

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



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

SINH VIÊN: HOÀNG TRỌNG VINH -2251061925 – 64CNTT1

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1. Làm quen.....	4
Bài 1) Tạo ứng dụng đầu tiên.....	4
1.1) Android Studio và Hello World	4
1.2) Giao diện người dùng tương tác đầu tiên.....	5
1.3) Trình chỉnh sửa bố cục.....	5
1.4) Văn bản và các chế độ cuộn.....	5
1.5) Tài nguyên có sẵn.....	5
Bài 2) Activities.....	5
2.1) Activity và Intent.....	5
2.2) Vòng đời của Activity và trạng thái.....	5
2.3) Intent ngầm định.....	5
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	5
3.1) Trình gỡ lỗi.....	5
3.2) Kiểm thử đơn vị.....	5
3.3) Thư viện hỗ trợ.....	5
CHƯƠNG 2. Trải nghiệm người dùng.....	6
Bài 1) Tương tác người dùng.....	6
1.1) Hình ảnh có thể chọn.....	6
1.2) Các điều khiển nhập liệu.....	6
1.3) Menu và bộ chọn.....	6
1.4) Điều hướng người dùng.....	6
1.5) RecyclerView.....	6
Bài 2) Trải nghiệm người dùng thú vị.....	6
2.1) Hình vẽ, định kiểu và chủ đề.....	6
2.2) Thẻ và màu sắc.....	6
2.3) Bố cục thích ứng	6
Bài 3) Kiểm thử giao diện người dùng.....	6
3.1) Espresso cho việc kiểm tra UI.....	6
CHƯƠNG 3. Làm việc trong nền.....	6

Bài 1) Các tác vụ nền	6
1.1) AsyncTask.....	6
1.2) AsyncTask và AsyncTaskLoader.....	6
1.3) Broadcast receivers.....	6
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	6
2.1) Thông báo.....	6
2.2) Trình quản lý cảnh báo.....	6
2.3) JobScheduler	6
CHƯƠNG 4. Lưu trữ dữ liệu người dùng.....	7
Bài 1) Tùy chọn và cài đặt.....	7
1.1) Shared preferences.....	7
1.2) Cài đặt ứng dụng	7
Bài 2) Lưu trữ dữ liệu với Room.....	7
2.1) Room, LiveData và ViewModel.....	7
2.2) Room, LiveData và ViewModel	7
3.1) Trình gỡ lỗi	

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

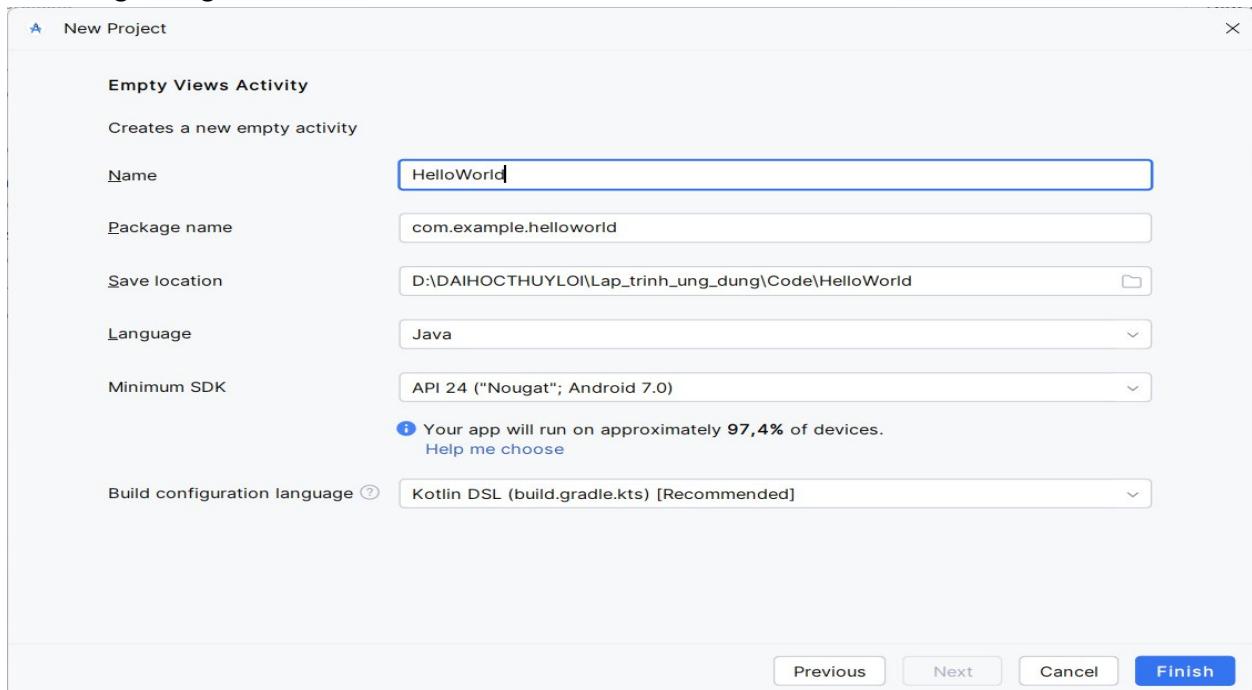
Giới thiệu

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.

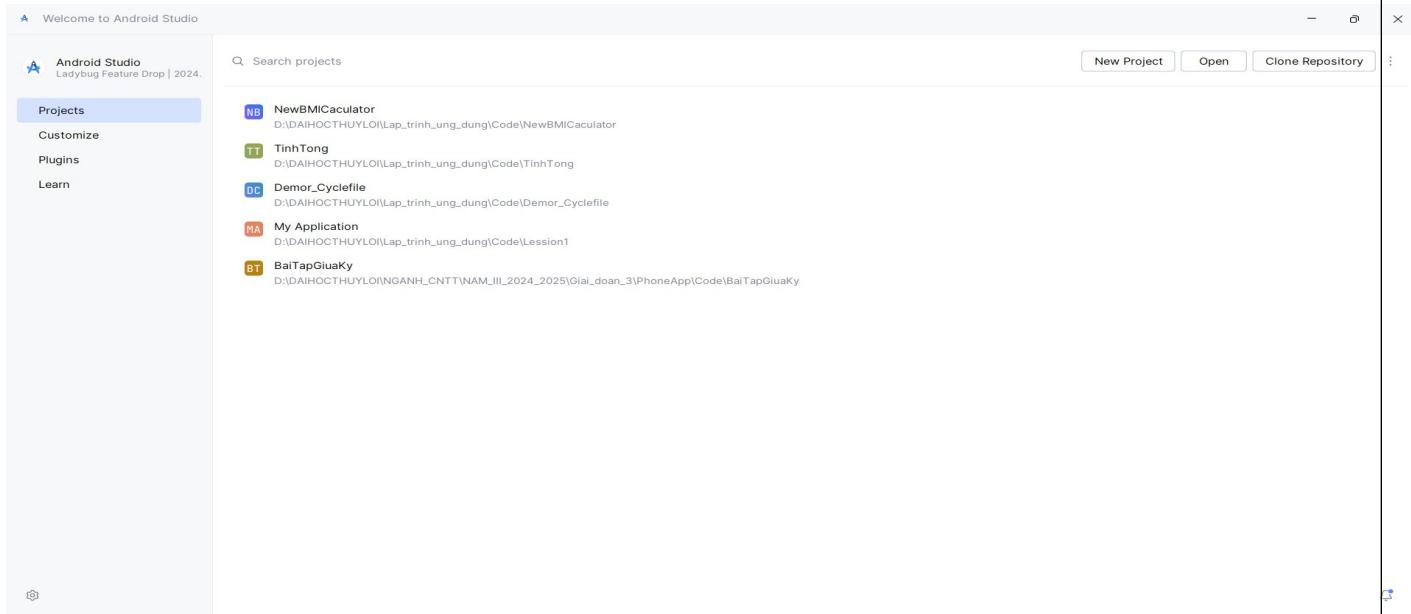
- Cách thêm thông điệp ghi lại vào ứng dụng của bạn để phục vụ mục đích gỡ lỗi.

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**

TASK 1:Cài đặt Android Studio

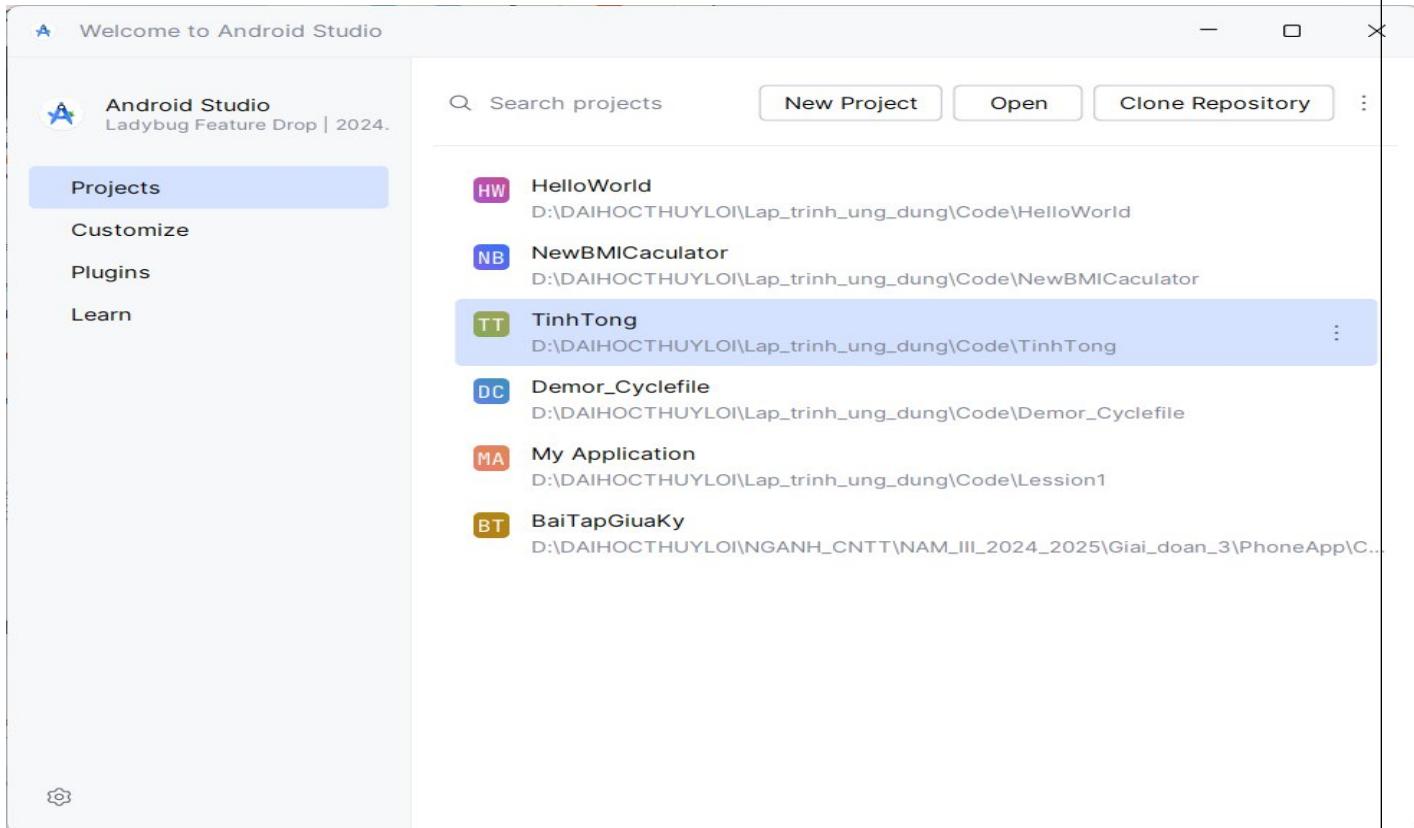
Ứng dụng đã được cài và có giao diện như trên khi đã hoàn tất cài đặt



