

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.....	Error! Bookmark not defined.
1.3) Trình chỉnh sửa bộ cục	Error! Bookmark not defined.
1.4) Văn bản và các chế độ cuộn.....	3
1.5) Tài nguyên có sẵn	75
Bài 2) Activities	75
2.1) Activity và Intent	75
2.2) Vòng đời của Activity và trạng thái	75
2.3) Intent ngầm định	75
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ.....	75
3.1) Trình gỡ lỗi.....	75
3.2) Kiểm thử đơn vị	75
3.3) Thư viện hỗ trợ.....	75
CHƯƠNG 2. Trải nghiệm người dùng	76
Bài 1) Tương tác người dùng.....	76
1.1) Hình ảnh có thể chọn	76
1.2) Các điều khiển nhập liệu.....	97
1.3) Menu và bộ chọn.....	97
1.4) Điều hướng người dùng	97
1.5) RecycleView	97
Bài 2) Trải nghiệm người dùng thú vị	97
2.1) Hình vẽ, định kiểu và chủ đề.....	97
2.2) Thẻ và màu sắc.....	97
2.3) Bộ cục thích ứng	97
Bài 3) Kiểm thử giao diện người dùng	97

3.1) Espresso cho việc kiểm tra UI.....	97
CHƯƠNG 3. Làm việc trong nền	97
Bài 1) Các tác vụ nền	97
1.1) AsyncTask	97
1.2) AsyncTask và AsyncTaskLoader	97
1.3) Broadcast receivers	97
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền	97
2.1) Thông báo.....	97
2.2) Trình quản lý cảnh báo.....	97
2.3) JobScheduler	97
CHƯƠNG 4. Lưu trữ dữ liệu người dùng	97
Bài 1) Tùy chọn và cài đặt	97
1.1) Shared preferences	97
1.2) Cài đặt ứng dụng	134
Bài 2) Lưu trữ dữ liệu với Room	134
2.1) Room, LiveData và ViewModel	134
2.2) Room, LiveData và ViewModel	134
3.1) Trinhf 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 đã cài đặt thành công Android Studio, bạn sẽ tạo ứng dụng Hello World từ một mẫu, một dự án mới. Đó là một ứng dụng đơn giản hiển thị dòng chữ “Hello World” trên màn hình của thiết bị Android ảo hoặc thiết bị vật lý.

Khi mà ứng dụng chạy thành công thì sẽ hiển thị như sau:

17:15

...0,8KB/s

Hello World!

Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường tích hợp (IDE) hoàn chỉnh bao gồm chỉnh sửa mã nâng cao và một bộ mẫu ứng dụng. Ngoài ra nó còn chứa các công cụ dành cho phát triển, sửa lỗi, kiểm thử và hiệu suất giúp việc phát triển các ứng dụng trở nên nhanh chóng và dễ dàng hơn. Bạn có thể thử nghiệm ứng dụng của mình với một loạt các trình giả lập đã được cấu hình sẵn hoặc trên thiết bị di động của riêng bạn, xây dựng ứng dụng sẵn sàng sử dụng và xuất bản trên cửa hàng Google Play.

Chú ý: Android Studio đang liên tục được cải tiến. Để biết thông tin mới nhất về yêu cầu hệ thống và hướng dẫn cài đặt, xem Android Studio.

Android Studio có sẵn cho các máy tính chạy Windows hoặc Linux và cho máy Mac chạy macOS. OpenJDK mới nhất (Java Development Kit) được đi kèm với Android Studio.

Để khởi động và chạy với Android Studio, đầu tiên kiểm tra các yêu cầu hệ thống để đảm bảo rằng hệ thống đáp ứng chúng. Việc cài đặt tương tự cho tất cả các nền tảng. Bất kì sự khác biệt nào cũng được ghi chú dưới đây:

1. Điều hướng đến Android developers site và làm theo các hướng dẫn để tải về 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 được chọn để cài đặt.
3. Sau khi hoàn thành việc cài đặt, Setup Wizard sẽ được 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 mạng của bạn và một số bước có thể bị lặp lại.
4. Khi quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng để tạo dự án đầu tiên của mình.

Nhiệm vụ 2: Tạo ứng dụng Hello World

Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị “Hello World” để xác minh rằng Android Studio đã được cài đặt đúng cách và để tìm hiểu những điều cơ bản của việc phát triển với Android Studio.

Bước 1: Tạo dự án ứng dụng

1. Mở Android Studio nếu nó chưa được mở.
2. Trong cửa sổ chính **Welcom to Android Studio**, nhấp chuột vào **Start a new Android Studio project**.
3. Trong cửa sổ **Create Android Studio**, gõ **Hello World** vào dòng **Application name**.

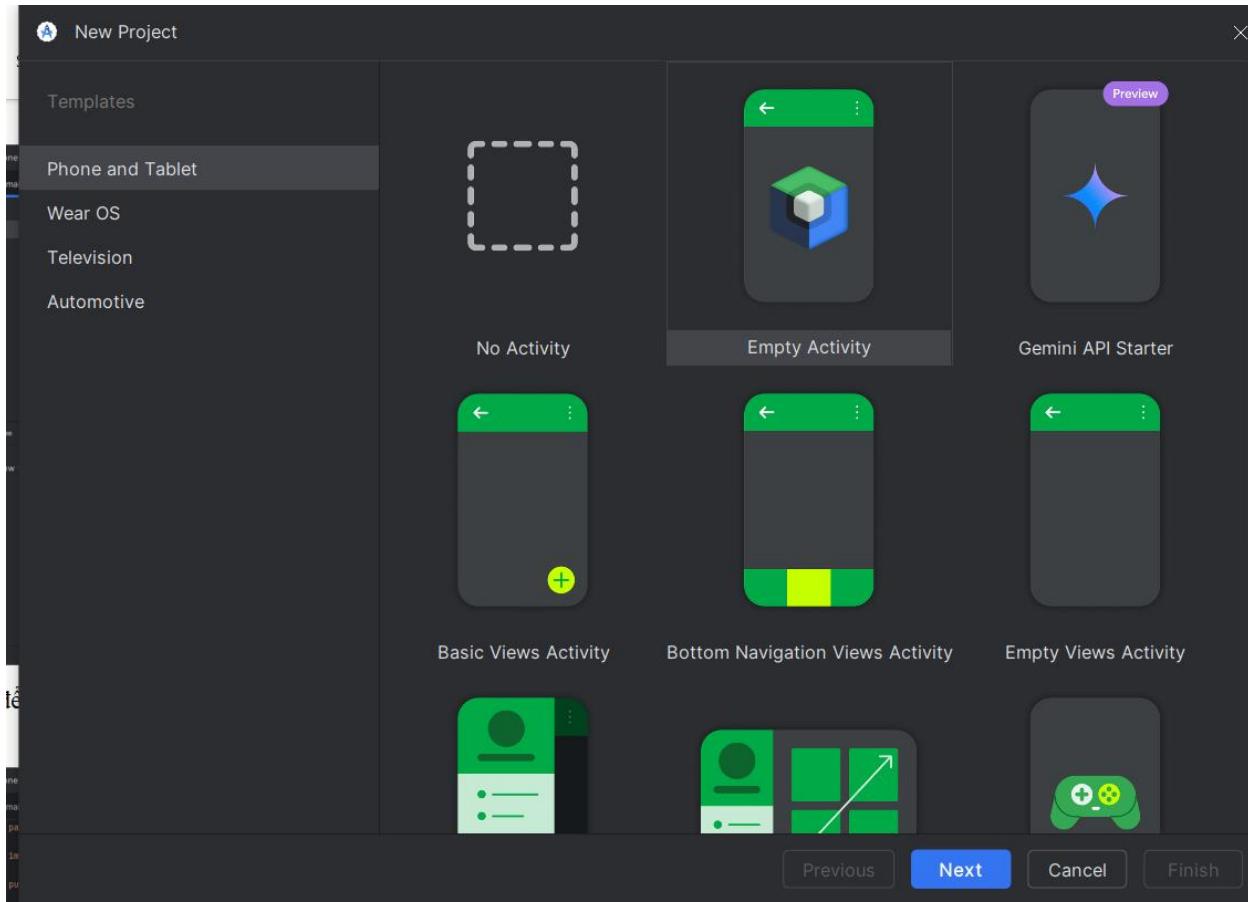
4. Xác nhận rằng **Project location** mặc định là nơi mà bạn muốn lưu trữ ứng dụng Hello World của bạn và các dự án Android Studio khác hoặc thay đổi thành thư mục được chọn bởi bạn.
5. Chấp nhận **android.example.com** mặc định cho tên miền **Company Domanic** hoặc tạo một tên miền công ty riêng biệt.

Nếu bạn không có kế hoạch về phát hành ứng dụng của bạn, bạn có thể chấp nhận mặc định. Hãy lưu ý rằng việc thay đổi gói tên của ứng dụng sau này sẽ tăng thêm công sức.

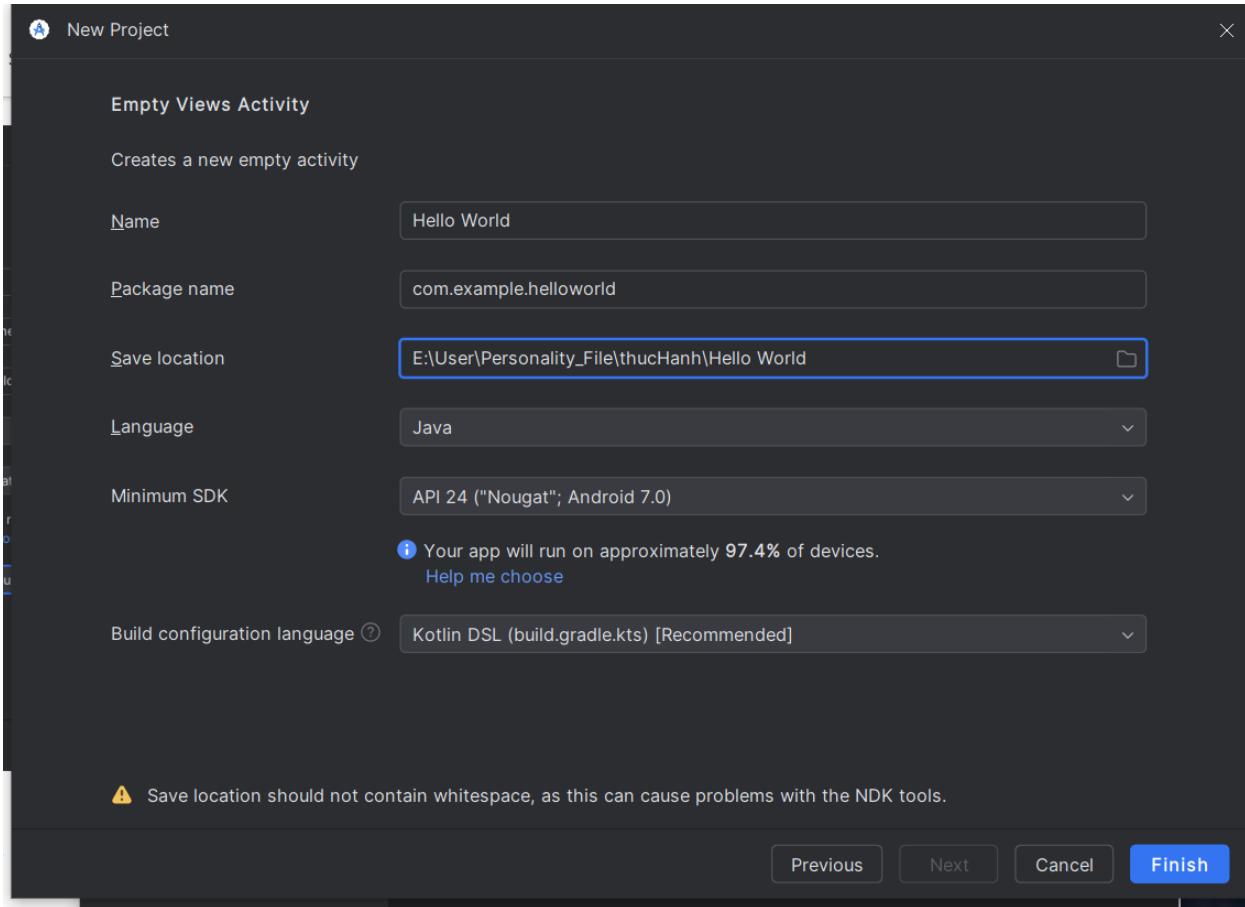
6. Bỏ chọn các tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấp vào **Next**.
7. Trên màn hình **Target Android Devices**, hãy đảm bảo rằng **Phone and Tablet** được chọn. Đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt là Minimum SDK; nếu không hãy sử dụng hộp thoại menu để đặt lại.

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

8. Bỏ chọn tùy chọn **Include Instant App support** và tắt cả các tùy chọn khác. Sau đó, nhấp vào Next. Nếu dự án của bạn yêu cầu các thành phần bổ sung cho SDK mục tiêu đã chọn, Andoid Studio sẽ tự động cài đặt chúng.
9. Cửa sổ **Add an Activity** xuất hiện. Activity là đơn vị duy nhất, tập trung những thứ mà người dùng có thể thực hiện. Nó là một thành phần quan trọng của bất kỳ ứng dụng Android nào. Activity thường có một giao diện (layout) liên quan đến nó, định nghĩa cách mà các phần tử giao diện người dùng xuất hiện trên màn hình. Android Studio cung cấp các mẫu Activity để giúp bạn bắt đầu. Đối với dự án Hello World, chọn **Empty Activity** như hình dưới đây và nhấp vào **Next**.



10. Màn hình **Configure Activity** xuất hiện (khác nhau tùy thuộc vào mẫu bạn chọn ở bước trước). Theo mặc định, Activity trống do mẫu cung cấp có tên là **MainActivity**. Bạn có thể thay đổi tên này nếu muốn, nhưng bài học này sử dụng **MainActivity**.
11. Đảm bảo rằng tùy chọn **Generate Layout** file được chọn. Tên bô cục theo mặc định là `activity_main`. Bạn có thể thay đổi nếu muốn, nhưng bài học này 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. Nhập vào **Finish**.



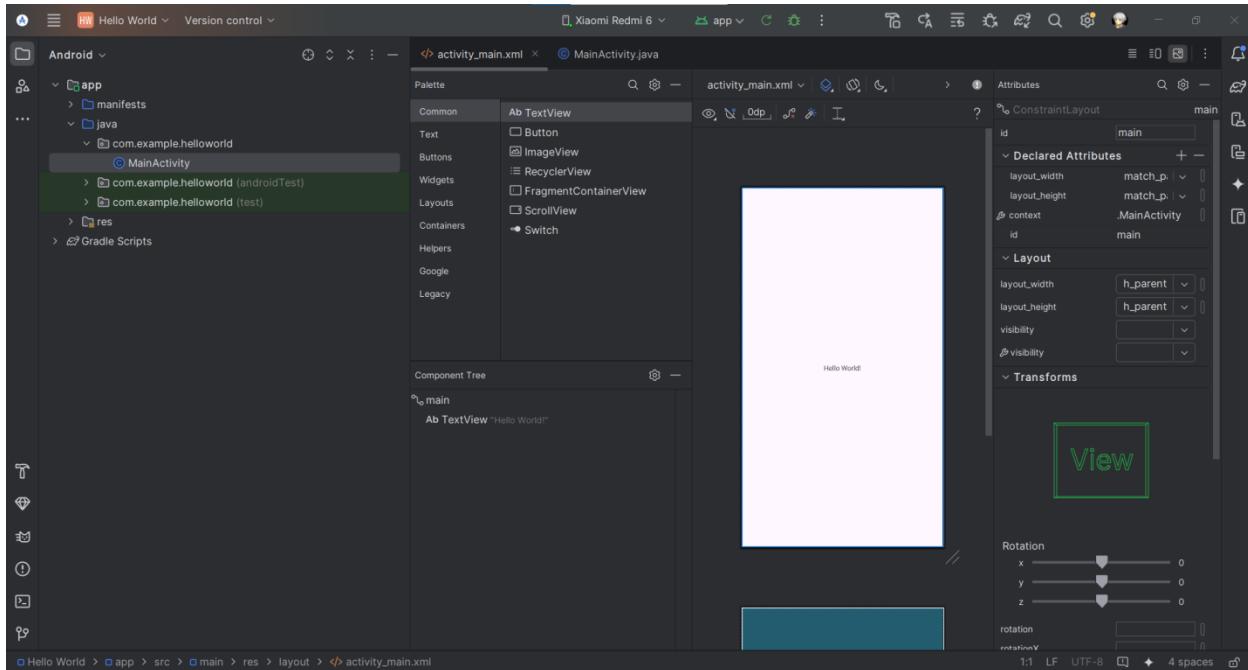
Android Studio tạo một thư mục cho các dự án của bạn và xây dựng dự án với Gradle (điều này có thể mất vài phút).

Mẹo: Xem trang **Configure your build developer** để biết thông tin chi tiết.

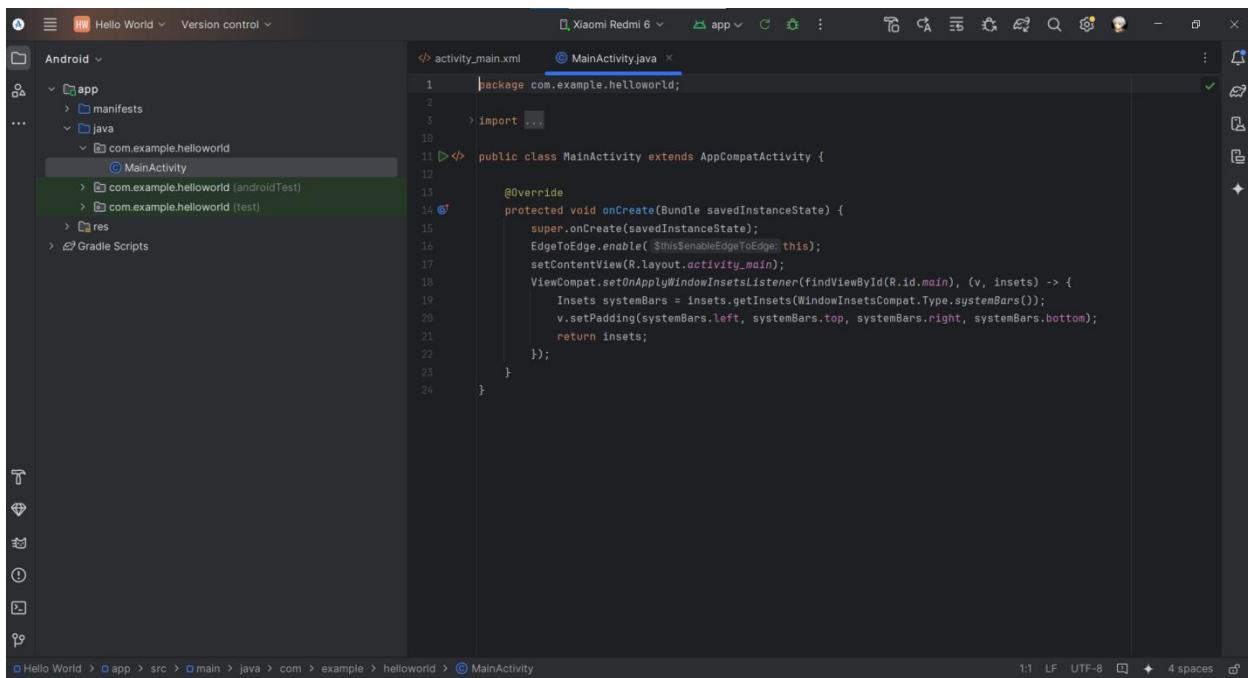
Bạn cũng có thể thấy thông báo “Tip of the day” với các phím tắt và các mẹo hữu ích khác. Nhấp vào **Close** để tắt thông báo.

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

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



3. Nhập vào MainActivity.java để xem trình chỉnh sửa mã, được hiển thị như bên dưới:

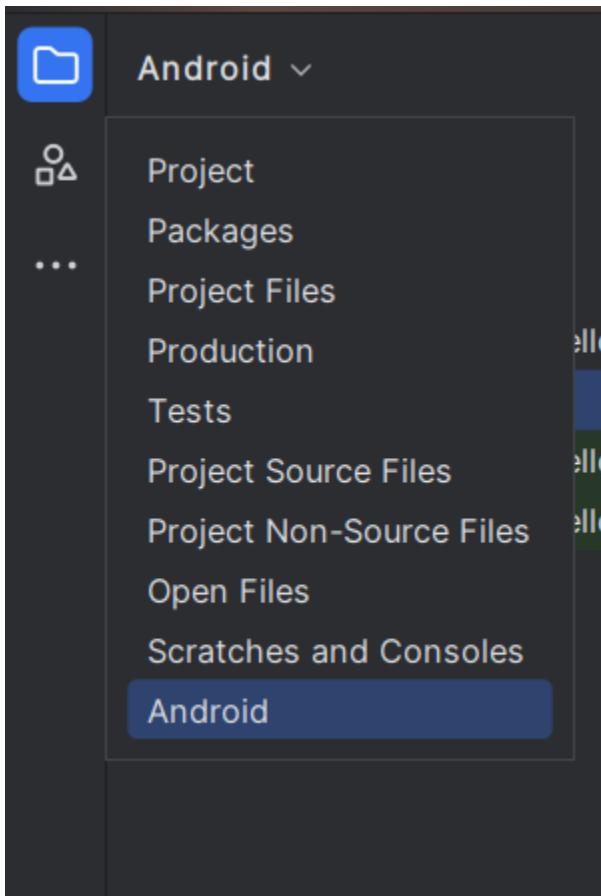


Bước 2: Khám phá dự án > bảng điều khiển Android

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

Nếu chưa chọn một dự án nào, nhập vào tab **Project** trong cột tab dọc ở phía bên trái của cửa sổ Android Studio. Cửa sổ Project sẽ xuất hiện.

Để xem dự án trong phân cấp dự án Android chuẩn, hãy chọn **Android** từ hộp thoại menu bật lên phía trên của bảng Project, như hiển thị bên dưới.

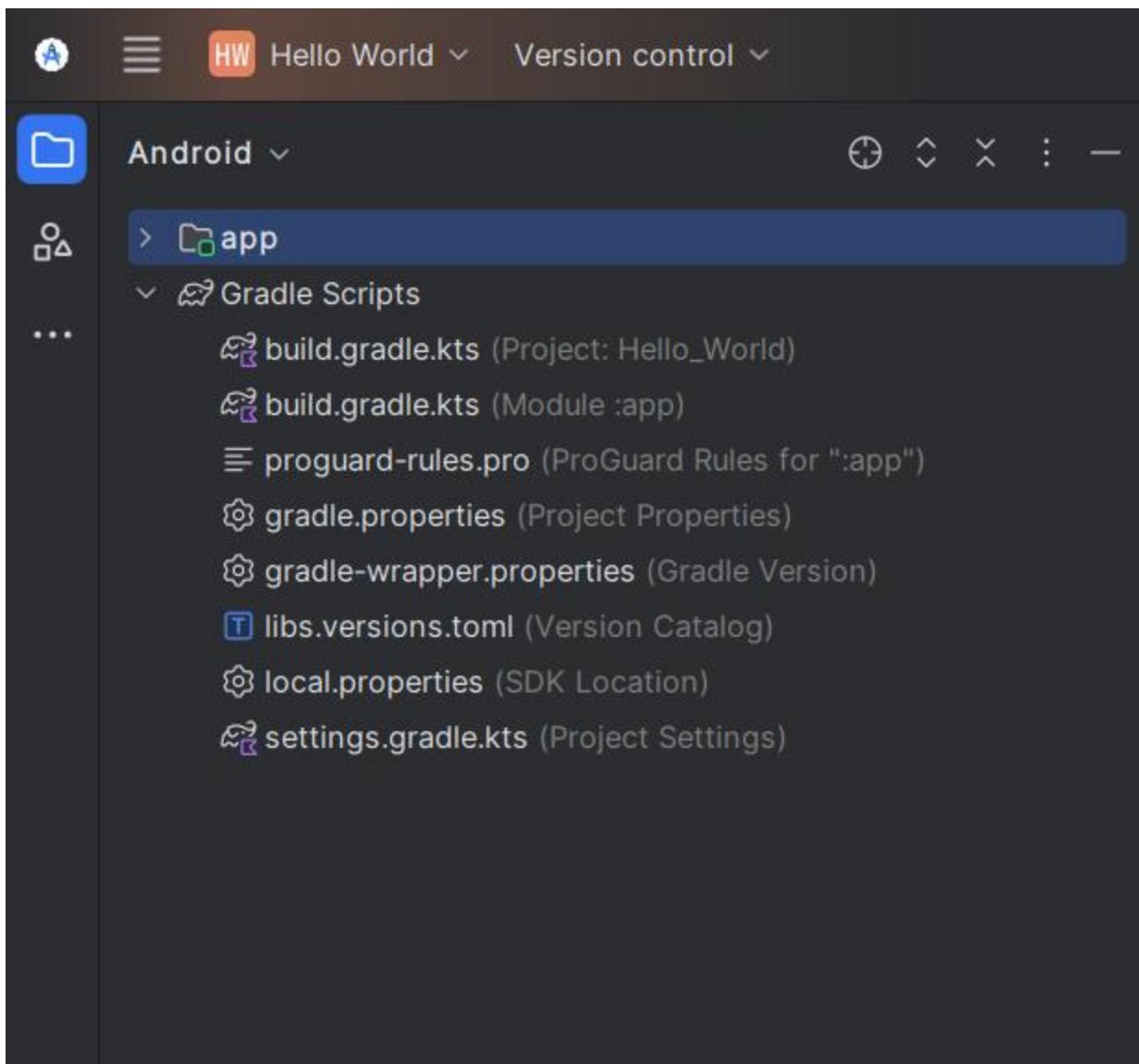


Chú ý: Trong chương này và các chương khác để cập nhật khung Project, khi cài đặt là **Android** như **Project > bảng Android**.

Bước 3: Khám phá thư mục Gradle Scripts

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

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



Thực hiện theo từng bước để khám phá hệ thống Gradle:

1. Nếu thư mục **Gradle Scripts** chưa được mở rộng, hãy nhấp vào hình tam giác để mở rộng nó.

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

2. Tìm tệp **build.gradle** (**Dự án: HelloWorld**).

Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô-đun tạo nên dự án của bạn. Mỗi dự án Android Studio đều chứa duy nhất một tệp xây dựng Gradle cấp cao. Hầu hết thời gian, bạn sẽ không cần phải thực hiện bất kỳ thay đổi nào đối với tệp này nhưng vẫn rất hữu ích để hiểu nội dung của nó.

Theo mặc định, tệp xây dựng cấp cao sử dụng khôi buildscript để định nghĩa các kho lưu trữ và phụ thuộc Gradle chung cho tất cả các mô-đun trong dự án. Khi các

yếu tố phụ thuộc của bạn là một thứ gì đó khác ngoài thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm kiếm các tệp trong các kh lưu trữ trực tuyến được chỉ định trong kh lưu trữ của tệp. Theo mặc định, các dự án Android Studio mới khai báo Jcenter và Google (bao gồm cả Google Maven repository) là các vị trí kho lưu trữ:

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

3. Tìm tệp build.gradle(Module:app).

Ngoài tệp build.gradle cấp dự án, mỗi mô-đun có một tệp build.gradle riêng, cho phép bạn cấu hình các cài đặt xây dựng cho từng mô-đun cụ thể (ứng dụng HelloWorld chỉ có một mô-đun). Việc cấu hình các cài đặt xây dựng này cho phép bạn cung cấp các tùy chọn đóng gói tùy chỉnh, chẳng hạn như bổ sung các kiểu xây dựng và các phiên bản của sản phẩm. Bạn cũng có thể ghi đè các cài đặt trong tệp AndroidManifest.xml hoặc tệp build.gradle cấp cao.

Tệp này thường là tệp cần chỉnh sửa khi thay đổi các cấu hình cấp ứng dụng, chẳng hạn như khai báo các yếu tố trong phần dependencies. Bạn có thể khai báo một thư viện phụ thuộc bằng nhiều cách cấu hình khác nhau. Mỗi cấu hình phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện. Ví dụ: câu lệnh implementation fileTree(dir: 'libs', include: ['*.jar']) thêm một phụ thuộc cho tất cả các tệp ".jar" bên trong thư mục libs.

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

```
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 {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
}
compileOptions {
    sourceCompatibility JavaVersion.VERSION_11
    targetCompatibility JavaVersion.VERSION_11
}
}

dependencies {

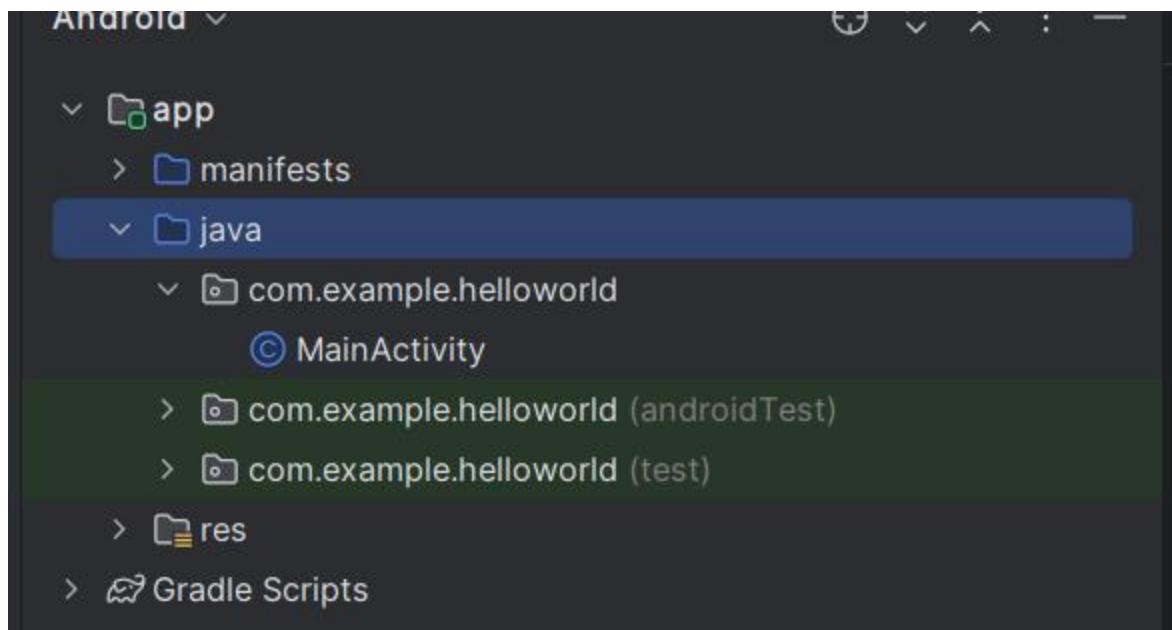
    implementation libs.appcompat
    implementation libs.material
    implementation libs.activity
    implementation libs.constraintlayout
    testImplementation libs.junit
    androidTestImplementation libs.ext.junit
    androidTestImplementation libs.espresso.core
}
```

- Nhập vào hình tam giác để đóng Gradle Scripts.

Bước 4: Khám phá thư mục app và thư mục res

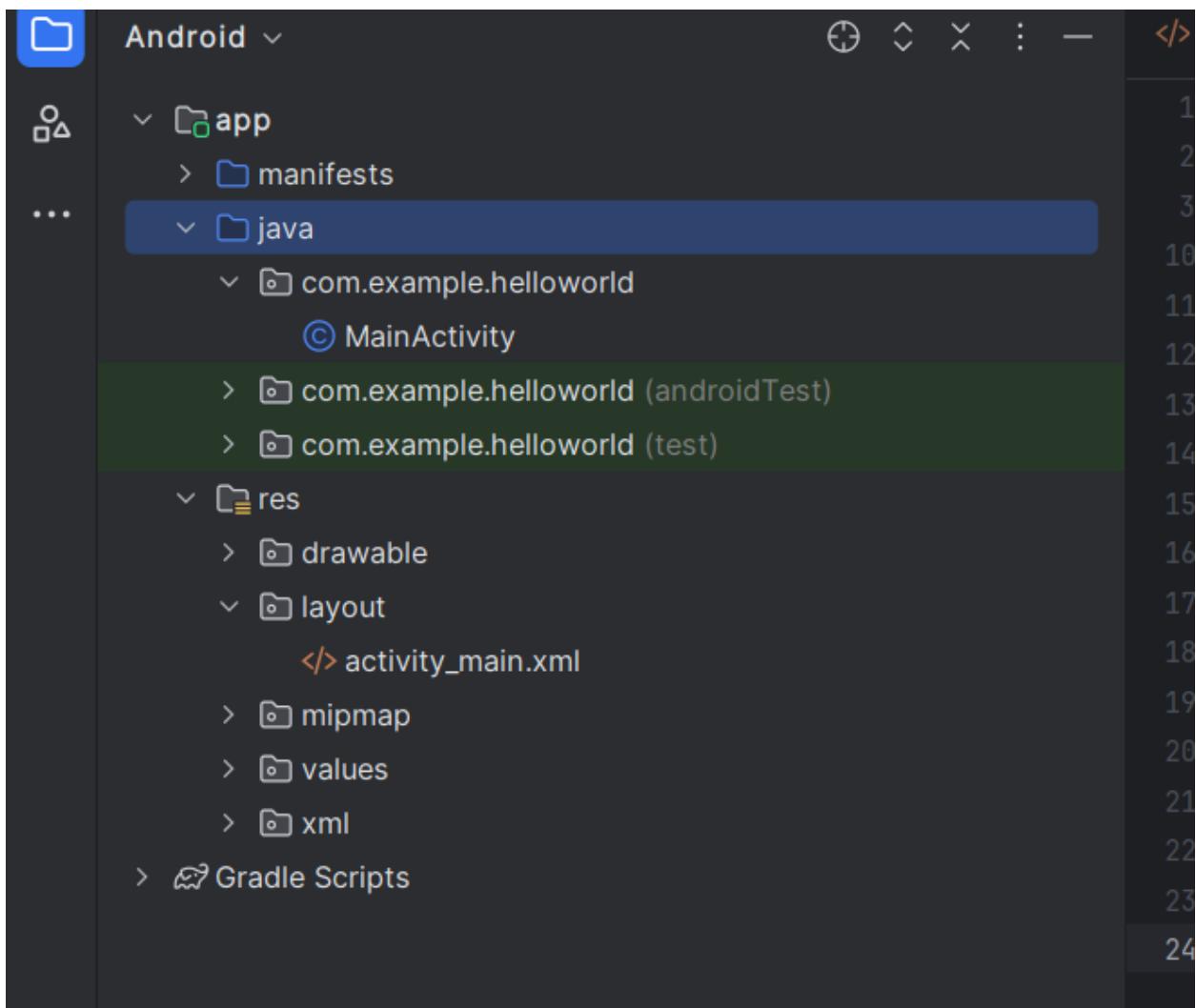
Tất cả mã và tài nguyên của ứng dụng đều nằm trong thư mục app và res.

- Mở rộng thư mục **app**, thư mục **java** và thư mục **com.example.android.helloworld** để xem tệp java **MainActivity**. Nhấp đúp vào tệp để mở tệp trong trình chỉnh sửa mã.



Thư mục **java** chứa các tệp lớp Java được tổ chức trong ba thư mục con như hình minh họa trong hình trên. Thư mục **com.example.android.helloworld** (hoặc tên miền đã chỉ định) chứa tất cả các tệp của 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 goid duy nhất và nó chứa tệp `MainActivity.java`. Đây là Activity (màn hình) đầu tiên mà người dùng nhìn thấy, đồng thời khởi tạo các tài nguyên chung cho ứng dụng, thường được đặt tên là **MainActivity** (phần mở rộng tệp bị ẩn trong bảng **Project > Android**).

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



Thư mục res chứa các tài nguyên, chẳng hạn như bộ cục, chuỗi và hình ảnh. Một Activity thường được liên kết với bộ cục của các chế độ xem giao diện người dùng được định nghĩa là tệp XML. Tệp này thường được đặt tên theo Activity của nó.

Bước 5: Khám phá thư mục manifests

Thư mục manifests chứa các file cung cấp thông tin cần thiết về ứng dụng của bạn cho hệ thống Android, mà hệ thống phải có trước khi có thể chạy bất kỳ mã nào của ứng dụng.

