

**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	3
Bài 1) Tạo ứng dụng đầu tiên	3
1.1) Android Studio và Hello World.....	3
1.2) Giao diện người dùng tương tác đầu tiên	24
1.3) Trình chỉnh sửa bố cục	46
1.4) Văn bản và các chế độ cuộn	46
1.5) Tài nguyên có sẵn	46
Bài 2) Activities.....	46
2.1) Activity và Intent.....	46
2.2) Vòng đời của Activity và trạng thái	46
2.3) Intent ngầm định	46
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	46
3.1) Trình gỡ lỗi.....	46
3.2) Kiểm thử đơn vị	46
3.3) Thư viện hỗ trợ	46
CHƯƠNG 2. Trải nghiệm người dùng.....	47
Bài 1) Tương tác người dùng	47
1.1) Hình ảnh có thể chọn.....	47
1.2) Các điều khiển nhập liệu	47
1.3) Menu và bộ chọn	47
1.4) Điều hướng người dùng	47
1.5) RecyclerView	47
Bài 2) Trải nghiệm người dùng thú vị.....	47
2.1) Hình vẽ, định kiểu và chủ đề	47
2.2) Thẻ và màu sắc	47
2.3) Bố cục thích ứng.....	47
Bài 3) Kiểm thử giao diện người dùng	47

3.1) Espresso cho việc kiểm tra UI	47
CHƯƠNG 3. Làm việc trong nền	47
Bài 1) Các tác vụ nền.....	47
1.1) AsyncTask.....	47
1.2) AsyncTask và AsyncTaskLoader	47
1.3) Broadcast receivers	47
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	47
2.1) Thông báo.....	47
2.2) Trình quản lý cảnh báo	47
2.3) JobScheduler	47
CHƯƠNG 4. Lưu dữ liệu người dùng	48
Bài 1) Tùy chọn và cài đặt.....	48
1.1) Shared preferences	48
1.2) Cài đặt ứng dụng	48
Bài 2) Lưu trữ dữ liệu với Room	48
2.1) Room, LiveData và ViewModel.....	48
2.2) Room, LiveData và ViewModel.....	48
3.1) Trình gowx lỗi	

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

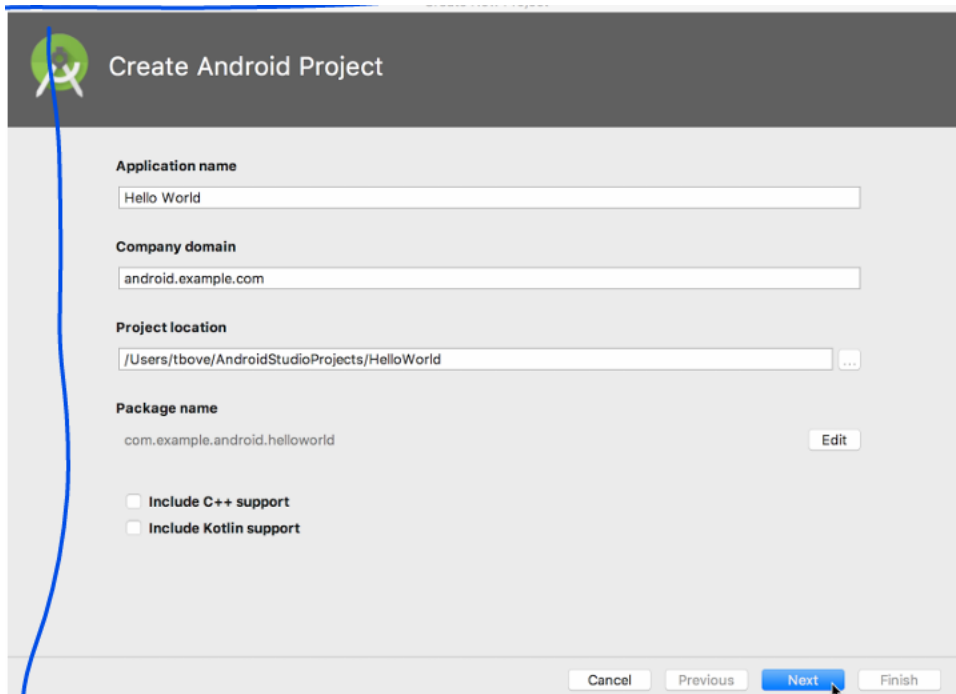
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

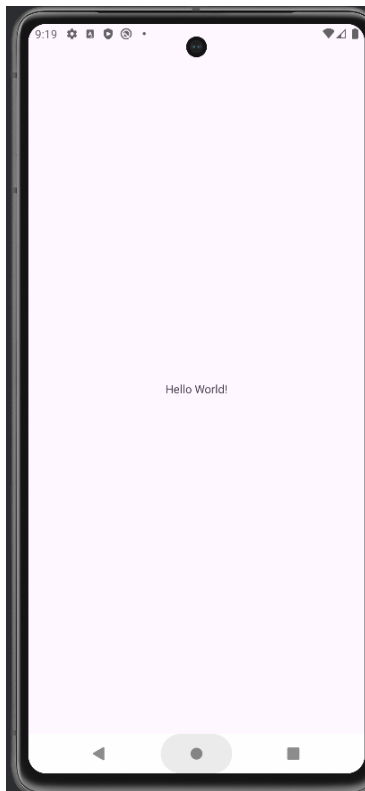
Những gì bạn sẽ làm

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

Ứng dụng tổng quát

Sau khi bạn cài đặt thành công Android Studio, bạn sẽ tạo từ các giao diện, cho dự án ứng dụng HelloWorld. Đây là một ứng dụng đơn giản hiển thị xâu “Hello World” trên màn hình máy ảo Android hoặc thiết bị vật lý.

Ứng dụng khi hoàn thành sẽ trông giống như thế này:



Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp môi trường phát triển tích hợp đầy đủ bao gồm tiện ích của trình soạn thảo mã nguồn và giao diện mẫu. Thêm vào đó, Android Studio bao gồm các công cụ cho phát triển, gỡ lỗi, thử nghiệm và chạy, điều đó giúp phát triển

ứng dụng nhanh hơn và dễ dàng hơn. Bạn có thể thử nghiệm ứng dụng của bạn với đa dạng các cấu hình trình giả lập được thiết lập trước trên thiết bị di động mà bạn sở hữu, xây dựng ứng dụng và xuất bản lên Google Play Store.

Android Studio cho phép chạy trên Window hoặc Linux, và thiết bị Macs chạy trên macOS. Phiên bản OpenJDK (Java Development Kit) mới nhất được đóng gói trong Android Studio.

Để khởi động và sử dụng với Android Studio, đầu tiên cần kiểm tra hệ yêu cầu hệ thống có đáp ứng với hệ thống máy của bạn không. Cài đặt tương tự cho mọi nền tảng. Tất cả các trường hợp khác được ghi chú bên dưới.

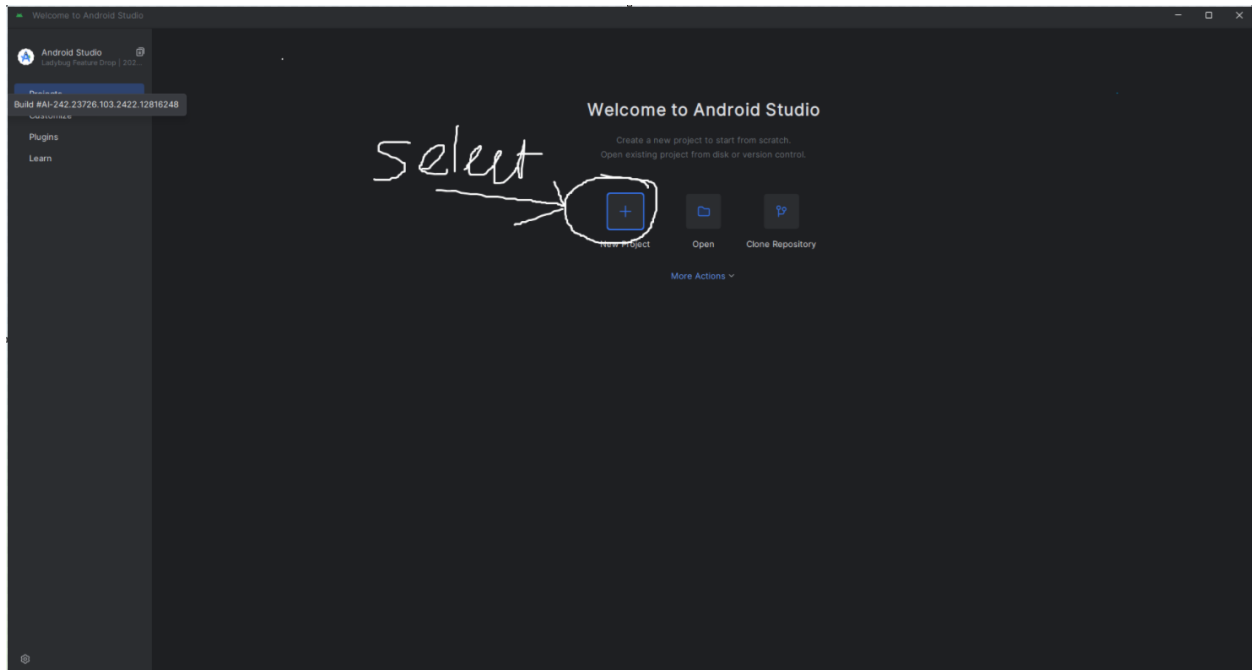
1. Đến trang Android developer và theo dõi hướng dẫn từ tải đến cài đặt Android Studio.
2. Chấp nhận cấu hình mặc định cho tất cả các bước, và đảm bảo tất cả các thành phần cần có được cài đặt.
3. Sau khi cài đặt xong, Setup Wizard sẽ được tải và cài đặt thêm các thành phần bao gồm Android SDK. Hãy kiên nhẫn điều này có thể tốn thời gian phụ thuộc vào tốc độ mạng internet của bạn, và một số bước có thể dư thừa.
4. Khi việc tải hoàn thành, Android Studio sẽ khởi động và bạn đã sẵn sàng để tạo ứng dụng đầu tiên cho bản thân.

Nhiệm vụ 2: Tạo ứng dụng Hello World

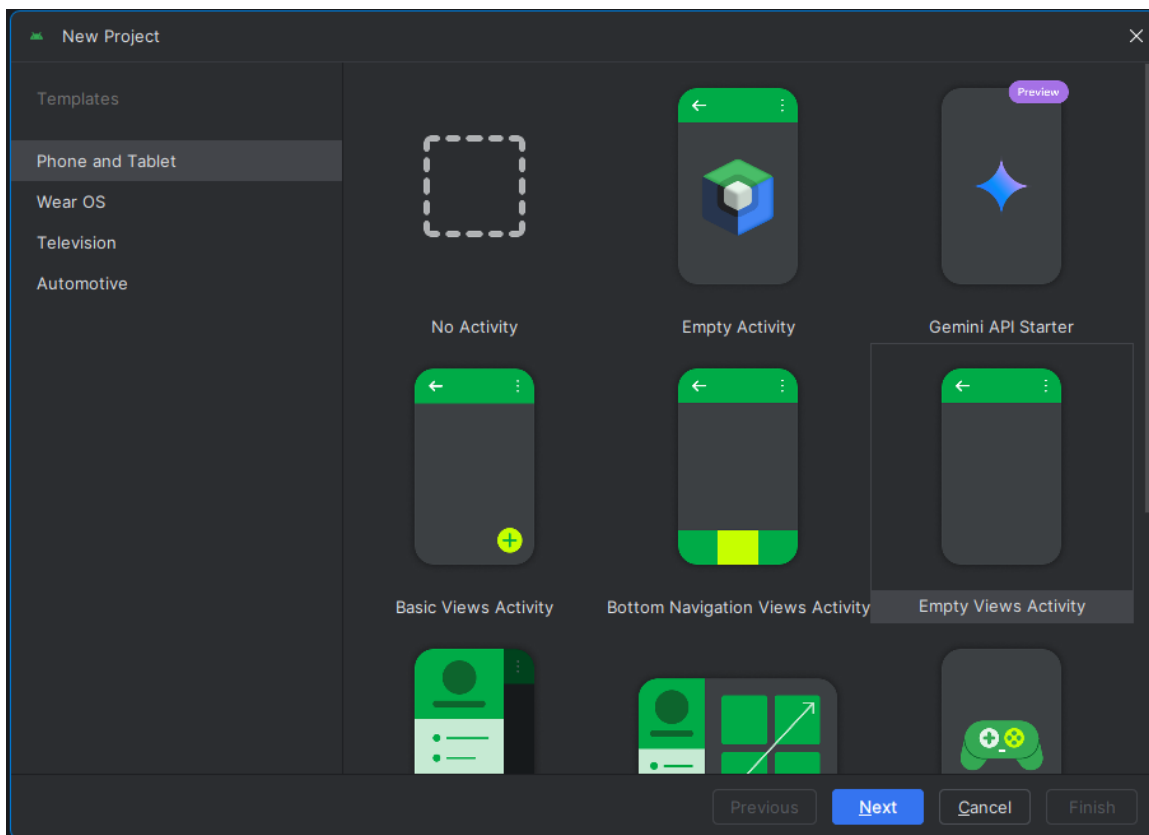
Trong nhiệm vụ này, bạn sẽ tạo ứng dụng hiển thị “Hello world” để xác nhận Android Studio đã cài đặt chính xác, và nghiên cứu việc phát triển cơ bản với AS.

