

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	2
Bài 1) Tạo ứng dụng đầu tiên	2
1.1) Android Studio và Hello World	2
1.2) Giao diện người dùng tương tác đầu tiên	29
1.3) Trình chỉnh sửa bố cục	57
1.4) Văn bản và các chế độ cuộn	83
1.5) Tài nguyên có sẵn	104
Bài 2) Activities	104
2.1) Activity và Intent	104
2.2) Vòng đời của Activity và trạng thái	104
2.3) Intent ngầm định	104
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	104
3.1) Trình gỡ lỗi	104
3.2) Kiểm thử đơn vị	104
3.3) Thư viện hỗ trợ	104

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 bạn cài đặt thành công **Android Studio**, bạn sẽ tạo dự án mới cho ứng dụng Hello World từ một mẫu. Ứng dụng đơn giản này hiển thị chuỗi “Hello World” trên màn hình của thiết bị Android ảo hoặc thiết bị vật lý,
- Đây là giao diện của ứng dụng sau khi hoàn thành:

Task1: Cài đặt Android Studio

- **Android Studio** cung cấp một môi trường phát triển tích hợp (IDE) hoàn chỉnh bao gồm trình soạn thảo mã nâng cao và một bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ hỗ trợ phát triển, gỡ lỗi, kiểm thử và tối ưu hiệu suất, giúp việc phát triển ứng dụng trở nên nhanh chóng và dễ dàng hơn. Bạn có thể kiểm thử ứng dụng của mình trên nhiều trình giả lập được cấu hình sẵn hoặc trên thiết bị di động của chính mình, tạo ứng dụng chính thức và phát hành 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 các máy tính chạy Windows hoặc Linux và máy MAC chạy macOS. Phiên bản OpenJDK (Java Development Kit) mới nhất được tích hợp sẵn với Android Studio.
- Để thiết lập và chạy Android Studio, trước tiên hãy kiểm tra yêu cầu hệ thống để đảm bảo hệ thống bạn đáp ứng được. Quá trình cài đặt tương tự trên tất cả các nền tảng. Mọi điểm khác biệt sẽ được ghi chú bên dưới.
 1. Truy cập trang web dành cho nhà phát triển Android và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
 2. Chấp nhận cấu hình đặc biệt cho tất cả các bước và đảm bảo rằng tất cả các thành phần đều được chọn để cài đặt.
 3. Sau khi hoàn tất cài đặt, trình hướng dẫn thiết lập sẽ tải xuống và cài đặt một số thành phần bổ sung bao gồm Android SDK. Hãy kiên nhẫn, quá trình này có thể mất một chút thời gian tùy thuộc vào tốc độ Internet của bạn và một số bước có thể trông có vẻ lặp lại.
 4. Khi quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng tạo dự án đầu tiên của mình.
- Khắc phục sự cố : Nếu bạn gặp vấn đề trong quá trình cài đặt , hãy kiểm tra ghi chú phát hành của Android Studio hoặc nhờ sự trợ giúp từ giảng viên của bạn.

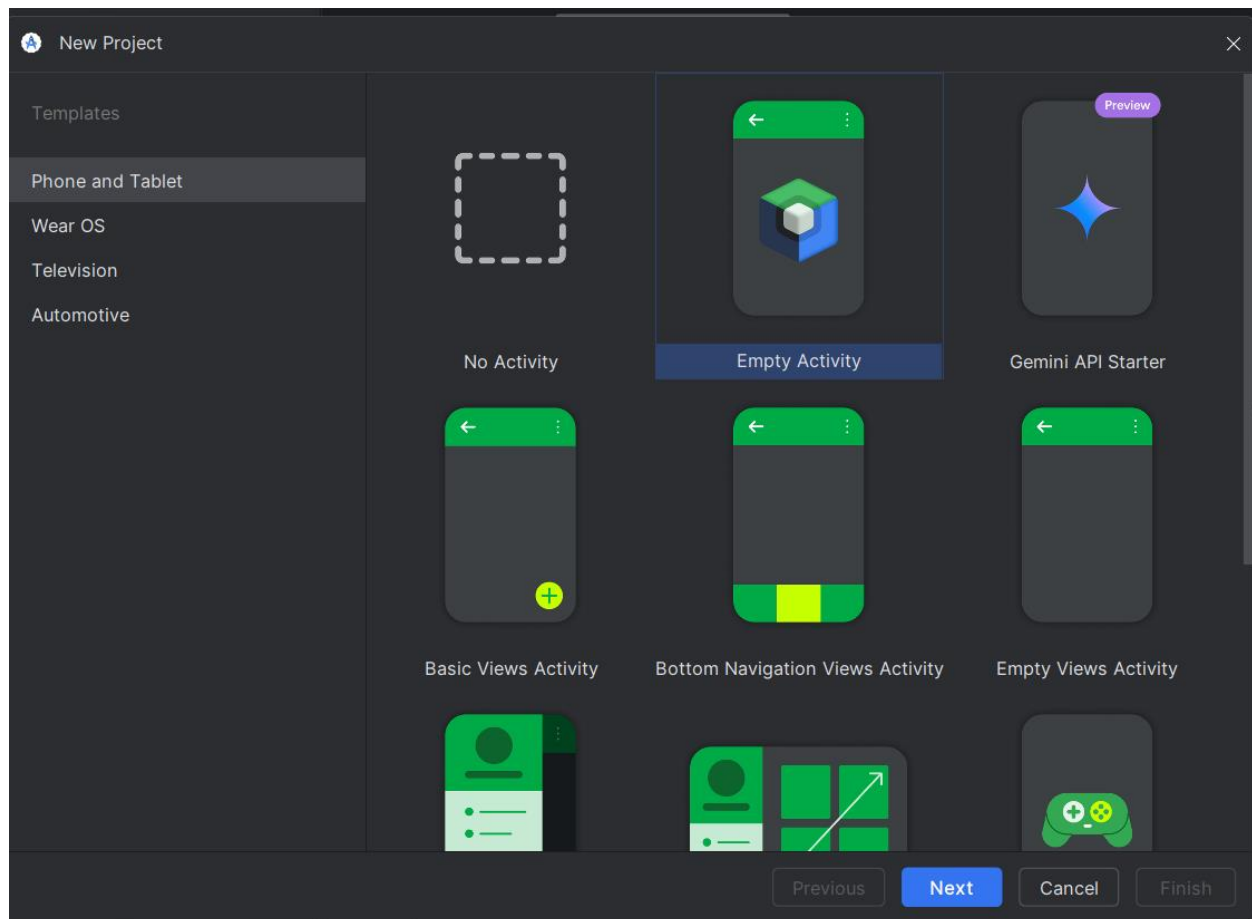
Task 2 : Tạo ứng dụng Hello World

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

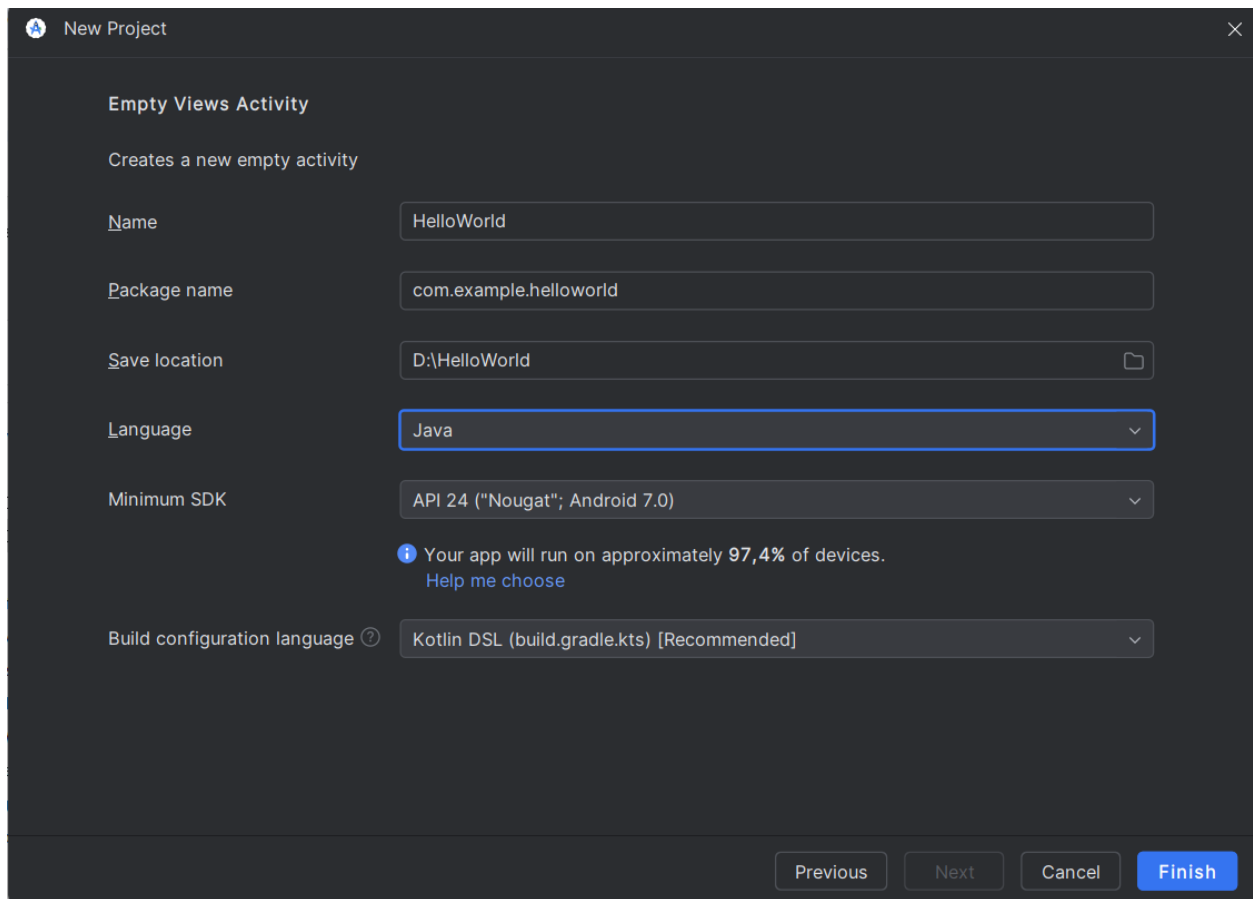
2.1 Tạo dự án ứng dụng

1. Mở Android Studio nếu nó chưa được mở
2. Trong cửa sổ Welcome to Android Studio , nhấn vào Start a new Android Studio Project.
3. Trong cửa sổ Create Android Studio, nhập Hello World cho Application name.
4. Xác minh rằng Project location mặc định là nơi bạn muốn lưu trữ ứng dụng Hello World và các dự án Android Studio khác hoặc thay đổi nó sang thư mục mà bạn ưu tiên.
5. Chấp nhận android.example.com mặc định cho Company Domain, hoặc tạo một tên miền công ty riêng.
Nếu bạn không có kế hoạch phát hành ứng dụng của mình, bạn có thể giữ nguyên mặc định. Lưu ý rằng việc thay đổi tên gói của ứng dụng sau này sẽ tốn thêm công sức.
6. Để nguyên không chọn các tùy chọn Include C++ support và Include Kotlin support và nhấn Next
7. Trên màn hình Target Android Devices, đảm bảo rằng Phone and Table được chọn. Đảm bảo API 15 : Android 4.0.3 IceCreamSandwich được làm bằng MiniumSDK; Nếu không, hãy sử dụng popup menu để thiết lập nó.
Đây là các cài đặt được sử dụng cho các ví dụ trong các bài học của khóa học này. Tính tới thời điểm hiện tại, các cài đặt này giúp ứng dụng Hello World của bạn tương thích với 97% thiết bị Android đang hoạt động trên Google Play Store.
8. Để nguyên không chọn Include Instant App support và tất cả các tùy chọn khác. Sau đó, nhấn Next. Nếu dự án của bạn yêu cầu các thành phần bổ sung cho target SDK đã chọn, Android Studio sẽ tự động cài đặt chúng.
9. Cửa sổ Add an Activity xuất hiện. Activity là một thành phần quan trọng trong bất kỳ ứng dụng Android nào, đại diện cho một tác vụ đơn lẻ mà người dùng có thể thực hiện. Thông thường một Activity sẽ đi kèm với một bố cục(layout) xác định cách các thành phần giao diện người dùng (UI) hiện thị trên màn hình. Android Studio cung cấp các mẫu Activity để

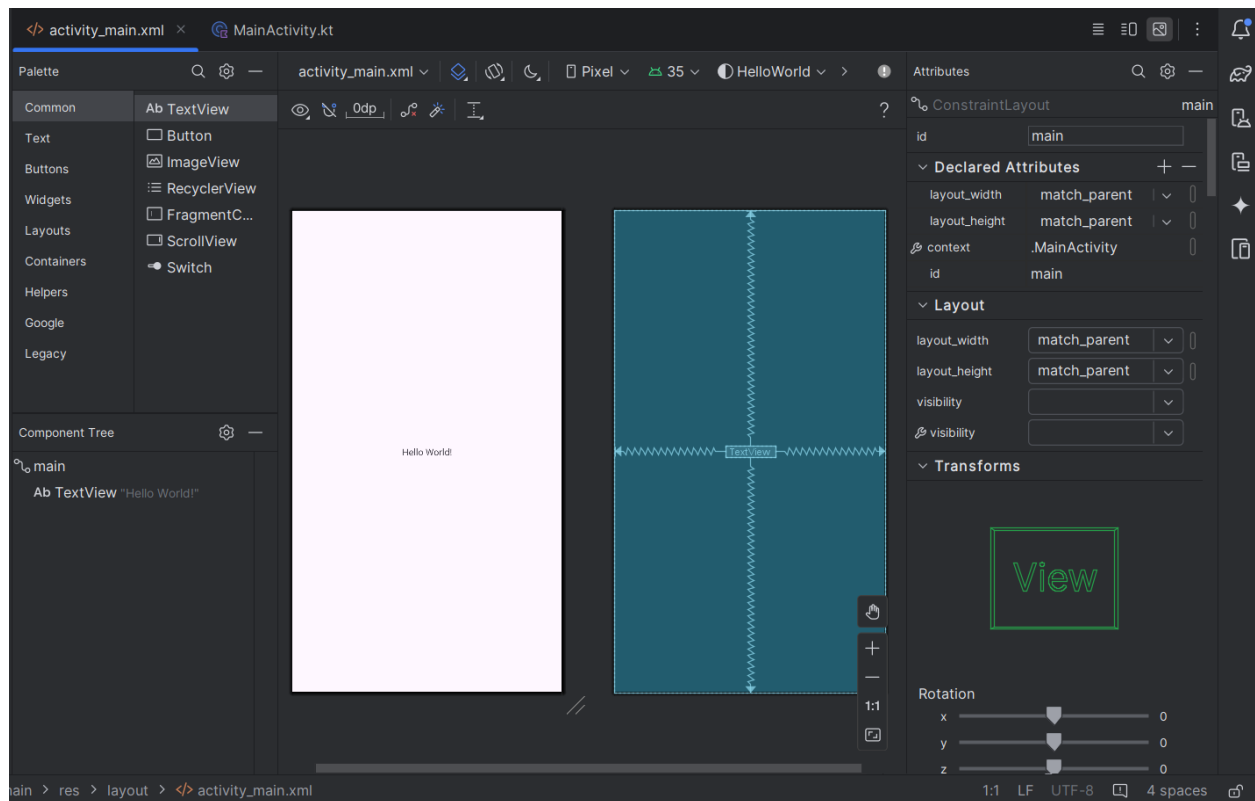
giúp bạn bắt đầu. Đối với dự án Hello World, hãy chọn Empty Activity như bên hình dưới và nhấn Next.



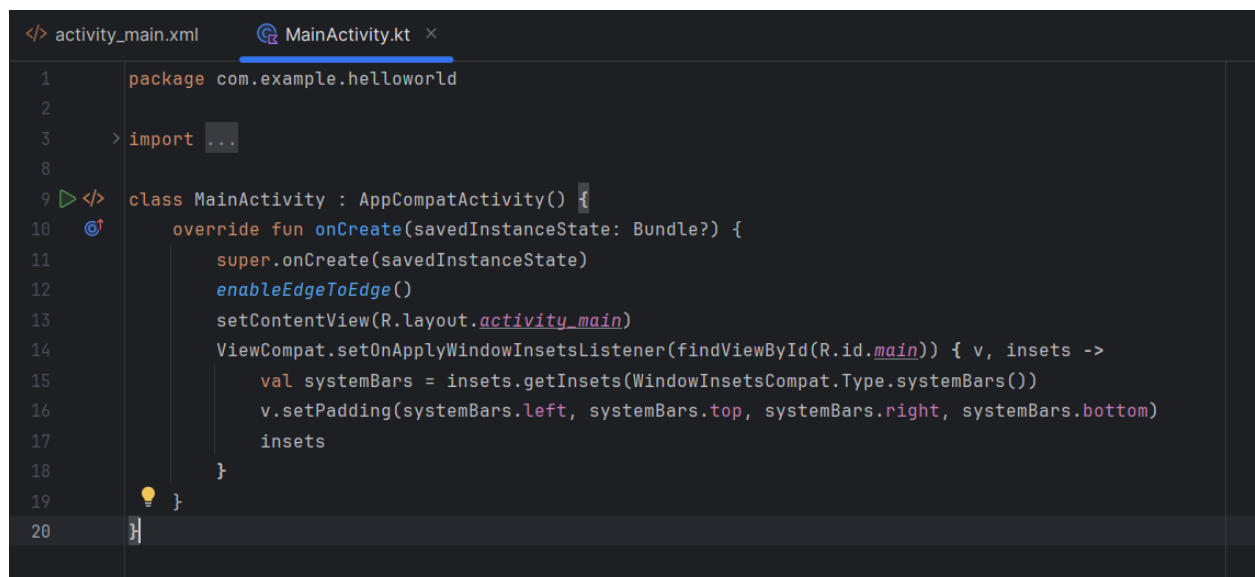
10. Màn hình Configure Activity xuất hiện (giao diện này có thể khác nhau tùy thuộc vào mẫu mà bạn đã chọn ở bước trước). Theo mặc định, Empty Activity do mẫu cung cấp được đặt tên là MainActivity. Bạn có thể thay đổi tên này nếu muốn, nhưng trong bài học này chúng ta sẽ sử dụng MainActivity.
11. Đảm bảo rằng tùy chọn Generate Layout file được chọn. Tên bố cục mặc định là activity_main. Bạn có thể thay đổi tên này nếu muốn, nhưng trong bài học này chúng ta sẽ sử dụng activity_main.
12. Đảm bảo rằng tùy chọn Backwards Compatibility(App Compat) được chọn. Tùy chọn này giúp ứng dụng của bạn tương thích ngược với các phiên bản Android trước đó.
13. Nhấn Finish.



- Android Studio sẽ tạo một thư mục cho các dự án của bạn và xây dựng dự án bằng Gradle (quá trình này có thể mất vài phút)
- Mẹo : Tham khảo trang dành cho nhà phát triển Configure your build để biết thông tin chi tiết.
- Bạn cũng có thể thấy thông báo “Tip of the day” với các phím tắt và mẹo hữu ích khác. Nhấp Close để tắt thông báo này
- Trình soạn thảo của Android Studio xuất hiện. Hãy làm theo các bước sau :
 1. Nhấp vào activiti_main.xml để xem trình chỉnh sửa bố cục.
 2. Nhấp vào tab Design trong trình chỉnh sửa bố cục (nếu chưa được chọn) để hiển thị bản xem trước giao diện đồ họa của bố cục như hình dưới đây.