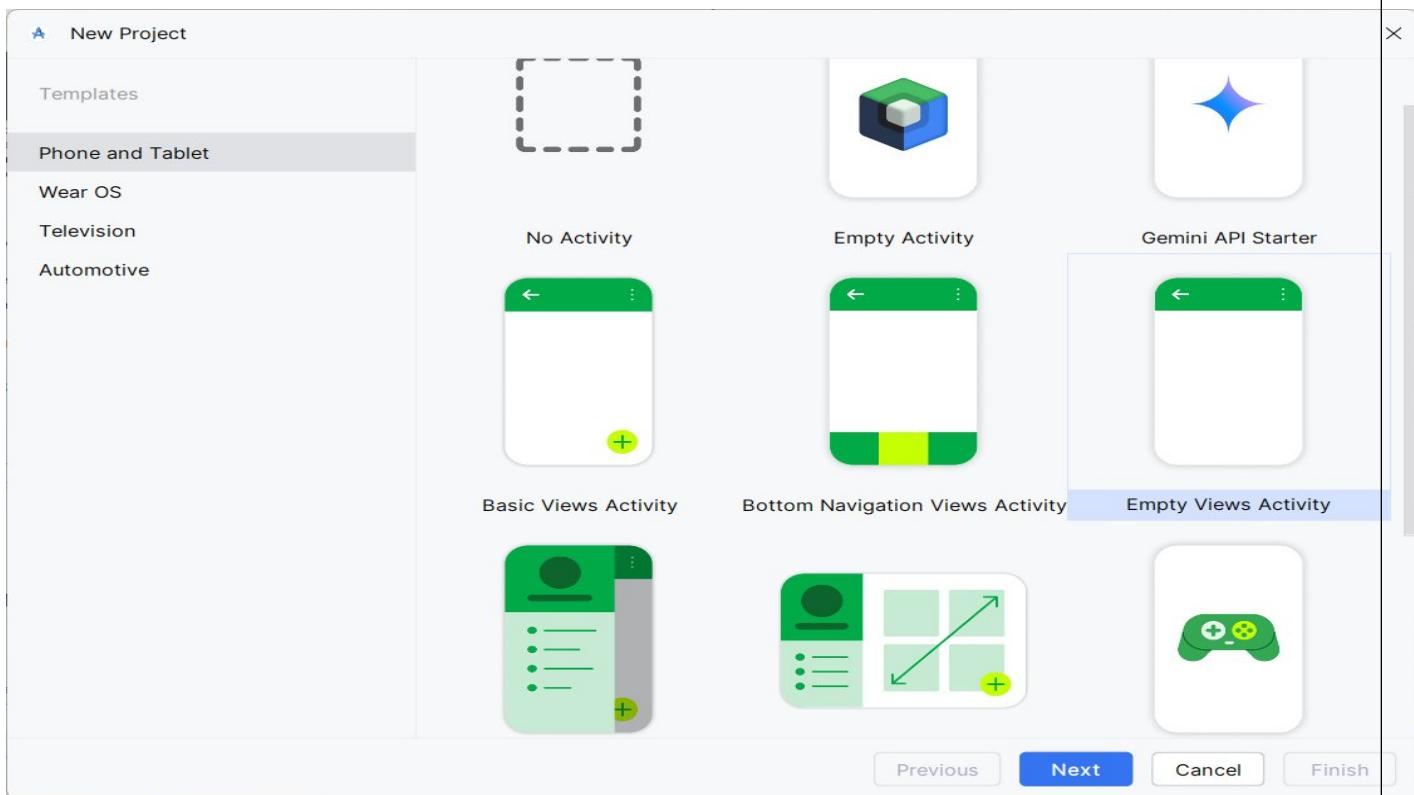
TASK 2:Tạo ứng dụng Hello World

2.1 Tạo một dự án

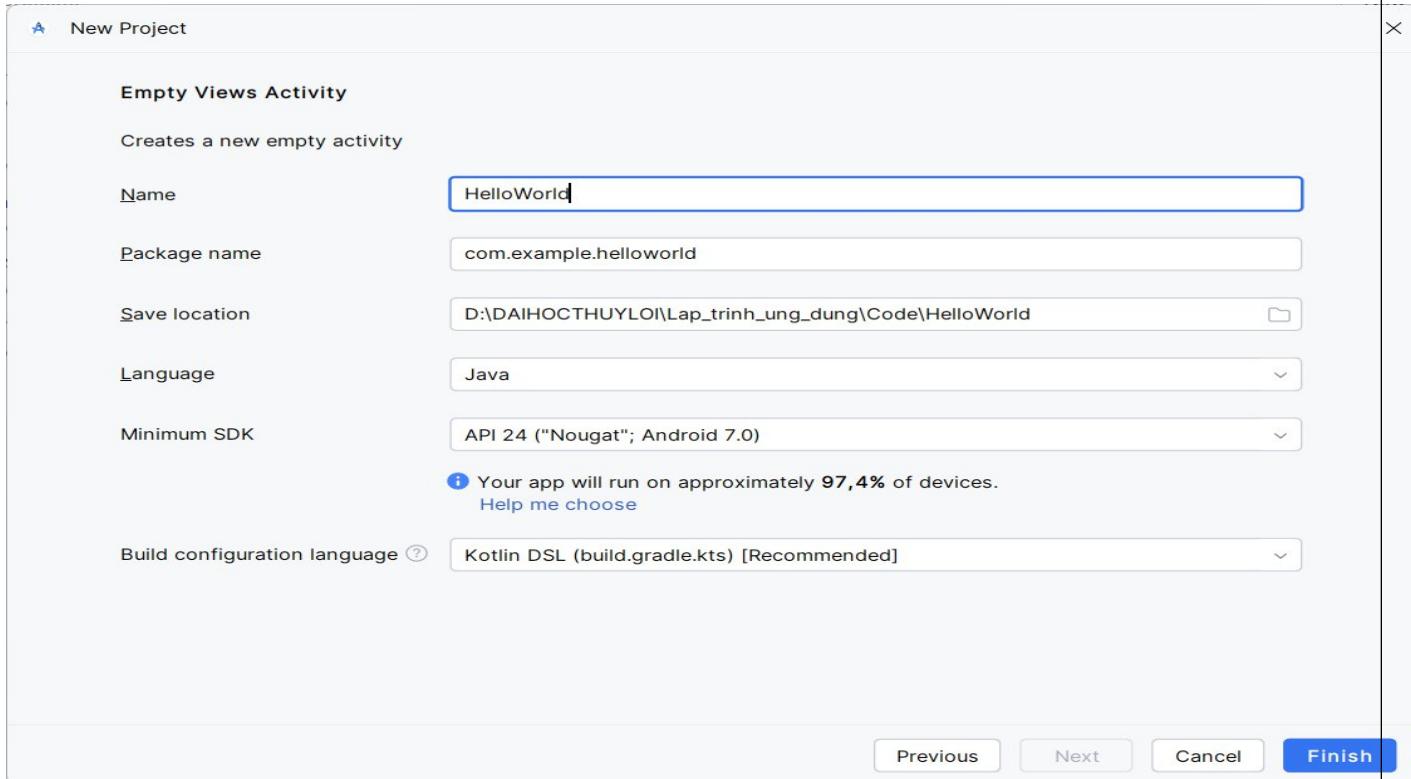
Ở trang giao diện chọn new project



Sau đó chọn Empty Views Activity

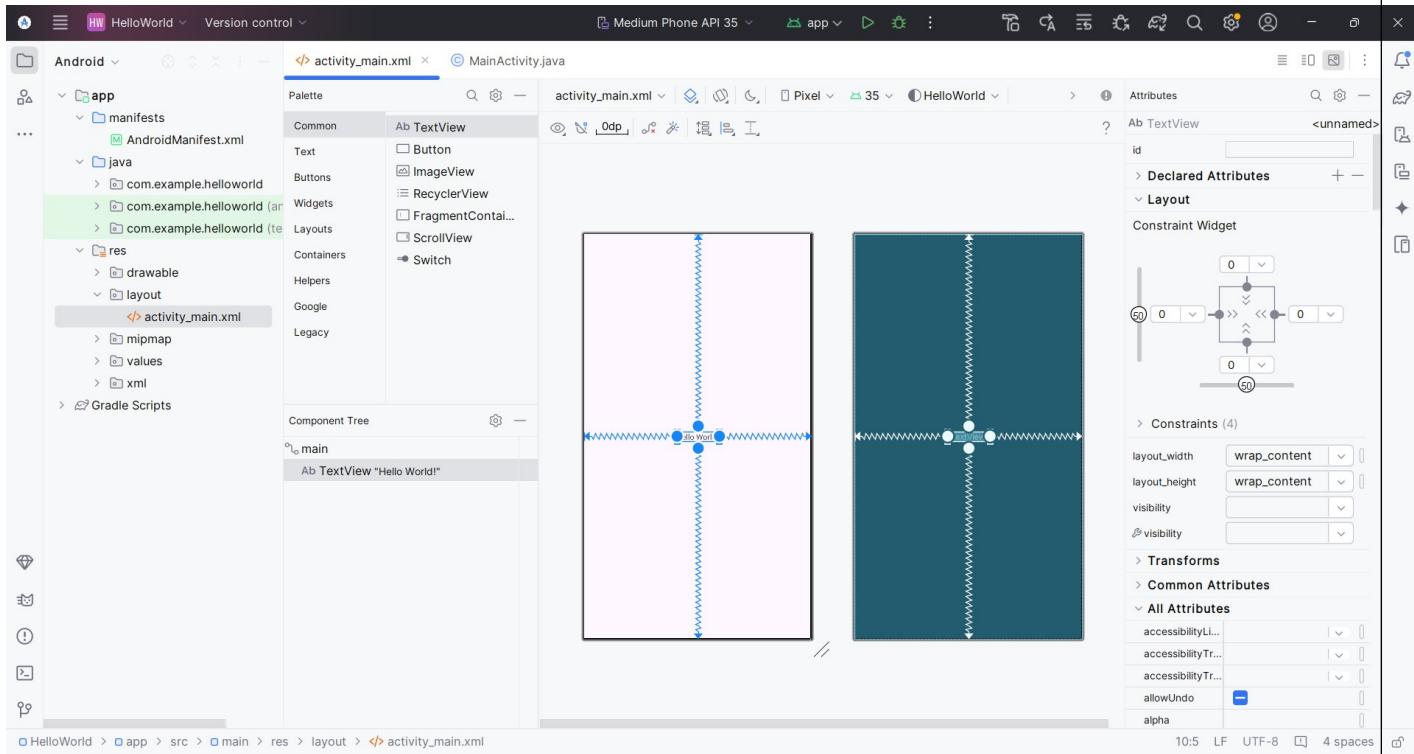


Sau đó ấn **next** rồi sẽ điền tên dự án , chọn ngôn ngữ code là java hoặc kotlin sau đó ấn **Finish**



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

1. Nhập vào tab activity_main.xml để xem trình chỉnh sửa bố cục.
2. Nhập vào tab Thiết kế của trình chỉnh sửa bố cục, nếu chưa chọn, để hiển thị bản trình bày đồ họa của bố cục như được hiển thị bên dưới.



3. Nhập vào tab MainActivity.java để xem trình soạn thảo mã như hiển thị bên dưới.

```

1 package com.example.helloworld;
2
3 > import ...
4
5 > public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
11        setContentView(R.layout.activity_main);
12        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
13            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
14            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
15        });
16    }
17
18 }
19
20
21
22
23
24

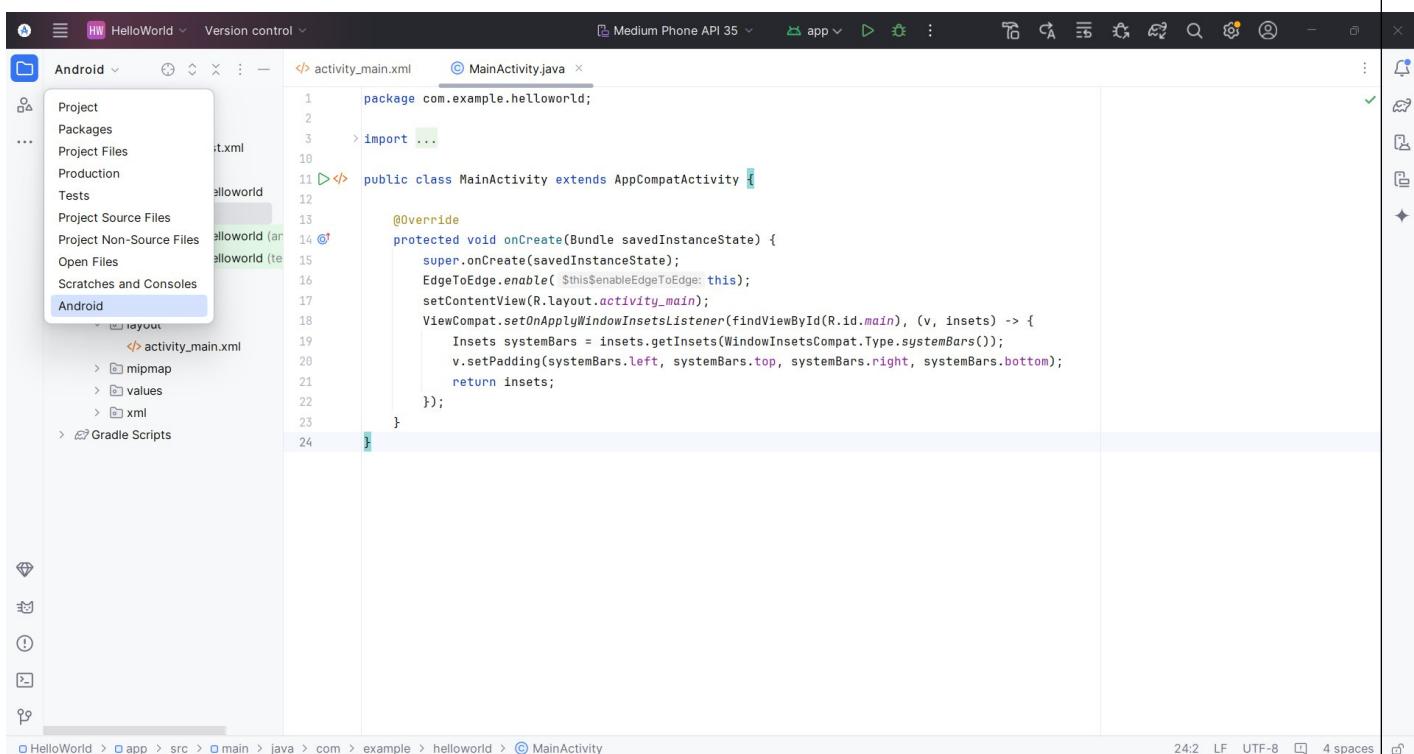
```

The screenshot shows the code editor for MainActivity.java. The code defines a public class MainActivity that extends AppCompatActivity. It overrides the onCreate method to set the content view to R.layout.activity_main and handles window insets using ViewCompat.setOnApplyWindowInsetsListener. The code is well-formatted with proper indentation and syntax highlighting.

2.2 Khám phá ngăn Project > Android

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

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

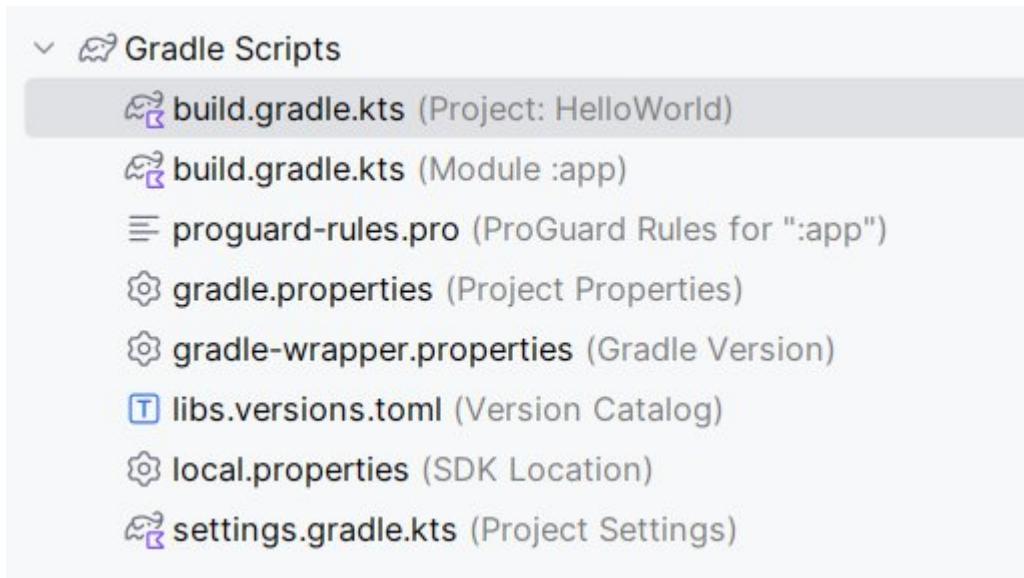


Lưu ý: Chương này và các chương khác đề cập đến khung Dự án, khi được đặt thành Android, là Dự án > Khung Android.