2.1 Tạo dự án ứng dụng

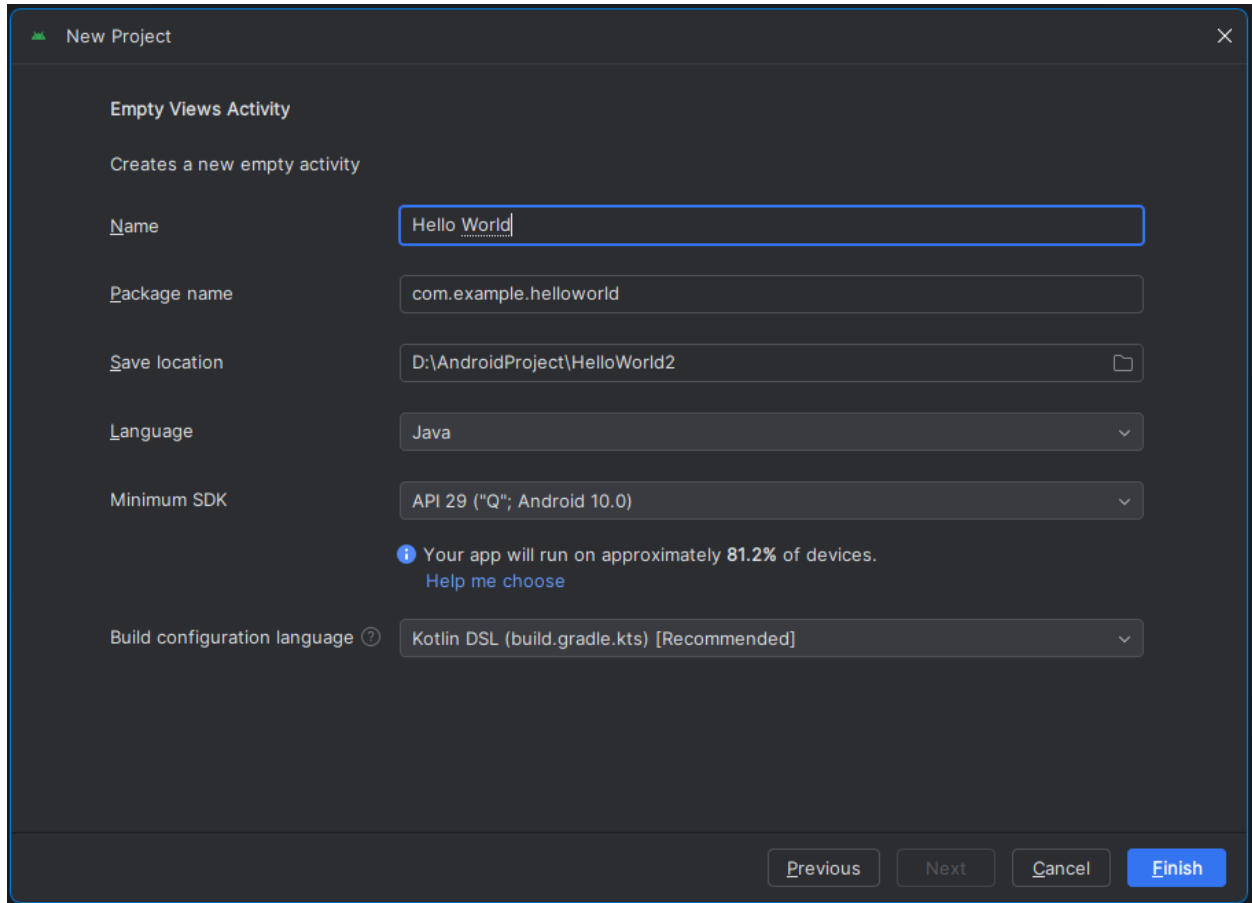
1. Mở Android Studio nếu nó chưa được mở
2. Tại màn hình Welcome to Android Studio, chọn Start a new Android Studio project



3. Chọn Empty Views Activity



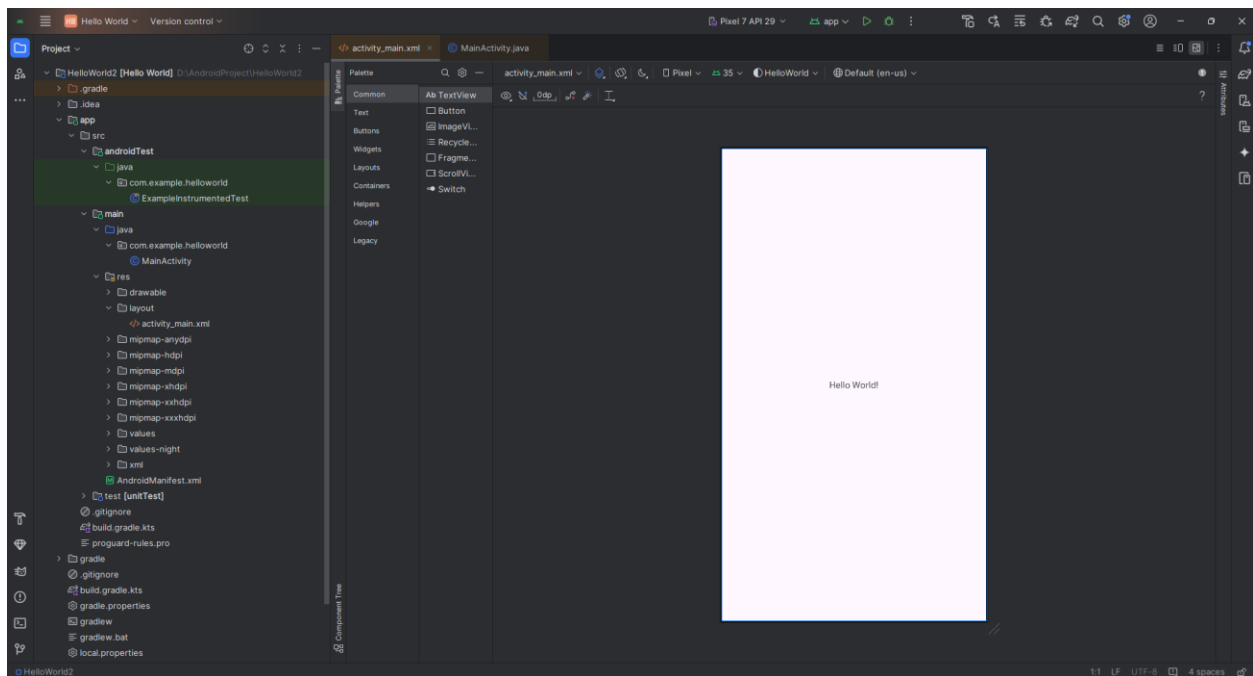
- Trong cửa sổ Create Android Project, điền Hello World vào Application name.



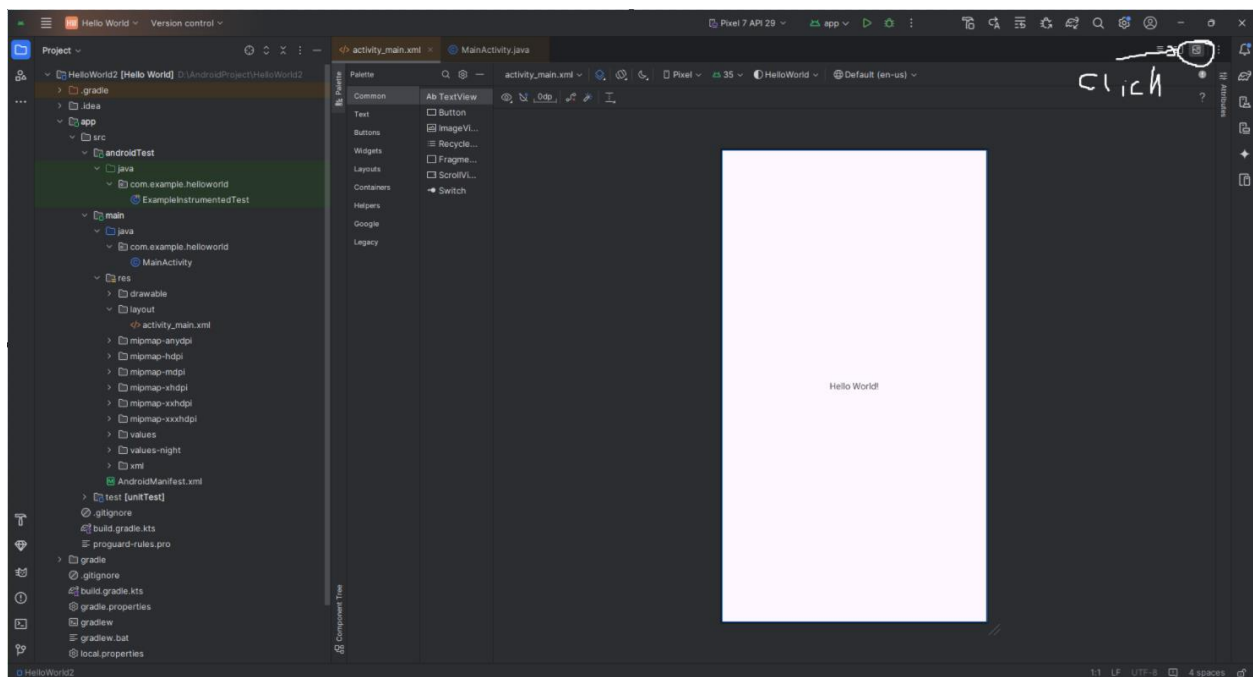
- Xác nhận vị trí mặc định của Project location là nơi bạn muốn lưu ứng dụng Hello World và các Android Studio projects khác hoặc thay đổi theo đường dẫn chính xác.
- Chấp nhận mặc định android.example.com cho Company Domain, hoặc tạo đường dẫn duy nhất cho công ty của bạn
- Ấn Finish sau khi xác nhận các lựa chọn.

Công cụ chỉnh sửa của Android Studio xuất hiện. Làm theo các bước sau đây:

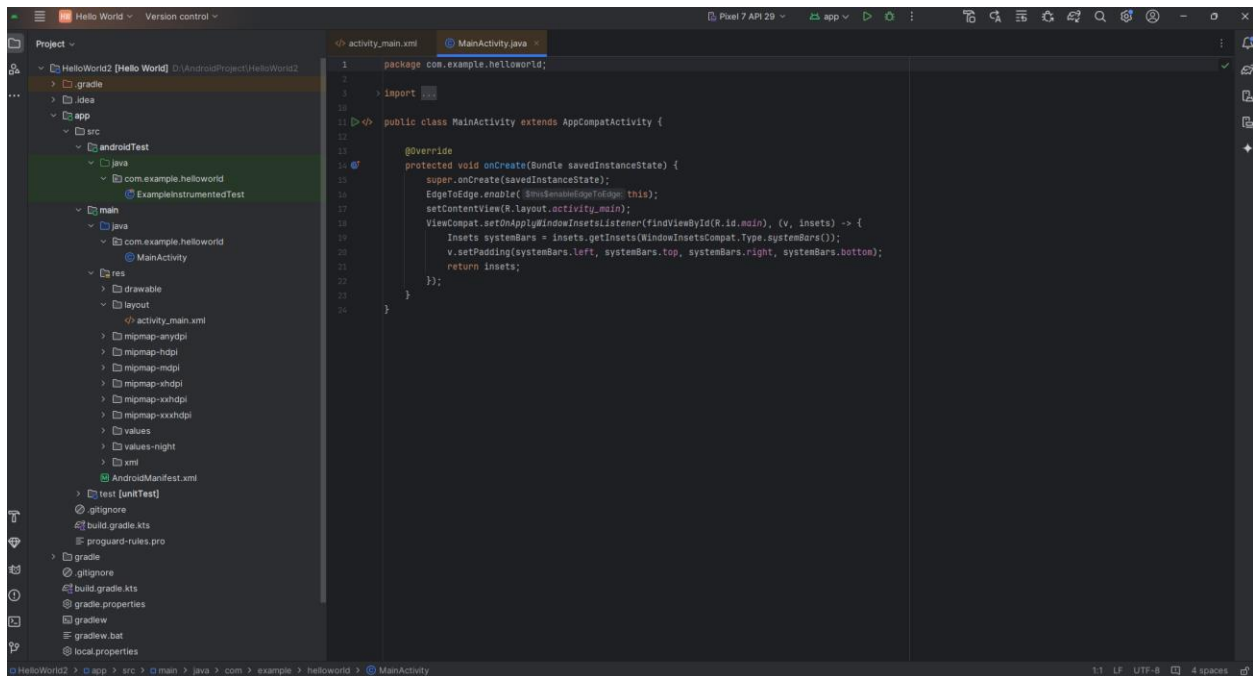
- Nhấn chuột vào activity_main.xml cửa sổ để xem trình chỉnh sửa giao diện.



2. Nhấn chuột vào Design tab, nếu nó không được chọn, để hiển thị một bản tái hiện đồ họa của Bố cục như hình dưới đây.



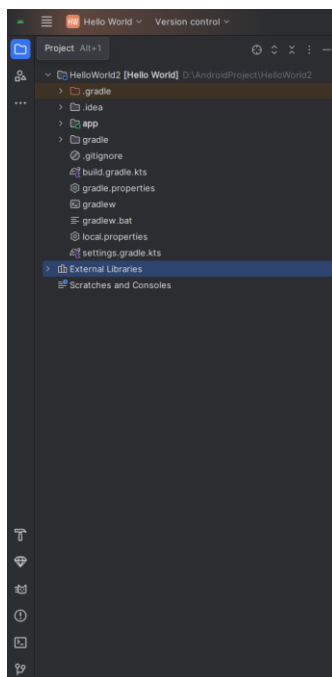
3. Chọn MainActivity.java để xem mã code



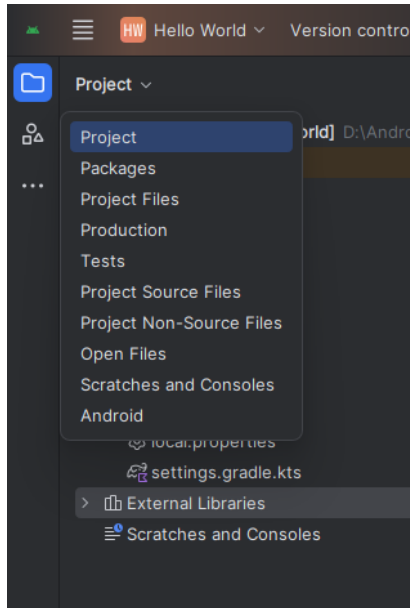
2.2 Khám phá Project > Android pane

Trong phần thực hành này, bạn sẽ khám phá cách tổ chức Project Android Studio

1. Nếu chưa được chọn, bấm chuột vào Project tại thanh quản lý tác vụ dọc bên trái của sổ làm việc. Project sẽ hiện ra



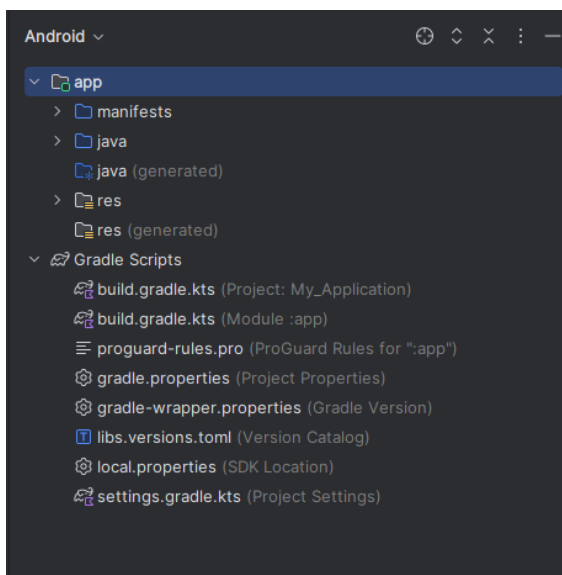
- Để xem dự án trong hệ thống phân cấp dự án Android tiêu chuẩn, chọn Android từ menu bật lên ở đầu ngăn dự án, như được hiển thị bên dưới



2.3 Tìm hiểu thư mục Gradle Scripts

Gradle xây dựng hệ thống trong Android Studio khiến dễ dàng hơn, nó bao gồm file nhị phân hoặc các modules thư viện được bạn khai báo.