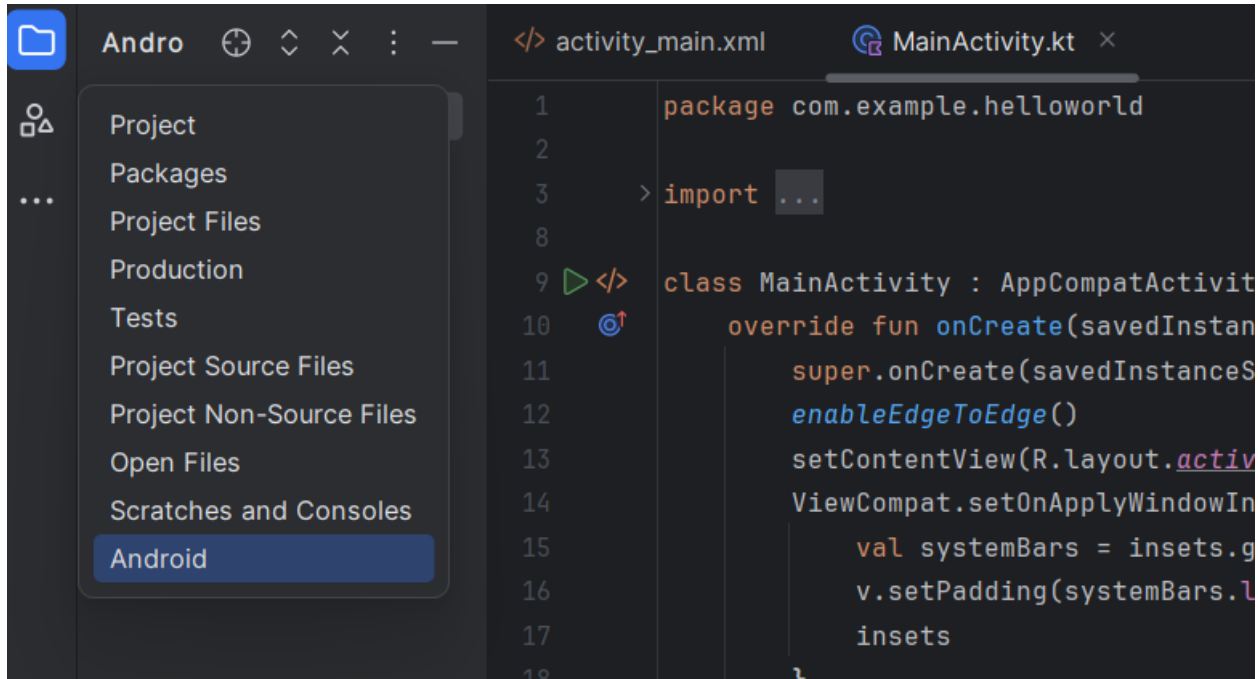
3. Nhấp vào tab MainActivity.java để xem trình chỉnh sửa mã nguồn như hình dưới



2.2 Khám phá Project > Android pane

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

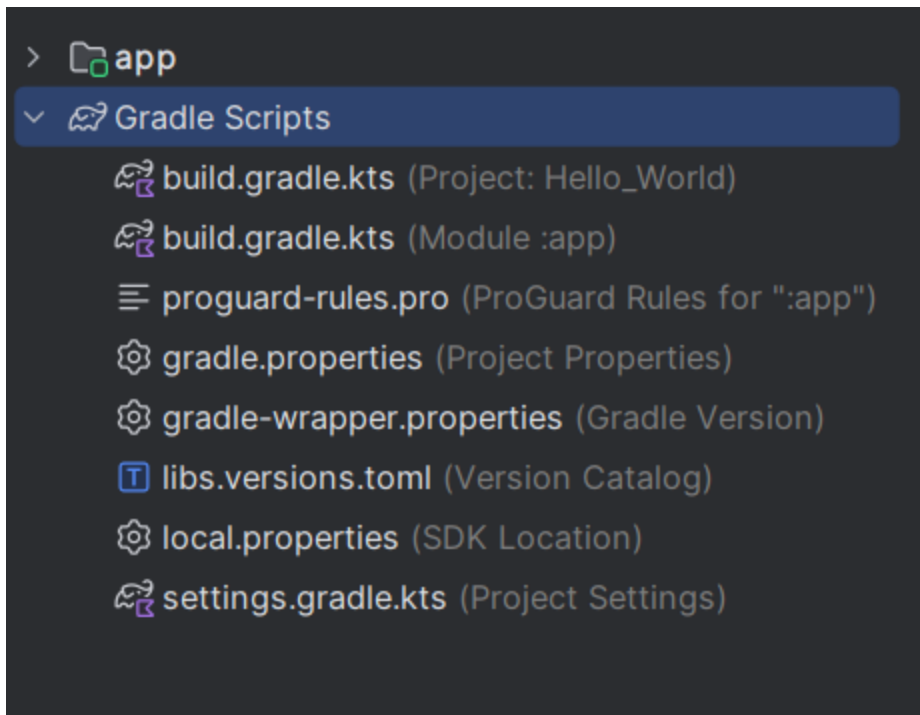
1. Nếu chưa được chọn, hãy nhấp vào tab Project trong cột tab dọc ở phía bên trái cửa sổ Android Studio. Project pane sẽ xuất hiện.
2. Để xem Project theo tiêu chuẩn của Android Project , hãy chọn Android từ popup menu ở đầu Project pane, như hình bên dưới



Lưu ý : Chương này và các chương khác đề cập đến Project khi được đặt thành là Android là Project -> Android pane

2.3 Khám phá thư mục Gradle Scripts

- Hệ thống dựng Gradle trong Android Studio giúp bạn dễ dàng thêm các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác vào bản dựng dưới dạng phụ thuộc.
- Khi bạn lần đầu tạo một dự án ứng dụng, Project -> Android pane sẽ hiện với thư mục Gradle Scripts được mở rộng như hình bên dưới đây



- Thực hiện các bước sau để khám phá hệ thống Gradle
 1. Nếu thư mục Gradle Scripts chưa được mở rộng, hãy nhấp vào biểu tượng tam giác để mở rộng nó.

Thư mục này chứa tất cả các tệp cần thiết cho hệ thống dựng.
 2. Tìm tệp build.gradle (Project: HelloWorld).
 - Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô -đun trong Project của mình. Mỗi Project trong Android Studio đều chứa một tệp Gradle build cấp cao nhất. Hầu hết thời gian, bạn không cần chỉnh sửa tệp này, nhưng việc hiểu nội dung của nó vẫn rất hữu ích.
 - Theo mặc định, tệp build cấp cao nhất sử dụng khối buildscripts để xác định các kho lưu trữ Gradle và các phần phụ thuộc chung cho tất cả các mô-đun trong Project. Khi phần phụ thuộc của bạn không phải là thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm các tệp trong các kho lưu trữ trực tuyến được chỉ định trong khối repositories của tệp này. Theo mặc định, các Project mới trong Android Studio khai báo JCenter và Google (bao gồm Google Maven repository) làm vị trí kho lưu trữ:

```
> activity_main.xml    MainActivity.kt    build.gradle.kts (Hello World) ×    build.gradle.kts (:app)
1 // Top-level build file where you can add configuration options common to all sub-projects/modules.
2 plugins {
3     alias(libs.plugins.android.application) apply false
4     alias(libs.plugins.kotlin.android) apply false
5 }
```

3. Tìm tệp build.gradle(Module:app).

- Ngoài tệp build.gradle cấp dự án, mỗi mô-đun đều có tệp build.gradle riêng, tệp này cho phép bạn cấu hình cài đặt build riêng cho từng mô-đun cụ thể (ứng dụng HelloWorld chỉ có một mô-đun). Việc cấu hình các cài đặt bản dựng cho phép bạn cung cấp các tùy chọn gói tùy chỉnh, chẳng hạn như các kiểu build và product flavors bổ sung. Bạn cũng có thể ghi đè các cài đặt trong tệp AndroidManifest.xml hoặc tệp build.gradle cấp cao nhất.
- Đây là tệp thường được chỉnh sửa nhất khi thay đổi cấu hình cấp ứng dụng, chẳng hạn như khai báo các phần phụ thuộc trong phần dependencies. Bạn có thể khai báo một phụ thuộc thư viện bằng một trong số các cấu hình phụ thuộc khác nhau. Mỗi cấu hình phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện. Ví dụ, câu lệnh implementation fileTree(dir: 'libs', include: ['*.jar']) thêm một phụ thuộc cho tất cả các tệp ".jar" bên trong thư mục libs.
- Dưới đây là tệp build.gradle(Module:app) dành cho ứng dụng HelloWorld:

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

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

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

        testInstrumentationRunner =
```

```

"androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-
optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = "11"
    }
}