1. Mở rộng thư mục **manifests**.
2. Mở tệp **AndroidManifests.xml**.

Tệp AndroidManifests.xml mô tả tất cả các thành phần của ứng dụng Android của bạn. Tất cả các thành phần cho 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 sửa đổi tệp này để thêm các tính năng và quyền tính năng. Để đọc phần giới thiệu, hãy xem [App Manifest Overview](#).

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

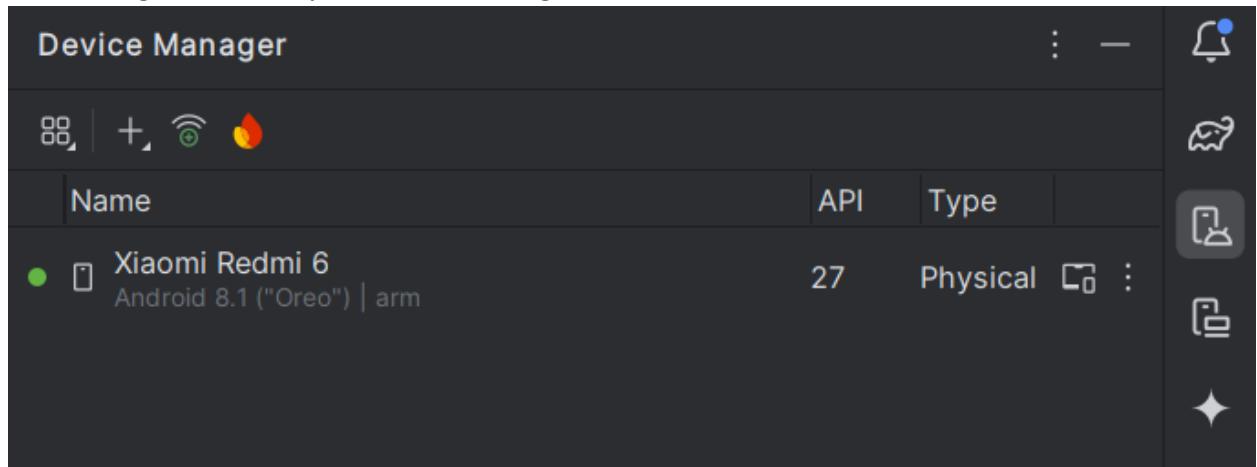
Trong nhiệm vụ này, bạn sẽ sử dụng Android Virtual Device (AVD) manager để tạo một thiết bị ảo (cũng có thể hiểu là trình giả lập) mô phỏng cấu hình cho một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng Android Emulator có các yêu cầu bổ sung ngoài các yêu cầu hệ thống cơ bản đối với Android Studio.

Sử dụng ADV Manager, bạn xác định các đặc điểm phần cứng của thiết bị, cấp độ API, bộ nhớ, giao diện và các thuộc tính khác của thiết bị và lưu dưới dạng thiết bị ảo. Với các thiết bị ảo, bạn có thể kiểm tra ứng dụng trên các cấu hình thiết bị khác nhau (như máy tính bảng và điện thoại) với các cấp độ API khác nhau mà không cần phải sử dụng thiết bị vật lý.

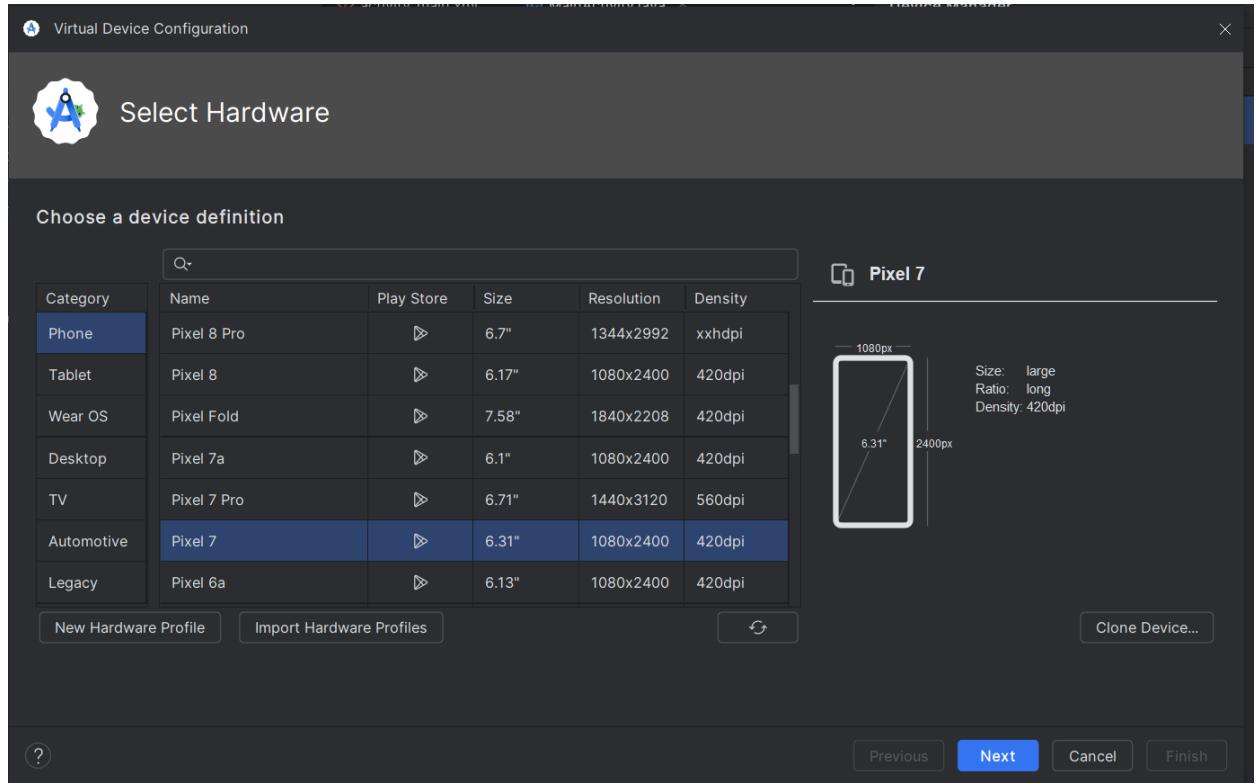
Bước 1: Tạo một thiết bị Android ảo (AVD)

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

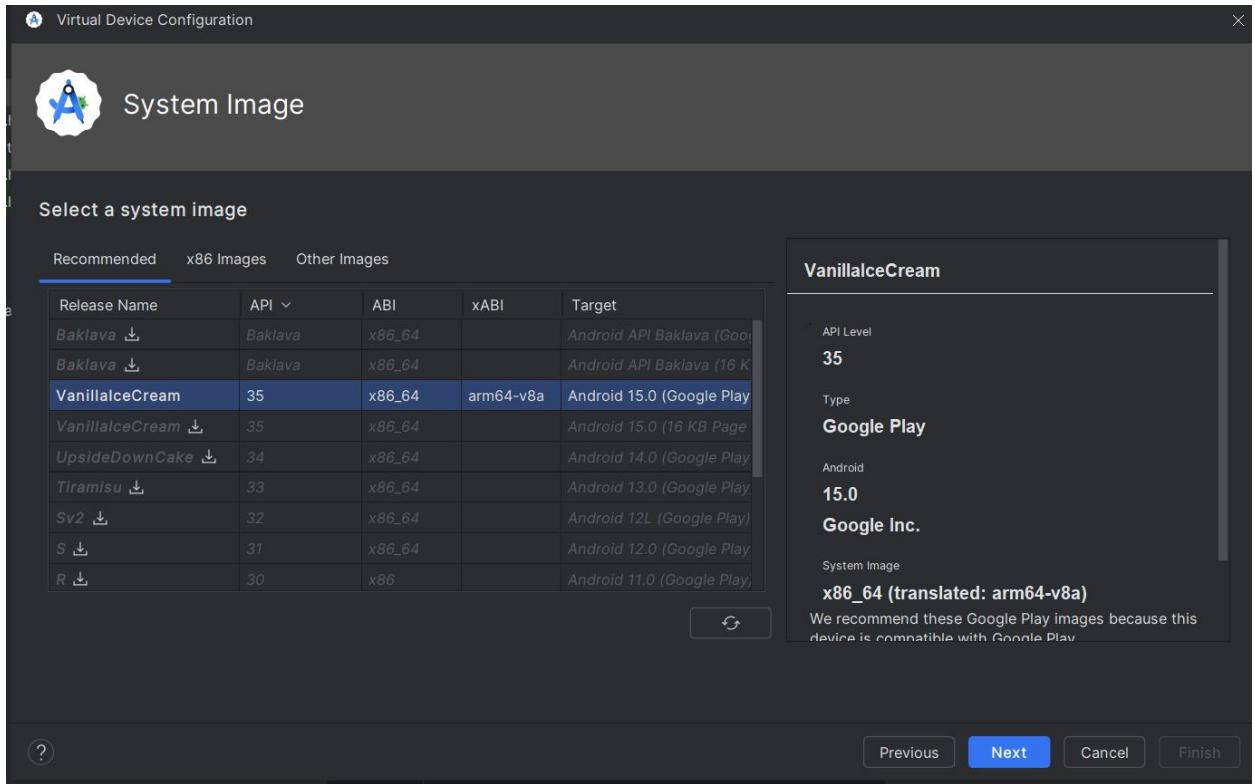
- Trong Android Studio, chọn **Tools > Android > AVD Manager** hoặc nhấp vào biểu tượng AVD Manager trên thanh công cụ. Màn hình Your Virtual Devices xuất hiện. Nếu bạn đã tạo thiết bị ảo, màn hình sẽ hiển thị chúng (như trong hình bên dưới); nếu không, bạn sẽ thấy danh sách trống.



- Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** sẽ 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 màn hình chéo (**Size**), độ phân giải màn hình tính bằng pixel (**Resolution**) và mật độ pixel (**Density**).



3. Chọn một thiết bị như **Nexus 5x** hoặc **Pixel XL** và nhấp vào **Next**. Màn hình **System Image** xuất hiện.
4. Nhấp vào tab **Recommended** nếu chưa chọn và chọn phiên bản hệ thống Android nào để chạy trên thiết bị ảo (như **VanillaleCream**).



Có nhiều phiên bản khả dụng hơn so với những phiên bản được hiển thị trong tab **Recommended**. Hãy xem các tab **x86 Image** và **Other Images** để xem chúng.

Nếu liên kết **Download** hiển thị bên cạnh ảnh hệ thống (system image) mà bạn muốn sử dụng, nếu ảnh đó vẫn chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống và nhấp vào **Finish** khi hoàn tất.

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

Bước 2: Chạy ứng dụng trên thiết bị ảo

Trong nhiệm vụ này, bạn sẽ hoàn tất việc chạy ứng dụng Hello World của bạn.

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

Trình giả lập khởi động và khởi động giống như một thiết bị vật lý. Tùy thuộc vào tốc độ máy tính của bạn, việc này có thể mất một lúc. Ứng dụng của bạn được xây dựng và khi trình giả lập đã sẵn sàng, Android Studio sẽ tải ứng dụng lên trình giả lập và chạy ứng dụng đó. Bạn sẽ thấy ứng dụng Hello World như trong hình sau.

Mẹo: Khi thử nghiệm trên thiết bị ảo, bạn nên khởi động thiết bị một lần, ngay khi bắt đầu phiên làm việc. Bạn không nên đóng thiết bị cho đến khi hoàn tất thử nghiệm ứng dụng, để ứng dụng không phải trải qua quá trình khởi động thiết bị một lần nữa. Để đóng thiết bị ảo, hãy nhấp vào nút X ở đầu trình giả lập, chọn **Quit** từ menu hoặc nhấn **Control-Q** trong Windows hoặc Command-Q trong macOS.

Nhiệm vụ 4: (Không bắt buộc) Sử dụng thiết bị vật lý

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

Những điều bạn cần có:

- Một thiết bị Android như điện thoại hoặc máy tính bảng.
- Một dây cáp truyền dữ liệu kết nối thiết bị Android của bạn với máy tính của bạn thông qua cổng USB.
- Nếu bạn đang sử dụng Hệ thống Linux hoặc Windows, bạn có thể cần thực hiện thêm các bước để chạy trên phần cứng thiết bị. 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 dựa trên Windows, hãy xem **OEM USB Drivers**.

Bước 1: Bật chế độ USB debugging

Để Android Studio kết nối với thiết bị của bạn, bạn phải bật chế độ USB Debugging trên thiết bị Android của bạn. Tùy chọn này được kích hoạt trong phần **Developer options** trên thiết bị của bạn.

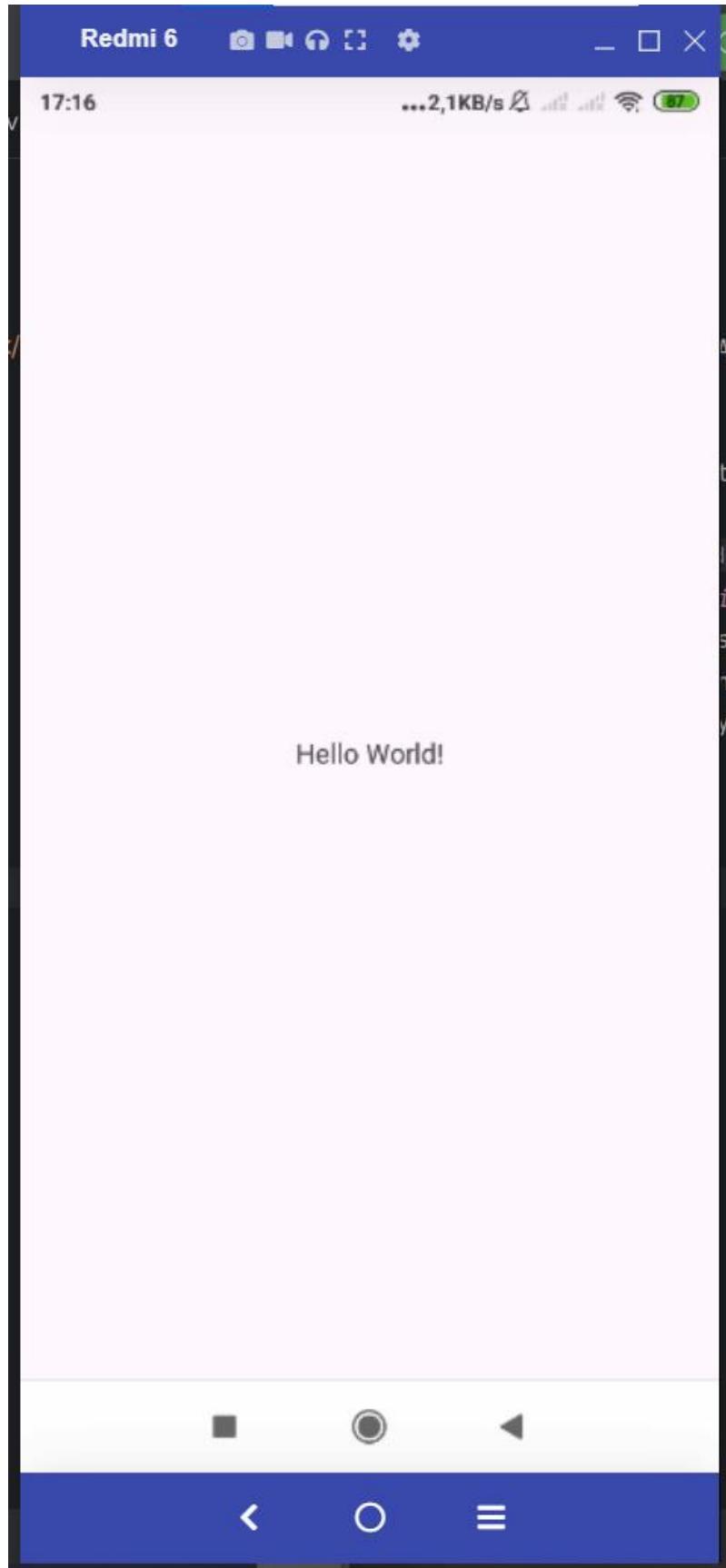
Trên Android 4.2 trở lên, màn hình **Developer options** bị ẩn theo mặc định. Để hiển thị tùy chọn nhà phát triển và bật USB Debugging, hãy làm theo các bước sau:

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

Bước 2: Chạy ứng dụng của bạn trên một 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 tính phát triển bằng cáp USB.
2. Nhấn vào nút **Run** trên thanh công cụ. Cửa sổ **Select Deployment Target** sẽ mở ra, hiển thị danh sách các trình giả lập và thiết bị đã kết nối.
3. Chọn thiết bị của bạn và nhấn **OK**.



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

Khắc phục sự cố:

Nếu Android Studio không nhận diện thiết bị của bạn, hãy thử:

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

Nếu máy tính vẫn không nhận diện thiết bị hoặc 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 có thông báo yêu cầu cấp quyền, hãy chấp nhận.

Bạn có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Xem thêm tại [Using Hardware Devices documentation](#).

Nhiệm vụ 5: Thay đổi cấu hình Gradle của ứng dụng

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

Bước 1: Thay đổi phiên bản tối thiểu SDK cho ứng dụng

Thực hiện theo các bước sau:

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

Nội dung của tệp sẽ hiển thị trong trình chỉnh sửa mã.

```

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"

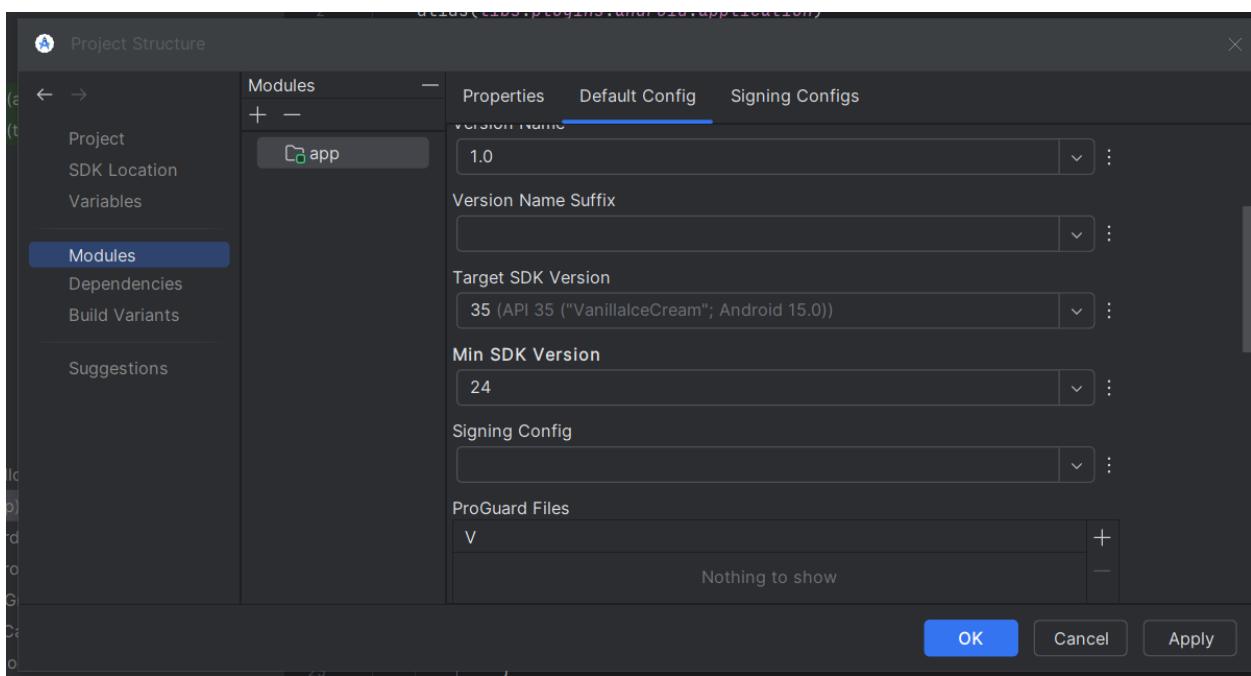
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}

```

2. Trong khối defaultConfig, thay đổi giá trị của minSdkVersion thành 24, như minh họa bên dưới (nếu giá trị ban đầu không được đặt là 24).



Trình chỉnh sửa mã hiển thị một thanh thông báo ở phía trên cùng với liên kết Sync Now(Open(Ctrl+Alt+Shift+S)).

Bước 2: Đồng bộ cấu hình Gradle mới

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

Để đồng bộ tệp dự án, hãy nhấp vào **Sync Now** trên thanh thông báo xuất hiện sau khi 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 tham khảo tài liệu [Build System Overview](#) và [Configuring Gradle Builds](#).

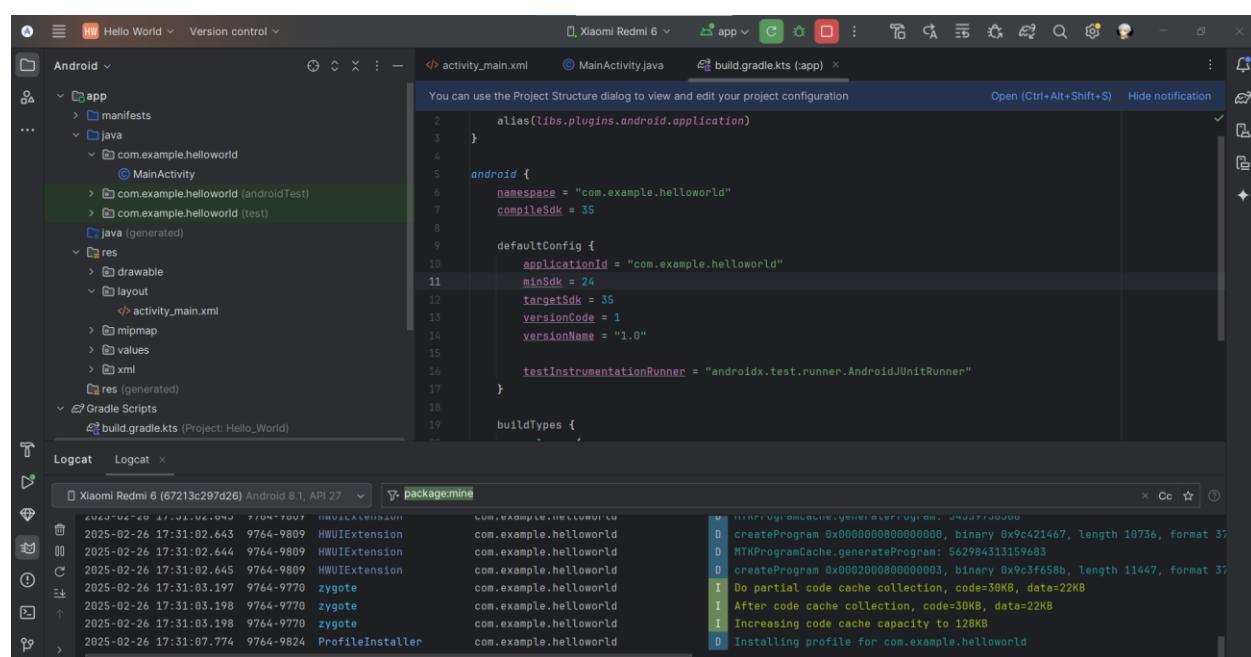
Nhiệm vụ 6: Thêm câu lệnh log vào ứng dụng của bạn

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

Các thông báo Log là một công cụ gỡ lỗi mạnh mẽ, cho phép bạn kiểm tra giá trị, theo dõi luồng thực thi và báo cáo các ngoại lệ.

Bước 1: Xem bảng Logcat

Để mở bảng **Logcat**, nhấp vào tab **Logcat** ở phía dưới cửa sổ Android Studio, như hình minh họa dưới đây:



Trong hình trên:

1. Tab **Logcat** dùng để mở và đóng bảng **Logcat**, nơi hiển thị thông tin về ứng dụng khi nó đang chạy. Nếu bạn thêm các câu lệnh Log vào ứng dụng, các thông báo Log sẽ xuất hiện tại đây.
2. Menu Log level được đặt ở chế độ **Verbose** (mặc định), hiển thị tất cả các thông báo Log. Các tùy chọn khác bao gồm **Debug**, **Error**, **Info** và **Warn**.

Bước 2: Thêm câu lệnh log vào ứng dụng của bạn

Các câu lệnh Log trong mã ứng dụng sẽ hiển thị thông báo trong bảng Logcat. Ví dụ:

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

Các thành phần của thông báo log:

- Log: Lớp Log dùng để gửi thông báo log đến bảng Logcat
- d: Mức log **Debug** để lọc và hiển thị thông báo trong bảng Logcat. Các mức log khác bao gồm: e cho **Error**, w cho **Warn** và i cho **Info**.
- “MainActivity”: Đối số đầu tiên là một tag (nhãn), có thể được sử dụng để lọc thông báo trong bảng Logcat. Thông thường, tag này là tên của Activity nơi thông báo được ghi lại. Tuy nhiên, bạn có thể đặt bất kỳ giá trị nào hữu ích cho quá trình gỡ lỗi.

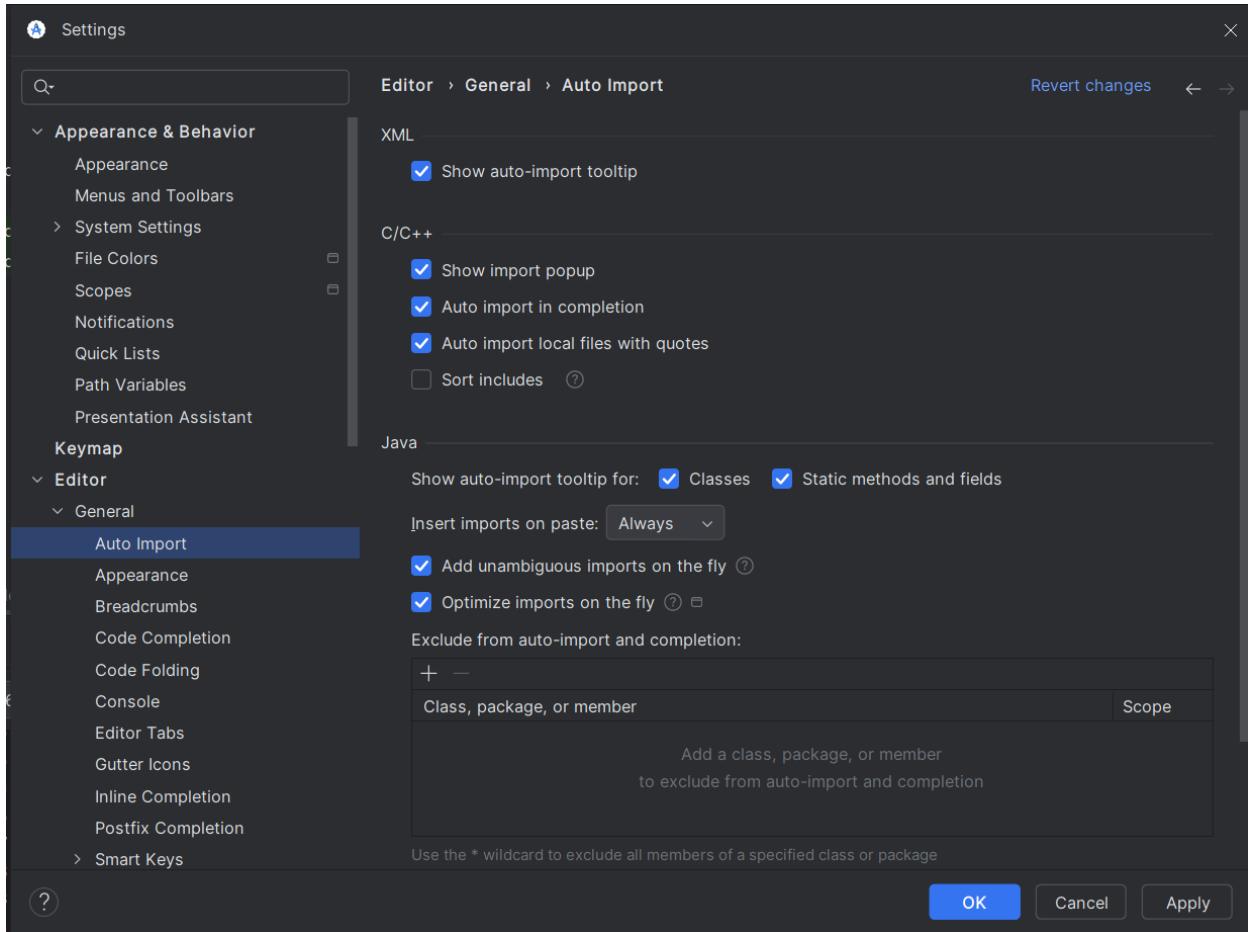
Theo quy ước, các tag log thường được định nghĩa dưới dạng hằng số trong Activity.

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

- “Hello world”: Đối số thứ hai là thông điệp thực tế.

Thực hiện theo các bước sau:

1. Mở ứng dụng Hello World trong Android Studio và mở tệp MainActivity.
2. Để tự động thêm các thư viện import cần thiết vào dự án (chẳng hạn như android.util.Log để sử dụng Log), thực hiện: với Windows: Vào **File > Settings** và với macOS: Vào **Android Studio > Preferences**.
3. Chọn **Editor > General > Auto Import**. Tích chọn tất cả các ô và đặt **Insert imports on paste** thành All.
4. Nhấn **Apply**, sau đó nhấn **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 đoạn mã sau:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable(this);  
    setContentView(R.layout.activity_main);  
    Log.d("MainActivity", "Hello World");  
    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;  
    });  
}
```