Khi bạn tạo ứng dụng lần đầu, Project Android pane xuất hiện với thư mục Gradle Scripts mở động như hình bên dưới



Theo dõi các bước để tìm hiểu và hệ thống Gradle:

1. Nếu thư mục Gradle Scripts không mở, bấm chuột vào hình mũi tên để mở nó. Thư mục này chứa tất cả các tập tin cần thiết để xây dựng hệ thống.
2. Xem tập tin build.gradle(Project: HelloWorld).

Đây là nơi bạn sẽ tìm thấy các lựa chọn cấu hình phổ thông cho tất cả các modules tạo nên dự án của bạn. Phần lớn, bạn sẽ không cần thay đổi file này, nhưng nó vẫn sẽ hữu ích để hiểu nội dung.

Theo mặc định, tập bản dựng cấp cao nhất sử dụng khối BuildScript để xác định các kho lưu trữ và phụ thuộc Gradle phổ biến cho tất cả các mô-đun trong dự án. Khi sự phụ thuộc của bạn là một thứ khác ngoài thư viện hoặc cây tập cục bộ, Gradle tìm các tập trong bất kỳ kho lưu trữ trực tuyến nào được chỉ định trong khối kho lưu trữ của tập này. Theo mặc định, các dự án Android Studio mới khai báo JCenter và Google (bao gồm kho lưu trữ Google Maven) làm vị trí kho lưu trữ:

3. Xem tập tin build.gradle(Module:app)

Thêm nữa tập tin build.gradle có project-level, mỗi mô-đun có một tập build.gradle cho riêng mình, nó cho phép bạn cấu hình cài đặt cho mỗi mô-đun cá nhân. Xây dựng cấu hình cài đặt cho phép bạn bổ sung các lựa chọn tùy chỉnh, như các loại build bổ xung

Tập này thường là tập được chỉnh sửa khi thay đổi các cấu hình cấp ứng dụng, chẳng hạn như khai báo các thư viện phụ thuộc trong phần **dependencies**. Bạn có thể khai báo một thư viện phụ thuộc bằng một trong số các cấu hình phụ thuộc khác nhau. Mỗi cấu hình phụ thuộc cung cấp cho Gradle những hướng dẫn khác nhau về cách sử dụng thư viện.

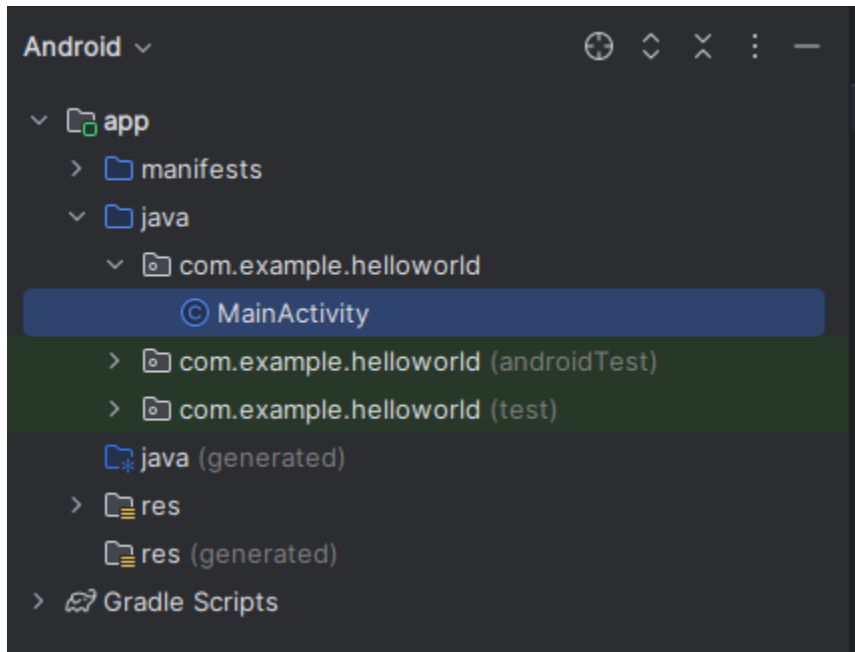
```
activity_main.xml  build.gradle.kts (Hello World)  build.gradle.kts (:app)  settings.gradle.kts (Hello World)
1  plugins {
2      alias(libs.plugins.android.application)
3  }
4
5  android {
6      namespace = "com.example.helloworld"
7      compileSdk = 35
8
9      defaultConfig {
10         applicationId = "com.example.helloworld"
11         minSdk = 29
12         targetSdk = 35
13         versionCode = 1
14         versionName = "1.0"
15
16         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
17     }
18
19     buildTypes {
20         release {
21             isMinifyEnabled = false
22             proguardFiles(
23                 getDefaultProguardFile("name: "proguard-android-optimize.txt"),
24                 "proguard-rules.pro"
25             )
26         }
27     }
28     compileOptions {
29         sourceCompatibility = JavaVersion.VERSION_11
30         targetCompatibility = JavaVersion.VERSION_11
31     }
32 }
33
34 dependencies {
35
36     implementation(libs.appcompat)
37     implementation(libs.material)
38     implementation(libs.activity)
39     implementation(libs.constraintlayout)
40     testImplementation(libs.junit)
41     androidTestImplementation(libs.ext.junit)
42     androidTestImplementation(libs.espresso.core)
```

4. Bấm chuột vào mũi tên để đóng Gradle Scripts

2.4 Tìm hiểu thư mục app và res

Tất cả đoạn mã và tài nguyên của ứng dụng đều được nằm trong thư mục app và res

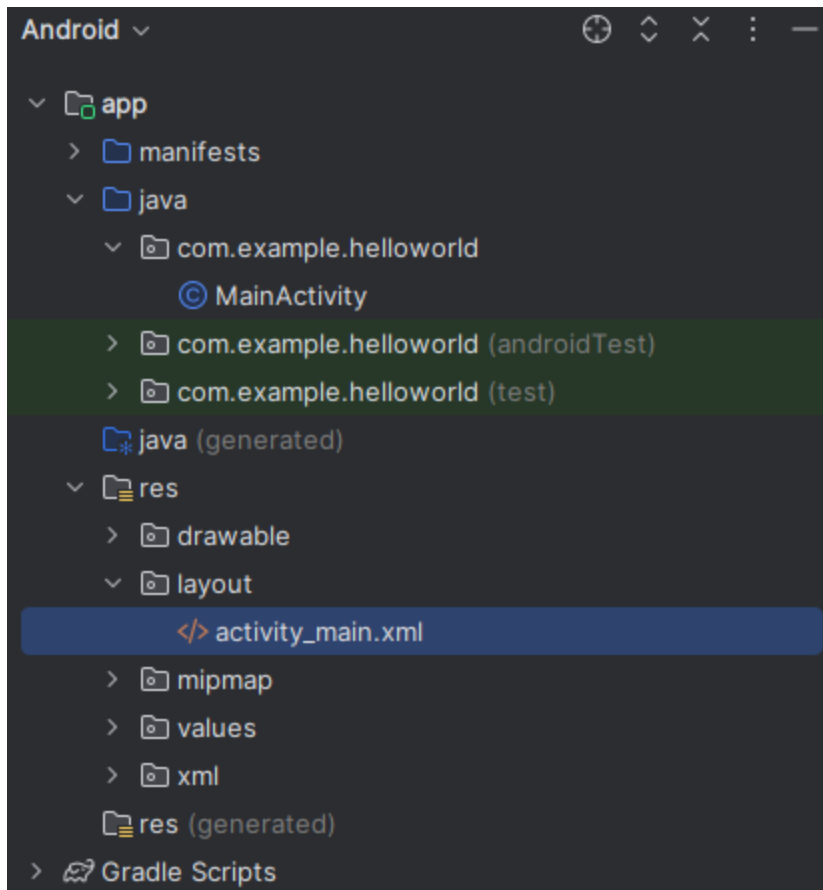
1. Mở rộng thư mục app, thư mục java, và thư mục com.example.android.helloworld để xem tệp MainActivity.java. Nháy đúp chuột vào tệp để mở giao diện code.



Thư mục **java** chứa các tệp lớp Java trong ba thư mục con, như được hiển thị trong hình phía trên. Thư mục **com.example.hello.helloworld** (hoặc tên miền mà bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục còn lại được sử dụng để kiểm thử và sẽ được mô tả trong một bài học khác.

Đối với ứng dụng "Hello World", chỉ có một gói và nó chứa tệp **MainActivity.java**. Tên của **Activity** (màn hình) đầu tiên mà người dùng nhìn thấy, đồng thời cũng khởi tạo các tài nguyên trên toàn bộ ứng dụng, thường được gọi là **MainActivity** (phần mở rộng tệp được lược bỏ trong bảng **Project > Android**).

2. Mở thư mục res và thư mục layout, nháy đúp chuột vào tệp tin activity_main.xml để mở tập tin layout.

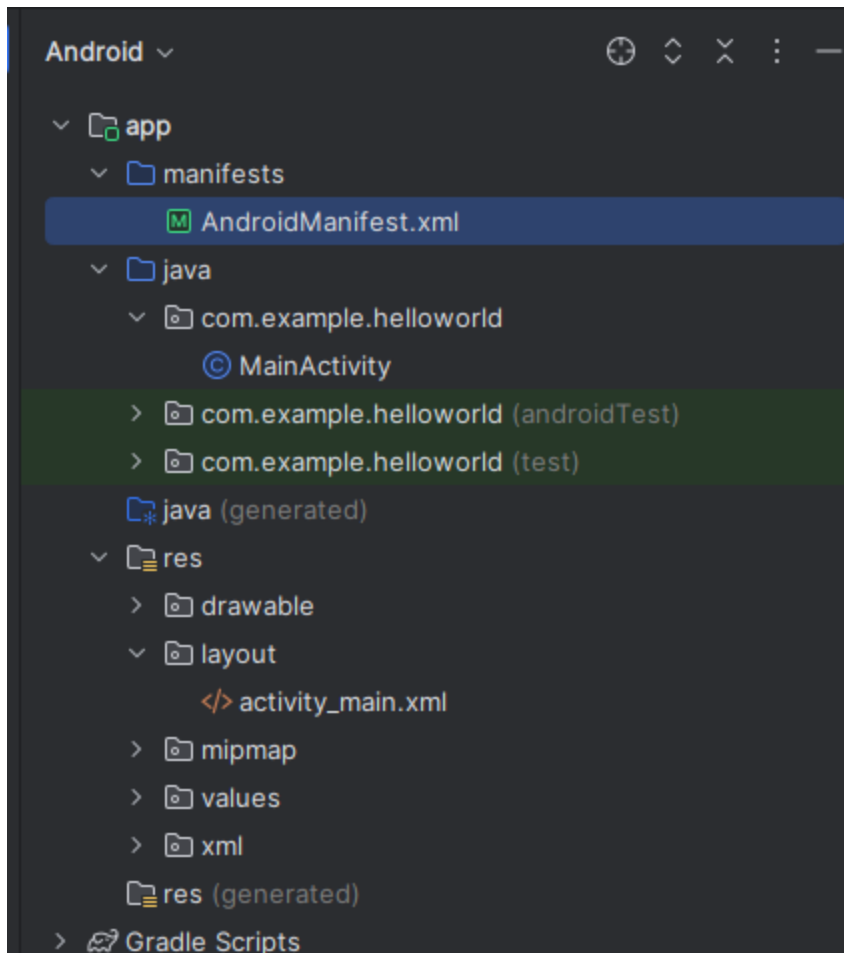


Thư mục res giữ tài nguyên, như layout, chuỗi và hình ảnh. Hành động được kết nối với layout bởi giao diện UI được định nghĩa bằng XML. Đây là tập tin đặt tên theo tên hành động.

2.5 Tìm hiểu về thư mục manifests

Thư mục manifests bao gồm các tập tin cung cấp thông tin quan trọng về hệ thống Android cho ứng dụng của bạn, hệ thống phải có trước khi bạn chạy code.

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



Task 3: Sử dụng máy ảo

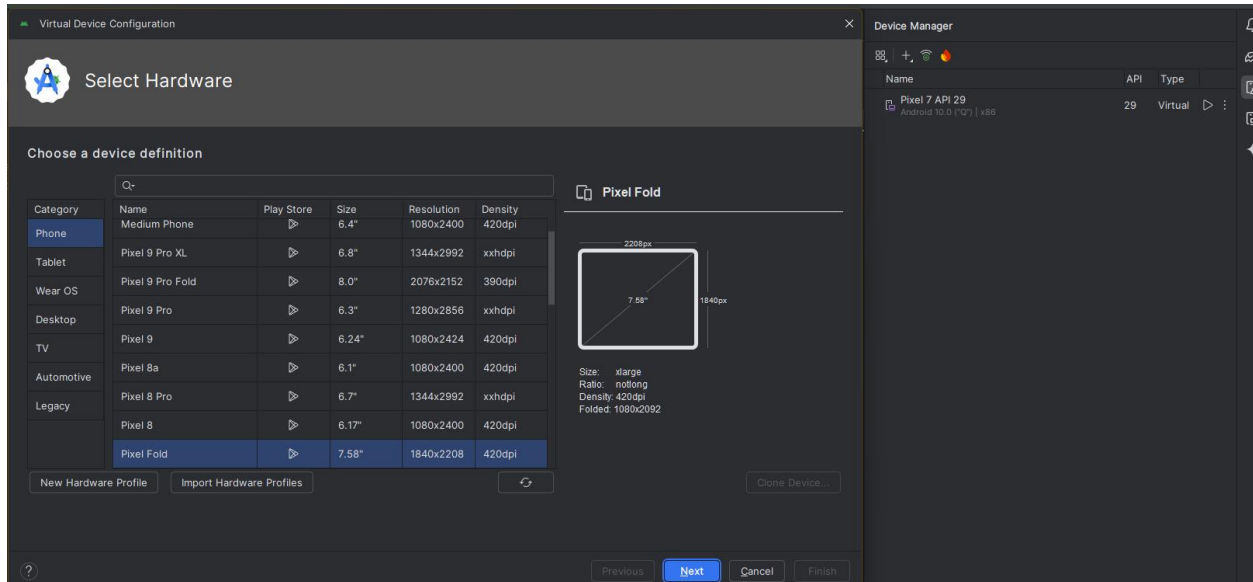
Trong nhiệm vụ này, bạn sẽ sử dụng quản lý thiết bị máy ảo Android để tạo máy ảo đây là giả lập cấu hình cho thiết bị Android và sử dụng máy ảo để chạy ứng dụng. Ghi chú: Máy ảo có yêu cầu thêm so với yêu cầu cơ bản hệ điều hành.

