

**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 .....	63
1.4) Văn bản và các chế độ cuộn .....	63
1.5) Tài nguyên có sẵn .....	63
Bài 2) Activities .....	63
2.1) Activity và Intent .....	63
2.2) Vòng đời của Activity và trạng thái .....	63
2.3) Intent ngầm định .....	63
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ .....	63
3.1) Trình gỡ lỗi .....	63
3.2) Kiểm thử đơn vị .....	63
3.3) Thư viện hỗ trợ .....	63

## 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.

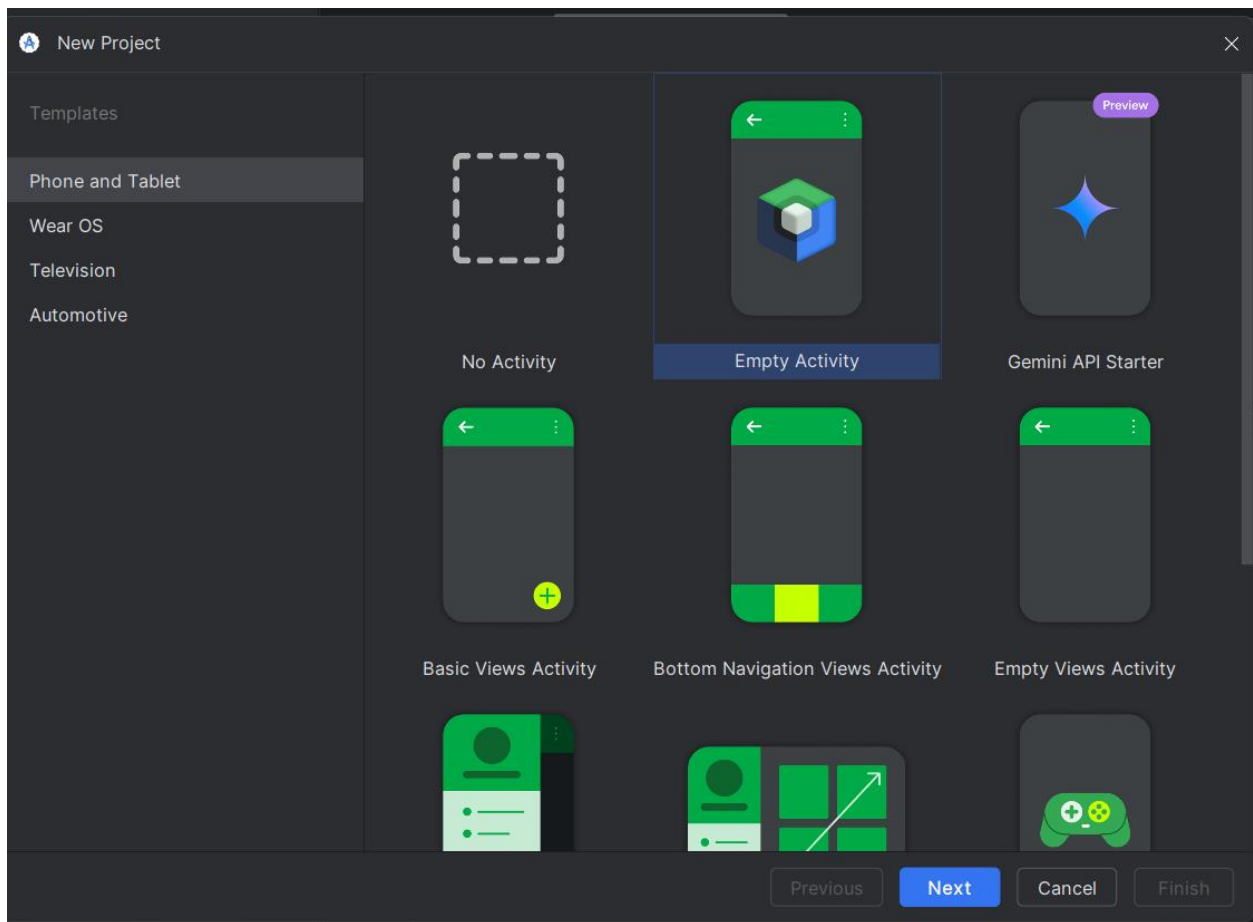
## Task 2 : Tạo ứng dụng Hello World

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.

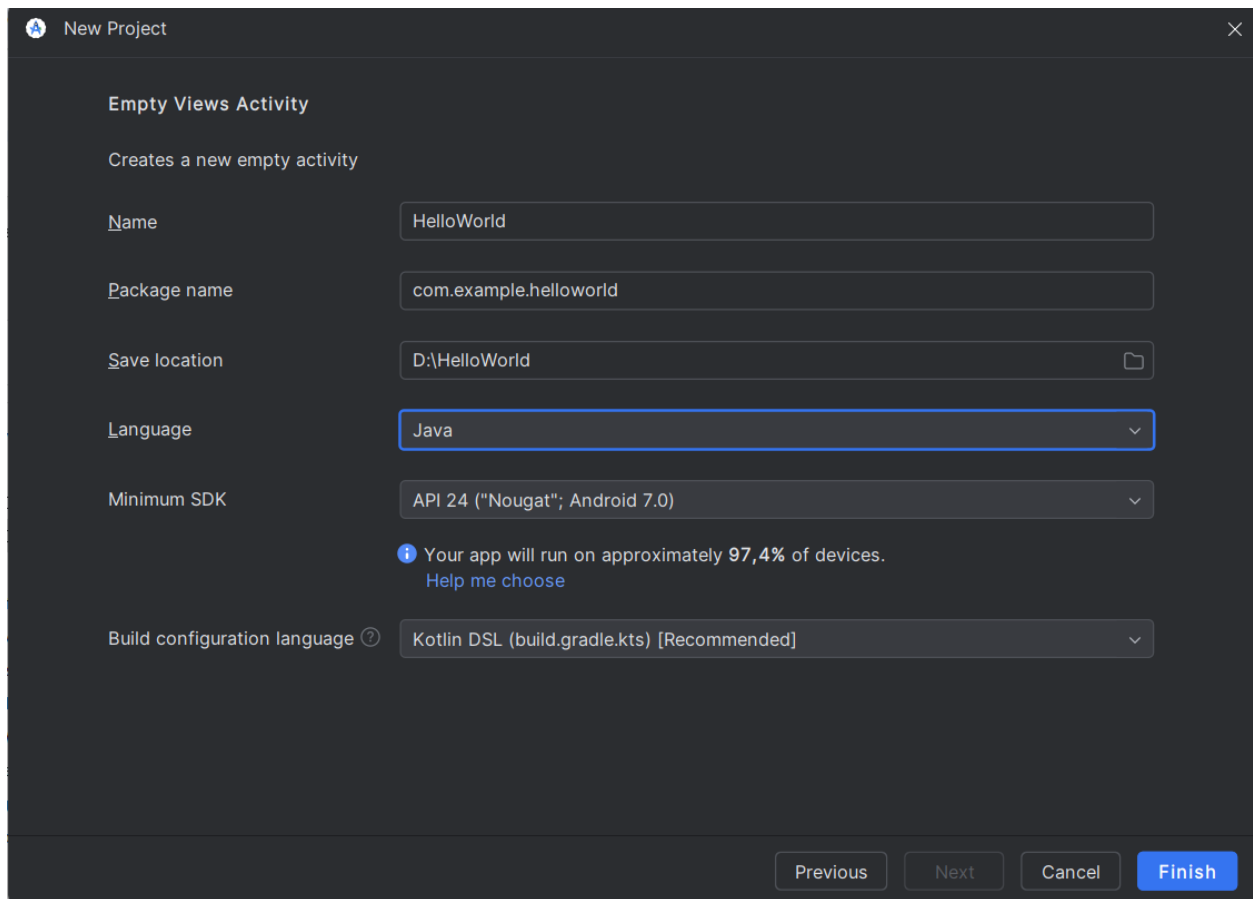
## 2.1 Tạo dự án ứng dụng

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

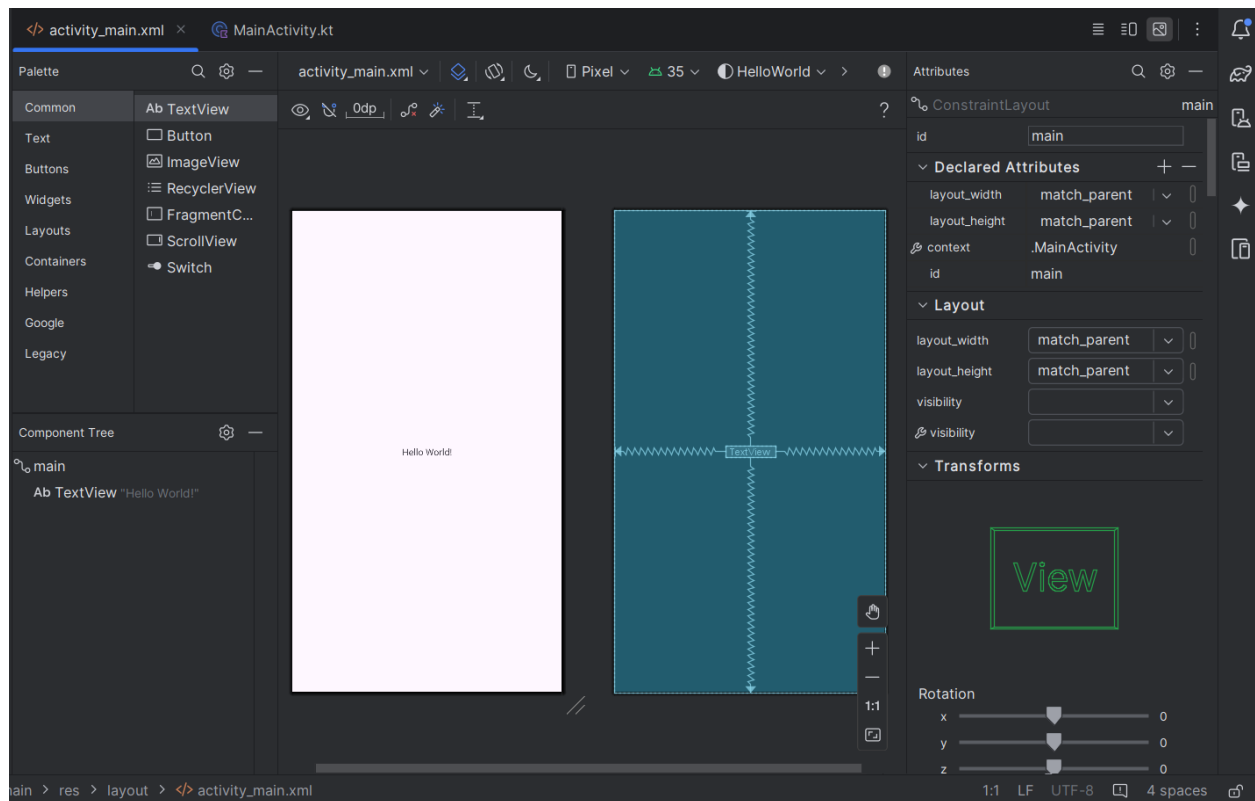
giúp bạn bắt đầu. Đối với dự án Hello World, hãy chọn Empty Activity như bên hình dưới và nhấn Next.