2.3 Khám phá thư mục Gradle Scripts

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

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



Hãy làm theo các bước sau để khám phá hệ thống Gradle:

1. Nếu thư mục Tập lệnh Gradle không được mở rộng, hãy nhấp vào hình tam giác để mở rộng.

Thư mục này chứa tất cả các tệp cần thiết cho hệ thống xây dựng.

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

Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các module tạo nên dự án của bạn. Mỗi dự án Android Studio đều chứa một tệp Gradle build cấp cao nhất. Trong hầu hết các trường hợp, bạn sẽ không cần thay đổi gì trong tệp này, nhưng việc hiểu nội dung của nó vẫn rất hữu ích.

Theo mặc định, tệp build cấp cao nhất sử dụng khôi buildscript để định nghĩa các kho lưu trữ (repositories) và các phụ thuộc (dependencies) của Gradle chung cho tất cả các module trong dự án. Khi phụ thuộc của bạn không phải là một thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm kiếm các tệp trong các kho lưu trữ trực tuyến được chỉ định trong khôi repositories của tệp này. Theo mặc định, các dự án Android Studio mới khai báo JCenter và Google (bao gồm cả kho lưu trữ Maven của Google) làm các vị trí kho lưu trữ.

The screenshot shows the Android Studio interface with the project 'HelloWorld' open. The left sidebar displays the project structure under 'Android'. The main editor window shows the 'build.gradle.kts (HelloWorld)' file, which contains the following code:

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
    alias(libs.plugins.android.application) apply false
}
```

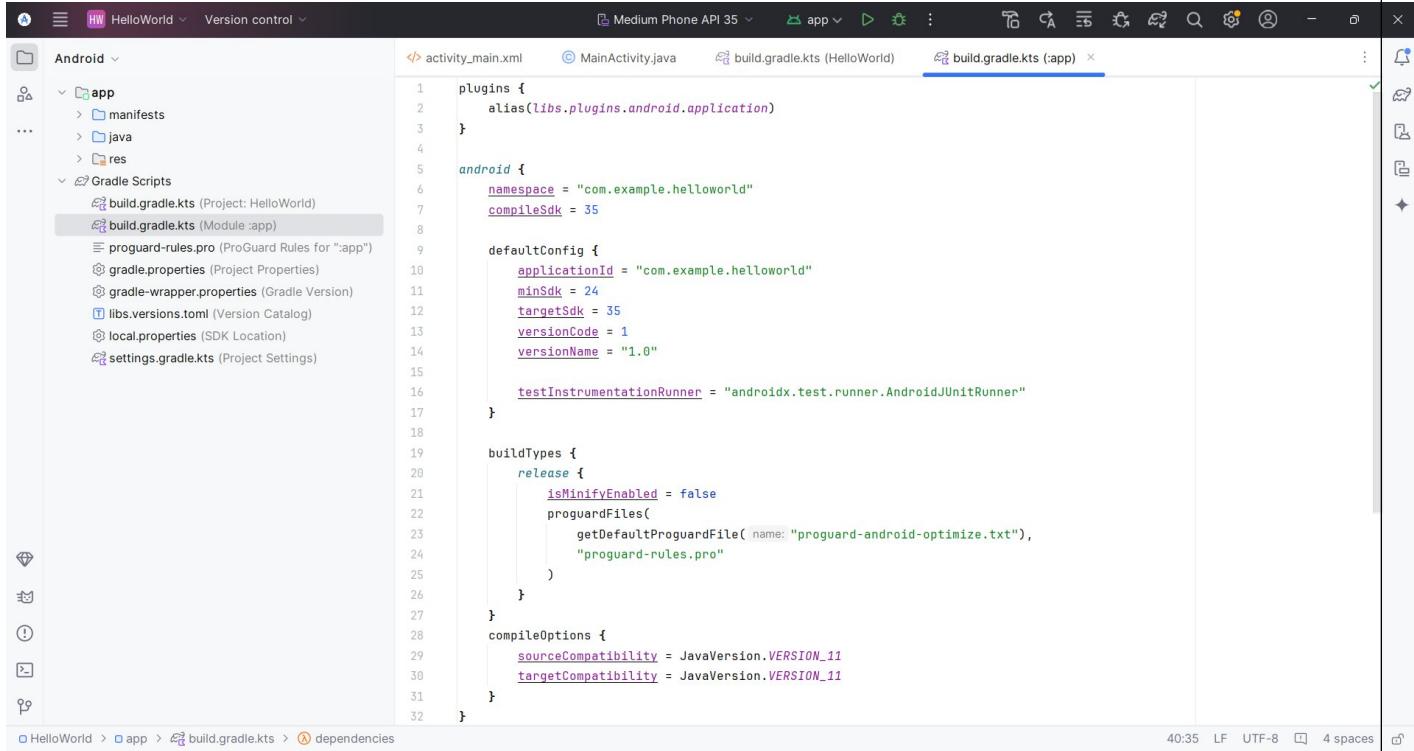
The status bar at the bottom indicates the file is 'build.gradle.kts' with '4 spaces' indentation.

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

Ngoài tệp **build.gradle** cấp dự án, mỗi module cũng có một tệp **build.gradle** riêng, cho phép bạn cấu hình các thiết lập build cho từng module cụ thể (trong ứng dụng HelloWorld chỉ có một module). Việc cấu hình các thiết lập build này cho phép bạn tùy chỉnh các tùy chọn đóng gói, chẳng hạn như thêm các loại build (build types) và biến thể sản phẩm (product flavors). Bạn cũng có thể ghi đè các thiết lập trong tệp **AndroidManifest.xml** hoặc tệp **build.gradle** cấp cao nhất.

Tệp này thường là tệp cần chỉnh sửa khi thay đổi các cấu hình cấp ứng dụng, chẳng hạn như khai báo các phụ thuộc (dependencies) trong phần **dependencies**. Bạn có thể khai báo một thư viện phụ thuộc bằng cách sử dụng một trong 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 từ tất cả các tệp “.jar” trong thư mục **libs**.

Dưới đây là nội dung tệp **build.gradle(Module:app)** cho ứng dụng HelloWorld:



The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the file tree for the 'HelloWorld' project, specifically the 'app' module. The 'build.gradle.kts' file is selected and open in the main editor area. The code in the editor is as follows:

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

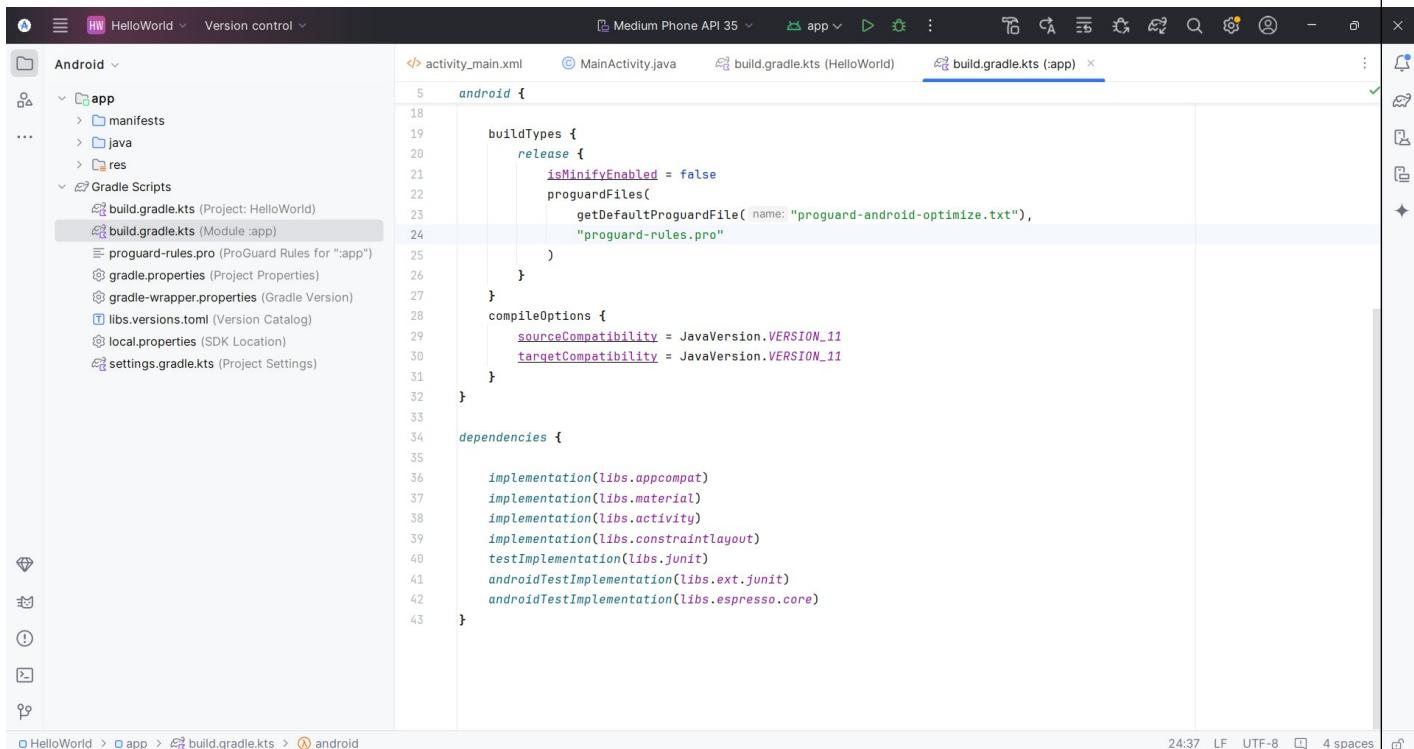
android {
    namespace = "com.example.helloworld"
    compileSdk = 35

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

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

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

The status bar at the bottom indicates the file path as 'HelloWorld > app > build.gradle.kts > dependencies', and the time as '40:35'. The code editor has syntax highlighting for Kotlin and Gradle.



This screenshot shows the same Android Studio environment as the previous one, but with changes made to the 'build.gradle.kts' file. The 'dependencies' block has been added and contains several library implementations. The code now looks like this:

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

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

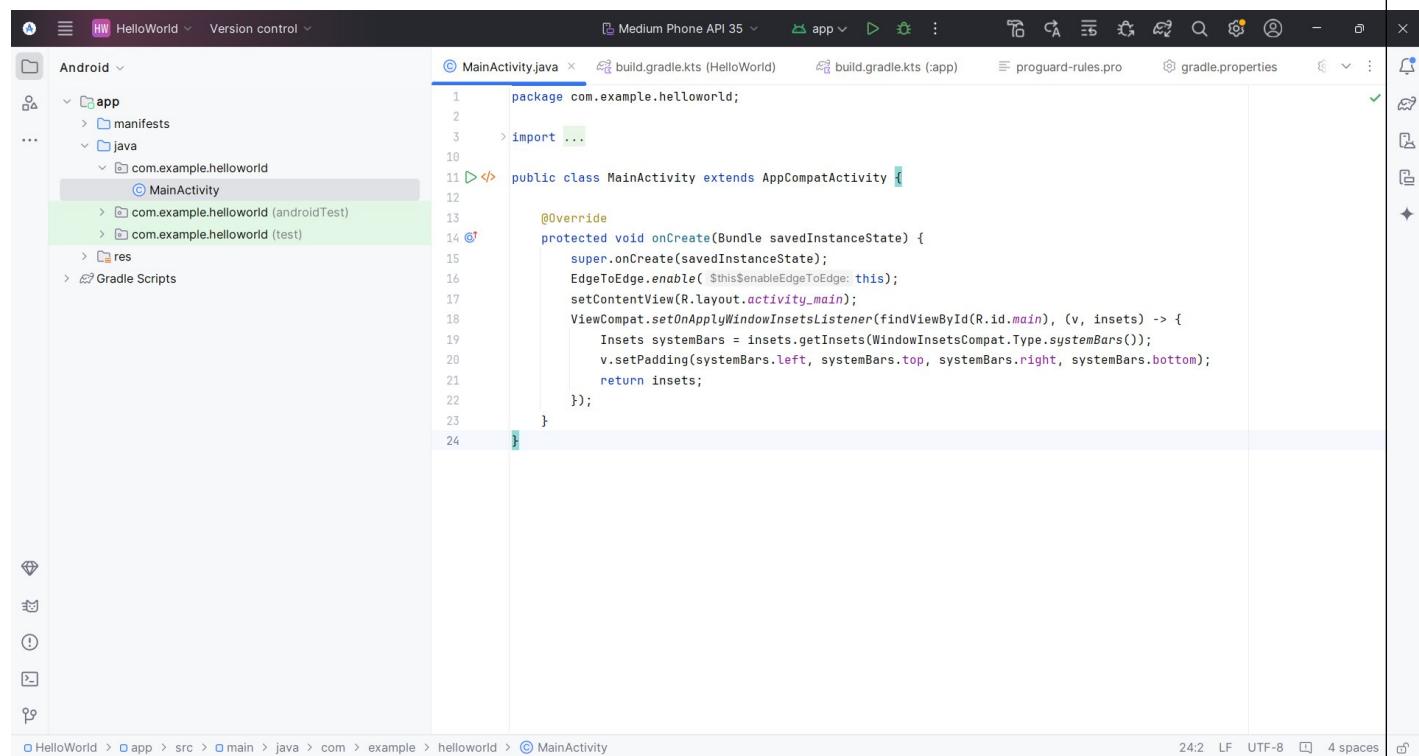
The status bar at the bottom indicates the file path as 'HelloWorld > app > build.gradle.kts > android', and the time as '24:37'. The code editor has syntax highlighting for Kotlin and Gradle.

4. Nhập vào hình tam giác để đóng Tập lệnh Gradle.

2.4 Khám phá thư mục app và res

Tất cả mã nguồn và tài nguyên của 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.java**. Nhấp đúp vào tệp để mở nó trong trình chỉnh sửa mã.



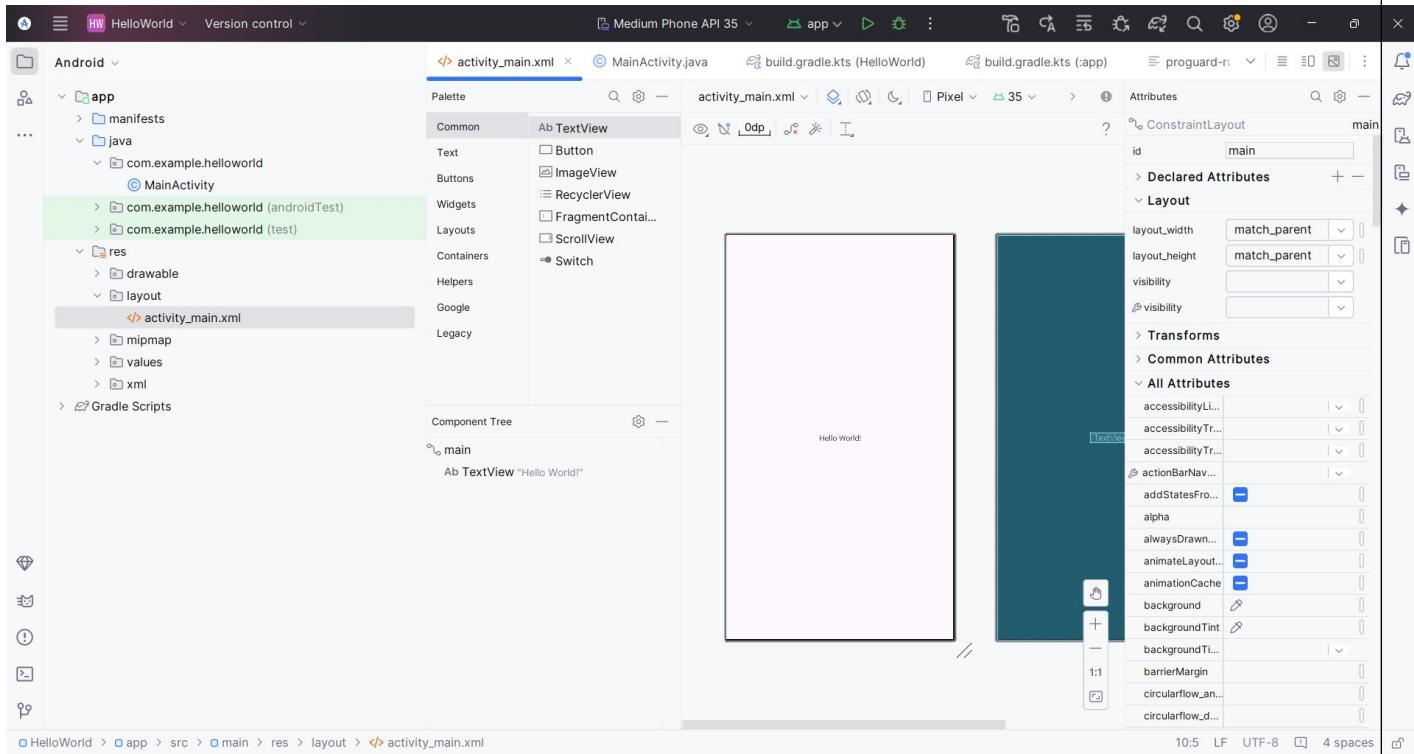
The screenshot shows the Android Studio interface. On the left, the Project Navigational Bar displays the project structure under 'Android'. The 'app' module is expanded, showing 'src' (containing 'main', 'java', 'res', and 'Gradle Scripts'), 'manifests', and 'java' (containing 'com.example.helloworld' which has 'MainActivity'). The 'MainActivity.java' file is selected and highlighted in green. On the right, the main code editor window shows the Java code for 'MainActivity'. The code defines a class 'MainActivity' that extends 'AppCompatActivity'. It overrides the 'onCreate' method to handle window insets for system bars. The code editor includes line numbers, syntax highlighting, and a status bar at the bottom indicating the file path (HelloWorld > app > src > main > java > com > example > helloworld > MainActivity), the current time (24:2), encoding (UTF-8), and font size (4 spaces).