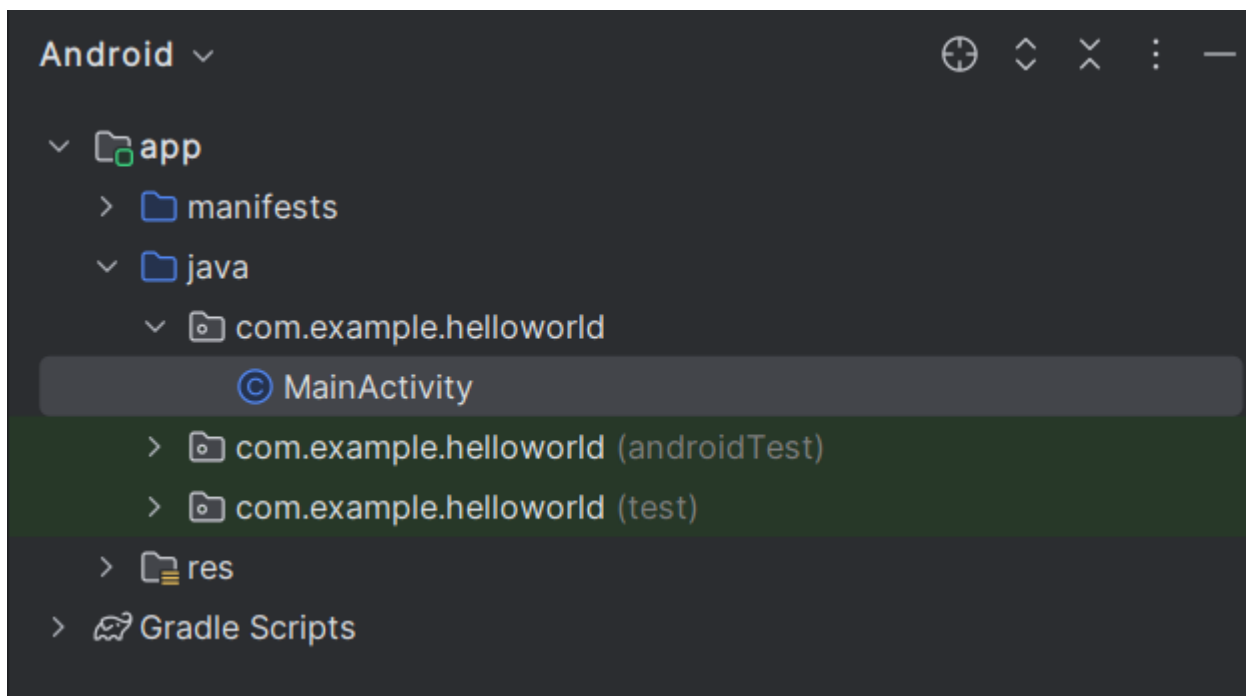
dependencies {
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
}

```

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

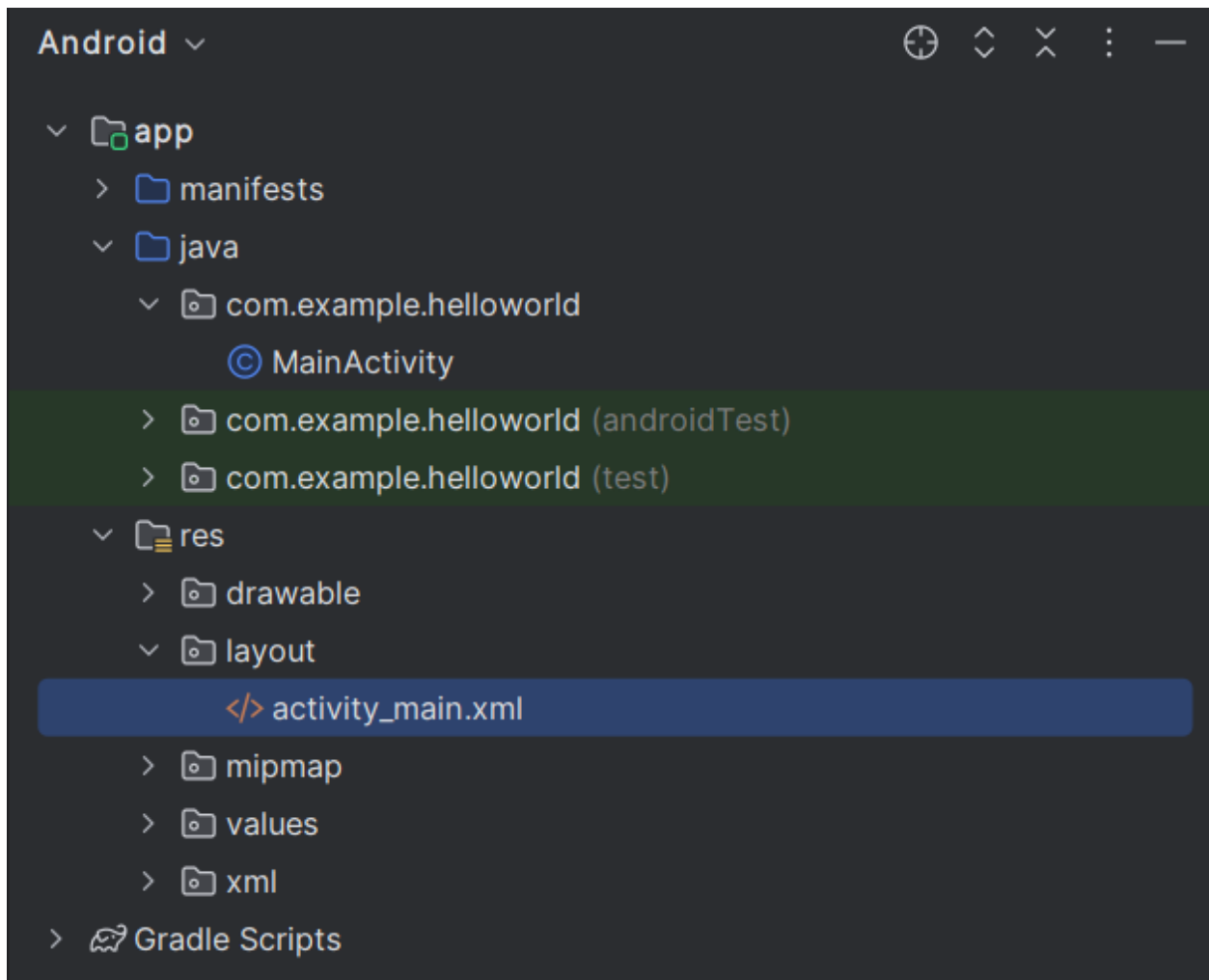
2.4 Khám phá ứng dụng và thư mục res

- Tất cả mã nguồn và tài nguyên cho ứng dụng đều nằm trong các thư mục app và res
 1. Mở rộng thư mục app, thư mục java và thư mục com.example.android.helloworld để xem tệp MainActivity. Nhấp đúp vào tệp để mở nó trong trình soạn thảo mã.



Thư mục Java bao gồm các tệp lớp Java trong ba thư mục con, như hình minh họa trên. Thư mục `com.example.android.helloworld` (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp trong một gói ứng dụng. Hai thư mục còn lại được sử dụng cho việc kiểm thử và sẽ được giải thích trong một bài học khác. Đối với ứng dụng Hello World, chỉ có một gói và nó chứa tệp `MainActivity.java`. `MainActivity` là tên thông dụng cho Activity đầu tiên mà người dùng nhìn thấy (trong Project->Android pane phần mở rộng tệp thường bị ẩn đi).

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



- Thư mục res chứa các tài nguyên như layouts, strings và images. Một Activity thường được liên kết với một bố cục giao diện người dùng được định nghĩa dưới tệp XML. Tệp này thường được đặt tên theo tên của Activity tương ứng.

2.5 Khám phá thư mục manifests

- Thư mục manifests chứa các tệp cung cấp thông tin quan trọng về ứng dụng của bạn cho hệ thống Android, hệ thống bắt buộc phải có những thông tin này trước khi có thể chạy bất kì mã nào của ứng dụng.
 1. Mở rộng thư mục **manifests**.
 2. Mở tệp **AndroidManifest.xml**.
- Tệp **AndroidManifest.xml** mô tả tất cả các thành phần của ứng dụng Android của bạn. Mọi thành phần trong ứng dụng, chẳng hạn như mỗi **Activity**, đều phải được khai báo trong tệp **XML** này. Trong các bài học khác, bạn sẽ chỉnh sửa tệp

này để thêm các tính năng và quyền cho ứng dụng. Để tìm hiểu tổng quan, hãy xem **App Manifest Overview**.


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

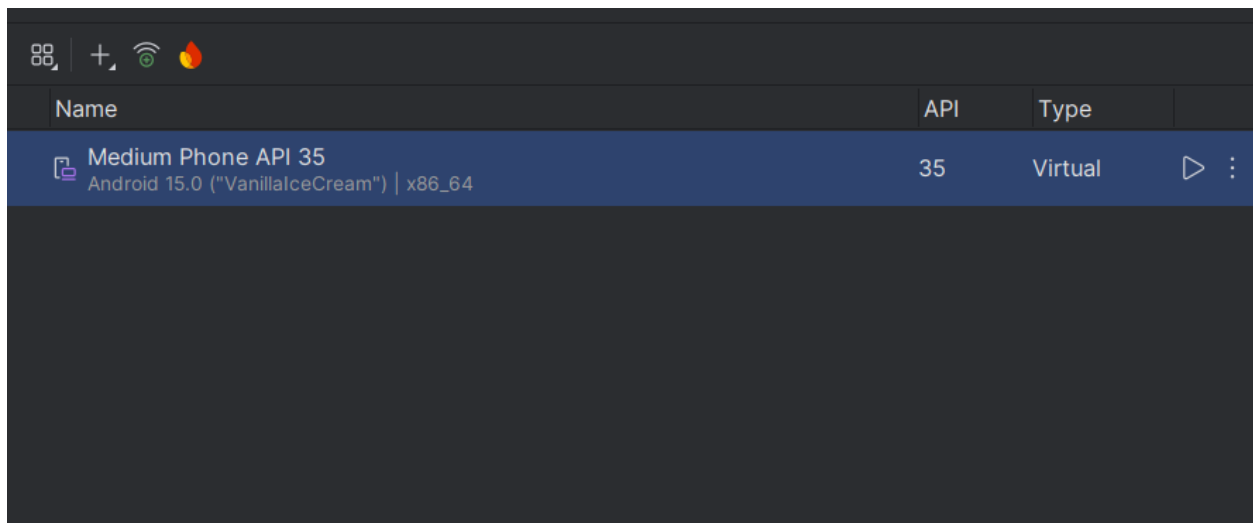
- Trong nhiệm vụ này, bạn sẽ sử dụng Android Virtual Device(ADV) manager để tạo một thiết bị ảo (còn gọi là **trình giả lập**) mô phỏng cấu hình của một loại thiết bị Android cụ thể, sau đó sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý trình giả lập Android yêu cầu thêm một số yêu cầu hệ thống ngoài các yêu cầu cơ bản cho Android Studio.
- Khi sử dụng AVD Manager bạn có thể xác định các đặc điểm phần cứng của thiết bị, chọn mức độ API, bộ nhớ , giao diện và các thuộc tính khác và lưu cấu hình đó dưới dạng một thiết ảo. Với các thiết bị ảo, bạn có thể kiểm tra ứng dụng trên các cấu hình thiết bị khác nhau (như máy tính bảng và điện thoại) với các cấp độ API khác nhau mà không cần sử dụng các thiết bị vật lý.

3.1 Tạo Android virtual device (ADV)

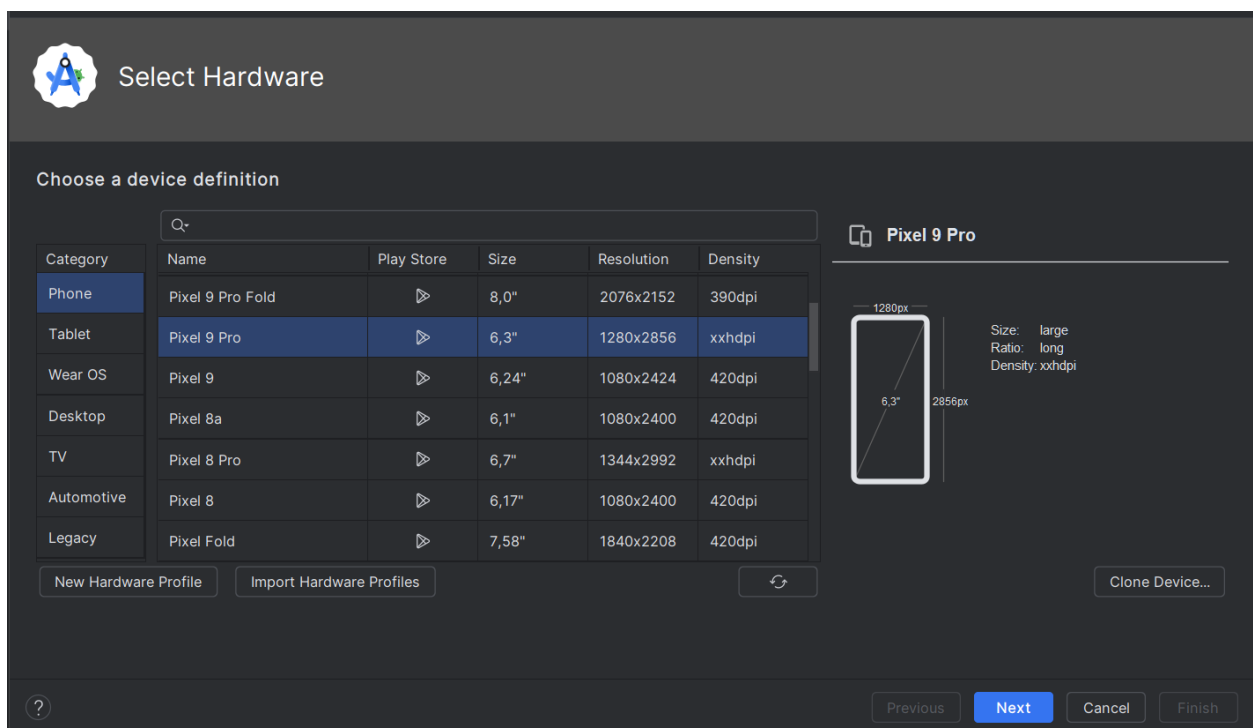
Để chạy trình giả lập trên máy tính của bạn, bạn cần tạo một cấu hình mô tả thiết bị ảo đó.

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

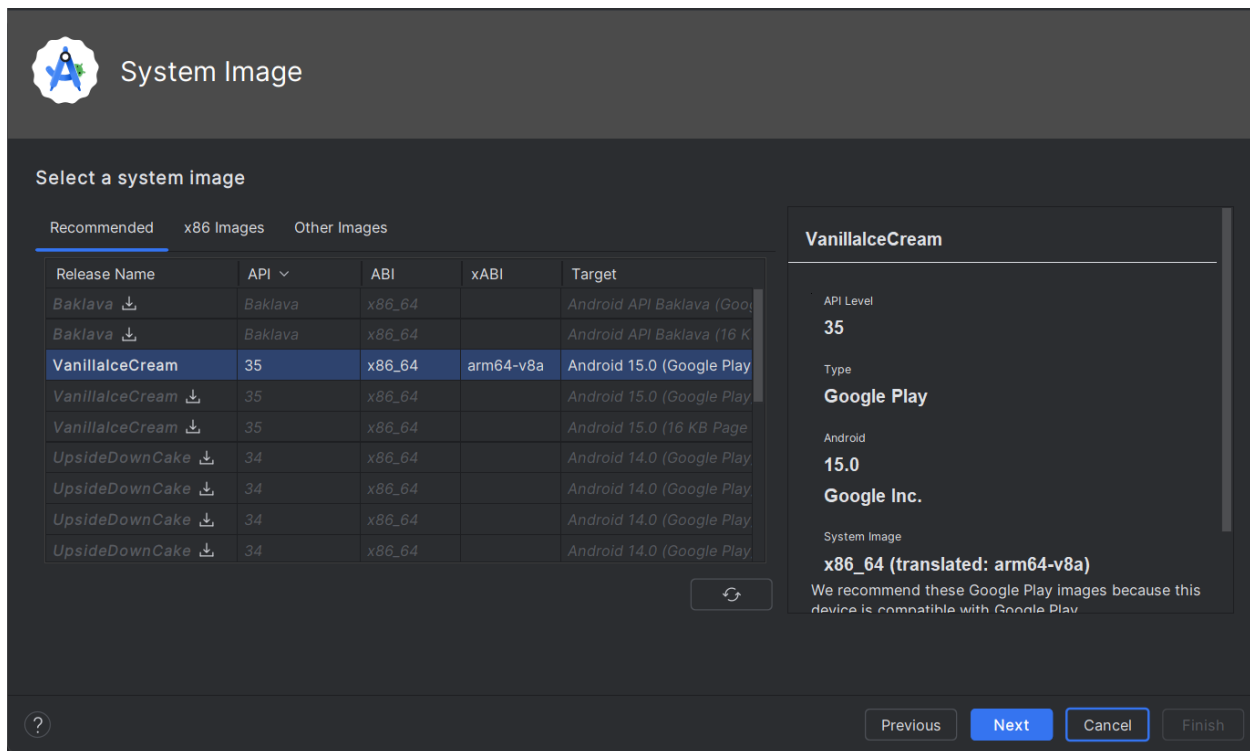
vào biểu tượng **AVD Manager**  trên thanh công cụ. Màn hình **Your Virtual Devices** xuất hiện. Nếu bạn đã tạo các thiết bị ảo trước đó, màn hình sẽ hiển thị chúng (như hình minh họa bên dưới); nếu chưa có thiết bị ảo nào, danh sách sẽ trống.



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 sẽ cung cấp các cột tương ứng với kích thước đường chéo màn hình, độ phân giải màn hình tính bằng pixel và mật độ điểm ảnh.



3. Chọn một thiết bị như Pixel 9 Pro hoặc Pixel 9 và nhấp vào Next. Màn hình System Image xuất hiện.
4. Nhấp vào tab Recommended nếu nó chưa được chọn, rồi chọn phiên bản hệ điều hành Android mà bạn muốn chạy trên thiết bị ảo (như Oreo).




Có nhiều phiên bản khác ngoài những phiên bản hiển thị trong tab Recommended. Hãy xem các tab **x86 Images** và **Other Images** để thấy thêm.

Nếu bên cạnh hình ảnh hệ thống mà bạn muốn sử dụng xuất hiện liên kết **Download**, điều đó có nghĩa là phiên bản đó chưa được cài đặt. Nhấp vào liên kết đó để bắt đầu tải xuống và nhấp **Finish** khi quá trình tải xong.

- Sau khi chọn một hình ảnh hệ thống, nhấp vào **Next**. Cửa sổ **Android Virtual Device(AVD)** sẽ xuất hiện. Bạn cũng có thể thay đổi tên của AVD. Hãy kiểm tra cấu hình của bạn và nhấp **Finish**.

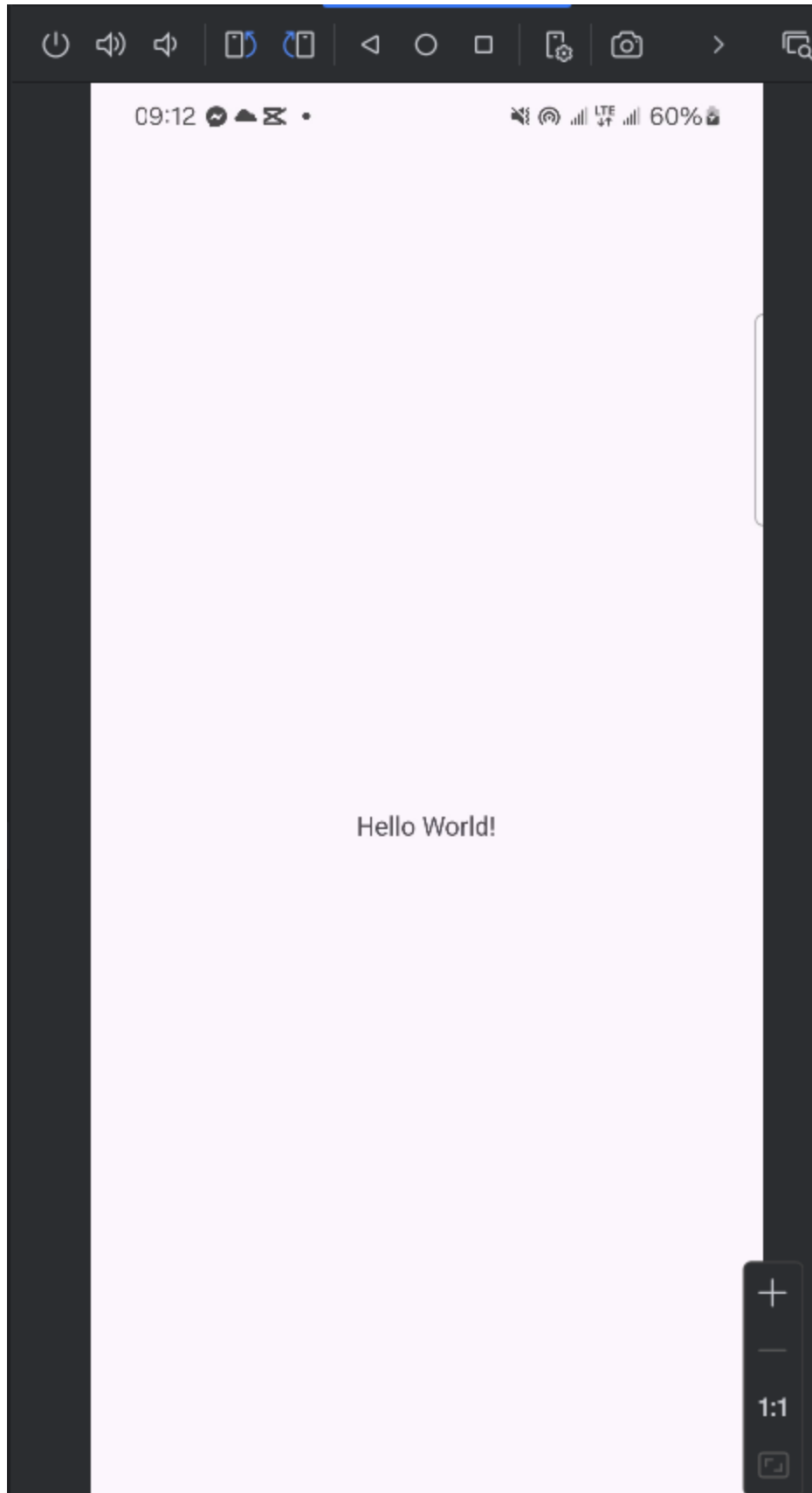
3.2 Chạy ứng dụng trên thiết bị ảo

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

- Tong **Android Studio**, chọn **Run > Run** hoặc nhấp vào biểu tượng **Run**  trên thanh công cụ.
- Trong cửa sổ **Select Deployment Target**, dưới phần **Available Virtual Devices**, chọn thiết bị ảo mà bạn vừa tạo và nhấp **OK**.

Trình giả lập sẽ khởi động và hoạt động giống như một thiết bị vật lý. Tùy thuộc vào tốc độ của máy tính, quá trình này có thể mất một chút thời gian. Ứng dụng của bạn sẽ được xây dựng 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.



Mẹo: Khi kiểm tra trên thiết bị ảo, bạn nên khởi động nó một lần ngay từ đầu phiên làm việc. Bạn không nên đóng thiết bị ảo cho đến khi hoàn tất việc kiểm tra ứng dụng, để tránh phải chờ quá trình khởi động lại thiết bị. Để đóng thiết bị ảo, hãy nhấp vào nút **X** ở đầu trình giả lập, chọn **Thoát** từ menu hoặc nhấn **Control-Q** trên Windows hoặc **Command-Q** trên macOS.

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

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

Những gì bạn cần :

- Một thiết bị Android như điện thoại hoặc máy tính bảng.
- Một cáp dữ liệu để kết nối thiết bị Android của bạn với máy tính qua cổng USB.
- Nếu bạn đang sử dụng hệ điều hành **Linux** hoặc **Windows**, bạn có thể cần thực hiện thêm một số bước để chạy ứng dụng trên thiết bị phần cứng. Hãy 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, hãy tham khảo mục OEM USB Drivers .

4.1 Bật gỡ lỗi USB (USB Debugging)

Để cho Android Studio có thể giao tiếp với thiết bị của bạn, bạn phải bật USB Debugging cho thiết bị Android của mình. Tùy chọn này được bật trong Developer options

Trên Android 4.2 trở lên, màn hình Developer options được ẩn theo mặc định. Để hiển thị Developer options và bật USB Debugging:

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

4.2 Chạy ứng dụng của bạn thiết bị

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

1. Kết nối thiết bị của bạn với máy tính phát triển bằng cáp USB
2. Nhấp vào Run trên thanh công cụ. Cửa sổ Select Deployment Target sẽ mở ra với danh sách các trình giả lập và thiết bị đã kết nối.
3. Chọn thiết bị của bạn và nhấp OK

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

Khắc phục sự cố

- Nếu Android Studio không dạng thiết bị của bạn, hãy thử các bước sau:
 1. Rút cáp và cắm lại thiết bị của bạn.
 2. Khởi động lại Android Studio.
- Nếu máy tính của bạn vẫn không tìm thấy thiết bị hoặc hiển thị là "không được ủy quyền (unauthorized)", hãy làm theo các bước sau:
 1. Rút kết nối thiết bị.
 2. Trên thiết bị, mở **Developer Options** trong **Settings app**.
 3. Nhấn **Revoke USB Debugging authorizations**.
 4. Kết nối lại thiết bị với máy tính.
 5. Khi được nhắc, hãy cấp quyền được ủy quyền.

Bạn có thể cần cài đặt USB driver phù hợp cho thiết bị của mình. Xem tài liệu **Using Hardware Devices** để biết thêm chi tiết.

TASK 5 : Thay đổi cấu hình Gradle của ứng dụng

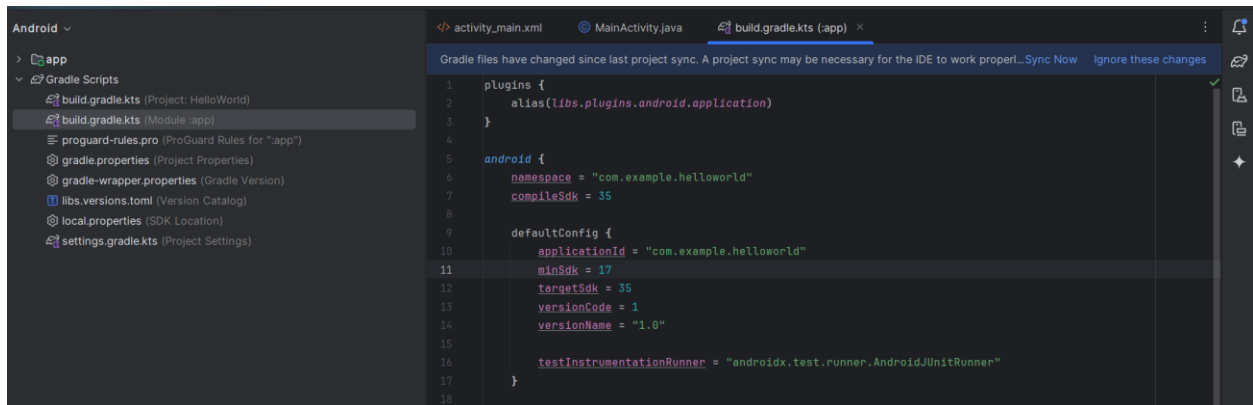
Trong nhiệm vụ này, bạn sẽ thay đổi một số cấu hình của ứng dụng trong tệp `build.gradle` (Module: app) để tìm hiểu cách thực hiện thay đổi và đồng bộ chúng với dự án Android Studio của bạn.

5.1 Thay đổi phiên bản minimum SDK cho ứng dụng

Làm theo các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu chưa mở và nhấp đúp vào tệp **build.gradle (Module: app)**.
 - Nội dung của tệp sẽ xuất hiện trong trình chỉnh sửa mã.

2. Trong khối **defaultConfig**, thay đổi giá trị của **minSdkVersion** thành **17** như hình dưới đây (giá trị ban đầu là **15**):



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

5.2 Đồng bộ cấu hình Gradle mới

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

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

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

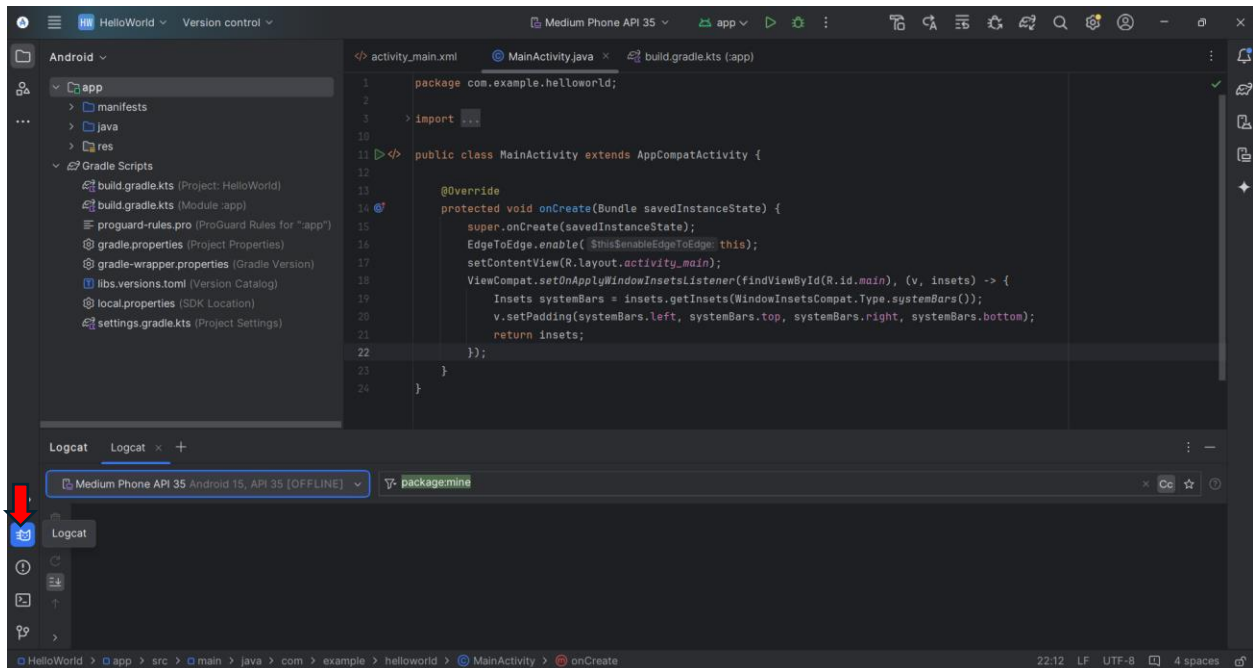
Để tìm hiểu sâu hơn về Gradle, hãy tham khảo tài liệu **Build System Overview** và **Configuring Gradle Builds**.

Task 6: Thêm các câu lệnh log vào ứng dụng của bạn

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

6.1 Xem Logcat pane

Để xem bảng Logcat, hãy nhấp vào tab Logcat ở phía dưới cửa sổ Android Studio, như hình bên dưới.



Trong hình trên :

1. **Tab Logcat** dùng để mở và đóng bảng **Logcat**, nơi hiển thị thông tin về ứng dụng của bạn khi nó đang chạy. Nếu bạn thêm các câu lệnh **Log** vào ứng dụng, các thông báo **Log** sẽ xuất hiện tại đây.
2. Menu cấp độ Log được đặt ở chế độ Verbose , hiển thị tất cả các thông báo Log. Các tùy chọn khác 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 sẽ hiện thông báo trong bảng Logcat. Ví dụ :

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

Các phần thông báo Log bao gồm :

- **Log** : Lớp Log dùng để gửi thông báo ghi log đến bảng Logcat

- d : Cấp độ log Debug, được sử dụng để lọc và hiển thị thông báo trong bảng. Các cấp độ log khác bao gồm e là lỗi nghiêm trọng, w là cảnh báo, i là thông tin chung.
- “MainActivity” : Tham số đầu tiên là tag, được sử dụng để lọc thông báo trong bảng Logcat. Thông thường, đây là tên của Activity nơi thông báo được tạo ra. Tuy nhiên bạn có thể đặt bất kỳ giá trị nào hữu ích cho việc gỡ lỗi

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

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

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

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


6. Nếu **Logcat** chưa mở, nhấp vào tab **Logcat** ở dưới cùng của **Android Studio** để mở nó.
7. Kiểm tra xem tên của mục tiêu (**target**) và tên gói (**package name**) của ứng dụng có đúng không.
8. Thay đổi mức **Log** trong khung **Logcat** thành **Debug** (hoặc để nguyên **Verbose** vì có rất ít thông điệp log).
9. Chạy ứng dụng của bạn.

Thông báo sau đây sẽ xuất hiện trong khung **Logcat**:

Thử thách lập trình

Ghi chú : Tất cả các thử thách lập trình đều là tùy chọn và không phải là điều kiện tiên quyết cho các bài học sau.

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

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

Tóm tắt

- Để cài đặt Android Studio, truy cập trang Android Studio và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo một ứng dụng mới, đảm bảo rằng API 15: Android 4.0.3 IceCreamSandwich được đặt làm Minimum SDK.
