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

3.1) Trinfh gowx loi .....

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

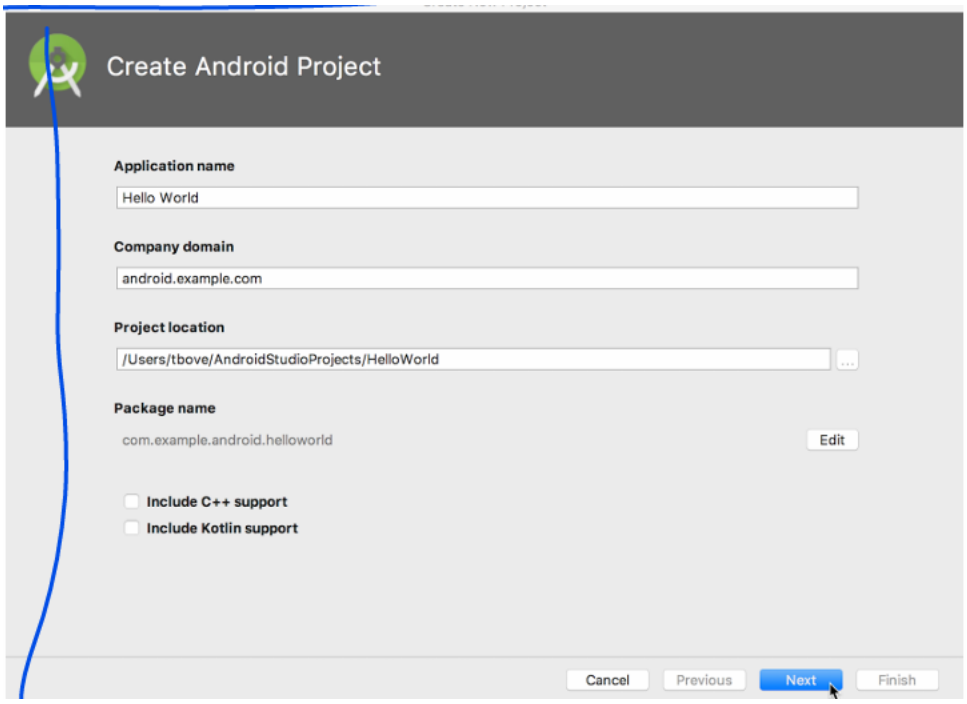
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

Sau khi bạn cài đặt thành công **Android Studio**, bạn sẽ tạo một **dự án mới** cho ứng dụng **Hello World** từ một **mẫu (template)** có sẵn. Ứng dụng đơn giản này sẽ hiển thị chuỗi ký tự **“Hello World”** trên màn hình của **thiết bị Android ảo (Android Virtual Device - AVD)** hoặc **thiết bị vật lý**.

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

ảnh

Bước 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp (IDE) hoàn chỉnh, bao gồm trình soạn thảo mã nâng cao và bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ cho phát triển, gỡ lỗi, kiểm thử và hiệu suất giúp việc phát triển ứng dụng nhanh chóng và dễ dàng hơn. Bạn có thể kiểm tra ứng dụng của mình với một loạt các

emulator đã được cấu hình sẵn hoặc trên thiết bị di động của riêng bạn, xây dựng ứng dụng sản phẩm và công bố trên Google Play Store.

Android Studio có sẵn cho các máy tính chạy Windows hoặc Linux, và cho các máy Mac chạy macOS. OpenJDK mới nhất (Java Development Kit) được đóng gói cùng với Android Studio. Để bắt đầu sử dụng Android Studio, trước tiên hãy kiểm tra yêu cầu hệ thống để đảm bảo rằng hệ thống của bạn đáp ứng được chúng. Việc cài đặt tương tự trên tất cả các nền tảng. Bất kỳ sự khác biệt nào sẽ được lưu ý dưới đây.

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

Khắc phục sự cố: Nếu bạn gặp phải vấn đề với việc cài đặt, hãy kiểm tra **release notes** của Android Studio hoặc nhờ sự trợ giúp từ giảng viên của bạn.

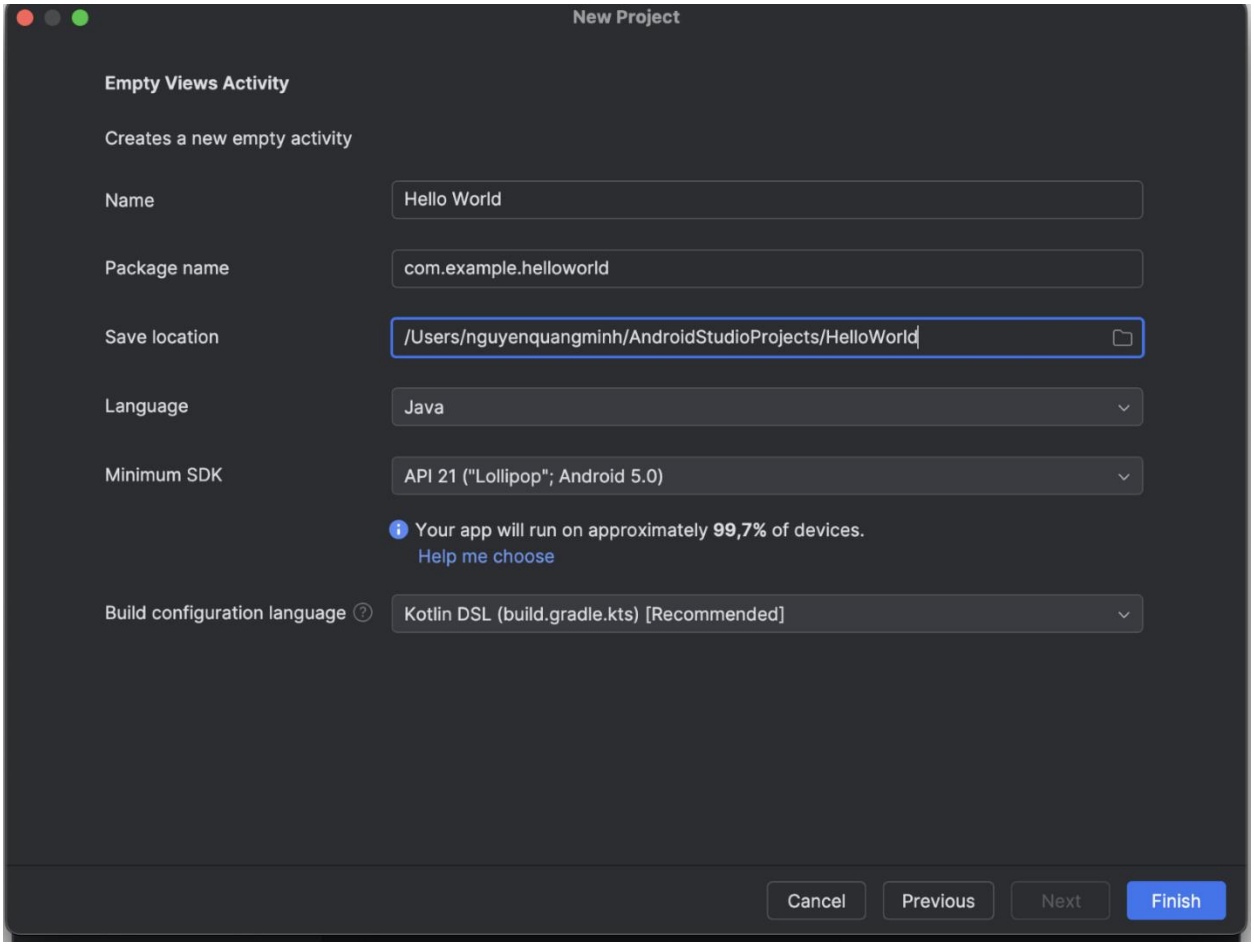
Bước 2: Tạo ứng dụng Hello World

Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị "Hello World" để xác nhận rằng Android Studio đã được cài đặt đúng và để học các bước cơ bản trong việc phát triển ứng dụng với Android Studio.