```
});  
}
```

6. Nếu bảng Logcat chưa mở, hãy nhấp vào tab **Logcat** ở phía dưới Android Studio để mở.
7. Kiểm tra xem tên mục tiêu (target) và tên gói (package name) của ứng dụng có chính xác không.
8. Thay đổi mức **Log** trong bảng **Logcat** thành **Debug** (hoặc giữ nguyên **Verbose** nếu có ít thông báo log).
9. Chạy ứng dụng của bạn.

Thông báo sau sẽ xuất hiện trong bảng Logcat:



2025-02-26 17:35:29.279 10207-10207 MainActivity I com.example.helloworld D Hello World

Thử thách lập trình

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

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

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

Tóm tắt

- Để cài đặt Android Studio, truy cập trang Android Studio và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo ứng dụng mới, đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm Minimum SDK.

- Để xem cấu trúc phân cấp của ứng dụng trong Project pane, nhấp vào tab **Project** trong cột tab dọc, sau đó chọn **Android** trong menu bật lên ở trên cùng.
- Chính sửa tệp build.gradle(Module:app) khi bạn 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 res. Thư mục java chứa các Activity, bài kiểm tra và các thành phần khác được viết bằng mã nguồn Java. Thư mục res chứa tài nguyên như bố cục (layouts), chuỗi ký tự (strings), và hình ảnh (images).
- Chính sửa tệp AndroidManifest.xml để thêm các thành phần tính năng và quyền (permissions) cho ứng dụng Android. Tất cả các thành phần của ứng dụng, chẳng hạn như nhiều Activity, phải được khai báo trong tệp XML này.
- Sử dụng Android Virtual Device (AVD) Manager để tạo một thiết bị ảo (emulator) 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 gỡ lỗi cơ bản.
- Để chạy ứng dụng trên thiết bị Android thực tế bằng Android Studio, hãy bật USB Debugging trên thiết bị: Mở **Cài đặt > Giới thiệu điện thoại (Settings > About phone)**, sau đó nhấn vào **Số bản dựng (Build number)** bảy lần. Quay lại màn hình trước (**Cài đặt**), nhấn vào **Tùy chọn nhà phát triển (Developer options)**, sau đó bật **Gỡ lỗi USB (USB Debugging)**.

Các khái niệm liên quan

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

Tìm hiểu thêm

Tài liệu về Android Studio:

Khác:

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 log với các cấp độ khác nhau trong phương thức `onCreate()` của `MainActivity`.
- Tạo một thiết bị giả lập (emulator) với phiên bản Android tùy chọn và chạy ứng dụng.
- Sử dụng bộ lọc trong **Logcat** để tìm các câu lệnh log của bạn và điều chỉnh mức hiển thị chỉ bao gồm debug hoặc error.

Trả lời các câu hỏi

Câu 1: Tên của tệp tin layout cho main activity là gì ?

- `MainActivity.java`
- `AndroidManifest.xml`
- **activity_main.xml**
- `build.gradle`

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

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

Câu 3: Bạn sử dụng công cụ nào để tạo một trình giả lập (emulator) mới ?

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

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

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

Bạn sẽ thấy dòng log “`MainActivity layout is complete`” trong Logcat nếu mức Log được đặt thành tùy chọn nào sau đây? (Gợi ý: Có thể có nhiều đáp án đúng.)

- **Verbose**
- `Debug`
- **Info**
- `Warn`
- `Error`

- Assert

Nội dung của bạn để chấm điểm

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

- Một Activity hiển thị “Hello World” trên màn hình.
- Câu lệnh log trong phương thức onCreate() của MainActivity.
- Mức Log trong **Logcat** chỉ hiển thị lệnh debug hoặc error.

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là *views* – mọi phần tử trên màn hình đều là một View. Lớp View đại diện cho khái niệm cơ bản của tất cả các thành phần UI và là lớp cơ sở cho các lớp cung cấp các thành phần UI tương tác như button, checkbox và trường nhập văn bản. Các lớp con View thường được sử dụng, được mô tả trong nhiều bài học, bao gồm:

- TextView để hiển thị văn bản.
- EditText để cho phép người dùng nhập và chỉnh sửa văn bản.
- Button và các phần tử có thể nhấp khác (chẳng hạn như RadioButton, CheckBox và Spinner) để cung cấp hành vi tương tác.
- ScrollView và RecyclerView để hiển thị các mục có thể cuộn.
- ImageView để hiển thị hình ảnh.
- ConstraintLayout và LinearLayout để chứa các phần tử View khác và định vị chúng.

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 bộ cục UI được định nghĩa trong một tệp XML (eXtended Markup Language). Tệp XML này thường được đặt tên theo Activity của nó và xác định cách bố trí các phần tử View trên màn hình.

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị một bộ cục được định nghĩa trong tệp layout activity_main.xml, trong đó bao gồm một TextView hiển thị 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 chạm của người dùng, vẽ nội dung đồ họa hoặc yêu cầu dữ liệu từ cơ sở dữ liệu hoặc internet. Bạn sẽ tìm hiểu thêm về lớp Activity trong một bài học khác.

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

Những gì bạn nên biết

Bạn cần 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 trình chỉnh sửa bố cục (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 tìm hiểu các định nghĩa một cách dễ hiểu.

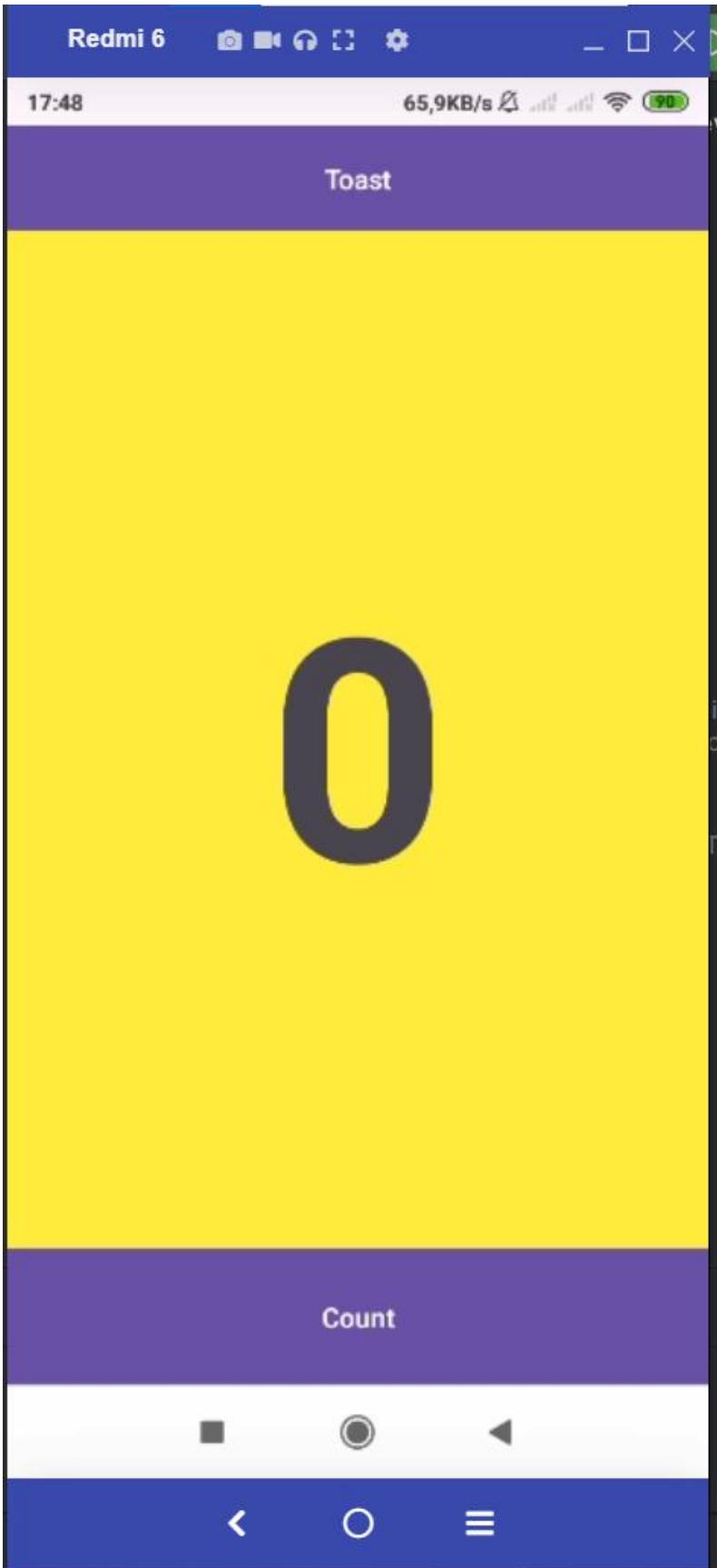
Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

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

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



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

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

Bước 1: Tạo dự án Android Studio

14. Chạy 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	API15: Android 4.0.3 IceCreamSandwich
Template	Empty Activity
Generate Layout file box	Selected
Backwards Compatibility box	Selected

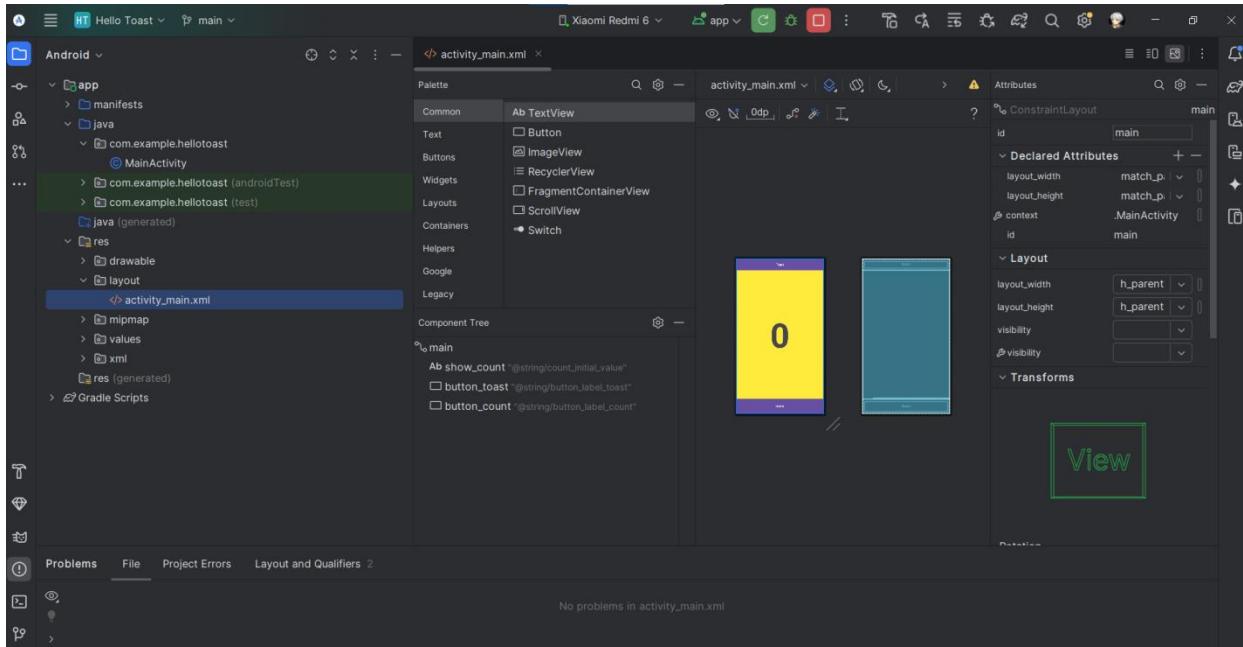
15. Chọn **Run > Run app** hoặc nhập vào **biểu tượng Run** trên thanh công cụ để biên dịch và chạy ứng dụng trên trình giả lập hoặc thiết bị của bạn.

Bước 2: Khám phá trình chỉnh sửa bố cục

Android Studio cung cấp trình chỉnh sửa bố cục (layout editor) để xây dựng nhanh chóng bố cục giao diện người dùng (UI) của ứng dụng. Công cụ này cho phép bạn kéo các phần tử vào chế độ xem thiết kế trực quan và bản thiết kế, định vị chúng trong bố cục, thêm ràng buộc (constraints) và thiết lập thuộc tính.

Constraints xác định vị trí của một phần tử UI trong bố cục. Một ràng buộc đại diện cho một kết nối hoặc căn chỉnh với một View khác, bố cục cha, hoặc một đường hướng dẫn vô hình.

Khám phá trình chỉnh sửa bố cục và tham khảo hình bên dưới khi bạn thực hiện theo các bước được đánh số:



- Trong bảng điều hướng Project > **Android**, điều hướng đến thư mục **app > res > layout**, sau đó nhấp đúp vào tệp **activity_main.xml** để mở, nếu nó chưa được mở.
- Nhập vào tab Design nếu nó chưa được chọn. Bạn sử dụng tab Design để thao tác các phần tử và bố cục, còn tab Text để chỉnh sửa mã XML của bố cục.
- Ngăn Palettes hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục của ứng dụng.
- Ngăn Component Tree hiển thị hệ thống phân cấp của các phần tử UI. Các phần tử View được tổ chức theo một cây phân cấp với quan hệ cha - con, trong đó phần tử con kế thừa thuộc tính từ phần tử cha. Trong hình trên, TextView là phần tử con của ConstraintLayout. Bạn sẽ tìm hiểu thêm về các phần tử này trong bài học này.
- Các ngăn thiết kế và bản thiết kế của trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình trên, bố cục chỉ chứa một phần tử: một TextView hiển thị dòng chữ "Hello World".
- Tab Attributes hiển thị ngăn Attributes để thiết lập các thuộc tính cho một phần tử UI.

Mẹo: Xem tài liệu 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 tài liệu đầy đủ về Android Studio.

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

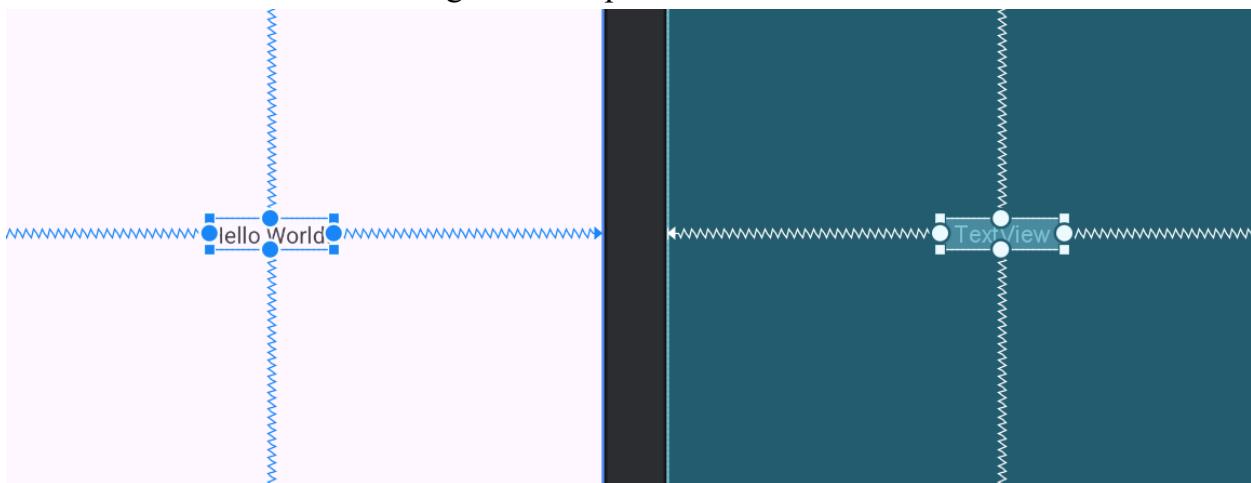
Trong nhiệm vụ này, bạn sẽ tạo bố cục giao diện người dùng cho ứng dụng HelloToast bằng cách sử dụng các tính năng của ConstraintLayout trong trình chỉnh sửa bố cục. Bạn

có thể tạo ràng buộc (constraints) theo cách thủ công như hướng dẫn bên dưới hoặc tự động bằng công cụ Autoconnect.

Bước 1: Kiểm tra ràng buộc của phần tử

Thực hiện các bước sau:

1. Mở tệp activity_main.xml từ bảng điều hướng **Project > Android** nếu nó chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào nó.
Nếu không có blueprint, hãy nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**.
2. Công cụ **Autoconnect** cũng nằm trên thanh công cụ và được bật theo mặc định. Đảm bảo rằng công cụ này không bị tắt.
3. Nhấp vào nút **Zoom In** để phóng to chế độ xem thiết kế và bản thiết kế để quan sát rõ hơn.
4. Chọn **TextView** trong ngăn Component Tree. Phần tử TextView có nội dung "Hello World" sẽ được tô sáng trong các ngăn thiết kế và bản thiết kế, đồng thời các ràng buộc của phần tử sẽ hiển thị.
5. Làm theo hướng dẫn trong hình động bên dưới. Nhấp vào tay cầm hình tròn ở bên phải của TextView để xóa ràng buộc ngang đang gắn phần tử này vào cạnh phải của bố cục. Khi ràng buộc này bị xóa, TextView sẽ nhảy sang bên trái vì nó không còn bị ràng buộc với cạnh phải nữa. Để thêm lại ràng buộc ngang, nhấp vào cùng tay cầm đó và kéo một đường đến cạnh phải của bố cục.



Trong ngăn bản thiết kế (blueprint) hoặc thiết kế (design), các tay cầm sau xuất hiện trên phần tử TextView:

- **Constraint handle:** Để tạo một ràng buộc như trong hình động trên, hãy nhấp vào tay cầm ràng buộc (Constraint handle), 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 ziczac sẽ đại diện cho ràng buộc.



- **Resizing handle** (Tay cầm thay đổi kích thước): Để thay đổi kích thước của phần tử, hãy kéo các tay cầm hình vuông. Khi kéo, tay cầm sẽ thay đổi thành một góc xiên.

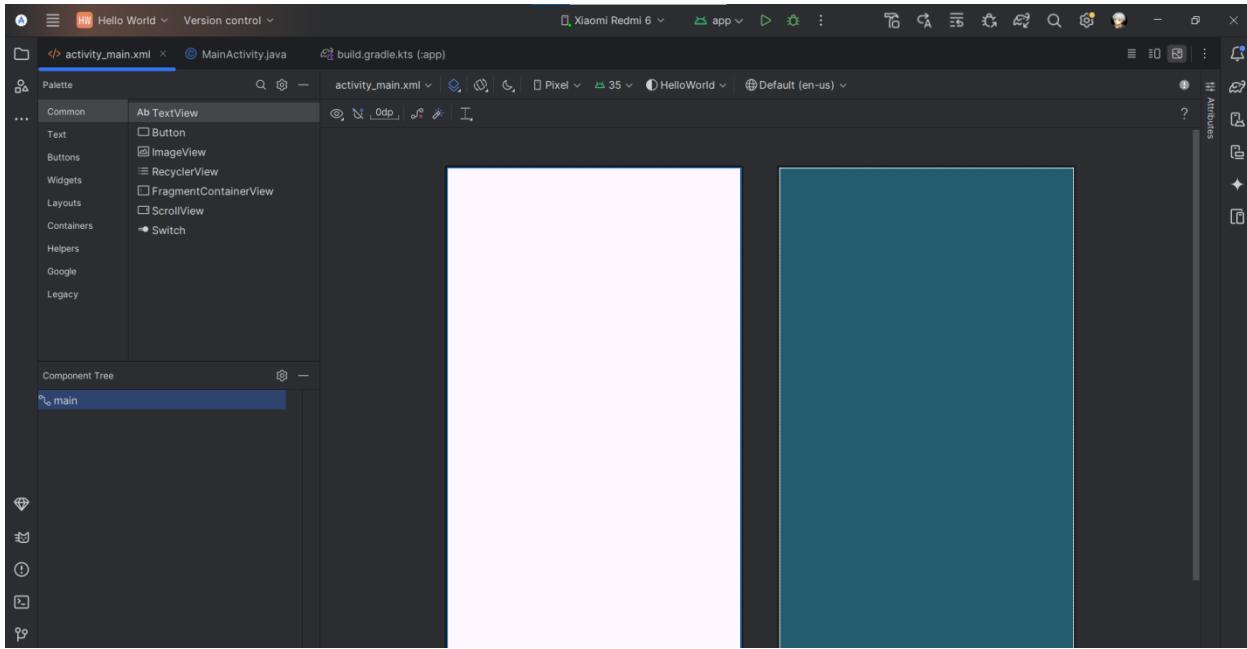


Bước 2: Thêm một Button vào bố cục

Khi được bật, công cụ **Autoconnect** sẽ tự động tạo hai hoặc nhiều ràng buộc (constraints) cho một phần tử UI với bố cục cha. Sau khi bạn kéo phần tử vào bố cục, công cụ này sẽ tạo các ràng buộc dựa trên vị trí của phần tử.

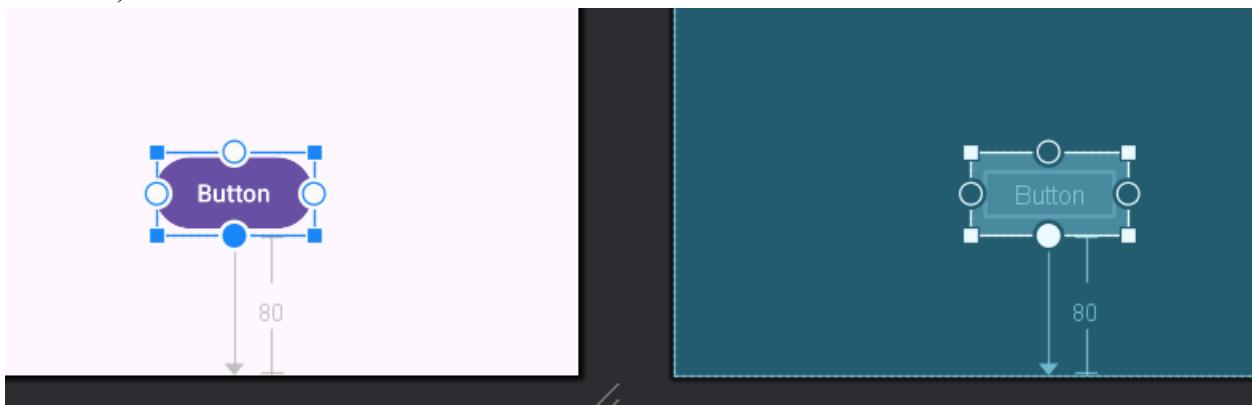
Thực hiện các bước sau để thêm một Button:

1. Bắt đầu với việc xóa bố cục cũ. Phần tử TextView không còn cần thiết, vì vậy khi nó vẫn đang được chọn, nhấn phím **Delete** hoặc chọn **Edit > Delete**. Lúc này, bố cục sẽ hoàn toàn trống.
2. Kéo một Button từ ngăn Palette vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả Button vào khu vực giữa phía trên của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc đến cạnh trên, cạnh trái và cạnh phải của bố cục như minh họa trong hình động bên dưới.



Bước 3: Thêm một Button thứ hai vào bố cục

1. Kéo thêm một Button từ bảng điều hướng Palette vào giữa bố cục như hình minh họa hình động bên dưới. Autoconnect có thể tự động tạo ràng buộc ngang cho bạn. Nếu không, bạn có thể tự kéo ràng buộc ngang.
2. Kéo một ràng buộc dọc từ Button đến cạnh dưới của bố cục (tham khảo hình bên 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 để hiển thị nút **Clear Constraints**. Nhấp vào nút này để xóa tất cả ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc cụ thể, hãy nhấp vào tay cầm thiết lập ràng buộc đó.

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

Nhiệm vụ 3: Thay đổi các thuộc tính của phần tử UI

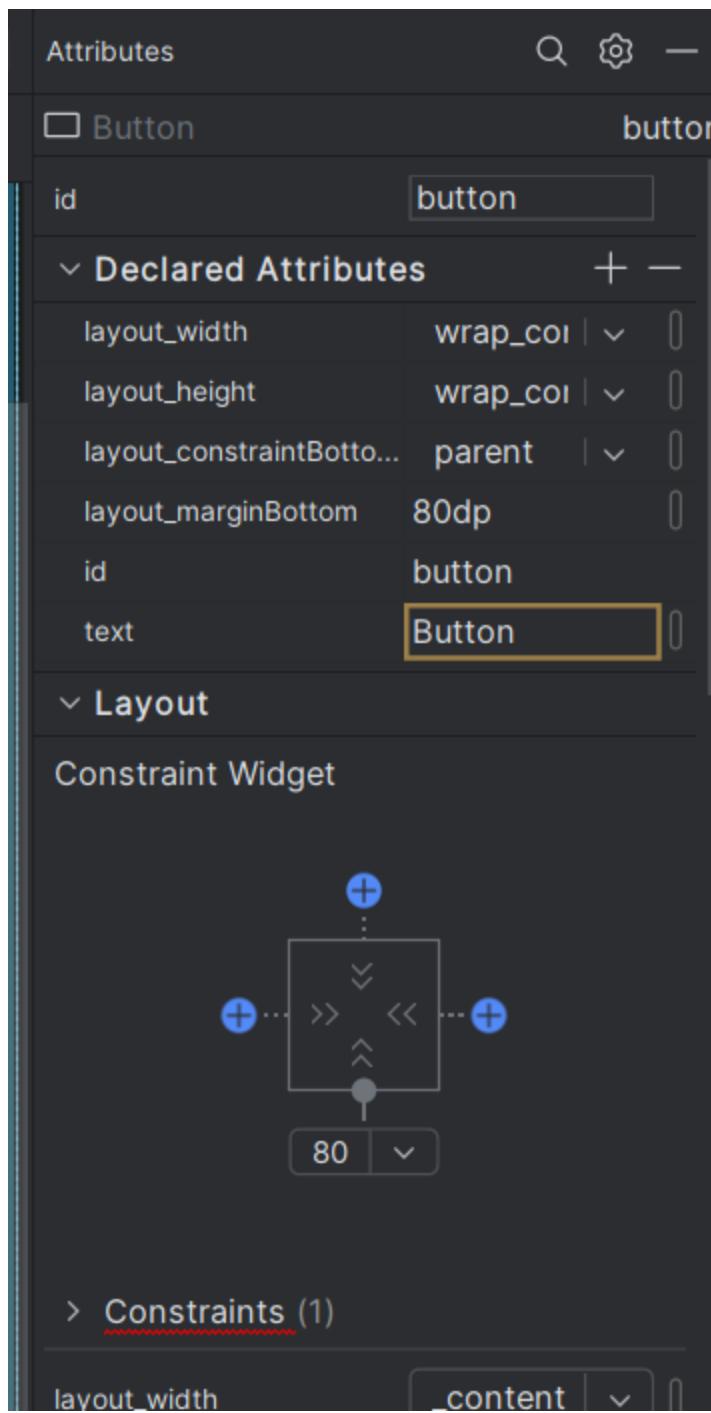
Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Bạn có thể tìm thấy các thuộc tính (được gọi là properties) chung cho tất cả các View trong [View class documentation](#).

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 áp dụng cho hầu hết các loại View.

Bước 1: Thay đổi kích thước Button

Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước (**Resizing handle**) ở cả bốn góc của một View, giúp bạn có thể thay đổi kích thước View một cách nhanh chóng. Bạn có thể kéo các tay cầm ở mỗi góc của View để thay đổi kích thước của nó, nhưng làm như vậy sẽ đặt kích thước chiều rộng và chiều cao dưới dạng giá trị cố định (hardcoded). Hạn chế sử dụng kích thước cố định cho hầu hết các phần tử View, vì các kích thước này không thể thích ứng với nội dung và kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn **Attributes** ở bên phải của trình chỉnh sửa bố cục để chọn một chế độ kích thước không sử dụng giá trị cố định. Ngăn **Attributes** bao gồm một bảng kích thước hình vuông, được gọi là *view inspector*, nằm ở phía trên. Các ký hiệu bên trong hình vuông này đại diện cho các cài đặt chiều cao và chiều rộng như sau:



Trong hình trên:

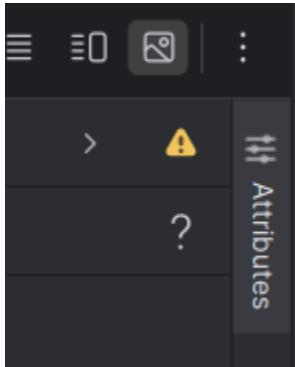
1. **Height control** (Điều khiển chiều cao): Điều khiển này xác định thuộc tính `layout_height` và xuất hiện dưới dạng hai đoạn trên các cạnh trên và dưới của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành `wrap_content`, có

nghĩa là View sẽ mở rộng theo chiều dọc khi cần thiết để phù hợp với nội dung của nó. Số "8" biểu thị một lề tiêu chuẩn được đặt thành 8dp.

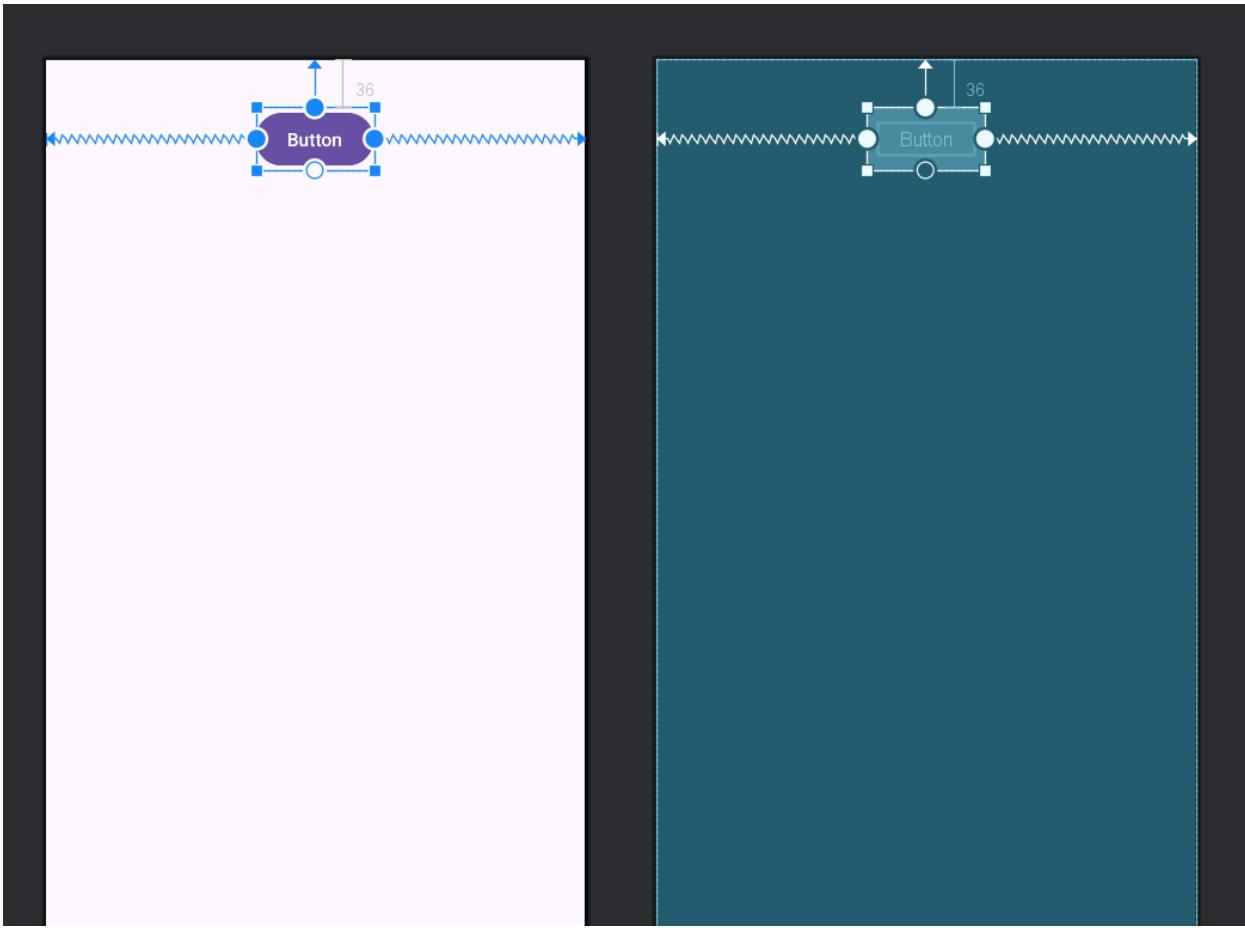
2. **Width control** (Điều khiển chiều rộng): Điều khiển này xác định thuộc tính layout_width và xuất hiện dưới dạng hai đoạn trên các cạnh trái và phải của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành wrap_content, điều này có nghĩa là View sẽ mở rộng theo chiều ngang khi cần thiết để phù hợp với nội dung của nó, tối đa đến một lề 8dp.
3. Nút đóng ngăn **Attributes**: Nhấp vào để đóng ngăn

Thực hiện các bước sau:

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

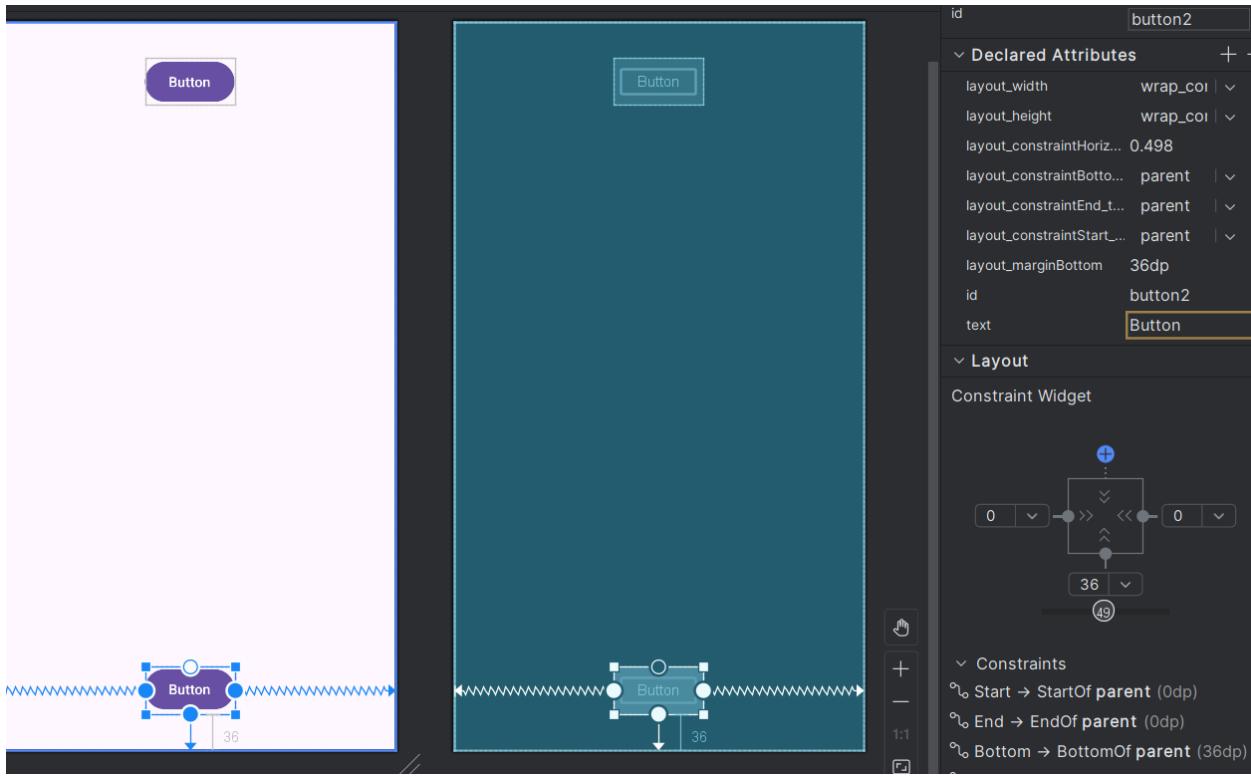


3. Nhấp vào điều khiển chiều rộng hai lần - Lần nhấp đầu tiên thay đổi nó thành **Fixed** với các đường thẳng và lần nhấp thứ hai thay đổi nó thành **Match Constraints** với các lò xo, như minh họa trong hình động bên dưới.



Do thay đổi kết quả điều khiển chiều rộng, thuộc tính layout_width trong ngăn **Attributes** hiển thị giá trị match_constraint, và phần tử Button mở rộng theo chiều ngang để lấp đầy không gian giữa cạnh trái và cạnh phải của bố cục.

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



Như đã thấy trong các bước trước, các thuộc tính `layout_width` và `layout_height` trong ngăn **Attributes** thay đổi khi bạn điều chỉnh các điều khiển chiều rộng và chiều cao trong inspector. Các thuộc tính này có thể nhận một trong ba giá trị trong bộ cục `ConstraintLayout`:

- Giá trị `match_constraint` mở rộng phần tử View để lấp đầy không gian của parent theo chiều rộng (width) hoặc chiều cao tối đa (height-up) đến một lề nếu có lề được đặt. Trong trường hợp này, parent là `ConstraintLayout`. Bạn sẽ tìm hiểu thêm về `ConstraintLayout` trong nhiệm vụ tiếp theo.
- Giá trị `wrap_content` thu nhỏ kích thước của phần tử View sao cho vừa đủ để bao bọc nội dung của nó. Nếu không có nội dung, phần tử View sẽ trở nên vô hình.
- Kích thước cố định để chỉ định một kích thước cố định có thể điều chỉnh theo kích thước màn hình của thiết bị, sử dụng một số cố định với đơn vị density-independent pixels (dp units). Ví dụ, 16dp có nghĩa là 16 pixel độc lập với mật độ màn hình.

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

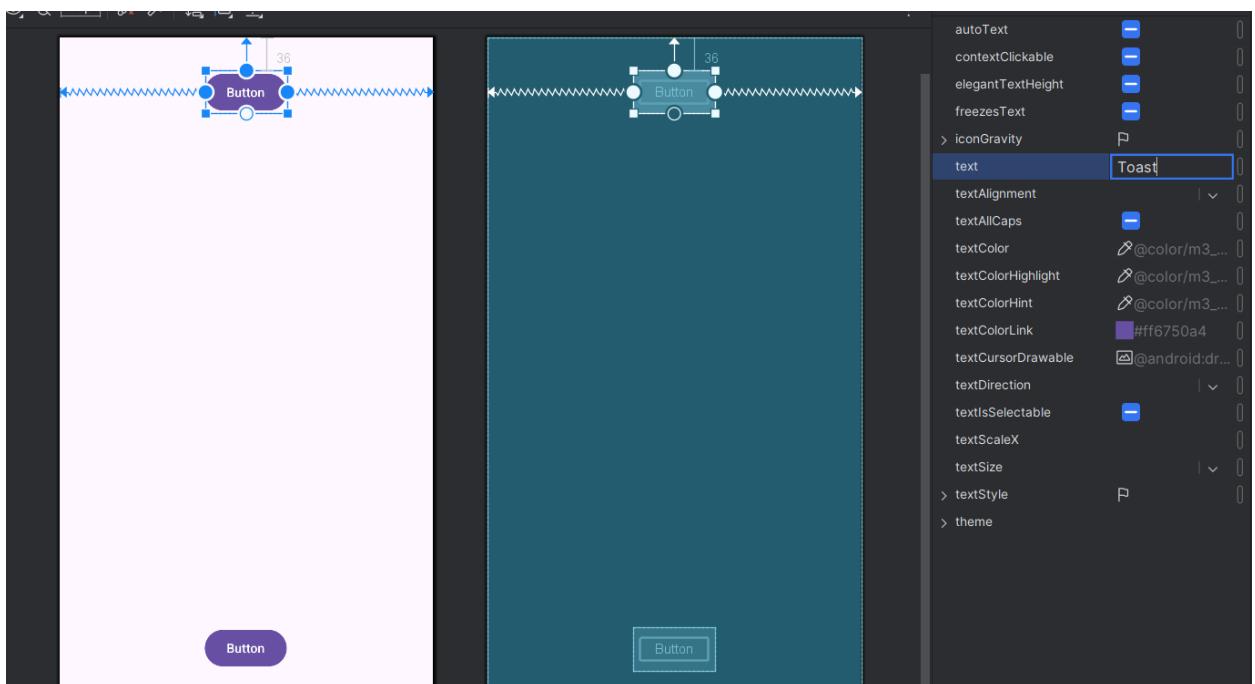
Bước 2: Thay đổi các thuộc tính của Button

Để xác định duy nhất mỗi View trong bố cục của một Activity, mỗi View hoặc lớp con của View (chẳng hạn như Button) cần có một ID duy nhất. Ngoài ra, để có thể sử dụng được, các phần tử Button cần có văn bản. Các phần tử View cũng có thể có nền, có thể là màu 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 sau:

1. Sau khi chọn Button đầu tiên, chỉnh sửa trường ID ở đầu ngăn **Attributes** thành **button_toast** cho thuộc tính android:id, thuộc tính này dùng 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 bạn dễ dàng chọn.)
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 cùng màu nền cũng như màu chữ như các bước trước đó.

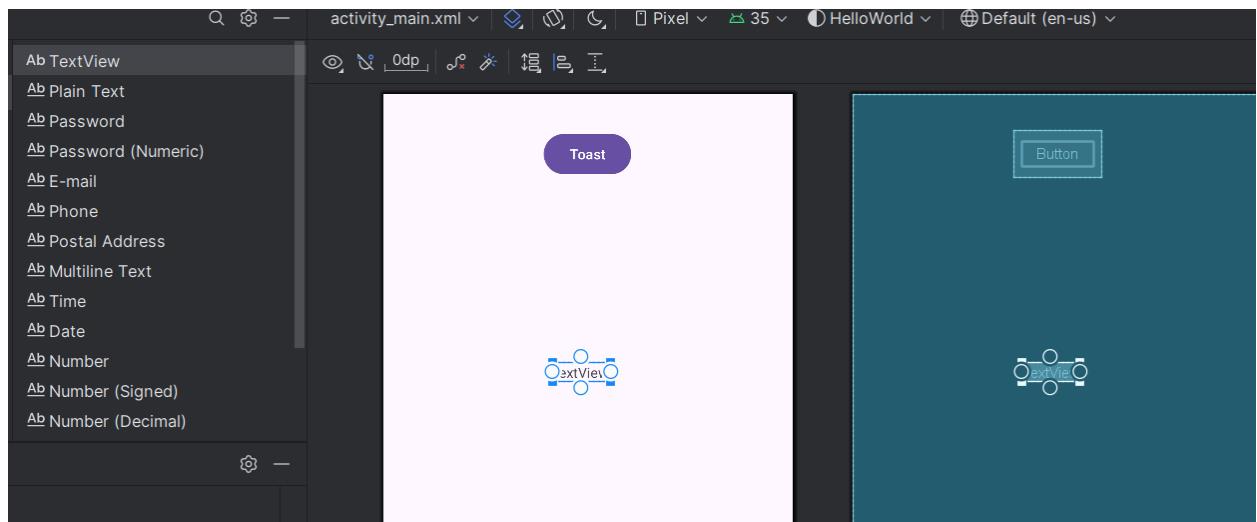
colorPrimary là màu chính của chủ đề, một trong các màu cơ bản được định nghĩa sẵn trong tệp tài nguyên colors.xml. Màu này được sử dụng cho thanh ứng dụng (app bar). Sử dụng các màu cơ bản này cho các phần tử UI khác giúp tạo ra một giao diện thống nhất. Bạn sẽ tìm hiểu thêm về chủ đề ứng dụng và Material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

Nhiệm vụ 4: Thêm một EditText và thiết lập các ràng buộc của nó

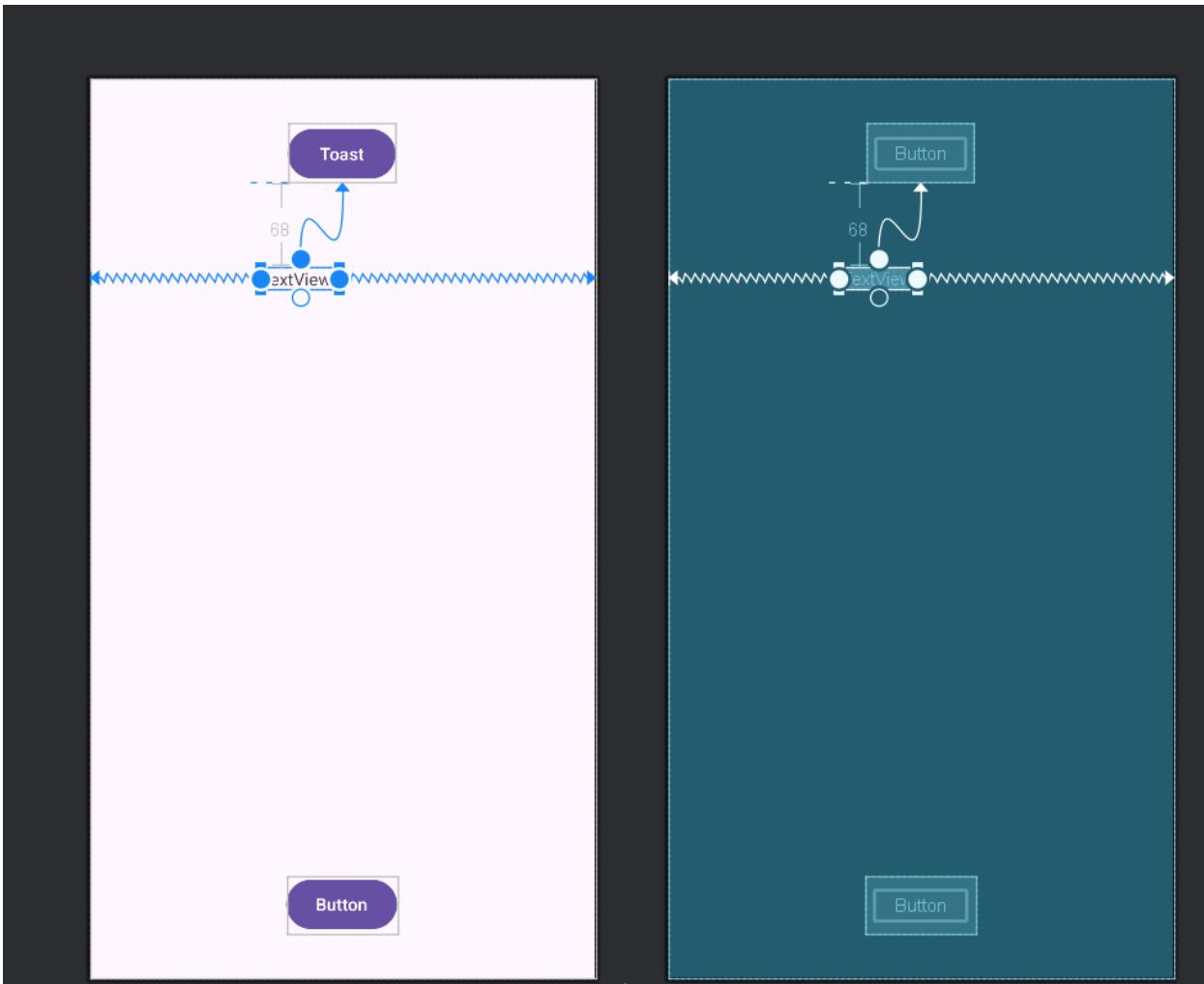
Một trong những lợi ích của ConstraintLayout là khả năng căn chỉnh hoặc ràng buộc các phần tử dựa trên các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một TextView vào giữa bố cục và ràng buộc nó theo chiều ngang với lề, theo chiều dọc với hai Button. Sau đó, bạn sẽ thay đổi các thuộc tính cho TextView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

Bước 1: Thêm một TextView và các ràng buộc

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



2. Như minh họa trong hình động bên dưới, kéo một ràng buộc từ phía dưới của TextView đến tay cầm ở phía trên của nút Count Button, và kéo các ràng buộc từ hai bên của TextView đến hai bên của bố cục. Điều này ràng buộc TextView nằm ở giữa bố cục, giữa hai Button.



Bước 2: Thiết lập các thuộc tính của TextView

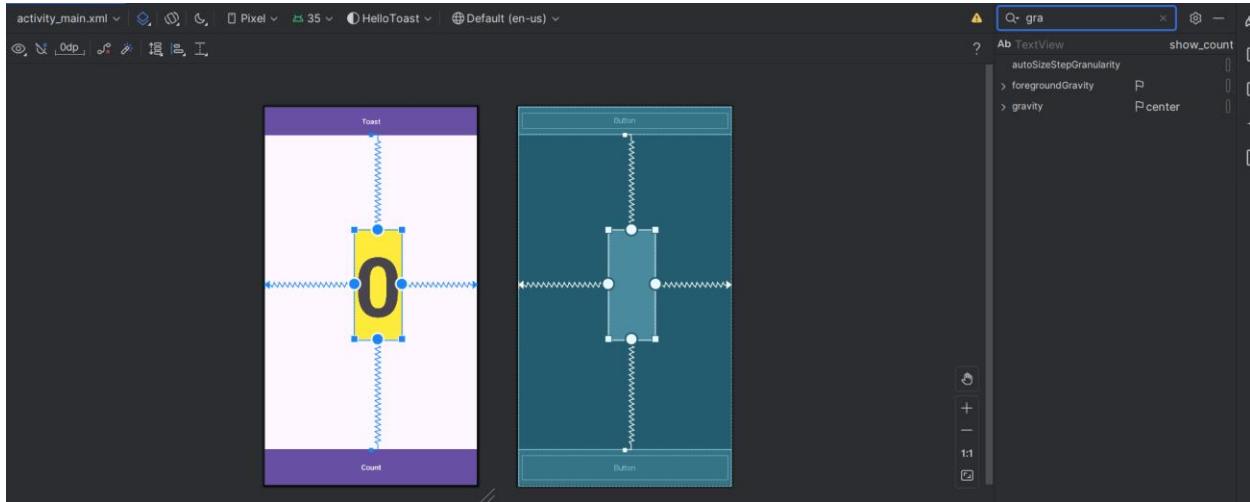
Khi TextView đang được chọn, mở bảng **Attributes** nếu nó chưa được mở. Thiết lập các thuộc tính cho TextView như minh họa trong hình động bên dưới. Những thuộc tính mà bạn chưa gặp trước đây sẽ được giải thích sau hình minh họa:

1. Đặt ID thành **show_count**.
2. Đặt text thành **0**.
3. Đặt textSize thành **160sp**.
4. Đặt textStyle thành **B** (đậm) và textAlign thành **ALIGNCENTER** (căn giữa đoạn văn bản).
5. Thay đổi chế độ kích thước ngang và dọc (layout_width và layout_height) thành **match_constraint**.

6. Đặt textColor thành @color/colorPrimary.

7. Cuộn xuống bảng và nhập vào **View all attributes**, tiếp tục cuộn xuống trang thuộc tính thứ hai đến background, sau đó nhập #FFF00 để chọn một tông màu vàng.

8. Cuộn xuống đến gravity, mở rộng gravity và chọn **center_ver** (để căn giữa theo chiều dọc).

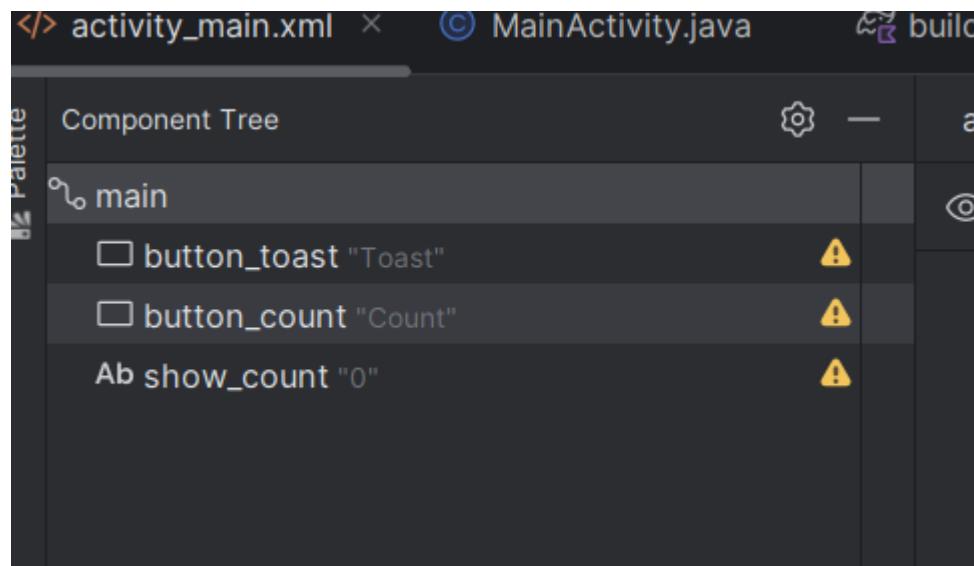


- **textSize:** Kích thước chữ của TextView. Trong bài học này, kích thước được đặt là 160sp. sp là viết tắt của scale-independent pixel (pixel độc lập với tỷ lệ), tương tự như dp, là một đơn vị tự điều chỉnh theo mật độ màn hình và tùy chọn kích thước phông chữ của người dùng. Hãy sử dụng đơn vị sp khi chỉ định kích thước phông chữ để đảm bảo chúng được điều chỉnh phù hợp với cả mật độ màn hình và sở thích của người dùng.
- **textStyle** và **textAlignment:** Kiểu chữ, được đặt thành **B** (bold - đậm) trong bài học này, và căn chỉnh văn bản, được đặt thành **ALIGNCENTER** (căn giữa đoạn văn).
- **gravity:** Thuộc tính gravity xác định cách một View được căn chỉnh bên trong View hoặc ViewGroup cha của nó. Trong bước này, bạn căn giữa TextView theo chiều dọc bên trong ConstraintLayout cha.

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

Nhiệm vụ 5: Chỉnh sửa bộ cục trong XML

Bộ 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 trong Component Tree. Di chuột qua các dấu chấm than này để xem thông báo cảnh báo, như hình minh họa bên dưới. Cả ba phần tử đều có cùng một cảnh báo: chuỗi được mã hóa cứng (hardcoded strings) nên sử dụng tài nguyên.



Cách dễ nhất để khắc phục các vấn đề về bộ cục là chỉnh sửa trực tiếp trong XML. Mặc dù trình chỉnh sửa bộ cục 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.

Bước 1: Mở mã XML của bộ cục

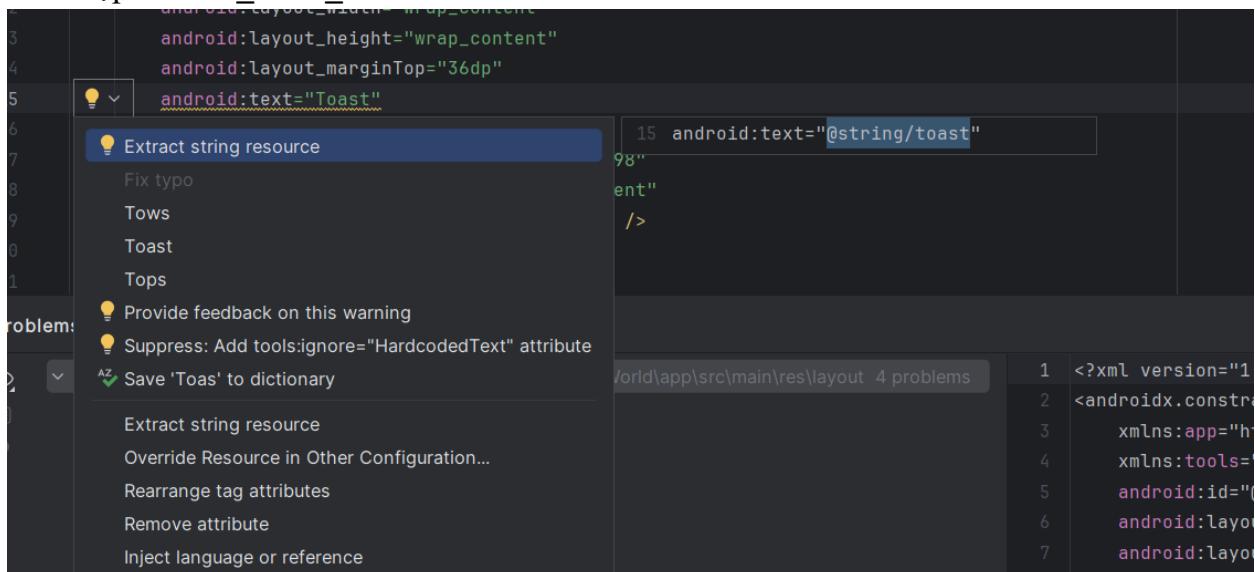
Đối với nhiệm vụ này, mở tệp activity_main.xml nếu nó chưa được mở, rồi nhấp vào tab Text ở dưới cùng của trình chỉnh sửa bộ cục.

Trình chỉnh sửa XML xuất hiện, thay thế các ngăn thiết kế và bản vẽ (design và blueprint panes). Như bạn có thể thấy trong hình bên dưới, hiển thị một phần mã XML của bộ cục, các cảnh báo được đánh dấu—các chuỗi được mã hóa cứng "Toast" và "Count". (Chuỗi "0" cũng được đánh dấu nhưng không hiển thị trong hình.) Di chuột qua chuỗi được mã hóa cứng "Toast" để xem thông báo cảnh báo.

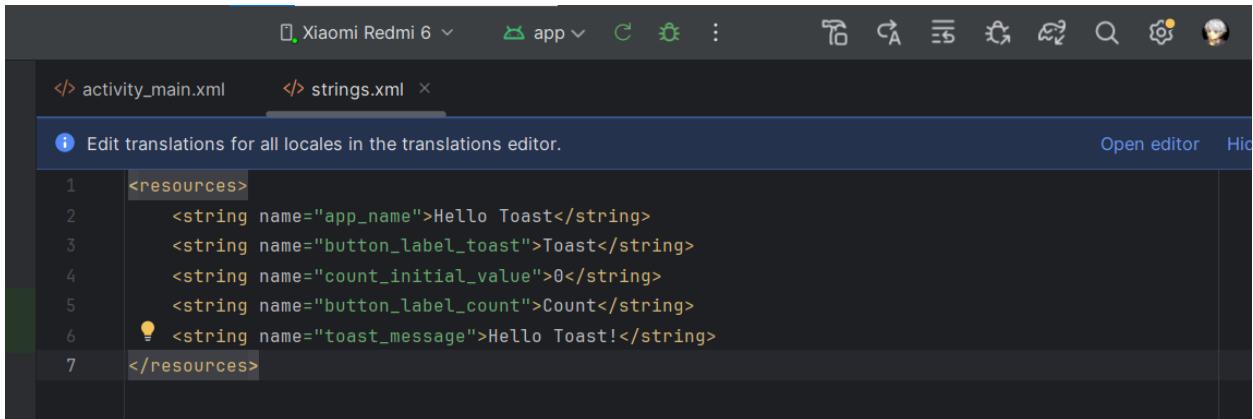
Bước 2: Trích xuất tài nguyên chuỗi (string resources)

Thay vì mã hóa cứng các chuỗi, một phương pháp tốt nhất là sử dụng tài nguyên chuỗi, giúp đại diện cho các chuỗi đó. Việc lưu trữ chuỗi trong một tệp riêng biệt giúp dễ dàng quản lý hơn, đặc biệt nếu bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và bản địa hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhập một lần vào từ "Toast" (cảnh báo đầu tiên được đánh dấu).
2. Nhấn Alt + Enter trên Windows hoặc Option + Enter trên macOS và chọn **Extract string resource** từ hộp thoại menu bật lên.
3. Nhập **button_label_toast** cho **Resource name**.

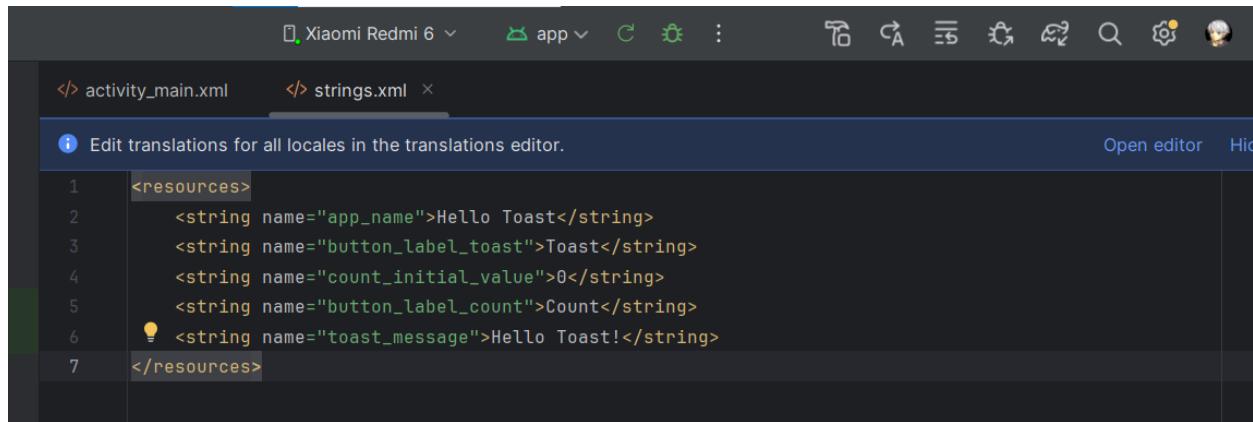


4. Nhập **OK**. Một tài nguyên chuỗi sẽ được tạo trong tệp values/res/values.xml, và chuỗi trong mã của bạn sẽ được thay thế bằng một tham chiếu đến tài nguyên:
`@string/button_label_toast`
5. Trích xuất các chuỗi còn lại: `button_label_count` cho "Count", và `count_initial_value` cho "0".
6. Trong bảng Project > **Android**, mở rộng thư mục **values** bên trong **res**, sau đó nhấp đúp vào **strings.xml** để xem các tài nguyên chuỗi của bạn trong tệp strings.xml.