```
1 package com.example.helloworld;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
11        setContentView(R.layout.activity_main);
12        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
13            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
14            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
15            return insets;
16        });
17    }
18
19 }
20
21
22 }
```

Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như được hiển thị trong hình trên. Thư mục **com.example.hello.helloworld** (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục còn lại được sử dụng cho mục đích kiểm thử và sẽ được mô tả trong một bài học khác. Đối với ứng dụng Hello World, chỉ có một gói duy nhất và nó chứa tệp **MainActivity.java**. Tên của **Activity** (màn hình) đầu tiên mà người dùng nhìn thấy, đồng thời cũng là nơi

khởi tạo các tài nguyên toàn ứng dụng, thường được gọi là **MainActivity** (phần mở rộng tệp được bỏ qua trong khung **Project > Android**).

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



Thư mục **res** chứa các tài nguyên như bộ cục (layouts), chuỗi (strings) và hình ảnh. Một **Activity** thường được liên kết với một bộ cục giao diện người dùng (UI) được định nghĩa trong một tệp XML. Tệp này thường được đặt tên theo **Activity** tương ứng.

2.5 Khám phá thư mục **manifests**

Thư mục **manifests** chứa các tệp cung cấp thông tin cần thiết về ứng dụng của bạn cho hệ thống Android, mà hệ thống cần phải có trước khi có thể chạy bất kỳ 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. Tất cả các thành phần của 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 truy cập tính năng. Để biết thêm thông tin giới thiệu, hãy xem **Tổng quan về App Manifest**.

TASK 3: Sử dụng thiết bị ảo (emulator)

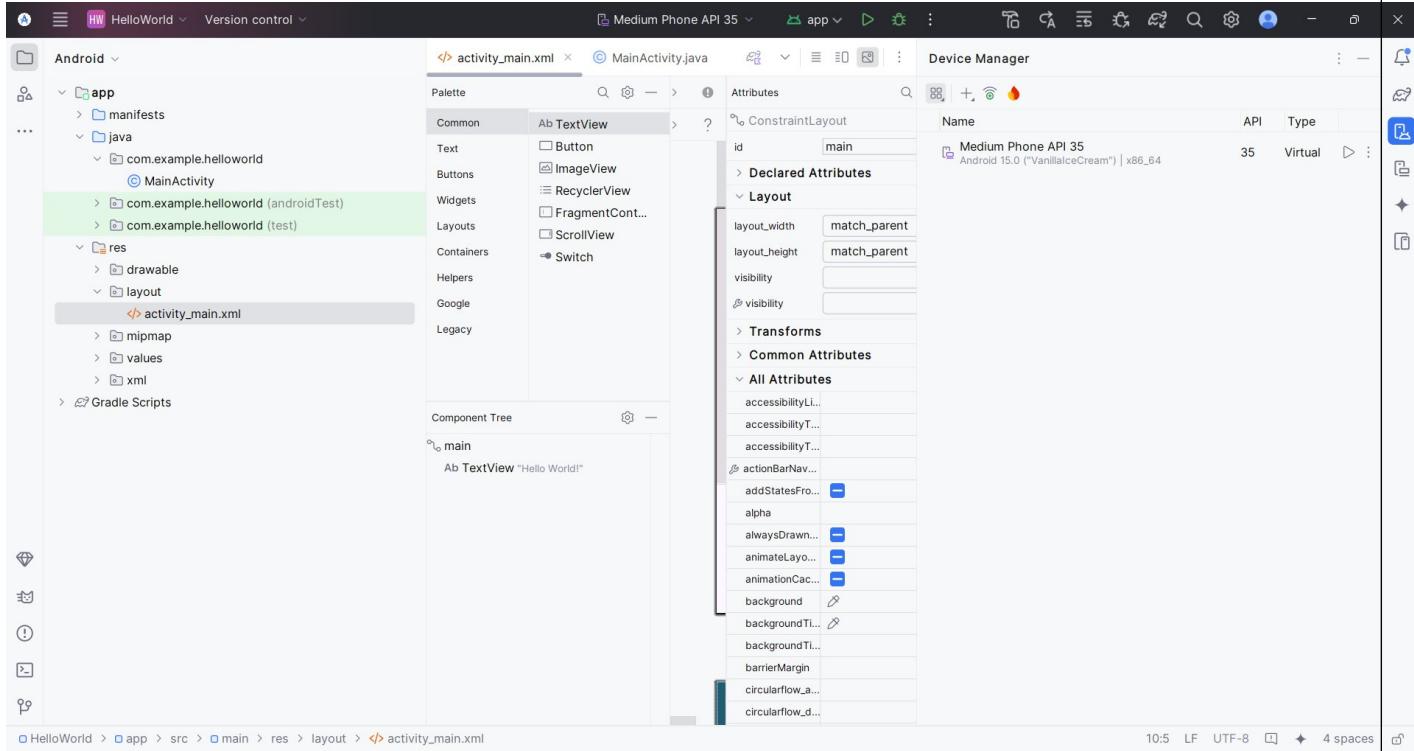
Trong nhiệm vụ này, bạn sẽ sử dụng trình quản lý Android Virtual Device (AVD) để tạo một thiết bị ảo (còn được gọi là emulator) mô phỏng cấu hình cho một loại thiết bị Android cụ thể, và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng Android Emulator có các yêu cầu bổ sung ngoài các yêu cầu hệ thống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

Bằng cách sử dụng trình quản lý AVD, bạn có thể định nghĩa các đặc điểm phần cứng của thiết bị, mức API, bộ nhớ, giao diện (skin) và các thuộc tính khác, sau đó lưu nó dưới dạng một thiết bị ảo. Với các thiết bị ảo, bạn có thể kiểm thử ứng dụng trên các cấu hình thiết bị khác nhau (chẳng hạn như máy tính bảng và điện thoại) với các mức API khác nhau mà không cần sử dụng thiết bị vật lý.

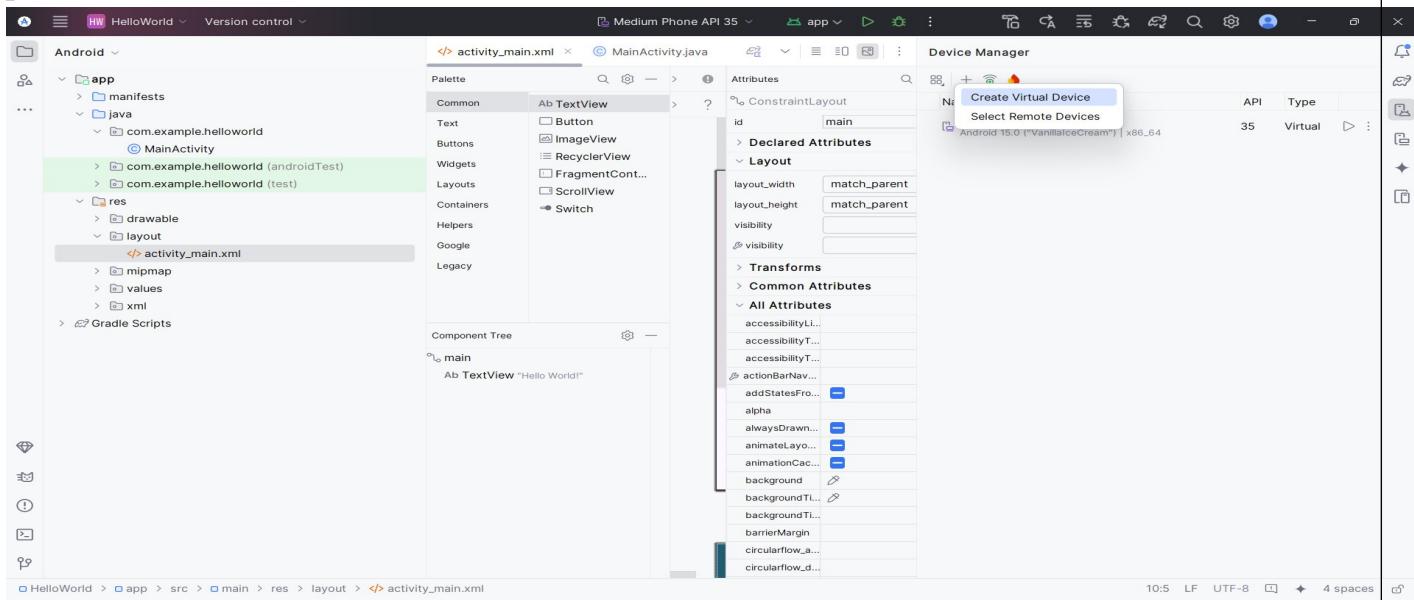
3.1 Tạo một thiết bị ảo Android (AVD)

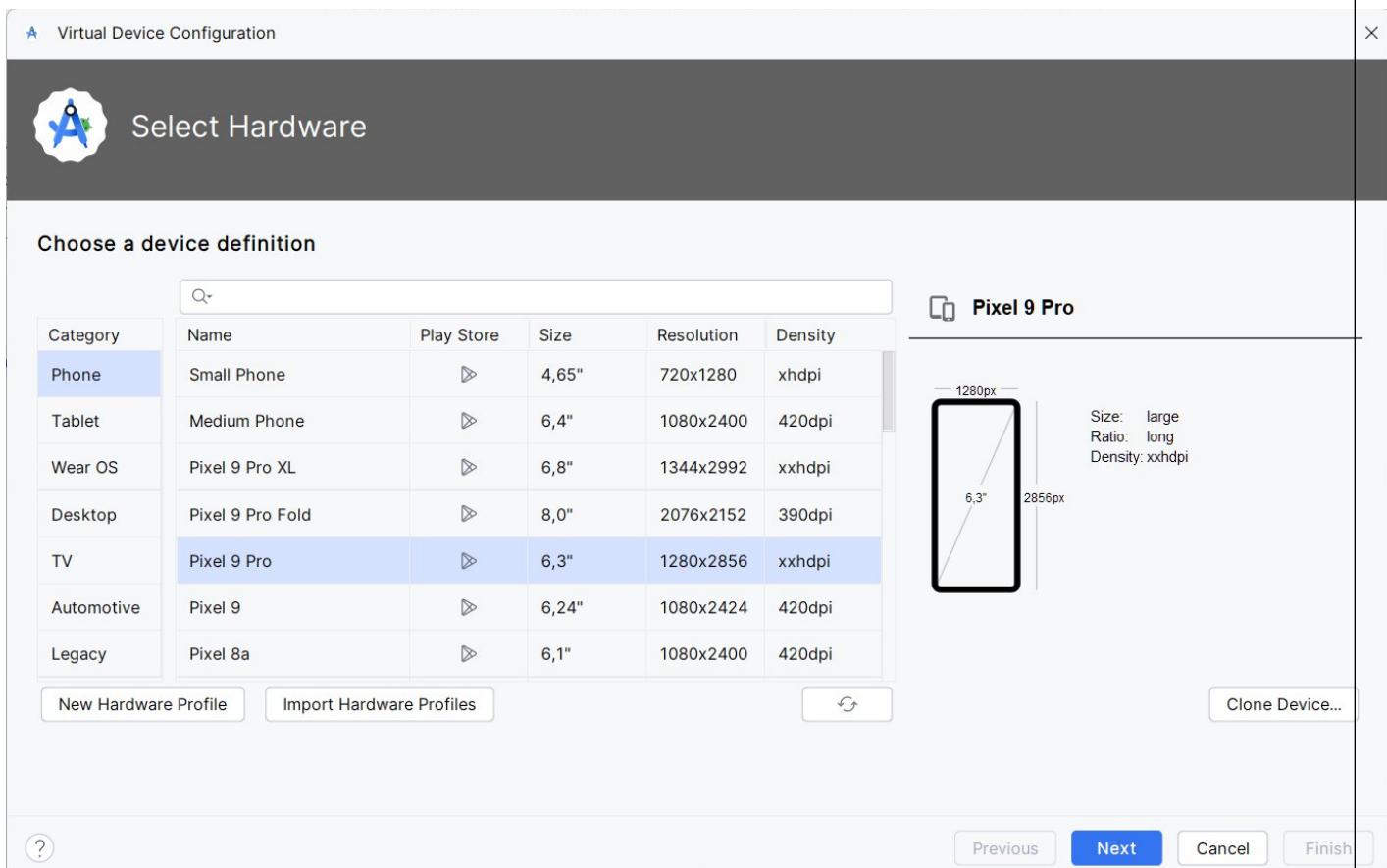
Để chạy một emulator trên máy tính của bạn, bạn cần tạo một cấu hình mô tả thiết bị ảo.

Bước 1 : Vào Device Manager phía thanh dọc bên phải màn hình



Nhập vào +Create Virtual Device. Cửa sổ Select Hardware sẽ xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng cung cấp một cột checkbox thuộc màn hình chéo (Size), độ phân giải màn hình tính bằng pixel.





Chọn một thiết bị như Nexus 5x hoặc Pixel 9 Pro và nhấp vào Tiếp theo. Màn hình
Ảnh hệ thống xuất hiện.

Nhấp vào tab Đề xuất nếu chưa chọn và chọn phiên bản hệ thống Android nào để
chạy trên thiết bị ảo (như Oreo).

Virtual Device Configuration

System Image

Select a system image

Document was last saved: Just now

Recommended x86 Images Other Images

Release Name	API	ABI	xABI	Target
Baklava ↴	Baklava	x86_64		Android API Baklava (Google Play)
Baklava ↴	Baklava	x86_64		Android API Baklava (16 KB Page)
VanillalceCream	35	x86_64	arm64-v8a	Android 15.0 (Google Play)
