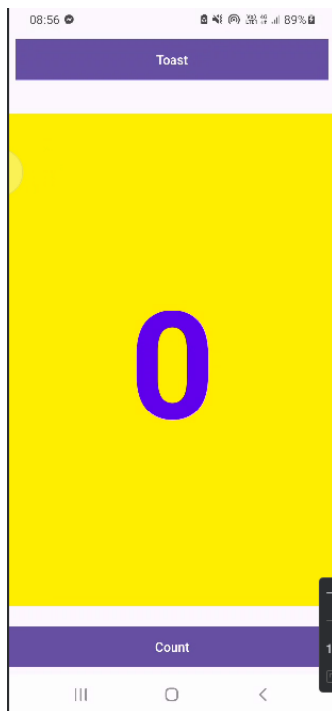
- Cách tạo biến thể bố cục theo hướng ngang
- Cách tạo biến thể bố cục cho máy tính bảng và màn hình lớn hơn
- Cách sử dụng ràng buộc đường cơ sở để căn chỉnh các thành phần UI với văn bản
- Cách sử dụng các nút đóng gói và căn chỉnh để căn chỉnh các thành phần trong bố cục
- Cách định vị chế độ xem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

Những gì bạn sẽ làm:

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

Tổng quan

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



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

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

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

Nhiệm vụ 1: Tạo biến thể bố cục

Trong bài học trước, thử thách mã hóa yêu cầu thay đổi bố cục của ứng dụng Hello Toast để nó có thể vừa vận theo hướng ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo các biến thể bố cục của mình theo hướng ngang và hướng dọc cho điện thoại và cho màn hình lớn hơn như máy tính bảng

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



Trong ảnh trên gồm:

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

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



Trong hình trên:

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


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

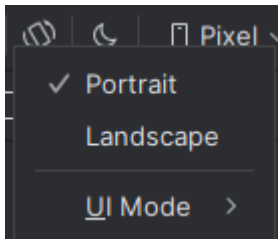


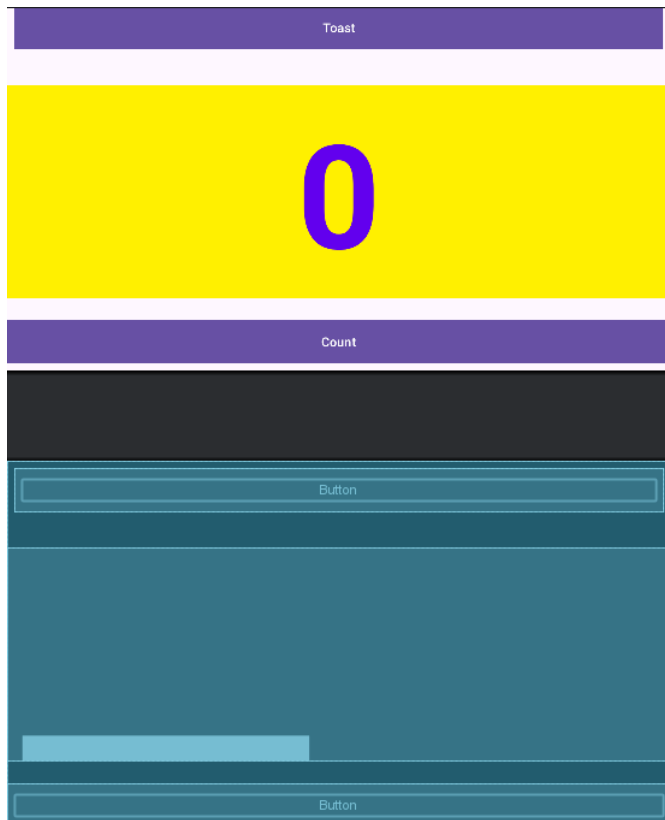
1. **Pan screen**: Chế độ con trỏ trong trình chỉnh sửa bố cục
2. **Zoom In**: Phóng to bố cục
3. **Zoom Out**: Thu nhỏ bố cục