Sử dụng AVD manager, bạn khai báo đặc trưng phần cứng của thiết bị, nó là API level, dung lượng, mẫu mã và các thuộc tính khác và lưu nó thành một máy ảo. Với thiết bị máy ảo, bạn có thể kiểm thử ứng dụng trên các cấu hình thiết bị khác nhau với các cấp API khác nhau mà không cần đến thiết bị vật lý.

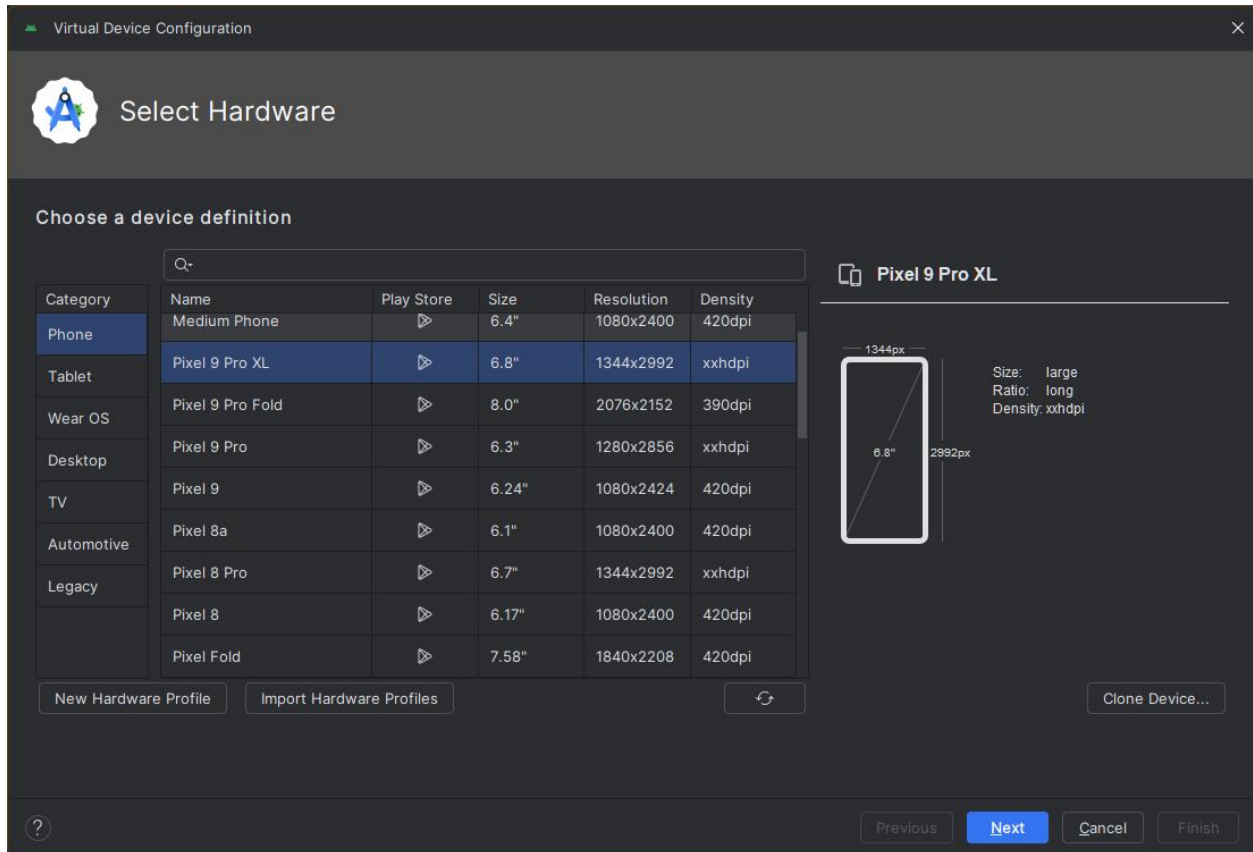
3.1 Tạo máy ảo Android (AVD)

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

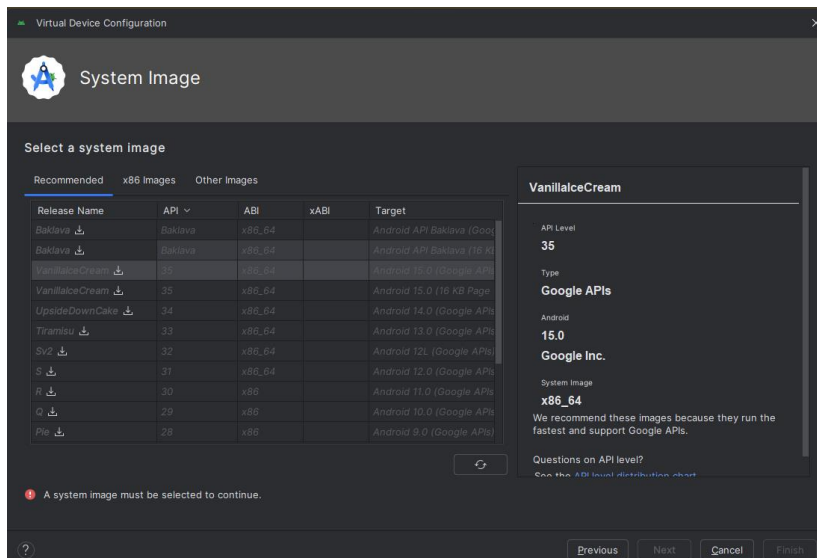
1. Trong Android Studio, chọn Tools > Android > AVD Manager, hoặc chọn biểu tượng AVD Manager tại thanh công cụ. Màn hình máy ảo của bạn sẽ xuất hiện. Nếu bạn đã tạo máy ảo, màn hình hiển thị chúng; nếu không bạn sẽ thấy danh sách trống.



2. Nhấp vào +Tạo thiết bị ảo. Cửa sổ chọn phần cứng xuất hiện hiển thị danh sách các thiết bị phần cứng được cấu hình trước. Đối với mỗi thiết bị, bảng cung cấp một cột cho kích thước hiển thị đường chéo (kích thước), độ phân giải màn hình theo pixel (độ phân giải) và mật độ pixel (mật độ).



3. Chọn một thiết bị như Nexus 5X hoặc Pixel XL và nhấp vào Tiếp theo. Màn hình hình ảnh hệ thống xuất hiện.
4. Nhấp vào tab được đề xuất nếu nó chưa được chọn và chọn phiên bản nào của hệ thống Android để chạy trên thiết bị ảo (như Oreo).

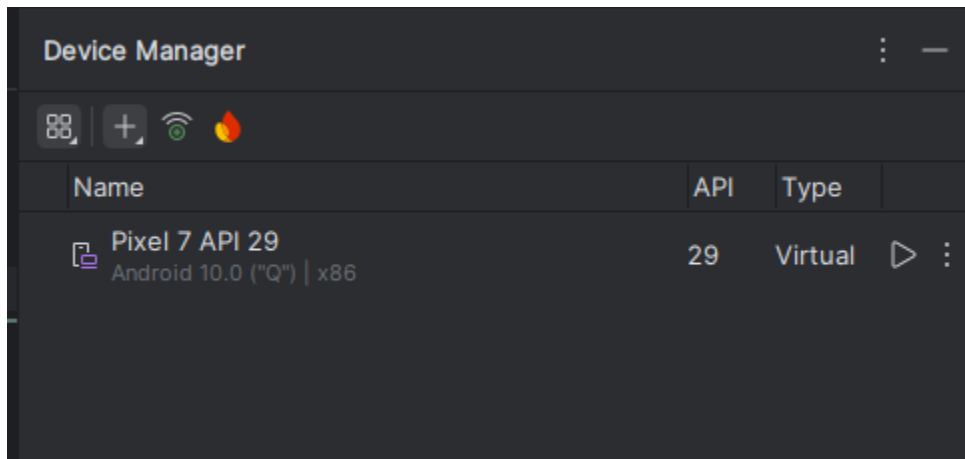


5. Sau chọn hình ảnh hệ thống, chọn NEXT. Máy ảo sẽ hiển thị lên cửa sổ. Bạn có thể thay đổi tên máy ảo. Kiểm tra cấu hình của bạn và chọn FINISH.

3.2 Chạy ứng dụng trên máy ảo

Trong nhiệm vụ này, bạn sẽ chạy ứng dụng HelloWorld.

1. Trong Android Studio, chọn Run > Run app hoặc chọn biểu tượng Run trên thanh công cụ.
2. Cửa sổ Select Deployment Target, bên dưới Available Virtual Devices, chọn máy ảo mà bạn đã tạo và chọn Ok.



Nhiệm vụ 4: (Mở rộng) Sử dụng thiết bị vật lý

Trong nhiệm vụ này, bạn sẽ chạy ứng dụng bằng thiết bị vật lý như điện thoại hoặc máy tính bảng. Bạn nên luôn luôn kiểm tra trên cả thiết bị thật và máy ảo.

Những thứ bạn cần:

- Thiết bị Android vật lý
- Cáp sạc có thể kết nối thiết bị Android với máy tính
- Nếu bạn sử dụng hệ điều hành Linux hoặc Window, bạn có thể cần thêm bước chạy trên phần cứng thiết bị. Kiểm tra tài liệu sử dụng phần cứng thiết bị. Bạn có thể cũng cần cài đặt USB driver tương ứng.

4.1 Bật USB debugging

Để Android Studio có thể giao tiếp với thiết bị của bạn, bạn phải bật chế độ gỡ lỗi USB trên thiết bị Android. Tùy chọn này được bật trong cài đặt **Developer options** (Tùy chọn nhà phát triển) của thiết bị.

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

1. Trên thiết bị của bạn, mở **Cài đặt** (*Settings*), tìm **Giới thiệu về điện thoại** (*About phone*), nhấn vào **Số bản dựng** (*Build number*) bảy lần.
2. Quay lại màn hình trước đó (**Cài đặt / Hệ thống** – *Settings / System*). **Developer options** sẽ xuất hiện trong danh sách. Nhấn vào **Developer options**.
3. Chọn **USB Debugging** (*Gỡ lỗi USB*).

4.2 Chạy ứng dụng của bạn trên thiết bị

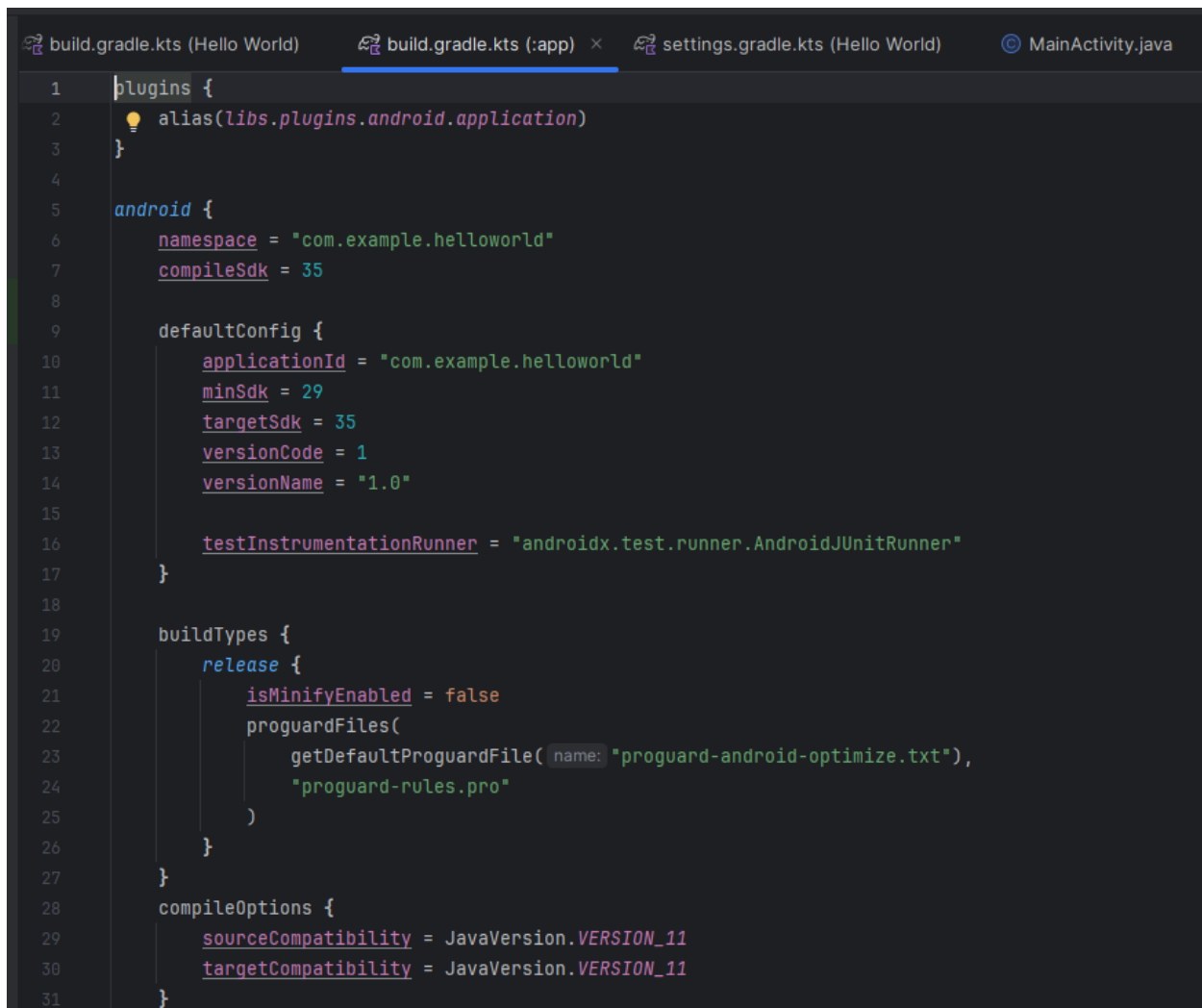
Bây giờ bạn có thể kết nối thiết bị của bạn và chạy ứng dụng từ Android Studio

1. Kết nối thiết bị của bạn với máy bạn phát triển qua cáp USB
2. Chọn nút Run trên thanh công cụ. Cửa sổ Select Deployment Target mở với danh sách cách thiết bị có thể kết nối.
3. Chọn thiết bị của bạn và chọn Ok.