2.1 Tạo dự án ứng dụng

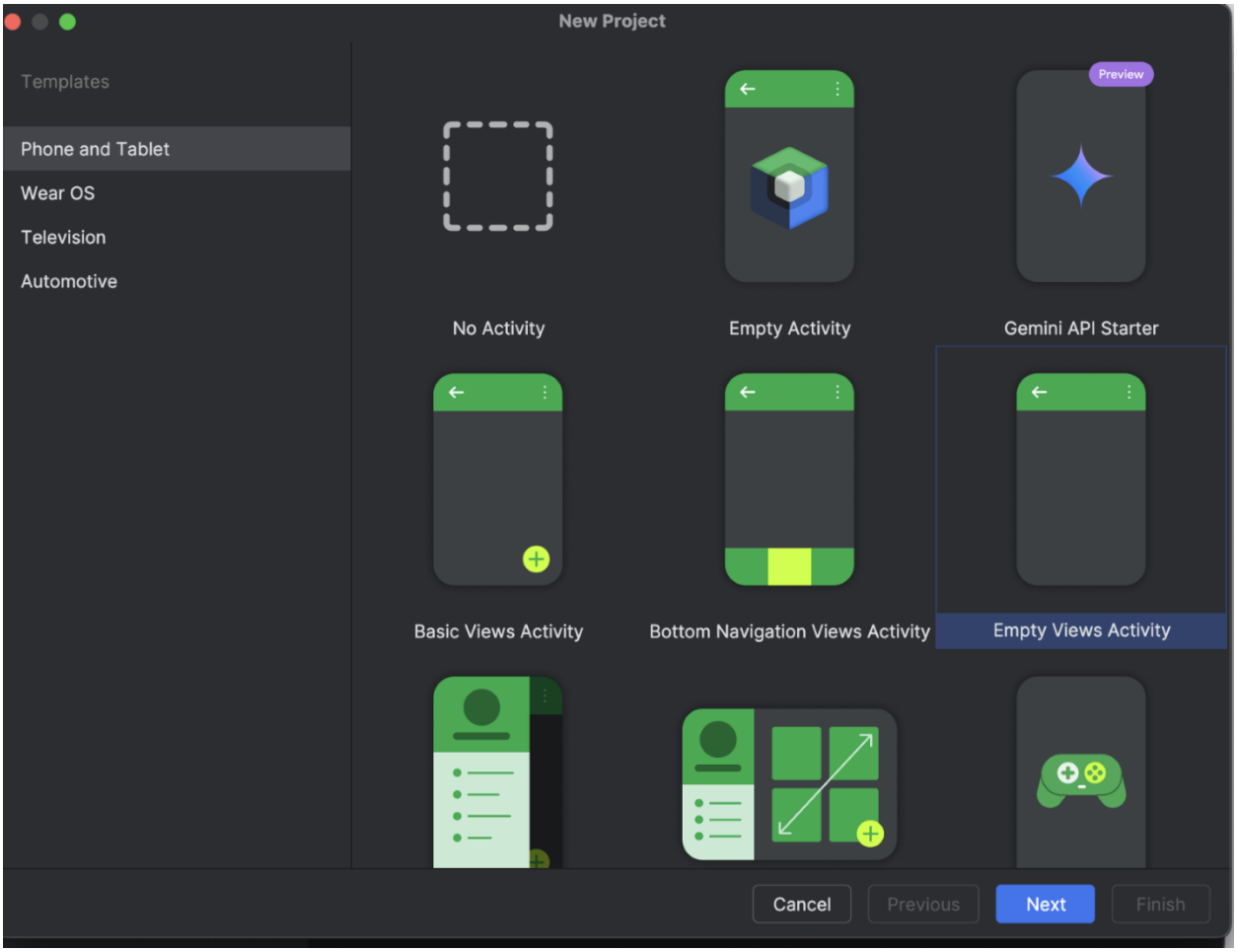
1. Mở Android Studio nếu chưa mở.

2. Trong cửa sổ **Welcome to Android Studio**, nhấp vào **Start a new Android Studio project**.
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** cho **Application name**.



4. Xác nhận rằng vị trí dự án mặc định là nơi bạn muốn lưu trữ ứng dụng Hello World và các dự án Android Studio khác, hoặc thay đổi nó thành thư mục bạn ưu tiên.
5. Chấp nhận **android.example.com** mặc định cho **Company Domain**, hoặc tạo một **company domain** độc đáo.
   Nếu bạn không có kế hoạch công bố ứng dụng của mình, bạn có thể chấp nhận mặc định. Lưu ý rằng việc thay đổi tên package của ứng dụng sau này sẽ là công việc bổ sung.

6. Bỏ chọn các tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấp **Next**.
7. Trên màn hình **Target Android Devices**, chọn **Phone and Tablet**. Đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm **Minimum SDK**; nếu không, hãy sử dụng menu pop-up để thiết lập nó



Đây là các cài đặt được sử dụng trong các ví dụ của các bài học trong khóa học này. Tính đến thời điểm viết, các cài đặt này giúp ứng dụng Hello World của bạn tương thích với 97% các thiết bị Android đang hoạt động trên Google Play Store.

8. Bỏ chọn tùy chọn **Include Instant App support** và tất cả các tùy chọn khác, sau đó nhấp **Next**. Nếu dự án của bạn yêu cầu các thành phần bổ sung cho **SDK** mục tiêu đã chọn, **Android Studio** sẽ tự động cài đặt chúng.

9. Cửa sổ **Add an Activity** xuất hiện. **Activity** là một hành động duy nhất, tập trung mà người dùng có thể thực hiện. Nó là một thành phần quan trọng của bất kỳ ứng dụng **Android** nào. Một **Activity** thường có một **layout** liên kết với nó, xác định cách các phần tử giao diện người dùng (**UI elements**) hiển thị trên màn hình. **Android Studio** cung cấp các mẫu **Activity** để giúp bạn bắt đầu. Đối với dự án **Hello World**, chọn **Empty Activity**, sau đó nhấp **Next**.

ảnh

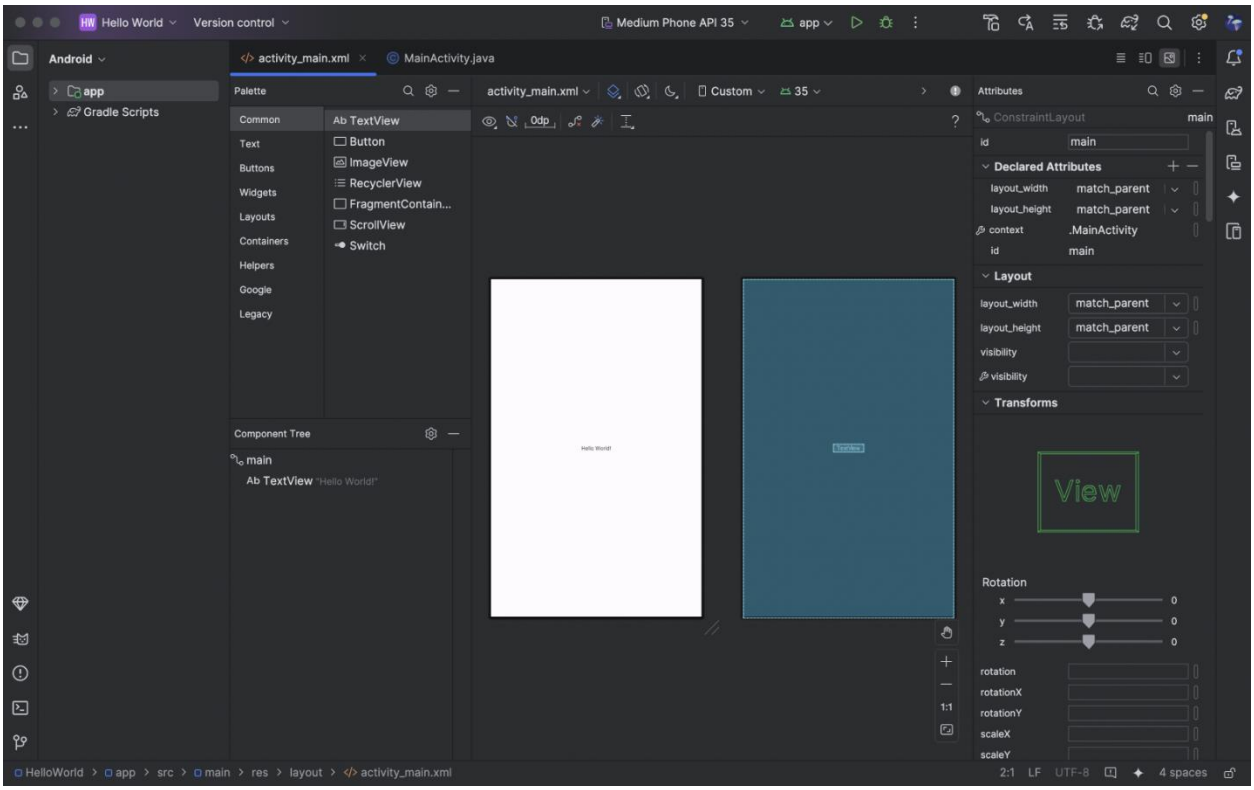
10. Cửa sổ **Configure Activity** xuất hiện (giao diện sẽ khác nhau tùy thuộc vào mẫu bạn đã chọn ở bước trước). Theo mặc định, **Activity** trống được cung cấp bởi mẫu sẽ được đặt tên là **MainActivity**. Bạn có thể thay đổi tên này nếu muốn, nhưng bài học này sẽ sử dụng **MainActivity**.
11. Đảm bảo rằng tùy chọn **Generate Layout file** được chọn. Tên **layout** mặc định là **activity_main**. Bạn có thể thay đổi tên này nếu muốn, nhưng bài học này sẽ sử dụng **activity_main**.
12. Đảm bảo rằng tùy chọn **Backwards Compatibility (App Compat)** được chọn. Điều này đảm bảo rằng ứng dụng của bạn sẽ tương thích ngược với các phiên bản **Android** trước đó.
13. Nhấp **Finish**.

Android Studio sẽ tạo một thư mục cho các dự án của bạn và xây dựng dự án bằng **Gradle** (quá trình này có thể mất vài phút).