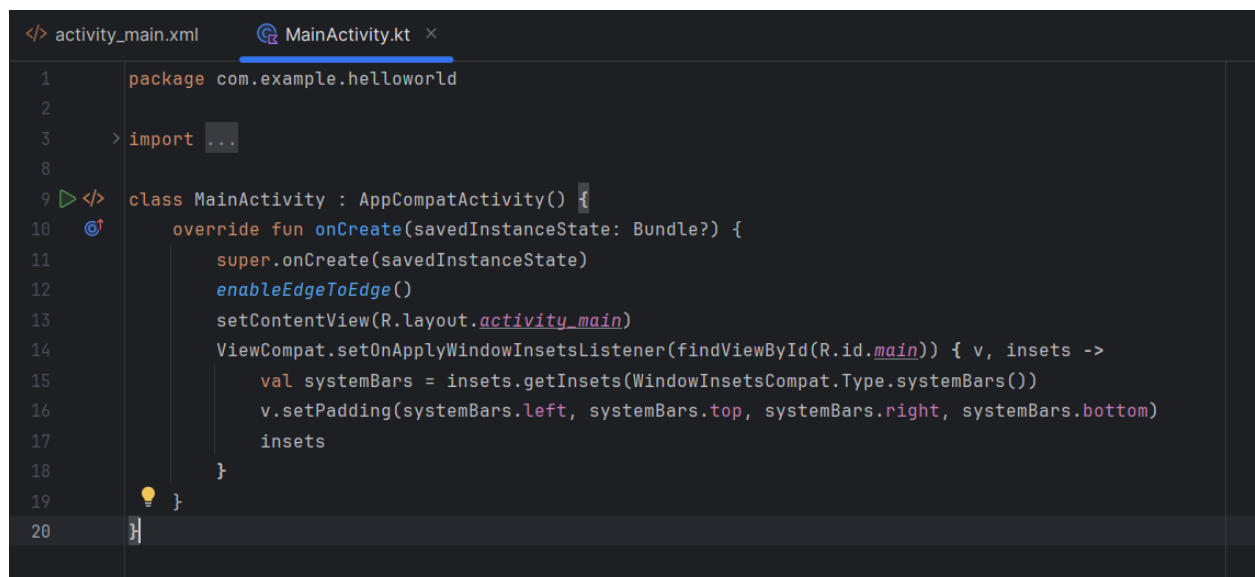
10. Màn hình Configure Activity xuất hiện (giao diện này có thể khác nhau tùy thuộc vào mẫu mà bạn đã chọn ở bước trước). Theo mặc định, Empty Activity do mẫu cung cấp được đặt tên là MainActivity. Bạn có thể thay đổi tên này nếu muốn, nhưng trong bài học này chúng ta sẽ sử dụng MainActivity.
11. Đảm 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. Nhấp 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

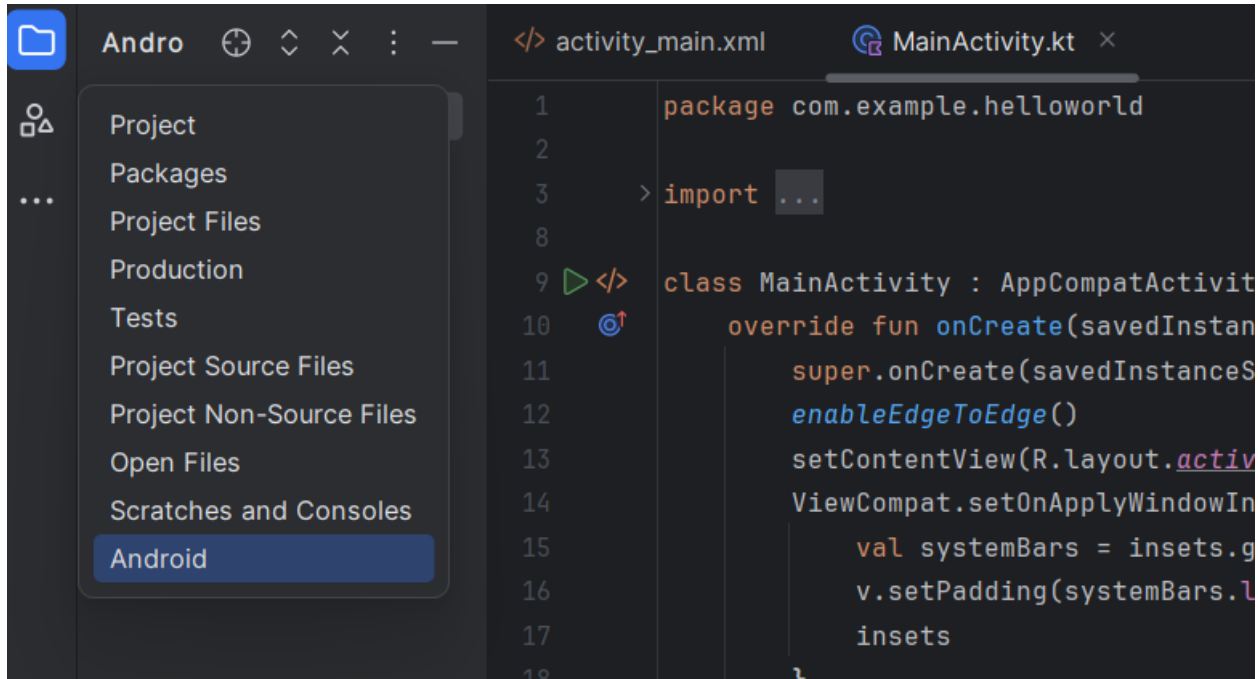


## 2.2 Khám phá Project > Android pane

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



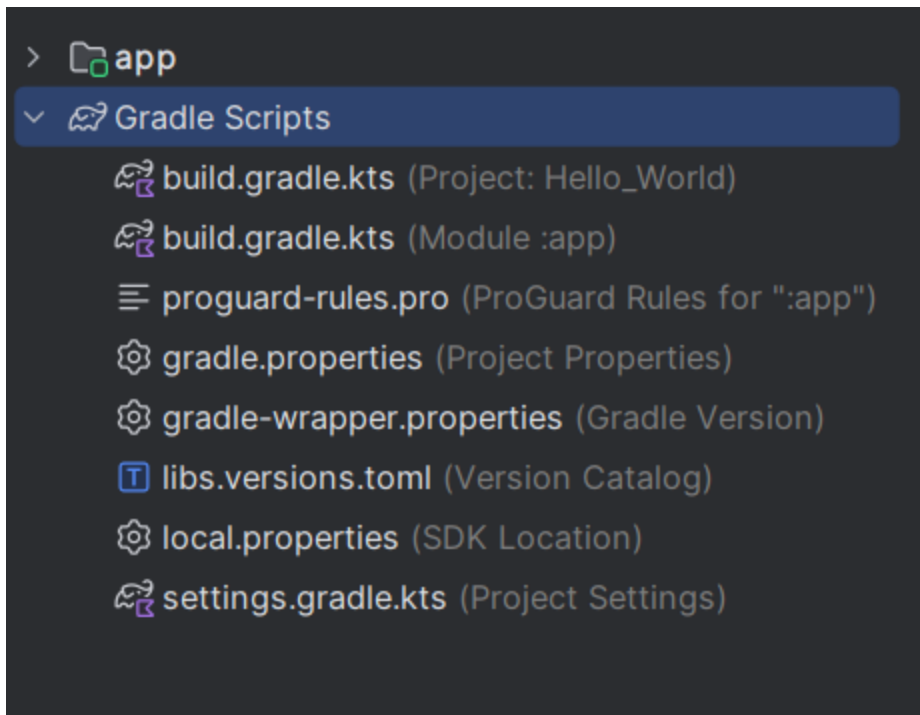
1. Nếu chưa được chọn, hãy nhấp vào tab Project trong cột tab dọc ở phía bên trái cửa sổ Android Studio. Project pane sẽ xuất hiện.