- Để xem cấu trúc thư mục của ứng dụng trong Project pane, nhấp vào tab Project trong cột tab dọc, sau đó chọn Android trong menu thả xuống ở trên cùng.
- Chỉnh sửa tệp build.gradle (Module: app) khi bạn cần thêm thư viện mới hoặc thay đổi phiên bản thư viện trong dự án.
- Tất cả mã nguồn và tài nguyên của ứng dụng nằm trong thư mục app và res. Thư mục java chứa các activity, bài kiểm tra, và các thành phần khác trong mã nguồn Java. Thư mục res chứa tài nguyên như layout, chuỗi văn bản (strings), và hình ảnh.
- Chỉnh sửa tệp AndroidManifest.xml để thêm các thành phần (components) và quyền (permissions) vào ứng dụng Android. Tất cả các thành phần như nhiều activity cần được khai báo trong tệp XML này.
- Sử dụng Android Virtual Device (AVD) Manager để tạo thiết bị ảo (emulator) và chạy ứng dụng trên đó.
- Thêm Log vào ứng dụng để hiển thị thông điệp trong Logcat, giúp gỡ lỗi (debugging) dễ dàng hơn.
- Để chạy ứng dụng trên thiết bị Android thực tế bằng Android Studio, hãy bật USB Debugging. Mở Settings > About phone và nhấn Build number 7 lần. Quay lại Settings, chọn Developer options. Bật USB Debugging.

Các khái niệm liên quan

Tài liệu về các khái niệm liên quan có trong 1.0: Introduction to Android và 1.1 Your first Android app .

Tìm hiểu thêm

Xem tài liệu chính thức về **Android Studio** tại:

- [Android Studio download page](#)
- [Android Studio release notes](#)
- [Meet Android Studio](#)
- [Logcat command-line tool](#)
- [Android Virtual Device \(AVD\) manager](#)
- [App Manifest Overview](#)
- [Configure your build](#)
- [Log class](#)
- [Create and Manage Virtual Devices](#)

Khác :

- [How do I install Java?](#)
- [Installing the JDK Software and Setting JAVA_HOME](#)
- [Gradle site](#)
- [Apache Groovy syntax](#)
- [Gradle Wikipedia page](#)

Bài tập về nhà

Xây dựng và chạy một ứng dụng

- Tạo một dự án Android mới từ mẫu Empty Template.
- Thêm các câu lệnh ghi log ở các mức khác nhau trong onCreate() của MainActivity.
- Tạo một trình giả lập (emulator) cho thiết bị, chọn bất kỳ phiên bản Android nào bạn muốn, và chạy ứng dụng.
- Sử dụng bộ lọc trong Logcat để tìm các câu lệnh log của bạn và điều chỉnh mức hiển thị chỉ để hiển thị các log ở mức Debug hoặc Error.

Trả lời các câu hỏi

Câu hỏi 1 : Tên của tệp layout cho Main Activity là gì?

- MainActivity.java
- AndroidManifest.xml

- activity_main.xml
- build.gradle

Câu hỏi 2 : Tên của resource chuỗi (string resource) xác định tên của ứng dụng là gì?

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

Câu hỏi 3 : Công cụ nào được sử dụng để tạo trình giả lập (emulator) mới?

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

Câu hỏi 4 : Giả sử ứng dụng của bạn chứa câu lệnh ghi log sau :

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

Bạn sẽ thấy thông báo "MainActivity layout is complete" trong bảng Logcat nếu mức Log level được đặt thành tùy chọn nào sau đây? (Gợi ý: Có thể có nhiều đáp án đúng.)

- Verbose
- Debug
- Info
- Warn
- Error
- Assert

Gửi ứng dụng của bạn để chấm điểm

Kiểm tra để đảm bảo ứng dụng có các yêu cầu sau:

- Một Activity hiển thị dòng chữ "Hello World" trên màn hình.
- Các câu lệnh log trong phương thức onCreate() của Activity chính.
- Mức Log trong bảng Logcat chỉ hiển thị các log debug hoặc error.

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

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

- TextView để hiển thị văn bản.
- EditText cho phép người dùng nhập và chỉnh sửa văn bản.
- Button và các phần tử 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 phần tử View khác và định vị chúng.

Lớp Java hiển thị và điều khiển giao diện người dùng (UI) được chứa trong một lớp mở rộng **Activity**. Một **Activity** thường được liên kết với một bố cục 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 **Activity** của nó và xác định cách sắp xếp các phần tử **View** trên màn hình.

Ví dụ, mã **MainActivity** trong ứng dụng Hello World hiển thị một bố cục được định nghĩa trong tệp **activity_main.xml**, trong đó có một **TextView** với nội dung "Hello World".

Trong các ứng dụng phức tạp hơn, một **Activity** có thể thực hiện 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—một ứng dụng cho phép người dùng tương tác. Bạn sẽ tạo một ứng dụng bằng cách sử dụng mẫu **Empty Activity**. Bạn cũng sẽ học cách sử dụng trình chỉnh sửa bố cục để

thiết kế giao diện, cũng như cách chỉnh sửa bố cục trong XML. Bạn cần phát triển những kỹ năng này để có thể hoàn thành các bài thực hành khác trong khóa học này.

Những kiến thức bạn nên biết trước

Bạn nên quen thuộc với:

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

Những gì bạn nên học

- Cách tạo một ứng dụng có hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế giao diện.
- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới. Hãy xem bảng thuật ngữ và khái niệm để hiểu rõ các định nghĩa một cách dễ dàng.

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 với lề và các phần tử khác.
- Thay đổi thuộc tính của các phần tử giao diện người dùng (UI elements).
- Chỉnh sửa bố cục của ứng dụng trong XML.
- Trích xuất các chuỗi mã cứng (hardcoded strings) thành tài nguyên chuỗi (string resources).
- Triển khai các phương thức xử lý sự kiện nhấp chuột (click-handler methods) 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 Button và một TextView. Khi người dùng nhấn vào Button đầu tiên, ứng dụng hiển thị một thông báo ngắn (Toast) trên màn hình. Khi nhấn vào Button thứ hai, ứng dụng tăng giá trị của bộ đếm số lần nhấn (click counter) được hiển thị trong TextView, bắt đầu từ 0.

Dưới đây là giao diện của ứng dụng khi hoàn thành:


Task 1 : Tạo và khám phá dự án mới

Trong bài thực hành này, bạn sẽ thiết kế và triển khai dự án cho ứng dụng **HelloToast**.

1.1 Tạo dự án Android Studio

Hãy mở **Android Studio** và tạo một dự án mới với các tham số sau:

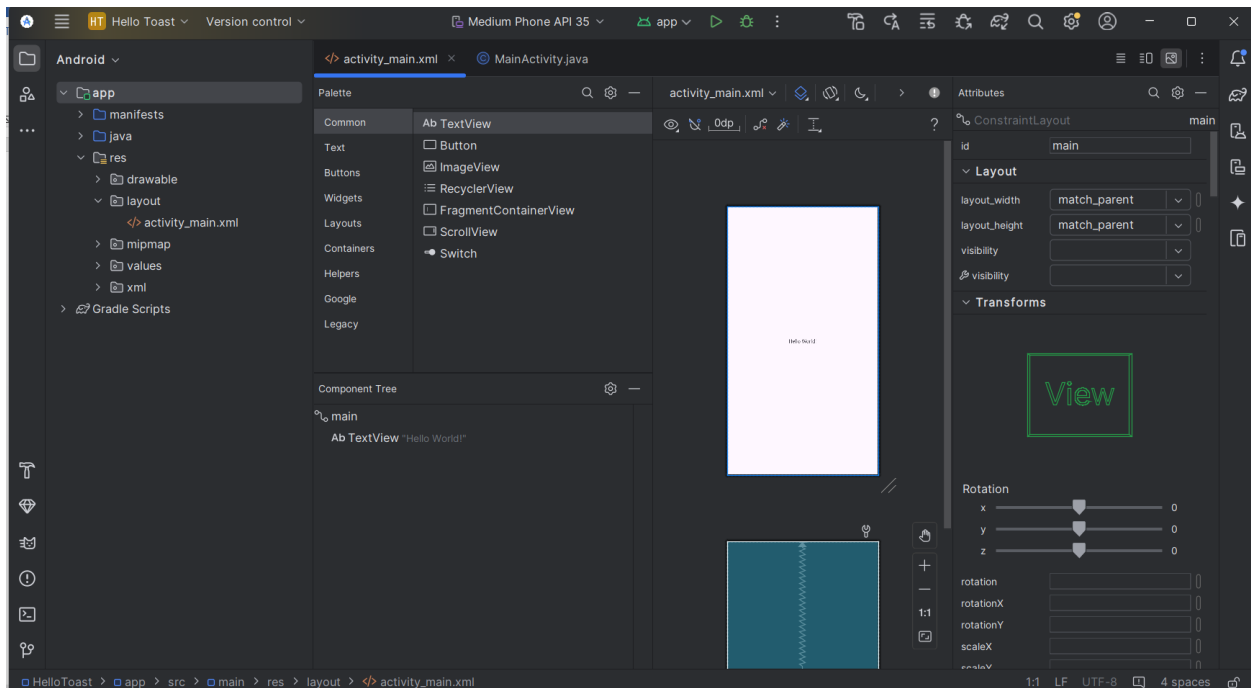
Attribute	Value
Application Name	Hello Toast
Company Name	com.example.android (or your own domain)
Phone and Tablet Minimum SDK	API15: Android 4.0.3 IceCreamSandwich
Template	Empty Activity
Generate Layout file box	Selected
Backwards Compatibility box	Selected

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 (Layout Editor)

Android Studio cung cấp trình chỉnh sửa bố cục (layout editor) để nhanh chóng xây dựng giao diện người dùng (UI) cho ứng dụng. Trình chỉnh sửa này cho phép bạn kéo thả các phần tử vào chế độ thiết kế trực quan và chế độ xem bản thiết kế, định vị chúng trong bố cục, thêm ràng buộc (constraints) và thiết lập thuộc tính. **Ràng buộc (Constraints)** xác định vị trí của một phần tử giao diện người dùng trong bố cục. Một ràng buộc đại diện cho kết nối hoặc căn chỉnh với một thành phần khác, bố cục cha, hoặc một đường hướng dẫn vô hình.

Hãy khám phá Layout Editor và làm theo các bước được đánh số trong hình minh họa bên dưới :



1. Trong ngăn Project > Android, điều hướng đến thư mục app > res > layout, sau đó nhấp đúp vào tệp activity_main.xml để mở nó, nếu nó chưa được mở.
2. Nhấp vào tab Design nếu nó chưa được chọn. Bạn sử dụng tab Design để thao tác với các phần tử và bố cục, còn tab Text để chỉnh sửa mã XML của bố cục.
3. Ngăn Palettes hiển thị các phần tử giao diện người dùng (UI) mà bạn có thể sử dụng trong bố cục của ứng dụng.
4. Ngăn Component tree hiển thị cấu trúc phân cấp (hierarchy) của các phần tử giao diện người dùng. Các phần tử View được tổ chức theo dạng cây gồm cha

và con, trong đó phần tử con sẽ kế thừa thuộc tính của phần tử cha. Trong hình minh họa ở trên, TextView là phần tử con của ConstraintLayout. Bạn sẽ tìm hiểu thêm về các phần tử này sau trong bài học.

5. Các ngăn thiết kế và bản thiết kế của layout editor hiển thị các phần tử giao diện trong bố cục. Trong hình minh họa ở trên, bố cục chỉ hiển thị một phần tử: một TextView hiển thị dòng chữ "Hello World".
6. Tab Attributes hiển thị ngăn Attributes, nơi bạn có thể thiết lập các thuộc tính cho một phần tử giao diện người dùng.



Mẹo: Xem Building a UI with Layout Editor để biết chi tiết về cách sử dụng trình chỉnh sửa bố cục, và tham khảo Meet Android Studio để xem đầy đủ tài liệu hướng dẫn về Android Studio.

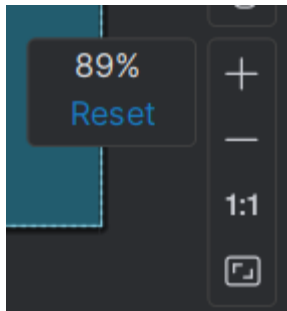
TASK 2 : Thêm các phần tử View trong Layout Editor

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

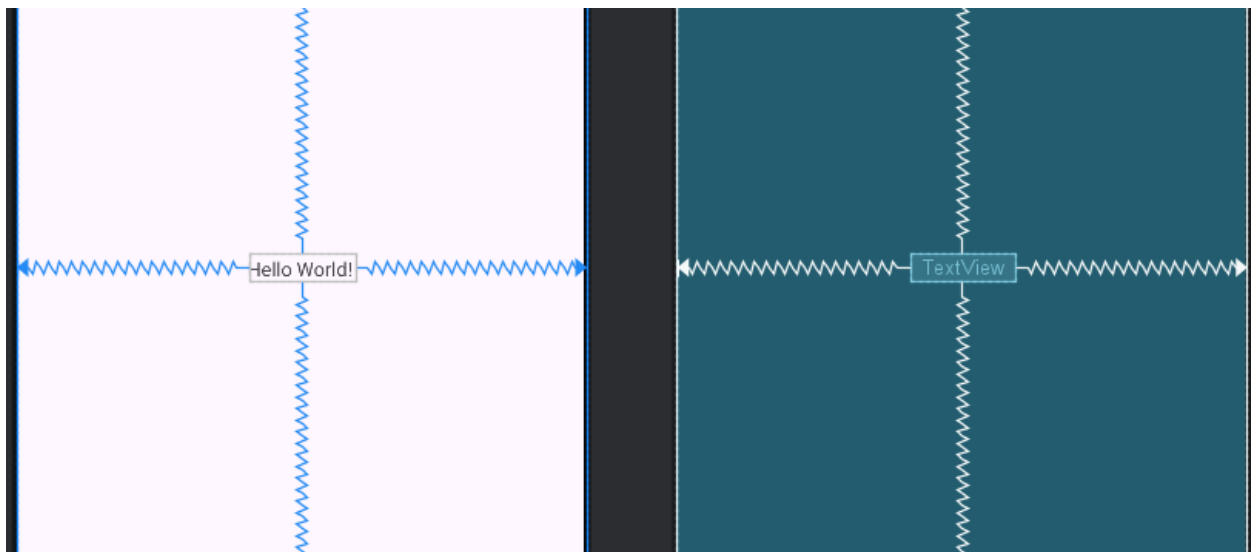
2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện các bước sau :

1. Mở activity_main.xml từ Project > Android nếu nó chưa được mở. Nếu tab Design chưa được chọn, hãy nhấp vào nó. Nếu không có Blueprint, hãy nhấp vào nút Select Design Surface  trên thanh công cụ và chọn Design + Blueprint.
2. Công cụ Autoconnect  cũng nằm trên thanh công cụ và được bật theo mặc định. Trong bước này, hãy đảm bảo rằng công cụ này không bị tắt.



3. Nhấp vào nút Zoom in để phóng to thiết kế và bảng điều khiển blueprint để quan sát rõ hơn.
4. Chọn TextView trong bảng Component Tree. TextView "Hello World" sẽ được làm nổi bật trong thiết kế và bảng blueprint, đồng thời các ràng buộc (constraints) của phần tử sẽ hiển thị.
5. Thực hiện theo hình ảnh động minh họa trong bước này. Nhấp vào nút tròn ở phía bên phải của TextView để xóa ràng buộc ngang (horizontal constraint) đang liên kết nó với phía bên phải của bố cục.
 - TextView sẽ nhảy sang bên trái vì nó không còn bị ràng buộc với phía bên phải nữa.
 - Để thêm lại ràng buộc ngang, hãy nhấp vào cùng một nút tròn đó và kéo một đường kết nối đến phía bên phải của bố cục.



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

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



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

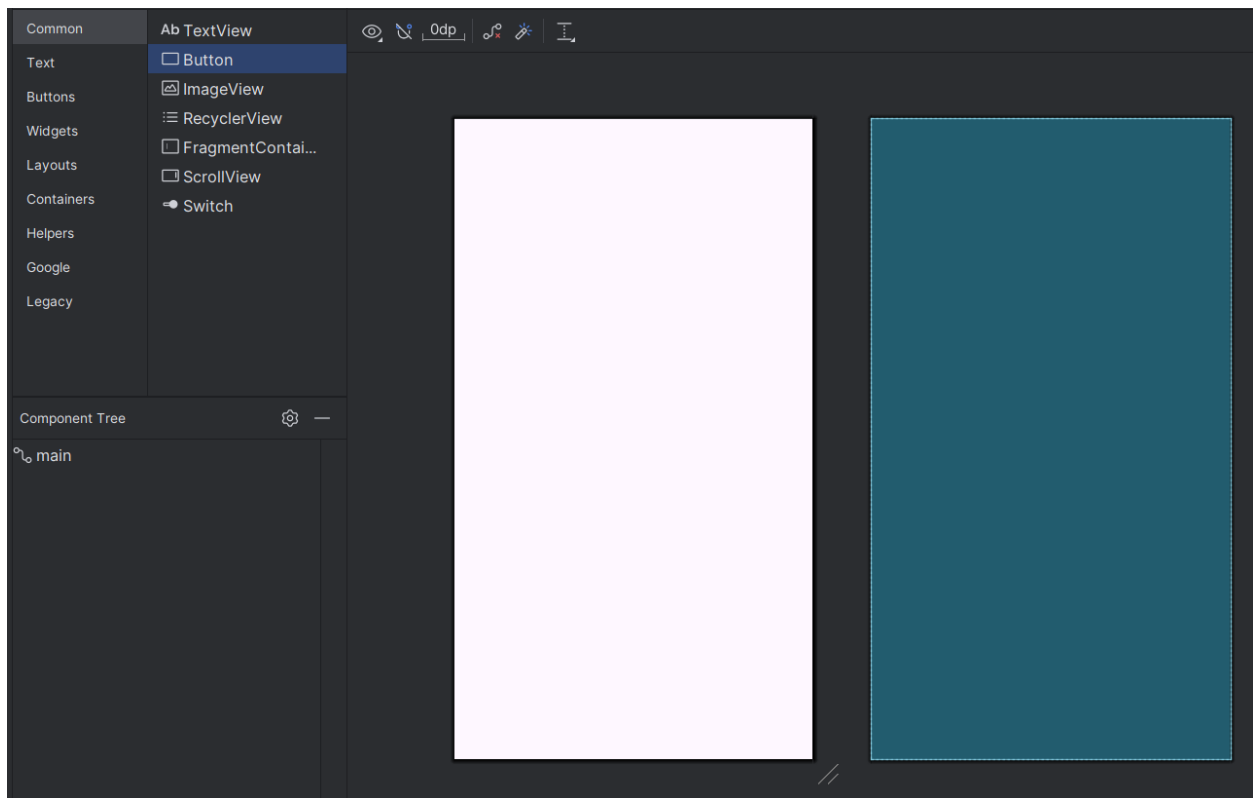


2.2 Thêm một Button vào Layout

Khi được bật, công cụ Autoconnect sẽ 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, nó sẽ tạo các 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 vẫn đang được chọn, hãy nhấn phím Delete hoặc chọn Edit > Delete. Bây giờ bạn có một bố cục hoàn toàn trống.
2. Kéo một Button từ ngăn Palette đến bất kỳ vị trí nào trong bố cục. Nếu bạn thả Button vào khu vực giữa phía trên của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc đến phía trên, bên trái và bên phải của bố cục như trong hình động bên dưới.



2.3 Thêm Button thứ hai vào Layout

1. Kéo một Button khác từ ngăn Palette vào giữa bố cục như trong hình động bên dưới. Autoconnect có thể tự động cung cấp các ràng buộc theo chiều ngang cho bạn (nếu không, bạn có thể kéo chúng thủ công).
2. Kéo một ràng buộc theo chiều dọc đến cạnh dưới của bố cục (tham khảo hình bên dưới).