```
<resources>
    <string name="app_name">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="count_initial_value">0</string>
    <string name="button_label_count">Count</string>
    <string name="toast_message">Hello Toast!</string>
</resources>
```

7. Bạn cần một chuỗi khác để sử dụng trong một nhiệm vụ tiếp theo nhằm hiển thị thông báo. Thêm vào tệp strings.xml một tài nguyên chuỗi mới có tên toast_message cho cụm từ "Hello Toast!":



```
<resources>
    <string name="app_name">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="count_initial_value">0</string>
    <string name="button_label_count">Count</string>
    <string name="toast_message">Hello Toast!</string>
</resources>
```

Mẹo: Các tài nguyên chuỗi bao gồm tên ứng dụng, tên này sẽ xuất hiện trên thanh ứng dụng ở đầu màn hình nếu bạn bắt đầu dự án ứng dụng bằng Empty Template. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa app_name resource.

Nhiệm vụ 6: Thêm onClick handlers cho các nút

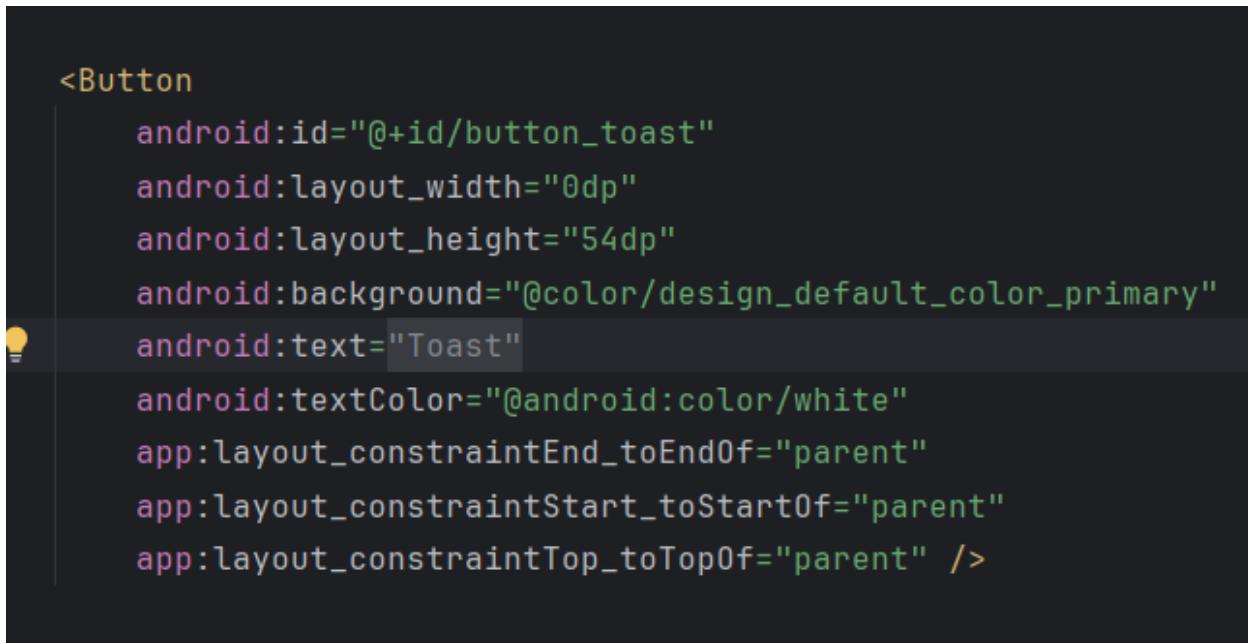
Trong nhiệm vụ này, bạn sẽ thêm một phương thức Java cho mỗi Button trong MainActivity, phương thức này sẽ được thực thi khi người dùng nhấn vào Button.

Bước 1: Thêm thuộc tính onClick và trình xử lý sự kiện cho mỗi Button

Một click handler là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick của **Attributes** trong tab **Design**. Bạn cũng có thể chỉ định tên của phương thức xử lý trong XML editor bằng cách thêm thuộc tính android:onClick vào

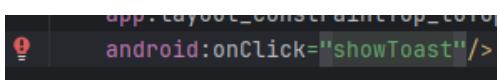
Button. Bạn sẽ sử dụng cách thứ hai vì hiện tại bạn chưa tạo các phương thức xử lý, và XML editor cung cấp một cách tự động để tạo các phương thức này.

1. Khi trình chỉnh sửa XML đang mở (tab Text), tìm Button có thuộc tính android:id được đặt thành button_toast.



```
<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="54dp"
    android:background="@color/design_default_color_primary"
    android:text="Toast"
    android:textColor="@android:color/white"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

2. Thêm thuộc tính android:onClick vào cuối phần tử button_toast, sau thuộc tính cuối cùng và trước ký hiệu kết thúc >.



3. Nhập vào biểu tượng bóng đèn màu đỏ xuất hiện bên cạnh thuộc tính. Chọn **Create click handler**, chọn **MainActivity**, rồi nhấp **OK**.

Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy nhập vào tên phương thức ("showToast"). Nhấn Alt-Enter (hoặc Option-Enter trên Mac), chọn Create 'showToast(view)' in MainActivity, rồi nhấp **OK**.

Hành động này sẽ tạo một phương thức khung (placeholder) cho phương thức showToast() trong MainActivity, như được hiển thị ở cuối các bước này.

4. Lặp lại hai bước cuối với nút button_count: Thêm thuộc tính android:onClick vào cuối và thêm trình xử lý sự kiện nhập (click handler).

```
    app:layout_constraintStart_toStartOf="parent"
    android:onClick="countUp"/>
```

Mã XML cho các phần tử giao diện người dùng trong ConstraintLayout hiện trông như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
<TextView
    android:id="@+id/show_count"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="#FFEB3B"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textSize="160sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/button_count"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button_toast"
    app:layout_constraintVertical_bias="1.0" />
```

```
<Button  
    android:id="@+id/button_toast"  
    android:layout_width="0dp"  
    android:layout_height="54dp"  
    android:background="@color/design_default_color_primary"  
    android:text="@string/button_label_toast"  
    android:textColor="@android:color/white"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
    android:onClick="showToast" />
```

```
<Button  
    android:id="@+id/button_count"  
    android:layout_width="0dp"  
    android:layout_height="70dp"  
    android:background="@color/design_default_color_primary"  
    android:text="@string/button_label_count"  
    android:textColor="@android:color/white"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.529"  
    app:layout_constraintStart_toStartOf="parent"  
    android:onClick="countUp"/>
```

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

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

```
package com.example.hellotoast;

import android.os.Bundle;

import android.view.View;

import android.widget.TextView;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

import org.w3c.dom.Text;public class MainActivity extends AppCompatActivity

{

    private int mCount = 0;

    private TextView mShowCount;

    @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;

        // });

        mShowCount = (TextView) findViewById(R.id.show_count);

    }

}
```

```

}

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

public void countUp(View view) {
    ++mCount;
    if(mShowCount != null) {
        mShowCount.setText(Integer.toString(mCount));
    }
}
}

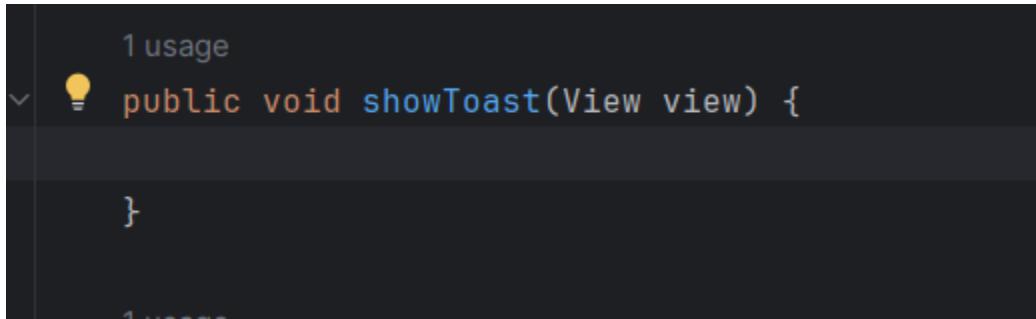
```

Bước 2: Chính sửa trình xử lý sự kiện của Toast Button

Bây giờ, bạn sẽ chỉnh sửa phương thức `showToast()` - trình xử lý sự kiện khi nhấn nút **Toast** trong `MainActivity` - để hiển thị một thông báo. `Toast` cung cấp một cách hiển thị thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ chiếm không gian đủ để hiển thị nội dung tin nhắn, trong khi activity hiện tại vẫn hiển thị và tương tác được. `Toast` hữu ích để kiểm tra tính tương tác của ứng dụng—bạn có thể thêm một thông báo `Toast` để hiển thị kết quả khi người dùng nhấn vào một `Button` hoặc thực hiện một hành động.

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

- Xác định vị trí phương thức `showToast()` vừa được tạo.



- Để tạo một thẻ hiển của `Toast`, hãy gọi phương thức tạo `makeText()` trên lớp `Toast`.

```
1 usage
public void showToast(View view) {
    Toast toast = Toast.makeText();
    toast.show();
}
```

Câu lệnh này chưa hoàn chỉnh cho đến khi bạn hoàn thành tất cả các bước.

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

```
1 usage
public void showToast(View view) {
    Toast toast = Toast.makeText(context: this);
    toast.show();
}
```

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

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

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

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

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

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

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

Chạy ứng dụng và kiểm tra rằng thông báo Toast xuất hiện khi nút **Toast** được nhấn.



Bước 3: Chỉnh sửa trình xử lý nút đếm (Count Button handler)

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

Mã của trình xử lý sự kiện cần:

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

Thực hiện các bước sau để chỉnh sửa trình xử lý sự kiện của nút Count:

1. Xác định vị trí phương thức countUp() vừa được tạo.

```
1 usage
💡 public void countUp(View view) {
    }
}
```

2. Để theo dõi số đếm, bạn cần một biến thành viên riêng (private). Mỗi lần nhấn nút Count, giá trị của biến này sẽ tăng lên.

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

Hãy nhập đoạn mã sau, đoạn này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn màu đỏ:

Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy chọn biểu thức mCount++. Biểu tượng bóng đèn sẽ xuất hiện sau một lúc.

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'mCount'** từ menu bật lên. Thao tác này sẽ tạo một biến thành viên private ở đầu MainActivity, và Android Studio sẽ mặc định biến này là kiểu số nguyên (int):

```
public class MainActivity extends AppCompatActivity {
```

💡 1 usage

```
    private int mCount ;
```

4. Thay đổi câu lệnh khai báo biến thành viên private để khởi tạo biến với giá trị bằng 0:

💡 1 usage

```
    private int mCount = 0;
```

1 usage

5. Cùng với biến trên, bạn cũng cần một biến thành viên private để tham chiếu đến TextView có ID là show_count, biến này sẽ được sử dụng trong trình xử lý sự kiện nhấn nút. Đặt tên biến là mShowCount.

💡 1 usage

```
    private int mCount = 0;
```

1 usage

```
    private TextView mShowCount;
```

Open code

6. Bây giờ, khi đã có mShowCount, bạn có thể lấy tham chiếu đến TextView bằng ID đã đặt trong tệp bố cục (layout file). Để chỉ lấy tham chiếu này một lần, hãy khai báo nó trong phương thức onCreate(). Như bạn sẽ tìm hiểu trong bài học khác, phương thức onCreate() được sử dụng để inflate layout, nghĩa là thiết lập nội dung của màn hình theo bố cục XML.

Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện khác trong bố cục, chẳng hạn như TextView. Tìm phương thức onCreate() trong MainActivity:

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: 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;
        });
    }
}
```

7. Thêm câu lệnh findViewById vào cuối phương thức onCreate():

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: 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;
        });
        mShowCount = (TextView) findViewById(R.id.show_count);
    }
}
```

Một View, giống như một chuỗi ký tự, là một tài nguyên có thể có ID. Lệnh findViewById nhận ID của một View làm tham số và trả về View đó. Vì phương thức này trả về một View, bạn cần ép kiểu kết quả về loại View mà bạn mong đợi, trong trường hợp này là (TextView).

8. Nay, sau khi đã gán TextView cho biến mShowCount, bạn có thể sử dụng biến này để đặt nội dung văn bản của TextView thành giá trị của biến mCount. Thêm đoạn mã sau vào phương thức countUp():

```
if(mShowCount != null){
    mShowCount.setText(Integer.toString(mCount));
}
```

Toàn bộ phương thức countUp() hiện trông như thế này:

```
public void countUp(View view) {  
    mCount++;  
    if(mShowCount != null){  
        mShowCount.setText(Integer.toString(mCount));  
    }  
}
```

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



Mẹo: Để có hướng dẫn chuyên sâu về cách sử dụng ConstraintLayout, hãy xem [Codelab Using ConstraintLayout to design your views](#).

Bài giải

Thử thách

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

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

Thử thách: Thay đổi bố cục để ứng dụng trông đẹp ở cả chế độ ngang và dọc:

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

Bài giải cho thử thách

Tóm tắt

View, ViewGroup và bộ cục (layouts):

- Tất cả các phần tử giao diện người dùng (UI) đều là lớp con của lớp View, do đó kế thừa nhiều thuộc tính từ lớp cha View.

- Các phần tử View có thể được nhóm lại bên trong một ViewGroup, đóng vai trò như một vùng chứa. Mỗi quan hệ này là cha-con, trong đó cha là một ViewGroup và con là một View hoặc một ViewGroup khác.
- Phương thức `onCreate()` được sử dụng để inflate bộ cục, tức là thiết lập nội dung hiển thị của màn hình theo bộ cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện người dùng khác trong bộ cục.
- Một View, giống như một chuỗi, là một tài nguyên có thể có ID. Lời gọi `findViewById` nhận ID của một View làm tham số và trả về View đó.

Sử dụng trình chỉnh sửa bộ cục:

- Nhập vào tab **Design** để thao tác với các phần tử và bộ cục, và tab **Text** để chỉnh sửa mã XML của bộ cục.
- Trong tab **Design**, ngăn **Palettes** hiển thị các phần tử giao diện người dùng (UI) mà bạn có thể sử dụng trong bộ cục của ứng dụng, còn ngăn Component tree hiển thị cấu trúc phân cấp của các phần tử UI.
- Các ngăn thiết kế và bản thiết kế trong trình chỉnh sửa bộ cục hiển thị các phần tử UI trong bộ cục.
- Tab **Attributes** hiển thị ngăn **Attributes**, cho phép thiết lập các thuộc tính cho một phần tử UI.
- Tay nắm ràng buộc (Constraint handle): Nhập vào tay nắm ràng buộc, được hiển thị dưới dạng vòng tròn ở mỗi cạnh của phần tử, sau đó kéo đến một tay nắm ràng buộc khác hoặc biên của phần tử cha để tạo ràng buộc. Ràng buộc được biểu diễn bằng đường gấp khúc.
- Tay nắm thay đổi kích thước (Resizing handle): Bạn có thể kéo các tay nắm vuông để thay đổi kích thước phần tử. Khi kéo, tay nắm sẽ chuyển thành một góc xiên.
- Công cụ Autoconnect: Khi được bật, công cụ này sẽ tự động tạo hai hoặc nhiều ràng buộc giữa một phần tử UI và bộ cục cha. Sau khi bạn kéo phần tử vào bộ cục, nó sẽ tạo ràng buộc dựa trên vị trí của phần tử.
- Xóa ràng buộc: Bạn có thể xóa ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di chuột qua nó để hiển thị nút Clear Constraints. Nhập vào nút này để xóa tất cả ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc cụ thể, hãy nhấp vào tay nắm thiết lập ràng buộc đó.
- Ngăn **Attributes**: Cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Ngoài ra, nó còn có một bảng định cỡ hình vuông

gọi là view inspector ở phía trên. Các biểu tượng bên trong hình vuông này biểu thị các thiết lập chiều cao và chiều rộng.

Thiết lập layout_width và layout_height

Các thuộc tính layout_width và layout_height sẽ thay đổi khi bạn điều chỉnh kích thước chiều rộng và chiều cao trong view inspector. Các thuộc tính này có thể nhận một trong ba giá trị đối với ConstraintLayout:

- Giá trị match_constraint: Mở rộng View để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao—tối đa đến phần lề nếu có thiết lập.
- Giá trị wrap_content: Thu nhỏ kích thước của View sao cho vừa đủ chứa nội dung bên trong. Nếu không có nội dung, View sẽ trở nên vô hình.
- Sử dụng số dp cố định (density-independent pixels): Để chỉ định một kích thước cố định, được điều chỉnh theo kích thước màn hình của thiết bị.

Trích xuất tài nguyên chuỗi (Extracting string resources)

Thay vì mã hóa cứng (hard-coding) chuỗi văn bản, thực tiễn tốt nhất là sử dụng string resources, đại diện cho các chuỗi. Thực hiện các bước sau:

1. Nhập vào chuỗi mã hóa cứng cần trích xuất, nhấn **Alt + Enter** (**Option + Enter** trên Mac), và chọn **Extract string resources** từ menu bật lên.
2. Đặt tên tài nguyên (**Resource name**).
3. Nhấn **OK**. Thao tác này sẽ tạo một tài nguyên chuỗi trong tệp values/res/values.xml, và chuỗi trong mã nguồn của bạn sẽ được thay thế bằng một tham chiếu đến tài nguyên đó: @string/button_label_toast.

Xử lý sự kiện nhấp chuột (Handling clicks)

- Một click handler là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử giao diện người dùng (UI).
- Xác định một click handler cho một phần tử UI như Button bằng cách nhập tên phương thức vào trường onClick trong ngăn **Attributes** của tab **Design** hoặc thêm thuộc tính android:onClick vào phần tử UI trong trình chỉnh sửa XML.

- Tạo click handler trong Activity chính bằng cách sử dụng tham số View. Ví dụ:
public void showToast(View view) {...}.
- Bạn có thể tìm thấy thông tin về tất cả các thuộc tính của Button trong tài liệu lớp Button, và tất cả các thuộc tính của TextView trong tài liệu lớp TextView.

Hiển thị thông báo Toast:

Toast cung cấp một cách để hiển thị một thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ chiếm đúng lượng không gian cần thiết để hiển thị thông điệp. Để tạo một thẻ hiện của Toast, thực hiện các bước sau:

1. Gọi phương thức tạo makeText() trên lớp Toast.
2. Cung cấp context của Activity trong ứng dụng và thông điệp cần hiển thị (chẳng hạn như một tài nguyên chuỗi).
3. Cung cấp thời gian hiển thị, ví dụ Toast.LENGTH_SHORT để hiển thị trong thời gian ngắn. Thời gian hiển thị có thể là Toast.LENGTH_LONG hoặc Toast.LENGTH_SHORT.
4. Hiển thị Toast bằng cách gọi phương thức show().

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [1.2: Layouts and resources for the UI](#).

Tìm hiểu thêm

Tài liệu phát triển Android:

Khác:

Bài thực hành lập trình tiếp theo: [Android fundamentals 1.2 Part B: The layout editor](#)

1.3) Trình chỉnh sửa bố cục

Giới thiệu

Như những gì bạn đã học ở mục 1.2) Giao diện người dùng tương tác đầu tiên, bạn có thể xây dựng một giao diện người dùng (UI) sử dụng ConstraintLayout trong trình chỉnh sửa bố cục, nơi chứa các thành phần của UI trong bố cục một bố cục có sử dụng các ràng buộc để kết nối với các thành phần khác và với các cạnh của bố cục. ConstraintLayout đã được thiết kế để giúp bạn dễ dàng kéo các thành phần UI vào trình chỉnh sửa bố cục.

ConstraintLayout là một ViewGroup, tức là một View đặc biệt có thể chứa các đối tượng View khác (gọi là children hoặc child views). Bài thực hành này giới thiệu thêm 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 chiều dọc.
- RelativeLayout: Một nhóm các phần tử View con, trong đó mỗi View được định vị và căn chỉnh dựa trên các phần tử View khác trong ViewGroup. Vị trí của các View con được xác định tương quan với nhau hoặc với ViewGroup cha.

Những gì bạn cần biết

Bạn cần có khả năng làm:

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

Những gì bạn sẽ học

- Cách tạo một biến thể bố cục cho hướng ngang (landscape).
- Cách tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Cách sử dụng baseline constraint để căn chỉnh các phần tử giao diện người dùng 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.

Những gì bạn sẽ làm

- Tạo một biến thể bố cục cho hướng hiển thị ngang.

- Tạo một biến thể bố cục cho máy tính bảng và các màn hình lớn hơn.
- Chính sửa bộ cục để thêm ràng buộc cho các phần tử giao diện người dùng.
- Sử dụng baseline constraints của ConstraintLayout để căn chỉnh các phần tử với văn bản.
- Sử dụng các nút pack và align của ConstraintLayout để căn chỉnh các phần tử.
- Thay đổi bộ cục để sử dụng LinearLayout.
- Định vị các phần tử trong LinearLayout.
- Thay đổi bộ cục để sử dụng RelativeLayout.
- Sắp xếp lại các View trong bộ cục chính để chúng tương quan với nhau.

Tổng quan về ứng dụng

Trong bài học trước, ứng dụng Hello Toast sử dụng ConstraintLayout để sắp xếp các thành phần UI trong bộ cục Activity layout, hiển thị như trong hình bên dưới:

Để thực hành nhiều hơn với ConstraintLayout, bạn sẽ tạo một biến thể của bộ cục này theo hướng ngang hiển thị như trong hình bên dưới:

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

Bạn cũng sẽ tìm hiểu về các lớp con ViewGroup khác như LinearLayout và RelativeLayout và thay đổi bộ cục ứng dụng Hello Toast để sử dụng chúng.

Nhiệm vụ 1: Tạo các biến thể bộ cục (layout variants)

Trong bài học trước, thử thách lập trình yêu cầu bạn thay đổi bộ cục của ứng dụng Hello Toast để nó hiển thị đúng trong cả hướng ngang và hướng dọc. Trong nhiệm vụ này, bạn sẽ học một cách dễ dàng hơn để tạo các biến thể của bộ cục cho hướng ngang (còn được gọi là landscape) và hướng dọc (còn được gọi là portrait) trên điện thoại, cũng như cho các màn hình lớn hơn như máy tính bảng.

Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trong hai thanh công cụ trên cùng của trình chỉnh sửa bộ cục. Thanh công cụ trên cùng cho phép bạn cấu hình cách hiển thị bản xem trước bộ cục trong trình chỉnh sửa bộ cục:

Trong hình trên:

1. Chọn bìa mặt thiết kế (**Design Surface**): Chọn **Design** để hiển thị bản xem trước có màu của bố cục hoặc **Blueprint** để chỉ hiển thị đường viền của từng phần tử giao diện người dùng. Để xem cả hai dạng hiển thị song song, chọn **Design + Blueprint**.
2. Hướng hiển thị trong trình chỉnh sửa (**Orientation in Editor**): Chọn **Portrait** hoặc **Landscape** để hiển thị bản xem trước theo hướng dọc hoặc ngang. Điều này hữu ích để xem trước bố cục mà không cần chạy ứng dụng trên trình giả lập hoặc thiết bị. Để tạo bố cục thay thế, chọn **Create Landscape Variation** hoặc các biến thể khác.
3. Thiết bị trong trình chỉnh sửa (**Device in Editor**): Chọn loại thiết bị (điện thoại/máy tính bảng, Android TV hoặc Android Wear).
4. Phiên bản API trong trình chỉnh sửa (**API Version in Editor**): Chọn phiên bản Android để hiển thị bản xem trước.
5. Chủ đề trong trình chỉnh sửa (**Theme in Editor**): Chọn một chủ đề (chẳng hạn như **AppTheme**) để áp dụng cho bản xem trước.
6. Ngôn ngữ và vùng (**Locale in Editor**): Chọn ngôn ngữ và vùng cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có sẵn trong tài nguyên chuỗi (string resources). (Xem bài học về localization để biết chi tiết cách thêm ngôn ngữ). Bạn cũng có thể chọn **Preview as Right To Left** để xem bố cục theo hướng của một ngôn ngữ viết từ phải sang trái (RTL).

Thanh công cụ thứ hai cho phép bạn cấu hình giao diện của các phần tử trong ConstraintLayout, cũng như phóng to và di chuyển bản xem trước.

Trong hình trên:

1. Hiển thị (Show): Chọn Show Constraints và Show Margins để hiển thị hoặc ẩn chúng trong bản xem trước.
2. Tự động kết nối (Autoconnect): Bật hoặc tắt tính năng Autoconnect. Khi bật Autoconnect, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như Button) đến bất kỳ vị trí nào trong bố cục để tự động tạo ràng buộc với bố cục cha.

3. Xóa tất cả ràng buộc (Clear All Constraints): Xóa tất cả các ràng buộc trong toàn bộ bố cục.
4. Suy luận ràng buộc (Infer Constraints): Tạo ràng buộc bằng cách suy luận.
5. Lề mặc định (Default Margins): Thiết lập giá trị lề mặc định.
6. Đóng gói (Pack): Đóng gói hoặc mở rộng các phần tử đã chọn.
7. Căn chỉnh (Align): Căn chỉnh các phần tử đã chọn.
8. Đường hướng dẫn (Guidelines): Thêm đường hướng dẫn dọc hoặc ngang.
9. Điều khiển thu phóng/dịch chuyển (Zoom/Pan Controls): Phóng to hoặc thu nhỏ.

Mẹo: Để tìm hiểu thêm về cách sử dụng trình chỉnh sửa bố cục, xem Build a UI with Layout Editor. Để tìm hiểu cách tạo bố cục với ConstraintLayout, xem Build a Responsive UI with ConstraintLayout.

Bước 1: Xem trước bố cục theo hướng ngang

Để xem trước bố cục của ứng dụng Hello Toast trong hướng ngang, hãy làm theo các bước sau:

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

Lưu ý: Nếu bạn đã tải xuống mã giải pháp của HelloToast, bạn cần xóa các bố cục hướng ngang (landscape) và bố cục màn hình cực lớn (extra-large) đã hoàn chỉnh, vì bạn sẽ tạo lại chúng trong nhiệm vụ này. Chuyển từ Project > Android sang Project > Project Files trong bảng Project. Mở rộng thư mục: app > src/main > res. Chọn cả hai thư mục layout-land và layout-xlarge, sau đó chọn Edit > Delete. Chuyển bảng Project trở lại chế độ Project > Android.

2. Mở tệp bố cục activity_main.xml. Nhấp vào tab Design nếu nó chưa được chọn.
3. Nhấp vào nút Orientation in Editor trên thanh công cụ phía trên.
4. Chọn Switch to Landscape trong menu thả xuống. Bố cục sẽ hiển thị ở hướng ngang như hình bên dưới. Để quay lại hướng dọc, chọn Switch to Portrait.

Bước 2: Tạo một biến thể bố cục cho hướng ngang

Sự khác biệt trực quan giữa hướng dọc và hướng ngang của bố cục này là chữ số (0) trong phần tử TextView của show_count bị đặt quá thấp khi ở hướng ngang—quá gần với nút **Count**. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử TextView có thể hiển thị quá lớn hoặc không căn giữa vì kích thước văn bản được cố định ở 160sp.

Để khắc phục vấn đề này trong hướng ngang mà vẫn giữ nguyên hướng dọc, bạn có thể tạo một biến thể của bố cục ứng dụng Hello Toast dành riêng cho hướng ngang. Thực hiện theo các bước sau:

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

Một cửa sổ chỉnh sửa mới sẽ mở ra với tab land/activity_main.xml, hiển thị bố cục dành cho hướng ngang. Bạn có thể thay đổi bố cục này riêng cho hướng ngang mà không ảnh hưởng đến bố cục gốc ở hướng dọc.

3. Trong bảng **Project > Android**, mở thư mục **res > layout**, bạn sẽ thấy Android Studio đã tự động tạo biến thể bố cục có tên activity_main.xml (land).

Bước 3: Xem trước bố cục cho các thiết bị khác

Bạn có thể xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị hoặc trình giả lập. Thực hiện theo các bước sau:

1. Tab **land/activity_main.xml** vẫn phải đang mở trong trình chỉnh sửa bố cục; nếu chưa, hãy nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục layout.
2. Nhấp vào nút **Device in Editor** trên thanh công cụ phía trên.
3. Chọn một thiết bị khác trong menu thả xuống. Ví dụ: chọn **Nexus 4**, **Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong bản xem trước. Những khác biệt này là do kích thước văn bản cố định của TextView.

Bước 4: Thay đổi bố cục theo hướng ngang

Bạn có thể sử dụng bảng Attributes trong tab Design để thiết lập hoặc thay đổi thuộc tính, nhưng đôi khi sẽ nhanh hơn nếu chỉnh sửa trực tiếp mã XML trong tab Text. Tab Text hiển thị mã XML và cung cấp tab Preview ở phía bên phải cửa sổ để hiển thị bản xem trước bố cục, như trong hình minh họa dưới đây.

Hình trên hiển thị các phần sau:

1. Tab Preview, dùng để hiển thị bảng xem trước.
2. Bảng xem trước.
3. Mã XML.

Để thay đổi bố cục, thực hiện theo các bước sau:

1. Tab **land/activity_main.xml** vẫn phải đang mở trong trình chỉnh sửa bố cục; nếu chưa, hãy nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục layout.
2. Nhấp vào tab **Text** và tab **Preview** (nếu chưa được chọn).
3. Tìm phần tử TextView trong mã XML.
4. Thay đổi thuộc tính `android:textSize="160sp"` thành `android:textSize="120sp"`.
Bảng xem trước bố cục sẽ hiển thị kết quả thay đổi.
5. Chọn các thiết bị khác nhau trong menu thả xuống **Device in Editor** để xem bố cục hiển thị như thế nào trên các thiết bị khác ở hướng ngang.

Trong bảng chỉnh sửa, tab **land/activity_main.xml** hiển thị bố cục dành cho hướng ngang. Tab **activity_main.xml** hiển thị bố cục chưa thay đổi dành cho hướng dọc. Bạn có thể chuyển đổi qua lại bằng cách nhấp vào các tab.

6. Chạy ứng dụng trên trình giả lập hoặc thiết bị, sau đó chuyển đổi giữa hướng dọc và hướng ngang để xem cả hai bố cục.

Bước 5: Tạo một biến thể bố cục cho máy tính bảng

Như bạn đã học trước đó, bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ phía trên. Nếu bạn chọn một thiết bị như **Nexus 10** (một máy tính bảng) từ menu, bạn sẽ thấy rằng bố cục hiện tại không tối ưu cho màn hình máy tính bảng - bản trên mỗi Button quá nhỏ và cách sắp xếp các Button ở trên cùng và dưới cùng không phù hợp với màn hình lớn của máy tính bảng.

Để điều chỉnh bố cục cho máy tính bảng mà không ảnh hưởng đến bố cục trên điện thoại ở chế độ ngang và dọc, hãy làm theo các bước sau:

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị các khung xem thiết kế và bản vẽ phác thảo (design & blueprint panes).

- Nhấp vào nút **Orientation in Editor** trên thanh công cụ phía trên.
- Chọn **Create layout x-large Variation**.

Sau đó, một cửa sổ trình chỉnh sửa mới sẽ mở ra với tab **xlarge/activity_main.xml**, hiển thị bố cục dành riêng cho thiết bị có kích thước màn hình máy tính bảng.

Trình chỉnh sửa cũng tự động chọn một thiết bị máy tính bảng, chẳng hạn như Nexus 9 hoặc Nexus 10, để hiển thị bản xem trước. Bạn có thể thay đổi bố cục này để tối ưu cho máy tính bảng mà không làm ảnh hưởng đến các bố cục khác.

Bước 6: Thay đổi biến thể bố cục cho máy tính bảng

Bạn có thể sử dụng bảng Attributes trong tab **Design** để thay đổi thuộc tính cho bố cục này.

- Tắt công cụ Autoconnect trên thanh công cụ. Đảm bảo rằng công cụ này đã bị vô hiệu hóa.
- Xóa tất cả ràng buộc (constraints) trong bố cục bằng cách nhấp vào nút **Clear All Constraints** trên thanh công cụ.

Khi các ràng buộc bị loại bỏ, bạn có thể di chuyển và thay đổi kích thước các phần tử trên bố cục một cách tự do.

- Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước ở bốn góc của mỗi phần tử. Trong **Component Tree**, chọn TextView có tên `show_count`. Để di chuyển Button dễ dàng hơn, hãy kéo một góc của TextView để thay đổi kích thước của nó (như trong hình động bên dưới).

Việc thay đổi kích thước một phần tử theo cách này sẽ gán cố định chiều rộng và chiều cao. Tránh gán kích thước cố định cho hầu hết các phần tử vì bạn không thể đoán trước cách hiển thị của chúng trên các màn hình có kích thước và mật độ khác nhau. Bạn chỉ đang làm điều này để di chuyển phần tử sang một vị trí khác, và sẽ thay đổi kích thước của nó trong một bước sau.

- Trong **Component Tree**, chọn Button có tên `button_toast`, nhấp vào tab **Attributes** để mở bảng **Attributes** và thay đổi các thuộc tính sau: `textSize = 60sp` (#1 trong hình minh họa bên dưới), `layout_width = wrap_content` (#2 trong hình minh họa bên dưới)

Như được hiển thị ở bên phải của hình minh họa (#2), bạn có thể nhấp vào điều khiển chiều rộng trong view inspector (xuất hiện dưới dạng hai đoạn ở bên trái và bên phải của hình vuông) cho đến khi nó hiển thị Wrap Content. Ngoài ra, bạn có thể chọn **wrap_content** từ menu layout_width.

Sử dụng wrap_content giúp nút có thể thay đổi kích thước linh hoạt nếu văn bản trong Button được dịch sang một ngôn ngữ khác, đảm bảo nút có chiều rộng phù hợp với từ ngữ trong từng ngôn ngữ.

5. Trong **Component Tree**, chọn Button có tên button_count, thay đổi các thuộc tính sau: textSize = **60sp**, layout_width = **wrap_content** và di chuyển button_count lên trên TextView, vào một khoảng trống trong bố cục.

Bước 7: Sử dụng ràng buộc cơ sở (baseline constraint)

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

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của một thiết bị chạy Android bao gồm một hệ thống phân cấp các đối tượng gọi là Views. Mỗi phần tử trên màn hình đều là một View.

Lớp View đại diện cho khối xây dựng cơ bản của tất cả các thành phần giao diện người dùng. View là lớp cơ sở cho các lớp cung cấp các thành phần giao diện tương tác, chẳng hạn như phần tử Button (nút). Button là một phần tử UI mà người dùng có thể nhấn hoặc bấm để thực hiện một hành động.

Bạn có thể biến bất kỳ View nào, chẳng hạn như ImageView, thành một phần tử UI có thể nhấn hoặc bấm. Bạn phải lưu trữ hình ảnh cho ImageView trong thư mục **drawables** của dự án.

Trong bài thực hành này, bạn sẽ học cách sử dụng hình ảnh làm phần tử mà người dùng có thể nhấn hoặc bấm.

Những gì bạn cần biết trước

Bạn nên có kĩ năng:

- Tạo một dự án Android Studio từ mẫu và tạo bố cục chính.
- Chạy ứng dụng trên trình giả lập hoặc một thiết bị được kết nối.
- Tạo và chỉnh sửa các phần tử giao diện người dùng bằng trình chỉnh sửa bố cục và mã XML.
- Truy cập các phần tử giao diện từ mã bằng **findViewById()**.
- Xử lý sự kiện nhấn **Button**.
- Hiển thị thông báo **Toast**.
- Thêm hình ảnh vào thư mục **drawable** của dự án.

Những gì bạn sẽ học

- Cách sử dụng hình ảnh làm phần tử tương tác để thực hiện hành động.
- Cách đặt thuộc tính cho phần tử **ImageView** trong trình chỉnh sửa bố cục.
- Cách thêm phương thức **onClick()** để hiển thị thông báo **Toast**.

Những gì bạn sẽ làm

- Tạo một dự án Android Studio mới cho một ứng dụng giả lập đặt món tráng miệng, sử dụng hình ảnh làm phần tử tương tác.
- Thiết lập trình xử lý **onClick()** cho các hình ảnh để hiển thị các thông báo **Toast** khác nhau.
- Thay đổi **Floating Action Button** (FAB) được cung cấp bởi mẫu để hiển thị một biểu tượng khác và mở một **Activity** khác.

Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ tạo và xây dựng một ứng dụng mới bằng cách bắt đầu với mẫu **Basic Activity**, mô phỏng một ứng dụng đặt món tráng miệng.

Người dùng có thể nhấn vào một hình ảnh để thực hiện một hành động—trong trường hợp này là hiển thị một thông báo **Toast**, như minh họa trong hình dưới đây. Ngoài ra, người dùng có thể nhấn vào nút **giỎ HÀNG** để chuyển sang **Activity** tiếp theo.

Nhiệm vụ 1: Thêm hình ảnh vào bố cục

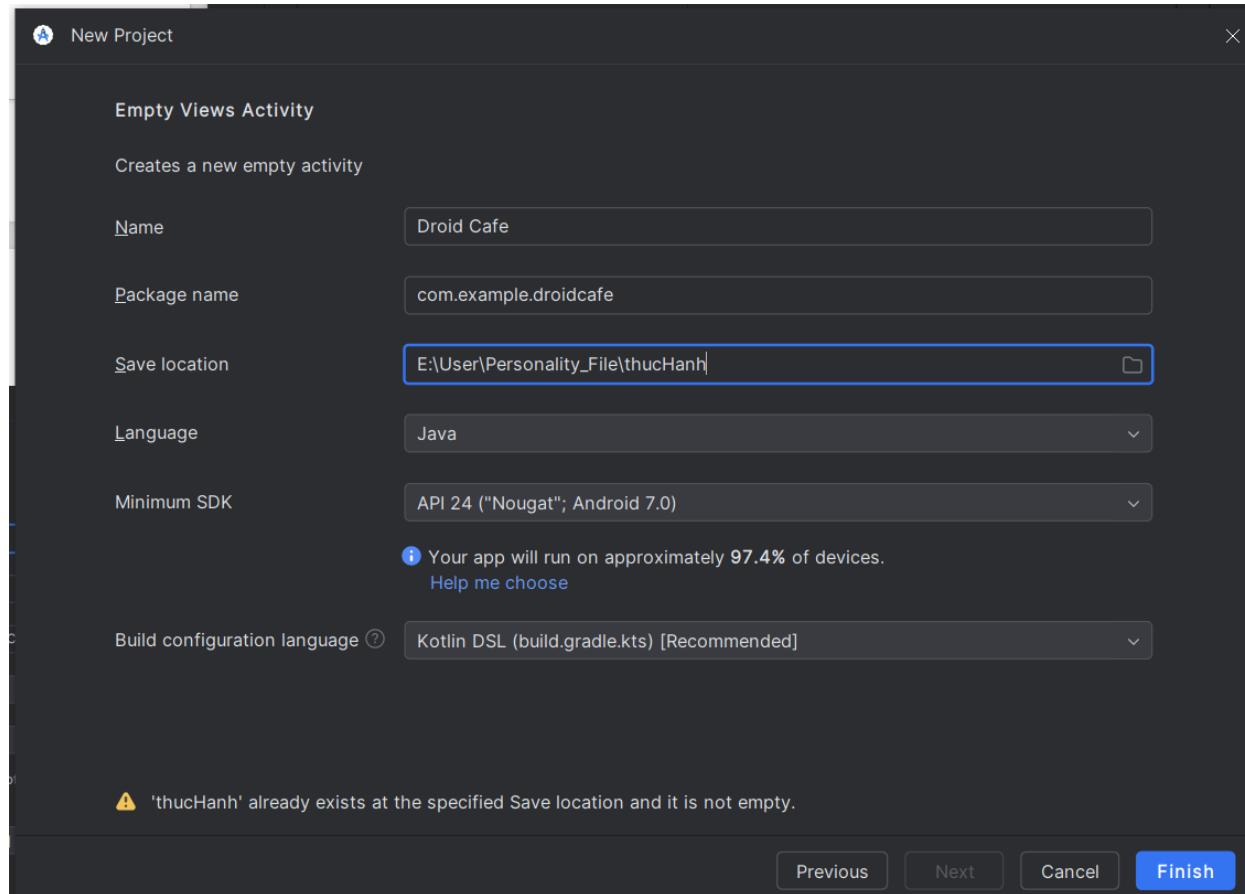
Bạn có thể làm cho một **View** có thể nhấn giống như một **Button** bằng cách thêm thuộc tính `android:onClick` trong bố cục XML. Ví dụ, bạn có thể làm cho một hình ảnh hoạt động như một nút bằng cách thêm `android:onClick` vào **ImageView**.

Trong nhiệm vụ này, bạn sẽ tạo một nguyên mẫu ứng dụng đặt món tráng miệng cho một quán cà phê. Sau khi bắt đầu một dự án mới dựa trên mẫu **Basic Activity**, bạn sẽ:

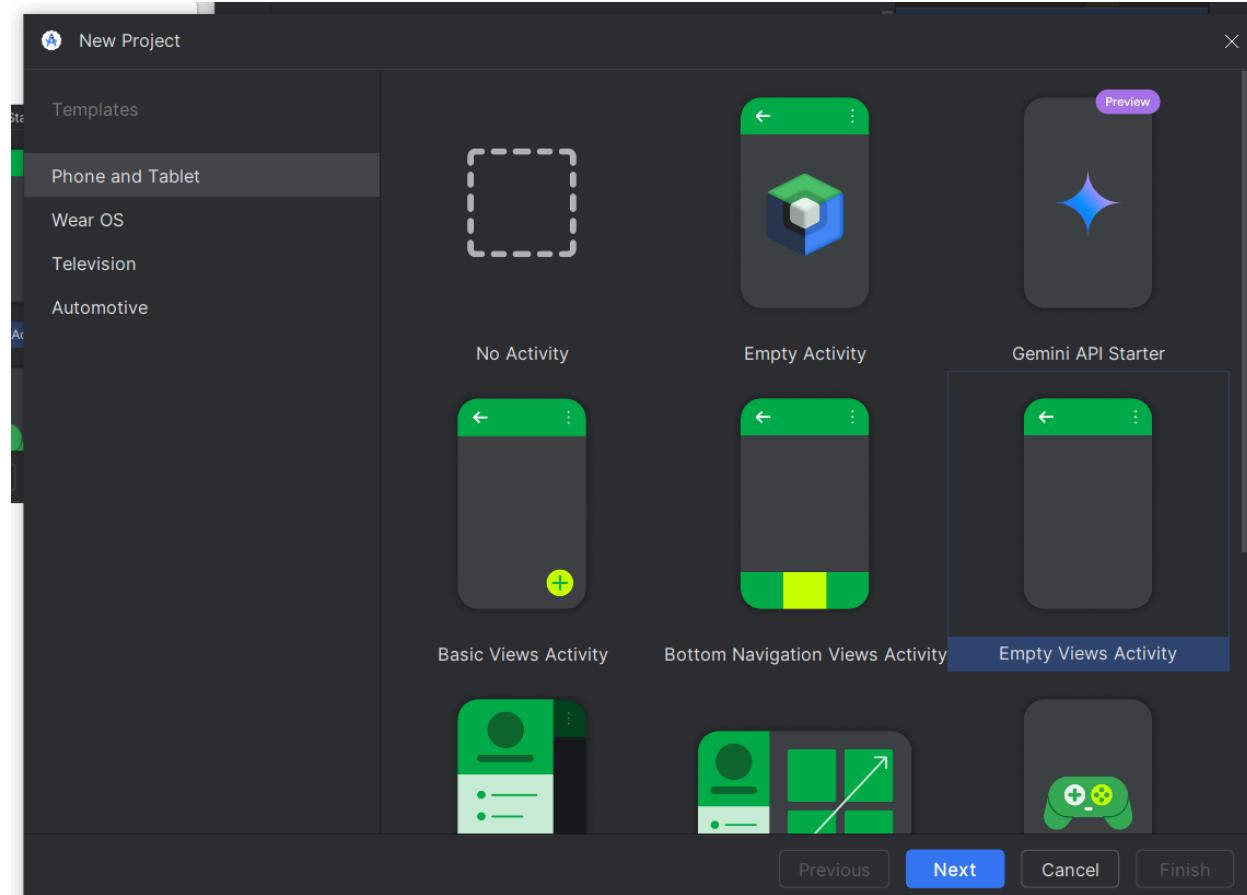
- Chính sửa **TextView "Hello World"** với văn bản phù hợp.
- Thêm các hình ảnh mà người dùng có thể nhấn vào.

1.1 Bắt đầu dự án mới:

1. Bắt đầu một dự án Android Studio mới với tên ứng dụng là **Droid Cafe**.



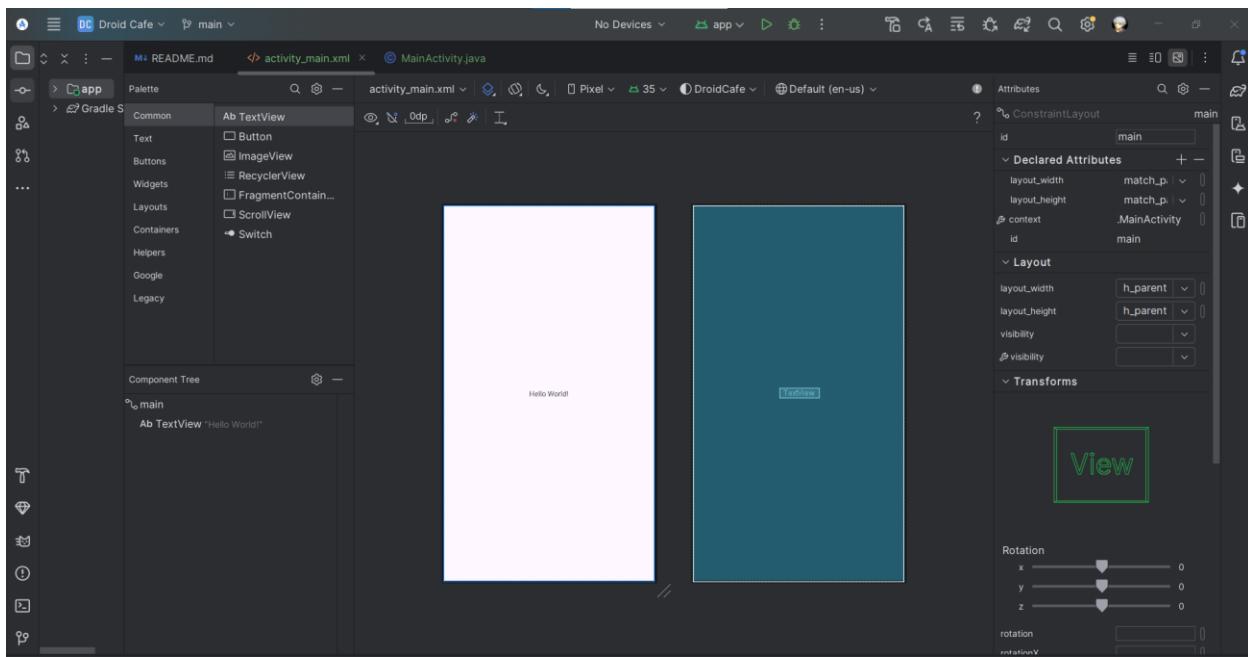
2. Chọn mẫu **Basic Activity** và chấp nhận tên Activity mặc định (**MainActivity**).
Đảm bảo rằng các tùy chọn **Generate Layout file** và **Backwards Compatibility (AppCompat)** được chọn.



3. Nhấn **Finish**.

Dự án sẽ mở ra với hai tệp bối cảnh trong thư mục **res > layout**: **activity_main.xml** dành cho thanh ứng dụng (app bar) và nút **Floating Action Button** (bạn sẽ không thay đổi trong nhiệm vụ này). **content_main.xml** dành cho mọi thứ khác trong bối cảnh.

4. Mở **content_main.xml** và nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị trình chỉnh sửa bối cảnh.

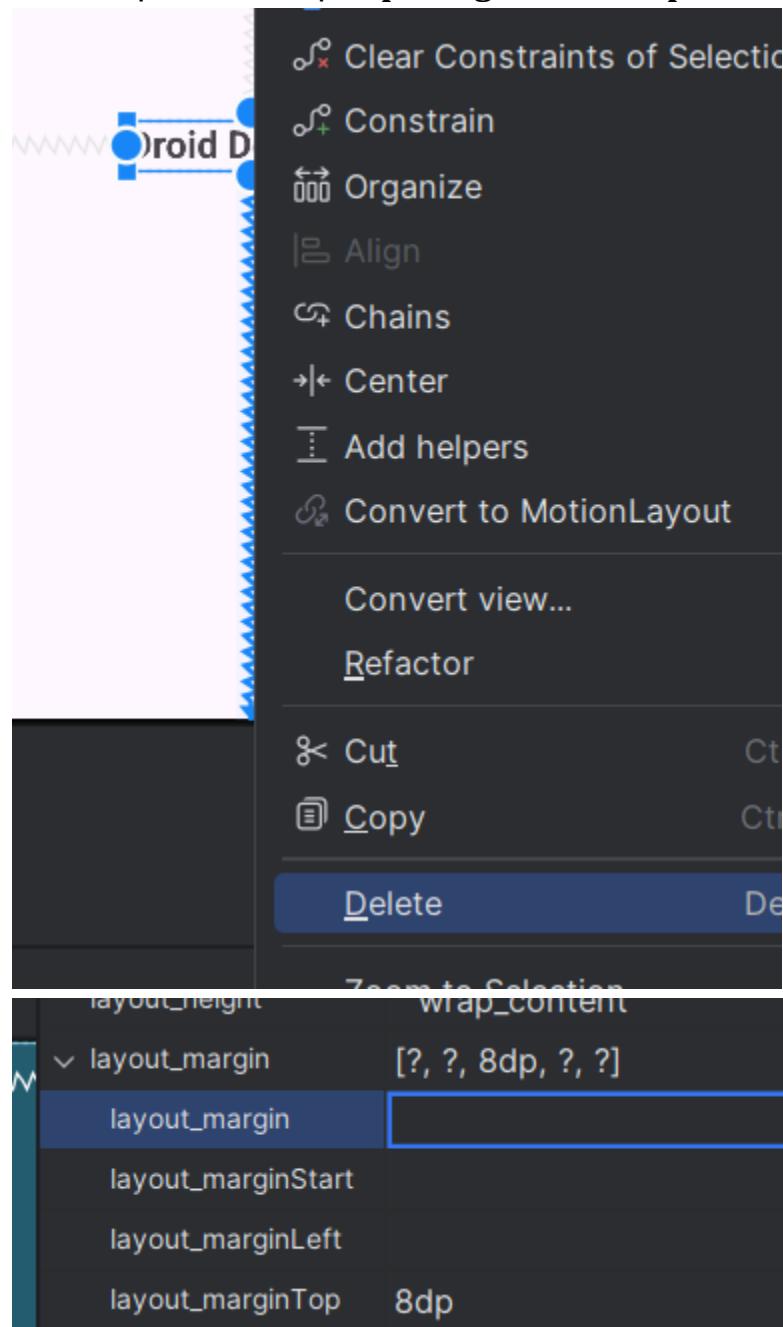


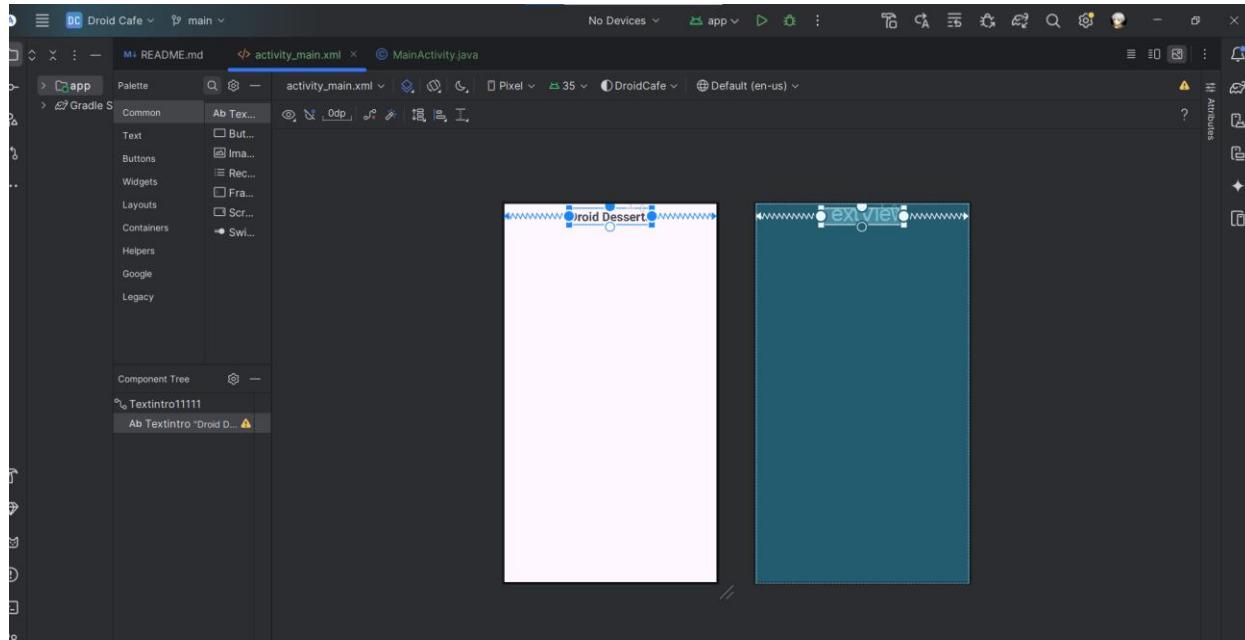
5. Chọn **TextView "Hello World"** trong bố cục và mở bảng **Attributes**.
6. Thay đổi các thuộc tính **textIntro** như sau:

Trường thuộc tính (Attribute Field)	Điền vào như sau:
ID	<p>Textintro</p>
text	<p>Đổi</p> <p>Hello world thành Droid Desserts</p>
textStyle	<p>B (In đậm)</p>
textSize	<p>24sp</p>

Điều này sẽ thêm thuộc tính android:id vào **TextView** với **id** được đặt là `textintro`, thay đổi nội dung văn bản, làm cho văn bản in đậm và đặt kích thước chữ lớn hơn là **24sp**.

7. Xóa ràng buộc (**constraint**) kéo dài từ cạnh dưới của **TextView** `textintro` đến cạnh dưới của bố cục, để **TextView** tự động gắn vào phía trên cùng của bố cục. Sau đó, đặt **top margin** thành **8dp**, như minh họa bên dưới.





8. Trong bài học trước, bạn đã học cách trích xuất một tài nguyên chuỗi (**string resource**) từ một chuỗi văn bản trực tiếp.

Nhấp vào tab **Text** để chuyển sang mã XML, trích xuất chuỗi "Droid Desserts" trong **TextView** và nhập **intro_text** làm tên tài nguyên chuỗi.