2. Để xem Project theo tiêu chuẩn của Android Project, hãy chọn Android từ popup menu ở đầu Project pane, như hình bên dưới



**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"
    }
}

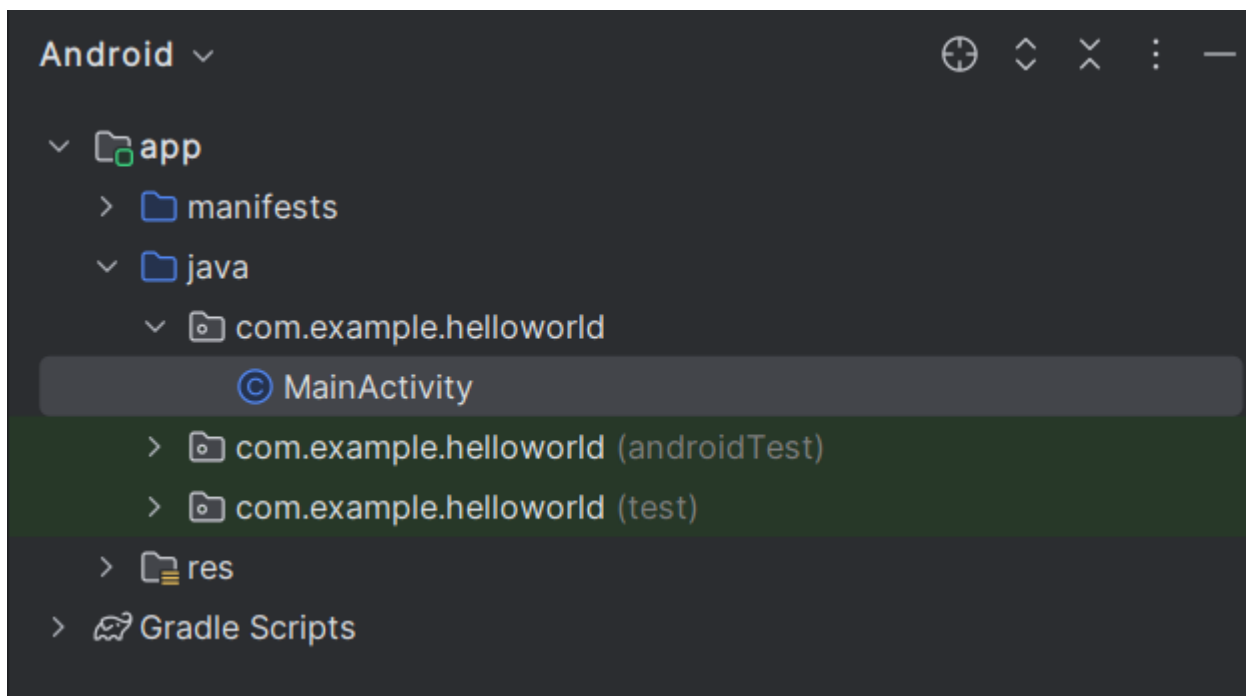
dependencies {
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)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

```

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

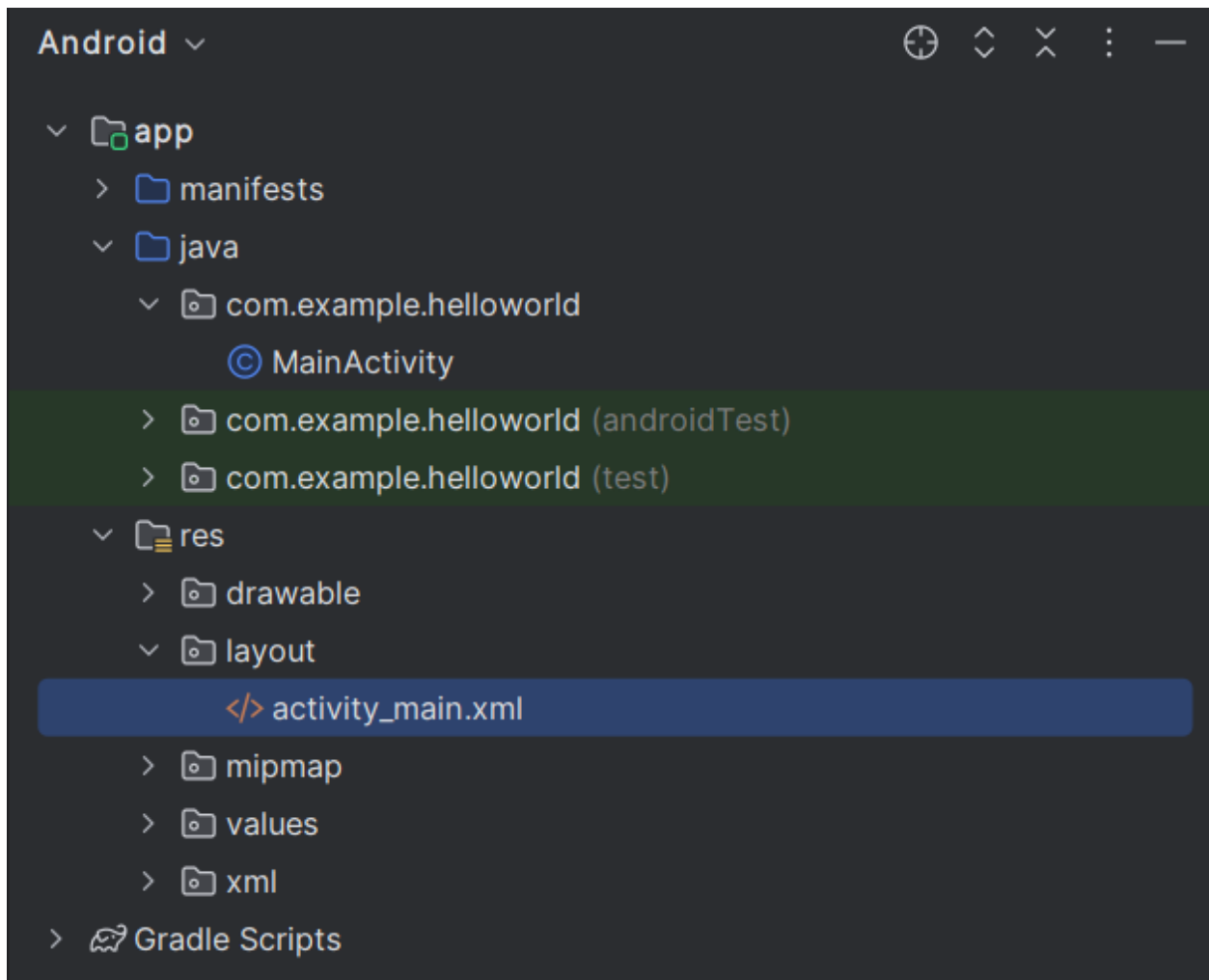
## 2.4 Khám phá ứng dụng và thư mục res

- 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)

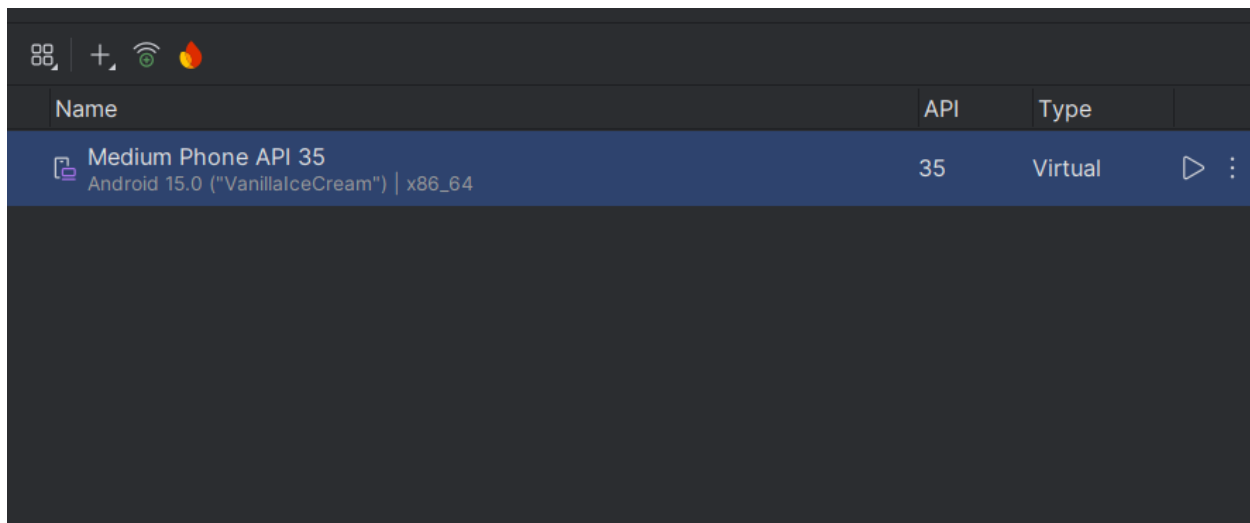
- Trong nhiệm vụ này, bạn sẽ sử dụng Android Virtual Device(ADV) manager để tạo một thiết bị ảo (còn gọi là **trình giả lập**) mô phỏng cấu hình của một loại thiết bị Android cụ thể, sau