1.1. Xem trước bố cục theo hướng ngang

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

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



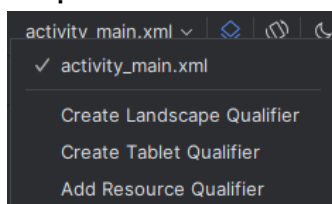


1.2. Tạo biến thể bố cục theo hướng ngang

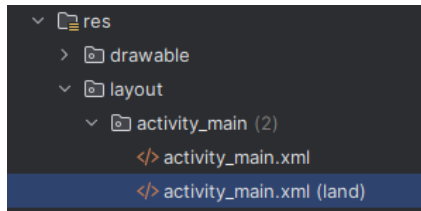
Sự khác biệt trực quan giữa hướng dọc và hướng ngang cho bố cục này là chữ số (0) trong phần tử TextView quá thấp so với hướng ngang - quá gần nút **Count**. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử TextView có thể xuất hiện quá lớn hoặc không được căn giữa vì kích thước văn bản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

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

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



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



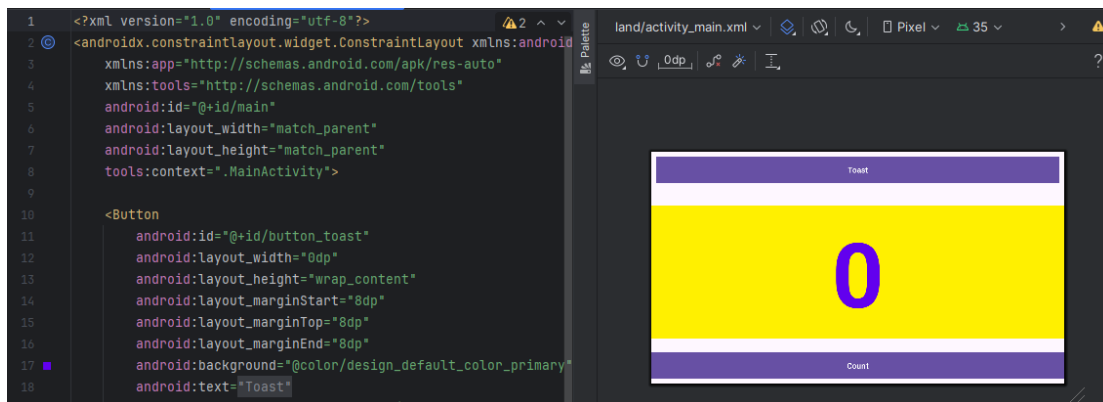
1.3. Xem trước bố cục cho các thiết bị khác nhau