```
<TextView  
    android:id="@+id/Textintro"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="8dp"  
    android:text="@string/intro_text"  
    android:textSize="24sp"  
    android:textStyle="bold"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

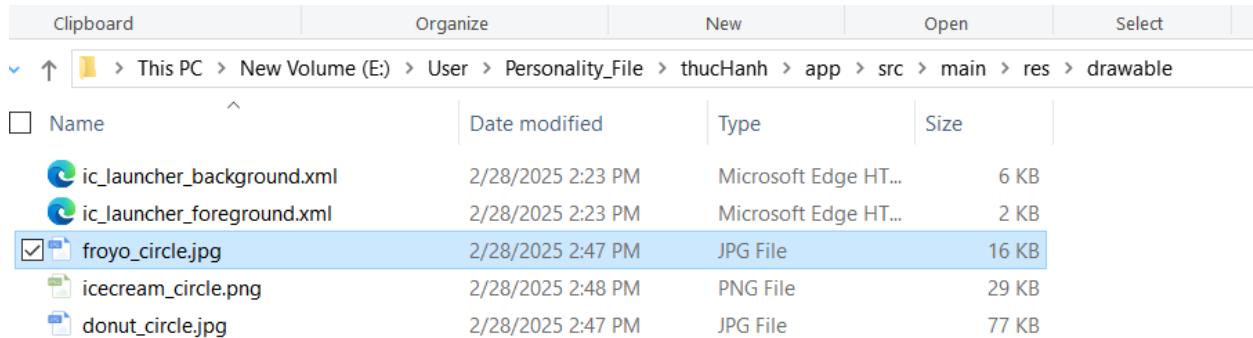
1.2 Thêm hình ảnh

Ba hình ảnh (**donut_circle.png**, **froyo_circle.png**, và **icecream_circle.png**) được cung cấp cho ví dụ này bạn có thể tải xuống. Ngoài ra, bạn cũng có thể thay thế

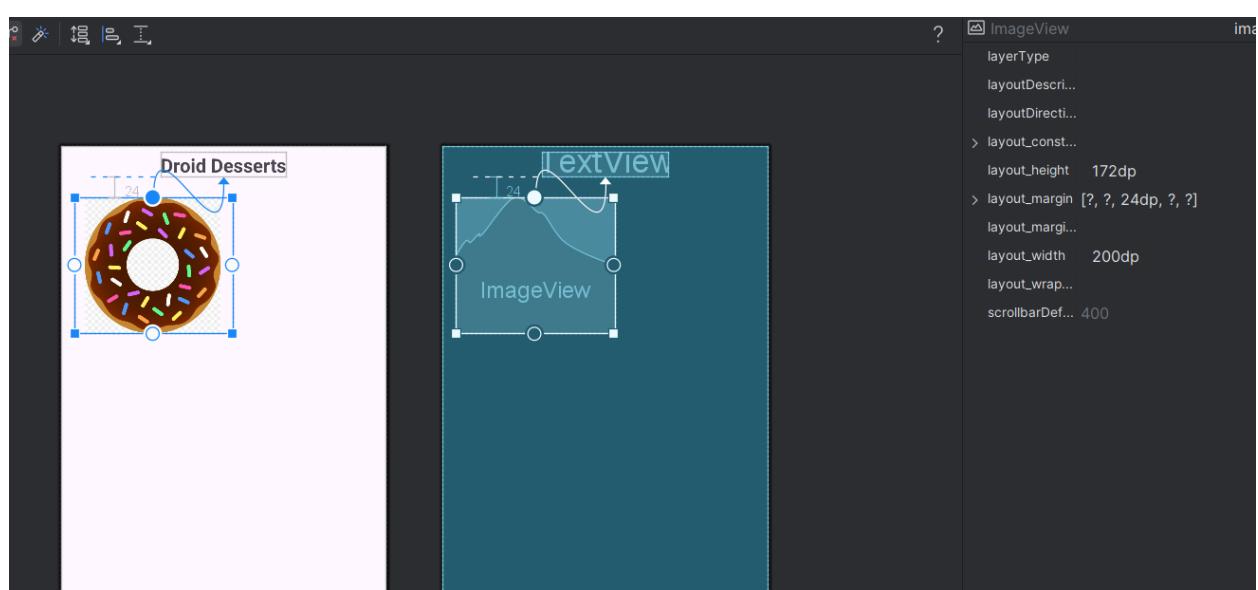
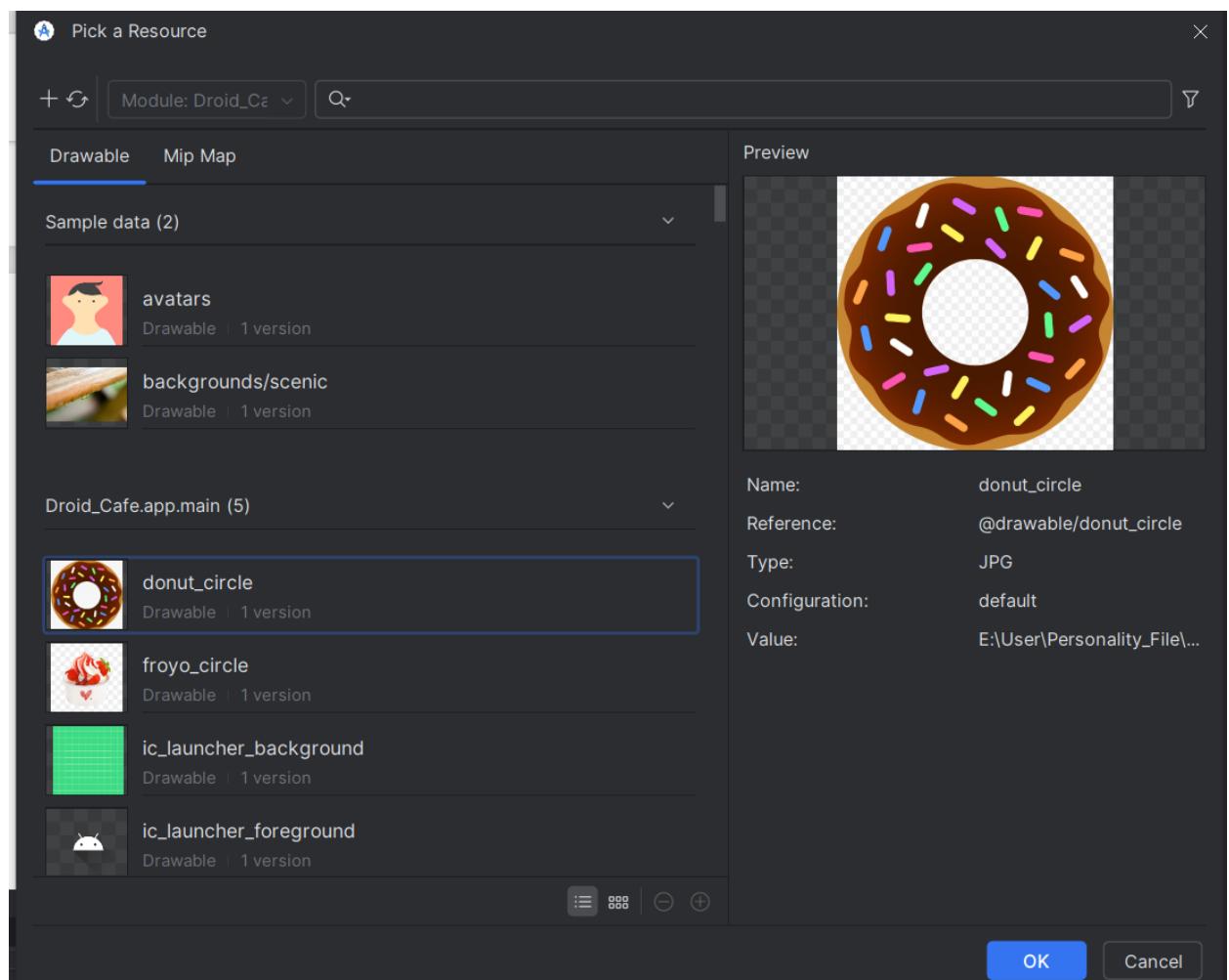
bằng hình ảnh của riêng mình dưới dạng tệp **PNG**, nhưng chúng phải có kích thước khoảng **113 x 113 pixels** để sử dụng ví dụ này.

Bước này cũng giới thiệu một kỹ thuật mới trong trình chỉnh sửa bối cảnh: sử dụng nút **Fix** trong thông báo cảnh báo để trích xuất **string resources**.

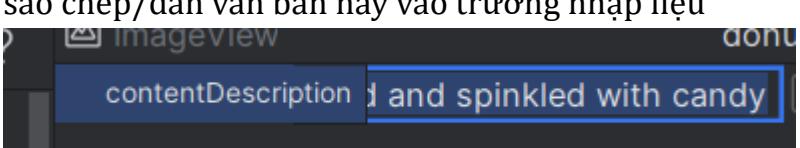
1. Để sao chép hình ảnh vào dự án, đầu tiên đóng dự án.
2. Sao chép các tệp hình ảnh vào thư mục **drawable** theo đường dẫn:
project_name > app > src > main > res > drawable



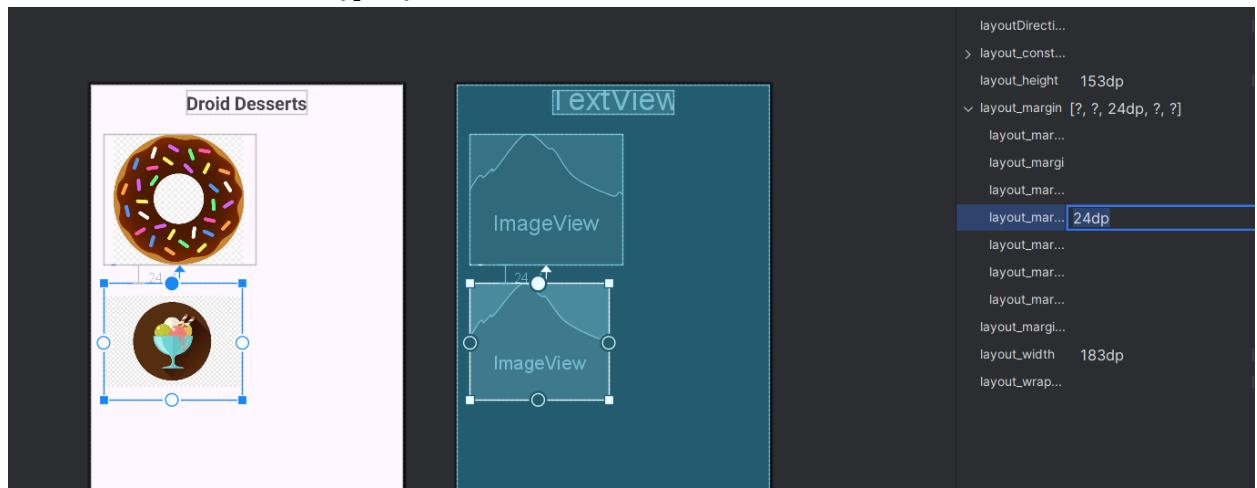
3. Mở lại dự án.
4. Mở tệp **content_main.xml** và nhấp vào tab **Design** (nếu chưa được chọn).
5. Kéo một **ImageView** vào bối cảnh. Chọn hình ảnh **donut_circle** cho **ImageView**. Ràng buộc **ImageView** với **TextView** phía trên. Ràng buộc **ImageView** với cạnh trái của bối cảnh. Đặt **khoảng cách (margin)** là **24dp** cho cả hai ràng buộc.



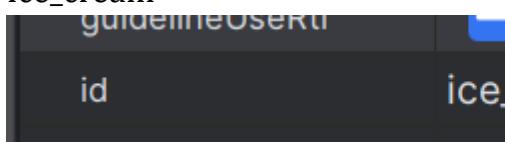
6. Trong trang Thuộc tính (Attributes), điền giá trị thuộc tính:

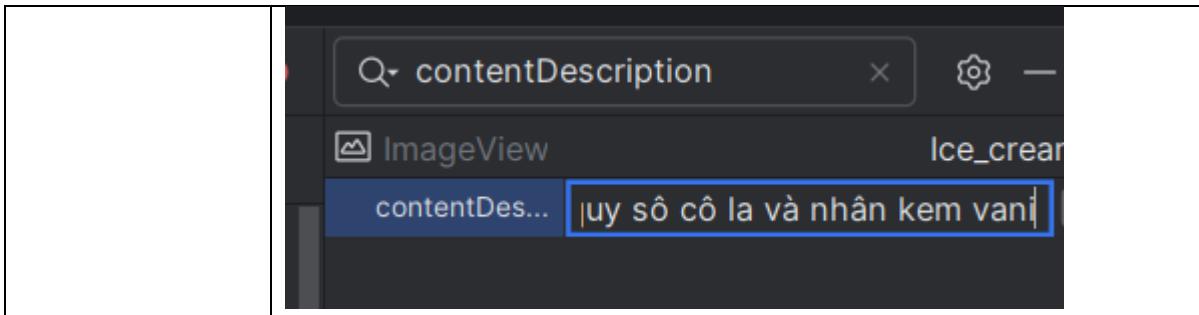
Trường thuộc tính	Điền như sau
ID	donut 
contentDescription	Bánh rán được phủ lớp đường và rắc kẹo (Bạn có thể sao chép/dán văn bản này vào trường nhập liệu) 

7. Kéo ImageView thứ hai vào bố cục, chọn ảnh icecream_circle, và ràng buộc nó với cạnh dưới của ImageView đầu tiên và cạnh trái của bố cục với khoảng cách (**margin**) là **24dp** cho cả hai ràng buộc. Thay đổi các thuộc tính **textIntro** như sau. Các điều khiển nhập liệu

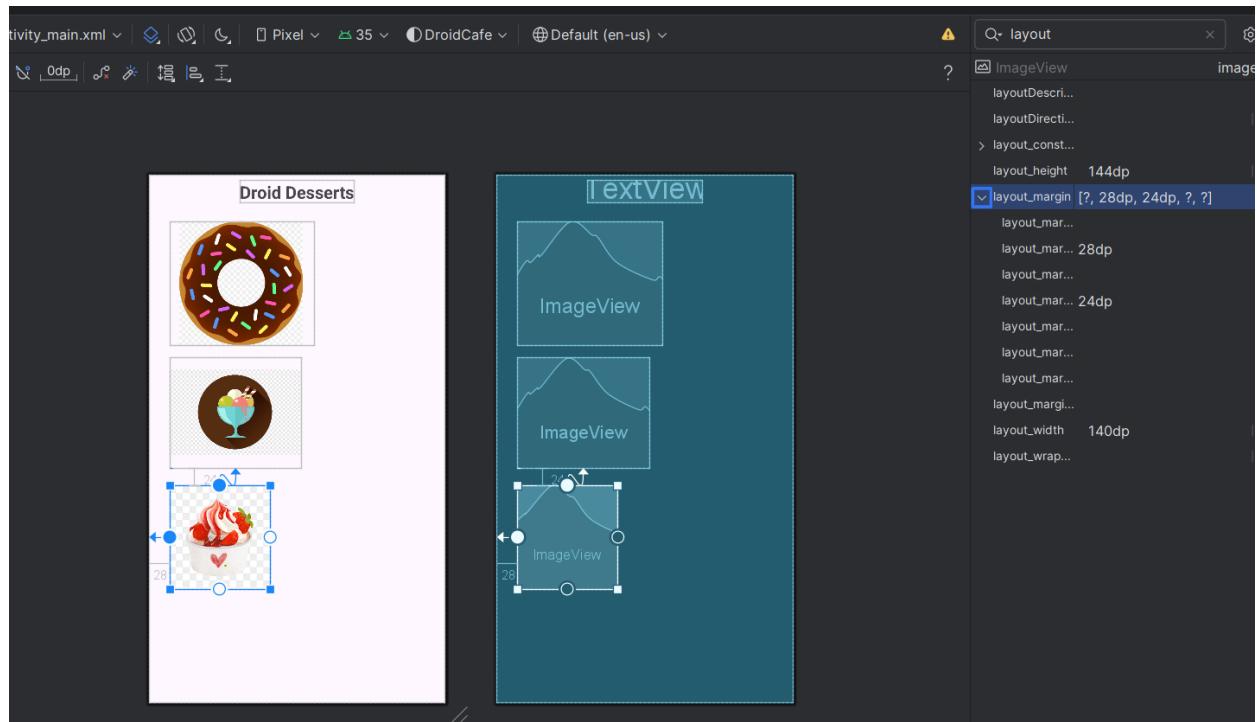


8. Trong trang Thuộc tính (Attributes), điền giá trị thuộc tính:

Trường thuộc tính	Điền như sau
ID	Ice_cream 
contentDescription	Bánh sandwich kem có lớp bánh quy sô cô la và nhân kem vani (Bạn có thể sao chép/dán văn bản này vào trường nhập liệu)

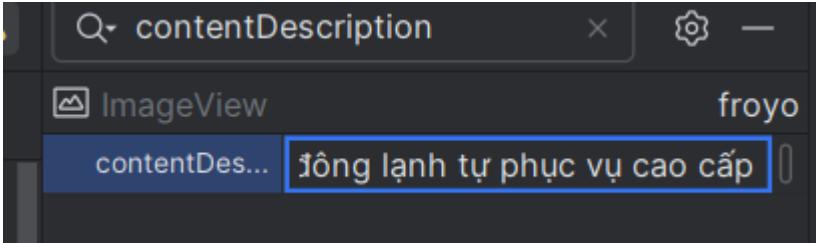


9. Kéo một **ImageView** thứ ba vào bố cục, chọn hình ảnh **froyo_circle** cho nó, và ràng buộc nó với cạnh dưới của **ImageView** thứ hai và cạnh trái của bố cục với khoảng cách (**margin**) là **24dp** cho cả hai ràng buộc.

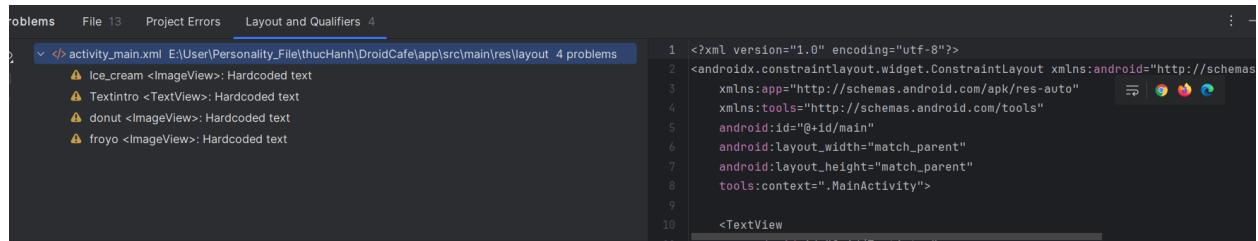


10. Trong trang Thuộc tính (Attributes), điền giá trị thuộc tính:

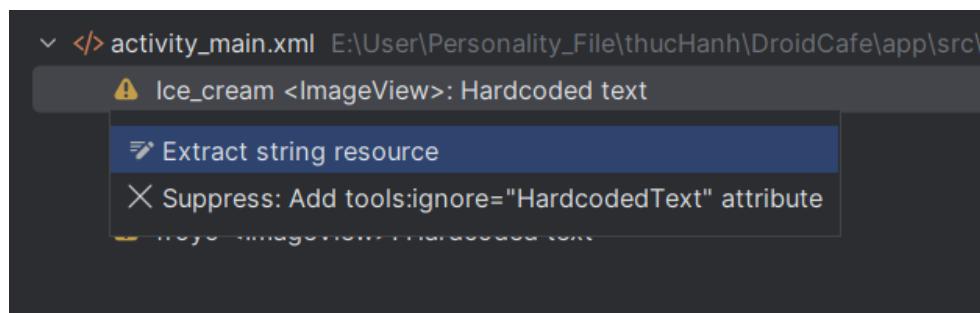
Trường thuộc tính	Điền như sau
ID	Froyo

	id	froyo
contentDescription	FroYo là sữa chua đông lạnh tự phục vụ cao cấp (Bạn có thể sao chép/dán văn bản này vào trường nhập liệu)	

11. Nhấp vào biểu tượng cảnh báo ở góc trên bên trái của **trình chỉnh sửa bối cục** để mở **bảng cảnh báo**, nơi sẽ hiển thị các cảnh báo về văn bản được mã hóa cứng (**hardcoded text**).



12. Mở rộng mỗi cảnh báo Hardcore text, cuộn xuống dưới cuối của thông báo cảnh báo và nhấn vào nút Fix (sửa) được hiển thị như bên dưới:



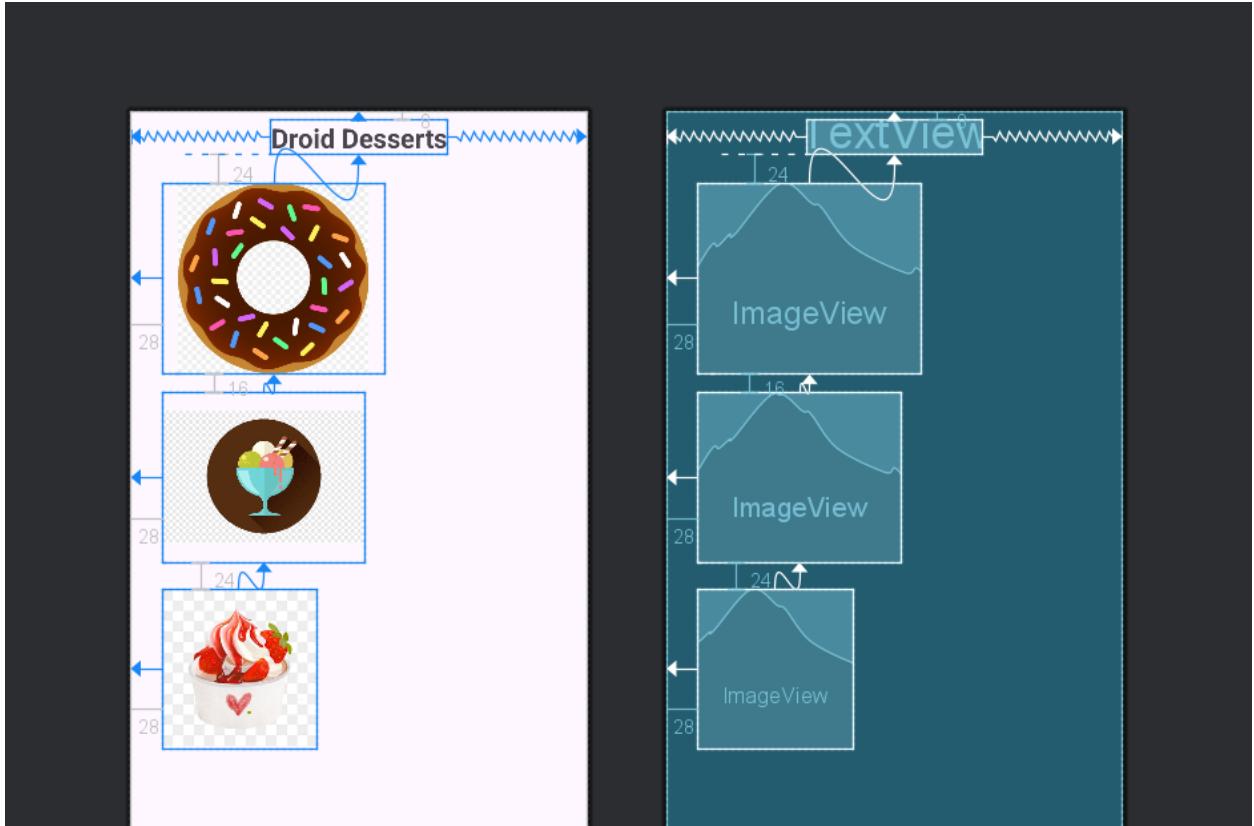
Bản sửa lỗi cho từng cảnh báo về văn bản được mã hóa cứng sẽ trích xuất chuỗi văn bản thành một tài nguyên chuỗi. Hộp thoại **Extract Resource** sẽ xuất hiện, và bạn có thể nhập tên cho tài nguyên chuỗi.

Nhập các tên sau cho các tài nguyên chuỗi:

Chuỗi	Điền tên như sau:
Bánh donut được phủ lớp đường và rắc kẹo	donuts
Bánh sandwich kem có lớp bánh quy	ice_cream_sandwiches

sô cô la và nhân vani	
FroYo là sữa chua đông lạnh tự phục vụ cao cấp	froyo

Bố cục sẽ trông như các hình bên dưới.



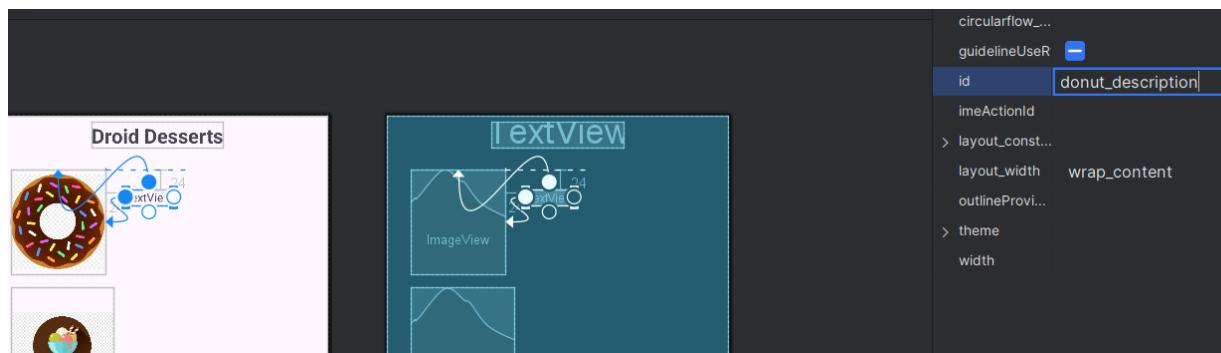
1.3 Thêm mô tả:

Trong bước này, bạn thêm một mô tả văn bản (**TextView**) cho mỗi món tráng miệng. Vì bạn đã trích xuất các tài nguyên chuỗi cho các trường **contentDescription** của các phần tử **ImageView**, bạn có thể sử dụng cùng một tài nguyên chuỗi cho mỗi **TextView** mô tả.

1. Kéo một phần tử **TextView** vào bố cục.
2. Ràng buộc cạnh trái của phần tử vào cạnh phải của **ImageView** donut và cạnh trên của nó vào cạnh trên của **ImageView** donut, cả hai với lề **24dp**.

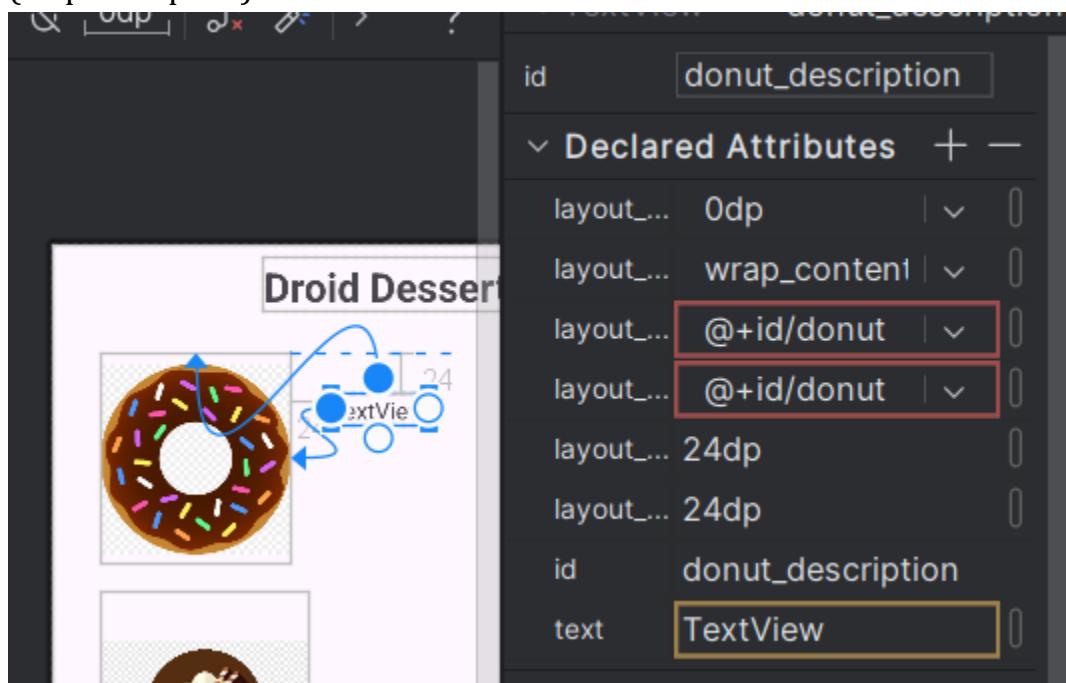


Ràng buộc cạnh phải của phần tử vào cạnh phải của bố cục và sử dụng cùng một lề **24dp**. Nhập **donut_description** vào trường **ID** trong ngăn **Attributes**.

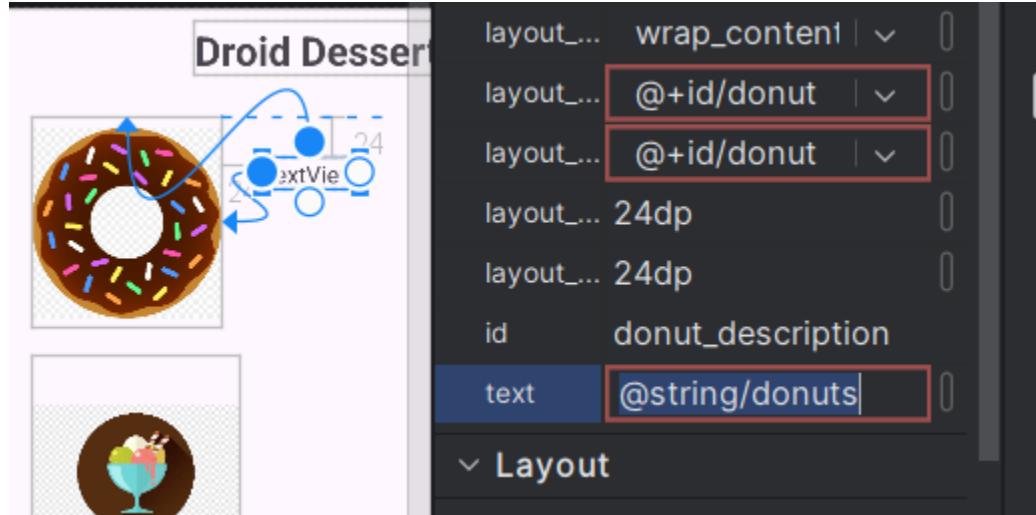


TextView mới sẽ xuất hiện bên cạnh hình ảnh donut như hiển thị trong hình dưới đây.

- Trong trang **Attributes**, thay đổi thuộc tính **width** trong bảng kiểm tra (inspector pane) thành **Match Constraints**.



5. Trong trang **Attributes**, bắt đầu nhập tài nguyên chuỗi cho trường văn bản bằng cách thêm ký hiệu @ phía trước: @d. Nhập vào tên tài nguyên chuỗi xuất hiện dưới dạng gợi ý (@string/donuts).