đó sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý trình giả lập Android yêu cầu thêm một số yêu cầu hệ thống ngoài 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)

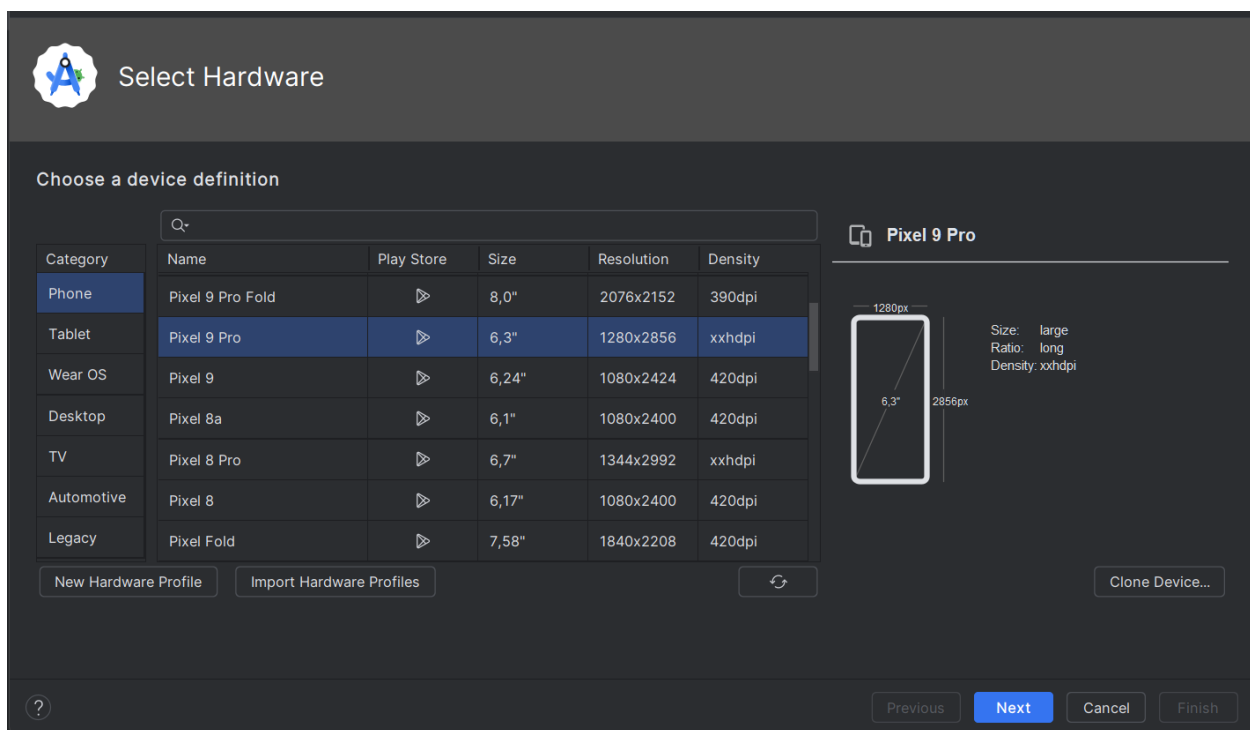
Để chạy trình giả lập trên máy tính của bạn, bạn cần tạo một cấu hình mô tả thiết bị ảo đó.

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.

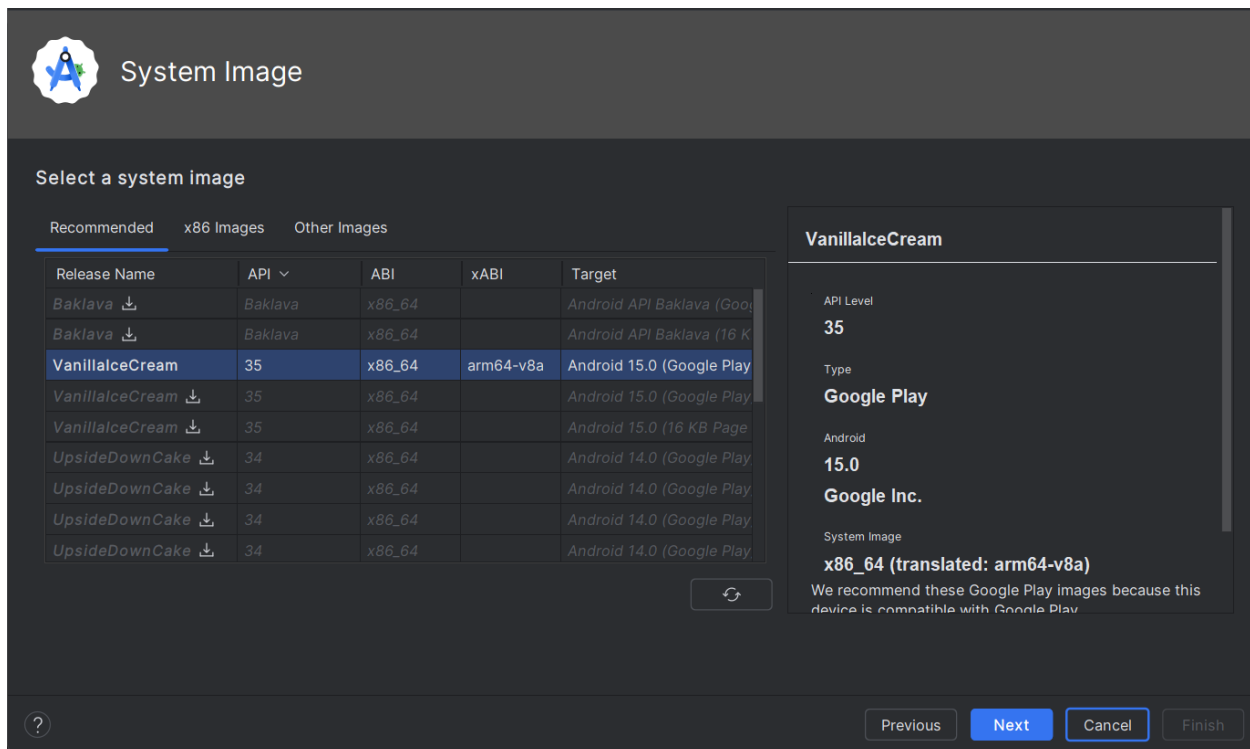


2. Nhấp vào +Create Virtual Device. Cửa sổ Select Hardware xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng sẽ cung cấp các cột tương ứng với kích thước đường chéo màn hình, độ phân giải màn hình tính bằng pixel và mật độ điểm ảnh.



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 nhấn **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 nhấn **Finish**.