Bạn có thể xóa 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 ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc cụ thể, hãy nhấp vào tay cầm ràng buộc của nó.

Để xóa tất cả ràng buộc trong toàn bộ bố cục, nhấp vào công cụ Clear All Constraints trên thanh công cụ. Công cụ này hữu ích nếu bạn muốn thiết lập lại tất cả ràng buộc trong bố cục của mình.

TASK3 : Thay đổi thuộc tính của phần tử giao diện người dùng

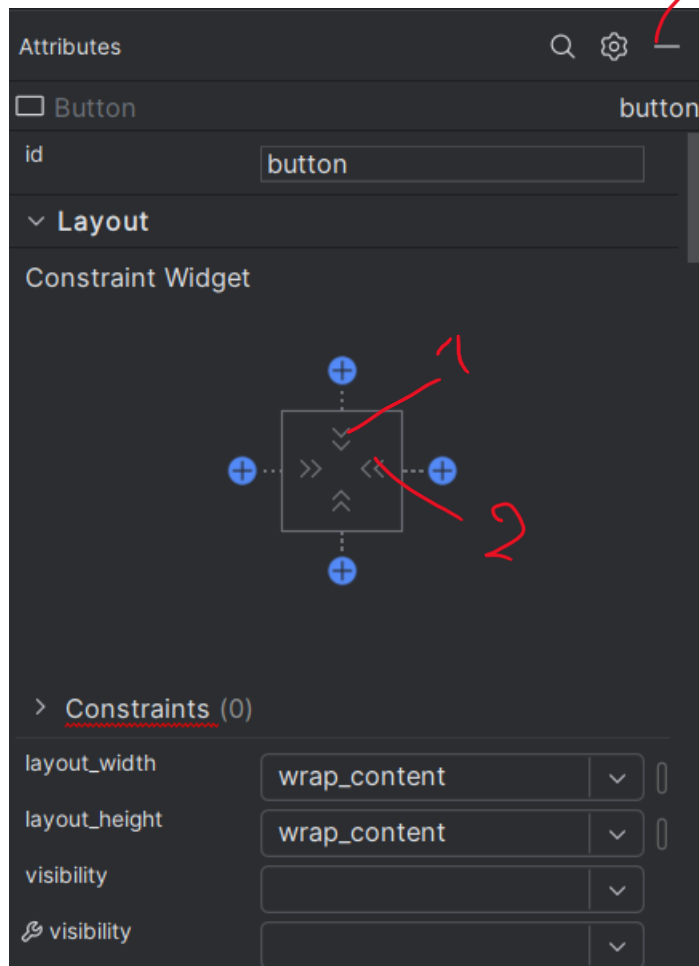
Ngăn Attributes cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng. Bạn có thể tìm thấy các thuộc tính (còn gọi là properties) chung cho tất cả các View trong tài liệu của lớp View.

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

3.1 Thay đổi kích thước Button

Trình chỉnh sửa bố cục (**Layout Editor**) cung cấp **tay cầm thay đổi kích thước** ở bốn góc của một **View**, giúp bạn thay đổi kích thước **View** nhanh chóng. Bạn có thể kéo các tay cầm này để thay đổi kích thước, nhưng cách này sẽ **mã hóa cứng** (hardcode) các kích thước **chiều rộng (width)** và **chiều cao (height)**. **Tránh mã hóa cứng kích thước** cho hầu hết các phần tử **View**, vì chúng không thể thích ứng với các nội dung và kích thước màn hình khác nhau.

Thay vì kéo tay cầm, hãy sử dụng **ngăn Attributes** (Thuộc tính) ở bên phải trình chỉnh sửa bố cục để chọn **chế độ kích thước** không sử dụng kích thước cố định. Trong **ngăn Attributes**, có một **bảng điều chỉnh kích thước** hình vuông (**view inspector**) ở phía trên. Các biểu tượng bên trong hình vuông đại diện cho các thiết lập **chiều cao (height)** và **chiều rộng (width)** như sau:



Trong hình trên:

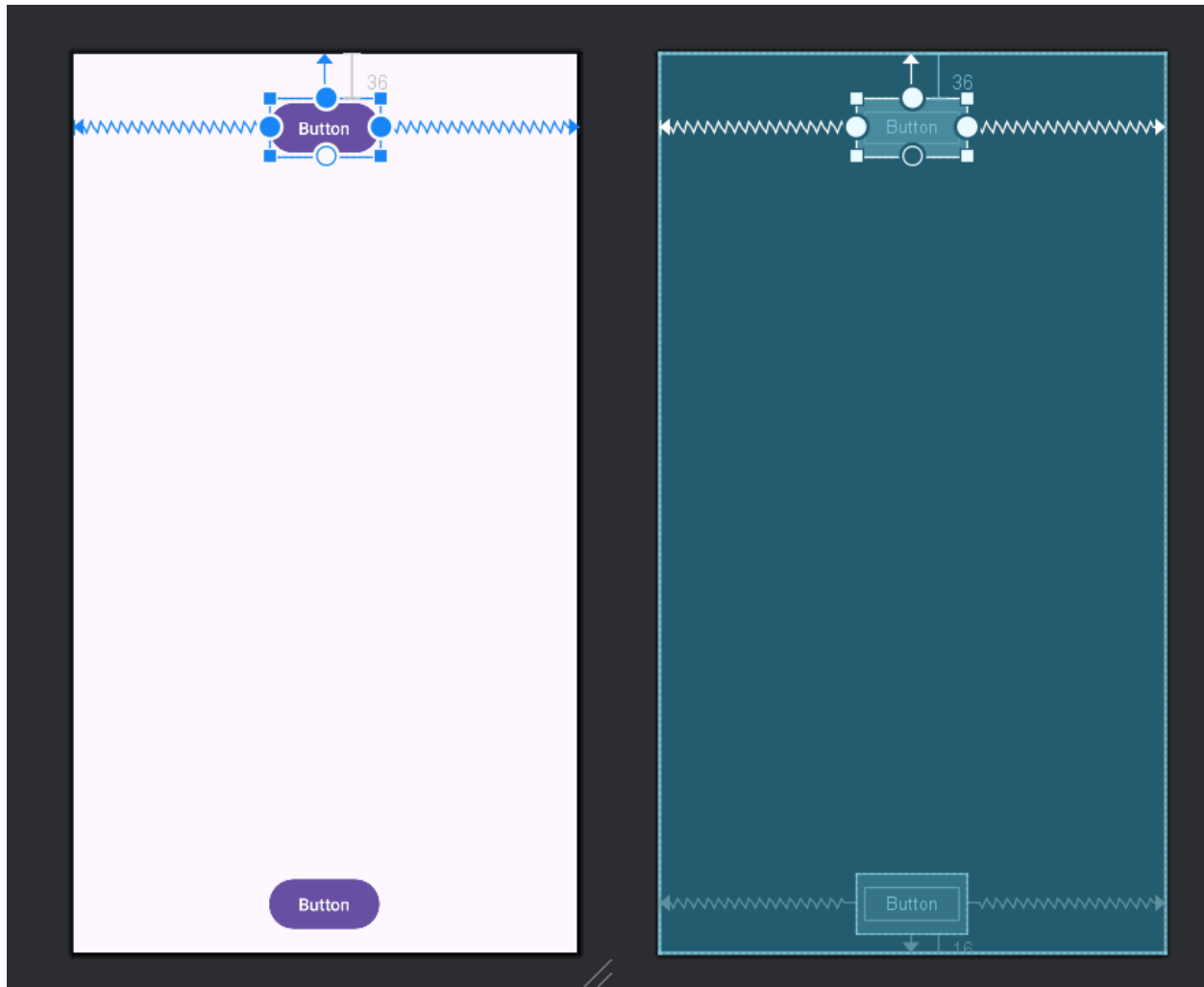
1. **Height control.** Điều khiển này xác định thuộc tính `layout_height` và xuất hiện ở hai đoạn trên và dưới của hình vuông. Các góc chỉ ra rằng điều khiển này được đặt thành `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều dọc khi cần để vừa với nội dung của nó. Số "8" cho biết một lề tiêu chuẩn 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 bên phải của hình vuông. Các góc chỉ ra rằng điều khiển này được đặt thành `wrap_content`. Điều đó có nghĩa là View sẽ mở rộng theo chiều ngang khi cần để vừa với nội dung của nó, lên đến một lề 8dp.
3. **Attributes** Nhấp để đóng bảng.

Thực hiện theo các bước sau:

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

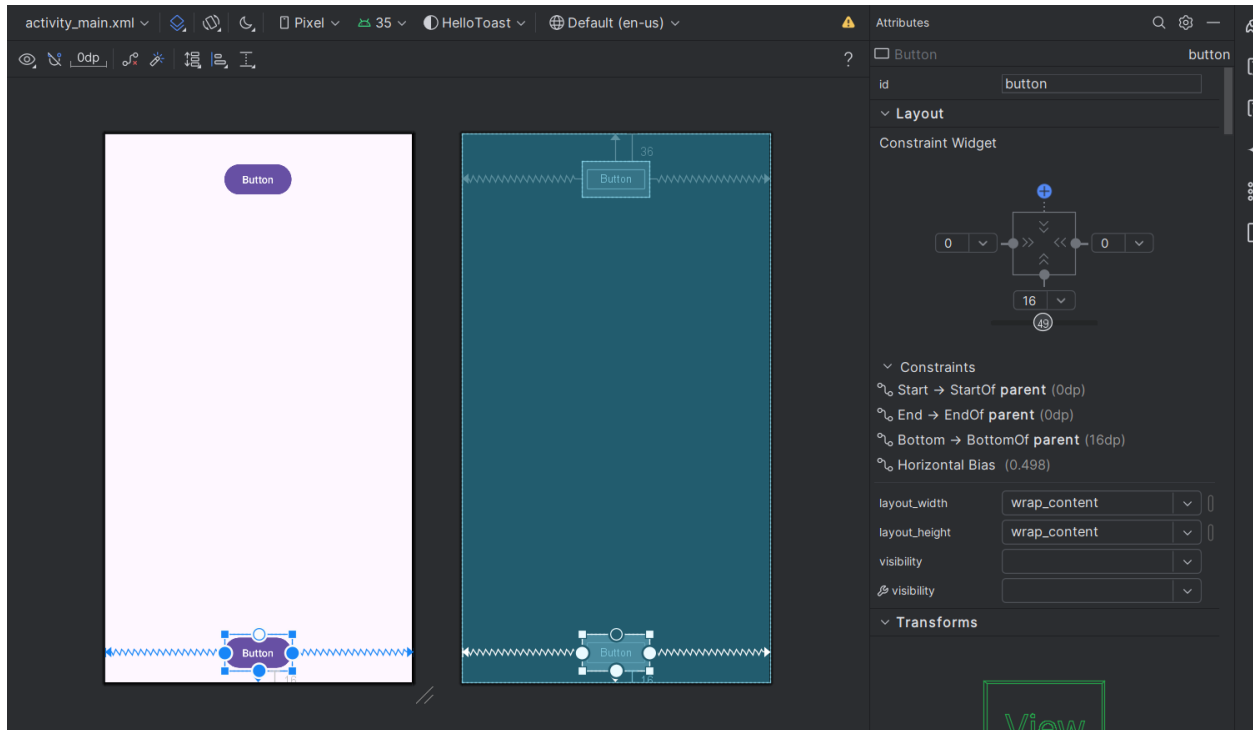


3. Nhấp vào điều khiển chiều rộng hai lần—lần nhấp đầu tiên thay đổi nó thành **Fixed** với các đường thẳng, và lần nhấp thứ hai thay đổi nó thành **Match Constraints** với các lò xo, như hiển thị trong hình động bên dưới.



Do thay đổi điều khiển chiều rộng, thuộc tính `layout_width` trong bảng **Attributes** hiển thị giá trị **match_constraint**, và phần tử **Button** sẽ kéo giãn theo chiều ngang để lấp đầy không gian giữa hai cạnh trái và phải của bố cục.

4. Chọn nút **Button** thứ hai và thực hiện các thay đổi tương tự đối với **layout_width** như ở bước trước, như hiển thị trong hình dưới đây.



Như đã hiển thị trong các bước trước, các thuộc tính `layout_width` và `layout_height` trong bảng **Attributes** sẽ thay đổi khi bạn điều chỉnh các điều khiển chiều cao và chiều rộng trong **inspector**. Những thuộc tính này có thể nhận một trong ba giá trị cho bố cục, khi sử dụng **ConstraintLayout**:

- **match_constraint**: Cài đặt này mở rộng phần tử **View** để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao—tối đa đến lề nếu có thiết lập. Phần tử cha trong trường hợp này là **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.
- **wrap_content**: Cài đặt này thu nhỏ kích thước của phần tử **View** sao cho nó vừa đủ để chứa nội dung của mình. Nếu không có nội dung, phần tử **View** sẽ trở nên vô hình.
- **Kích thước cố định**: Để xác định một kích thước cố định có thể điều chỉnh theo kích thước màn hình của thiết bị, hãy sử dụng một số cố định với đơn vị **density-independent pixels (dp)**. Ví dụ, **16dp** có nghĩa là 16 pixel độc lập với mật độ màn hình.

Mẹo: Nếu bạn thay đổi thuộc tính `layout_width` bằng cách sử dụng menu bật lên của nó, thuộc tính `layout_width` sẽ được đặt thành **0** vì không có kích thước cố định được thiết lập. Cài đặt này tương đương với **match_constraint**—phần tử **View** có thể mở rộng tối đa để đáp ứng các ràng buộc và cài đặt lề.

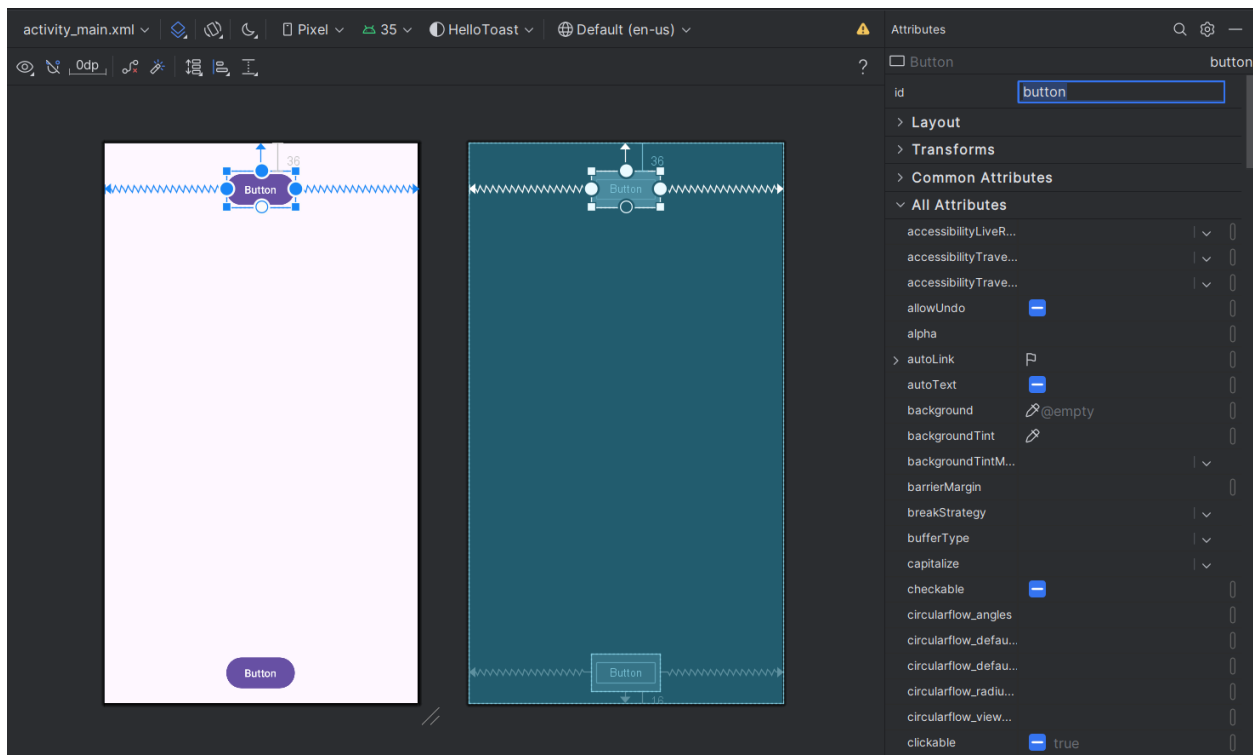
3.2 Thay đổi các thuộc tính của Button

Để xác định từng **View** một cách duy nhất trong bố cục **Activity**, mỗi **View** hoặc lớp con của **View** (chẳng hạn như **Button**) cần một **ID** duy nhất. Ngoài ra, để có chức năng, các phần tử **Button** cần có văn bản hiển thị. Các phần tử **View** cũng có thể có nền là màu hoặc hình ảnh.

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

Dưới đây là hình minh họa cách thực hiện các bước :

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



- 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 giữ nguyên màu nền cùng màu chữ như ở các bước trước.

colorPrimary là màu chính của giao diện ứng dụng, một trong các màu cơ bản được định nghĩa trước trong tệp tài nguyên **colors.xml**. Màu này được sử dụng cho **thanh ứng dụng (app bar)**. Sử dụng các màu cơ bản cho các phần tử giao diện người dùng khác giúp tạo ra một giao diện đồng nhất. Bạn sẽ tìm hiểu thêm về **chủ đề ứng dụng (app themes)** và **Material Design** trong bài học khác.

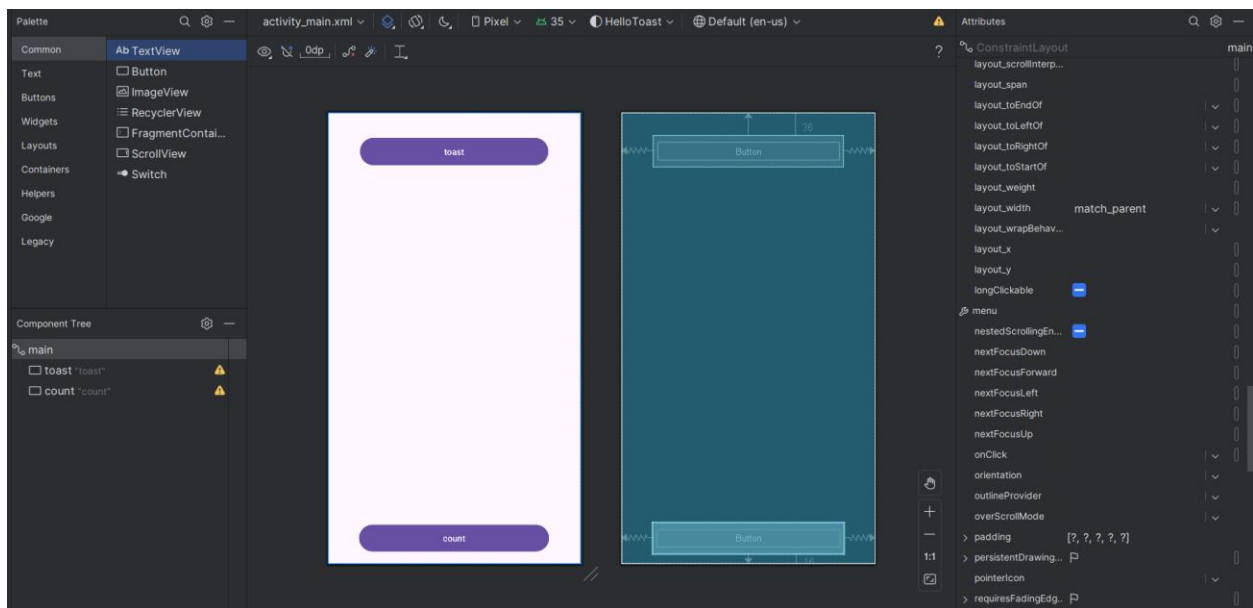
TASK 4 : Thêm một TextView và thiết lập thuộc tính

Một trong những lợi ích của **ConstraintLayout** là khả năng căn chỉnh hoặc ràng buộc các phần tử so với các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một **TextView** vào giữa bố cục, ràng buộc nó theo chiều ngang với lề và theo chiều dọc với hai phần

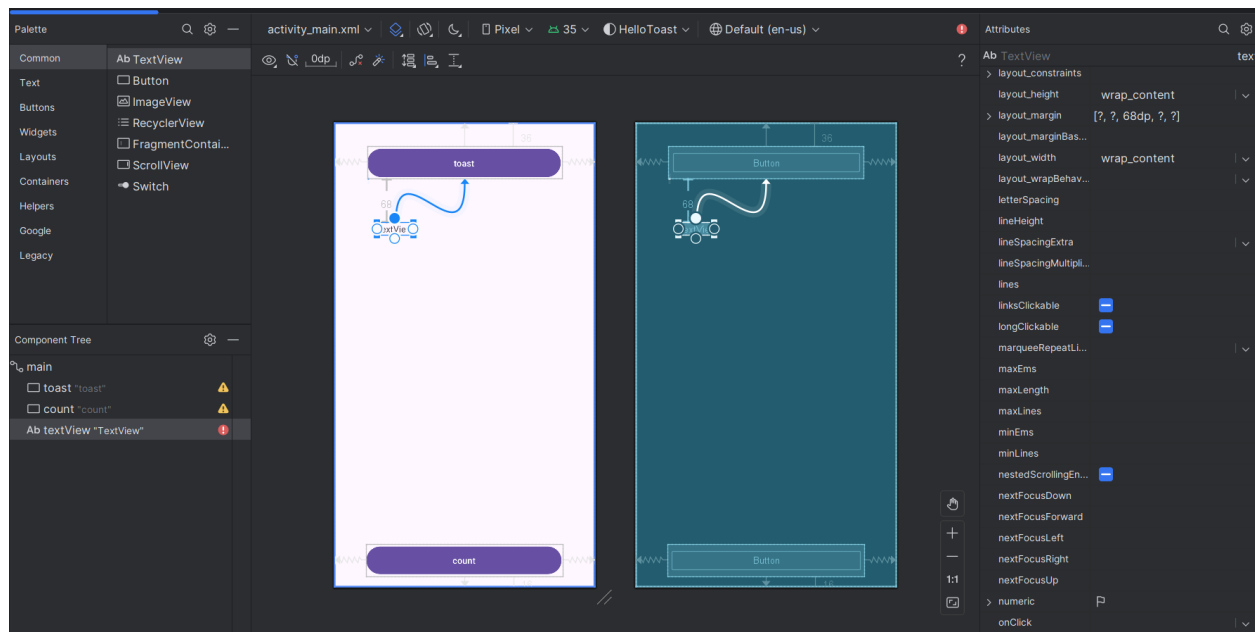
từ **Button**. Sau đó, bạn sẽ thay đổi các thuộc tính cho **TextView** trong bảng **Attributes**.

4.1 Thêm một **TextView** và thiết lập ràng buộc

1. Như minh họa trong hình động bên dưới, kéo một **TextView** từ bảng **Palette** vào phần trên của bố cục, sau đó kéo một ràng buộc từ mép trên của **TextView** đến tay cầm phía dưới của nút **Toast**. Điều này sẽ ràng buộc **TextView** nằm bên dưới nút **Button** này.




2. Tiếp theo, kéo một ràng buộc từ mép dưới của **TextView** đến tay cầm phía trên của nút **Count**, và từ hai bên của **TextView** đến mép của bố cục. Điều này sẽ căn chỉnh **TextView** vào giữa bố cục, giữa hai nút **Button**.



4.2 Thiết lập thuộc tính cho TextView

Với **TextView** đã được chọn, mở bảng **Attributes** nếu nó chưa được mở. Thiết lập các thuộc tính cho **TextView** như trong hình động bên dưới. Các thuộc tính mới sẽ được giải thích sau hình minh họa:

1. Đặt **ID** thành **show_count**.
2. Đặt **text** thành **0**.
3. Đặt **textSize** thành **160sp**.
4. Đặt **textStyle** thành **B** (đậm) và **textAlignment** thành **ALIGN_CENTER** (căn giữa đoạn văn bản).
5. Thay đổi điều khiển kích thước ngang và dọc (**layout_width** và **layout_height**) thành **match_constraint**.
6. Đặt **textColor** thành **@color/colorPrimary**.
7. Cuộn xuống trong bảng **Attributes**, nhấp vào **View all attributes**, tiếp tục cuộn xuống trang thứ hai đến thuộc tính **background**, sau đó nhập **#FFF00** để đặt màu nền thành một tông màu vàng.

8. Cuộn xuống đến **gravity**, mở rộng thuộc tính **gravity**, và chọn **center_ver** (căn giữa theo chiều dọc).
- **textSize**: Kích thước chữ của **TextView**. Trong bài học này, kích thước được đặt là **160sp**. **sp** là viết tắt của **scale-independent pixel**, tương tự như **dp**, là một đơn vị điều chỉnh theo mật độ màn hình và kích thước chữ mà người dùng đã chọn. Khi đặt kích thước chữ, hãy sử dụng **sp** để đảm bảo chúng điều chỉnh phù hợp với cả mật độ màn hình và sở thích của người dùng.
 - **textStyle** và **textAlignment**: Kiểu chữ, được đặt thành **B** (đậm) trong bài học này, và căn chỉnh văn bản, được đặt thành **ALIGN_CENTER** (căn giữa đoạn văn bản). 
 - **gravity**: Thuộc tính **gravity** xác định cách một **View** được căn chỉnh trong **View** hoặc **ViewGroup** cha của nó. Trong bước này, bạn căn giữa **TextView** theo chiều dọc trong **ConstraintLayout**.

Bạn có thể nhận thấy rằng thuộc tính **background** xuất hiện trên trang đầu tiên của bảng **Attributes** đối với **Button**, nhưng lại nằm trên trang thứ hai đối với **TextView**. Bảng **Attributes** thay đổi tùy theo loại **View**: Các thuộc tính phổ biến nhất của loại **View** sẽ hiển thị trên trang đầu tiên, còn các thuộc tính khác được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của bảng **Attributes**, hãy nhấp vào biểu tượng trong thanh công cụ ở đầu bảng.

TASK 5 : Chỉnh sửa giao diện trong XML

Ứng dụng **Hello Toast** gần như đã hoàn thành! Tuy nhiên, có một dấu chấm than xuất hiện bên cạnh mỗi phần tử giao diện trong **Component Tree**. Khi di chuột vào các dấu chấm than này, bạn sẽ thấy thông báo cảnh báo.

Tất cả các phần tử đều có cùng một cảnh báo: "**Chuỗi ký tự cứng (hardcoded strings) nên được sử dụng từ tài nguyên.**"

Cách dễ nhất để sửa lỗi này là chỉnh sửa giao diện trong XML. Trình chỉnh sửa giao diện (**Layout Editor**) rất mạnh mẽ, nhưng một số thay đổi sẽ dễ thực hiện hơn trong mã XML.

5.1 Mở mã XML của giao diện

1. Mở tệp **activity_main.xml**, nếu chưa mở.
2. Nhấn vào tab **Text** ở phía dưới của trình chỉnh sửa giao diện.

Trình chỉnh sửa XML sẽ xuất hiện, thay thế chế độ xem thiết kế và bản thiết kế (**Blueprint**).

Bạn sẽ thấy các cảnh báo được đánh dấu, ví dụ: chuỗi ký tự "Toast" và "Count". (Chuỗi "0" cũng được đánh dấu nhưng không hiển thị trong hình minh họa.)

5.2 Trích xuất chuỗi tài nguyên (Extract String Resources)

Thay vì mã hóa cứng (hardcode) chuỗi trong mã nguồn, ta nên sử dụng **tài nguyên chuỗi (string resources)**. Việc lưu trữ chuỗi trong một tệp riêng biệt giúp dễ dàng quản lý hơn, đặc biệt nếu sử dụng lại nhiều lần hoặc cần dịch ứng dụng sang nhiều ngôn ngữ

Thực hiện trích xuất chuỗi tài nguyên

1. Nhấp vào từ "Toast" (cảnh báo đầu tiên).
2. Nhấn **Alt + Enter** trên Windows hoặc **Option + Enter** trên macOS, sau đó chọn **Extract string resource**.
3. Đặt tên tài nguyên là **button_label_toast**.
4. Nhấn **OK**. Một tài nguyên chuỗi sẽ được tạo trong tệp **res/values/strings.xml**, và chuỗi trong mã XML sẽ được thay thế bằng **@string/button_label_toast**.
5. Lặp lại các bước trên cho các chuỗi:
 - "Count" → **button_label_count**
 - "0" → **count_initial_value**

6. Mở **Project > Android**, mở thư mục **res > values**, sau đó mở **strings.xml** để kiểm tra danh sách tài nguyên chuỗi:

```
<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_initial_value">0</string>

</resources>
```

7. Thêm một chuỗi mới để hiển thị thông báo **"Hello Toast!"**:

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

TASK 6: Thêm trình xử lý sự kiện onClick cho các nút

Bạn sẽ thêm một phương thức Java cho mỗi **Button** trong **MainActivity** để thực thi khi người dùng nhấn vào nút.

6.1 Thêm thuộc tính onClick và trình xử lý sự kiện vào Button

Một **click handler** là một phương thức sẽ được gọi khi người dùng nhấp hoặc chạm vào một phần tử có thể nhấn (**clickable UI element**).

Có hai cách để thêm **click handler** trong Android Studio:

- **Cách 1:** Điền tên phương thức trong thuộc tính **onClick** của **Attributes Pane** (trong tab **Design**).
- **Cách 2:** Thêm thuộc tính **android:onClick** trong XML (cách này sẽ được sử dụng trong bài tập này).

Thực hiện

1. Mở **activity_main.xml** trong chế độ **Text**. Tìm **Button** có thuộc tính **android:id="@+id/button_toast"**:<Button


```
android:id="@+id/button_toast"
android:layout_width="0dp"
...
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

3. Thêm thuộc tính **android:onClick** vào Button này, đặt tên phương thức là `showToast`:

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    ...
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="showToast" />
```

4. Khi thấy biểu tượng bóng đèn đỏ xuất hiện bên cạnh thuộc tính **onClick**, nhấp vào nó và chọn **Create click handler**.

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

Mã XML cho các phần tử giao diện người dùng bên trong **ConstraintLayout** hiện trông như thế này:

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
```

```
android:background="@color/colorPrimary"
android:text="@string/button_label_toast"
android:textColor="@android:color/white"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent"
android:onClick="showToast"/>
```

<Button

```
android:id="@+id/button_count"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_marginBottom="8dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:background="@color/colorPrimary"
android:text="@string/button_label_count"
android:textColor="@android:color/white"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
android:onClick="countUp"
```

/>

<TextView

```
android:id="@+id/show_count"
android:layout_width="0dp"
android:layout_height="0dp"
android:layout_marginBottom="8dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:layout_marginTop="8dp"
android:background="#FFFF00"
android:gravity="center_vertical"
android:text="@string/count_initial_value"
```

```
android:textAlignment="center"
android:textColor="@color/colorPrimary"
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" />
```

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

```
package com.example.android.hellotoast;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void showToast(View view) {
    }
    public void countUp(View view) {
    }
}
```

6.2 Chỉnh sửa trình xử lý nút Toast

Bây giờ bạn sẽ chỉnh sửa phương thức **showToast()**—trình xử lý khi nhấn nút **Toast** trong **MainActivity**—để hiển thị một thông báo.

Toast cung cấp một cách hiển thị thông báo đơn giản trong một cửa sổ popup nhỏ. Nó chỉ chiếm một khoảng không gian vừa đủ cho nội dung thông báo. **Toast** không làm gián đoạn hoạt động của ứng dụng, và cửa sổ hiện tại vẫn hiển thị cũng như có thể tương tác.

Toast rất hữu ích để kiểm tra tính tương tác trong ứng dụng—bạn có thể thêm một thông báo **Toast** để hiển thị kết quả khi nhấn một **Button** hoặc thực hiện một hành động nào đó.

Thực hiện các bước sau để chỉnh sửa trình xử lý khi nhấn nút **Toast**:

1. Tìm phương thức **showToast()** vừa được tạo.

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

2. Để tạo một thể hiện của **Toast**, hãy gọi phương thức **makeText()** của lớp **Toast**.

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

3. Cung cấp **context** của **Activity** trong ứng dụng. Vì **Toast** hiển thị trên giao diện của **Activity**, hệ thống cần thông tin về **Activity** hiện tại. Khi bạn đang ở trong **Activity** mà bạn cần **context**, có thể sử dụng **this** như một cách viết tắt.

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

4. Cung cấp thông báo cần hiển thị, chẳng hạn như một **string resource** (chuỗi **toast_message** mà bạn đã tạo ở bước trước). Chuỗi **toast_message** được xác định bằng **R.string.toast_message**.

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

5. Cung cấp thời gian hiển thị. Ví dụ, **Toast.LENGTH_SHORT** sẽ hiển thị **Toast** trong một khoảng thời gian ngắn.

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

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

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

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

6.3 Chỉnh sửa trình xử lý nút **Count**

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

Mã cho trình xử lý này cần phải:

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

Thực hiện các bước sau để chỉnh sửa trình xử lý khi nhấn nút **Count**:

1. Tìm phương thức **countUp()** vừa được tạo.

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

2. Để theo dõi số đếm, bạn cần một biến thành viên **private**. Mỗi lần nhấn nút **Count** sẽ tăng giá trị của biến này. Nhập đoạn mã sau, nó sẽ được tô đỏ và hiển thị biểu tượng bóng đèn màu đỏ:

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

3. Nhấp vào biểu tượng bóng đèn màu đỏ và chọn **Create field 'mCount'** từ menu bật lên. Thao tác này sẽ tạo một biến thành viên **private** ở đầu **MainActivity**, và Android Studio sẽ mặc định kiểu dữ liệu của nó là **int**.
4. Thay đổi câu lệnh khai báo biến thành viên **private** để khởi tạo biến với giá trị **0**.

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

5. Cùng với biến ở trên, bạn cũng cần một biến thành viên **private** để tham chiếu đến **TextView** có **id** là **show_count**, và bạn sẽ sử dụng nó trong trình xử lý sự kiện khi nhấn nút. Đặt tên cho biến này là **mShowCount**.
6. Bây giờ bạn đã có **mShowCount**, bạn có thể lấy tham chiếu đến **TextView** bằng cách sử dụng **ID** đã đặt trong tệp layout. Để chỉ lấy tham chiếu này một lần, hãy khai báo nó trong phương thức **onCreate()**. Như bạn sẽ học trong một bài học khác, phương thức **onCreate()** được sử dụng để **inflate layout**, nghĩa là thiết lập nội dung của màn hình dựa trên tệp XML layout. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện khác trong layout, chẳng hạn như **TextView**. Tìm phương thức **onCreate()** trong **MainActivity**.
7. Thêm câu lệnh **findViewById** vào cuối phương thức.

Một **View**, giống như một chuỗi, là một tài nguyên có thể có **ID**.

Lệnh **findViewById** nhận **ID** của một **View** làm tham số và trả về đối tượng **View** tương ứng.

Vì phương thức này trả về một **View**, bạn cần ép kiểu kết quả về loại **View** mong đợi, trong trường hợp này là **(TextView)**.

8. Bây giờ, sau khi đã gán **TextView** cho biến **mShowCount**, bạn có thể sử dụng biến này để cập nhật nội dung văn bản trong **TextView** với giá trị của biến **mCount**. Thêm đoạn mã sau vào phương thức **countUp()**:

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

9. Chạy ứng dụng để kiểm tra rằng số đếm tăng lên khi bạn nhấn vào nút **Count**.

Thử thách lập trình

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

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

Thử thách: Thay đổi bố cục để ứng dụng hiển thị tốt ở cả hai hướng ngang và dọc

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

- **Giải pháp thử thách: Điều chỉnh bố cục trong Android Studio**
- **Dự án: HelloToastChallenge**
- **Tóm tắt kiến thức quan trọng**
- **1. View, ViewGroup và Layouts:**
 - Tất cả các phần tử UI đều là **subclass** của lớp View, do đó chúng kế thừa nhiều thuộc tính từ View.
 - Các phần tử View có thể được nhóm trong một ViewGroup, đóng vai trò là **container**.
 - Quan hệ giữa ViewGroup (cha) và View (con) có dạng **cha - con**, trong đó ViewGroup là **cha** còn View hoặc ViewGroup khác là **con**.
- **2. onCreate() và inflate layout:**
 - Phương thức `onCreate()` được sử dụng để **inflate layout**, có nghĩa là thiết lập giao diện hiển thị trên màn hình từ XML layout.
 - Phương thức này cũng có thể được sử dụng để lấy tham chiếu đến các phần tử UI trong layout.
- **3. Sử dụng findViewById:**
 - Một View cũng giống như một **resource** (tài nguyên) có thể có một id.
 - Lệnh `findViewById` lấy id của một View làm tham số và trả về View.

-
- **Sử dụng Layout Editor:**

- **Tab Design:** Để kéo, thả và điều chỉnh phần tử UI trực quan.
- **Tab Text:** Để chỉnh sửa mã XML của layout.
- **Pane Palettes:** Chứa các thành phần UI có thể sử dụng.
- **Component Tree:** Hiển thị cấu trúc cây của các phần tử UI.
- **Pane Attributes:** Hiển thị và chỉnh sửa thuộc tính của các phần tử UI.

4. Các công cụ quan trọng trong Layout Editor:

- **Constraint Handle:** Được biểu thị bằng **vòng tròn** trên mỗi cạnh của phần tử. Kéo để tạo ràng buộc (constraint).
- **Resizing Handle:** Dùng để thay đổi kích thước phần tử bằng cách kéo các góc vuông.
- **Autoconnect Tool:** Tạo ràng buộc tự động cho phần tử UI dựa trên vị trí của nó.
- **Clear Constraints:** Dùng để xóa tất cả hoặc từng ràng buộc cụ thể.

5. Cài đặt chiều rộng và chiều cao của Layout:

- layout_width và layout_height có thể có các giá trị sau:
 - **match_constraint (0dp):** Mở rộng phần tử để lấp đầy parent.
 - **wrap_content:** Giữ kích thước phần tử vừa đủ để hiển thị nội dung.
 - **Giá trị cố định (dp):** Đặt kích thước cố định theo **density-independent pixels**.

6. Xử lý sự kiện Click:

- **Click handler** là phương thức được gọi khi người dùng nhấn vào một phần tử UI.
- Cách khai báo click handler:
 - Trong **Attributes** → onClick
 - Trong XML: android:onClick="showToast"
 - Trong MainActivity:

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

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

```
}
```

7. Hiển thị Toast Message:

1. Gọi phương thức `makeText()` từ lớp `Toast`.
 2. Truyền vào **context**, **nội dung thông báo** (string resource).
 3. Truyền vào **thời gian hiển thị** (`Toast.LENGTH_SHORT` hoặc `Toast.LENGTH_LONG`).
 4. Gọi `show()` để hiển thị **Toast**.
-

Tham khảo thêm tài liệu:

- **Android Studio**
- **Build UI với Layout Editor**
- **ConstraintLayout**
- **Button & TextView**
- **Android Resources**
- **Hỗ trợ Density khác nhau**
- **Sự kiện Input trên Android**
- **Context trong Android**

1.3) Trình chỉnh sửa bố cục

Bài học 1.2 Phần B: Trình chỉnh sửa Layout

Giới thiệu

Như bạn đã học trong **Bài 1.2 Phần A: Giao diện người dùng tương tác đầu tiên**, bạn có thể xây dựng giao diện người dùng (UI) bằng **ConstraintLayout** trong trình chỉnh sửa bố cục (**layout editor**).

- **ConstraintLayout** sắp xếp các phần tử UI trong bố cục bằng cách sử dụng các ràng buộc (**constraints**) với các phần tử khác hoặc với mép của bố cục.
- **ConstraintLayout** là một **ViewGroup**, một loại **View đặc biệt** có thể chứa các **View** khác (được gọi là **child views**).

- Bài thực hành này sẽ giúp bạn khám phá thêm nhiều tính năng của **ConstraintLayout** và trình chỉnh sửa bố cục (**layout editor**).
- **Các loại ViewGroup khác**

Bên cạnh **ConstraintLayout**, bài học này giới thiệu thêm hai **ViewGroup** quan trọng:

1. **LinearLayout**

- Xếp thẳng hàng các phần tử con theo **chiều ngang** hoặc **chiều dọc**.

2. **RelativeLayout**

- Căn chỉnh các phần tử dựa trên **vị trí của các phần tử khác** trong cùng một **ViewGroup**.

- **Kiến thức yêu cầu trước khi bắt đầu**

Bạn nên biết cách:

- Tạo ứng dụng **Hello World** bằng **Android Studio**.
- Chạy ứng dụng trên **giả lập hoặc thiết bị thực tế**.
- Tạo bố cục đơn giản bằng **ConstraintLayout**.
- Trích xuất và sử dụng **string resources**.

- **Những gì bạn sẽ học**

- Cách tạo **layout biến thể** cho màn hình ngang (**landscape**).
- Cách tạo **layout biến thể** cho **máy tính bảng** và màn hình lớn hơn.
-

- **Những gì bạn sẽ làm**

- Tạo một biến thể layout cho chế độ ngang (landscape).
- Sử dụng baseline constraints để căn chỉnh UI.
- Thử nghiệm **LinearLayout** và **RelativeLayout**.

→ **Mục tiêu cuối cùng:** Giúp bạn hiểu rõ hơn về cách thiết kế giao diện linh hoạt cho nhiều loại màn hình khác nhau trên Android!

TỔNG QUAN VỀ ỨNG DỤNG

Ứng dụng **Hello Toast** trong bài học trước sử dụng **ConstraintLayout** để sắp xếp các phần tử giao diện người dùng (UI) trong bố cục **Activity**, như được hiển thị trong hình dưới đây

TASK 1 : Tạo layout variants

Trong bài học trước, thử thách lập trình yêu cầu thay đổi bố cục của ứng dụng **Hello Toast** để nó có thể hiển thị đúng trong chế độ dọc và ngang. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng tạo các biến thể của bố cục cho các chế độ **ngang (landscape)** và **dọc (portrait)** trên điện thoại, cũng như cho màn hình lớn như **máy tính bảng (tablet)**.

Bạn sẽ sử dụng một số nút trong hai thanh công cụ phía trên của trình chỉnh sửa bố cục (**Layout Editor**). Thanh công cụ trên cùng cho phép bạn định cấu hình cách hiển thị bản xem trước bố cục trong trình chỉnh sửa:

Các chức năng trong thanh công cụ trên cùng:

1. Chọn bề mặt thiết kế:

- **Design:** Hiển thị bản xem trước đầy màu sắc của bố cục.
- **Blueprint:** Chỉ hiển thị đường viền của từng phần tử UI.
- **Design + Blueprint:** Hiển thị cả hai chế độ song song.

2. Hướng màn hình trong trình chỉnh sửa:

- Chọn **Portrait** hoặc **Landscape** để hiển thị bản xem trước ở chế độ dọc hoặc ngang.
- Để tạo bố cục thay thế, chọn **Create Landscape Variation** hoặc các biến thể khác.

3. Thiết bị trong trình chỉnh sửa: Chọn loại thiết bị (**điện thoại/máy tính bảng, Android TV, hoặc Android Wear**).

4. **Phiên bản API trong trình chỉnh sửa:** Chọn phiên bản Android để hiển thị bản xem trước.

5. **Chủ đề trong trình chỉnh sửa:** Chọn chủ đề (**AppTheme** hoặc chủ đề khác).

6. **Ngôn ngữ trong trình chỉnh sửa:** Chọn ngôn ngữ để hiển thị bản xem trước. Bạn cũng có thể chọn **Preview as Right to Left** để xem bố cục theo hướng ngôn ngữ RTL.

Các chức năng trong thanh công cụ thứ hai:

1. Hiển thị ràng buộc (Show Constraints) và lề (Show Margins).
2. Autoconnect: Bật/tắt tự động tạo ràng buộc khi kéo thả phần tử UI.
3. Xóa tất cả ràng buộc (Clear All Constraints).
4. Suy luận ràng buộc (Infer Constraints): Tạo ràng buộc tự động.
5. Lề mặc định (Default Margins).
6. Sắp xếp các phần tử (Pack).
7. Căn chỉnh (Align).
8. Thêm đường hướng dẫn (Guidelines).
9. Thu phóng và di chuyển (Zoom/pan controls).

1.1 Xem trước bố cục ở chế độ ngang

Cách xem trước bố cục của ứng dụng Hello Toast ở chế độ ngang:

1. Mở ứng dụng Hello Toast từ bài học trước.
2. Mở tệp `activity_main.xml` trong trình chỉnh sửa bố cục.
3. Nhấp vào nút Orientation in Editor trên thanh công cụ trên cùng.
4. Chọn Switch to Landscape để xem bố cục ở chế độ ngang.

1.2 Tạo một layout variant cho chế độ ngang

Trong chế độ ngang, số hiển thị trong TextView có thể bị đặt quá thấp so với nút Count. Để khắc phục điều này mà không ảnh hưởng đến chế độ dọc, bạn có thể tạo một biến thể riêng cho chế độ ngang:

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

- Một cửa sổ trình chỉnh sửa mới sẽ mở với tab `land/activity_main.xml`.
3. Trong Project > Android, bạn sẽ thấy tệp `activity_main.xml` (land) được tạo tự động.

1.3 Xem trước bố cục trên các thiết bị khác nhau

Bạn có thể xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng:

1. Mở tab **land/activity_main.xml**.
2. Nhấp vào nút **Device in Editor** trên thanh công cụ.
3. Chọn các thiết bị khác nhau như **Nexus 4**, **Nexus 5**, **Pixel** để thấy sự khác biệt.

1.4 Chỉnh sửa bố cục cho chế độ ngang

Bạn có thể sử dụng tab **Text** trong trình chỉnh sửa XML để chỉnh sửa trực tiếp mã XML:

1. Mở **land/activity_main.xml**.
2. Chuyển sang tab **Text** và mở **Preview**.
3. Tìm phần tử **TextView** trong mã XML.
4. Thay đổi thuộc tính `android:textSize="160sp"` thành `android:textSize="120sp"`.
5. Chạy ứng dụng trên trình giả lập hoặc thiết bị thật, sau đó thay đổi hướng màn hình để kiểm tra.

1.5 Tạo một layout variant cho máy tính bảng

Khi xem bố cục trên thiết bị máy tính bảng như **Nexus 10**, bạn sẽ thấy bố cục không phù hợp. Để khắc phục điều này:

1. Chuyển sang tab **Design**.
2. Nhấp vào **Orientation in Editor** trên thanh công cụ.
3. Chọn **Create layout x-large Variation**.