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


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

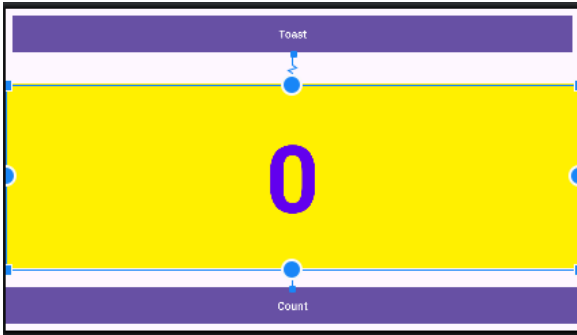
1.4. Thay đổi bố cục theo hướng ngang

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

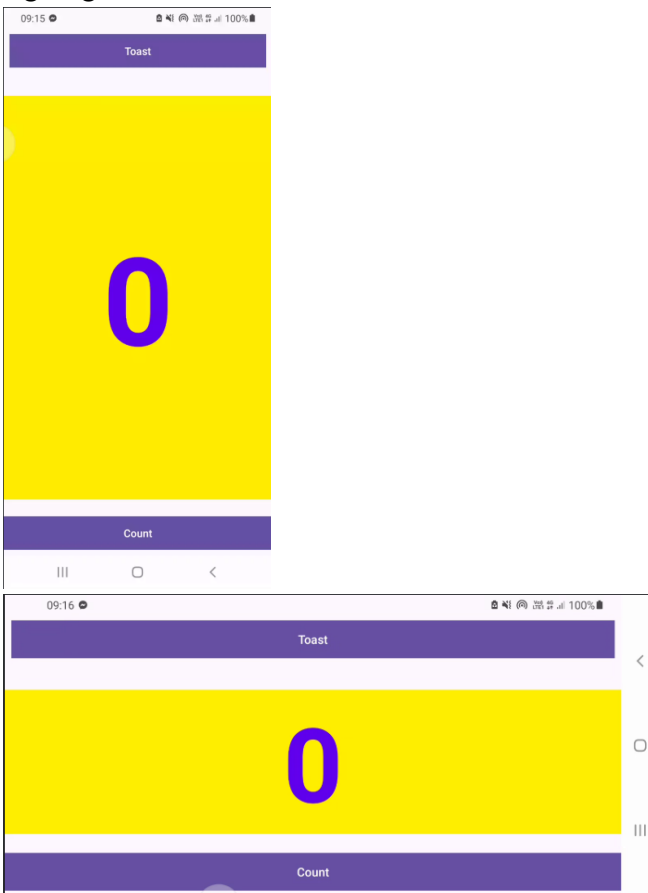


Để thay đổi bố cục, làm theo các bước sau:

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



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



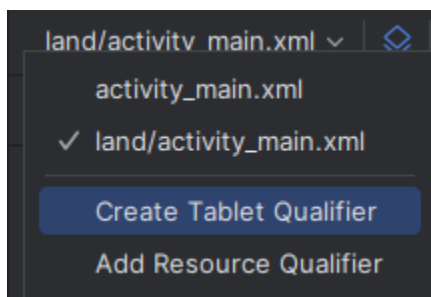
1.5. Tạo biến thể bố cục cho tablets

Như bạn đã học trước đó, bạn có thể xem trước bố cục cho các thiết bị khác nhau bằng cách nhấp vào nút **Devices for Preview** trên thanh công cụ trên cùng. Nếu bạn chọn một thiết bị như Nexus 10 (máy tính bảng) từ menu, bạn có thể thấy rằng bố cục không lý tưởng cho màn hình máy tính bảng - văn bản của mỗi nút quá nhỏ và cách sắp xếp các thành phần nút ở trên cùng và dưới cùng không lý tưởng cho máy tính bảng màn hình lớn



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


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

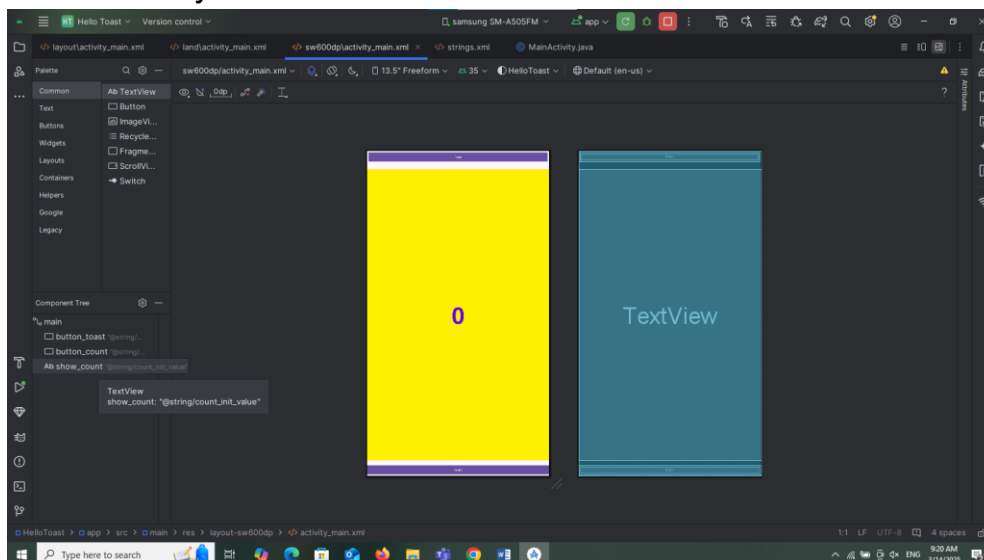


Một cửa sổ mới mở ra với tab **sw600dp/activity_main.xml** hiển thị bố cục cho thiết bị có kích thước máy tính bảng. Trình chỉnh sửa cũng chọn một thiết bị máy tính bảng, chẳng hạn như Nexus 10, để xem trước. Bạn có thể thay đổi bố cục này, dành riêng cho máy tính bảng, mà không cần thay đổi các bố cục khác