Nhiệm vụ 5: Thay đổi cấu hình Gradle ứng dụng

5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

1. Mở rộng thư mục Gradle Scripts nếu nó không mở, nhấp đúp vào tệp `build.gradle(Module:app)`
2. Với khối cài đặt cấu hình mặc định, thay đổi giá trị của `minSdkversion` thành 29 giống hình ảnh



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

5.2 Đồng bộ hóa cấu hình Gradle mới

Khi bạn thực hiện thay đổi xây dựng cấu hình trong dự án, Android Studio yêu cầu bạn đồng bộ toàn bộ tệp dự án để nó có thể khai báo cấu hình thay đổi và chạy thực hiện kiểm tra chắc chắn cấu hình sẽ không lỗi.

Đồng bộ hóa tệp tin dự án, chọn Sync Now trong thanh thông báo xuất hiện khi thực hiện thay đổi.

Khi Đồng bộ hóa Gradle hoàn thành, thông báo Gradle xây dựng hoàn thành xuất hiện bên dưới cửa sổ làm việc.

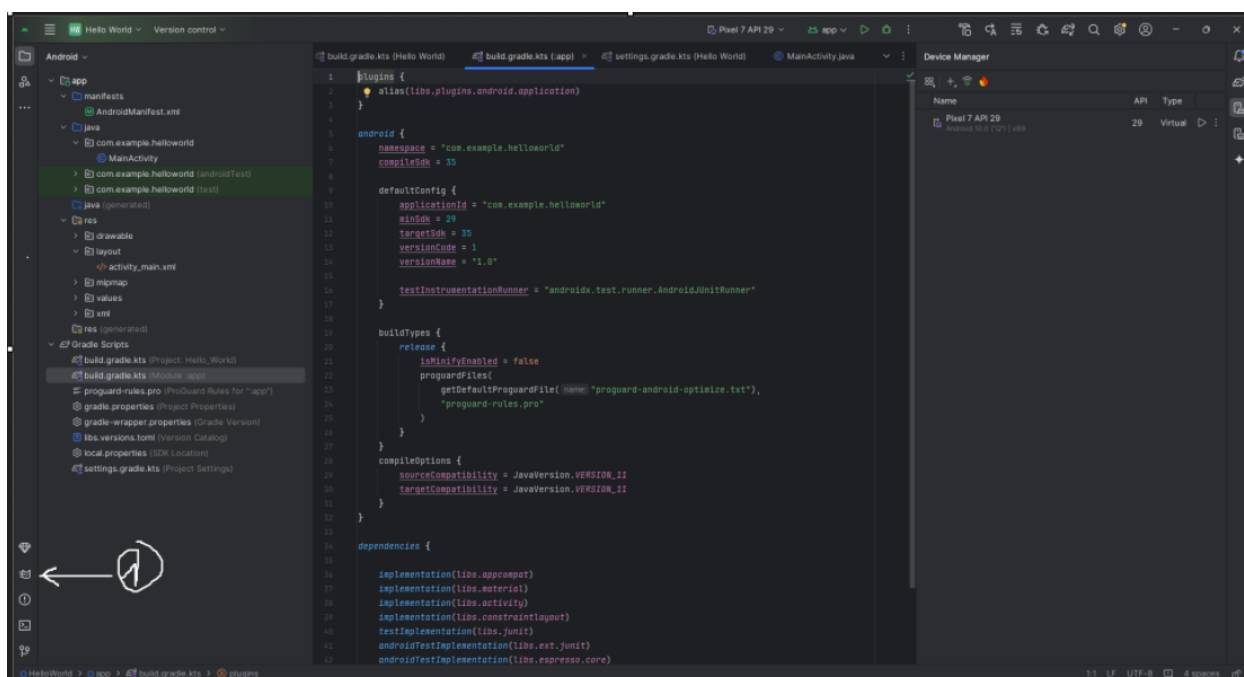
Nhiệm vụ 6: Thêm ghi chú trạng thái vào ứng dụng của bạn

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh **Log** vào ứng dụng của mình, 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 các ngoại lệ.

6.1 Giao diện Logcat pane

Để xem Logcat pane, chọn Logcat tab ở dưới cửa sổ làm việc



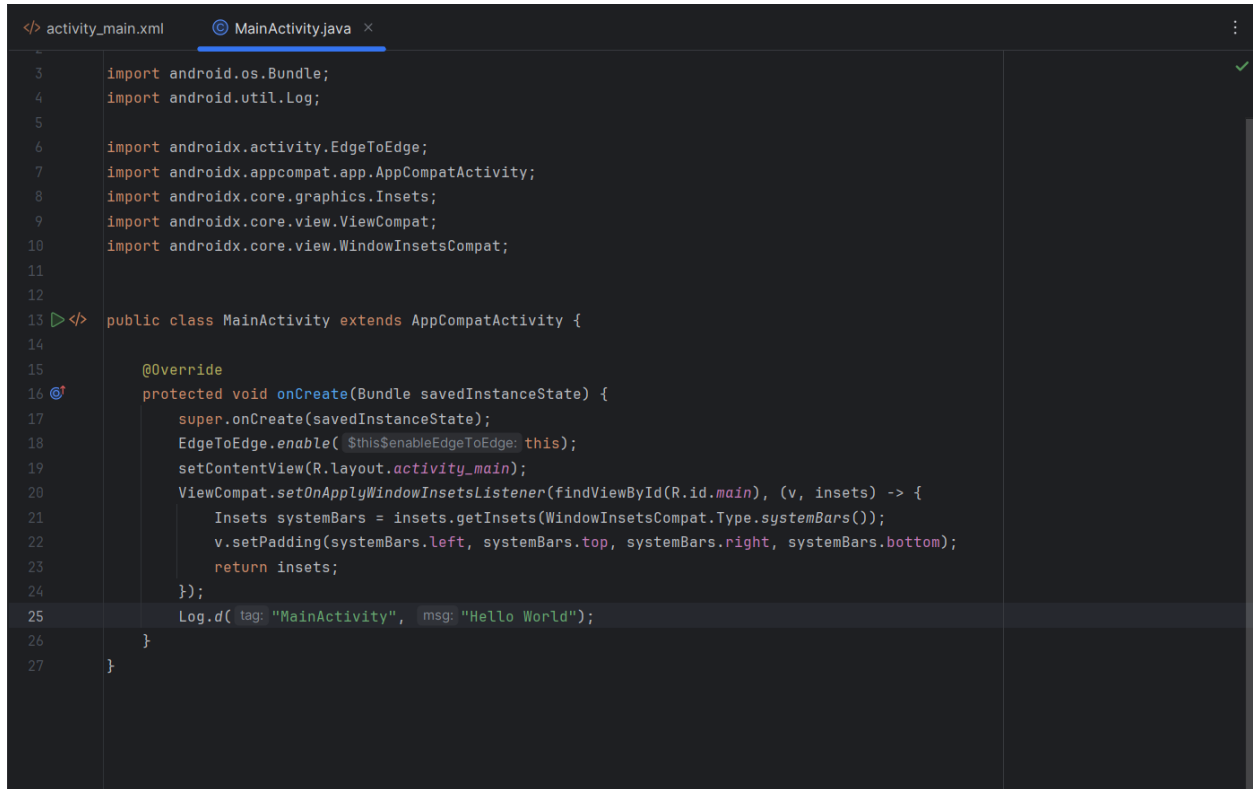
Trong hình bao gồm:

1. Logcat tab mở và đóng Logcat pane, nó hiển thị thông tin về ứng dụng đang chạy của bạn. Nếu bạn thêm trạng thái Log vào ứng dụng của bạn, tin nhắn trạng thái hiển thị tại đây.
2. Cấp menu của Log theo Verbose nó hiển thị trạng thái log của bạn. Các cài đặt khác bao gồm Debug, Error, Info, và Warn.

6.2 Thêm thông báo log vào mã code hiển thị thông báo lên Logcat pane.

Các bước thực hiện:

1. Bật ứng dụng HelloWorld bằng Android Studio, mở MainActivity.
2. Khai báo tự động



```
<?xml version="1.0" encoding="utf-8"?>
<include layout="@layout/activity_main" />
</xml>

import android.os.Bundle;
import android.util.Log;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

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

3. Chọn Editor > General>Auto Import. Chọn tất cả các lựa chọn và cài Insert import on paste to All.
4. Chọn Apply và ấn Ok.
5. Trong phương thức onCreate() của MainActivity, thêm vào câu lệnh:
`Log.d("MainActivity", "Hello World") ;`
6. Nếu logcat hiển thị chưa sẵn sàng, bấm Logcat tab bên dưới Android Studio để mở nó.
7. Kiểm tra tên của mục tiêu và các tên gói của ứng dụng nếu đúng.
8. Thay đổi LogLevel trong logcat pane để gỡ lỗi
9. Chạy ứng dụng.

Tóm tắt

- Để cài đặt Android Studio, truy cập [Android Studio](#) và làm theo hướng dẫn để tải xuống và cài đặt.

- Khi tạo một ứng dụng mới, hãy đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm SDK tối thiểu.
- Để xem cấu trúc thư mục của ứng dụng Android trong cửa sổ Project, nhấp vào tab **Project** trên thanh dọc, sau đó chọn **Android** trong menu bật lên ở phía trên.
- Chỉnh sửa tệp **build.gradle(Module:app)** khi cần thêm thư viện mới vào dự án hoặc thay đổi phiên bản thư viện.
- Tất cả mã nguồn và tài nguyên của ứng dụng được đặt trong thư mục **app** và các thư mục con của nó.
 - Thư mục **java** chứa các tệp mã nguồn Java, bao gồm các hoạt động (activities), kiểm thử (tests), và các thành phần khác.
 - Thư mục **res** chứa tài nguyên như bố cục giao diện (layouts), chuỗi văn bản (strings) và hình ảnh (images).
- Chỉnh sửa tệp **AndroidManifest.xml** để thêm các thành phần và quyền (permissions) cho ứng dụng Android. Mọi thành phần của ứng dụng, chẳng hạn như nhiều activities, phải được khai báo trong tệp XML này.
- Sử dụng [Android Virtual Device \(AVD\) manager](#) để tạo một thiết bị ảo (còn gọi là trình giả lập) để chạy ứng dụng.
- Thêm các câu lệnh **Log** vào ứng dụng để hiển thị thông báo trên **Logcat**, giúp gỡ lỗi cơ bản.
- Để chạy ứng dụng trên một thiết bị Android thực bằng Android Studio, hãy bật **USB Debugging** trên thiết bị.
 - Mở **Settings > About phone** và nhấn **Build number** bảy lần.
 - Quay lại màn hình trước (**Settings**) và chọn **Developer options**.
 - Chọn **USB Debugging**.

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện xuất hiện trên màn hình thiết bị Android bao gồm hệ thống phân cấp gọi là views – tất cả các phần tử trên màn hình. Các lớp View hiển thị các khối xây dựng giao diện cơ bản và các lớp cung cấp tương tác vào thành phần UI như nút, checkbox và các trường text. Thông thường View Subclasses miêu tả bao gồm:

- TextView for displaying text
- EditText cho phép người dùng nhập và chỉnh sửa chữ.
- Button và các phần tử click chuột khác(như RadioButton, CheckBox và Spinner) để cung cấp tương tác hành vi.
- ScrollView và RecyclerView để hiển thị thanh lăn chuột.
- ImageView để hiển thị hình ảnh
- ConstraintLayout và LinearLayout để chứa các phần tử giao diện và vị trí của chúng

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

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị một bố cục được định nghĩa trong tệp activity_main.xml. Tệp này chứa 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 truy xuất dữ liệu từ cơ sở dữ liệu hoặc internet. Bạn sẽ tìm hiểu thêm về lớp Activity trong các bài học khác.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên—một ứng dụng cho phép người dùng tương tác. Bạn sẽ tạo ứng dụng bằng mẫu Empty Activity. Ngoài ra, bạn sẽ học cách sử dụng layout editor để thiết kế bố cục, 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 điều bạn cần biết

Bạn nên làm quen với:

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

Những gì bạn sẽ học

- Cách tạo một ứng dụng có hành vi tương tác.

- Cách sử dụng layout editor để thiết kế bố cục.
- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới. Hãy xem phần Vocabulary words and concepts glossary để có định nghĩa dễ hiểu.

Những gì bạn sẽ làm

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

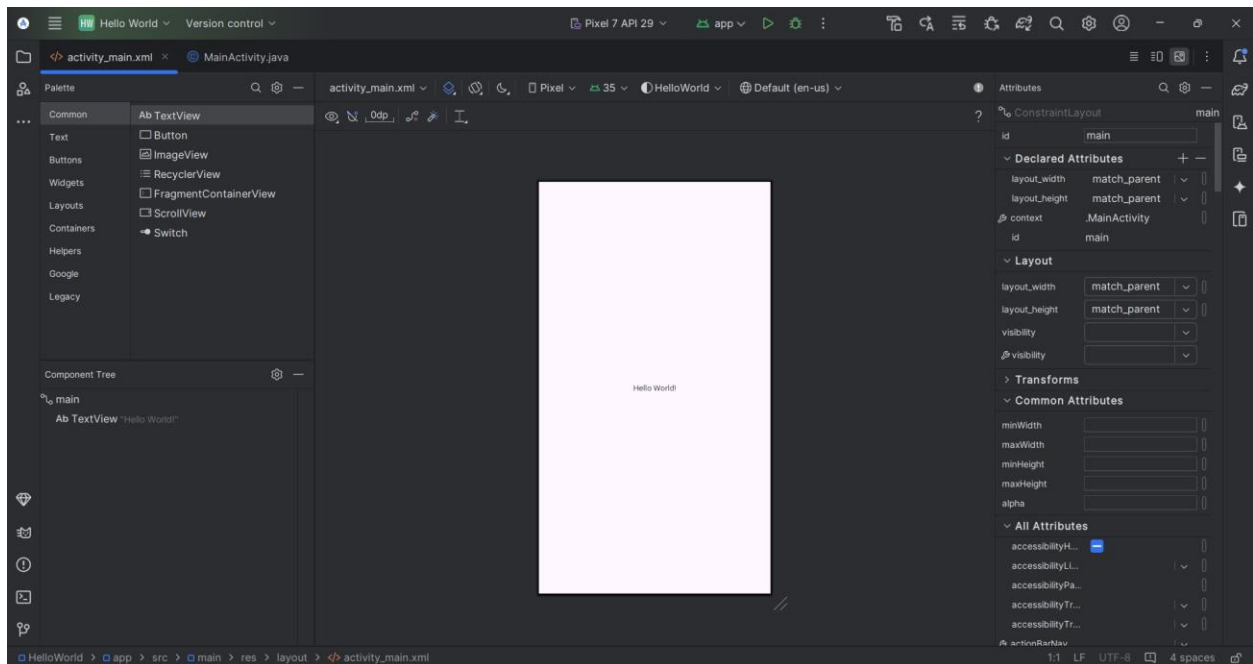
Ứng dụng tổng quan

Ứng dụng HelloToast bao gồm hai phần tử Button và một TextView. Khi người dùng nhấn vào Button thứ nhất, ứng dụng sẽ hiển thị một thông báo ngắn (Toast) trên màn hình. Khi nhấn vào Button thứ hai, bộ đếm số lần nhấn (click counter) trong TextView sẽ tăng lên, bắt đầu từ số 0.

Nhiệm vụ 1: Tạo và tìm hiểu dự án mới

1.1 Tạo dự án Android Studio

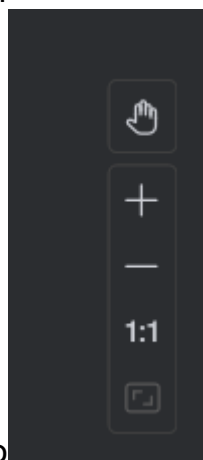
1.2 Tìm hiểu giao diện Layout



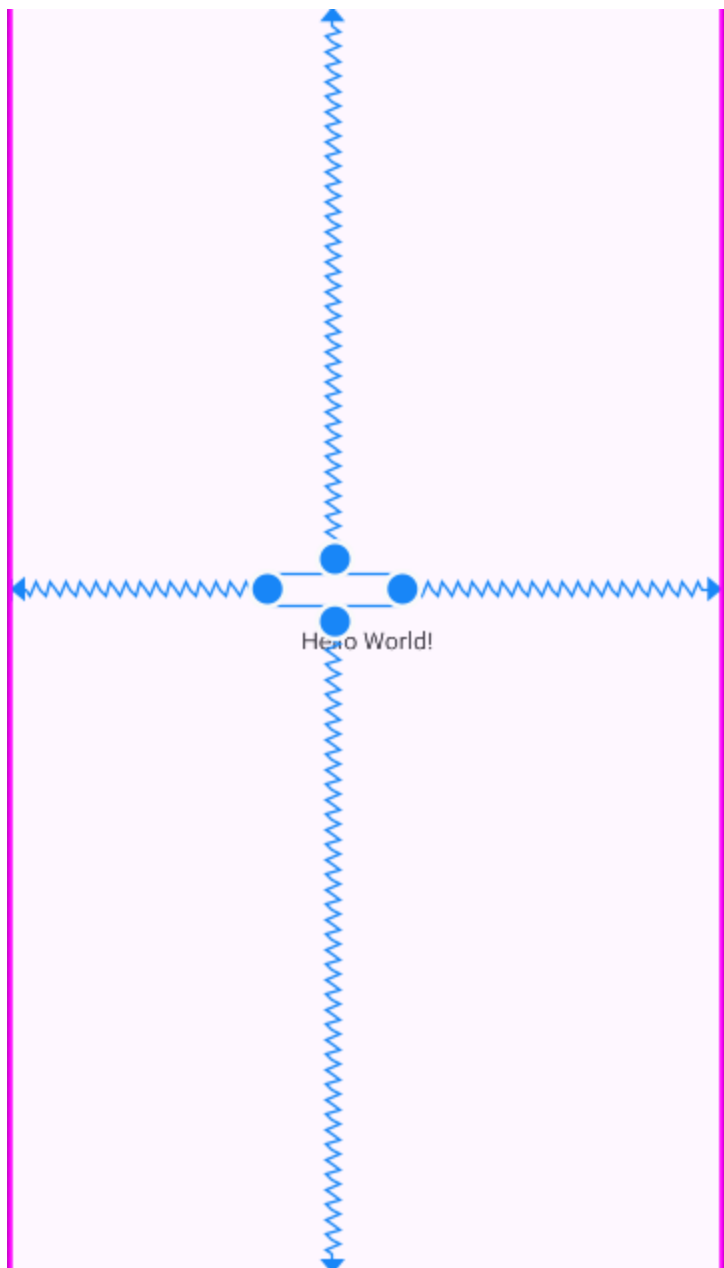
Nhiệm vụ 2: Thêm phân tử vào giao diện

2.1 Ví dụ về phân tử constraints

1. Mở activity_main.xml trong Project>Android pane nếu nó chưa bật sẵn. Nếu Design tab không được chọn, bấm vào nó
2. Công cụ Autoconnect sẽ luôn nằm ở thanh công cụ. Nó cho phép chọn mặc định. Bước tiếp theo, đảm bảo công cụ không bị vô hiệu hóa

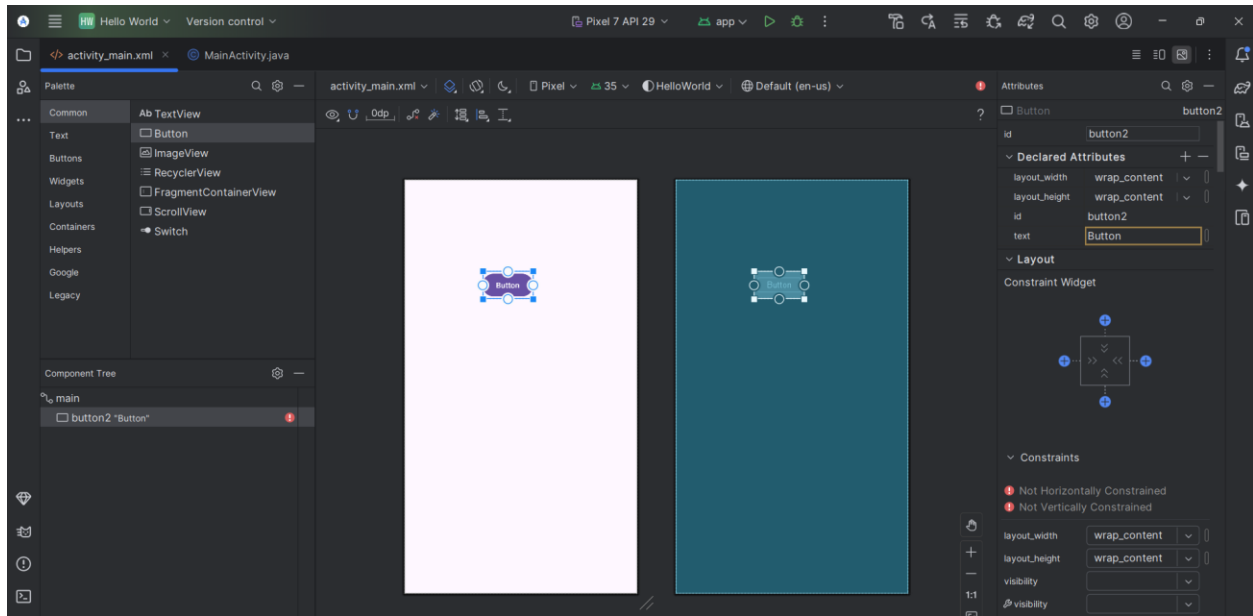


3. Chọn phóng to để phóng đại design và blueprint panes.
4. Chọn TextView trong Component Tree pane. Dòng chữ Hello World được làm nổi bật trong design and blueprint pane và các ràng buộc các phân tử nhìn được

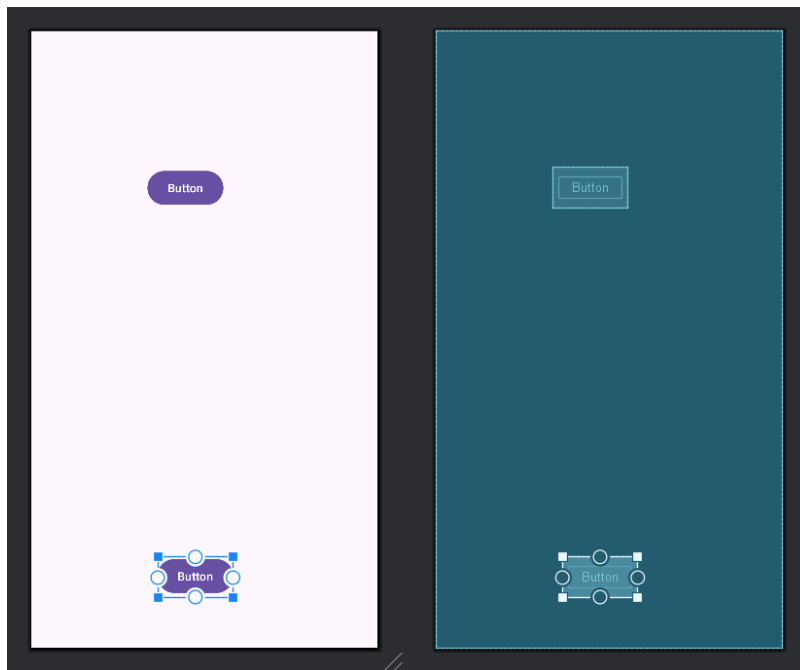


2.2 Thêm nút vào layout

1. Chọn text Hello World , chọn DELETE.
2. Kéo Button từ Palette pane vào vị trí bất kỳ.



2.3 Thêm nút thứ hai vào layout



Nhiệm vụ 3: Thay đổi thuộc tính UI

Bảng thuộc tính (Attributes pane) 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 (UI element). Bạn có thể tìm thấy các thuộc tính (còn được gọi là properties) chung cho tất cả các View trong tài liệu của lớp View.

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

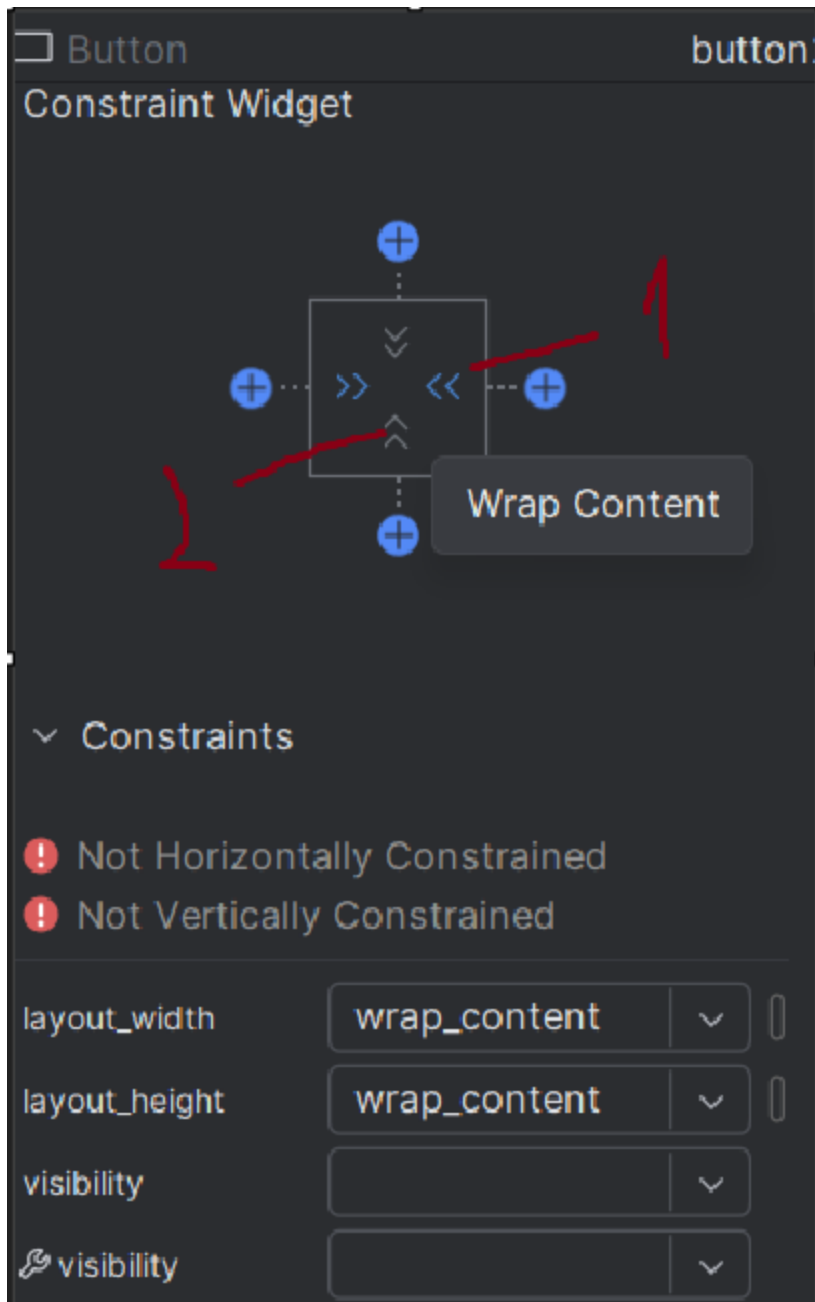
3.1 Thay đổi kích cỡ nút

Trình chỉnh sửa bố cục (layout editor) cung cấp tay cầm thay đổi kích thước (resizing handles) ở cả bốn góc của một View, giúp bạn có thể nhanh chóng thay đổi kích thước của nó. Bạn có thể kéo các tay cầm ở mỗi góc của View để thay đổi kích thước, nhưng cách này sẽ gán cứng (hardcode) kích thước chiều rộng và chiều cao.

Hạn chế gán cứng kích thước cho hầu hết các phần tử View, vì kích thước cố định không thể thích ứng với nội dung và các kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng bảng thuộc tính (Attributes pane) ở bên phải trình chỉnh sửa bố cục để chọn chế độ kích thước không sử dụng giá trị cố định.

Bảng thuộc tính bao gồm một bảng điều chỉnh kích thước hình vuông (view inspector) ở phía trên. Các biểu tượng bên trong hình vuông này đại diện cho các thiết lập chiều rộng và chiều cao như sau:



Trong hình trên:

Điều khiển chiều cao (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 nghiêng chỉ ra rằng điều khiển này đang được đặt thành `wrap_content`, nghĩa là View sẽ mở rộng theo chiều dọc nếu cần để phù hợp với nội dung. Số "8" biểu thị rằng lề tiêu chuẩn được đặt là 8dp.

Điều khiển chiều rộng (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 nghiêng chỉ ra rằng điều khiển này được đặt thành `wrap_content`, nghĩa là View sẽ mở rộng theo chiều ngang nếu cần để phù hợp với nội dung, với lề tối đa 8dp.

Nút đóng bảng thuộc tính (Attributes pane close button): Nhấp vào để đóng bảng.

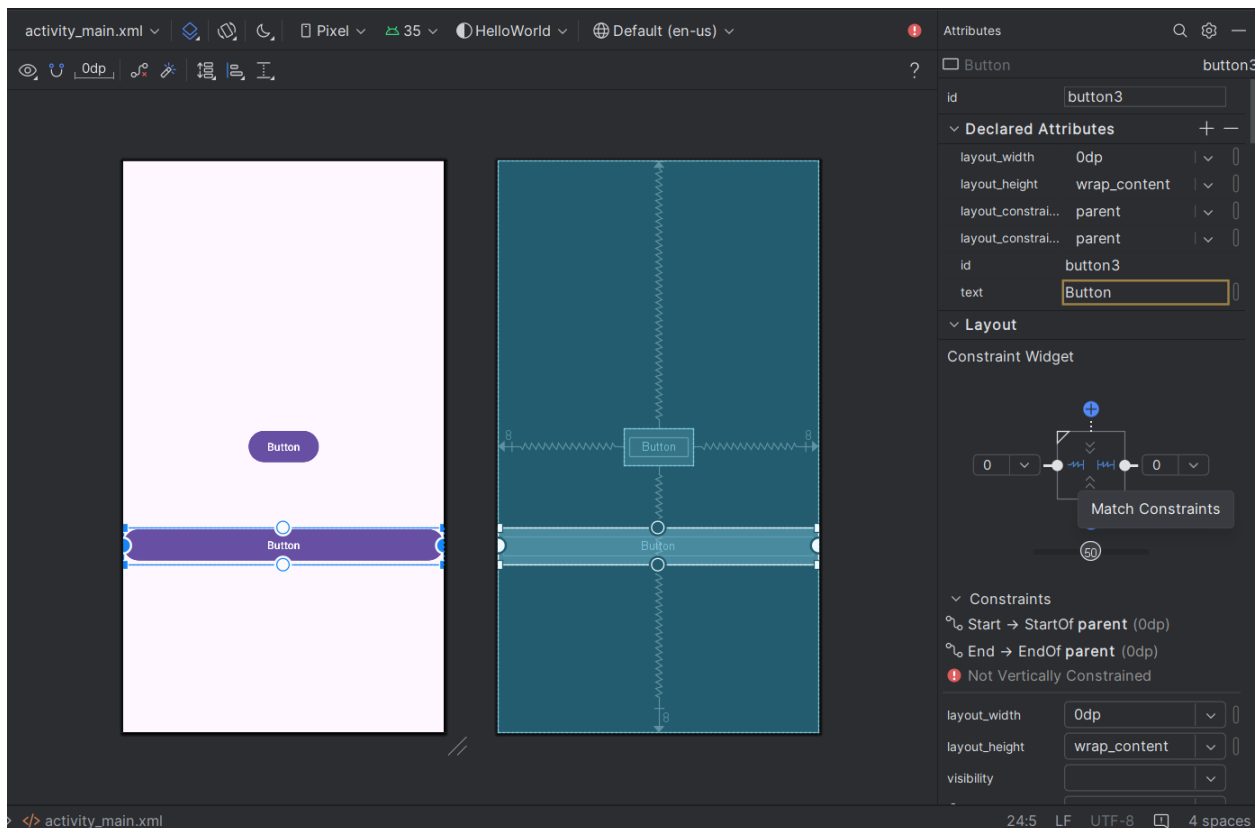
Thực hiện theo các bước sau:

Chọn Button trên cùng trong bảng Component Tree.

Nhấp vào tab Attributes ở bên phải cửa sổ trình chỉnh sửa bố cục.

Nhấp hai lần vào điều khiển chiều rộng:

Lần nhấp đầu tiên thay đổi nó thành Fixed (cố định) với các đường thẳng.



Lần nhấp thứ hai thay đổi nó thành Match Constraints (phù hợp với ràng buộc) với các lò xo đàn hồi (spring coils), như trong hình động bên dưới.

Khi thay đổi điều khiển chiều rộng, thuộc tính layout_width trong Attributes pane sẽ hiển thị giá trị match_constraint, và phần tử Button sẽ kéo dài theo chiều ngang để lấp đầy khoảng trống giữa hai cạnh trái và phải của bố cục.