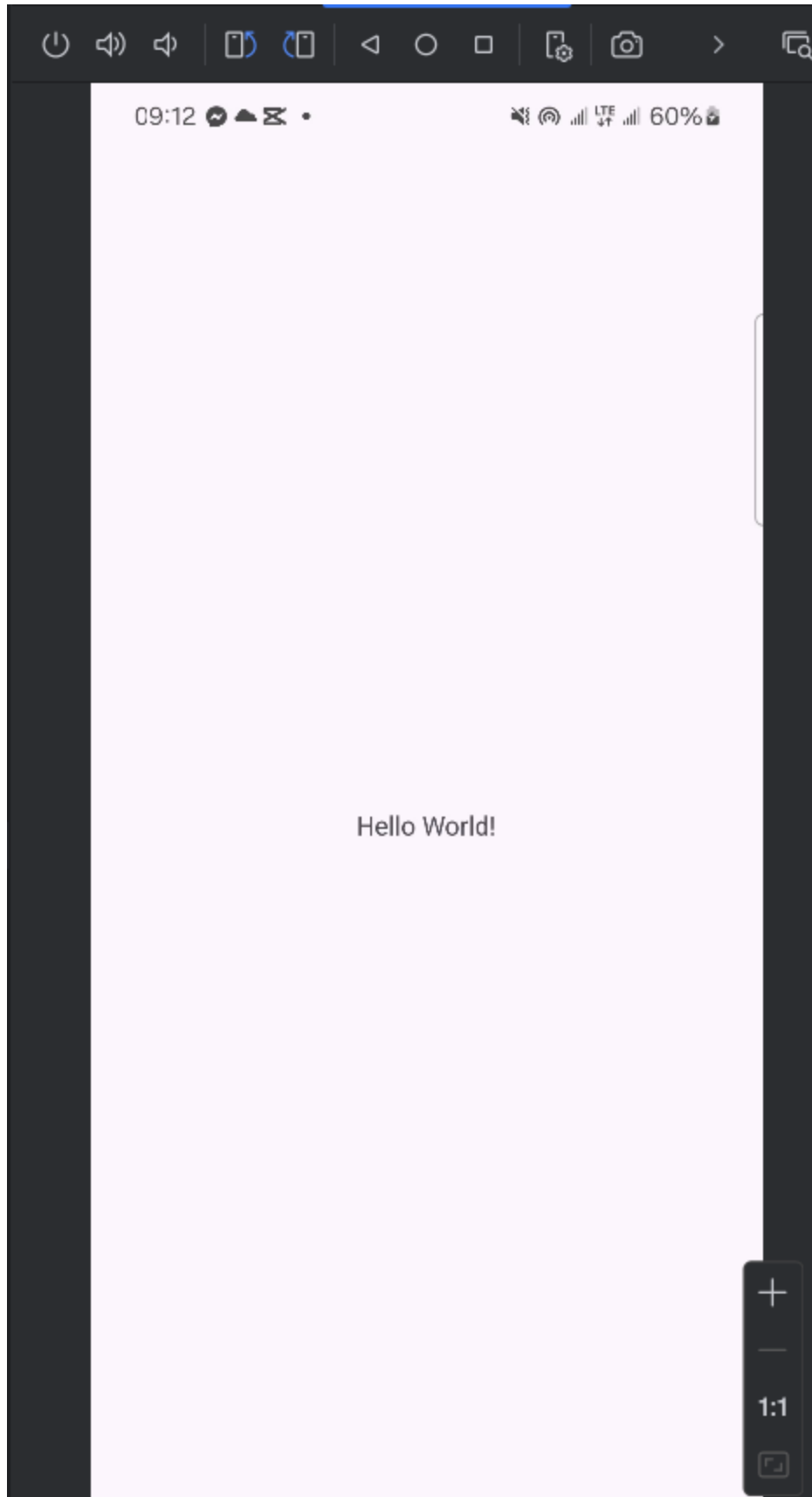
### 3.2 Chạy ứng dụng trên thiết bị ảo

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 nhấn **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

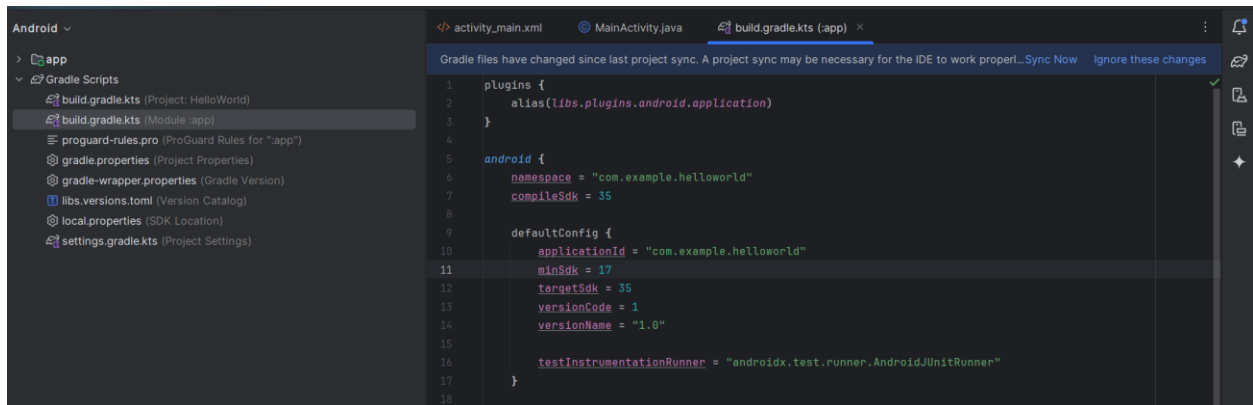
Trong nhiệm vụ này, bạn sẽ thay đổi một số cấu hình của ứng dụng trong tệp `build.gradle` (Module: app) để tìm hiểu cách thực hiện thay đổi và đồng bộ chúng với dự án Android Studio của bạn.

### 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.

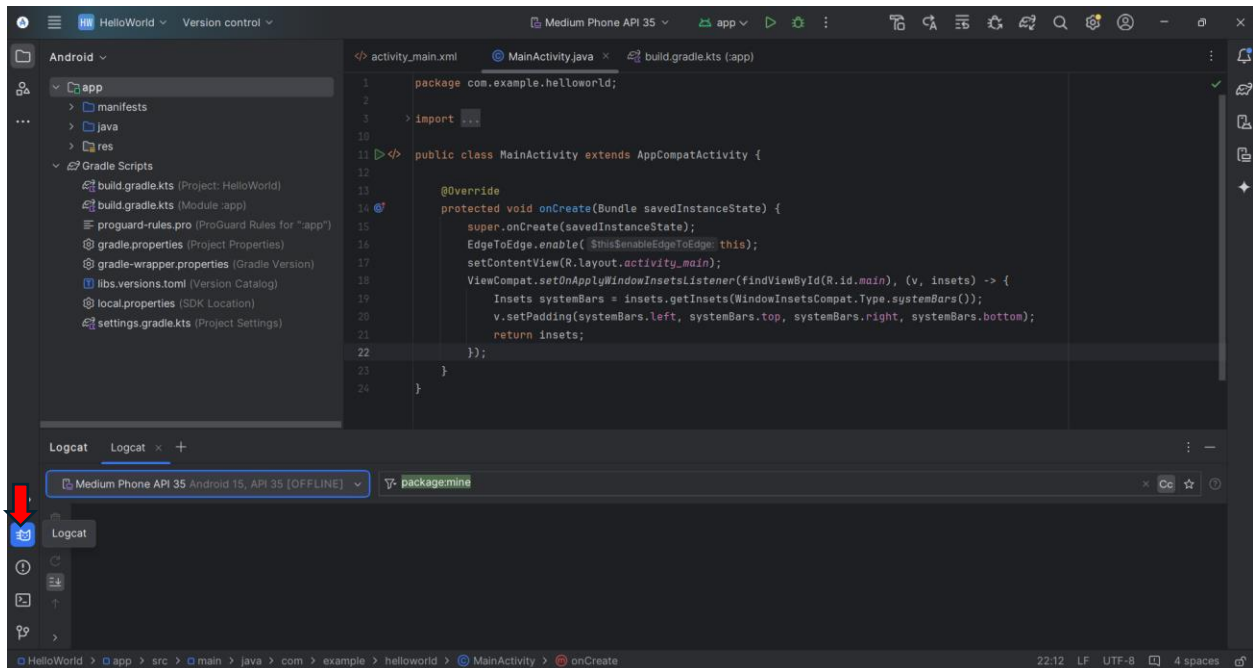
Để tìm hiểu sâu hơn về Gradle, hãy tham khảo tài liệu **Build System Overview** và **Configuring Gradle Builds**.