1.6. Thay đổi biến thể bố cục cho tablets

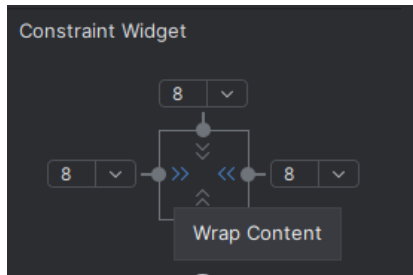
Bạn có thể sử dụng ngăn **Attributes** trong tab **Design** để thay đổi thuộc tính cho bố cục này

1. Tắt công cụ **Autoconnect**  trên thanh công cụ. Đối với bước này, hãy đảm bảo rằng công cụ đã bị vô hiệu hóa
2. Xóa tất cả các ràng buộc trong bố cục bằng cách nhấp vào biểu tượng **Clear All Constraints** trên thanh công cụ
Khi các ràng buộc được gỡ bỏ, bạn có thể di chuyển và thay đổi kích thước các thành phần trên bố cục một cách tự do
3. Trình chỉnh sửa bố cục cung cấp các nút điều khiển thay đổi kích thước ở cả bốn góc của một phần tử để thay đổi kích thước của phần tử đó. Trong **Component Tree**, chọn TextView có tên **show_count**. Để TextView không cản trở bạn có thể tự do kéo các phần tử Button, hãy kéo một góc của phần tử đó để thay đổi kích thước



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

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



id	button_toast
onClick	showToast ▾
text	@string/button_lab...
textColor	@android:color/...
textSize	60sp ▾


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

Sử dụng ràng buộc cơ bản


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

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

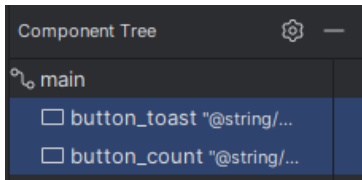


2. Sử dụng ràng buộc đường cơ sở, bạn có thể ràng buộc **button_count** sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của **button_toast**. Chọn phần tử **button_count**, sau đó nhấp chuột phải để hiển thị menu, nhấp vào biểu tượng  **Show Baseline** để hiển thị base line
3. Nhấp và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

Mở rộng các nút theo chiều ngang

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

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

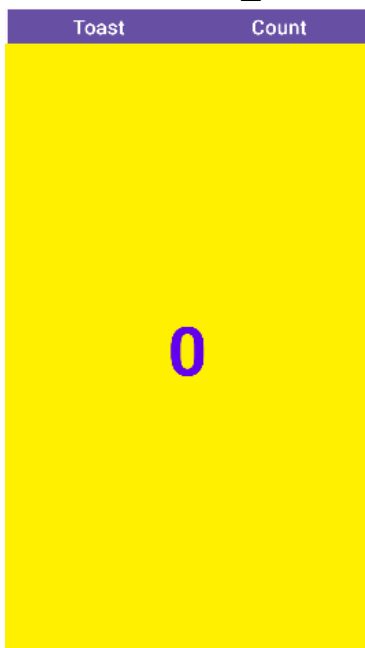


2. Nhấn chuột phải và **Organize** chọn trong menu hiện lên, và chọn **Expand Horizontally**

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



3. Để hoàn thiện bố cục, hãy ràng buộc TextView **show_count** vào cuối **btn_toast** và vào các cạnh và cuối của bố cục
4. Các bước cuối cùng là thay đổi **layout_width** và **layout_height** của TextView **show_count** thành **Match Constraints** và **textSize** thành **200sp**



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

Nhiệm vụ 2: Thay đổi bố cục thành LinearLayout

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

LinearLayout yêu cầu các thuộc tính này:

- `layout_width`
- `layout_height`
- `orientation`

Thuộc tính **`layout_width`** và **`layout_height`** có thể lấy một trong các giá trị sau:

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

Thuộc tính **`orientation`** có thể là:

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

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

2.1. Thay đổi ViewGroup gốc thành LinearLayout

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

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

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

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

4. Đảm bảo thẻ đóng ở cuối mã đã thay đổi thành **</LinearLayout>**. Nếu thẻ chưa tự động thay đổi, hãy thay đổi thủ công.
5. Dưới dòng tag **<LinearLayout**, thêm thuộc tính sau **android:layout_height**:

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

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

2.2. Thay đổi các thuộc tính của phần tử trong LinearLayout

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

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

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

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

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

5. Tìm đến nút **button_count**, thay đổi các thuộc tính sau:

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

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

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

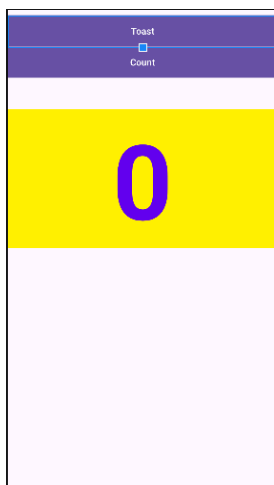
7. Tìm đến TextView **show_count**, và thay đổi các thuộc tính sau:

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

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

```
app:layout_constraintBottom_toTopOf="@+id/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 bên phải cửa sổ Android Studio (nếu chưa được chọn) để xem bản xem trước của bố cục:



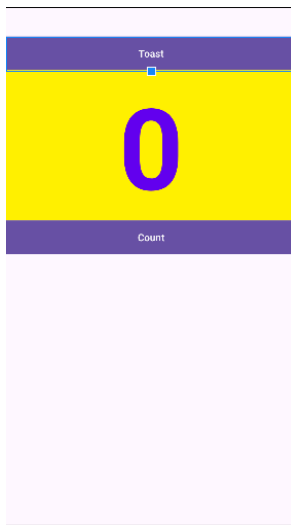
2.3. Thay đổi vị trí các phần tử trong LinearLayout

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

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

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

Bằng cách di chuyển **btn_count** bên dưới **TextView**, nút Count ở phía dưới. Bản xem trước của bố cục hiện trông như sau:



2.4. Thêm trọng số cho phần tử TextView

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

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

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

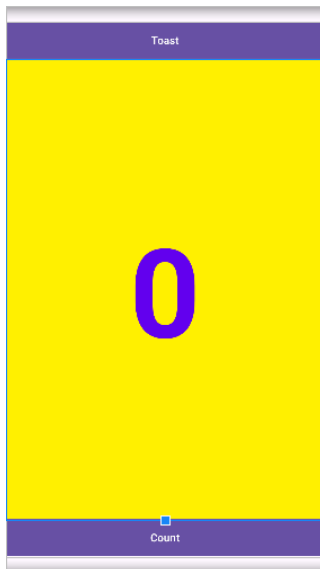
Thuộc tính **android:layout_weight** cho biết lượng không gian bổ sung trong LinearLayout sẽ được phân bổ cho View. Nếu chỉ có một View có thuộc tính này, View sẽ nhận được toàn bộ không gian màn hình bổ sung. Đối với nhiều phần tử View, không gian được tính theo tỷ lệ. Ví dụ: nếu mỗi phần tử Button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

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

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

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

Bản xem trước bây giờ như thế này:



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

Mã

Mã XML trong tệp activity_main.xml:

```

<?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
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:background="@color/design_default_color_primary"
        android:onClick="showToast"
        android:text="Toast"
        android:textColor="@android:color/white" />

    <TextView
        android:id="@+id/show_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="#FFF000"
        android:gravity="center"
        android:text="0"
        android:textAlignment="center"
        android:textColor="@color/design_default_color_primary"
        android:textSize="16sp"
        android:textStyle="bold"

```

```

        app:layout_constraintBottom_toTopOf="@+id/button_count"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button_toast"
        tools:ignore="RtlCompat" />