Chọn Button thứ hai và thực hiện các thay đổi giống như bước trước, như trong hình minh họa.

Như đã thấy trong các bước trước, các thuộc tính layout_width và layout_height trong Attributes pane sẽ thay đổi khi bạn điều chỉnh điều khiển kích thước trong view inspector.

Các thuộc tính này có thể nhận một trong ba giá trị sau trong ConstraintLayout:

`match_constraint`: Mở rộng phần tử View để lấp đầy không gian của parent theo chiều rộng hoặc chiều cao (tùy thuộc vào thiết lập) — nhưng không vượt quá lề (margin) nếu có. Parent ở đây là `ConstraintLayout`. Bạn sẽ tìm hiểu thêm về `ConstraintLayout` trong bài tiếp theo.

`wrap_content`: Thu nhỏ kích thước của View sao cho vừa đủ chứa nội dung của nó. Nếu không có nội dung, View sẽ trở nên vô hình.

Fixed size: Nếu muốn đặt kích thước cố định nhưng vẫn thích ứng với kích thước màn hình thiết bị, hãy sử dụng đơn vị `density-independent pixels (dp)`. Ví dụ, `16dp` có nghĩa là 16 pixel độc lập với mật độ.

💡 **Mẹo**: Nếu bạn thay đổi thuộc tính `layout_width` bằng menu bật lên của nó, giá trị `layout_width` sẽ được đặt thành 0 vì không có kích thước cố định nào được đặt. Giá trị này tương đương với `match_constraint`—tức là View có thể mở rộng tối đa trong phạm vi các ràng buộc và lề đã thiết lập.

3.2 Thay đổi thuộc tính của Button

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

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

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

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

3. Đặt thuộc tính **textColor** thành `@android:color/white`.
4. Chỉnh sửa thuộc tính **text** thành "Toast".
5. Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng **button_count** làm ID, **Count** cho thuộc tính văn bản (**text**), và giữ nguyên màu nền cũng như màu chữ giống như các bước trước.

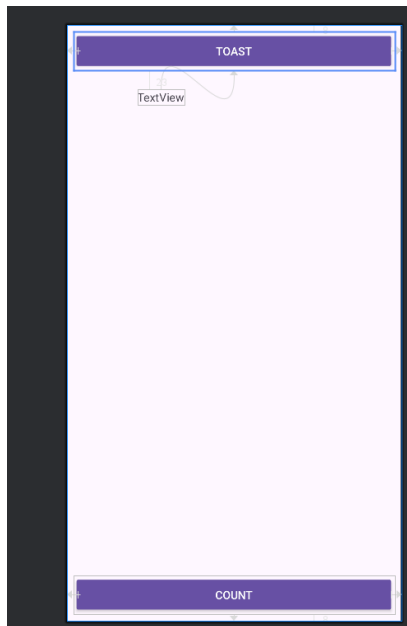
Màu colorPrimary là màu chính của giao diện, một trong các màu cơ bản được định nghĩa trước trong tệp tài nguyên `colors.xml`. Nó được sử dụng cho thanh ứng dụng (**app bar**). Việc sử dụng các màu cơ bản cho các thành phần giao diện khác giúp tạo ra một giao diện đồng nhất. Bạn sẽ tìm hiểu thêm về chủ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

Nhiệm vụ 4: Thêm một **TextView** và thiết lập các thuộc tính

Một trong những lợi ích của **ConstraintLayout** là khả năng căn chỉnh hoặc ràng buộc các phần tử tương đối 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 nút **Button**. Sau đó, bạn sẽ thay đổi các thuộc tính của **TextView** trong ngăn **Attributes**.

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

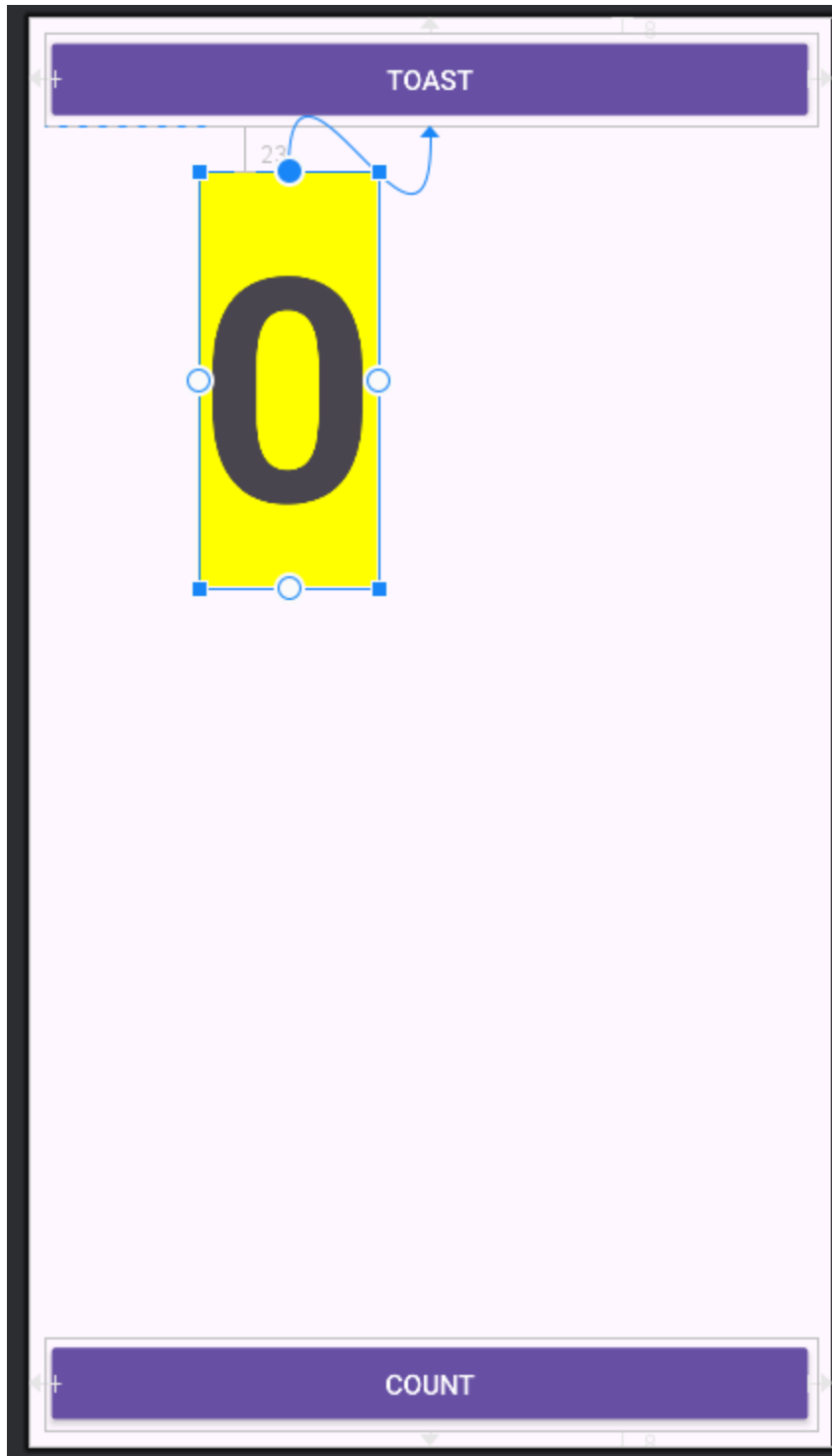
1. Như minh họa trong hình động bên dưới, kéo một **TextView** từ ngăn **Palette** vào phần trên của bố cục, sau đó kéo một ràng buộc từ phía trên của **TextView** đến tay cầm (**handle**) ở dưới của nút **Toast**. Điều này giúp ràng buộc **TextView** nằm bên dưới nút **Button**.
2. Kéo một ràng buộc từ phía dưới của **TextView** đến tay cầm ở trên của nút **Count**, đồng thời kéo ràng buộc từ hai bên của **TextView** đến hai bên của bố cục. Điều này giúp **TextView** được căn giữa trong bố cục, nằm giữa hai nút **Button**.



4.3 Thiết lập thuộc tính cho TextView

Sau khi chọn **TextView**, mở ngăn **Attributes** (nếu chưa mở) và thiết lập các thuộc tính như minh họa trong hình động bên dưới. Các thuộc tính bạn chưa gặp trước đó sẽ được giải thích sau hình.

1. Đặt **ID** thành `show_count`.
2. Đặt thuộc tính **text** thành 0.
3. Đặt **textSize** thành 160sp.
4. Đặt **textStyle** thành **B (bold)** và **textAlignment** thành **ALIGN_CENTER** (căn giữa đoạn văn bản).
5. Thay đổi kích thước ngang (**layout_width**) và kích thước dọc (**layout_height**) thành **match_constraint**.
6. Đặt **textColor** thành `@color/colorPrimary`.
7. Cuộn xuống trong ngăn **Attributes**, nhấn **View all attributes**, tiếp tục cuộn xuống trang thứ hai đến thuộc tính **background**, sau đó nhập `#FFFF00` để đặt màu nền thành **màu vàng**.
8. Cuộn xuống đến thuộc tính **gravity**, mở rộng mục **gravity**, và chọn **center_ver** để căn giữa văn bản theo chiều dọc.



- **textSize:** Kích thước văn bản của **TextView**. Trong bài học này, kích thước được đặt là 160sp.
 - **sp** (scale-independent pixel) là đơn vị tỷ lệ độc lập với mật độ màn hình và cài đặt kích thước phông chữ của người dùng.

- Khi chỉ định kích thước phông chữ, nên sử dụng **dp** để đảm bảo kích thước được điều chỉnh theo cả mật độ màn hình và tùy chỉnh của người dùng.
- **textStyle** và **textAlignment**:
 - **textStyle** được đặt thành **B (bold)** để làm đậm chữ.

textAlignment được đặt thành **ALIGNCENTER**, tức là căn giữa đoạn văn bản.

- **gravity**:
 - Thuộc tính **gravity** xác định cách một **View** được căn chỉnh bên trong **View cha** hoặc **ViewGroup**.
 - Trong bước này, **TextView** được căn giữa theo chiều dọc trong **ConstraintLayout** bằng cách đặt **gravity** thành **center_ver**.

Ngoài ra:

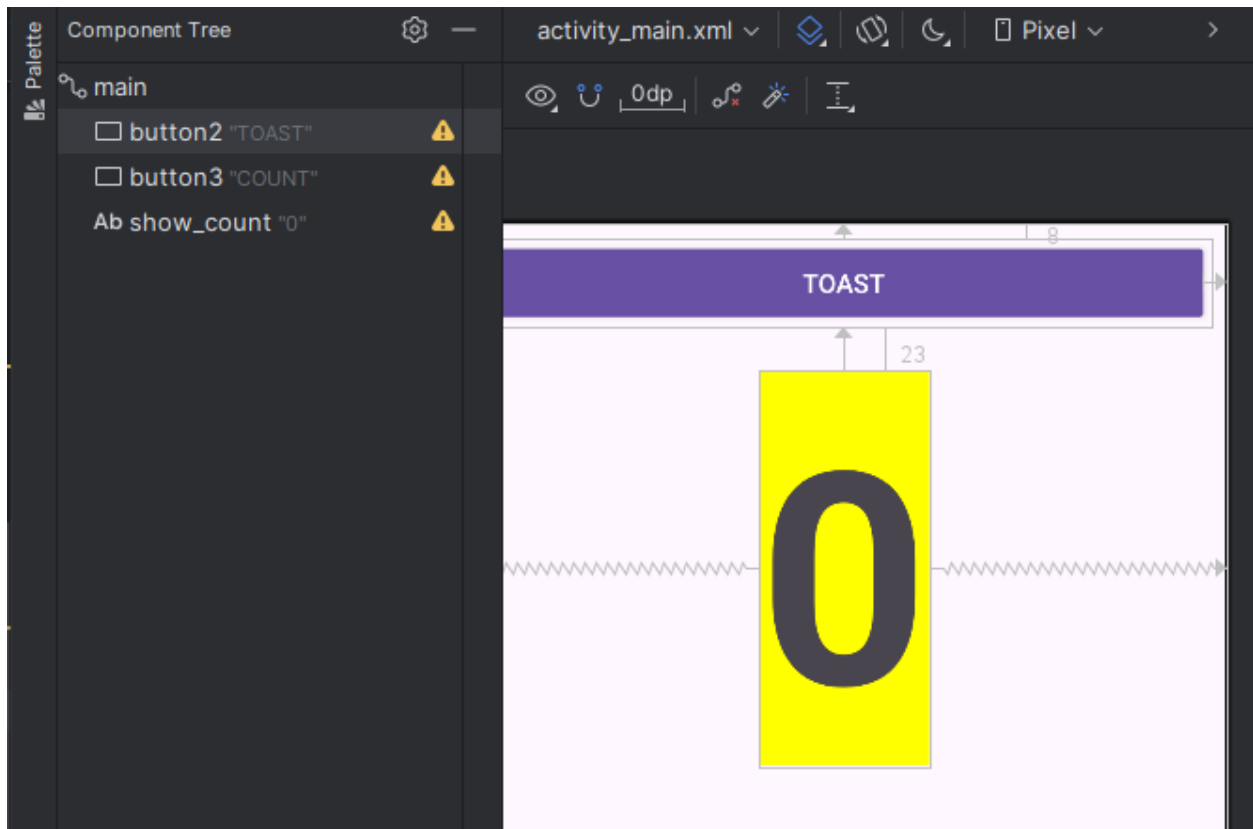
- Bạn có thể nhận thấy rằng thuộc tính **background** xuất hiện ở **trang đầu tiên** của ngăn **Attributes** đối với **Button**, nhưng lại nằm ở **trang thứ hai** đối với **TextView**.
- Ngăn **Attributes** thay đổi tùy theo từng loại **View**:
 - Các thuộc tính phổ biến nhất của loại **View** đó sẽ xuất hiện trên **trang đầu tiên**.
 - Các thuộc tính còn lại sẽ nằm ở **trang thứ hai**.
- Để quay lại **trang đầu tiên** của ngăn **Attributes**, nhấp vào **biểu tượng** trong thanh công cụ ở đầu ngăn.

Nhiệm vụ 5: Chỉnh sửa xml

Bố cục của ứng dụng **Hello Toast** gần như đã hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử giao diện người dùng (**UI element**) trong **Component Tree**.

Di chuột qua các dấu chấm than này để xem thông báo cảnh báo, như minh họa bên dưới.

Tất cả ba phần tử đều hiển thị cùng một cảnh báo: **Chuỗi được mã hóa cứng (hardcoded strings) nên sử dụng tài nguyên (resources)**.



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

5.1 Mở mã bố cục XML

Trong nhiệm vụ này, hãy mở tệp **activity_main.xml** nếu nó chưa được mở, sau đó nhấn vào tab **Text** ở phía dưới của **Layout Editor**.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    android:padding="16dp">

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="TOAST"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button2"
        style="@style/Widget.AppCompat.Button.Colored"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp"
        android:text="COUNT"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/show_count"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="23dp"
        android:background="@color/white"
        android:gravity="center_vertical"
        android:text="0"
        android:textSize="16sp"
        android:textStyle="bold" />

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

Trình chỉnh sửa **XML** sẽ xuất hiện, thay thế các khung thiết kế (**Design**) và bản vẽ (**Blueprint**).

Như bạn có thể thấy trong hình minh họa bên dưới, một phần mã **XML** của bố cục hiển thị các cảnh báo — các chuỗi mã hóa cứng (**hardcoded strings**) "Toast" và "Count". (Chuỗi "0" cũng bị cảnh báo nhưng không hiển thị trong hình.)

Di chuột qua chuỗi mã hóa cứng "Toast" để xem thông báo cảnh báo.

5.1 Trích xuất tài nguyên

Thay vì mã hóa cứng các chuỗi (**hard-coded strings**), một **best practice** là sử dụng **string resources** để quản lý chuỗi tốt hơn, đặc biệt nếu bạn sử dụng nhiều lần.

Việc này cũng quan trọng cho quá trình dịch thuật và nội địa hóa ứng dụng, vì bạn cần tạo một tệp tài nguyên chuỗi riêng cho từng ngôn ngữ.

1. **Các bước thực hiện:**
2. **Nhấp chuột** vào từ "Toast" (cảnh báo đầu tiên được đánh dấu).
3. **Nhấn Alt + Enter** (Windows) hoặc **Option + Enter** (macOS) và chọn **Extract string resource** từ menu bật lên.