- Một tab mới **xlarge/activity_main.xml** sẽ mở ra để chỉnh sửa bố cục dành riêng cho máy tính bảng.

1.6 Chỉnh sửa bố cục cho máy tính bảng

1. **Tắt Autoconnect** trong thanh công cụ.
2. **Xóa tất cả ràng buộc** trong bố cục.
3. **Thay đổi kích thước các phần tử UI:**
 - Chọn **TextView (show_count)**, kéo góc để thu nhỏ nó lại.
 - Chọn **Button (button_toast)**, thay đổi textSize thành 60sp, layout_width thành wrap_content.
 - Làm tương tự với **Button (button_count)**, sau đó kéo nút này lên phía trên **TextView**.

1.7 Sự ràng buộc baseline

Baseline constraint giúp căn chỉnh văn bản giữa các phần tử UI:

1. Ràng buộc button_toast vào cạnh trên và trái của bố cục.
2. Kéo button_count đến gần button_toast và ràng buộc vào cạnh trái của button_toast.
3. Nhấp vào baseline constraint button trên button_count, kéo đường ràng buộc đến button_toast để căn chỉnh văn bản theo dòng cơ sở.
- 4.

1.8 Mở rộng các nút theo chiều ngang

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

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

2. Nhấp vào nút pack trên thanh công cụ và chọn Expand Horizontally (Mở rộng theo chiều ngang), như trong hình minh họa bên dưới.

Các phần tử **Button** sẽ mở rộng theo chiều ngang để lấp đầy bố cục như hình minh họa bên dưới.

3. Để hoàn thành bố cục, hãy ràng buộc (**constraint**) **show_count TextView** vào phần dưới của nút **button_toast**, cũng như vào hai cạnh bên và cạnh dưới của bố cục, như được hiển thị trong hình minh họa động bên dưới.
4. Bước cuối cùng là thay đổi **layout_width** và **layout_height** của **show_count TextView** thành **Match Constraints** và đặt **textSize** thành **200sp**. Bố cục cuối cùng sẽ trông như hình minh họa bên dưới.
5. Nhấp vào nút **Orientation in Editor** trên thanh công cụ phía trên và chọn **Switch to Landscape**. Bố cục của máy tính bảng sẽ hiển thị ở chế độ ngang như hình minh họa bên dưới. (Bạn có thể chọn **Switch to Portrait** để quay lại chế độ dọc).
6. Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng màn hình sau khi chạy ứng dụng để xem giao diện hiển thị trên các thiết bị khác nhau. Bạn đã tạo thành công một ứng dụng có giao diện phù hợp trên điện thoại và máy tính bảng với các kích thước màn hình và mật độ điểm ảnh khác nhau.

Mẹo: Để có hướng dẫn chuyên sâu về cách sử dụng **ConstraintLayout**, hãy xem **Sử dụng ConstraintLayout để thiết kế giao diện của bạn**.

Giải pháp cho Task 1

Dự án Android Studio: HelloToast

Thử thách lập trình 1

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

Thử thách: Để phù hợp với chế độ ngang (landscape) trên máy tính bảng, bạn có thể căn giữa các phần tử **Button** trong tệp `activity_main.xml` (xlarge) để chúng hiển thị như trong hình bên dưới.

Gợi ý: Chọn các phần tử, nhấp vào nút **Align** trên thanh công cụ và chọn **Center Horizontally**.

Giải pháp cho Thử thách 1

Dự án Android Studio: HelloToastChallenge2

TASK 2 : THAY ĐỔI LAYOUT THÀNH LINEARLAYOUT

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

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

Một LinearLayout yêu cầu các thuộc tính sau:

- layout_width
- layout_height
- orientation

Các giá trị có thể có của layout_width và layout_height:

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

Giá trị của orientation:

- horizontal: Các view được sắp xếp từ trái sang phải.

- vertical: Các view được sắp xếp từ trên xuống dưới.

Trong bài tập này, bạn sẽ thay đổi **ConstraintLayout** (view group gốc) của ứng dụng **Hello Toast** thành **LinearLayout** để thực hành cách sử dụng **LinearLayout**.

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

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

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

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

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

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

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

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

2.2 Thay đổi thuộc tính phần tử cho **LinearLayout**

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

1. Mở ứng dụng Hello Toast từ bài trước.
2. Mở tệp activity_main.xml (nếu chưa mở), sau đó nhấp vào tab Text.
3. Tìm phần tử Button có ID button_toast và thay đổi 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 phần tử **button_count** và thay đổi 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 phần tử **show_count** (TextView) và thay đổi các thuộc tính sau:

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

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

app:layout_constraintBottom_toTopOf="@+id/button_count"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/button_toast"

9. Nhấp vào tab **Preview** ở phía bên phải của cửa sổ Android Studio (nếu chưa được chọn) để xem bản xem trước của bố cục cho đến thời điểm này.

2.3 Thay đổi vị trí của các phần tử trong **LinearLayout**

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

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

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

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

`<Button`

```
android:id="@+id/button_toast"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:layout_marginTop="8dp"
```

```
android:background="@color/colorPrimary"
android:onClick="showToast"
android:text="@string/button_label_toast"
android:textColor="@android:color/white" />
```

<TextView

```
android:id="@+id/show_count"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginBottom="8dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:layout_marginTop="8dp"
android:background="#FFFF00"
android:gravity="center_vertical"
android:text="@string/count_initial_value"
android:textAlignment="center"
android:textColor="@color/colorPrimary"
android:textSize="160sp"
android:textStyle="bold" />
```

<Button

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

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

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

2.4 Thêm thuộc tính **weight** cho phần tử **TextView**

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

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

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

- Thuộc tính **android:layout_weight** xác định mức độ phân bổ không gian trống còn lại trong **LinearLayout** cho một **View**. Nếu chỉ có một **View** có thuộc tính này, nó sẽ chiếm toàn bộ không gian trống. Nếu có nhiều **View**, không gian sẽ được chia tỷ lệ.

Ví dụ: Nếu mỗi **Button** có **weight** bằng 1 và **TextView** có **weight** bằng 2, tổng cộng là 4, thì mỗi **Button** sẽ chiếm $\frac{1}{4}$ không gian, còn **TextView** chiếm $\frac{1}{2}$ không gian.

Trên các thiết bị khác nhau, bố cục có thể hiển thị phần tử **show_count TextView** chiếm một phần hoặc hầu hết không gian giữa các nút **Toast** và **Count**. Để mở rộng **TextView** và lấp đầy không gian trống trên mọi thiết bị, hãy đặt thuộc tính **android:layout_weight** cho **TextView** theo các bước sau:

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

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

Bản xem trước bây giờ sẽ trông như hình minh họa sau.

Phần tử **show_count TextView** sẽ chiếm toàn bộ không gian giữa các nút **Toast** và **Count**. Bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ phía trên của khung xem trước và chọn một thiết bị khác.

Dù bạn chọn thiết bị nào để xem trước, phần tử **show_count TextView** vẫn sẽ mở rộng để lấp đầy toàn bộ không gian giữa các nút.

Giải pháp cho TASK 2

Mã XML trong **activity_main.xml** :

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

<Button

```
    android:id="@+id/button_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="@color/colorPrimary"
    android:onClick="showToast"
    android:text="@string/button_label_toast"
    android:textColor="@android:color/white" />
```

<TextView

```
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="#FFFF00"
    android:text="@string/count_initial_value"
    android:textAlignment="center"
```

```
android:textColor="@color/colorPrimary"
android:textSize="160sp"
android:textStyle="bold"
android:layout_weight="1"/>
```

```
<Button
```

```
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:onClick="countUp"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white" />
```

```
</LinearLayout>
```

TASK 3: THAY ĐỔI BỐ CỤC SANG RELATIVE LAYOUT

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

3.1 Thay đổi LinearLayout thành RelativeLayout

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

1. Mở tệp activity_main.xml và nhấp vào tab Text ở cuối khung chỉnh sửa để xem mã XML.
2. Thay đổi thẻ <LinearLayout ở đầu thành <RelativeLayout, để dòng mã trông như sau:

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

3. Cuộn xuống để đảm bảo rằng thẻ kết thúc </LinearLayout> cũng đã thay đổi thành </RelativeLayout>. Nếu chưa, hãy thay đổi nó 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 trong tab Text.

1. Nhấp vào tab Preview ở bên cạnh trình chỉnh sửa (nếu chưa được chọn) để xem bản xem trước bố cục, hiện trông giống như hình bên dưới.

Với sự thay đổi sang RelativeLayout, trình chỉnh sửa bố cục cũng đã thay đổi một số thuộc tính của các view.

Ví dụ:

- Nút Count (button_count) đè lên nút Toast (button_toast), khiến bạn không thể nhìn thấy nút Toast.
 - Phần trên của TextView (show_count) đè lên các nút Button.
2. Thêm thuộc tính android:layout_below cho nút button_count. Thuộc tính này giúp định vị nút Button ngay bên dưới TextView (show_count). Đây là một trong số các thuộc tính được sử dụng để định vị các view trong RelativeLayout, cho phép bạn sắp xếp các view dựa trên vị trí của các view khác.
android:layout_below="@+id/show_count"
 3. Thêm thuộc tính android:layout_centerHorizontal vào cùng nút **Button** để căn giữa nó theo chiều ngang bên trong **RelativeLayout**, tức là căn giữa trong **view group** cha của nó.

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

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

```
<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    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 **show_count** TextView:

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

The android:layout_alignParentLeft căn chỉnh view về phía bên trái của **RelativeLayout** – nhóm view cha. Mặc dù thuộc tính này đủ để căn chỉnh view sang bên trái, nhưng bạn có thể muốn căn chỉnh nó sang bên 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** đảm bảo rằng cạnh "bắt đầu" của view này khớp với cạnh bắt đầu của view cha. "Bắt đầu" là cạnh bên trái của màn hình

nếu cài đặt ngôn ngữ là từ trái sang phải, hoặc là cạnh bên phải nếu cài đặt ngôn ngữ là từ phải sang trái.

5. Xóa thuộc tính **android:layout_weight="1"** khỏi **show_count** TextView, vì nó không liên quan đến **RelativeLayout**. Bản xem trước của bố cục bây giờ sẽ trông giống như hình minh họa sau đây.

Mẹo: RelativeLayout giúp dễ dàng sắp xếp nhanh chóng các phần tử giao diện người dùng trong bố cục. Để tìm hiểu thêm về cách định vị các view trong **RelativeLayout**, hãy xem phần **“Positioning Views”** trong chủ đề **“Relative Layout”** của **API Guide**.

Giải pháp cho TASK 3

Mã xml trong activity_main

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

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

```
android:background="@color/colorPrimary"
android:onClick="showToast"
android:text="@string/button_label_toast"
android:textColor="@android:color/white" />
```

<TextView

```
android:id="@+id/show_count"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:gravity="center_vertical"
android:layout_marginBottom="8dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:layout_marginTop="8dp"
android:background="#FFFF00"
android:text="@string/count_initial_value"
android:textAlignment="center"
android:textColor="@color/colorPrimary"
android:textSize="160sp"
android:textStyle="bold"
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="8dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:background="@color/colorPrimary"
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 biến thể:**
- Để xem trước bố cục ứng dụng với hướng ngang trong trình chỉnh sửa bố cục, nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng và chọn **Switch to Landscape**. Chọn **Switch to Portrait** để quay lại hướng dọc.
- Để tạo một biến thể bố cục khác cho hướng ngang, nhấp vào nút **Orientation in Editor** và chọn **Create Landscape Variation**. Một cửa sổ chỉnh sửa mới sẽ mở ra với tab **land/activity_main.xml** hiển thị bố cục cho hướng ngang.
- Để xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị hoặc trình giả lập, nhấp vào nút **Device in Editor** trên thanh công cụ trên cùng và chọn một thiết bị.

- Để tạo một biến thể bố cục khác dành cho máy tính bảng (màn hình lớn hơn), nhấp vào nút **Orientation in Editor** và chọn **Create layout x-large Variation**. Một cửa sổ chỉnh sửa mới sẽ mở ra với tab **xlarge/activity_main.xml** hiển thị bố cục cho thiết bị có kích thước máy tính bảng.
-

- **Sử dụng ConstraintLayout:**

- Để xóa tất cả các ràng buộc trong bố cục với **ConstraintLayout**, nhấp vào nút **Clear All Constraints** trên thanh công cụ.
 - Bạn có thể căn chỉnh một phần tử giao diện người dùng chứa văn bản (ví dụ: **TextView** hoặc **Button**) với một phần tử giao diện người dùng khác cũng chứa văn bản. **Baseline constraint** cho phép bạn ràng buộc các phần tử sao cho dòng cơ sở của văn bản khớp nhau.
 - Để tạo **baseline constraint**, di chuột qua phần tử giao diện người dùng cho đến khi nút **baseline constraint** xuất hiện bên dưới phần tử.
 - Nút **pack** trên thanh công cụ cung cấp các tùy chọn để gói hoặc mở rộng các phần tử giao diện người dùng đã chọn. Bạn có thể sử dụng nó để căn chỉnh đều các nút **Button** theo chiều ngang trong bố cục.
-

- **Sử dụng LinearLayout:**

- **LinearLayout** là một **ViewGroup** sắp xếp tập hợp các **View** của nó theo hàng ngang hoặc hàng dọc.
- **LinearLayout** yêu cầu có các thuộc tính: **layout_width**, **layout_height**, và **orientation**.
- **match_parent** cho **layout_width** hoặc **layout_height**: Mở rộng **View** để lấp đầy **View** cha của nó theo chiều rộng hoặc chiều cao. Khi **LinearLayout** là **View** gốc, nó sẽ mở rộng theo kích thước của màn hình (**View** cha).
- **wrap_content** cho **layout_width** hoặc **layout_height**: Thu nhỏ kích thước **View** sao cho vừa đủ để chứa nội dung của nó. Nếu không có nội dung, **View** sẽ trở nên vô hình.

- **Số dp cố định (density-independent pixels)** cho **layout_width** hoặc **layout_height**: Xác định một kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ, **16dp** có nghĩa là 16 pixel độc lập với mật độ.
 - **orientation** của **LinearLayout** có thể là **horizontal** (sắp xếp các phần tử từ trái sang phải) hoặc **vertical** (sắp xếp các phần tử từ trên xuống dưới).
 - Việc chỉ định các thuộc tính **gravity** và **weight** giúp bạn kiểm soát bổ sung việc sắp xếp các **View** và nội dung trong **LinearLayout**.
 - **android:gravity** xác định vị trí căn chỉnh nội dung của **View** trong chính nó.
 - **android:layout_weight** xác định lượng không gian bổ sung trong **LinearLayout** sẽ được phân bổ cho **View**. Nếu chỉ một **View** có thuộc tính này, nó sẽ nhận tất cả không gian trống. Nếu nhiều **View** có thuộc tính này, không gian sẽ được phân bổ theo tỷ lệ. Ví dụ, nếu hai nút **Button** có trọng số là **1** và một **TextView** có trọng số là **2**, tổng trọng số là **4**, mỗi nút **Button** sẽ nhận $\frac{1}{4}$ không gian, và **TextView** nhận $\frac{1}{2}$ không gian.
-

- **Sử dụng RelativeLayout:**
- **RelativeLayout** là một **ViewGroup** trong đó mỗi **View** được định vị và căn chỉnh tương đối với các **View** khác trong nhóm.
- Sử dụng **android:layout_alignParentTop** để căn chỉnh **View** lên trên cùng của **View** cha.
- Sử dụng **android:layout_alignParentLeft** để căn chỉnh **View** với cạnh trái của **View** cha.
- Sử dụng **android:layout_alignParentStart** để cạnh bắt đầu của **View** khớp với cạnh bắt đầu của **View** cha. Thuộc tính này hữu ích nếu bạn muốn ứng dụng của mình hoạt động trên các thiết bị sử dụng ngôn ngữ hoặc cài đặt khu vực khác nhau.
 - **start** là cạnh trái của màn hình nếu ngôn ngữ được đặt từ trái sang phải (LTR).
 - **start** là cạnh phải của màn hình nếu ngôn ngữ được đặt từ phải sang trái (RTL).

Các khái niệm liên quan

Tài liệu liên quan đến các khái niệm này có trong **1.2: Layouts and resources for the UI**.

- **Tìm hiểu thêm**

Tài liệu về Android Studio:

- [Giới thiệu về Android Studio](#)
- [Tạo biểu tượng ứng dụng với Image Asset Studio](#)

Tài liệu dành cho nhà phát triển Android:

- [Layouts](#)
- [Xây dựng giao diện người dùng với Layout Editor](#)
- [Tạo giao diện linh hoạt với ConstraintLayout](#)
- [LinearLayout](#)
- [RelativeLayout](#)
- [View](#)
- [Button](#)
- [TextView](#)
- [Hỗ trợ các mật độ điểm ảnh khác nhau](#)

Tài liệu khác:

- [Codelabs: Sử dụng ConstraintLayout để thiết kế giao diện Android](#)
- [Bảng thuật ngữ về từ vựng và khái niệm](#)

-
- **Bài tập về nhà**
 - **Thay đổi ứng dụng**

Mở ứng dụng **HelloToast** và thực hiện các thay đổi sau:

1. Đổi tên dự án thành **HelloConstraint**, sau đó tái cấu trúc (refactor) dự án để đổi tên thành **Hello Constraint**.
 - (Để biết hướng dẫn sao chép và tái cấu trúc dự án, xem phần **Appendix: Utilities**).
2. Sửa đổi bố cục trong **activity_main.xml** để căn chỉnh các nút **Toast** và **Count** theo cạnh trái của **TextView show_count** (hiển thị số "0").
 - (Tham khảo hình minh họa trong tài liệu).
3. Thêm một nút thứ ba có tên **Zero**, nằm giữa các nút **Toast** và **Count**.
4. Sắp xếp các nút **Button** theo chiều dọc, phân bố đều từ trên xuống dưới xung quanh **TextView show_count**.
5. Đặt màu nền ban đầu của nút **Zero** thành **màu xám**.
6. Đảm bảo rằng nút **Zero** cũng có mặt trong bố cục **land/activity_main.xml** (hướng ngang) và **xlarge/activity_main.xml** (bố cục dành cho màn hình lớn như máy tính bảng).
7. Khi nhấn nút **Zero**, giá trị của **TextView show_count** phải được đặt về **0**.
8. Cập nhật trình xử lý sự kiện (click handler) cho nút **Count** để thay đổi màu nền của chính nó, tùy thuộc vào giá trị **chẵn hoặc lẻ** của bộ đếm.
 - **Gợi ý:** Không sử dụng `findViewById` để tìm nút **Count**. Có phương thức nào khác để làm điều này không?
9. Cập nhật trình xử lý sự kiện cho nút **Count** để khi được nhấn, màu nền của nút **Zero** cũng thay đổi thành màu khác (không phải màu xám) để cho biết nút **Zero** đã được kích hoạt.
 - **Gợi ý:** Có thể sử dụng `findViewById` trong trường hợp này.
10. Cập nhật trình xử lý sự kiện cho nút **Zero** để khi nhấn, nó sẽ đặt lại màu nền về **màu xám** nếu giá trị bộ đếm là **0**.

Trả lời các câu hỏi

Câu hỏi 1 Hai thuộc tính ràng buộc bố cục nào trên nút **Zero** giúp căn chỉnh nó theo khoảng cách đều giữa hai nút **Toast** và **Count**? (Chọn 2 câu trả lời.)

app:layout_constraintBottom_toTopOf="@+id/button_count"

app:layout_constraintTop_toBottomOf="@+id/button_toast"

Câu hỏi 2 Thuộc tính ràng buộc bố cục nào trên nút **Zero** giúp căn chỉnh nó theo chiều ngang với hai nút còn lại?

app:layout_constraintLeft_toLeftOf="parent"

Câu hỏi 3

Chữ ký (signature) nào là đúng cho một phương thức sử dụng thuộc tính **android:onClick** trong XML?

public void callMethod(View view)

Câu hỏi 4

Trong trình xử lý sự kiện của nút **Count** với phương thức:

public void countUp(View view)

Kỹ thuật nào là hiệu quả nhất để thay đổi màu nền của nút **Count**?

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

view.setBackgroundColor()

- **Hướng dẫn chấm điểm ứng dụng**

Kiểm tra ứng dụng để đảm bảo rằng nó có các tính năng sau:

Hiển thị nút Zero.Nút Zero nằm giữa nút Toast và Count.Ứng dụng bao gồm các tệp bố cục:

- activity_main.xml
- activity_main.xml (land) (cho chế độ ngang)
- activity_main.xml (xlarge) (cho màn hình lớn)**Nút Zero có trình xử lý sự kiện để đặt lại bộ đếm về 0.**
- Giá trị trong TextView show_count hiển thị số **0**.
- Màu nền của nút **Zero** được đặt lại thành **xám**.**Trình xử lý sự kiện của nút Count được cập nhật:**
- **Thay đổi màu nền của chính nó** dựa vào giá trị **chẵn/lẻ** của bộ đếm.
- **Sử dụng tham số view trong phương thức** để thay đổi màu nền.
- **Thay đổi màu nền của nút Zero thành một màu khác (không phải xám)** khi giá trị bộ đếm tăng.