    <Button
        android:id="@+id/button_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:background="@color/design_default_color_primary"
        android:onClick="countUp"
        android:text="@string/button_label_count"
        android:textColor="@android:color/white" />
</LinearLayout>

```

Nhiệm vụ 3: Thay đổi bố cục thành RelativeLayout

RelativeLayout là một nhóm chế độ xem trong đó mỗi chế độ xem được định vị và căn chỉnh tương đối với các chế độ xem khác trong nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục bằng RelativeLayout

3.1. Thay LinearLayout thành RelativeLayout

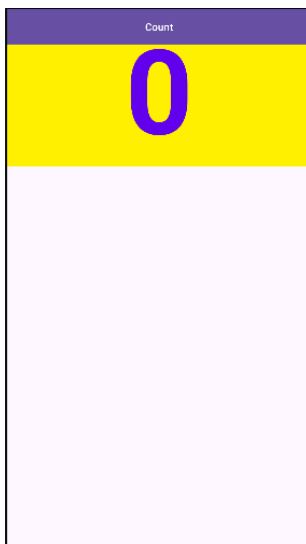
Một cách dễ dàng để thay đổi LinearLayout thành RelativeLayout là thêm các thuộc tính XML vào tab Text

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

3.2. Sắp xếp lại các View trong RelativeLayout

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

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



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

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

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

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

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

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

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

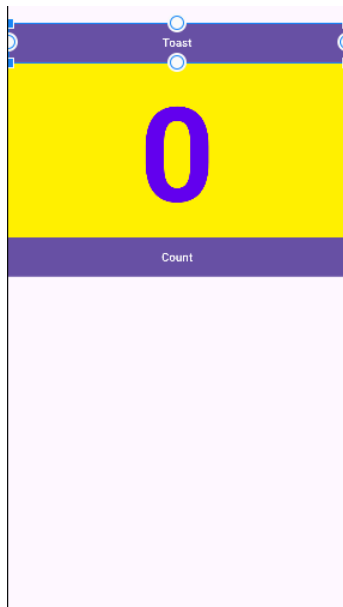
```
<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp"
    android:background="@color/design_default_color_primary"
    android:onClick="countUp"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white"
    android:layout_below="@id/show_count"
    android:layout_centerHorizontal="true"/>
```

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

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

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

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



Mã giải pháp

Mã XML trong **activity_main.xml**:

```
<?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
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:background="@color/design_default_color_primary"
        android:onClick="showToast"
        android:text="Toast"
        android:textColor="@android:color/white" />

    <TextView
        android:id="@+id/show_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FFF000"
        android:gravity="center"
        android:text="0"
        android:textAlignment="center"
        android:textColor="@color/design_default_color_primary"
        android:textSize="160sp"
        android:textStyle="bold">
```

```


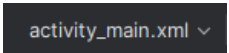
app:layout_constraintBottom_toTopOf="@+id/button_count"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/button_toast"
tools:ignore="RtlCompat"
android:layout_below="@id/button_toast"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true" />