4. **Nhập** button_label_toast cho **Resource name**.
5. **Nhấp OK**. Một tài nguyên chuỗi sẽ được tạo trong tệp **res/values/strings.xml**, và chuỗi trong mã sẽ được thay thế bằng:

a) Phần B: Chỉnh sửa bố cục

Giới thiệu

Như bạn đã học trong Phần 1.2A: Giao diện UI 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 cho phép đặt các thành phần UI vào bố cục bằng cách sử dụng các ràng buộc (constraint) kết nối với các phần tử khác và với cạnh của bố cục. ConstraintLayout được thiết kế để giúp bạn dễ dàng kéo các phần tử UI vào trình chỉnh sửa bố cục.

ConstraintLayout là một ViewGroup, tức là một loại View đặc biệt có thể chứa các đối tượng View khác (gọi là con hoặc child views). Trong bài thực hành này, bạn sẽ tìm hiểu thêm về các tính năng của ConstraintLayout và trình chỉnh sửa bố cục.

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

- **LinearLayout:** Một nhóm sắp xếp các phần tử View con theo chiều ngang hoặc dọc.
- **RelativeLayout:** Một nhóm trong đó mỗi phần tử View con được định vị và căn chỉnh dựa trên các phần tử View khác trong ViewGroup. Vị trí của các phần tử con được xác định so với nhau hoặc so với ViewGroup cha.

Những gì bạn cần biết trước

Bạn nên có khả năng:

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

Bạn sẽ học được gì?

- Cách tạo biến thể bố cục cho chế độ hiển thị ngang (landscape).
- Cách tạo biến thể bố cục cho máy tính bảng và màn hình lớn.
- Cách sử dụng ràng buộc đường cơ sở (baseline constraint) để căn chỉnh các phần tử UI với văn bản.
- Cách sử dụng các nút pack và align để căn chỉnh các phần tử trong bố cục.

- Cách định vị các View trong LinearLayout.
- Cách định vị các View trong RelativeLayout.

Bạn sẽ làm gì?

- Tạo biến thể bố cục cho chế độ hiển thị ngang.
- Tạo biến thể bố cục cho máy tính bảng và màn hình lớn.
- Chỉnh sửa bố cục để thêm ràng buộc cho các phần tử UI.
- Sử dụng ràng buộc đường cơ sở (baseline constraints) trong ConstraintLayout để căn chỉnh các phần tử với văn bản.
- Sử dụng các nút pack và align trong ConstraintLayout để căn chỉnh các phần tử.
- Chuyển đổi bố cục sang LinearLayout.
- Định vị các phần tử trong LinearLayout.
- Chuyển đổi bố cục sang RelativeLayout.
- Sắp xếp lại các View trong bố cục chính sao cho chúng có vị trí tương đối với nhau.

Tổng quan về ứng dụng

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

Để thực hành thêm với **ConstraintLayout**, bạn sẽ tạo một **biến thể của bố cục này** dành cho **chế độ hiển thị ngang**, như được minh họa trong hình bên dưới.

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

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

Nhiệm vụ 1: Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu bạn thay đổi bố cục của ứng dụng Hello Toast để nó hiển thị phù hợp ở cả chế độ dọc (portrait) và ngang (landscape). Trong nhiệm vụ này, bạn sẽ học cách tạo các biến thể bố cục một cách dễ dàng hơn cho:

Chế độ ngang (landscape) và dọc (portrait) trên điện thoại. Các màn hình lớn hơn như máy tính bảng (tablet).

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

Thanh công cụ trên cùng giúp bạn tùy chỉnh giao diện xem trước trong trình chỉnh sửa bố cục:

Như trong hình minh họa ở trên:

1. Chọn bề mặt thiết kế (Select Design Surface):

Chọn **Design** để hiển thị bản xem trước màu của bố cục. Chọn **Blueprint** để chỉ hiển thị đường viền của các phần tử UI. Chọn **Design + Blueprint** để xem cả hai cùng lúc.

2. Chế độ hiển thị trong trình chỉnh sửa (Orientation in Editor):

Chọn **Portrait** để xem trước bố cục ở **chế độ dọc**. Chọn **Landscape** để xem trước bố cục ở **chế độ ngang**. Để tạo **biến thể bố cục**, chọn **Create Landscape Variation** hoặc các tùy chọn khác.

3. Chọn thiết bị trong trình chỉnh sửa (Device in Editor):

Chọn loại thiết bị: **Điện thoại/máy tính bảng, Android TV hoặc Android Wear**.

4. Chọn phiên bản API trong trình chỉnh sửa (API Version in Editor):

Chọn phiên bản **Android** để hiển thị bản xem trước.

5. Chọn chủ đề trong trình chỉnh sửa (Theme in Editor):

Chọn chủ đề (ví dụ: **AppTheme**) để áp dụng vào bản xem trước.

6. Chọn ngôn ngữ trong trình chỉnh sửa (Locale in Editor):

Chọn **ngôn ngữ và khu vực** cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có trong **tài nguyên chuỗi (string resources)**. Bạn cũng có thể chọn **Preview as Right To Left** để xem bố cục ở chế độ **RTL (viết**

từ phải sang trái) nếu sử dụng ngôn ngữ như **tiếng Ả Rập hoặc tiếng Hebrew**.

Thanh công cụ thứ hai giúp bạn tùy chỉnh giao diện của các phần tử UI trong **ConstraintLayout** và phóng to, thu nhỏ và di chuyển bản xem trước:

1. **Trong hình minh họa trên:**
2. **Hiện thị (Show):**
 - Chọn **Show Constraints** và **Show Margins** để hiện thị hoặc ẩn ràng buộc và khoảng cách lề trong bản xem trước.
3. **Tự động kết nối (Autoconnect):**
 - Bật/tắt **Autoconnect**.
 - Khi bật, bạn có thể kéo một phần tử (ví dụ: **Button**) vào bất kỳ vị trí nào trong bố cục, hệ thống sẽ **tự động tạo ràng buộc** với bố cục cha.
4. **Xóa tất cả ràng buộc (Clear All Constraints):**
 - Xóa **toàn bộ ràng buộc** trong bố cục.
5. **Suy luận ràng buộc (Infer Constraints):**
 - Tạo **ràng buộc tự động** dựa trên vị trí hiện tại của các phần tử.
6. **Lề mặc định (Default Margins):**
 - Cài đặt **khoảng cách lề mặc định** cho các phần tử.
7. **Gói gọn/mở rộng (Pack):**
 - **Gói gọn hoặc mở rộng** các phần tử đã chọn.
8. **Căn chỉnh (Align):**
 - **Căn chỉnh** các phần tử đã chọn theo một quy tắc nhất định.
9. **Hướng dẫn (Guidelines):**
 - Thêm **đường hướng dẫn** theo chiều dọc hoặc ngang để bố cục dễ căn chỉnh hơn.
10. **Điều khiển thu phóng (Zoom/pan controls):**
 - **Phóng to, thu nhỏ hoặc di chuyển** bản xem trước.

Mẹo:

- Để tìm hiểu thêm về trình chỉnh sửa bố cục, xem bài **Xây dựng UI với Layout Editor**.
- Để tìm hiểu thêm về cách tạo bố cục với **ConstraintLayout**, xem bài **Xây dựng UI linh hoạt với ConstraintLayout**.

1.1 Xem trước bố cục ở chế độ ngang

Để xem trước bố cục của ứng dụng **Hello Toast** ở chế độ ngang (**horizontal orientation**), thực hiện các bước sau:

1. **Mở ứng dụng Hello Toast** từ bài học trước.

Lưu ý: Nếu bạn đã tải xuống mã nguồn hoàn chỉnh của **HelloToast**, hãy xóa các bố cục **landscape** và **extra-large** hiện có.

- Chuyển từ **Project > Android** sang **Project > Project Files** trong **Project pane**.
 - Mở thư mục: `app > src/main > res`.
 - Chọn **layout-land** và **layout-xlarge**, sau đó vào **Edit > Delete**.
 - Chuyển **Project pane** về lại **Project > Android**.
2. **Mở tệp `activity_main.xml`** và nhấp vào **tab Design** nếu chưa được chọn.
 3. **Nhấp vào nút "Orientation in Editor"** trên thanh công cụ trên cùng.
 4. **Chọn "Switch to Landscape"** trong menu thả xuống.
 - Bố cục sẽ hiển thị theo **hướng ngang**.
 - Để quay lại chế độ **dọc**, chọn **Switch to Portrait**.

- 1.3) Trình chỉnh sửa bố cục**
- 1.4) Văn bản và các chế độ cuộn**
- 1.5) Tài nguyên có sẵn**

Bài 2) Activities

- 2.1) Activity và Intent**
- 2.2) Vòng đời của Activity và trạng thái**
- 2.3) Intent ngầm định**

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

- 3.1) Trình gỡ lỗi**
- 3.2) Kiểm thử đơn vị**
- 3.3) Thư viện hỗ trợ**

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

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

Bài 2) Trải nghiệm người dùng thú vị

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

Bài 3) Kiểm thử giao diện người dùng

- 3.1) Espresso cho việc kiểm tra UI

CHƯƠNG 3. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

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

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

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

CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

1.1) Shared preferences

1.2) Cài đặt ứng dụng

Bài 2) Lưu trữ dữ liệu với Room

2.1) Room, LiveData và ViewModel

2.2) Room, LiveData và ViewModel