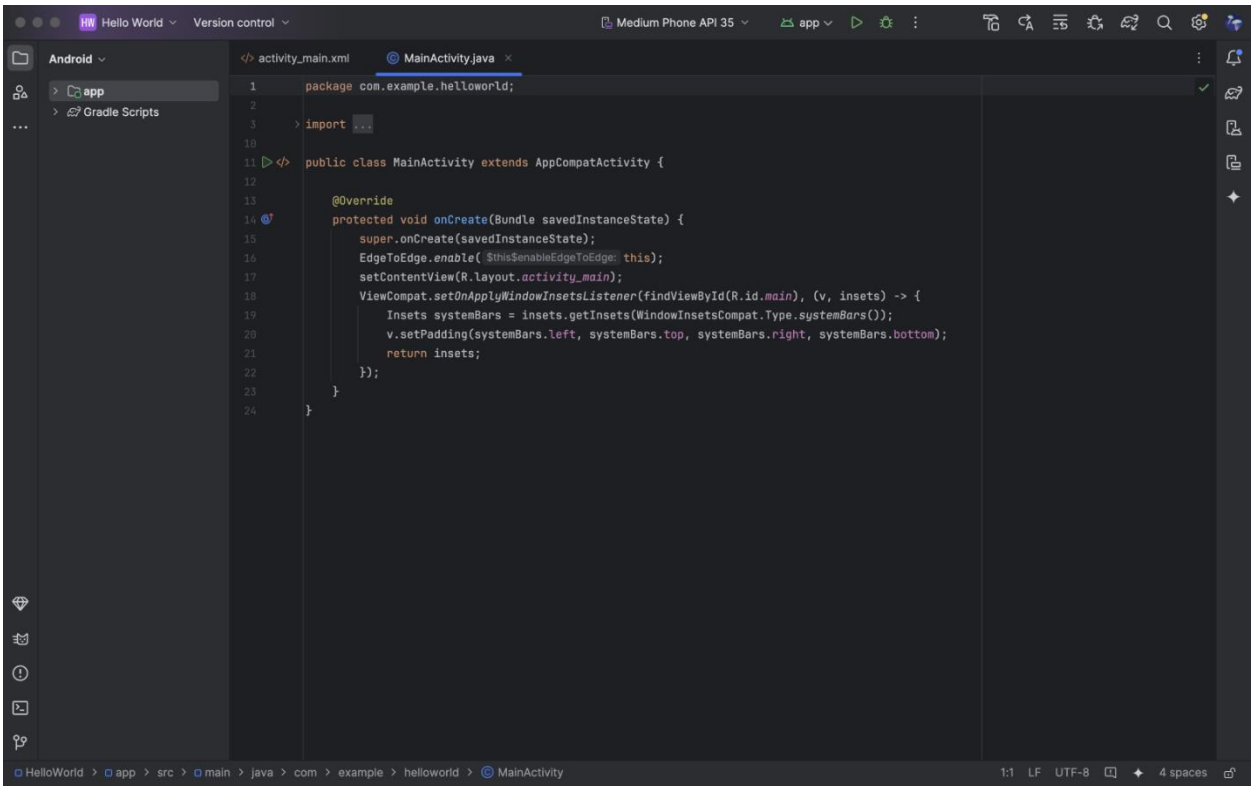
Mẹo: Tham khảo trang **Configure your build developer** để biết thêm thông tin chi tiết.
Bạn cũng có thể thấy thông báo **"Tip of the day"** với các phím tắt và mẹo hữu ích khác. Nhấp **Close** để đóng thông báo.

Giao diện chỉnh sửa của Android Studio sẽ xuất hiện. Thực hiện các bước sau:

1. Nhấp vào tab **activity_main.xml** để xem trình chỉnh sửa **layout**.
2. Nhấp vào tab **Design** trong trình chỉnh sửa **layout**, nếu chưa được chọn, để hiển thị bản dựng đồ họa của **layout** như hình dưới đây.



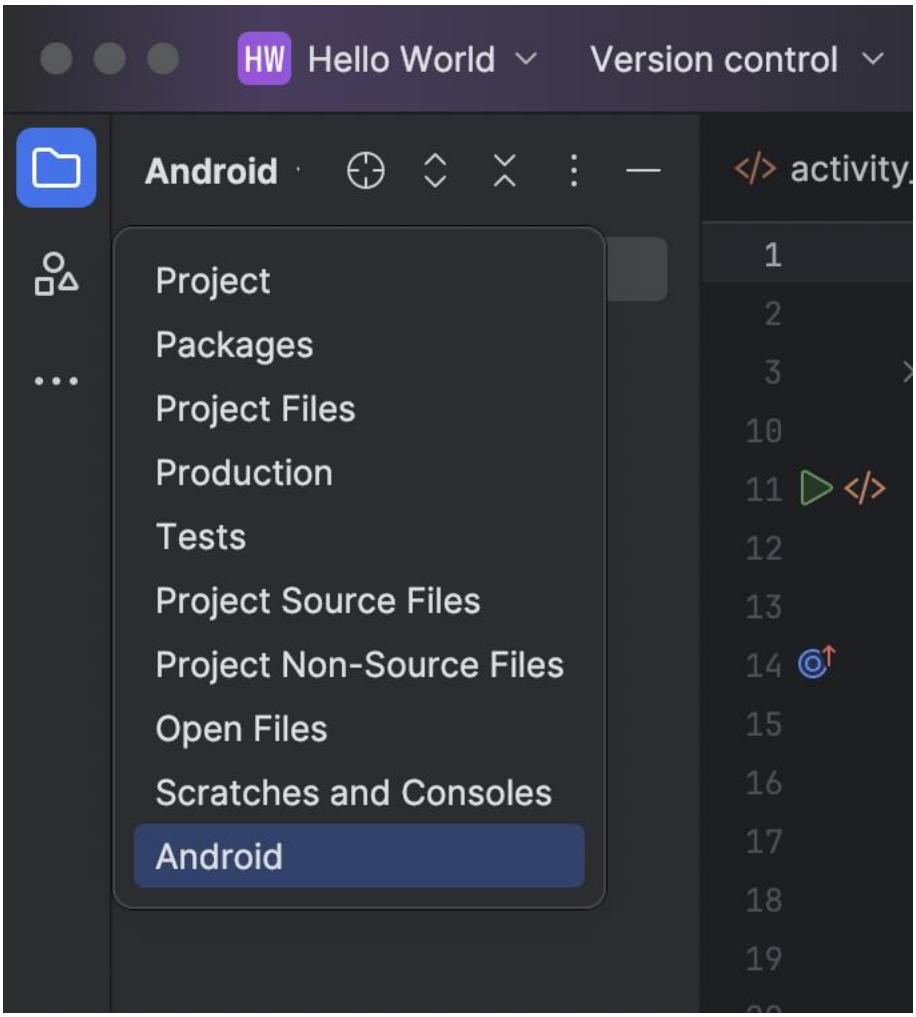
3. Nhấp vào tab **MainActivity.java** để xem trình chỉnh sửa mã nguồn như hình dưới đây.



2.2 Khám phá ngăn Project > Android

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

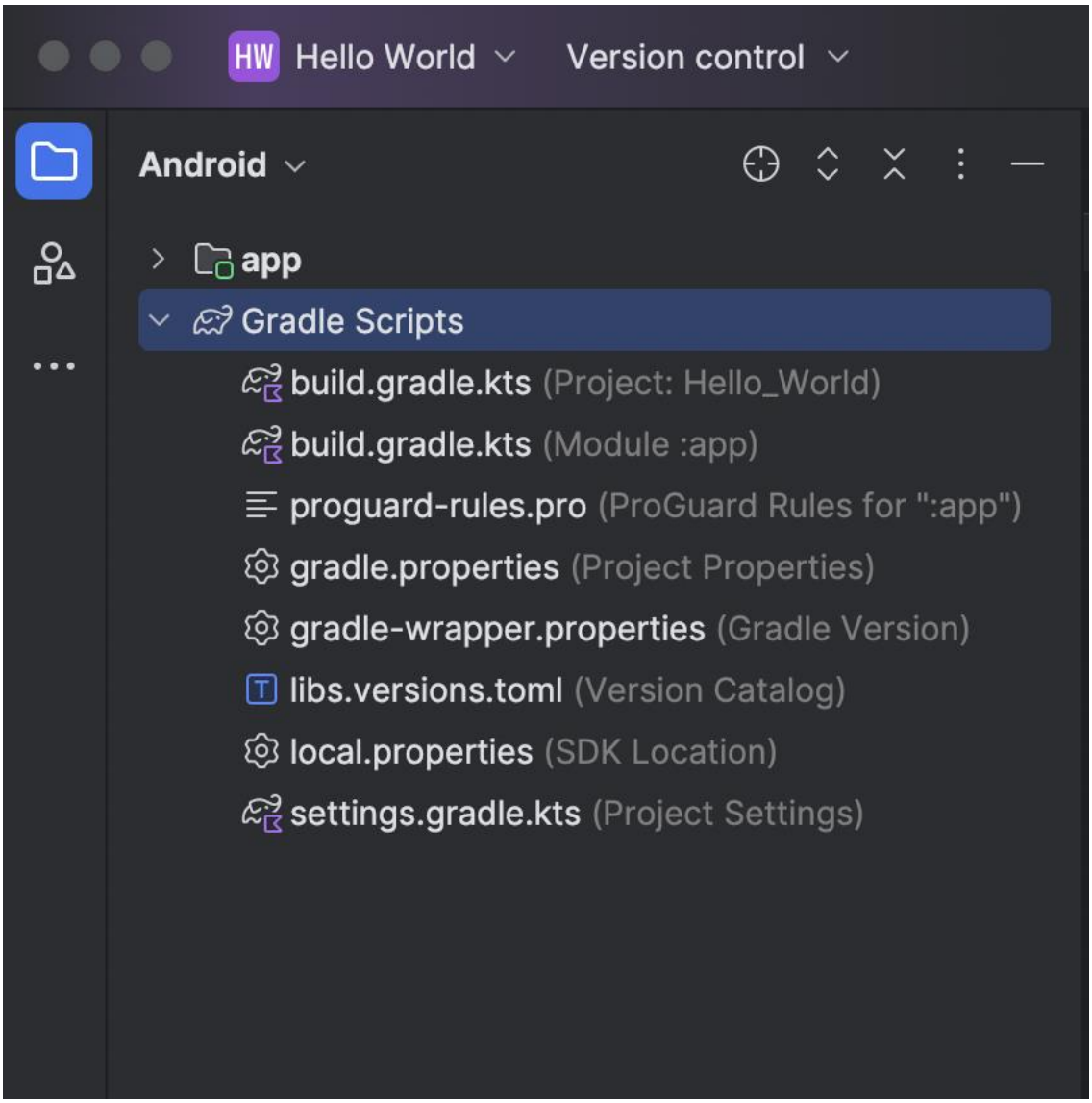
1. Nếu chưa được chọn, nhấp vào tab **Project** trong cột tab dọc ở bên trái cửa sổ **Android Studio**. Ngăn **Project** sẽ xuất hiện.
2. Để xem dự án theo cấu trúc tiêu chuẩn của dự án **Android**, chọn **Android** từ menu pop-up ở phía trên cùng của ngăn **Project**, như hình dưới đây.