<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp"
    android:background="@color/design_default_color_primary"
    android:onClick="countUp"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white"
    android:layout_below="@id/show_count"
    android:layout_centerHorizontal="true"/>
</RelativeLayout>

```

Tóm tắt

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

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

Sử dụng ConstraintLayout:

- Để xóa tất cả các ràng buộc trong bố cục với gốc ConstraintLayout, hãy nhấp vào nút **Clear All Constraints** trên thanh công cụ

- Bạn có thể căn chỉnh một phần tử UI chứa văn bản, chẳng hạn như TextView hoặc Button, với một phần tử UI khác chứa văn bản. Một ràng buộc đường cơ sở cho phép bạn ràng buộc các phần tử sao cho các đường cơ sở văn bản khớp nhau
- Để tạo ràng buộc đường cơ sở, hãy di con trỏ lên phần tử UI cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.

Sử dụng LinearLayout:

- LinearLayout là một ViewGroup sắp xếp bộ sưu tập các view của nó theo một hàng ngang hoặc dọc
- LinearLayout phải có thuộc tính **layout_width**, **layout_height** và **orientation**
- Thuộc tính **match_parent** cho **layout_width** hoặc **layout_height**: Mở rộng View để lấp đầy phần tử cha của nó theo chiều rộng hoặc chiều cao. Khi LinearLayout là View gốc, nó sẽ mở rộng đến kích thước của màn hình (View cha)
- Thuộc tính **wrap_content** cho **layout_width** hoặc **layout_height**: Thu nhỏ kích thước để View vừa đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, View sẽ trở nên vô hình
- Số dp cố định cho **layout_width** hoặc **layout_height**: Chỉ định kích thước cố định, được điều chỉnh cho mật độ màn hình của thiết bị. Ví dụ: 16dp có nghĩa là 16 pixel độc lập với mật độ
- Thuộc tính **orientation** cho LinearLayout có thể là **horizontal** để sắp xếp các phần tử từ trái sang phải, hoặc **vertical** để sắp xếp các phần tử từ trên xuống dưới
- Chỉ định các thuộc tính **gravity** và **weight** cho phép bạn kiểm soát thêm việc sắp xếp các view và nội dung trong LinearLayout
- Thuộc tính android:gravity chỉ định sự căn chỉnh nội dung của View bên trong chính View đó
- Thuộc tính **android:layout_weight** chỉ ra không gian thừa trong LinearLayout sẽ được phân bổ cho View. Nếu chỉ một View có thuộc tính này, nó sẽ nhận được tất cả không gian màn hình thừa. Đối với nhiều phần tử View, không gian được chia tỷ lệ. Ví dụ: nếu hai phần tử Button mỗi phần tử có trọng số là 1 và TextView là 2, tổng cộng là 4, thì các phần tử Button mỗi phần tử sẽ nhận được $\frac{1}{4}$ không gian và TextView một nửa.

Sử dụng RelativeLayout:

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

1.4) Văn bản và các chế độ cuộn

Giới thiệu

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

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

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

Với **ScrollView**, tất cả các phần tử UI đều ở trong bộ nhớ và trong cấu trúc view ngay cả khi chúng không được hiển thị trên màn hình. Điều này làm cho ScrollView trở nên lý tưởng để cuộn các trang văn bản tự do một cách mượt mà, bởi vì văn bản đã ở trong bộ nhớ. Tuy nhiên, ScrollView có thể sử dụng nhiều bộ nhớ, điều này có thể ảnh hưởng đến hiệu suất của phần còn lại của ứng dụng của bạn. Để hiển thị danh sách dài các mục mà người dùng có thể thêm, xóa hoặc chỉnh sửa, hãy cân nhắc sử dụng RecyclerView, được mô tả trong một bài học riêng biệt

Những gì bạn nên biết

Bạn cần có khả năng:

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

Những gì bạn sẽ học

- Cách sử dụng mã XML để thêm nhiều phần tử TextView
- Cách sử dụng mã XML để xác định một View cuộn
- Cách hiển thị văn bản tự do với một số thẻ định dạng HTML
- Cách tạo kiểu cho màu nền và màu văn bản của TextView
- Cách đưa một liên kết web vào văn bản

Những gì bạn sẽ làm

- Tạo ứng dụng ScrollingText
- Thay đổi ViewGroup ConstraintLayout thành RelativeLayout
- Thêm hai phần tử TextView cho tiêu đề và phụ đề của bài viết
- Sử dụng kiểu và màu TextAppearance cho tiêu đề và phụ đề của bài viết
- Sử dụng thẻ HTML trong chuỗi văn bản để kiểm soát định dạng
- Sử dụng thuộc tính lineSpacingExtra để thêm khoảng cách dòng để dễ đọc
- Thêm ScrollView vào bố cục để cho phép cuộn một phần tử TextView
- Thêm thuộc tính autoLink để kích hoạt các URL trong văn bản và có thể nhấp vào

Tổng quan ứng dụng

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

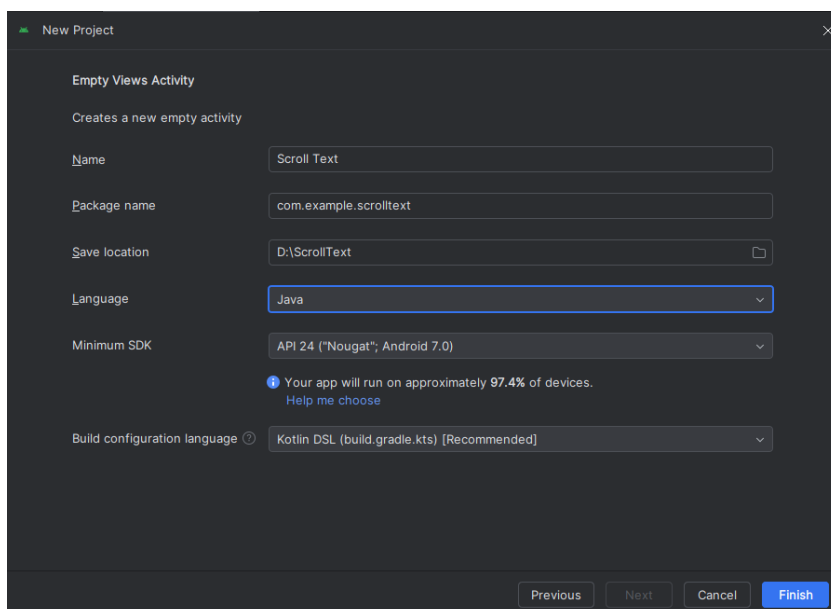
Nhiệm vụ 1: Thêm và chỉnh sửa các phần tử TextView

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

1.1. Tạo dự án và các phần tử TextView

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

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



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

Cấu trúc View là ViewGroup ConstraintLayout:

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

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

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

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

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

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

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

Đoạn mã XML ở trên cùng bây giờ như sau:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

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

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/article_heading"
    android:background="@color/design_default_color_primary"
    android:textColor="@color/white"
    android:padding="10dp"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
    android:textStyle="bold"
    android:text="Article Title"/>
```