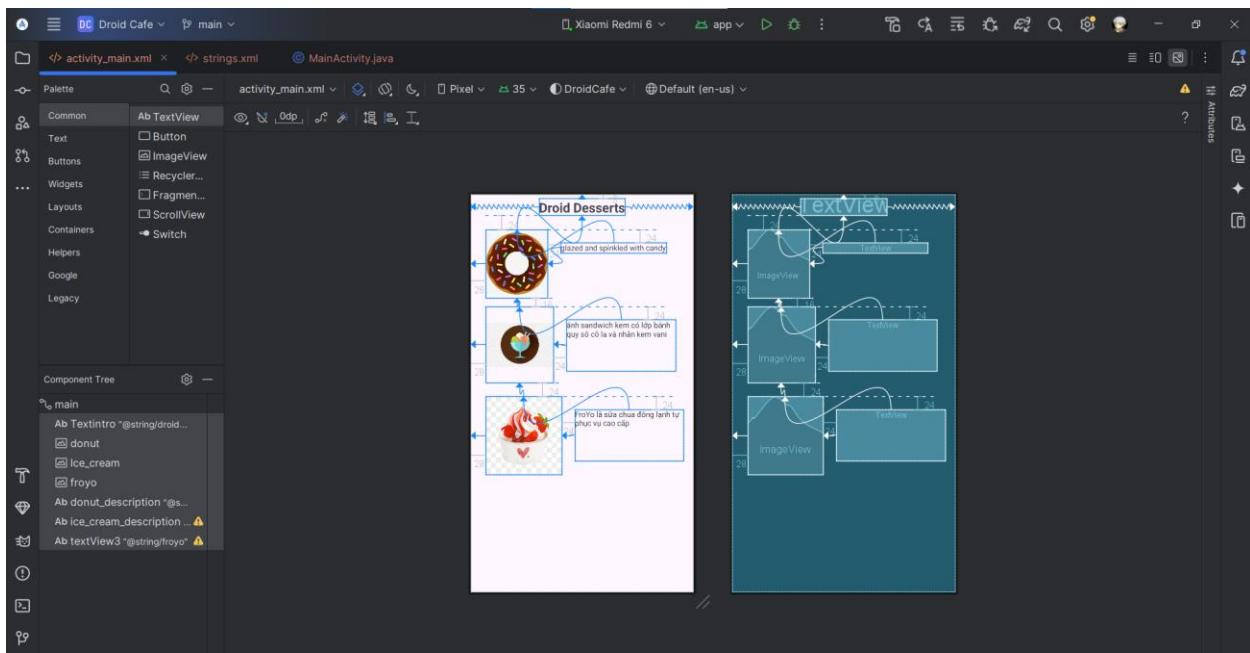
6. Lặp lại các bước trên để thêm một **TextView** thứ hai, được ràng buộc với cạnh phải và cạnh trên của **ImageView** ice_cream, và cạnh phải của nó được ràng buộc với cạnh phải của bố cục. Nhập các giá trị sau vào trang **Attributes**:

Trường thuộc tính	Điền như sau:
ID	ice_cream_description
Left, right, and top margins	24
layout_width	match_constraint
text	@string/ice_cream_sandwiches

7. Lặp lại các bước trên để thêm một **TextView** thứ ba, được ràng buộc với cạnh phải và cạnh trên của **ImageView** froyo, và cạnh phải của nó được ràng buộc với cạnh phải của bố cục. Nhập các giá trị sau vào trang **Attributes**:

Trường thuộc tính	Điền như sau:
ID	froyo_description
Left, right, and top margins	24
layout_width	match_constraint
text	@string/froyo

Bố cục trông sẽ như hình bên dưới:



Nhiệm vụ 1: Mã giải quyết

Bố cục XML cho tập tin **content.xml** được hiển thị bên dưới.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/Textintro"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/droid_deserts"
        android:textSize="24sp" />
```

```
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
<ImageView
    android:id="@+id/donut"
    android:layout_width="114dp"
    android:layout_height="126dp"
    android:layout_marginStart="28dp"
    android:layout_marginTop="24dp"
    android:contentDescription="@string/donuts"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/Textintro"
    app:srcCompat="@drawable/donut_circle" />
```

```
<ImageView
    android:id="@+id/ice_cream"
    android:layout_width="125dp"
    android:layout_height="140dp"
    android:layout_marginStart="28dp"
    android:layout_marginTop="16dp"
    android:contentDescription="@string/ice_cream_sandwiches"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/donut"
    app:srcCompat="@drawable/icecream_circle" />
```

```
<ImageView
    android:id="@+id/froyo"
    android:layout_width="140dp"
    android:layout_height="144dp"
    android:layout_marginStart="28dp"
    android:layout_marginTop="24dp"
    android:contentDescription="@string/froyo"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/ice_cream" />
```

```
        app:srcCompat="@drawable/froyo_circle" />
```

```
<TextView  
    android:id="@+id/donut_description"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="24dp"  
    android:layout_marginTop="24dp"  
    android:text="@string/donuts"  
    app:layout_constraintStart_toEndOf="@+id/donut"  
    app:layout_constraintTop_toTopOf="@+id/donut" />
```

```
<TextView  
    android:id="@+id/ice_cream_description"  
    android:layout_width="200dp"  
    android:layout_height="94dp"  
    android:layout_marginStart="24dp"  
    android:layout_marginTop="24dp"  
    android:text="@string/ice_cream_sandwiches"  
    app:layout_constraintStart_toEndOf="@+id/Ice_cream"  
    app:layout_constraintTop_toTopOf="@+id/Ice_cream" />
```

```
<TextView  
    android:id="@+id/textView3"  
    android:layout_width="200dp"  
    android:layout_height="94dp"  
    android:layout_marginStart="24dp"  
    android:layout_marginTop="24dp"  
    android:text="@string/froyo"  
    app:layout_constraintStart_toEndOf="@+id/froyo"  
    app:layout_constraintTop_toTopOf="@+id/froyo" />
```

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

Nhiệm vụ 2: Thêm phương thức onClick cho các hình ảnh

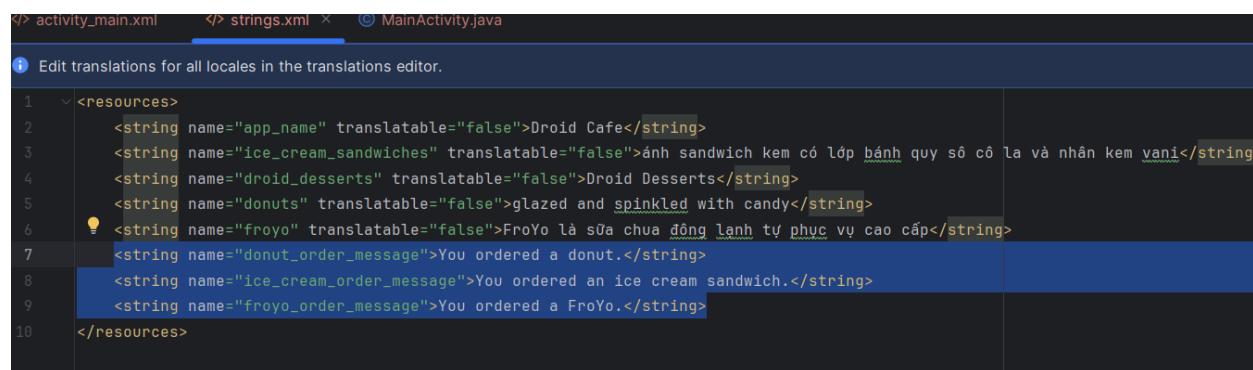
Để tạo một **View clickable** (có thể nhấp) mà người dùng có thể nhấp (hoặc nhấn), thêm thuộc tính **android:onClick** trong bố cục XML và trình xử lý nhấp chuột. Ví dụ, bạn có thể làm một ImageView như là một **Button**(nút) đơn giản bằng cách thêm **android:onClick** vào **ImageView**, trong nhiệm vụ này bạn sẽ làm hình ảnh trong bố cục của bạn có thể ấn được.

2.1 Tạo một phương thức Toast

Trong nhiệm vụ này bạn thêm mỗi phương thức cho mỗi thuộc tính **android:onClick** để gọi khi mỗi ảnh được nhấp. Trong nhiệm vụ này, những phương thức sẽ hiển thị đơn giản một thông báo Toast ở bức ảnh mà được nhấp vào. (Trong một chương khác, bạn sẽ chỉnh sửa các phương thức này để khởi chạy một Activity khác)

- Để sử dụng tài nguyên chuỗi (string resources) trong mã Java, trước tiên bạn cần thêm chúng vào tệp strings.xml.

Mở rộng **res > values** trong **Project > Android** pane, sau đó mở tệp **strings.xml**. Thêm các tài nguyên chuỗi sau để hiển thị trong thông báo **Toast**:



```
<resources>
    <string name="app_name" translatable="false">Droid Cafe</string>
    <string name="ice_cream_sandwiches" translatable="false">ánh sandwich kem có lớp bánh quy sô cô la và nhân kem vani</string>
    <string name="droid_desserts" translatable="false">Droid Desserts</string>
    <string name="donuts" translatable="false">glazed and spinkled with candy</string>
    <string name="froyo" translatable="false">FroYo là sữa chua đông lạnh tự phục vụ cao cấp</string>
    <string name="donut_order_message">You ordered a donut.</string>
    <string name="ice_cream_order_message">You ordered an ice cream sandwich.</string>
    <string name="froyo_order_message">You ordered a FroYo.</string>
</resources>
```

- Mở tệp **MainActivity.java**, sau đó thêm phương thức **displayToast()** vào cuối **MainActivity** (trước dấu ngoặc nhọn đóng } của lớp).

```
> import ...

<> public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: 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;
        });
    }

    no usages
    public void displayToast(String message) {
        Toast.makeText(getApplicationContext(), message,
            Toast.LENGTH_SHORT).show();
    }
}
```

Mặc dù bạn có thể thêm phương thức này ở bất kỳ vị trí nào trong **MainActivity**, nhưng theo **best practice**, bạn nên đặt các phương thức tự định nghĩa bên dưới các phương thức đã có sẵn trong **MainActivity** do mẫu tạo ra.

2.2 Tạo trình xử lý nhấp chuột

Mỗi hình ảnh có thể nhấp cần một trình xử lý nhấp chuột—một phương thức để thuộc tính android:onClick gọi. Trình xử lý nhấp chuột, nếu được gọi từ thuộc tính android:onClick, phải là **public**, trả về **void**, và nhận một **View** làm tham số duy nhất.

Hãy làm theo các bước sau để thêm trình xử lý nhấp chuột:

1. Thêm phương thức showDonutOrder() sau vào **MainActivity**. Trong nhiệm vụ này, sử dụng phương thức displayToast() đã tạo trước đó để hiển thị một thông báo **Toast**.

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

    no usages
    public void showDonutOrder(View view) {
        displayToast(getString(R.string.donut_order_message));
    }
}
```

Ba dòng đầu tiên là một chú thích theo định dạng Javadoc, giúp mã dễ hiểu hơn và cũng hỗ trợ tạo tài liệu cho mã của bạn. Đây là một **best practice** (Cách thực hành tốt nhất) để thêm chú thích như vậy vào mỗi phương thức mới mà bạn tạo.

Để biết thêm thông tin về cách viết chú thích, hãy xem **How to Write Doc Comments for the Javadoc Tool**.

2. Thêm các phương thức sau vào cuối **MainActivity** cho từng loại món tráng miệng:

```
    /**
     * Shows a message that the ice cream sandwich image was clicked.
     */
    no usages
    public void showIceCreamOrder(View view) {
        displayToast(getString(R.string.ice_cream_order_message));
    }
    /**
     * Shows a message that the froyo image was clicked.
     */
    no usages
    public void showFroyoOrder(View view) {
        displayToast(getString(R.string.froyo_order_message));
    }
```

3. (Tùy chọn) Chọn **Code > Reformat Code** để định dạng lại mã bạn đã thêm trong **MainActivity** theo tiêu chuẩn và làm cho nó dễ đọc hơn.

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) 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. LUU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

1.1) Shared preferences

Giới thiệu

Shared preferences cho phép bạn lưu trữ 1 lượng nhỏ primitive data theo kiểu cặp key/value trong tệp trên thiết bị. Để lấy 1 tệp preference và để đọc, viết và quản lý preference data, sử dụng lớp SharedPreferences. Android framework quản lý tệp sharedpreferences của nó. Tệp này cho phép truy cập bởi mọi thành phần trong ứng dụng của bạn, nhưng mà nó không cho ứng dụng khác truy cập vào.

Dữ liệu của bạn lưu ở Shared preferences khác với dữ liệu trong saved activity state, chúng tôi đã đề cập đến saved activity state ở chương trước.

- Dữ liệu trong saved activity instance state được dũ lại trong cùng phiên làm việc của người dùng.

- Shared preferences tồn tại trong suốt phiên làm việc của người dùng. Shared preferences được dũ lại ngay cả khi nếu ứng dụng của bạn dừng và khởi động lại hoặc nếu thiết bị khởi động lại

Sử dụng Shared preferences chỉ khi bạn cần lưu 1 lượng nhỏ dữ liệu như là cặp key/value đơn giản. Để quản lý lượng lớn dữ liệu lưu trữ lâu dài trong ứng dụng, sử dụng phương pháp lưu trữ khác như là Room library hoặc SQL Database.

CÓ THỂ BẠN ĐÃ BIẾT

Bạn nên thành thạo về:

- Tạo, xây và chạy ứng dụng trên android studio
- Thiết kế giao diện với buttons và text views
- Lưu và khôi phục activity instance state.

Bạn sẽ học về?

Bạn sẽ được học cách:

Xác định Shared preferences là gì

Tạo tệp Shared preferences cho ứng dụng

Lưu dữ liệu vào Shared preferences và đọc lại dữ liệu đó

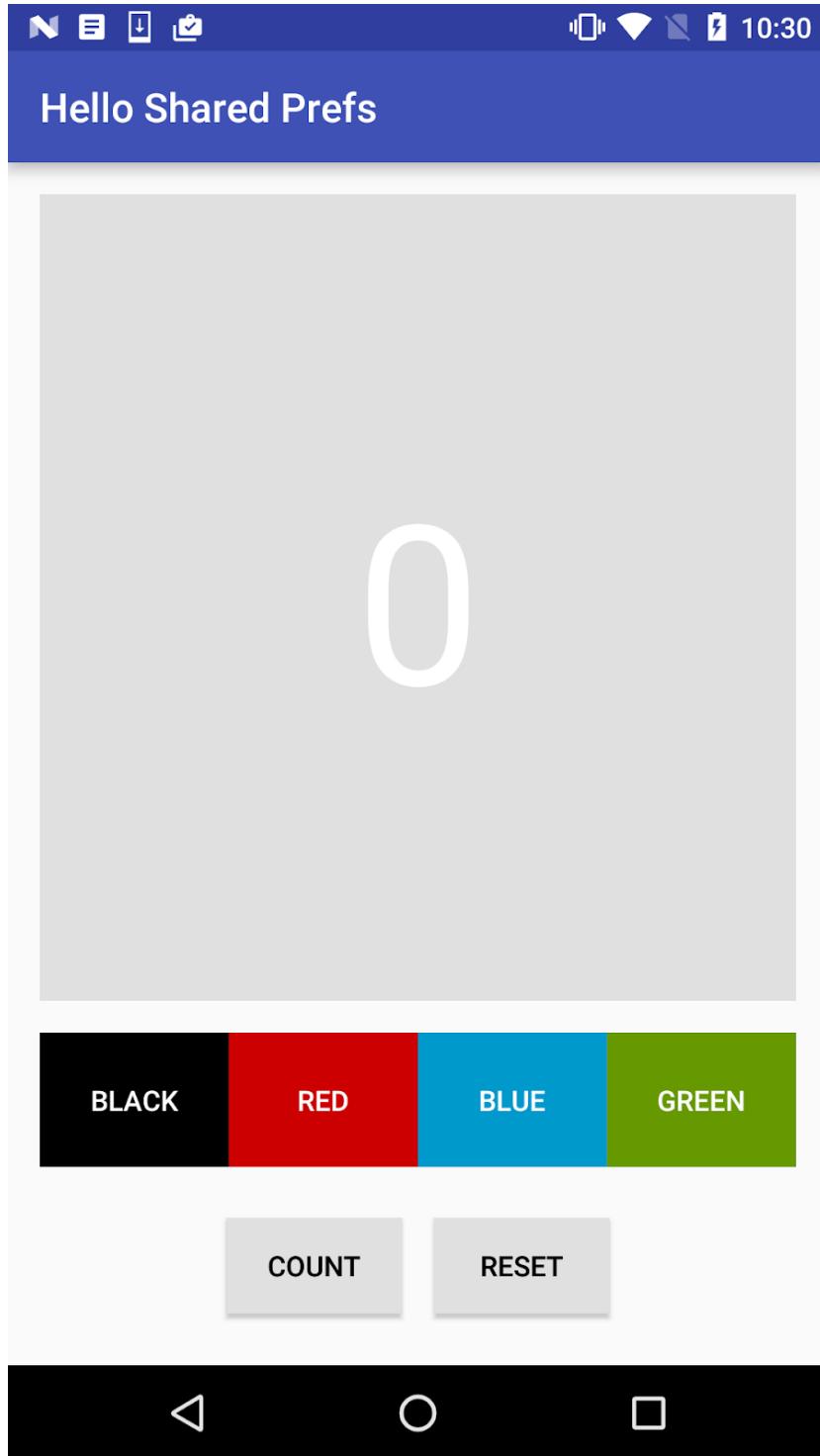
Xóa dữ liệu trong Shared preferences.

Bạn sẽ làm gì?

Nâng cấp ứng dụng để nó có thể lưu, truy xuất hoặc đặt lại Shared preferences.

Tổng quan về ứng dụng

Ứng dụng HelloSharedPrefs là 1 biến thể khác của ứng dụng HelloToast bạn đã tạo ở bài 1. nó có những buttons để tăng số count, thay đổi màu nền và khởi tạo lại count và màu về mặt định. Ứng dụng cũng sử dụng themes và style để định nghĩa buttons.



Bắt đầu với ứng dụng khởi tạo và thêm shared preferences vào main activity code. Bạn cũng thêm cả button “Reset” để thiết lập count và màu nền về mặc định và xóa hết tệp preferences.

Nhiệm vụ 1: khám phá HelloSharedPrefs

Project ứng dụng starter cho bài thực hành này có sẵn tại HelloSharedPrefs-Srarter. Trong nhiệm vụ này, bạn sẽ tải project into Android Studio và khám phá và đặc điểm của key trong ứng dụng.

1.1 Mở và chạy project HelloSharedPrefs-Srarter

1. Tải ứng dụng HelloSharedPrefs-Srarter và giải nén tệp.
2. Mở project trên Android Studio sau đó xây và chạy thử ứng dụng. Thử những điều sau:
 - Bấm vào nút “count” thì số trong text view chính sẽ tăng lên.
 - Bấm bất kì nút màu để thay đổi màu nền và text view chính cũng sẽ thay đổi
 - Xoay thiết bị và ghi chú rằng màu nền và count vẫn được giữ nguyên.
 - Bấm button “Reset” để cài lại màu và count về mặc định.
 - Buộc dừng ứng dụng sử dụng 1 trong những phương pháp sau:
 - Trong android, chọn Run > Stop ‘app’ hoặc ấn vào biểu tượng Stop  ở thanh công cụ.
 - Trên thiết bị, ấn vào nút quay lại (nút hình vuông ở góc dưới bên phải, vuốt thẻ ứng dụng HelloSharedPrefs để thoát hoặc là ấn vào X trên góc phải của thẻ. Nếu bạn thoát ứng dụng theo cách này, hãy đợi vài giây để hệ thống dọn dẹp trước khi khởi động lại
3. Chạy lại ứng dụng:

Khi bạn khởi động lại ứng dụng, giao diện mặc định sẽ hiển thị - count là 0 và màu nền là màu xám.

1.2 Khám phá Activity code

1. Mở MainActivity
2. Xem xét mã nguồn và lưu ý những điểm sau:

- Biến đếm (mCount) đã được định nghĩa ở kiểu integer. Phương thức onClick countUp() được gọi khi nhấn vào nút Count, giúp tăng giá trị này và cập nhật TextView chính.
- Màu nền (mColor) cũng là kiểu integer, ban đầu được đặt là màu xám từ tệp colors.xml với tên default_background.
- Phương thức changeBackground() được gọi khi nhấn vào một trong các màu sau đó sẽ đặt text view chính theo màu đó.
- Phương thức onSaveInstanceState() và khôi phục trong phương thức onCreate(). Các khóa(key) bundle cho count và màu được định nghĩa bởi các biến private COUNT_KEY và COLOR_KEY.

Nhiệm vụ 2: Lưu và khôi phục lại dữ liệu của tệp shared preferences

Trong nhiệm vụ này, bạn sẽ lưu lại trạng thái của ứng dụng ở tệp shared preferences và đọc dữ liệu lại mỗi khi ứng dụng khởi động lại. Bởi vì trạng thái dữ liệu mà bạn lưu ở shared preferences (nơi lưu count và color gần đây) trùng với dữ liệu mà bạn lưu trữ trong instance state nên không cần phải lưu 2 lần mà thay thế hoàn toàn instance state bằng shared preferences

2.1 Bắt đầu shared preferences:

1. Thêm biến vào lớp MainActivity để lưu tên của tệp shared preferences và tham chiếu đến đối tượng shared preferences.

```
private SharedPreferences mPreferences;
private String sharedPrefFile =
    "com.example.android.hellosharedprefs";
```

Bạn có thể đặt tên tệp shared preferences của bạn tùy thích nhưng theo quy ước thì phải trùng tên với tên package của ứng dụng.

2. Trong phương thức onCreate(), khởi tạo đối tượng shared preferences bằng cách thêm dòng mã sau trước câu lệnh if:

statement:

```
mPreferences = getSharedPreferences(sharedPrefFile, MODE_PRIVATE);
```

Phương thức getSharedPreferences (từ Context của activity) mở tệp tại tên tệp được chỉ định (sharePrefFile) với chế độ MODE_PRIVATE.

LƯU Ý: các phiên bản android cũ hơn đã có các chế độ khác mà cho phép bạn tạo tệp preferences có thể đọc/ghi bởi các ứng dụng khác. Các chế độ này đã bị loại bỏ từ API 17 và hiện nay không được khuyến nghị vì lý do bảo mật. Nếu bạn cần chia sẻ dữ liệu với các ứng dụng khác, hãy cân nhắc sử dụng content URI do FileProvider cung cấp.

Đoạn mã giải quyết vấn đề cho MainActivit, chỉ có 1 phần được hiển thị thôi:

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

    private SharedPreferences mPreferences;
    private String sharedPrefFile =
        "com.example.android.hellosharedprefs";

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

        mShowCount = (TextView) findViewById(R.id.textview);
        mColor = ContextCompat.getColor(this,
            R.color.default_background);
        mPreferences = getSharedPreferences(
            sharedPrefFile, MODE_PRIVATE);

        // ...
    }
}
```

2.2 Lưu preferences trong onPause()

Lưu preferences khá giống với lưu instance state - cả 2 thao tác đều đặt dữ liệu dưới dạng cặp key/value vào 1 đối tượng Bundle. Tuy nhiên, bạn lưu dữ liệu đó trong callback vòng đời onPause() và cần 1 đối tượng editor (SharedPreferences.Editor) để ghi vào đối tượng Shared Preferences.

1. Thêm phương thức vòng đời onPause vào MainActivity

```
@Override  
protected void onPause() {  
    super.onPause();  
  
    // ...  
}
```

2. Trong onPause(), lấy 1 đối tượng editor cho đối tượng SharedPreferences:

```
SharedPreferences.Editor preferencesEditor = mPreferences.edit();
```

Một editor của preferences được chấp thuận để viết vào đối tượng shared preferences. Thêm dòng này vào onPause() sau khi gọi đến super.onPause().

3. Sử dụng phương thức putInt() để đẩy kiểu integer của mCount và mColor đến shared preferences với các khóa tương ứng:

```
preferencesEditor.putInt(COUNT_KEY, mCount);  
preferencesEditor.putInt(COLOR_KEY, mColor);
```

Lớp SharedPreferences.Editors bao gồm đa phương thức “đẩy” cho những kiểu dữ liệu khác nhau bao gồm puInt() và puString().

4. Gọi apply() để lưu preferences:

```
preferencesEditor.apply();
```

Phương thức apply() lưu shared preferences 1 cách bất đồng bộ, ngoài luồng giao diện người dùng(UI thread). Đối tượng Shared preferences, editor cũng có phương thức commit() để lưu preferences 1 cách đồng bộ. Phương thức commit() không được khuyến khích vì nó có thể chặn các thao tác khác.

1. Xóa toàn bộ phương thức onSaveInstanceState(). Bởi vì ví activity instance state bao gồm dữ liệu giống như shared preferences, bạn có thể thay thế hoàn toàn instance state bằng shared preferences.

Code giải pháp cho phương thức onPause(): trong MainActivity

```
@Override  
protected void onPause() {  
    super.onPause();  
  
    SharedPreferences.Editor preferencesEditor = mPreferences.edit();  
    preferencesEditor.putInt(COUNT_KEY, mCount);  
    preferencesEditor.putInt(COLOR_KEY, mColor);  
    preferencesEditor.apply();  
}
```

2.3 Phục hồi preferences trong onCreate()

Cũng giống như instance state, phần mềm của bạn đọc tất cả shared preferences trong phương thức onCreate(). 1 lần nữa, bởi vì shared preferences bao gồm dữ liệu giống như instance state, chúng ta có thể thay thế instance state bằng shared preferences ở đây. Mỗi lần onCreate() được gọi - khi ứng dụng khởi động hoặc khi có thay đổi cấu hình - shared preferences sẽ được sử dụng để khôi phục trạng thái của giao diện.

1. Xác định phần trong phương thức onCreate() mà thử nếu đối số savedInstanceState là null và phục hồi instance state:

```
if (savedInstanceState != null) {  
    mCount = savedInstanceState.getInt(COUNT_KEY);  
    if (mCount != 0) {  
        mShowCountTextView.setText(String.format("%s", mCount));  
    }  
    mColor = savedInstanceState.getInt(COLOR_KEY);  
    mShowCountTextView.setBackgroundColor(mColor);  
}
```

2. Xóa toàn bộ khôi đó

3. Trong phương thức onCreate(), tại vị trí tương tự như phần code instance state, lấy biến đếm từ trong preferences với COUNT_KEY và gán cho nó biến mCount.

```
mCount = mPreferences.getInt(COUNT_KEY, 0);
```

Khi bạn đọc dữ liệu từ preferences bạn không cần lấy shared preferences editor. Sử dụng bất cứ phương thức get nào trong đối tượng shared preferences (như là getInt() hoặc getString()) để truy xuất dữ liệu preference.

Lưu ý rằng phương thức getInt() lấy 2 đối số: 1 cho key và 1 cho giá trị mặc định nếu key không tìm thấy. Trong trường hợp này giá trị mặc định là 0, giống với khởi tạo của mCount.

4. Cập nhật giá trị của TextView chính với count mới:

```
mShowCountTextView.setText(String.format("%s", mCount));
```

5. Lấy màu từ preferences bằng key COLOR_KEY và gán cho nó biến mColor
variable.

```
mColor = mPreferences.getInt(COLOR_KEY, mColor);
```

Trước đó, đối số getInt() đặt là giá trị mặc định để sử dụng trong trường hợp khóa không tồn tại trong shared preferences. Trong trường hợp này bạn có thể chỉ sử dụng lại giá trị mColor (đã được khởi tạo với nền mặc định trong phương thức).

6. Cập nhật màu nền của main text view

... Update the background color of the main text view...

```
mShowCountTextView.setBackgroundColor(mColor);
```

7. Chạy thử ứng dụng, nhập vào nút Count và thay đổi màu nền để cập nhật instance state và preferences.

8. Quay thiết bị hoặc máy ảo để phản hồi số đếm và màu đã được lưu lại khi có thay đổi cấu hình.

9. Buộc dừng ứng dụng bằng một trong các phương thức:

- Trong Android Studio, chọn Run > Stop ‘app’

- Trên thiết bị, nhấn nút Recents (nút hình vuông ở phía dưới góc phải). Vuốt thẻ ứng dụng HelloSharedPrefs để thoát ứng dụng hoặc nhấn vào X trên góc phải của thẻ.

10. Chạy lại ứng dụng. Ứng dụng khởi động lại và tải preferences, giữ nguyên các trạng thái.

Mã nguồn giải pháp cho phương thức onCreate() của MainActivity:

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

    // Initialize views, color, preferences
    mShowCountTextView = (TextView) findViewById(R.id.count_textview);
    mColor = ContextCompat.getColor(this, R.color.default_background);
    mPreferences = getSharedPreferences(
        mSharedPrefFile, MODE_PRIVATE);

    // Restore preferences
    mCount = mPreferences.getInt(COUNT_KEY, 0);
    mShowCountTextView.setText(String.format("%s", mCount));
    mColor = mPreferences.getInt(COLOR_KEY, mColor);
    mShowCountTextView.setBackgroundColor(mColor);
}
```

2.4 Đặt lại preferences trong trình xử lý sự kiện reset()

Nút “đặt lại” trong ứng dụng ban đầu sẽ cài lại cả số đếm và màu cho activity về giá trị mặc định. Bởi vì preferences giữ trạng thái của activity, nó rất quan trọng khi xóa cả preferences cùng lúc.

1. Trong phưorc thức reset() của sự kiện onClick, sau khi màu và số đếm đã được đặt lại, lấy editor của đối tượng SharedPreferences:

```
SharedPreferences.Editor preferencesEditor = mPreferences.edit();
```

2. Xóa tất cả shared preferences

```
preferencesEditor.clear();
```

1. Áp dụng sự thay đổi:

```
preferencesEditor.apply();
```

Đoạn mã giải quyết cho phương thức reset():

```
public void reset(View view) {
    // Reset count
    mCount = 0;
    mShowCountTextView.setText(String.format("%s", mCount));

    // Reset color
    mColor = ContextCompat.getColor(this, R.color.default_background);
    mShowCountTextView.setBackgroundColor(mColor);
```

```
// Clear preferences
SharedPreferences.Editor preferencesEditor = mPreferences.edit();
preferencesEditor.clear();
preferencesEditor.apply();

}
```

Mã giải quyết

Android Studio project: HelloSharedPrefs

Thách thức:

Lưu ý: chỉnh sửa ứng dụng HelloSharedPrefs để thay vì tự động hóa lưu trạng thái của tệp preferences, thêm activity thứ 2 để thay đổi, cài lại những preferences. Thêm nút có tên Settings để khởi chạy activity đó, bao gồm nút toggle và spinner để thay đổi preferences

Tổng kết

- lớp SharedPreferences chấp nhận ứng dụng lưu trữ lượng nhỏ dữ liệu primitive như là cặp key/value.
- Shared preferences được lưu trữ qua nhiều phiên làm việc của người dùng trong cùng 1 ứng dụng.
- Để ghi dữ liệu vào shared preferences, lấy đối tượng SharedPreferences.Editor
- Sử dụng các phương thức “put” trong đối tượng SharedPreferences.Editor, như là putInt() hoặc putString() để đẩy dữ liệu lên shared preferences với key và value.

- Sử dụng các phương thức “get” trong đối tượng SharedPreferences, như là getInt() hoặc getString() để lấy dữ liệu từ SharedPreferences bằng key.
- Sử dụng phương thức clear() trong đối tượng SharedPreferences.Editor để loại bỏ tất cả dữ liệu lưu trong preferences.
- Sử dụng phương thức apply() trong đối tượng SharedPreferences.Editor để lưu sự thay đổi trong tệp preferences.

Khái niệm liên quan

Tài liệu liên quan nằm trong 9.0 Data storage và 9.1 Shared preferences.

Tìm hiểu thêm

Tài liệu dành cho nhà phát triển android:

- Data and file storage overview
- Save key-value data
- SharedPreferences
- SharedPreferences.Editor

Stack Overflow:

- How to use SharedPreferences in Android to store, fetch and edit values
- onSavedInstanceState vs. SharedPreferences

BÀI TẬP VỀ NHÀ

Xây dựng và chạy 1 ứng dụng

Mở ứng dụng ScoreKeeper mà bạn đã tạo trong Android fundamentals 5.1: Drawables, style and themes lesson

1. Thay intance state đã lưu bằng shared preferences với mỗi điểm.
2. Chạy thử ứng dụng, quay thiết bị để đảm bảo cấu hình thay đổi đọc và lưu preferences và cập nhật UI
3. Dừng ứng dụng và khởi tạo lại nó để đảm bảo rằng preferences đã được lưu

4. Thêm nút Reset (nút cài lại tất giá trị điểm thành 0 và xóa shared preferences)

Trả lời những câu hỏi sau

Câu hỏi 1:

Bạn lưu trạng thái của ứng dụng vào shared preferences trong phương thức vòng đời nào?

Câu hỏi 2:

Bạn khôi phục trạng thái ứng dụng bằng phương thức vòng đời nào?

Câu hỏi 3:

Bạn có nghĩ ra trường hợp nếu nó tạo cả preferences và instance state là hợp lý không?

Nội dung của bạn cho người chấm điểm

Hướng dẫn cho người chấm điểm

- Ứng dụng giữ nguyên điểm số khi quay thiết bị
- Ứng dụng giữ nguyên điểm số sau khi ứng dụng bị dừng hoặc khởi động lại
- Ứng dụng lưu điểm của shared preferences trong phương thức onPause().
- Ứng dụng khôi phục shared preferences bằng phương thức onCreate()
- Ứng dụng hiển thị nút Reset để cài lại điểm = 0

Đảm bảo rằng việc triển khai phương thức xử lý sự kiện nhấn(on-click handler) thỏa mãn những điều sau:

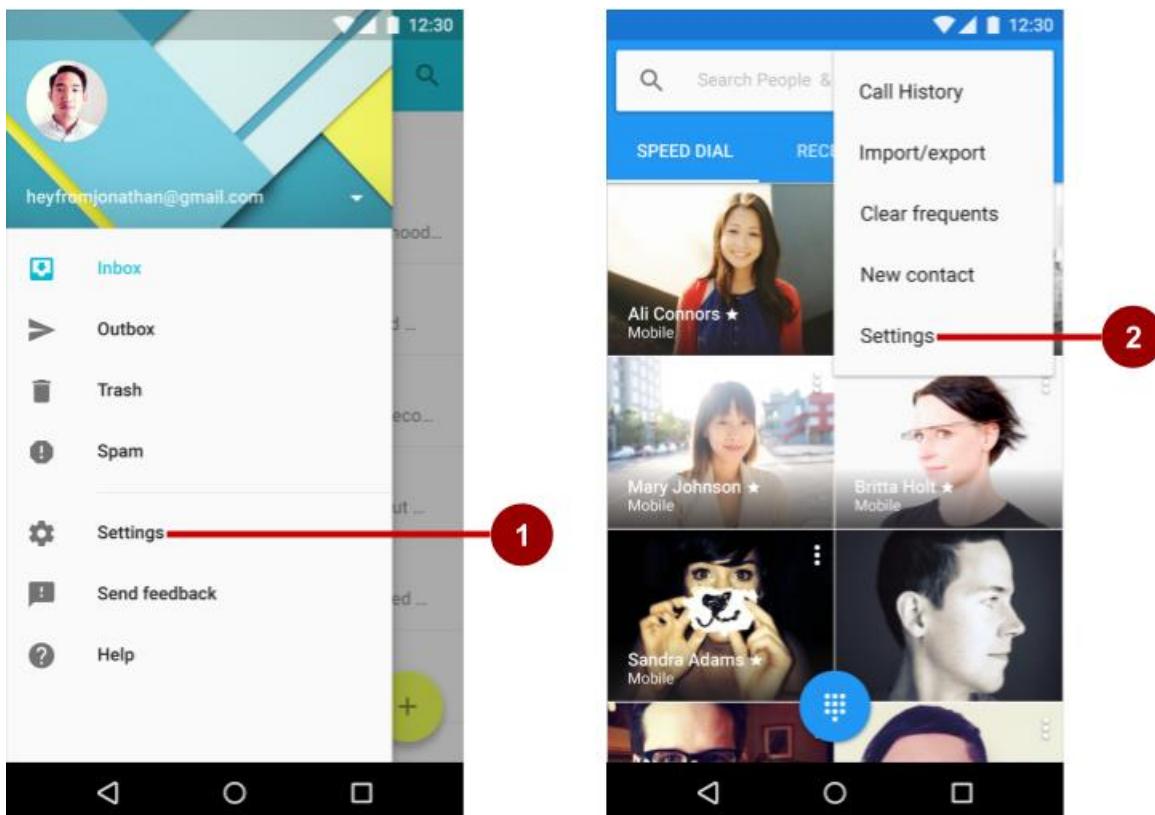
- Đặt lại cả 2 biến số về 0
- Cập nhật cả 2 text views
- Xóa shared preferences

BÀI 9.2 App settings

Giới thiệu

Ứng dụng thường bao gồm các cài đặt để cho phép người dùng sửa các tính năng và hành vi của ứng dụng. Ví dụ như vài ứng dụng cho phép người dùng cài địa điểm nhà, đơn vị mặc định, thước đo và các cài đặt khác áp dụng cho toàn bộ ứng dụng. Người dùng không truy cập cài đặt thường xuyên, bởi vì 1 khi người dùng thay đổi cài đặt, như là địa điểm nhà, họ hiếm khi cần thay đổi nó 1 lần nữa.

Người dùng mong đợi có thể truy cập cài đặt ứng dụng bằng cách nhấn vào Settings trong menu điều hướng, như là thanh điều hướng trượt (navigation drawer) như trong hình bên trái dưới đây, hoặc menu lựa chọn ở thanh ứng dụng, hiện lên ở phía phải màn hình dưới đây.



Trong bức ảnh phía trên

1. Settings trong menu điều hướng (navigation drawer)

2. Settings trong menu tùy chọn trên thanh ứng dụng

Trong bài thực hành này bạn sẽ thêm settings activity vào ứng dụng. Người dùng có thể truy cập cài đặt bằng cách bấm Settings (Cái mà được đặt ở menu tùy chọn trong thanh ứng dụng (app bar)).

Kiến thức yêu cầu:

- Tạo project Android Studio từ template và giao diện chính
- Chạy ứng dụng trên máy ảo hoặc máy thật được kết nối
- Tạo và sửa các phần tử UI bằng layout editor hoặc XML.
- Trích xuất tài nguyên chuỗi và sửa giá trị chuỗi
- Truy cập phần tử UI từ code của bạn bằng cách sử dụng findViewById()
- Xử lý sự kiện nhấn nút
- Hiển thị thông báo Toast
- Thêm 1 Activity vào ứng dụng
- Thêm menu options vào thanh ứng dụng (app bar)
- Thêm và chỉnh sửa các mục trong menu options
- Xử dụng styles và themes trong project
- Sử dụng SharedPreferences

Bạn sẽ được học

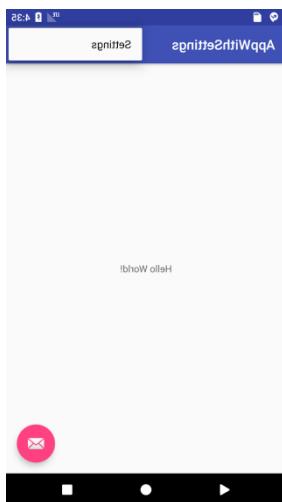
- Thêm Fragment cho quản lý cài đặt
- Tạo 1 tệp tài nguyên XML chứa các cài đặt cùng các thuộc tính của chúng
- Tạo điều hướng đến settings Activity
- Đặt giá trị mặc định cho settings
- Đọc giá trị settings thay đổi bởi người dùng
- Tùy chỉnh mẫu Settings Activity

Bạn sẽ làm gì?