Lưu ý: Chương này và các chương khác sẽ gọi ngắn **Project** khi được đặt ở chế độ **Android** là **Project > Android**.

2.3 Khám phá thư mục Gradle Scripts

Hệ thống xây dựng **Gradle** trong **Android Studio** giúp dễ dàng thêm các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác vào quá trình xây dựng dưới dạng **dependencies**.
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 dưới đây.



Thực hiện theo các bước sau để khám phá hệ thống **Gradle**:

1. Nếu thư mục **Gradle Scripts** chưa được mở rộng, hãy nhấp vào biểu tượng tam giác để mở rộng nó.
   Thư mục này chứa tất cả các tệp cần thiết cho hệ thống 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** trong dự án của mình. Mỗi dự án **Android Studio** đều chứa một tệp **Gradle build** cấp cao nhất. Hầu hết thời gian, bạn sẽ không cần thực hiện bất kỳ thay đổi nào đối với tệp này, nhưng việc hiểu nội dung của nó vẫn rất hữu

ích.
Theo mặc định, tệp **build** cấp cao nhất sử dụng khối **buildscript** để định nghĩa các **Gradle repositories** và **dependencies** chung cho tất cả các **module** trong dự án. Khi **dependency** của bạn không phải là thư viện cục bộ hoặc cây tệp, **Gradle** sẽ tìm các tệp trong bất kỳ **repository** trực tuyến nào được chỉ định trong khối **repositories** của tệp này.
Theo mặc định, các dự án **Android Studio** mới sẽ khai báo **JCenter** và **Google** (bao gồm **Google Maven repository**) làm vị trí **repository**.

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

Ngoài tệp **build.gradle** cấp dự án, mỗi module sẽ có tệp **build.gradle** riêng, cho phép bạn cấu hình các thiết lập build cho từng module cụ thể (ứng dụng **HelloWorld** chỉ có một module).

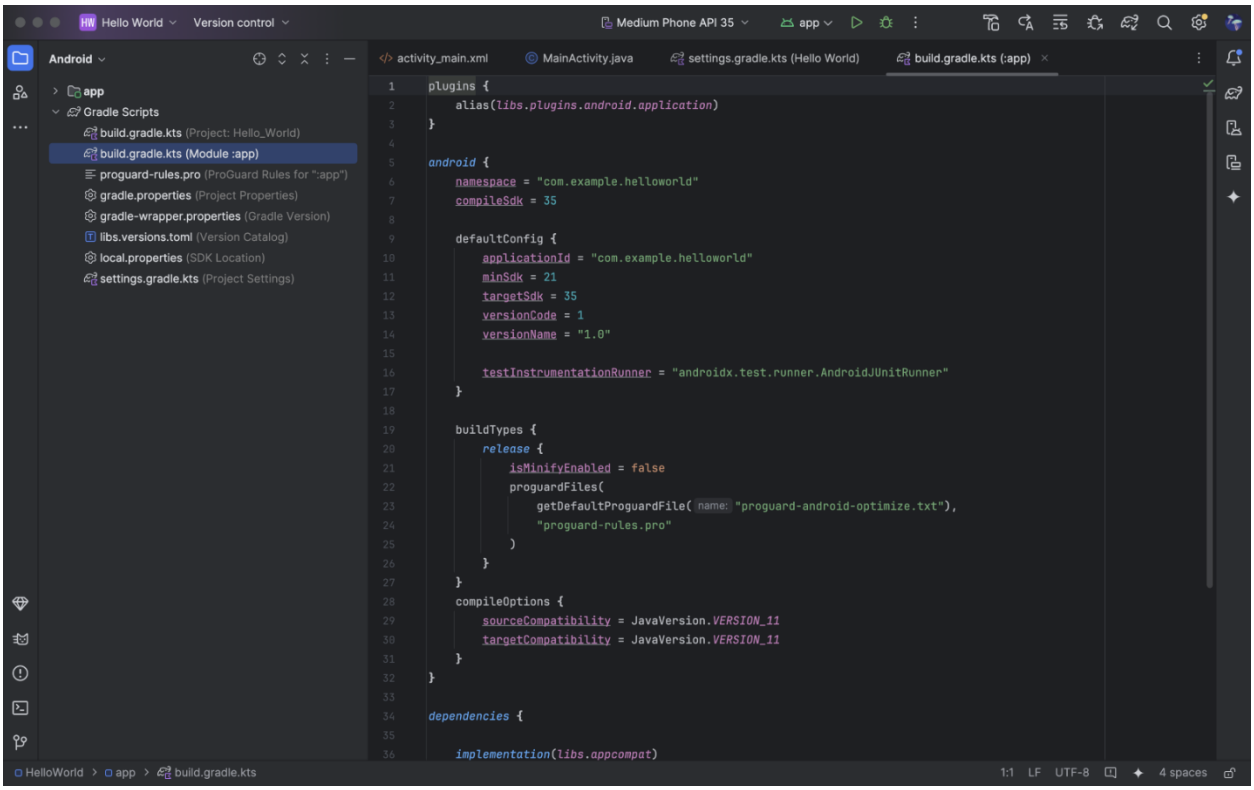
Việc cấu hình các thiết lập build này cho phép bạn cung cấp các tùy chọn đóng gói tùy chỉnh, chẳng hạn như các loại build bổ sung và **product flavors**. Bạn cũng có thể ghi đè các thiết lập trong tệp **AndroidManifest.xml** hoặc tệp **build.gradle** cấp cao nhất.

Đây thường là tệp được chỉnh sửa khi thay đổi các cấu hình ở cấp ứng dụng, chẳng hạn như khai báo **dependencies** trong phần **dependencies**.

Bạn có thể khai báo một thư viện phụ thuộc bằng cách sử dụng một trong số các cấu hình phụ thuộc khác nhau. Mỗi cấu hình phụ thuộc cung cấp cho **Gradle** các hướng dẫn khác nhau về cách sử dụng thư viện.

Ví dụ, câu lệnh **implementation fileTree(dir: 'libs', include: ['*.jar'])** sẽ thêm phụ thuộc cho tất cả các tệp ".jar" trong thư mục **libs**.