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

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

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

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

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

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/article_subheading"
    android:layout_below="@id/article_heading"
    android:padding="@dimen/padding_regular"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault"
    android:text="Article Subtitle"/>
```

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

- Trích xuất tài nguyên chuỗi cho chuỗi mã cứng của thuộc tính **android:text "Article Subtitle"** trong TextView thành **article_subtitle**
- Trong phần tử TextView "Hello World", xóa các thuộc tính **layout_constraint**:

```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

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

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Article Text"
    android:id="@+id/article"
    android:layout_below="@id/article_subheading"
    android:lineSpacingExtra="5sp"
    android:padding="@dimen/padding_regular" />
```

13. Trích xuất tài nguyên chuỗi cho "**Article text**" thành **article_text** và trích xuất tài nguyên kích thước cho "**5sp**" thành **line_spacing**
14. Định dạng lại và căn chỉnh mã bằng cách nhấn tổ hợp **Shift+Alt+F**.

1.2 Thêm văn bản cho bài báo

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

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

1. Trong thư mục **app > res > values**, mở **strings.xml**
2. Mở bất kỳ tệp văn bản nào có một lượng lớn văn bản
3. Nhập các giá trị cho các chuỗi **article_title** và **article_subtitle** với tiêu đề và phụ đề do bạn tự tạo. Hãy tạo các giá trị chuỗi thành văn bản một dòng mà không có thẻ HTML hoặc nhiều dòng
4. Nhập hoặc sao chép và dán văn bản cho chuỗi **article_text**

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

Hãy nhớ những điều sau:

- Khi bạn nhập hoặc dán văn bản trong tệp **strings.xml**, các dòng văn bản không tự động xuống dòng mà kéo dài ra ngoài lề phải. Đây là hành vi chính xác - mỗi dòng văn bản mới bắt đầu từ lề trái đại diện cho một đoạn văn hoàn chỉnh. Nếu bạn muốn văn bản trong **strings.xml** được xuống dòng,

bạn có thể nhấn phím **Enter** để nhập các ký tự kết thúc dòng cứng hoặc định dạng văn bản trước trong một trình soạn thảo văn bản với các ký tự kết thúc dòng cứng

- Nhập `\n` để biểu thị phần cuối của một dòng và một `\n` khác để biểu thị một dòng trống. Bạn cần thêm các ký tự kết thúc dòng để ngăn các đoạn văn chạy vào nhau
- Nếu bạn có dấu nháy đơn (') trong văn bản, bạn phải thoát nó bằng cách đặt một dấu gạch chéo ngược (\') trước nó. Nếu bạn có dấu ngoặc kép trong văn bản của mình, bạn cũng phải thoát nó ("). Bạn cũng phải thoát bất kỳ ký tự không phải ASCII nào khác
- Nhập các thẻ HTML `` và `` xung quanh các từ cần được in đậm
- Nhập các thẻ HTML `<i>` và `</i>` xung quanh các từ cần được in nghiêng. Nếu bạn sử dụng dấu nháy đơn uốn cong trong một cụm từ in nghiêng, hãy thay thế chúng bằng dấu nháy đơn thẳng
- Bạn có thể kết hợp in đậm và in nghiêng bằng cách kết hợp các thẻ, như trong `<i>... words...</i>`. Các thẻ HTML khác bị bỏ qua
- Bao quanh toàn bộ văn bản bên trong `<string name="article_text"></string>` trong tệp `strings.xml`
- Bao gồm một liên kết web để kiểm tra, chẳng hạn như `www.google.com`. (Ví dụ: `https://cafef.vn`) Không sử dụng thẻ HTML, vì mọi thẻ HTML ngoại trừ thẻ in đậm và in nghiêng đều bị bỏ qua và được trình bày dưới dạng văn bản, điều này không phải là điều bạn muốn

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