1.4) Văn bản và các chế độ cuộn

Giới thiệu

Lớp TextView là một lớp con của lớp View, được sử dụng để hiển thị văn bản trên màn hình. Bạn có thể kiểm soát cách hiển thị văn bản bằng cách sử dụng các thuộc tính của TextView trong tệp bố cục XML.

Bài thực hành này sẽ hướng dẫn cách làm việc với nhiều phần tử TextView, bao gồm một phần tử cho phép người dùng cuộn nội dung theo chiều dọc.

Nếu bạn có nhiều nội dung hơn mức có thể hiển thị trên màn hình của thiết bị, bạn có thể tạo một chế độ xem cuộn (scrolling view) để người dùng có thể cuộn theo chiều dọc bằng cách vuốt lên hoặc xuống, hoặc cuộn theo chiều ngang bằng cách vuốt sang trái hoặc phải.

Thông thường, bạn sẽ sử dụng scrolling view cho các tin tức, bài viết hoặc bất kỳ nội dung văn bản dài nào không thể hiển thị hết trên màn hình. Ngoài ra, bạn cũng có thể sử dụng scrolling view để cho phép người dùng nhập nhiều dòng văn bản hoặc để kết hợp các phần tử giao diện người dùng (chẳng hạn như trường nhập văn bản và nút bấm) trong một chế độ xem cuộn.

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

Các bố cục phức tạp có thể gặp vấn đề về hiệu suất khi chứa các phần tử con như hình ảnh. Một lựa chọn tốt cho một **View** bên trong **ScrollView** là **LinearLayout** được sắp xếp theo hướng dọc, hiển thị các phần tử mà người dùng có thể cuộn qua (chẳng hạn như các phần tử **TextView**).

Với **ScrollView**, tất cả các phần tử giao diện người dùng đều được lưu trữ trong bộ nhớ và nằm trong hệ thống phân cấp của chế độ xem, ngay cả khi chúng không được hiển thị trên màn hình. Điều này làm cho **ScrollView** trở thành một lựa chọn lý tưởng để cuộn qua các trang văn bản tự do một cách mượt mà, vì nội dung văn bản đã có sẵn trong bộ nhớ.

Tuy nhiên, **ScrollView** có thể tiêu tốn nhiều bộ nhớ, ảnh hưởng đến hiệu suất của ứng dụng. Nếu bạn cần hiển thị danh sách dài các mục mà người dùng có thể thêm, xóa hoặc chỉnh sửa, hãy cân nhắc sử dụng **RecyclerView**, được giới thiệu trong một bài học khác.

- **Những gì bạn nên biết trước**

Bạn nên có khả năng:

- Tạo ứng dụng **Hello World** bằng Android Studio.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị thật.
- Triển khai một **TextView** trong bố cục ứng dụng.
- Tạo và sử dụng tài nguyên chuỗi (**string resources**).
- **Những gì bạn sẽ học**
- Cách sử dụng XML để thêm nhiều phần tử **TextView**.
- Cách sử dụng XML để định nghĩa một **View** có thể cuộn (**scrolling view**).
- Cách hiển thị văn bản định dạng tự do với một số thẻ HTML.
- Cách thay đổi màu nền và màu chữ của **TextView**.
- Cách thêm 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 tiêu đề phụ của bài viết.
- Sử dụng các kiểu **TextAppearance** và màu sắc cho tiêu đề và tiêu đề phụ.
- Sử dụng thẻ HTML trong chuỗi văn bản để kiểm soát định dạng.
- Sử dụng thuộc tính **lineSpacingExtra** để thêm khoảng cách dòng giúp dễ đọc hơn.
- Thêm một **ScrollView** vào bố cục để cho phép cuộn một phần tử **TextView**.
- Thêm thuộc tính **autoLink** để kích hoạt các URL trong văn bản trở thành liên kết có thể nhấp.

Tổng quan về ứng dụng

Ứng dụng **Scrolling Text** trình bày cách sử dụng thành phần giao diện **ScrollView**. **ScrollView** là một **ViewGroup**, trong ví dụ này chứa một **TextView**. Nó hiển thị một trang văn bản dài—trong trường hợp này là bài đánh giá album nhạc—mà người dùng

có thể cuộn dọc bằng cách vuốt lên hoặc xuống. Một thanh cuộn sẽ xuất hiện ở lề phải.

Ứng dụng cũng minh họa cách sử dụng văn bản được định dạng với một số thẻ HTML tối thiểu để đặt chữ **đậm** hoặc *ngghiêng*, và sử dụng ký tự xuống dòng để tách đoạn văn. Bạn cũng có thể thêm các liên kết web có thể nhấp vào văn bản.

Trong hình trên, các yếu tố sau xuất hiện:

1. **Một liên kết web có thể nhấp** được nhúng trong văn bản định dạng tự do.
2. **Thanh cuộn** xuất hiện khi cuộn văn bản.

TASK 1 : THÊM VÀ CHỈNH SỬA CÁC PHẦN TỬ TEXTVIEW

Trong bài thực hành này, bạn sẽ tạo một dự án Android cho ứng dụng **ScrollingText**, thêm các phần tử **TextView** vào bố cục để hiển thị tiêu đề và phụ đề của bài viết, đồng thời thay đổi phần tử **TextView** mặc định “Hello World” để hiển thị một bài viết dài.

Hình minh họa bên dưới là sơ đồ của bố cục ứng dụng.

Bạn sẽ thực hiện tất cả các thay đổi này trong mã **XML** và tệp **strings.xml**. Bạn sẽ chỉnh sửa mã **XML** của bố cục trong **Text pane**, bằng cách nhấp vào tab **Text**, thay vì sử dụng tab **Design** để hiển thị giao diện thiết kế. Một số thay đổi đối với các phần tử giao diện người dùng (**UI elements**) và thuộc tính (**attributes**) sẽ dễ dàng thực hiện hơn khi chỉnh sửa trực tiếp trong **Text pane** bằng mã nguồn **XML**.

1.1 Tạo dự án và các phần tử TextView

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

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

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

Attribute	Value
Application Name	Scrolling Text
Company Name	android.example.com (or your own domain)
Phone and Tablet Minimum SDK	API15: Android 4.0.3 IceCreamSandwich
Template	Empty Activity
Generate Layout File checkbox	Selected
Backwards Compatibility (AppCompat) checkbox	Selected

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

Ở phần trên cùng, hay còn gọi là **root**, của hệ thống phân cấp **View** là **ConstraintLayout** – một **ViewGroup** dùng để bố trí giao diện.

`android.support.constraint.ConstraintLayout`

3. Thay đổi **ViewGroup** từ **ConstraintLayout** thành **RelativeLayout**.

Dòng thứ hai trong mã XML bây giờ sẽ trông giống như sau:

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

RelativeLayout cho phép bạn đặt các phần tử giao diện người dùng (**UI elements**) tương đối với nhau hoặc tương đối với chính **RelativeLayout**.

Mặc dù phần tử **TextView** mặc định "Hello World" được tạo bởi mẫu **Empty Layout** vẫn có các thuộc tính ràng buộc (**constraint attributes**) như:

```
app:layout_constraintBottom_toBottomOf="parent"
```

Đừng lo lắng—bạn sẽ xóa chúng trong bước tiếp theo.

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

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

Sau khi xóa, phần đầu của mã XML sẽ trông như sau:

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

android:layout_width="match_parent"

android:layout_height="match_parent"

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

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

TextView #1 attribute	Value
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:id	"@+id/article_heading"
android:background	"@color/colorPrimary"
android:textColor	"@android:color/white"
android:padding	"10dp"
android:textAppearance	"@android:style/TextAppearance.DeviceDefault.Large"
android:textStyle	"bold"
android:text	"Article Title"

7. Trích xuất tài nguyên chuỗi cho thuộc tính android:text

- Đặt con trỏ vào chuỗi cứng "Article Title" trong phần tử **TextView**.
- Nhấn **Alt + Enter** (hoặc **Option + Enter** trên Mac).
- Chọn **Extract string resource**.
- Đảm bảo rằng tùy chọn **Create the resource in directories** được chọn.
- Chỉnh sửa tên tài nguyên thành **article_title**.
- Tài nguyên chuỗi mới sẽ được tạo trong **strings.xml**.

8. Trích xuất tài nguyên kích thước cho thuộc tính android:padding

- Đặt con trỏ vào giá trị "10dp" trong thuộc tính android:padding của **TextView**.

- Nhấn **Alt + Enter** (hoặc **Option + Enter** trên Mac).
- Chọn **Extract dimension resource**.
- Đảm bảo rằng tùy chọn **Create the resource in directories** được chọn.
- Chỉnh sửa tên tài nguyên thành **padding_regular**.
- Tài nguyên kích thước mới sẽ được tạo trong **dimens.xml**.

9. Thêm một phần tử **TextView** mới

- Thêm một **TextView** mới phía trên phần tử "Hello World" và bên dưới **TextView** đã tạo trước đó.
- Thêm các thuộc tính cần thiết vào **TextView** này.

TextView #2 Attribute	Value
layout_width	"match_parent"
layout_height	"wrap_content"
android:id	"@+id/article_subheading"
android:layout_below	"@id/article_heading"
android:padding	"@dimen/padding_regular"
android:textAppearance	"@android:style/TextAppearance.DeviceDefault"
android:text	"Article Subtitle"

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

10. Trích xuất tài nguyên chuỗi cho thuộc tính `android:text`

- Đặt con trỏ vào chuỗi cứng "Article Subtitle" trong **TextView**.
- Nhấn **Alt + Enter** (hoặc **Option + Enter** trên Mac).
- Chọn **Extract string resource**.
- Đặt tên tài nguyên là **article_subtitle**.

11. Xóa các thuộc tính `layout_constraint` trong phần tử "Hello World" **TextView**

- Xóa các dòng sau:

```
app:layout_constraintBottom_toBottomOf="parent"
```

app:layout_constraintLeft_toLeftOf="parent"

app:layout_constraintRight_toRightOf="parent"

app:layout_constraintTop_toTopOf="parent"

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

TextView Attribute	Value
android:id	"@+id/article"
android:layout_below	"@id/article_subheading"
android:lineSpacingExtra	"5sp"
android:padding	"@dimen/padding_regular"
android:text	Change to "Article text"

13. Trích xuất tài nguyên chuỗi và kích thước:

- Trích xuất tài nguyên chuỗi cho "Article text" và đặt tên là **article_text**.
- Trích xuất tài nguyên kích thước cho "5sp" và đặt tên là **line_spacing**.

14. Định dạng lại và căn chỉnh mã nguồn:

- Chọn **Code > Reformat Code** trong Android Studio.
- Việc định dạng lại mã giúp mã nguồn dễ đọc, rõ ràng hơn, đồng thời cải thiện khả năng bảo trì cho bạn và những người khác.

1.2 Thêm văn bản của bài báo

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

Trong bài thực hành này, bạn sẽ tạo bài báo dưới dạng một chuỗi văn bản dài trong tệp tài nguyên **strings.xml**.

1. Trong thư mục **app > res > values**, mở tệp **strings.xml**.
2. Mở bất kỳ tệp văn bản nào có nội dung dài hoặc mở tệp **strings.xml** của ứng dụng **ScrollingText** đã hoàn thành.
3. Nhập giá trị cho chuỗi **article_title** và **article_subtitle** với tiêu đề và phụ đề do bạn tạo ra hoặc sử dụng giá trị có sẵn trong tệp **strings.xml** của ứng dụng **ScrollingText** đã hoàn thành. Hãy đảm bảo rằng các chuỗi này chỉ có một dòng và không chứa 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 từ tệp văn bản của mình hoặc sử dụng văn bản có sẵn trong tệp **strings.xml** của ứng dụng **ScrollingText** đã hoàn thành. Yêu cầu duy nhất là văn bản phải đủ dài để không thể hiển thị hết trên màn hình.

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

- Khi nhập hoặc dán văn bản vào tệp **strings.xml**, các dòng văn bản sẽ không tự động xuống dòng mà sẽ kéo dài ra ngoài lề phải. Đây là hành vi đúng—mỗi dòng văn bản mới bắt đầu từ lề trái thể hiện một đoạn văn hoàn chỉnh. Nếu bạn muốn văn bản trong **strings.xml** xuống dòng, bạn có thể nhấn **Enter** để tạo ngắt dòng cứng hoặc định dạng văn bản trước trong trình soạn thảo với các ngắt dòng cứng.
- Nhập **\n** để đại diện cho ký tự kết thúc dòng và nhập thêm một **\n** để tạo một dòng trống. Bạn cần thêm các ký tự xuống dòng này để các đoạn văn không bị dính vào nhau.
- Nếu trong văn bản có dấu nháy đơn ('), bạn phải đặt ký tự gạch chéo ngược (') phía trước để tránh lỗi.
- Nếu trong văn bản có dấu nháy kép ("), bạn cũng phải đặt ký tự gạch chéo ngược (") phía trước.
- Bạn cũng cần thoát (escape) bất kỳ ký tự nào không thuộc bảng mã ASCII. Để biết thêm chi tiết, hãy xem phần **Định dạng và tạo kiểu** trong tài liệu về tài nguyên chuỗi (**String resources**).

Quy tắc định dạng văn bản:

- Sử dụng thẻ HTML **** và **** để làm đậm các từ cần nhấn mạnh.

- Sử dụng thẻ HTML `<i>` và `</i>` để in nghiêng các từ cần nhấn mạnh. Nếu bạn sử dụng dấu nháy đơn 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 lồng các thẻ, như `<i>... văn bản ...</i>`.
- Các thẻ HTML khác sẽ bị bỏ qua.
- **Cách đặt văn bản trong strings.xml:**
- Bao bọc toàn bộ nội dung bài viết trong thẻ:
- `<string name="article_text">Nội dung bài viết của bạn...</string>`
- Nếu muốn thêm một liên kết web để kiểm tra, hãy sử dụng đường dẫn trần (ví dụ: www.google.com).
- **Không** sử dụng thẻ HTML để tạo liên kết vì tất cả các thẻ HTML khác (ngoại trừ `` và `<i>`) sẽ bị bỏ qua và hiển thị dưới dạng văn bản thông thường, điều này không đúng với mục đích của bạn.

1.3 Chạy ứng dụng

Hãy chạy ứng dụng. Lúc này, bài báo sẽ hiển thị, nhưng người dùng **không thể cuộn** bài báo vì bạn **chưa** thêm **ScrollView** (bạn sẽ thực hiện điều này trong bước tiếp theo).

Ngoài ra, hãy lưu ý rằng khi nhấn vào liên kết web, **không có gì xảy ra**. Bạn cũng sẽ khắc phục vấn đề này trong bước tiếp theo.

Mã giải pháp cho Task 1

Tập bố cục **activity_main.xml** sẽ có dạng như sau:

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

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

```
xmlns:tools="http://schemas.android.com/tools"
```

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

```
android:layout_height="match_parent"

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

```
<TextView
```

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

```
<TextView
```

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

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

</RelativeLayout>
```

TASK 2 : THÊM SCROLLVIEW VÀ KÍCH HOẠT WEB LINK

Trong nhiệm vụ trước, bạn đã tạo ứng dụng **ScrollingText** với các phần tử **TextView** để hiển thị tiêu đề bài viết, phụ đề và nội dung văn bản dài. Bạn cũng đã thêm một liên kết web, nhưng liên kết này **chưa hoạt động**. Trong bước này, bạn sẽ thêm mã để kích hoạt liên kết đó.

Ngoài ra, **TextView** không thể tự cuộn nội dung bài viết để hiển thị toàn bộ văn bản. Vì vậy, bạn sẽ thêm một **ViewGroup** mới có tên **ScrollView** vào bố cục XML để làm cho **TextView** có thể cuộn được.

2.1 Thêm thuộc tính autoLink để kích hoạt liên kết web

Hãy thêm thuộc tính `android:autoLink="web"` vào **TextView** hiển thị bài viết (article).

Mã XML cho TextView này bây giờ sẽ trông như sau:

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

2.2 Thêm ScrollView vào bố cục

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

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

Bây giờ, mã cho hai phần tử TextView và ScrollView trông như thế này:

```

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

<ScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"></ScrollView>

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

```

3. Di chuyển thẻ đóng `</ScrollView>` xuống sau article TextView để toàn bộ thuộc tính của article TextView nằm bên trong ScrollView.
4. Xóa thuộc tính sau từ article TextView và thêm nó vào ScrollView.

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

5. Chọn **Code > Reformat Code** để định dạng lại mã XML sao cho article TextView được thụt vào bên trong mã `<ScrollView>`.
6. Nhấp vào tab **Preview** ở bên phải trình chỉnh sửa bố cục để xem trước giao diện.
Bố cục bây giờ sẽ trông giống như phần bên phải của hình dưới đây.

2.3 Chạy ứng dụng

Để kiểm tra cách văn bản cuộn:

1. **Chạy ứng dụng** trên thiết bị hoặc trình giả lập.
 - Vuốt lên và xuống để cuộn bài viết. Thanh cuộn sẽ xuất hiện ở lề phải khi bạn cuộn.

- Nhấn vào liên kết web để mở trang web. Thuộc tính android:autoLink sẽ tự động biến bất kỳ URL nào trong TextView (chẳng hạn như www.rockument.com) thành liên kết có thể nhấp.

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

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

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

- Lưu ý rằng chế độ xem cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn chính xác.

Trong hình trên, xuất hiện các yếu tố sau:

1. Một liên kết web có thể nhấp được nhúng trong văn bản tự do.
2. Thanh cuộn xuất hiện khi cuộn văn bản.

Mã giải pháp cho Task 2

Mã XML cho bố cục với ScrollView như sau:

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

```
<TextView
    android:id="@+id/article_heading"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/colorPrimary"
```

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

<TextView

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

<ScrollView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/article_subheading">
```

<TextView

```
android:id="@+id/article"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:autoLink="web"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_text" />
</ScrollView>
</RelativeLayout>
```

TASK 3 : CUỘN NHIỀU PHẦN TỬ

Như đã đề cập trước đó, ScrollView chỉ có thể chứa một View con duy nhất (chẳng hạn như article TextView mà bạn đã tạo). Tuy nhiên, View đó có thể là một ViewGroup khác chứa nhiều phần tử View, chẳng hạn như LinearLayout.

Bạn có thể **lồng ghép** một ViewGroup như LinearLayout bên trong ScrollView, giúp cuộn tất cả nội dung bên trong LinearLayout.

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

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

3.1 Thêm một LinneatLayout vào ScrollView

1. Mở tệp activity_main.xml trong dự án **ScrollingText**, sau đó chọn tab **Text** để chỉnh sửa mã XML (nếu chưa được chọn).
2. Thêm một LinearLayout phía trên article TextView bên trong ScrollView.

- Khi bạn nhập `<LinearLayout`, Android Studio sẽ tự động thêm `</LinearLayout>` vào cuối và hiển thị các thuộc tính `android:layout_width` và `android:layout_height` với các gợi ý.
- Chọn `match_parent` cho chiều rộng và `wrap_content` cho chiều cao từ các gợi ý.

Bây giờ, phần đầu của mã trong `ScrollView` sẽ trông như sau:

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

Bạn sử dụng `match_parent` để chiều rộng của nó khớp với `ViewGroup` cha. Bạn sử dụng `wrap_content` để `LinearLayout` chỉ có kích thước đủ lớn để chứa nội dung bên trong.

Thực hiện các bước sau:

3. Di chuyển thẻ đóng `</LinearLayout>` xuống sau `article TextView` nhưng **trước** thẻ đóng `</ScrollView>`.
 - Bây giờ, `LinearLayout` sẽ bao gồm `article TextView` và hoàn toàn nằm bên trong `ScrollView`.
4. Thêm thuộc tính `android:orientation="vertical"` vào `LinearLayout` để thiết lập hướng dọc.
5. Chọn **Code > Reformat Code** để căn chỉnh mã XML đúng cách.

Bây giờ, `LinearLayout` sẽ trông như sau:

<ScrollView

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_subheading">
```

<LinearLayout

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
```

<TextView

```
    android:id="@+id/article"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:autoLink="web"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />
```

</LinearLayout>

</ScrollView>

3.2 Di chuyển các phần tử giao diện vào LinearLayout

Hiện tại, LinearLayout chỉ chứa một phần tử giao diện—article TextView. Bạn cần thêm article_subheading TextView vào LinearLayout để cả hai có thể cuộn cùng nhau.

Thực hiện các bước sau:

1. Di chuyển `article_subheading` `TextView` vào `LinearLayout`

- Chọn đoạn mã của `article_subheading` `TextView`, sau đó chọn **Edit > Cut**.
- Nhấp vào vị trí phía trên `article` `TextView` bên trong `LinearLayout`, sau đó chọn **Edit > Paste**.

2. Xóa thuộc tính `android:layout_below="@id/article_heading"` khỏi `article_subheading` `TextView`

- Vì `TextView` này đã nằm trong `LinearLayout`, thuộc tính trên sẽ gây xung đột với các thuộc tính của `LinearLayout`.

3. Chỉnh sửa thuộc tính `layout` của `ScrollView`

- Thay đổi thuộc tính `android:layout_below="@id/article_subheading"` thành `android:layout_below="@id/article_heading"`.
- Vì tiêu đề phụ (`article_subheading`) giờ đã thuộc `LinearLayout`, `ScrollView` cần được đặt bên dưới `article_heading`, thay vì `article_subheading`.

Bây giờ, mã XML cho `ScrollView` sẽ trông như sau:

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

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

```
<TextView
```

```
    android:id="@+id/article"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:autoLink="web"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />
```

```
</LinearLayout>
```

```
</ScrollView>
```

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