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

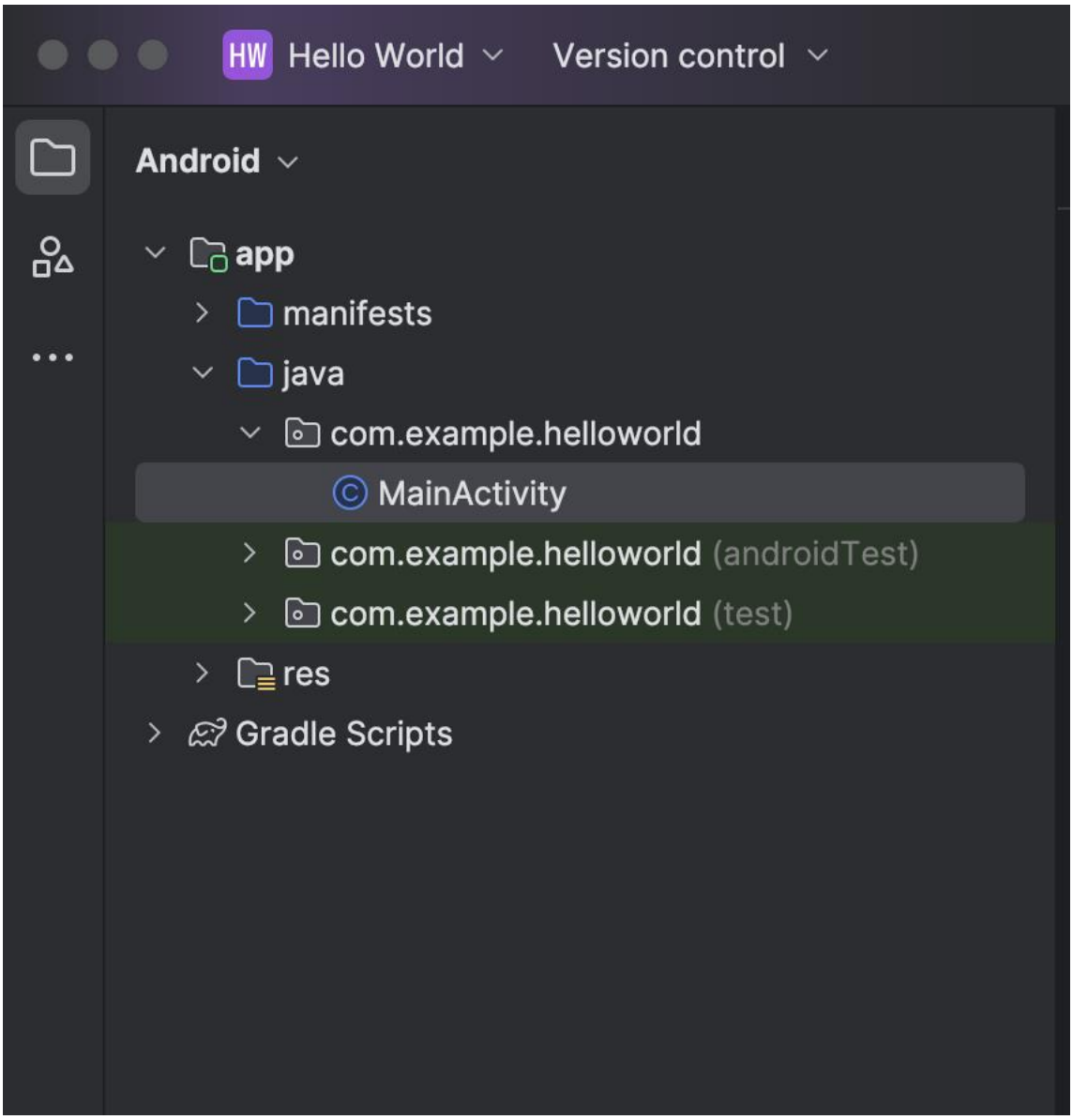


4. Nhấp vào biểu tượng tam giác để đóng **Gradle Scripts**.

2.4 Khám phá các thư mục app và res

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

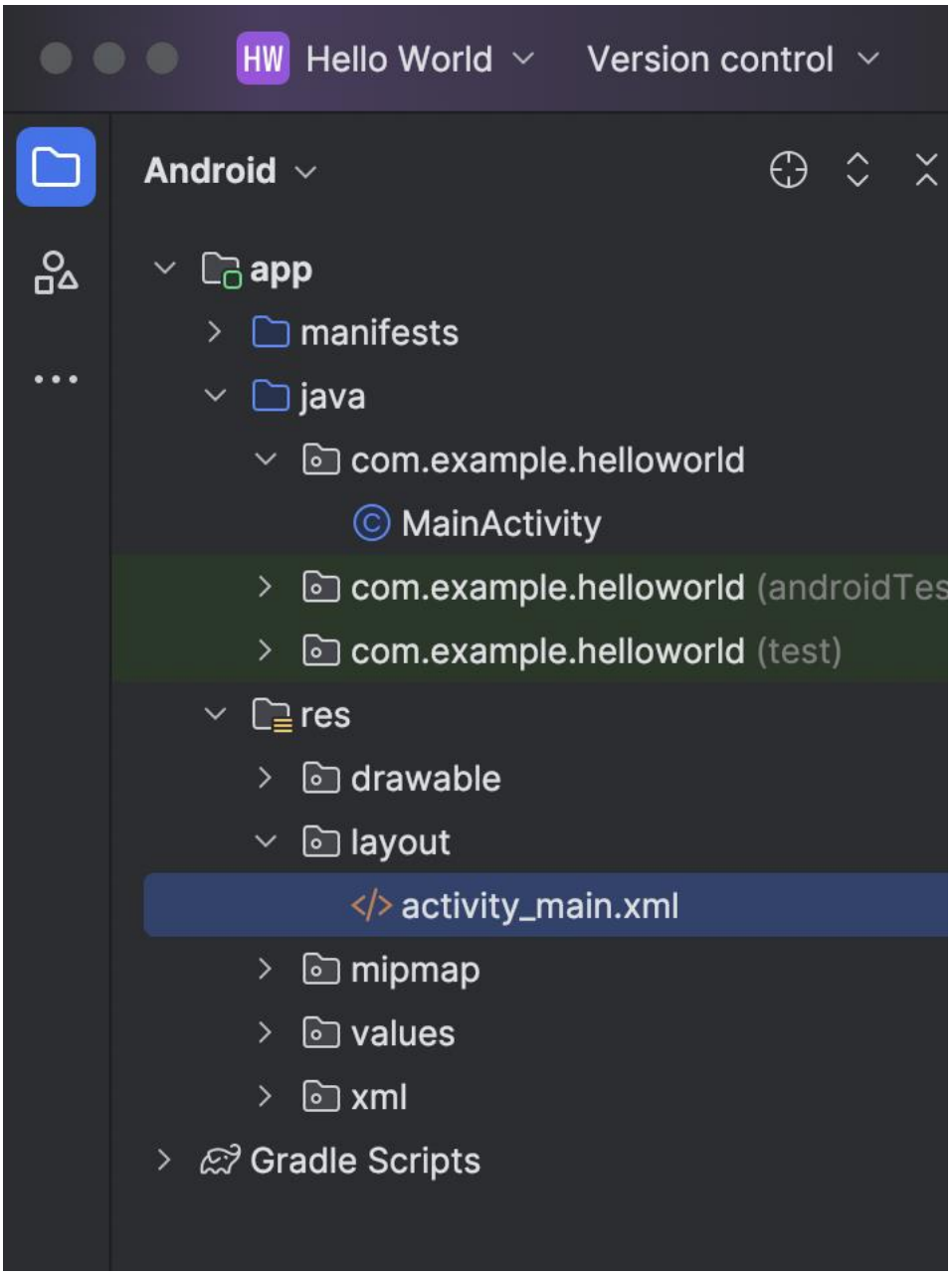
1. Mở rộng thư mục **app**, sau đó mở rộng thư mục **java** và thư mục **com.example.android.helloworld** để xem tệp **MainActivity.java**.
2. Nhấp đúp vào tệp để mở nó trong trình chỉnh sửa mã (**code editor**).



Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như minh họa ở hình 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 (**app package**). Hai thư mục còn lại được sử dụng cho mục đích kiểm thử (**testing**) và sẽ được mô tả trong bài học khác. Đối với ứng dụng **Hello World**, chỉ có một gói (**package**) và nó chứa **MainActivity.java**.

Tên của **Activity** (màn hình) đầu tiên mà người dùng nhìn thấy, đồng thời khởi tạo các tài nguyên trên toàn ứng dụng (**app-wide resources**), theo thông lệ được gọi là **MainActivity** (phần mở rộng của tệp bị lược bỏ trong **Project > Android pane**).

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 bố cục (**layout editor**).



Thư mục **res** chứa các tài nguyên (**resources**) như bố cục (**layouts**), chuỗi ký tự (**strings**) và hình ảnh (**images**).

Một **Activity** thường được liên kết với một bố cục giao diện người dùng (**UI views**) được định nghĩa dưới dạng tệp XML. Tệp này thường được đặt tên theo **Activity** tương ứng.

3. **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, những thông tin mà hệ thống cần phải có trước khi có thể chạy bất kỳ mã lệnh (**code**) 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 (**components**) của ứng dụng Android của bạn.

Tất cả các thành phần trong một ứng dụng, chẳng hạn như mỗi **Activity**, phải được khai báo trong tệp XML này.

Trong các bài học khác của khóa học, bạn sẽ chỉnh sửa tệp này để thêm tính năng (**features**) và quyền truy cập tính năng (**feature permissions**).

Để tìm hiểu thêm, hãy xem **App Manifest Overview**.

Bước 3: Sử dụng thiết bị ảo(emulator)

Trong nhiệm vụ này, bạn sẽ sử dụng **Android Virtual Device (AVD) Manager** để tạo một thiết bị ảo (**emulator**) mô phỏng cấu hình của 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 cho **Android Studio**.

Sử dụng **AVD Manager**, bạn có thể xác định các đặc điểm phần cứng (**hardware characteristics**) của thiết bị, cấp API (**API level**), dung lượng lưu trữ (**storage**), giao diện (**skin**) và các thuộc tính khác, sau đó lưu chúng dưới dạng thiết bị ảo.