VanillalceCream ↴	35	x86_64		Android 15.0 (16 KB Page)
UpsideDownCake ↴	34	x86_64		Android 14.0 (Google Play)
Tiramisu ↴	33	x86_64		Android 13.0 (Google Play)
Sv2 ↴	32	x86_64		Android 12L (Google Play)
S ↴	31	x86_64		Android 12.0 (Google Play)
R ↴	30	x86		Android 11.0 (Google Play)

VanillalceCream

API Level: 35

Type: Google Play

Android: 15.0

Google Inc.

System Image: x86_64 (translated: arm64-v8a)

We recommend these Google Play images because this device is compatible with Google Play.

?

Previous Next Cancel Finish

Virtual Device Configuration

Android Virtual Device (AVD)

Verify Configuration

AVD name: Pixel 9 Pro API 35

Pixel 9 Pro 6.3 1280x2856 xxhdpi Change...

VanillalceCream Android 15.0 x86_64 Change...

Preferred ABI: Optimal

Startup orientation: Portrait Landscape

Show Advanced Settings

Default Orientation

Sets the initial orientation of the device. During AVD emulation you can also rotate the device screen.

?

Previous Next Cancel Finish

Có nhiều phiên bản khả dụng hơn so với những phiên bản được hiển thị trong tab Đè xuất. Hãy xem các tab x86 Hình ảnh và Hình ảnh khác để xem chúng.

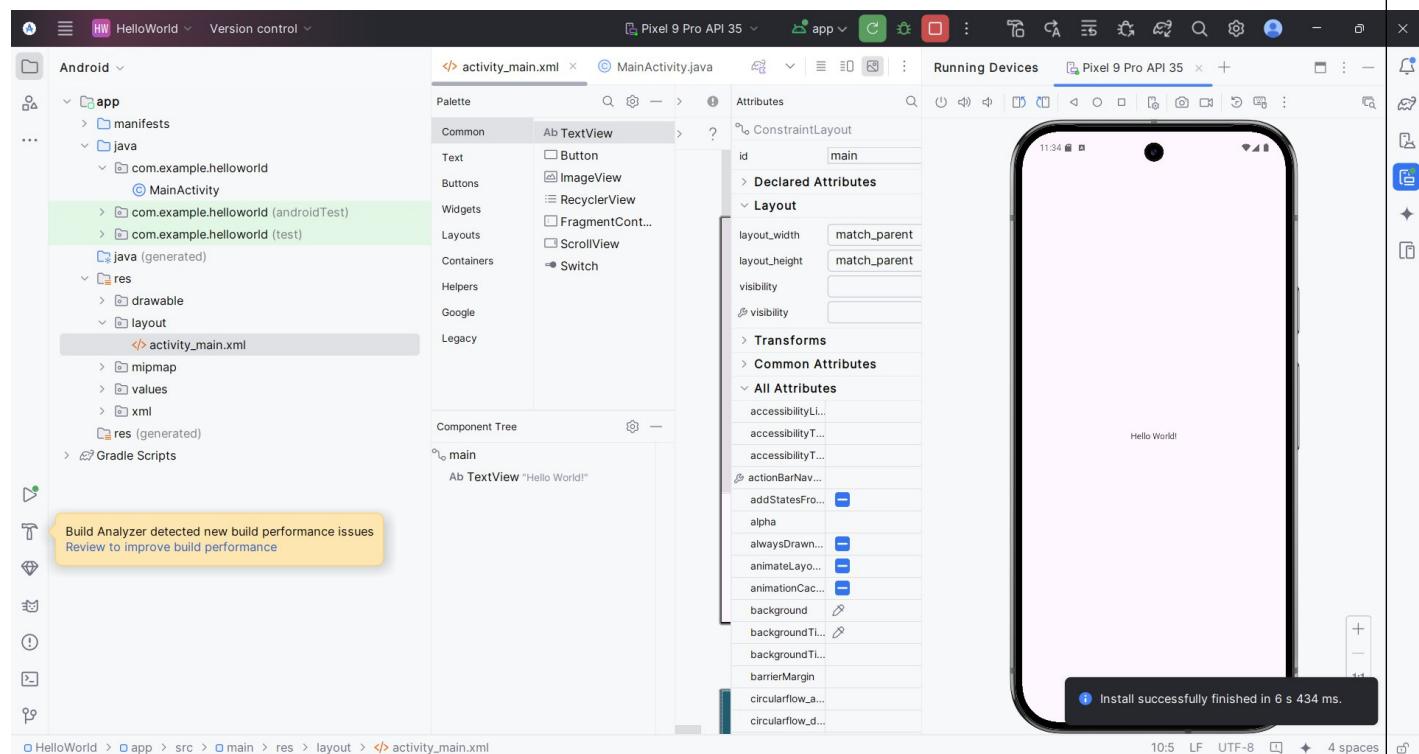
Nếu liên kết Tải xuống hiển thị bên cạnh hình ảnh hệ thống mà bạn muốn sử dụng, thì hình ảnh đó vẫn chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống và nhấp vào Hoàn tất khi hoàn tất.

5. Sau khi chọn hình ảnh hệ thống, hãy nhấp vào Tiếp theo. Cửa sổ Thiết bị ảo Android (AVD) sẽ xuất hiện.

Bạn cũng có thể thay đổi tên của AVD. Kiểm tra cấu hình của bạn và nhấp vào Hoàn tất.

3.2 Chạy ứng dụng trên thiết bị ảo

Sau khi tạo thành công và chạy với dự án hello world với máy ảo Pixel 9 Pro vừa tạo được giao diện như sau:



4.1 Bật chế độ USB Debugging

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

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

4.2 Chạy ứng dụng của bạn trên thiết bị

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

1. Kết nối thiết bị của bạn với máy phát triển bằng cáp USB.
 2. Nhấp vào nút **Run** trên thanh công cụ. Cửa sổ **Select Deployment Target** sẽ mở ra với danh sách các emulator và thiết bị được kết nối có sẵn.
 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 nhận diện được thiết bị của bạn, hãy thử các bước sau:

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

Nếu máy tính của bạn vẫn không tìm thấy thiết bị hoặc thông báo thiết bị "không được ủy quyền", hãy làm theo các bước sau:

1. Rút thiết bị ra.
2. Trên thiết bị, mở **Developer Options** trong ứng dụng **Settings**.
3. Nhấn vào **Revoke USB Debugging authorizations** (Thu hồi quyền ủy quyền USB Debugging).

4. Kết nối lại thiết bị với máy tính của bạn.

5. Khi được nhắc, hãy cấp quyền ủy quyền.

Bạn có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Hãy xem tài liệu **Using Hardware Devices**.

TASK 5: Thay đổi cấu hình Gradle của ứng dụng

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

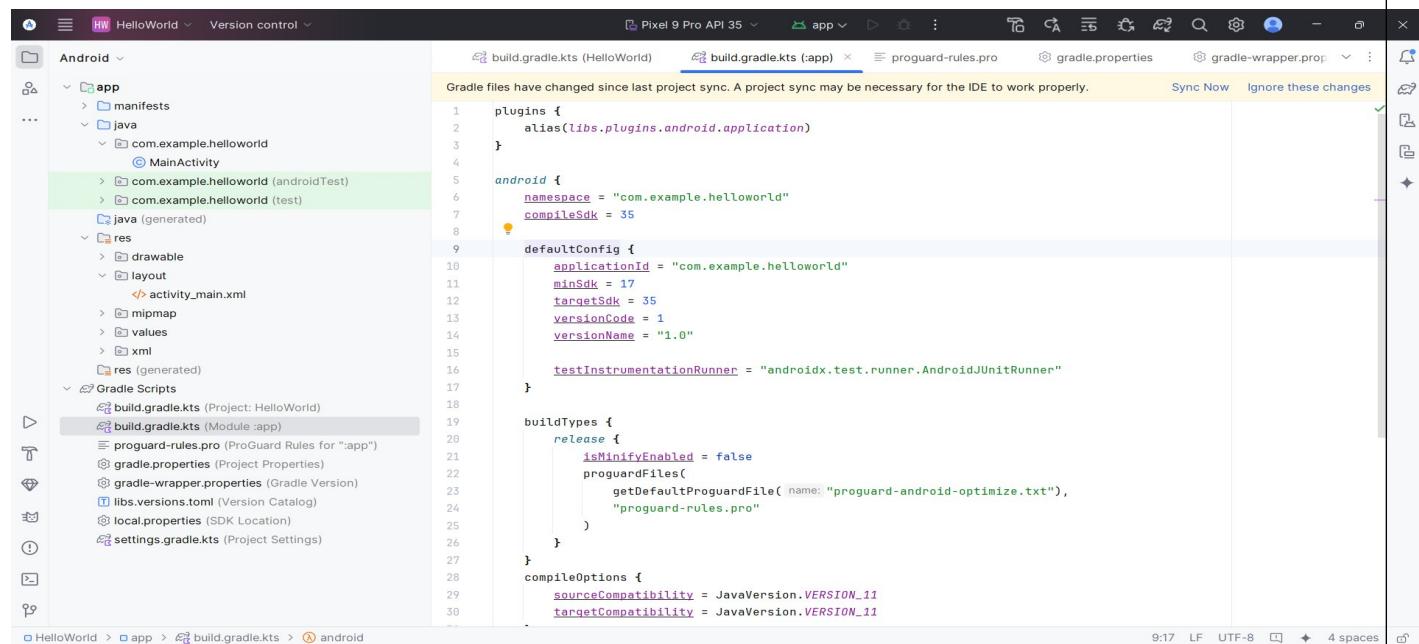
5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Làm theo các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu nó chưa được mở, và nhấp đúp vào tệp **build.gradle(Module:app)**.

Nội dung của tệp sẽ xuất hiện trong trình chỉnh sửa mã.

2. Trong khái **defaultConfig**, thay đổi giá trị của **minSdkVersion** thành **17** như hình dưới đây (ban đầu nó được đặt là **24**).



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

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

defaultConfig {
    applicationId = "com.example.helloworld"
    minSdk = 17
    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
}
```