## 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 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:

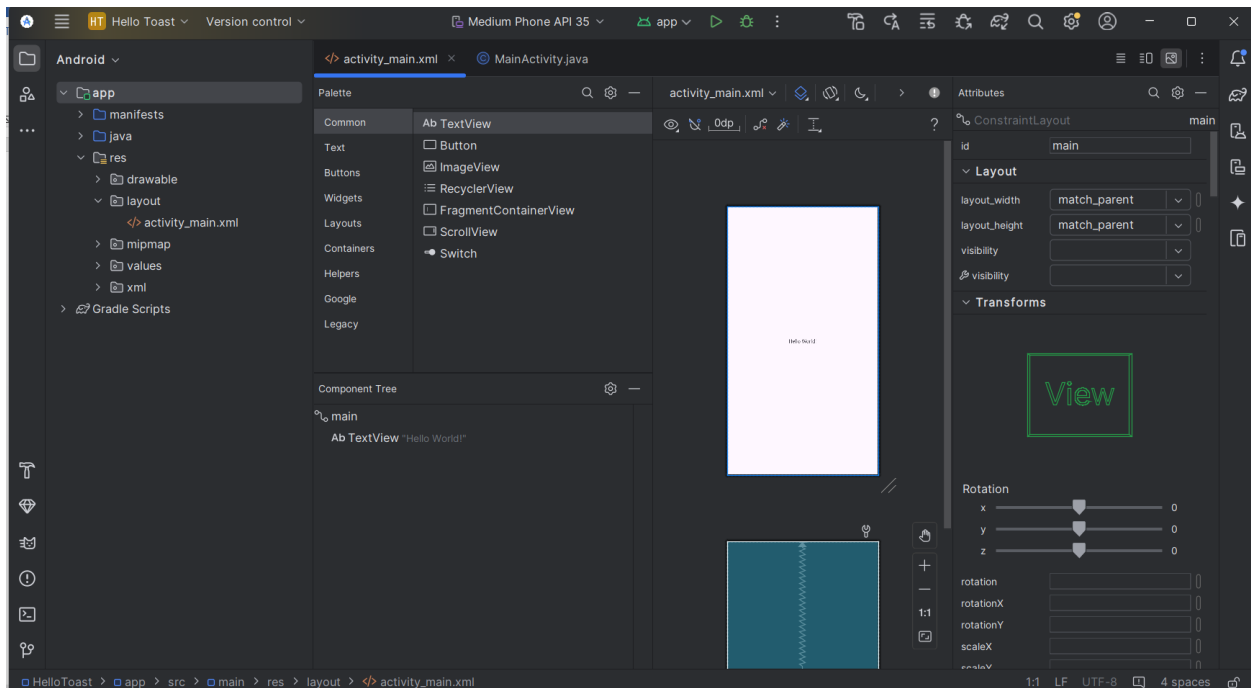
Attribute	Value
Application Name	Hello Toast
Company Name	com.example.android (or your own domain)
Phone and Tablet Minimum SDK	API15: Android 4.0.3 IceCreamSandwich
Template	Empty Activity
Generate Layout file box	Selected
Backwards Compatibility box	Selected

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.

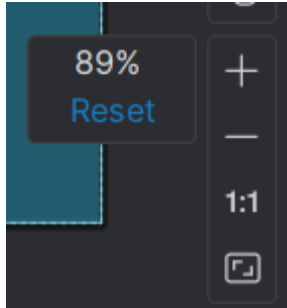
## TASK 2 : Thêm các phần tử View trong Layout Editor

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**.

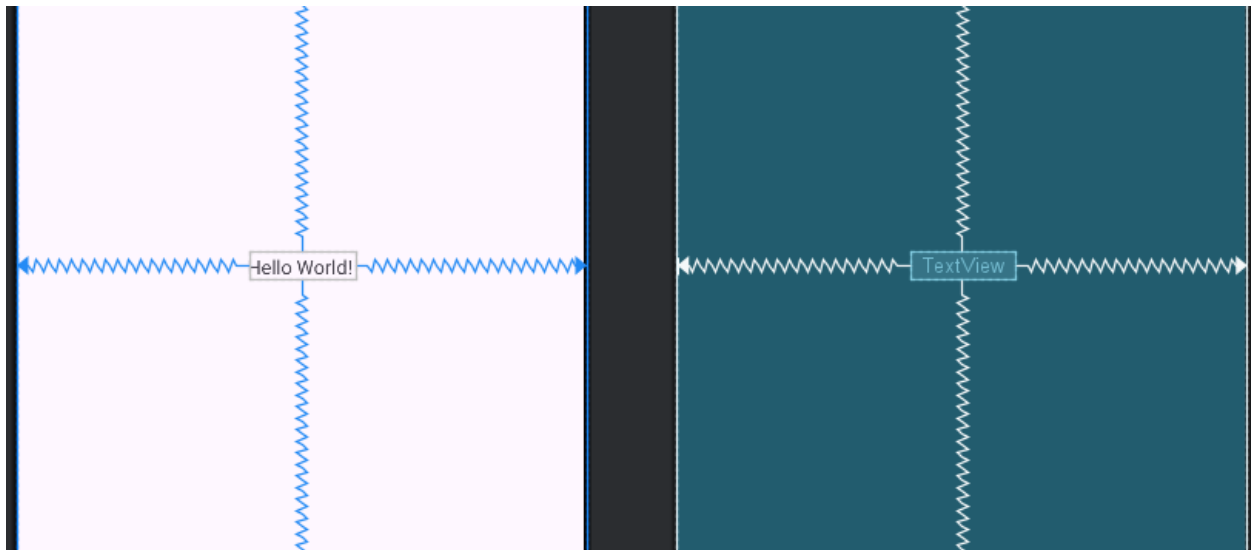
### 2.1 Kiểm tra các ràng buộc của phần tử

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.

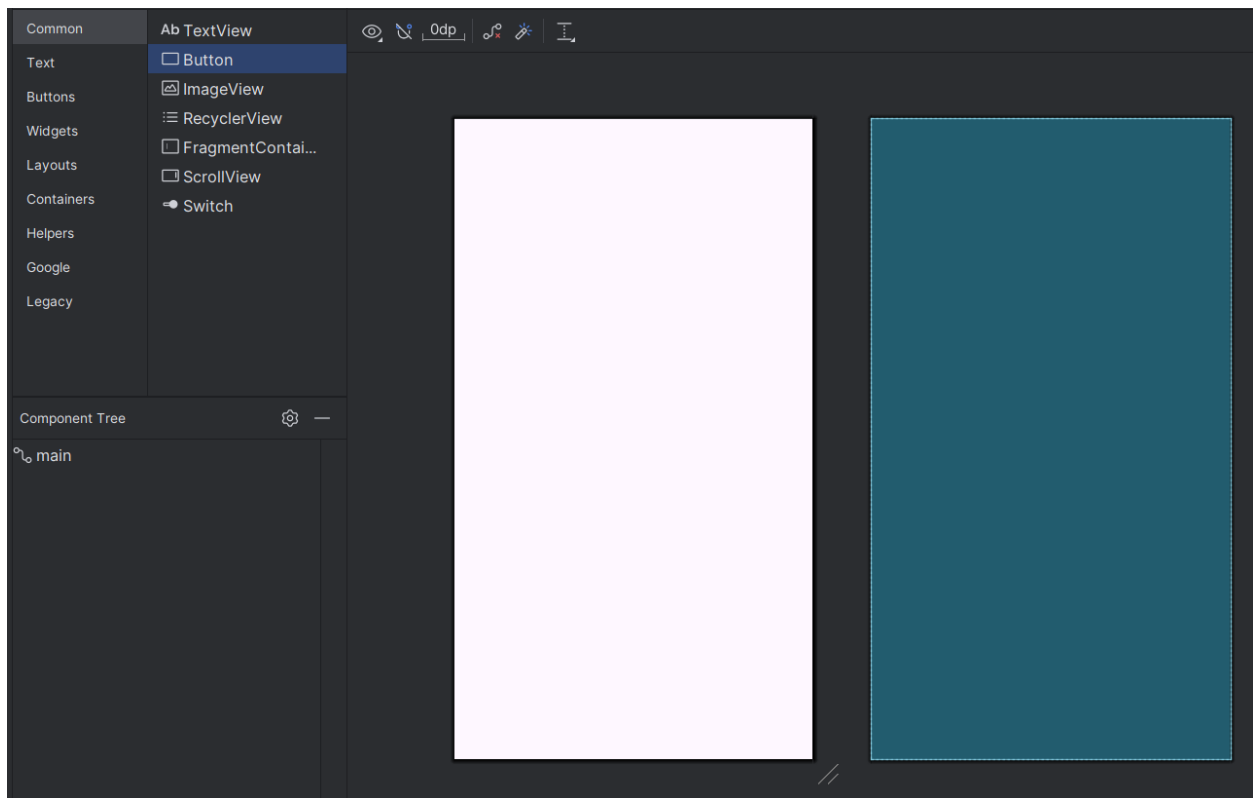


## 2.2 Thêm một Button vào Layout

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

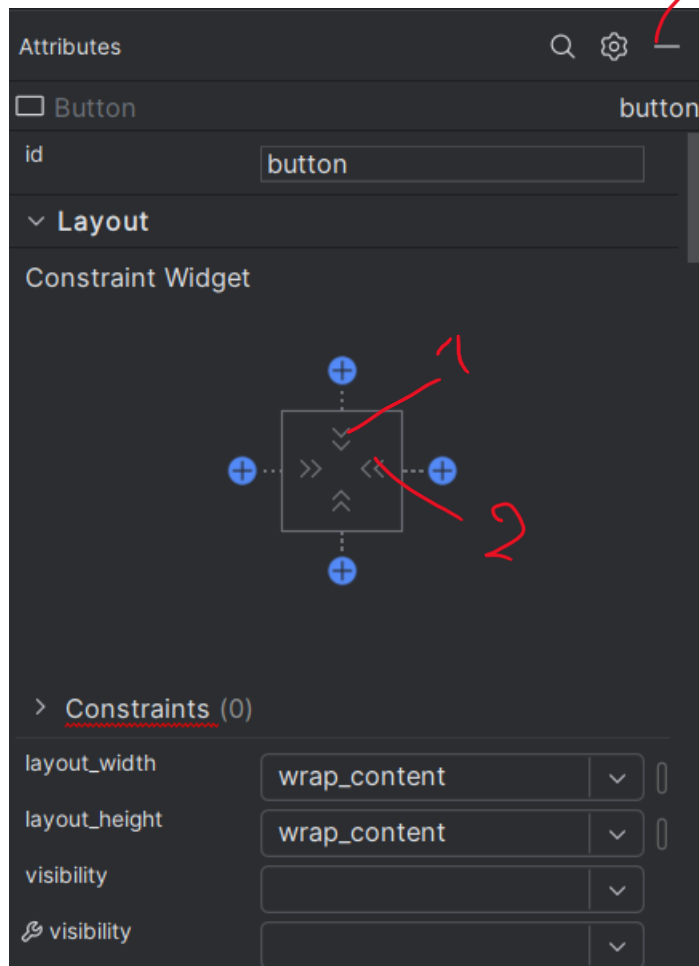
Ngăn Attributes cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng. Bạn có thể tìm thấy các thuộc tính (còn gọi là properties) chung cho tất cả các View trong tài liệu của lớp View.