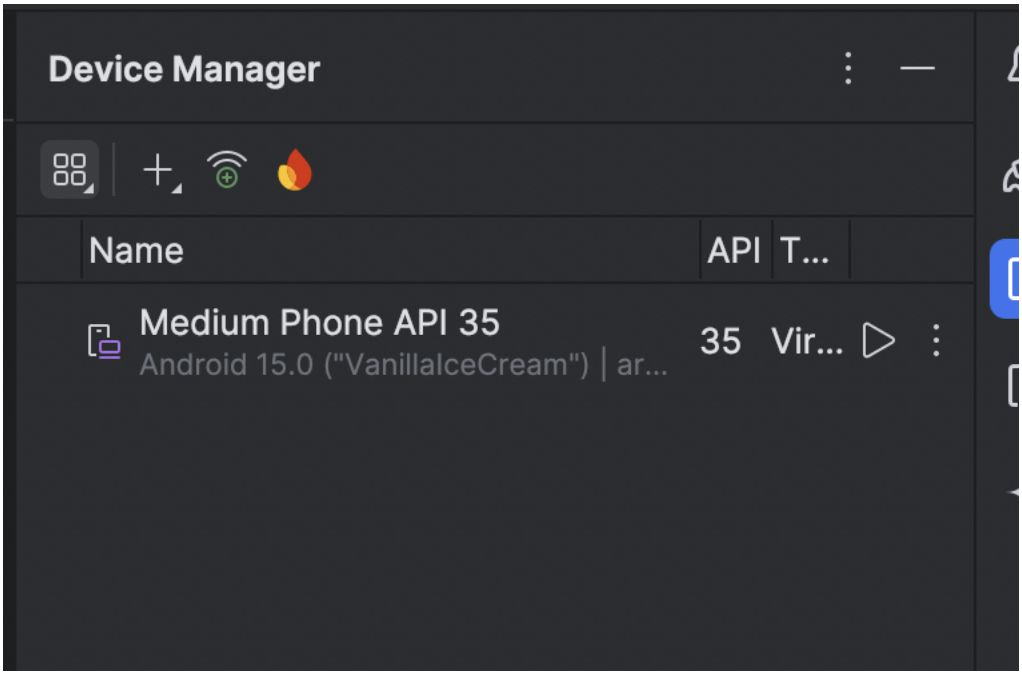
Với các thiết bị ảo, bạn có thể kiểm thử ứng dụng trên các cấu hình thiết bị khác nhau (như máy tính bảng và điện thoại) với các cấp API khác nhau mà không cần sử dụng thiết bị vật lý.

3.1 Tạo thiết bị ảo Android (AVD)

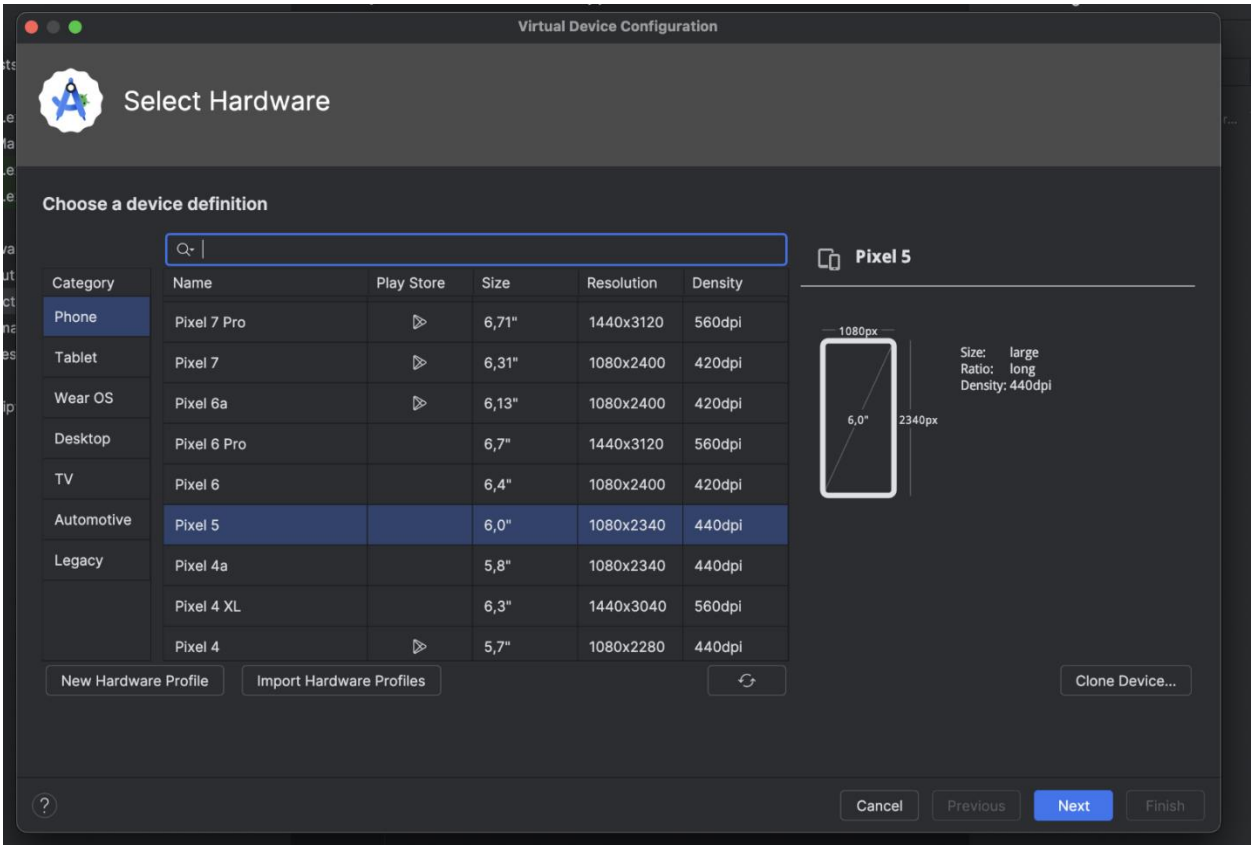
Để chạy trình giả lập (**emulator**) trên máy tính, bạn cần tạo một cấu hình mô tả thiết bị ảo.

1. Trong **Android Studio**, chọn **Tools > Android > AVD Manager**, hoặc nhấp vào biểu tượng **AVD Manager** trên thanh công cụ (**toolbar**).

   Màn hình **Your Virtual Devices** sẽ xuất hiện. Nếu bạn đã tạo thiết bị ảo trước đó, màn hình sẽ hiển thị chúng. Nếu chưa, bạn sẽ thấy danh sách trống.



2. Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** sẽ xuất hiện, hiển thị danh sách các thiết bị phần cứng đã được cấu hình sẵn. Với mỗi thiết bị, bảng sẽ cung cấp các cột sau: **Size**: Kích thước đường chéo của màn hình. **Resolution**: Độ phân giải màn hình tính bằng pixel. **Density**: Mật độ điểm ảnh (**pixel density**).



3. Chọn một thiết bị như **Nexus 5x** hoặc **Pixel XL**, sau đó nhấp **Next**. Cửa sổ **System Image** sẽ xuất hiện.
4. Nhấp vào tab **Recommended** nếu nó chưa được chọn và chọn phiên bản hệ điều hành Android để chạy trên thiết bị ảo (ví dụ như **Oreo**).

Có nhiều phiên bản khác ngoài những phiên bản được hiển thị trong tab **Recommended**. Hãy xem các tab **x86 Images** và **Other Images** để xem thêm. Nếu xuất hiện liên kết **Download** bên cạnh hệ điều hành bạn muốn sử dụng,

nghĩa là hệ điều hành đó chưa được cài đặt. Nhấp vào liên kết đó để bắt đầu tải xuống và nhấp **Finish** khi hoàn tất.

5. Sau khi chọn hệ điều hành, nhấp **Next**. Cửa sổ **Android Virtual Device (AVD)** sẽ xuất hiện. Bạn cũng có thể đổi tên AVD. Kiểm tra cấu hình của bạn và nhấp **Finish**.