Trình chỉnh sửa mã hiển thị thanh thông báo ở trên cùng với liên kết Đồng bộ hóa ngay.

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 các tệp dự án để nó có thể nhập các thay đổi cấu hình build và chạy một số kiểm tra để đảm bảo rằng cấu hình sẽ không gây ra lỗi build.

Để đồng bộ các tệp dự án, nhấp vào **Sync Now** trong thanh thông báo xuất hiện khi bạn thực hiện thay đổi (như trong hình 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.

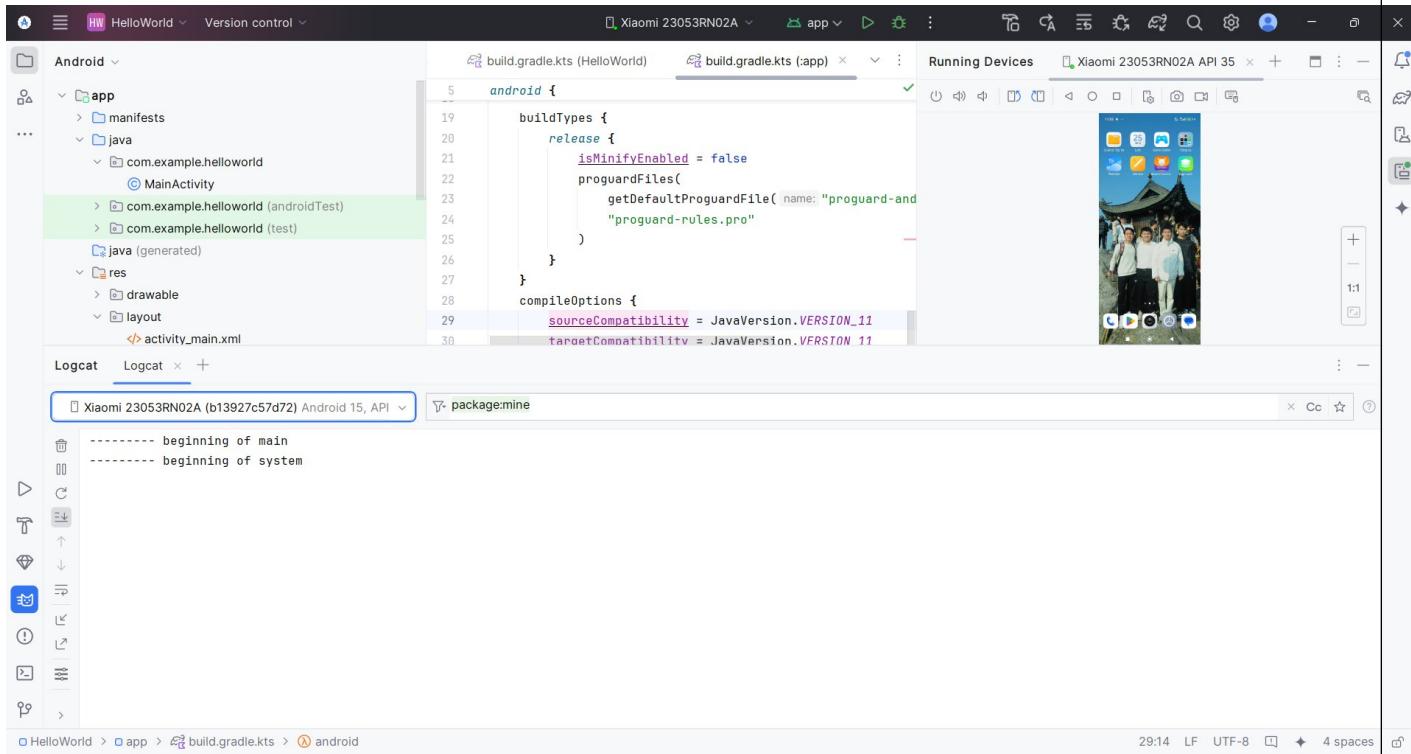
Để tìm hiểu sâu hơn về Gradle, hãy xem tài liệu **Build System Overview** và **Configuring Gradle Builds**.

Nhiệm vụ 6: Thêm các câu lệnh Log vào ứng dụng

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

6.1 Xem khung Logcat

Để xem khung **Logcat**, nhấp vào tab **Logcat** ở cuối cửa sổ Android Studio như hình dưới đây.



Trong hình trên:

1. Tab Logcat để mở và đóng ngắt Logcat, hiển thị thông tin về ứng dụng của bạn khi ứng dụng đang chạy. Nếu bạn thêm câu lệnh Log vào ứng dụng, thông báo Log sẽ xuất hiện ở đây.
2. Menu cấp độ Log được đặt thành Verbose (mặc định), hiển thị tất cả thông báo Log.

Các thiết lập khác bao gồm Debug, Error, Info và Warn.

6.2 Thêm câu lệnh log vào ứng dụng của bạn

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

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

Các phần của thông báo là:

- Nhật ký: Lớp Nhật ký để gửi thông báo nhật ký đến ngăn Logcat.

- d: Cài đặt mức Nhật ký gỡ lỗi để lọc hiển thị thông báo nhật ký trong ngăn Logcat. Các mức nhật ký khác là e cho Lỗi, w cho Cảnh báo và i cho Thông tin.

- "MainActivity": Đối số đầu tiên là một thẻ có thể được sử dụng để lọc thông báo trong ngăn

Logcat. Đây thường là tên của Hoạt động mà thông báo bắt nguồn. Tuy nhiên, bạn có thể biến điều này thành bất kỳ thứ gì hữu ích cho bạn để gỡ lỗi.

Theo quy ước, thẻ nhật ký được định nghĩa là hằng số cho Hoạt động

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

"Hello world": Đối số thứ hai là thông báo thực tế.

Thực hiện theo các bước sau:

1. Mở ứng dụng Hello World của bạn trong Android studio và mở MainActivity.
2. Để tự động thêm các mục nhập rõ ràng vào dự án của bạn (chẳng hạn như android.util.Log

cần thiết để sử dụng Log), hãy chọn File > Settings trong Windows hoặc Android Studio >

Preferences trong macOS.

3. Chọn Editor > General >Auto Import. Chọn tất cả các hộp kiểm và đặt Insert imports on

paste thành All .

4. Nhập vào Apply rồi nhập vào OK.

5. Trong phương thức onCreate() của MainActivity, hãy thêm câu lệnh sau:

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

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

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

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

6.Nếu ngăn Logcat chưa mở, hãy nhấp vào tab Logcat ở cuối Android Studio để mở nó.

7. Kiểm tra xem tên mục tiêu và tên gói của ứng dụng có đúng không.

8. Thay đổi mức Nhật ký trong ngăn Logcat thành Gõ lỗi (hoặc để nguyên là Chi tiết vì có quá ít thông báo nhật ký).

9. Chạy ứng dụng của bạn.

Thông báo sau sẽ xuất hiện trong ngăn Logcat:

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 được gọi là View — mọi thành phần trên màn hình đều là một View. Lớp View đại diện cho khối xây dựng cơ bản cho tất cả các thành phần UI và là lớp cơ sở cho các lớp cung cấp các thành phần UI tương tác như nút bấm, hộp kiểm và trường nhập văn bản. Các lớp con của View thường được sử dụng, sẽ được mô tả trong các bài học tiếp theo, bao gồm:

- TextView: Hiển thị văn bản.
- EditText: Cho phép người dùng nhập và chỉnh sửa văn bản.
- Button và các phần tử có thể nhập khác (như RadioButton, CheckBox, và Spinner): Cung cấp hành vi tương tác.

- ScrollView và RecyclerView: Hiển thị các mục có thể cuộn.
- ImageView: Hiển thị hình ảnh.
- ConstraintLayout và LinearLayout: Chứa các phần tử View khác và định vị chúng.

Mã Java hiển thị và điều khiển UI được chứa trong một lớp mở rộng từ **Activity**. Một **Activity** thường được liên kết với một bố cục (layout) của các **View** được định nghĩa trong một tệp XML (eXtended Markup Language). Tệp XML này thường được đặt tên theo **Activity** của nó và định nghĩa 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**, bao gồm một **TextView** với văn bản "Hello World".

Trong các ứng dụng phức tạp hơn, một **Activity** có thể thực hiện các hành động để phản hồi thao tác 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 các bài học khác.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên của mình — một ứng dụng cho phép người dùng tương tác. Bạn sẽ tạo một ứng dụng bằng mẫu **Empty Activity**. Bạn cũng sẽ học cách sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế bố cục và cách chỉnh sửa bố cục trong XML. Bạn cần phát triển các kỹ năng này để có thể hoàn thành các bài thực hành khác trong khóa học này.

Những gì bạn cần 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 sẽ học

- Cách tạo một ứng dụng có hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế bố cục.

- Cách chỉnh sửa bộ cục trong XML.
- Rất nhiều thuật ngữ mới. Hãy xem **Từ vựng và khái niệm** để biết các định nghĩa dễ hiểu.

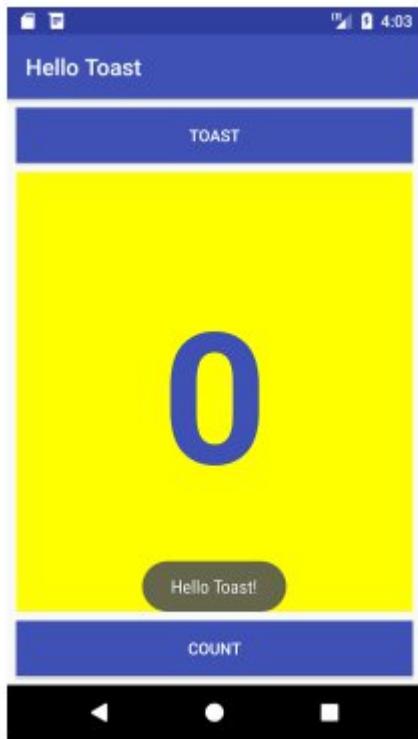
Những gì bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử **Button** và một **TextView** vào bộ cục.
- Thao tác với từng phần tử trong **ConstraintLayout** để ràng buộc chúng với lề và các phần tử khác.
- Thay đổi các thuộc tính của phần tử UI.
- Chỉnh sửa bộ 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 (click-handler methods) để hiển thị thông báo trên màn hình khi người dùng nhấp vào mỗi **Button**.

Tổng quan về ứng dụng

Ứng dụng **HelloToast** bao gồm hai phần tử **Button** và một **TextView**. Khi người dùng nhấp vào **Button** đầu tiên, một thông báo ngắn (Toast) sẽ xuất hiện trên màn hình. Nhấp vào **Button** thứ hai sẽ tăng một bộ đếm "click" hiển thị trong **TextView**, bắt đầu từ số 0.

Đây là giao diện của ứng dụng hoàn thiện:



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

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

1.1 Tạo dự án Android Studio

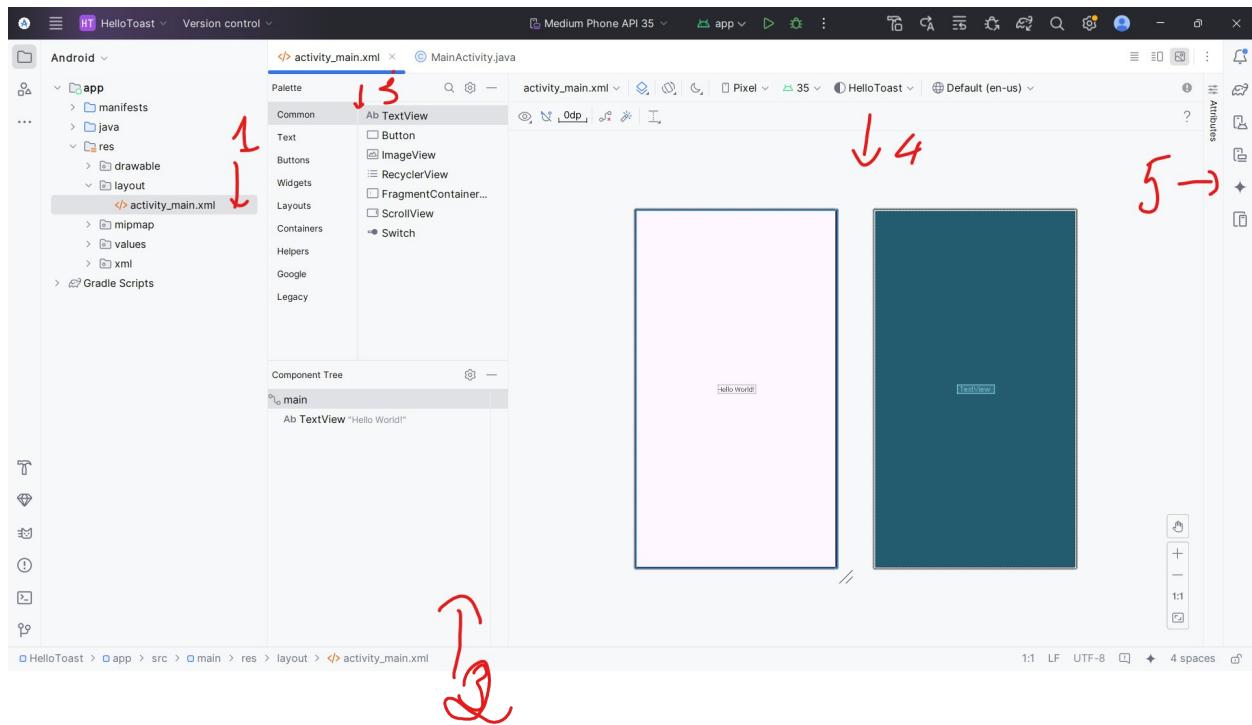
14.Khởi động Android Studio và tạo một dự án mới với các thông số sau:

Attribute	Value
Application Name	Hello Toast
Company Name	com.example.android (or your own domain)
Phone and Tablet Minimum SDK	API24
Template	Empty View Activity
Generate Layout file box	Selected
Backwards Compatibility box	Selected



15. Chọn Chạy > Chạy ứng dụng hoặc nhập vào biểu tượng trên thanh công cụ để xây dựng và thực thi ứng dụng trên trình mô phỏng hoặc thiết bị của bạn.

1.2 Khám phá trình chỉnh sửa bố cục



1. Trong thư mục **app > res > layout** trong bảng **Project > Android**, nhấp đúp vào tệp **activity_main.xml** để mở nó, nếu chưa được mở.

2. Bảng **Component tree** hiển thị cấu trúc phân cấp của các phần tử UI. Các phần tử view được tổ chức theo một cấu trúc cây, trong đó một phần tử con kế thừa các thuộc tính từ phần tử cha. Trong hình trên, **TextView** là phần tử con của **ConstraintLayout**. Bạn sẽ tìm hiểu về các phần tử này sau trong bài học.

3. Bảng **Palettes** hiển thị các phần tử giao diện người dùng (UI) mà bạn có thể sử dụng trong bố cục của ứng dụng.

4. Các bảng thiết kế và bản vẽ của trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình trên, bố cục chỉ hiển thị một phần tử: **TextView** với nội dung "Hello World".

5. Tab **Attributes** hiển thị bảng thuộc tính để thiết lập các thuộc tính cho một phần tử UI.

Mẹo: Xem phần **Building a UI with Layout Editor** để biết chi tiết về cách sử dụng trình chỉnh sửa bố cục, và **Meet Android Studio** để xem toàn bộ tài liệu của Android Studio.