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:

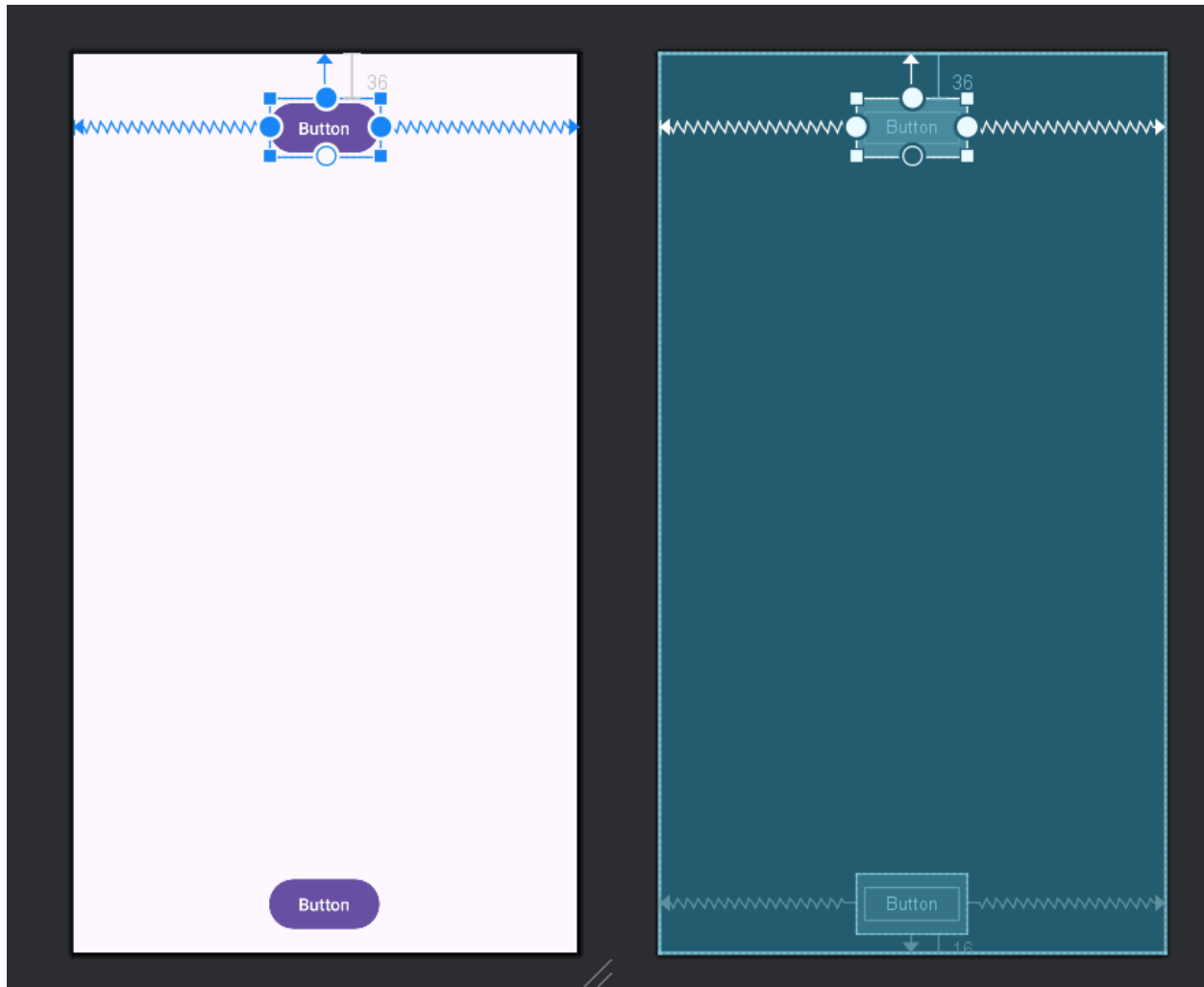
1. **Height control.** Điều khiển này xác định thuộc tính `layout_height` và xuất hiện ở hai đoạn trên và dưới của hình vuông. Các góc chỉ ra rằng điều khiển này được đặt thành `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều dọc khi cần để vừa với nội dung của nó. Số "8" cho biết một lề tiêu chuẩn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



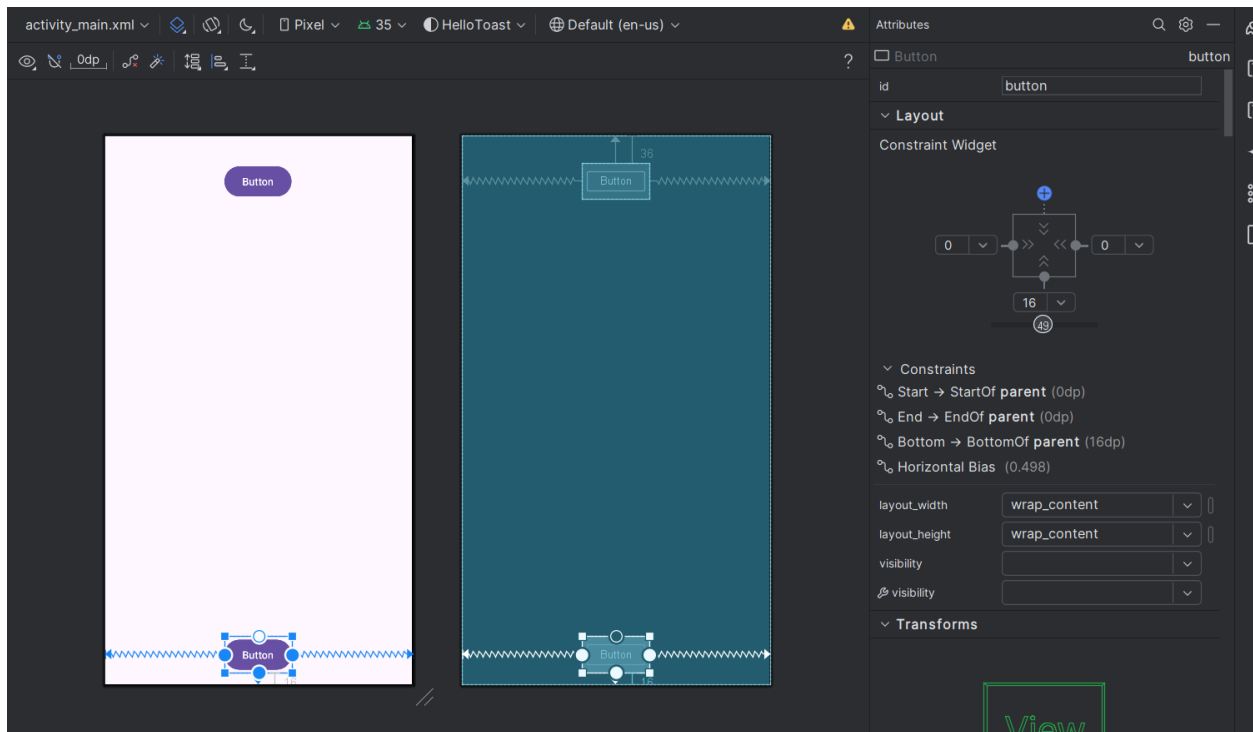
3. Nhấp vào điều khiển chiều rộng hai lần—lần nhấp đầu tiên thay đổi nó thành **Fixed** với các đường thẳng, và lần nhấp thứ hai thay đổi nó thành **Match Constraints** với các lò xo, như hiển thị trong hình động bên dưới.