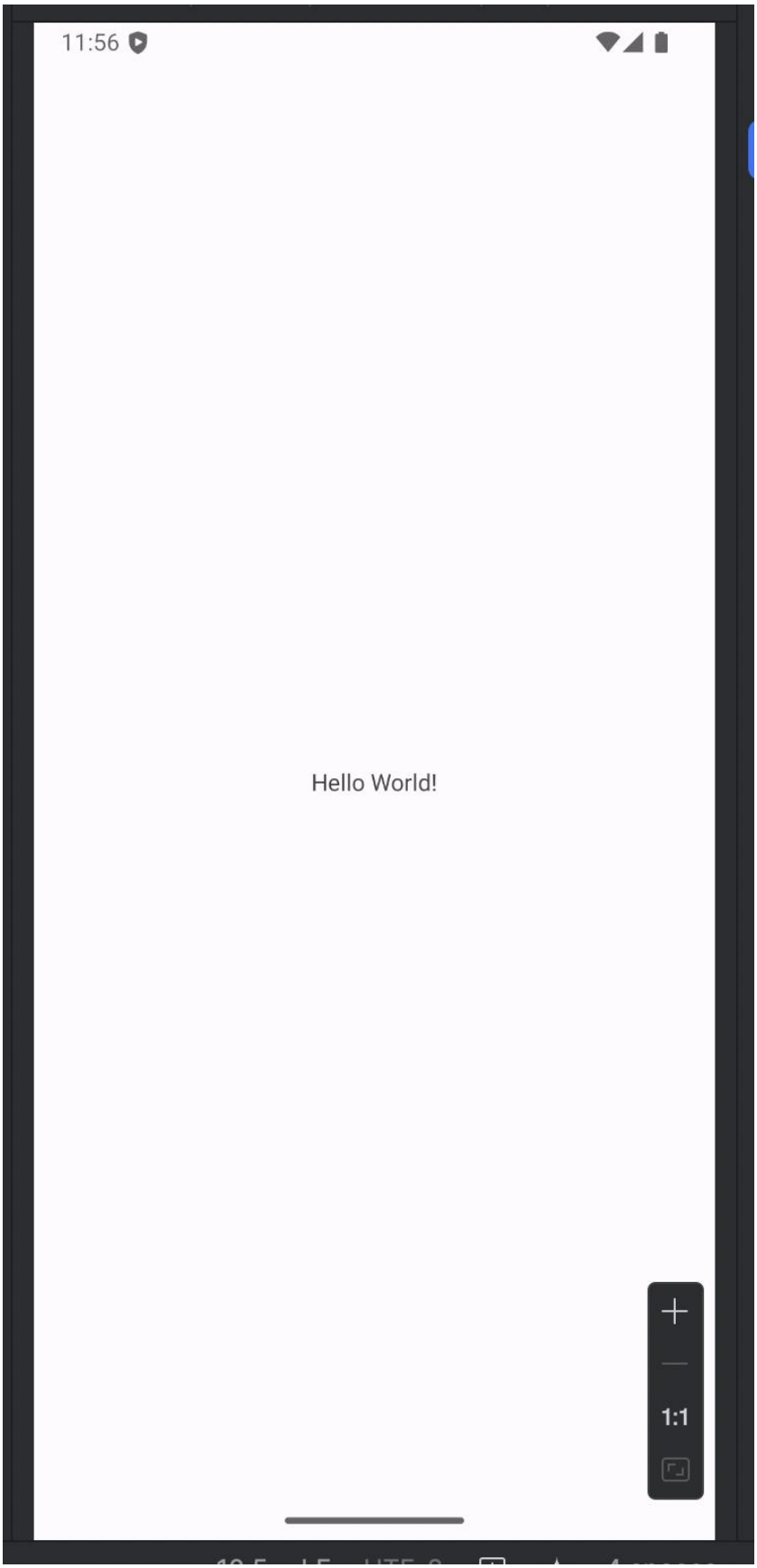
3.2 Chạy ứng dụng trên thiết bị ảo

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

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

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

Bạn sẽ thấy ứng dụng **Hello World** hiển thị như hình minh họa sau.


11:56
Hello World!
1:1

Mẹo: Khi kiểm tra trên thiết bị ảo (**virtual device**), bạn nên khởi động nó ngay từ đầu phiên làm việc. Bạn **không nên đóng** trình giả lập cho đến khi hoàn tất việc kiểm tra ứng dụng, để tránh phải khởi động lại thiết bị, gây mất thời gian.

Để đóng thiết bị ảo, bạn có thể:

- Nhấp vào nút **X** ở góc trên cùng của trình giả lập.
- Chọn **Quit** từ menu.
- Hoặc nhấn tổ hợp phím **Control + Q** trên **Windows** hoặc **Command + Q** trên **macOS**.

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

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

Những gì bạn cần:

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

4.1 Bật chế độ USB Debugging

Để cho phép Android Studio giao tiếp với thiết bị của bạn, bạn phải bật chế độ USB Debugging trên thiết bị Android. Tùy chọn này được bật trong phần **Developer options** của thiết bị.

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

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

4.2 Chạy ứng dụng 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 tính phát triển bằng cáp USB.
2. Nhấp vào nút **Run** trên thanh công cụ. Cửa sổ **Select Deployment Target** sẽ mở ra với danh sách các trình giả lập và thiết bị đã kết nối.
3. Chọn thiết bị của bạn và nhấp vào **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 cách 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 "unauthorized", hãy làm theo các bước sau:

1. Rút thiết bị ra.
2. Trên thiết bị, mở **Developer Options** trong ứng dụng **Cài đặt (Settings)**.
3. Nhấn vào **Revoke USB Debugging authorizations**.
4. Kết nối lại thiết bị với máy tính.
5. Khi được nhắc, hãy cấp quyền cho thiết bị.

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

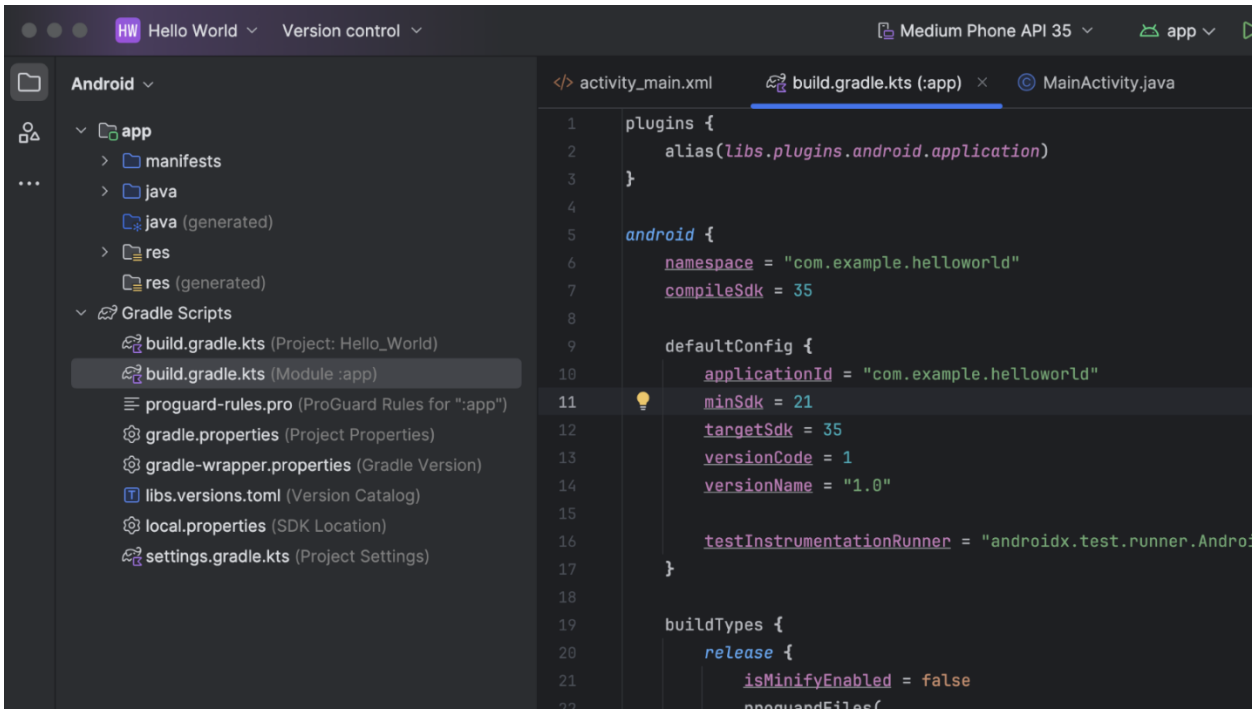
Bước 5: Thay đổi cấu hình Gradle của ứng dụng

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

5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Thực hiện 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 xuất hiện trong trình chỉnh sửa mã.
2. Trong khối **defaultConfig**, thay đổi giá trị của **minSdkVersion** thành **17** như minh họa bên dưới (ban đầu được đặt thành **15**).



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

5.2 Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện các thay đổi đối với các tệp cấu hình build trong một dự án, Android Studio yêu cầu bạn đồng bộ các tệp dự án để có thể nhập các thay đổi cấu hình build và chạy một số kiểm tra để đảm bảo cấu hình sẽ không tạo ra lỗi build.