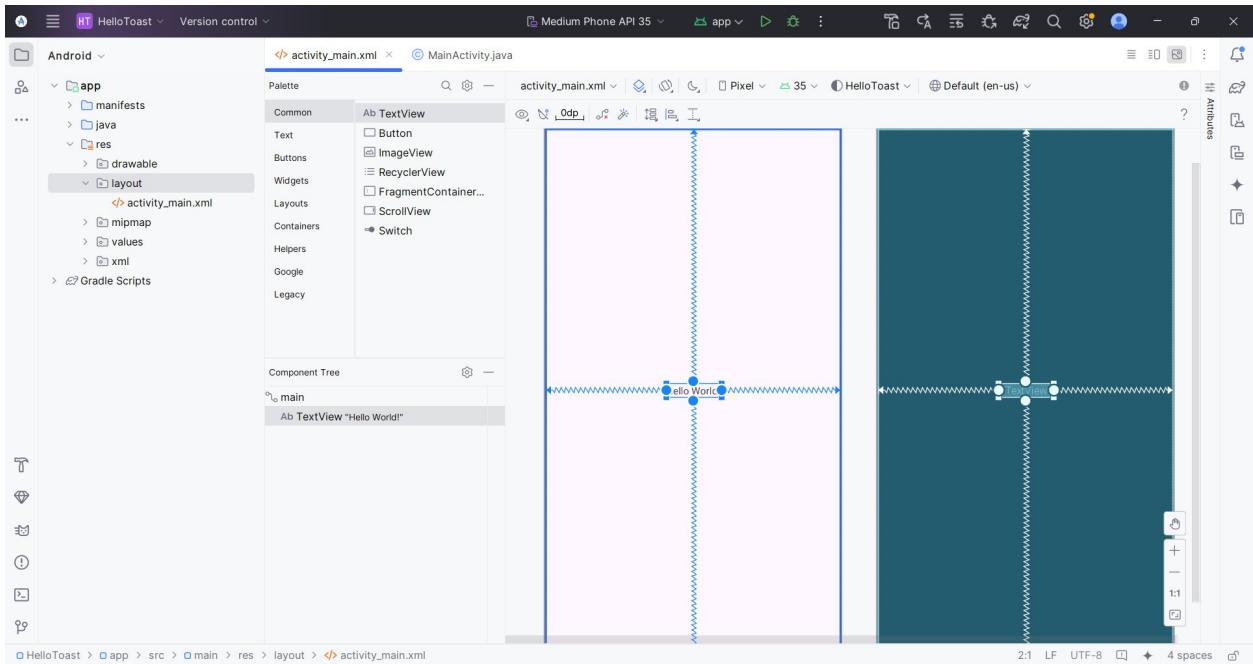
Nhiệm vụ 2: Thêm các phần tử View trong trình chỉnh sửa bố cục

Trong nhiệm vụ này, bạn sẽ tạo bố cục 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 thủ công, như sẽ được trình bày sau, hoặc tự động sử dụng công cụ **Autoconnect**.

2.1 Kiểm tra các ràng buộc phần tử

Thực hiện theo các bước sau:

1. Mở tệp **activity_main.xml** từ bảng **Project > Android** nếu nó chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào nó. Nếu không có chế độ bản vẽ, 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. Mặc định, công cụ này được kích hoạt. Đối với bước này, đảm bảo rằng công cụ không bị tắt.
3. Nhấp vào nút **zoom in** để phóng to các bảng thiết kế và bản vẽ để xem chi tiết.
4. Chọn **TextView** trong bảng **Component Tree**. **TextView** với nội dung "Hello World" sẽ được tô sáng trong các bảng thiết kế và bản vẽ, và các ràng buộc cho phần tử sẽ hiển thị.
5. Tham khảo hình ảnh động dưới đây cho bước này. Nhấp vào tay cầm hình tròn ở phía bên phải của **TextView** để xóa ràng buộc ngang liên kết view với phía bên phải của bố cục. **TextView** sẽ nhảy sang phía bên trái vì nó không còn bị ràng buộc vào bên phải nữa. Để thêm lại ràng buộc ngang, nhấp vào tay cầm đó và kéo một đường tới phía bên phải của bố cục.

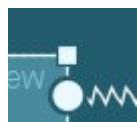


Trong các bảng thiết kế hoặc bản vẽ, các tay cầm sau xuất hiện trên phần tử **TextView**:

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



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

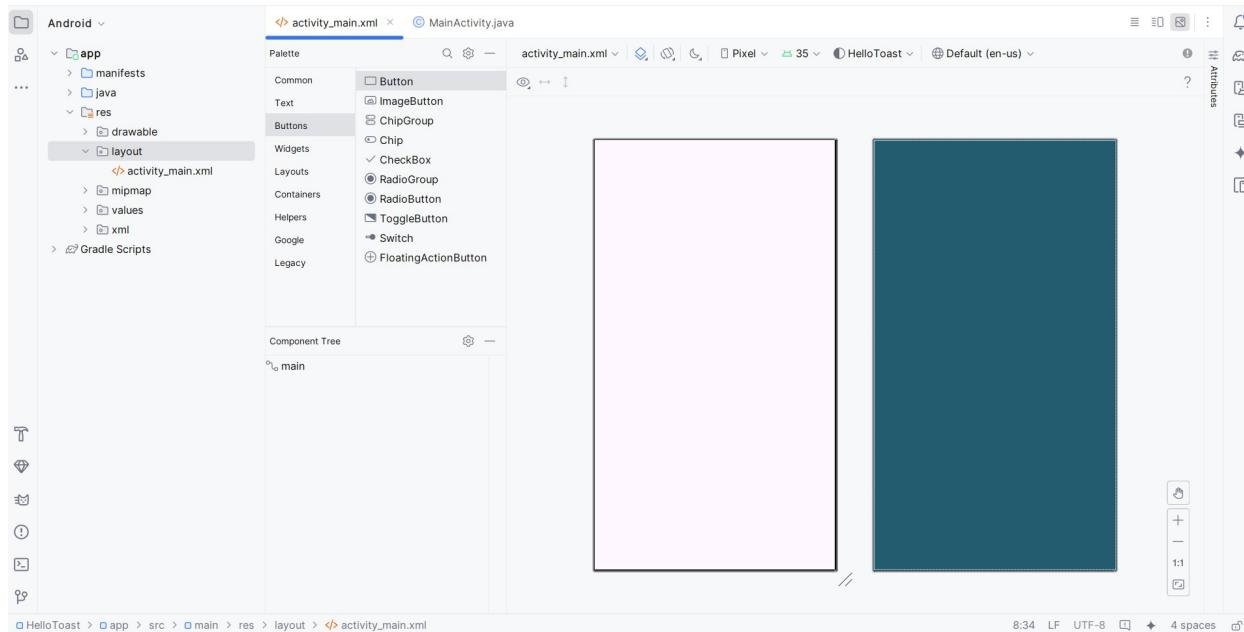


2.2 Thêm một Button vào bố cục

Khi được kích hoạt, công cụ **Autoconnect** sẽ tự động tạo hai hoặc nhiều ràng buộc cho một phần tử UI vào 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 theo 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, nhấn phím **Delete** hoặc chọn **Edit > Delete**. Jetzt Sie haben eine leere Layoutstruktur.
2. Kéo một **Button** từ bảng **Palette** vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả **Button** ở khu vực giữa trên cùng của bố cục, các ràng buộc có thể sẽ 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 đã minh họa trong hình động bên dưới.



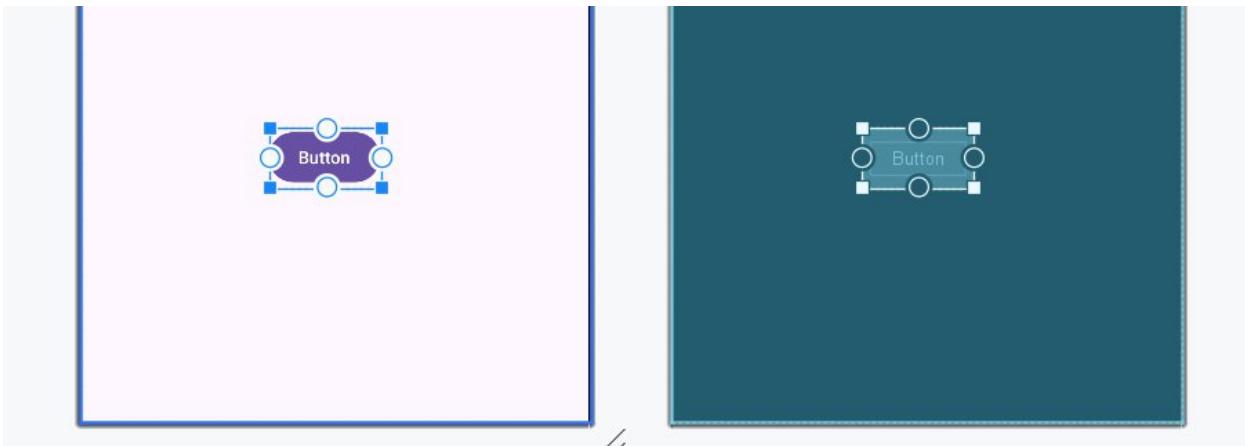
2.3 Thêm một Button thứ hai vào bố cục

1. Kéo một **Button** khác từ bảng **Palette** vào giữa bố cục như minh họa trong hình động dưới đây. **Autoconnect** có thể sẽ tự động cung cấp các ràng buộc ngang cho bạn (nếu không, bạn có thể tự kéo chúng).
2. Kéo một ràng buộc dọc đến phía dưới của bố cục (tham khảo hình dưới).

Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di chuyển con trỏ chuột qua nó để hiển thị nút **Clear Constraints**



Nhấp vào nút này để xóa tất cả các ràng buộc trên phần tử được chọn. Để xóa một ràng buộc cụ thể, nhấp vào tay cầm cụ thể đặt ra ràng buộc đó. Để xóa tất cả các ràng buộc trong toàn bộ bố cục, nhấp vào công cụ **Clear All Constraints** trên thanh công cụ. Công cụ này rất hữu ích nếu bạn muốn làm lại tất cả các ràng buộc trong bố cục của mình.

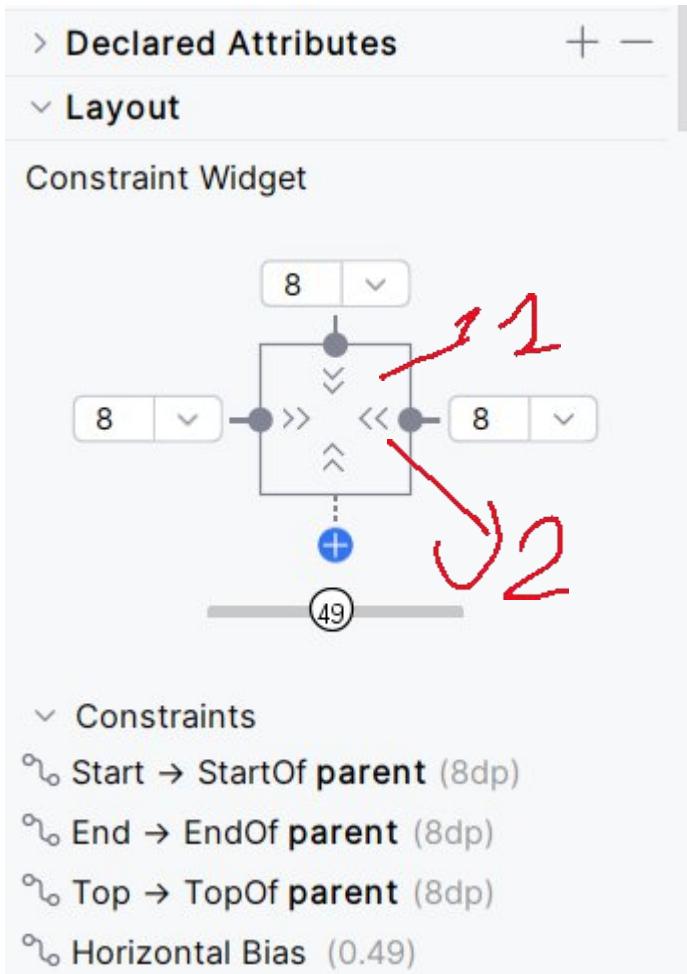


Nhiệm vụ 3: **Thay đổi các thuộc tính của phần tử giao diện người dùng (UI)**
Bảng **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Bạn có thể tìm các thuộc tính (còn gọi là thuộc tính properties) chung cho tất cả các view trong tài liệu của lớp **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**.

3.1 Thay đổi kích thước của Button

Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước ở bốn góc của một **View** để bạn có thể nhanh chóng thay đổi kích thước **View**. Bạn có thể kéo các tay cầm ở mỗi góc của **View** để thay đổi kích thước, nhưng làm như vậy sẽ cố định kích thước chiều rộng và chiều cao. Tránh việc cố định kích thước cho hầu hết các phần tử **View** vì các kích thước cố định không thể thích ứng với các nội dung và kích thước màn hình khác nhau.

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

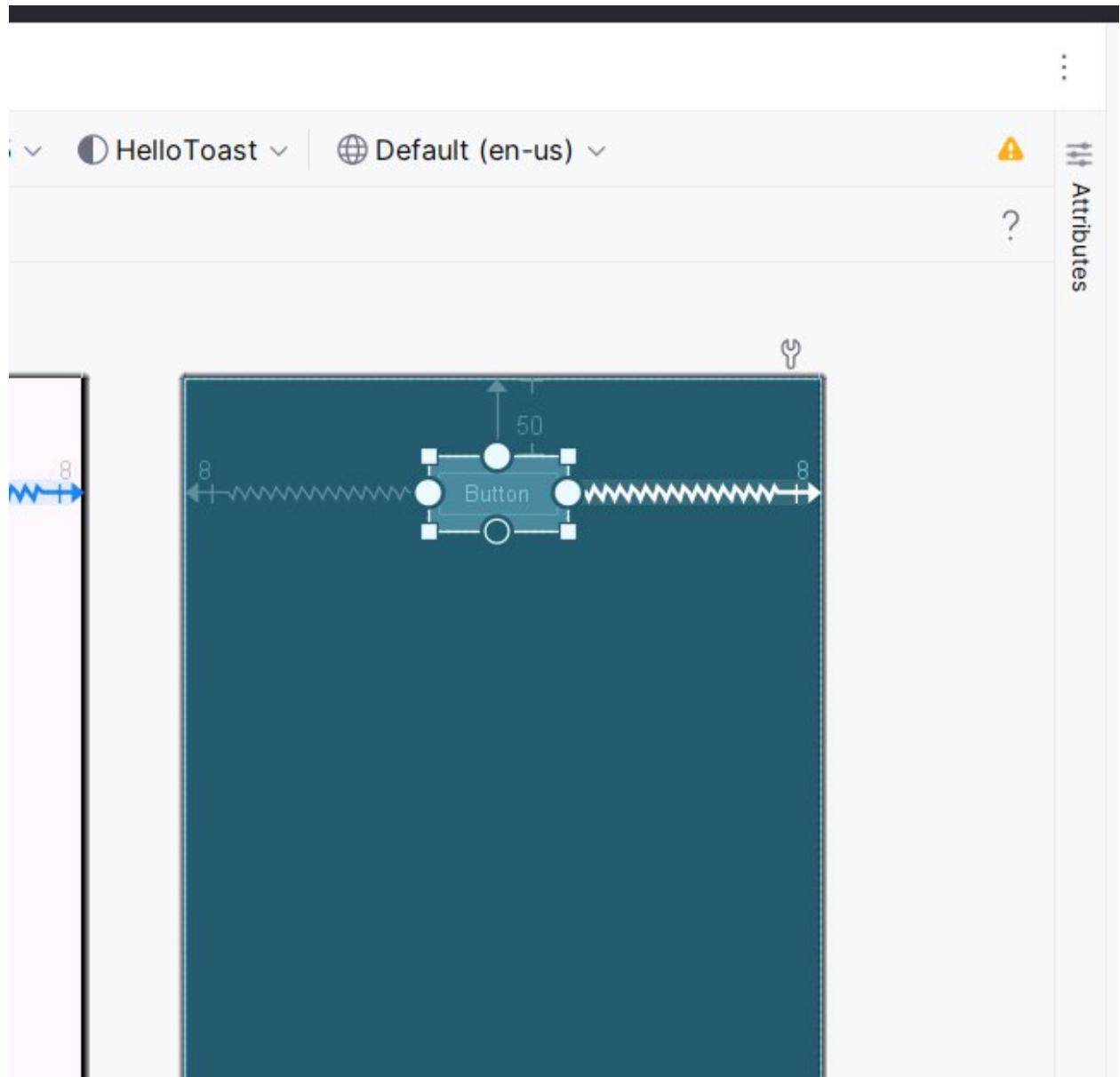


Trong hình trên:

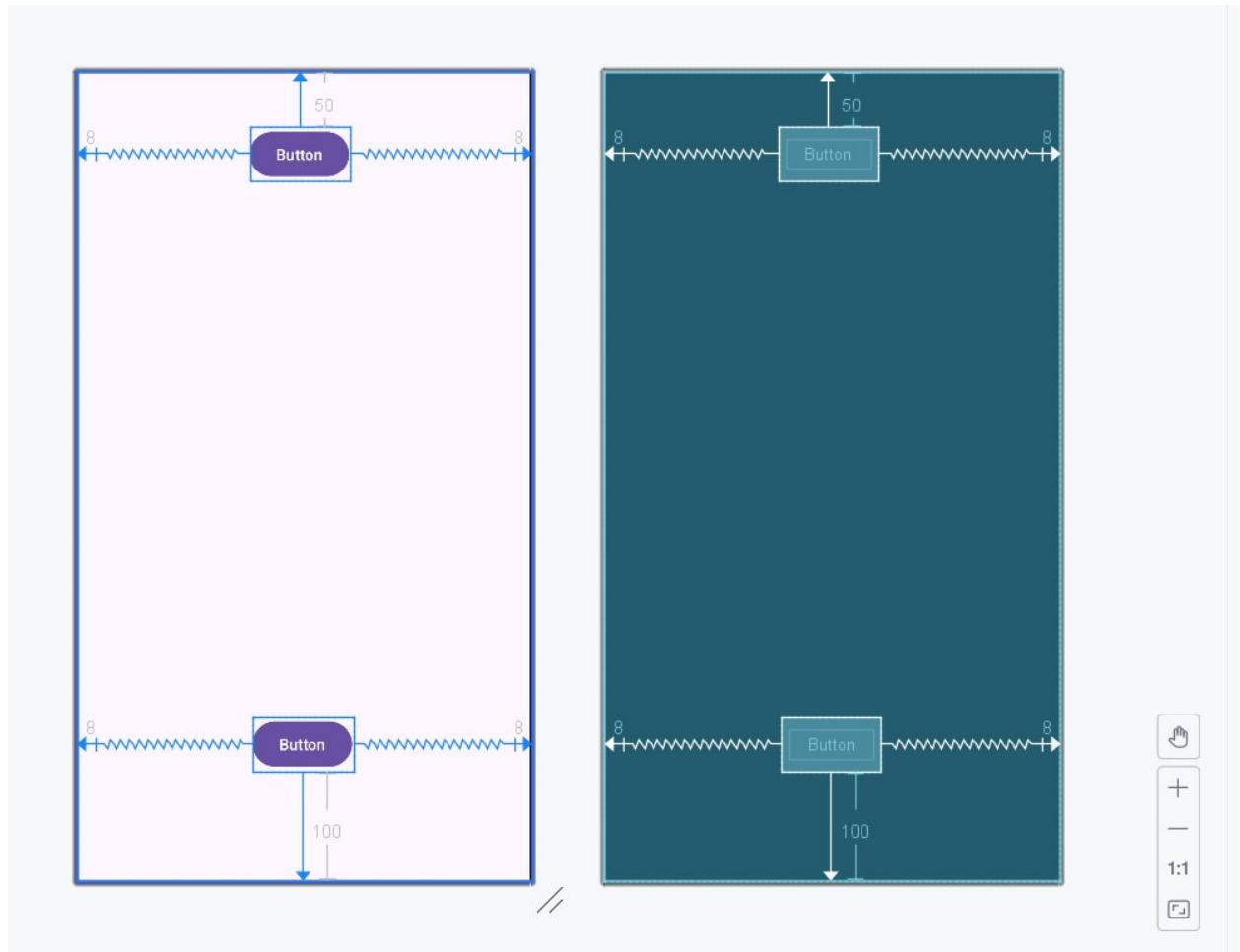
- Điều khiển chiều cao.** Điều khiển này chỉ định thuộc tính **layout_height** và xuất hiện ở hai đoạn trên và dưới của hình vuông. Các góc cho biết rằng điều khiển này được đặt thành **wrap_content**, nghĩa là **View** sẽ mở rộng theo chiều dọc tùy theo nhu cầu để phù hợp với nội dung của nó. Số "8" biểu thị một khoảng cách lề tiêu chuẩn được đặt là 8dp.
- Điều khiển chiều rộng.** Điều khiển này chỉ định thuộc tính **layout_width** và xuất hiện ở hai đoạn bên trái và bên phải của hình vuông. Các góc cho biết rằng điều khiển này được đặt thành **wrap_content**, nghĩa là **View** sẽ mở rộng theo chiều ngang tùy theo nhu cầu để phù hợp với nội dung của nó, với lề tối đa là 8dp.

Thực hiện theo các bước sau:

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

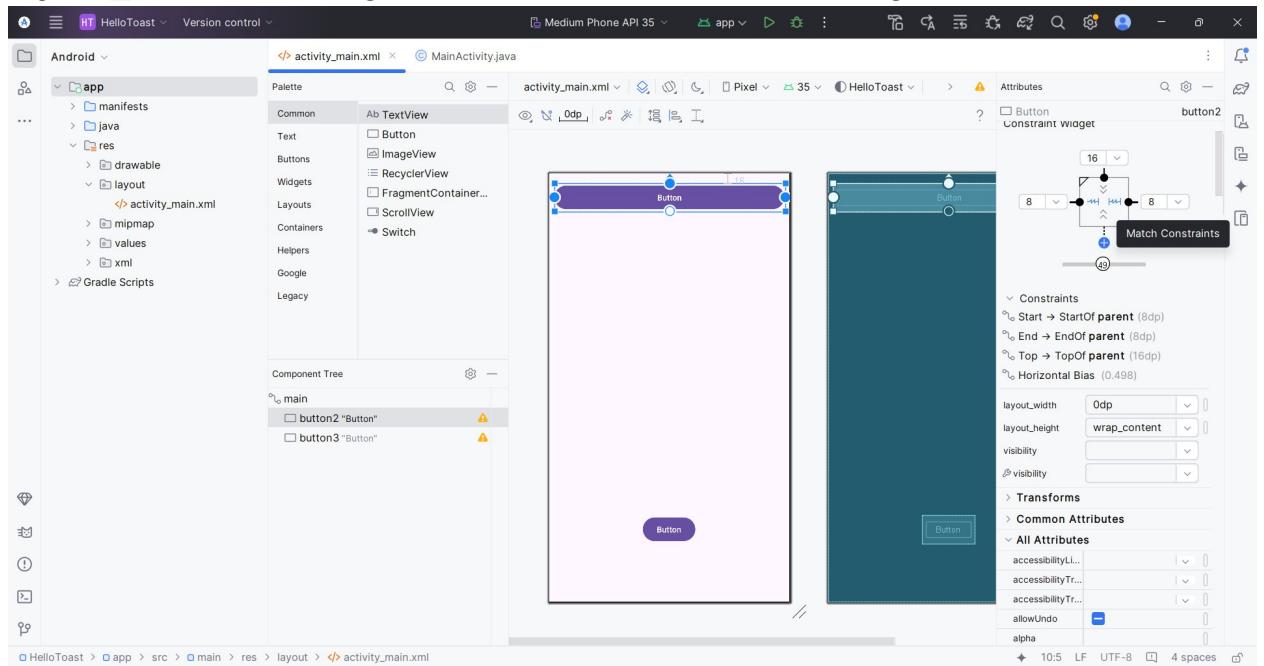


3. Nhấp vào điều khiển chiều rộng hai lần — lần nhấp đầu tiên thay đổi nó thành **Fixed** với các đường thẳng, và lần nhấp thứ hai thay đổi nó thành **Match Constraints** với các cuộn lò xo, như đã hiển thị trong hình động dưới đây.



Kết quả của việc thay đổi điều khiển chiều rộng, thuộc tính **layout_width** trong bảng **Attributes** hiển thị giá trị **match_constraint** và phần tử **Button** kéo dài theo chiều ngang để lấp đầy không gian giữa hai bên trái và phải của bố cục.

4. Chọn **Button** thứ hai và thực hiện các thay đổi tương tự cho thuộc tính **layout_width** như trong bước trước, như đã hiển thị trong hình dưới.



Như đã chỉ ra trong các bước trước, các thuộc tính **layout_width** và **layout_height** trong bảng **Attributes** thay đổi khi bạn thay đổi các điều khiển chiều cao và chiều rộng trong **inspector**. Các thuộc tính này có thể có một trong ba giá trị cho bố cục, là một **ConstraintLayout**:

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

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 **zero** vì không có kích thước được đặt. Cài đặt này giống như **match_constraint** — phần tử có thể mở rộng tối đa có thể để đáp ứng các ràng buộc và cài đặt lề.

3.2 Thay đổi các thuộc tính của Button

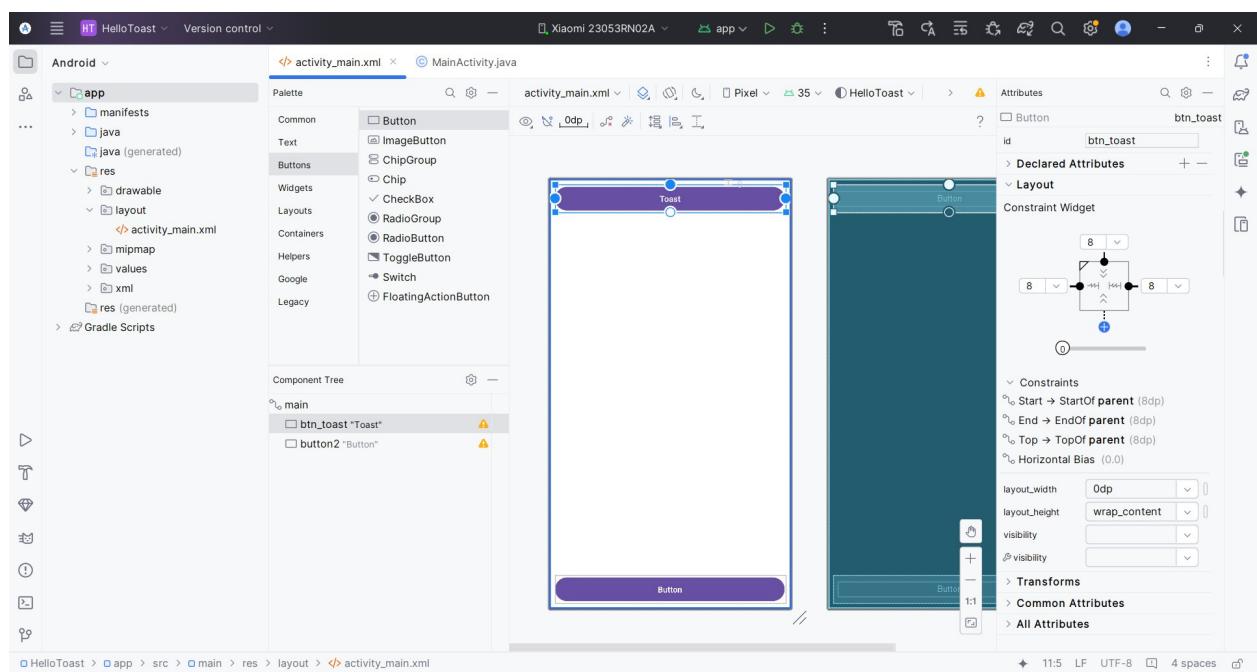
Để xác định duy nhất mỗi **View** trong một bố cục của **Activity**, mỗi **View** hoặc

lớp con của **View** (chẳng hạn như **Button**) cần có một ID duy nhất. Và để các nút **Button** có ý nghĩa sử dụng, chúng cần có văn bản. Các phần tử **View** cũng có thể có nền, đó có thể là màu sắc hoặc hình ảnh.

Bảng **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 các giá trị cho từng thuộc tính, chẳng hạn như **android:id**, **background**, **textColor**, và **text**.

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

1. Sau khi chọn **Button** đầu tiên, chỉnh sửa trường **ID** ở đầu bảng **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 bạn nhập **@c**, các tùy chọn sẽ xuất hiện để lựa 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 **text**, và các màu tương tự cho nền và văn bản như các bước trước.

colorPrimary là màu chính của chủ đề, là một trong các màu cơ bản của chủ đề được xác định trong tệp tài nguyên **colors.xml**. Nó được sử dụng cho thanh ứng dụng. Sử dụng các màu cơ bản cho các phần tử giao diện

người dùng khác giúp tạo ra một giao diện thống nhất. Bạn sẽ tìm hiểu thêm về chủ đề ứng dụng và **Material Design** trong bài học khác.

Nhiệm vụ 4: Thêm một TextView và đặt các thuộc tính của nó

Một trong những lợi ích của **ConstraintLayout** là khả năng căn chỉnh hoặc ràng buộc các phần tử liên quan đến các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một **TextView** vào giữa bố cục và ràng buộc nó theo chiều ngang với các lề và theo chiều dọc với hai nút **Button**. Sau đó, bạn sẽ thay đổi các thuộc tính cho **TextView** trong bảng **Attributes**.

4.1 Thêm một TextView và các ràng buộc

1. Như được chỉ ra trong hình động dưới đây, kéo một **TextView** từ bảng **Palette** đến phần trên của bố cục, và kéo một ràng buộc từ đỉnh của **TextView** đến tay cầm của nút **Toast Button**. Điều này sẽ ràng buộc **TextView** nằm bên dưới nút **Button**.

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) Bộ 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**