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.

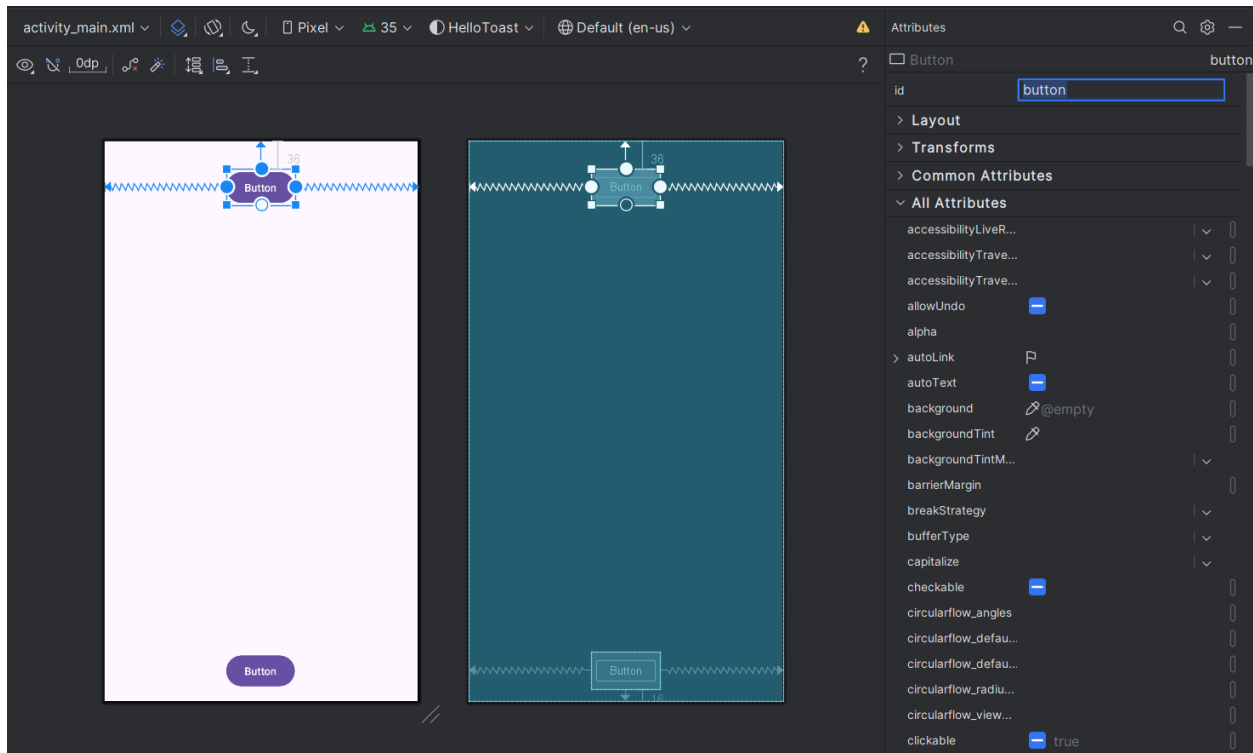
### 3.2 Thay đổi các thuộc tính của Button

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.

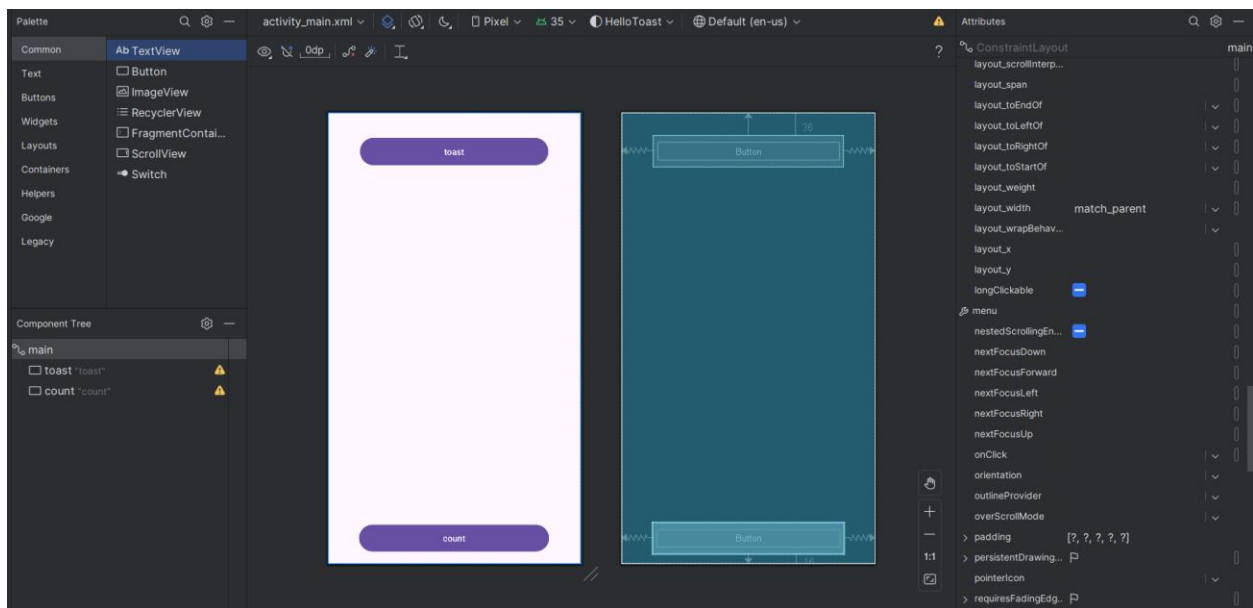
#### TASK 4 : Thêm một **TextView** và thiết lập thuộc tính

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

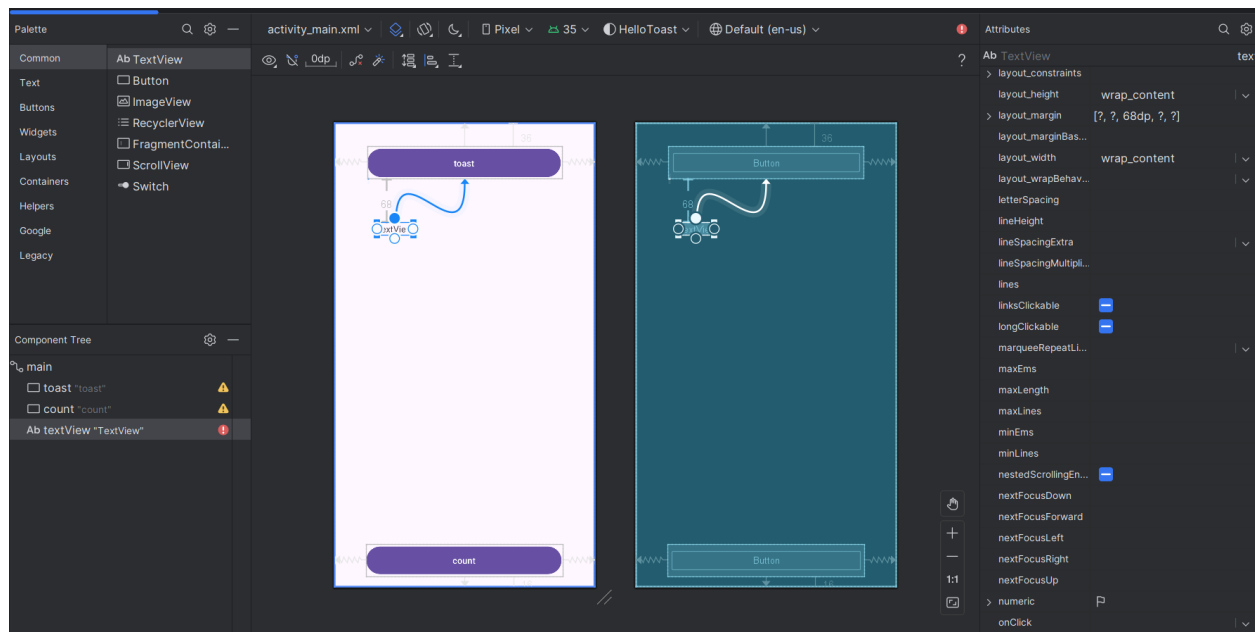
từ **Button**. Sau đó, bạn sẽ thay đổi các thuộc tính cho **TextView** trong bảng **Attributes**.

#### 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**

## 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

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.

**1.3) Trình chỉnh sửa bố cục**

**1.4) Văn bản và các chế độ cuộn**

**1.5) Tài nguyên có sẵn**

## **Bài 2) Activities**

**2.1) Activity và Intent**

**2.2) Vòng đời của Activity và trạng thái**

**2.3) Intent ngầm định**

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

**3.1) Trình gỡ lỗi**

**3.2) Kiểm thử đơn vị**

**3.3) Thư viện hỗ trợ**