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

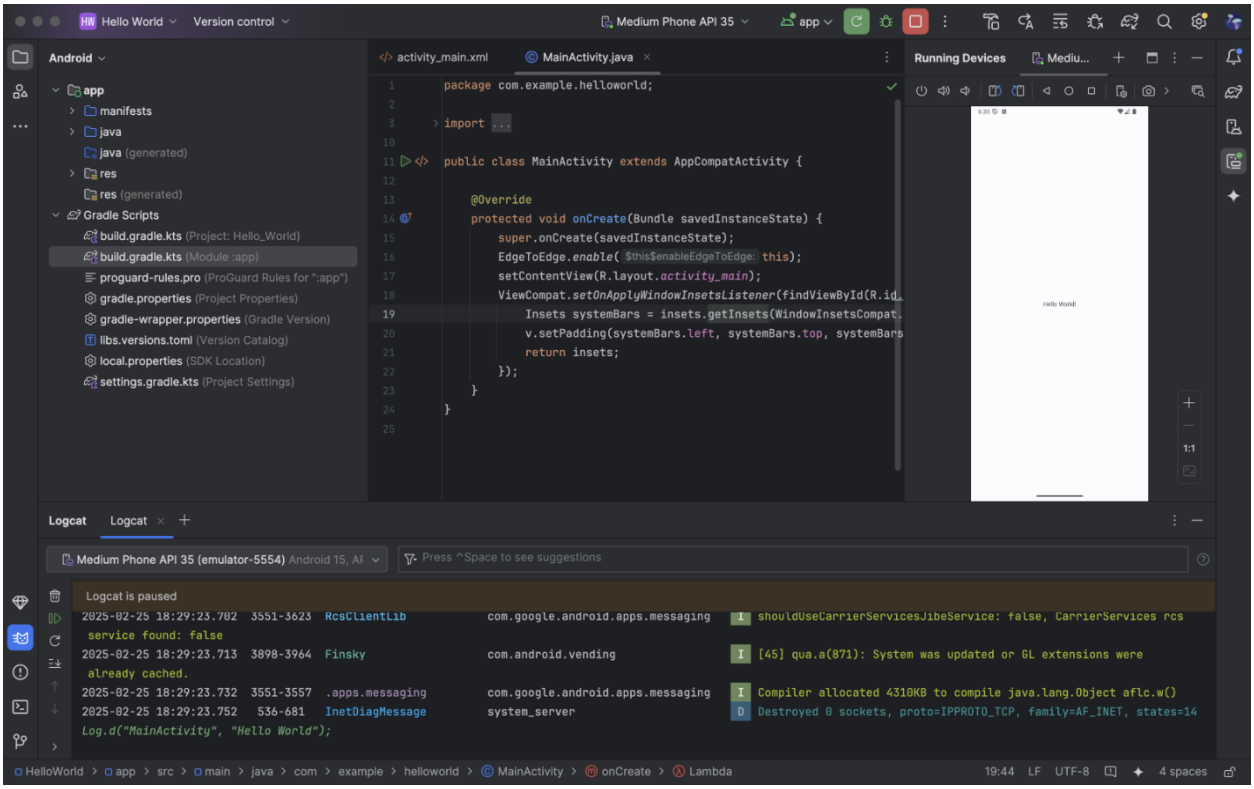
Bước 6 : Thêm các câu lệnh log vào ứng dụng của bạn

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh **Log** vào ứng dụng của mình, các câu lệnh này sẽ hiển thị thông báo trong bảng **Logcat**.

Thông báo log là một công cụ gỡ lỗi mạnh mẽ mà bạn có thể sử dụng để kiểm tra các giá trị, đường dẫn thực thi và báo cáo ngoại lệ.

6.1 Xem bảng Logcat

Để xem bảng **Logcat**, hãy nhấp vào tab **Logcat** ở phía dưới của cửa sổ **Android Studio** như minh họa trong hình dưới đây.



Trong hình minh họa trên:

1. Tab Logcat để mở và đóng bảng Logcat, bảng này hiển thị thông tin về ứng dụng của bạn khi ứng dụng đang chạy. Nếu bạn thêm các câu lệnh Log vào ứng dụng, các thông báo Log sẽ xuất hiện tại đây.

2. Menu cấp độ Log được đặt thành Verbose (mặc định), hiển thị tất cả các thông báo Log. Các cài đặt khác bao gồm Debug, Error, Info và Warn.

6.2 Thêm các câu lệnh log vào ứng dụng

Các câu lệnh Log trong mã ứng dụng của bạn sẽ hiển thị thông báo trong bảng Logcat. Ví dụ: Log.d("MainActivity", "Hello World");

Các phần của thông báo gồm:

- **Log:** Lớp Log để gửi các thông báo log đến bảng Logcat.
- **d:** Cấp độ Log Debug để lọc hiển thị thông báo log trong bảng Logcat. Các cấp độ log khác bao gồm:
    - **e** cho Error (lỗi),
    - **w** cho Warn (cảnh báo),
    - **i** cho Info (thông tin).
- **"MainActivity":** Tham số đầu tiên là một tag (nhãn) có thể được sử dụng để lọc các thông báo trong bảng Logcat. Thông thường, đây là tên của Activity nơi thông báo bắt nguồn. Tuy nhiên, bạn có thể đặt bất kỳ tên nào hữu ích cho việc gỡ lỗi.
    - Theo thông lệ, các tag log được định nghĩa là hằng số cho Activity như sau:
      private static final String LOG_TAG = MainActivity.class.getSimpleName();
- **"Hello world":** Tham 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 tệp **MainActivity**.
2. Để tự động thêm các import rõ ràng vào dự án (chẳng hạn như android.util.Log cần thiết để sử dụng Log), hãy chọn:
    - **File > Settings** trên Windows, hoặc
    - **Android Studio > Preferences** trên macOS.
3. Chọn **Editor > General > Auto Import**.
    - Tích chọn tất cả các hộp kiểm.
    - Đặt **Insert imports on paste** thành **All**.
4. Nhấp vào **Apply**, sau đó nhấp vào **OK**.

5. Trong phương thức **onCreate()** của **MainActivity**, thêm câu lệnh sau:
   Log.d("MainActivity", "Hello World");

Phương thức **onCreate()** bây giờ sẽ trông như sau:

```
@Override

protected void onCreate(Bundle savedInstanceState) {

  super.onCreate(savedInstanceState);

  setContentView(R.layout.activity_main);

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

}
```

6. Nếu bảng **Logcat** chưa được mở, hãy nhấp vào tab **Logcat** ở cuối cửa sổ **Android Studio** để mở nó.
7. Kiểm tra xem tên của thiết bị đích (**target**) và tên gói (**package name**) của ứng dụng có chính xác không.
8. Thay đổi cấp độ **Log** trong bảng **Logcat** thành **Debug** (hoặc để mặc định là **Verbose** vì chỉ có rất ít thông báo log).
9. Chạy ứng dụng của bạn.

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

```
11-24 14:06:59.001 4696-4696/? D/MainActivity: Hello World
```

---

1. Thử thách lập trình

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

Thử thách:

2. Tạo một dự án mới trong **Android Studio**.
3. Thay đổi thông điệp "Hello World" thành **"Happy Birthday to "** cùng với tên của một người có sinh nhật gần đây.

4. (**Tùy chọn**) Chụp ảnh màn hình ứng dụng đã hoàn thành của bạn và gửi email cho người có sinh nhật mà bạn đã quên.
5. Một cách sử dụng phổ biến của lớp **Log** là ghi lại các ngoại lệ Java khi chúng xảy ra trong chương trình của bạn.
    - Có một số phương thức hữu ích như **Log.e()** mà bạn có thể sử dụng cho mục đích này.
    - Hãy khám phá thêm.

- **Phương thức để thêm ngoại lệ vào thông báo Log**

Bạn có thể sử dụng các phương thức sau của lớp **Log** để bao gồm ngoại lệ (**exception**) trong thông báo Log:

- Log.e(String tag, String msg, Throwable tr) – Ghi thông báo lỗi kèm theo ngoại lệ.
- Log.w(String tag, String msg, Throwable tr) – Ghi cảnh báo kèm theo ngoại lệ.
- Log.i(String tag, String msg, Throwable tr) – Ghi thông báo thông tin kèm theo ngoại lệ.
- Log.d(String tag, String msg, Throwable tr) – Ghi thông báo gỡ lỗi kèm theo ngoại lệ.
- Log.v(String tag, String msg, Throwable tr) – Ghi thông báo chi tiết kèm theo ngoại lệ.

Yêu cầu: Viết mã trong ứng dụng của bạn để kích hoạt và ghi lại một ngoại lệ.

- **Tóm tắt**

- Để cài đặt **Android Studio**, hãy truy cập **Android Studio** và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo một ứng dụng mới, hãy đảm bảo đặt **API 15: Android 4.0.3 IceCreamSandwich** làm **Minimum SDK**.
- Để xem cấu trúc phân cấp Android của ứng dụng trong bảng **Project**, hãy nhấp vào tab **Project** trong cột tab dọc, sau đó chọn **Android** trong menu thả xuống ở trên cùng.
- Chỉnh sửa tệp **build.gradle (Module:app)** khi cần thêm thư viện mới hoặc thay đổi phiên bản thư viện cho dự án.
- Tất cả mã và tài nguyên cho ứng dụng nằm trong thư mục **app** và **res**.

    - Thư mục **java** chứa các hoạt động (**activities**), kiểm thử (**tests**) và các thành phần khác dưới dạng mã nguồn Java.
    - Thư mục **res** chứa các tài nguyên như bố cục (**layouts**), chuỗi văn bản (**strings**) và hình ảnh (**images**).
- Chỉnh sửa tệp **AndroidManifest.xml** để thêm các tính năng, thành phần và quyền (**permissions**) cho ứng dụng Android.
    - Tất cả các thành phần của ứng dụng, chẳng hạn như nhiều hoạt động (**activities**), phải được khai báo trong tệp XML này.
- Sử dụng **Android Virtual Device (AVD) Manager** để tạo thiết bị ảo (**emulator**) nhằm chạy ứng dụng của bạn.
- Thêm các câu lệnh **Log** vào ứng dụng để hiển thị thông báo trong bảng **Logcat** như một công cụ cơ bản để gỡ lỗi.
- Để chạy ứng dụng trên thiết bị Android vật lý 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**), chọn **Developer options**, rồi bật **USB Debugging**.

Bài tập về nhà

Xây dựng và chạy một ứng dụng

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

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) RecycleView

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