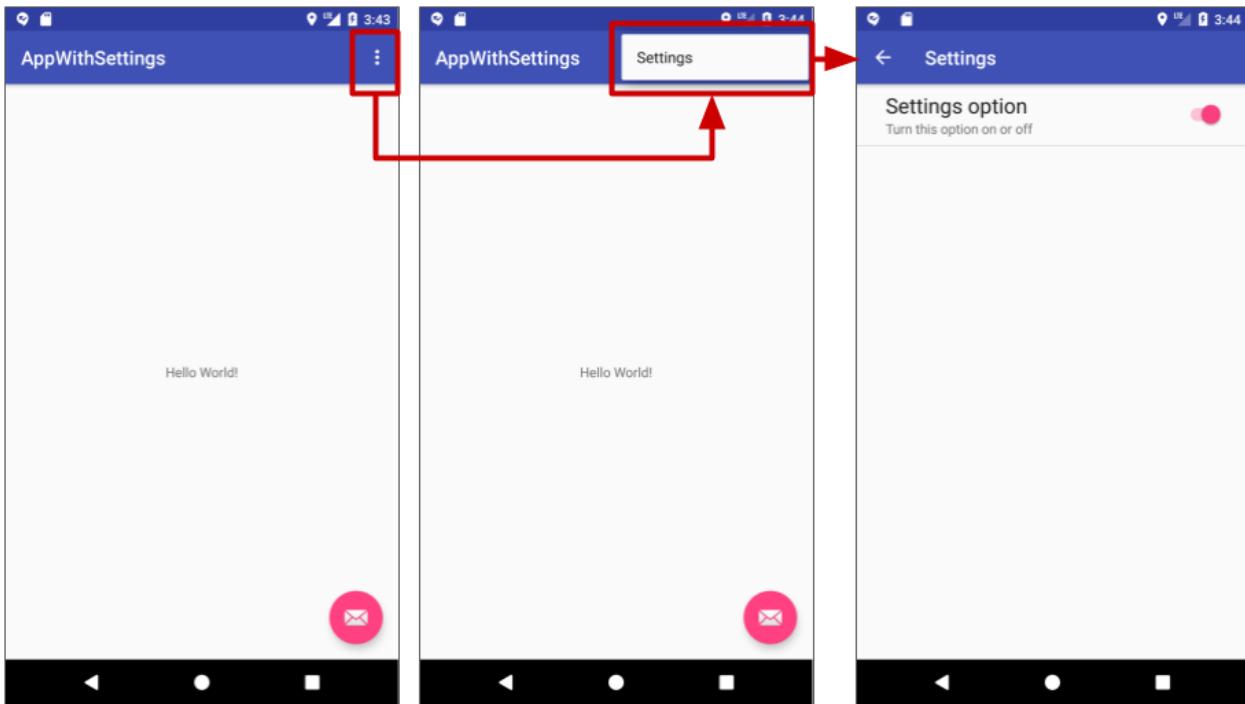
- Tạo 1 ứng dụng bao gồm Settings ở menu options
- Thêm công tắc chuyển đổi (Toggle switch) cho Settings option
- Thêm code để đặt giá trị mặc định cho cài đặt và truy cập giá trị setting sau khi nó đã được đổi.
- Sử dụng và tùy chỉnh mẫu Android Studio Settings Activity

Tổng quan ứng dụng

Android studio cung cấp 1 cách nhanh chóng cho thiết lập menu tùy chọn có chứa Settings. Nếu bạn bắt đầu 1 Android studio project cho điện thoại hoặc máy tính bảng sử dụng Basic Activity Template, Ứng dụng mới bao gồm Settings được trình bày ở phía dưới.



Mẫu này cũng bao gồm Floating Action Button ở góc dưới bên phải của màn hình với biểu tượng phong bì. Bạn có thể bỏ qua nút này trong phần thực hành này nếu bạn không muốn dùng tới nó. Bạn sẽ bắt đầu bằng cách tạo 1 ứng dụng có tên là AppWithSettings sử dụng Basic Activity template và bạn sẽ thêm settings Activity để cung cấp công tắc bật/tắt(toggle switch) để người dùng có thể bật/tắt nó.

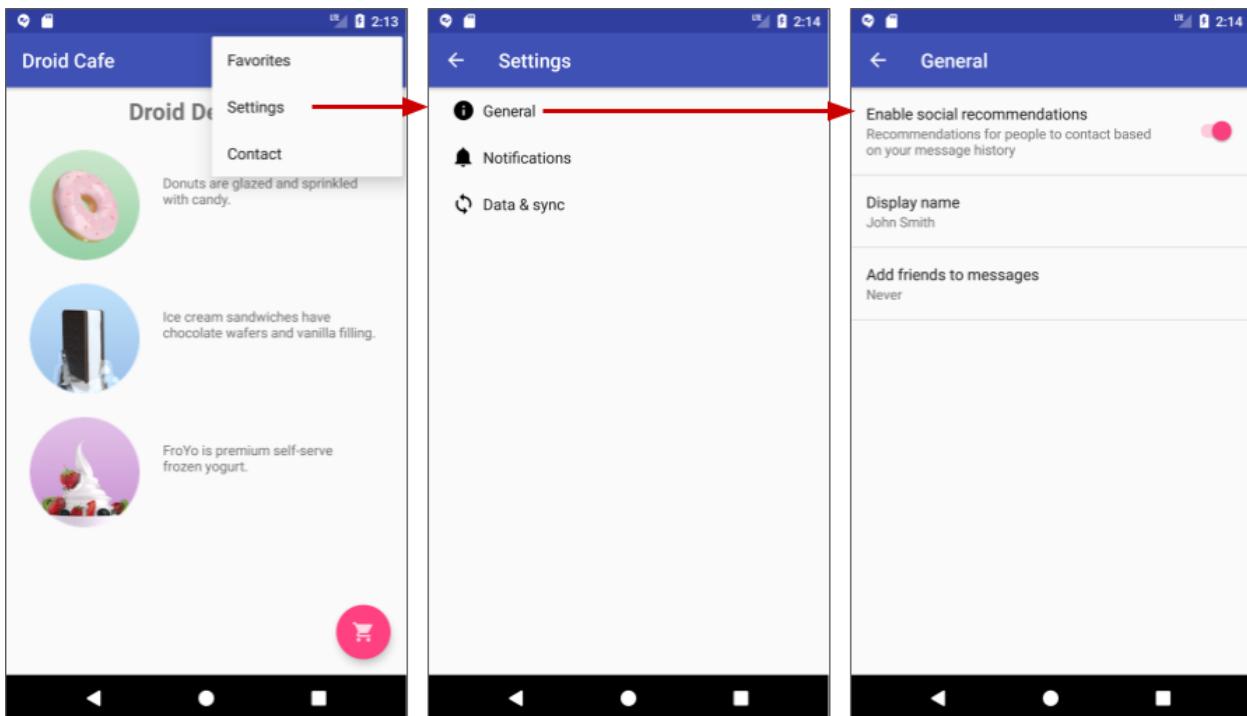


Bạn sẽ thêm code để đọc settings và thực hiện 1 hành động dựa trên giá trị của nó. Đơn giản như là hành động hiển thị thông báo Toast với giá trị của cài đặt.

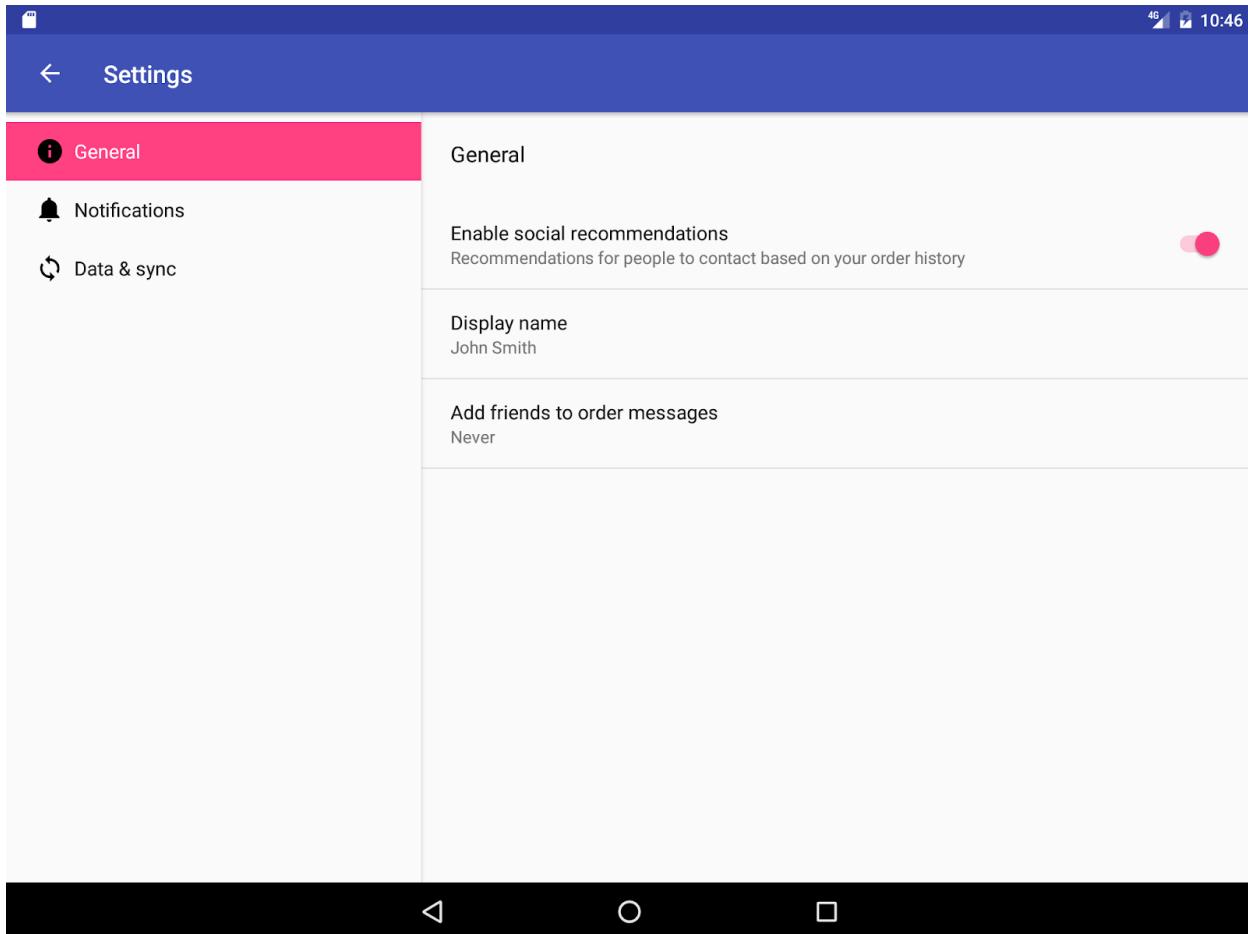
Trong nhiệm vụ thứ 2, bạn sẽ thêm mẫu Settings Activity được cung cấp bởi Android Studio cho ứng dụng DroidCafeOptionUp mà bạn đã tạo ở bài trước.

Mẫu Settings Activity được điền sẵn các cài đặt mà bạn có thể tùy chỉnh cho ứng dụng và cung cấp các bộ cục khác nhau cho điện thoại và cho máy tính bảng:

- Điện thoại: Màn hình Settings chính với 1 liên kết tiêu đề cho mỗi nhóm cài đặt, chẳng hạn như nhóm General cho cài đặt general, như được trình bày phía dưới



- Máy tính bảng: Giao diện màn hình bố trí theo kiểu master/detail với liên kết tiêu đề cho mỗi nhóm ở phía trái (master) và nhóm cài đặt ở bên phải (detail) được trình bày phía dưới



Để tùy chỉnh template này, bạn sẽ thay đổi headers, setting titles, setting descriptions và giá trị cho settings

Ứng dụng DroidCafeOptionUp đã được tạo trong bài trước từ Basic Activity template (cái mà cung cấp các tùy chọn menu trong ứng dụng trong app bar cho tùy chỉnh Settings option, bạn sẽ tùy chỉnh lại supplied Settings Activity template bằng cách thay đổi 1 cài đặt của title, description, giá trị và giá trị mặc định. Bạn sẽ thêm code để đọc giá trị của settings sau khi người dùng thay đổi nó và hiển thị giá trị đó.

Nhiệm vụ 1: thêm bộ chuyển đổi cho ứng dụng

Trong nhiệm vụ này, bạn sẽ được học những điều sau:

- Tạo mới khung project bằng Basic Activity template (thứ mà cung cấp option menu)
- Tạo nút công tắc (SwitchPreference) cùng các thuộc tính trong tệp preference XML

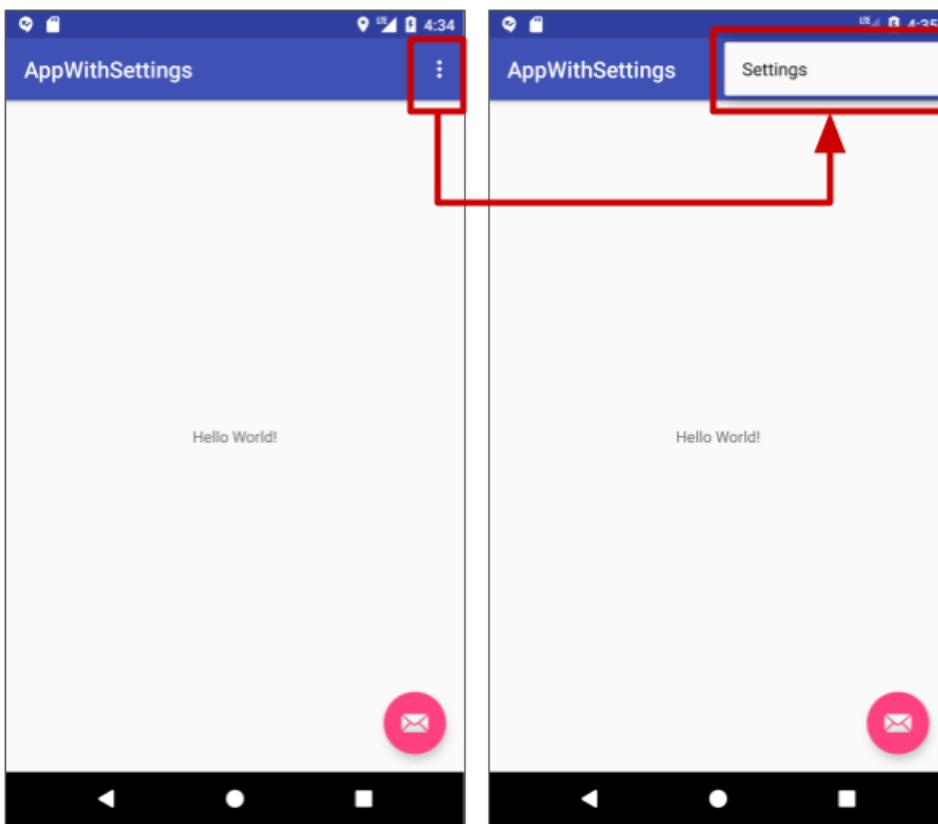
- Thêm activity cho settings và fragment cho cụ thể 1 cài đặt. Để đảm bảo khả năng tương thích với AppCompatActivity, bạn sử dụng PreferenceFragmentCompat nhiều hơn PreferenceFragment. Bạn cũng thêm thư viện android.support.v7.preference
- Kết nối mục Settings trong menu options với settings activity.

1.1 Tạo mới project và thêm thư mục XML cùng tệp tài nguyên

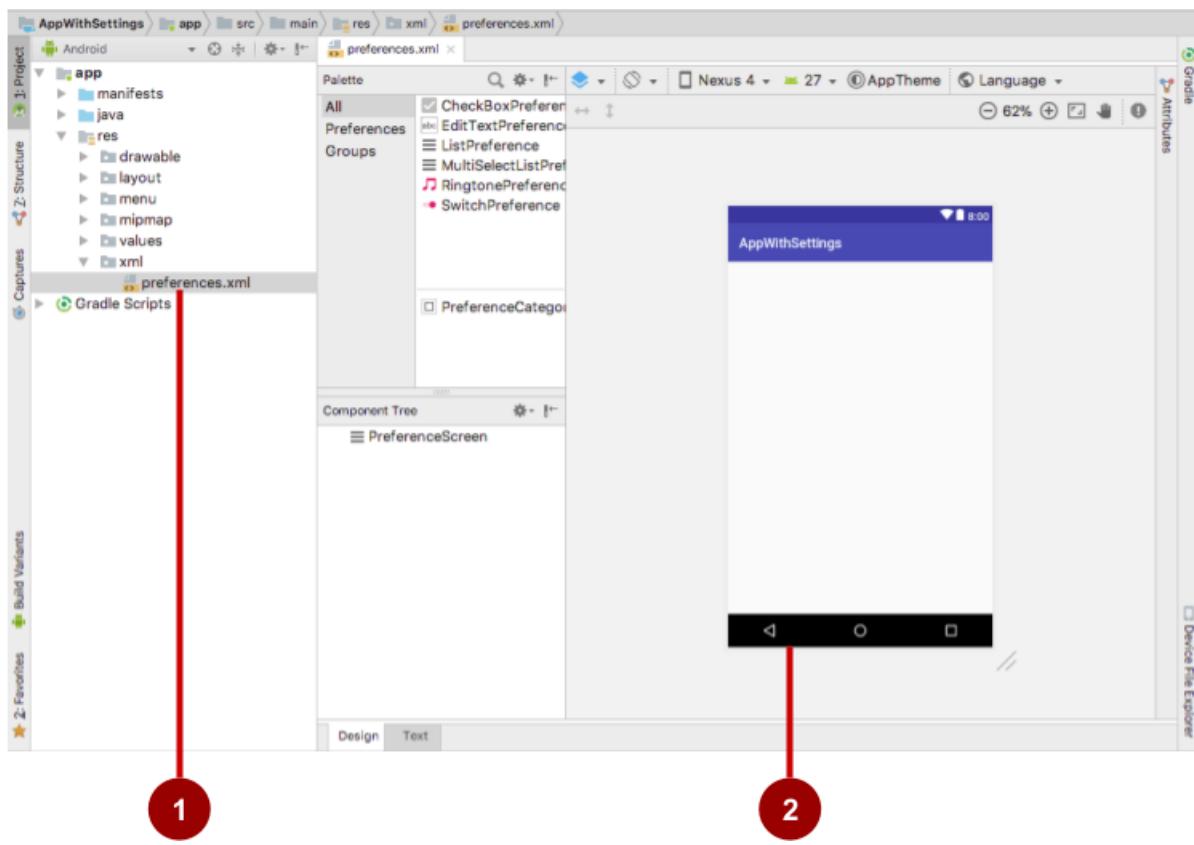
1. Trong Android Studio, tạo mới 1 project với những tham số sau:

Attribute	Value
Application Name	AppWithSettings
Company Name	Android.example.com (hoặc domain của bạn)
Project location	Đường dẫn đến thư mục các dự án của bạn
Phone and Tablet Minimum SDK	API15: Android 4.0.3 IceCreamSandwich
Template	Basic Activity
Activity Name	MainActivity
Layout Name	Activity_main
Title	MainActivity

2. Chạy ứng dụng và nhấn vào biểu tượng overflow trên app bar để xem menu tùy chọn, như hình bên dưới. Mục duy nhất trong menu tùy chọn là Settings.



3. Bạn cần tạo mới thư mục tài nguyên để giữ tệp XML bao gồm settings. Chọn thư mục res trong khung Project > Android và chọn File > New > Android Resource Directory. Hộp thoại New Resource Directory sẽ xuất hiện.
4. Trong Resource chọn drop-down menu, chọn xml. Directory name tự động chuyển sang xml. Bấm OK.
5. Thư mục xml xuất hiện trong bảng Project > Android trong thư mục res. Chọn xml và chọn File > New > XML resource file (hoặc chuột phải vào xml và chọn New > XML resource file)
6. Nhập tên của tệp XML là preferences, trong trường File name, và chọn OK. Tệp preferences.xml sẽ xuất hiện trong thư mục xml và layout editor sẽ xuất hiện, như hình bên dưới.

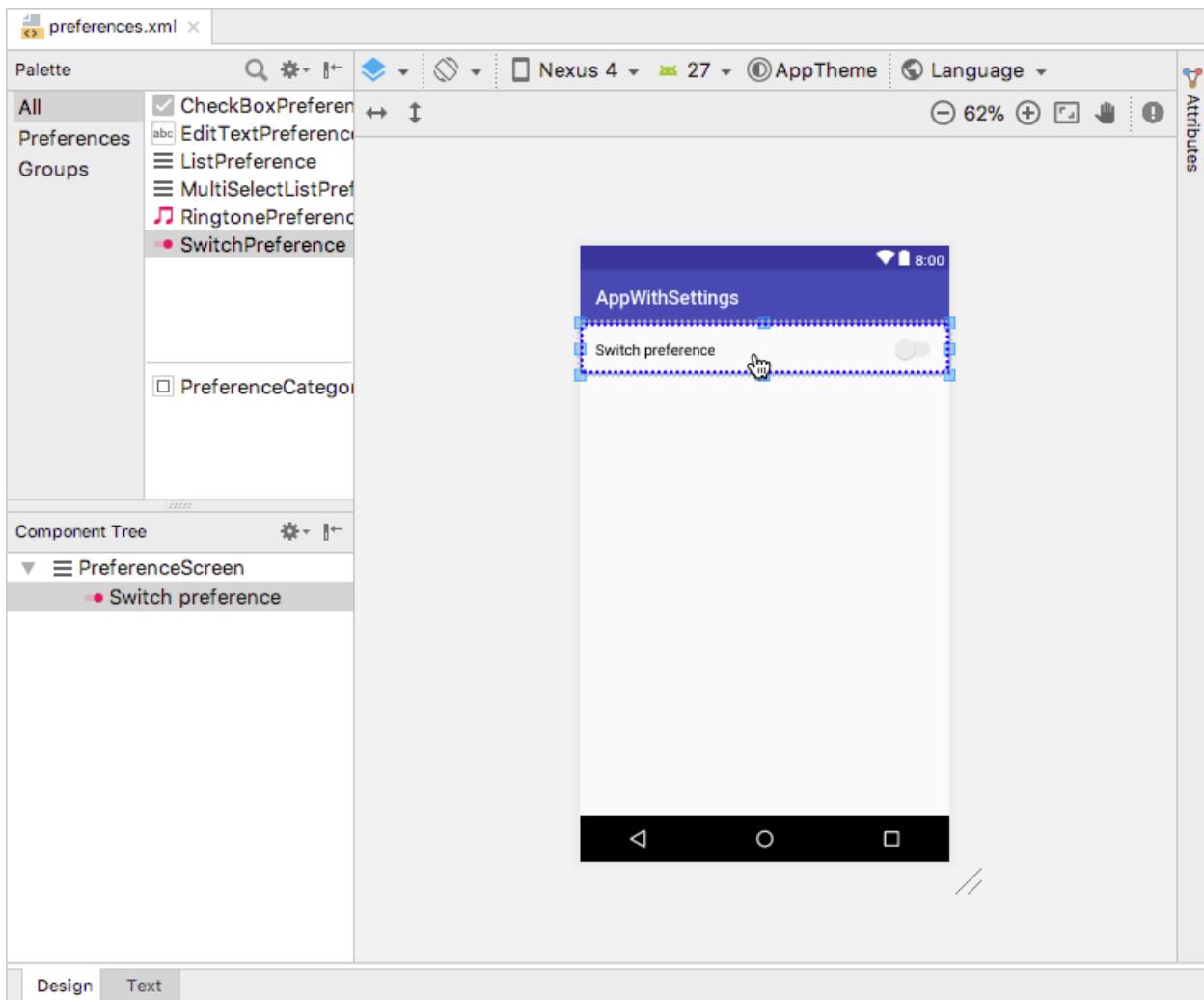


Trong ảnh trên:

1. Tệp preferences.xml trong thư mục (directory) xml.
2. Layout editor sẽ xuất hiện nội dung của preferences.xml

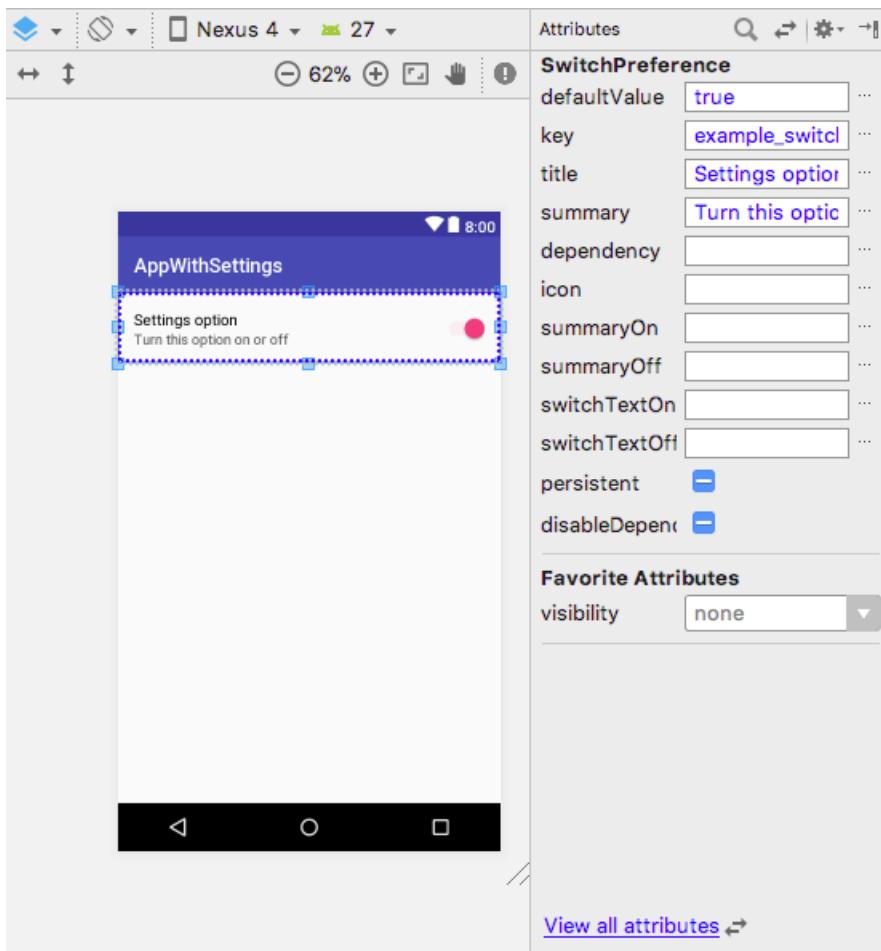
1.2 Thêm XML preferences và thuộc tính cho cài đặt

1. Kéo 1 SwitchPreference từ bảng Palette ở bên vào phía trên cùng của layout, như hình minh họa dưới đây.



2. Thay đổi giá trị trong bảng Attributes phía phải của layout editor như trong hình phía dưới:

- DefaultValue: true
- Key: example_switch
- Title: Settings option
- Summary: Turn this option on or off



3. Nhập vào tab Text ở dưới cùng của layout editor để xem XML code.

```

<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">

    <SwitchPreference
        android:defaultValue="true"
        android:key="example_switch"

        android:summary="Turn this option on or off"
        android:title="Settings option" />
</PreferenceScreen>

```

4. Trích xuất tài nguyên chuỗi cho các giá trị thuộc tính android:title và android:summary thành @string/switch_title và @string/switch_summary.

Thuộc tính XML cho các preference là:

- android:defaultValue: giá trị mặc định cho cài đặt khi ứng dụng khởi động lần đầu tiên.

- Android:title: tiêu đề của cài đặt cho SwitchPreference, tiêu đề xuất hiện ở phía trái của nút công tắc(toggle switch).
- Android:key: Khóa sử dụng để lưu trữ giá trị cài đặt. Mỗi cài đặt có một cặp key-value tương thích mà hệ thống sử dụng để lưu cài đặt trong tệp SharedPreferences mặc định cho cài đặt mà hệ thống sử dụng để lưu cài đặt ở tệp mặc định SharedPreferences cho ứng dụng cài đặt.
- Android:summary: Văn bản tóm tắt hiển thị bên dưới cài đặt

1.3 Sử dụng SwitchPreferenceCompat

Để sử dụng phiên bản PreferenceFragmentCompat của PreferenceFragment, bạn bắt buộc phải sử dụng phiên bản android.support.v7 của SwitchPreference(SwitchPreferenceCompat)

1. Trong bảng Project > Android, mở tệp build.gradle (Module:app) trong thư mục Gradle Scripts và thêm đoạn mã sau vào phần dependencies:

```
implementation 'com.android.support:preference-v7:26.1.0'
```

Câu lệnh được hiển thị ở bên trên thêm thư viện android.support.v7.preference để sử dụng phiên bản PreferenceFragmentCompat của PreferenceFragment.

2. Trong tệp preference.xml trong thư mục xml, thay đổi <SwitchPreference trong code thành <android.support.v7.preference.SwitchPreferenceCompat:

```
<android.support.v7.preference.SwitchPreferenceCompat
    android.defaultValue="true"
    android:key="example_switch"
    android:summary="@string/switch_summary"
    android:title="@string/switch_title" />
```

Dòng SwitchPreferenceCompat ở trên có thể hiện thị biểu tượng bóng đèn cảnh báo màu vàng nhưng bạn có thể bỏ qua nó.

3. Mở tệp style.xml trong tệp values và thêm mục preferenceTheme vào trong khai báo của Apptheme:

```
<item name="preferenceTheme">@style/PreferenceThemeOverlay</item>
```

Để sử dụng PreferenceFragmentCompat, bạn bắt buộc phải khai báo preferenceTheme với kiểu PreferenceThemeOverlay trong app theme.

1.4 Thêm Activity cho settings

Để tạo settings Activity(thứ cung cấp cài đặt UI) thêm Empty Activity vào ứng dụng, làm theo các bước sau:

1. Chọn app ở trên cùng của bảng Project>Android và chọn New>Activity>Empty Activity
2. Đặt tên Activity là SettingsActivity. Đặt tùy chọn Generate Layout File là unchecked (Bạn sẽ không cần đến nó) và để tùy chọn Launcher Activity là unchecked.
3. Đặt lựa chọn Backwards Compatibility (AppCompat) là checked. Tên của Package nên đặt là com.example.android.projectname.
4. Bấm Finish

1.5 Thêm Fragment cho cài đặt cụ thể

Fragment giống như 1 phần mô-đun của Activity, nó có vòng đời riêng và nhận các sự kiện đầu vào riêng. Bạn có thể thêm hoặc xóa 1 Fragment trong khi Activity đang chạy. Bạn sẽ sử dụng 1 lớp con chuyên biệt của Fragment để hiển thị danh sách cài đặt. Chỗ hay nhất của phần thực hành này là sử dụng Activity thường xuyên để lưu trữ và chạy (host) PreferenceFragment để hiển thị cài đặt ứng dụng. PreferenceFragment cung cấp rất nhiều kiến trúc linh hoạt hơn là sử dụng Activity cho preferences.

Bạn sẽ sử dụng PreferenceFragmentCompat nhiều hơn là PreferenceFragment để duy trì tính tương thích với AppCompatActivity.

Trong bước này bạn sẽ thêm 1 Fragment blank cho 1 nhóm các cài đặt tương tự(không có layout, phương thức factory hay interface callbacks) cho ứng dụng và mở rộng PreferenceFragmentCompat

Làm theo các bước sau:

1. Chọn app 1 lần nữa, và chọn New > Fragment > Fragment (Blank)
2. Đặt tên Settings của fragment. Bỏ chọn lựa chọn Create layout XML (vì bạn k cần đến nó)
3. Bỏ chọn lựa chọn bao gồm phương thức fragment factory và interface callbacks
4. Target Source Set nên đặt là main

5. Bấm Finish. Kết quả của lớp định nghĩa trong SettingsFragment như sau:

```
public class SettingsFragment extends Fragment {  
  
    public SettingsFragment() {  
        // Required empty public constructor  
    }  
  
    @Override  
    public View onCreateView(LayoutInflater inflater,  
        ViewGroup container, Bundle savedInstanceState) {  
        TextView textView = new TextView(getActivity());  
        textView.setText(R.string.hello_blank_fragment);  
        return textView;  
  
    }  
}
```

6. Chính sửa lớp định nghĩa của SettingsFragment để mở rộng PreferenceFragmentCompat:

```
public class SettingsFragment extends PreferenceFragmentCompat {
```

Khi bạn đã thay đổi lớp định nghĩa nên nó sẽ nối với định nghĩa được trình bày phía trên, biểu tượng bóng đèn đỏ xuất hiện bên lề trái. Nhấp vào bóng đèn đỏ và chọn **Implement methods** và sau đó chọn onCreatePreferences. Android Studio sẽ tạo ra 1 mẫu phương thức (stub) cho phương thức onCreatePreferences() như sau:

```
@Override  
public void onCreatePreferences(Bundle  
    savedInstanceState, String rootKey) {  
}
```

Để mở rộng Fragment, Android Studio thêm các câu lệnh import sau:

```
import android.support.v7.preference.PreferenceFragmentCompat;
```

7. Xóa tất cả phương thức onCreateView() trong fragment.

Lý do mà bạn quan tâm đến thay thế onCreateView() với onCreatePreferences() bởi vì bạn sẽ thêm SettingsFragment vào SettingsActivity hiện có thể hiển thị các tùy chọn(preferences), nhiều hơn là hiện thị màn hình Fragment riêng biệt. Thêm nó vào

Activity đã có sẵn và lmaf nó dễ dàng để thêm hoặc xóa Fragment trong khi Activity đang chạy. Fragment hiện thị các tùy chọn sẽ được gắn vào PreferenceScreen bằng rootkey.

Bạn có thể xóa 1 cách an toàn constructor rỗng khỏi SettingsFragment rất an toàn bởi vì Fragment không hiển thị bản thân nó.

```
public SettingsFragment() {  
    // Required empty public constructor  
}
```

8. Bạn cần liên kết Fragment này với tệp tài nguyên preferences.xml mà bạn đã tạo ở bước trước. Thêm vào phương thức mẫu vừa tạo(phương thức onCreatePreference()) 1 lệnh gọi setPreferencesFromResource(), truyền vào ID của tệp XML (R.xml.preference) và rootKey để xác định gốc của preference trong PreferenceScreen:

```
setPreferencesFromResource(R.xml.preferences, rootKey);
```

Phương thức onCreatePreference() nên giống như thế này:

```
@Override  
public void onCreatePreferences(Bundle  
                               savedInstanceState, String rootKey) {  
    setPreferencesFromResource(R.xml.preferences, rootKey);  
}
```

1.6 Hiển thị Fragment trong SettingsActivity

Để hiển thị Fragment trong SettingsActivity, làm theo những bước sau:

1. Mở SettingsActivity

2. Thêm những code sau vào cuối phương thức onCreate() để Fragment được hiển thị như là nội dung chính.

```
getSupportFragmentManager().beginTransaction()  
    .replace(android.R.id.content, new SettingsFragment())  
    .commit();
```

Đoạn mã dưới đây sử dụng mô hình phổ biến để thêm một Fragment vào một Activity, sao cho Fragment xuất hiện như nội dung chính của Activity:

- Sử dụng **getFragmentManager()** nếu lớp mở rộng từ **Activity** và **Fragment** mở rộng từ **PreferenceFragment**.

- Sử dụng **getSupportFragmentManager()** nếu lớp kế thừa từ **AppCompatActivity** và **Fragment** kế thừa từ **PreferenceFragmentCompat**.

Phương thức **onCreate()** trong **SettingsActivity** sẽ có dạng như sau:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getSupportFragmentManager().beginTransaction()
        .replace(android.R.id.content, new SettingsFragment())
        .commit();
}
```

1.7 Kết nối mục Settings trong menu với SettingsActivity

Sử dụng **Intent** để khởi chạy **SettingsActivity** từ **MainActivity** khi người dùng chọn **Settings** từ menu tùy chọn.

- Mở **MainActivity** và tìm khói **if** trong phương thức **onOptionsItemSelected()**, nơi xử lý sự kiện khi người dùng nhấn vào **Settings** trong menu tùy chọn:

```
if (id == R.id.action_settings) {
    return true;
}
```

- Thêm một **Intent** vào khói **if** để khởi chạy **SettingsActivity**.

```
if (id == R.id.action_settings) {
    Intent intent = new Intent(this, SettingsActivity.class);
    startActivity(intent);
    return true;
}
```

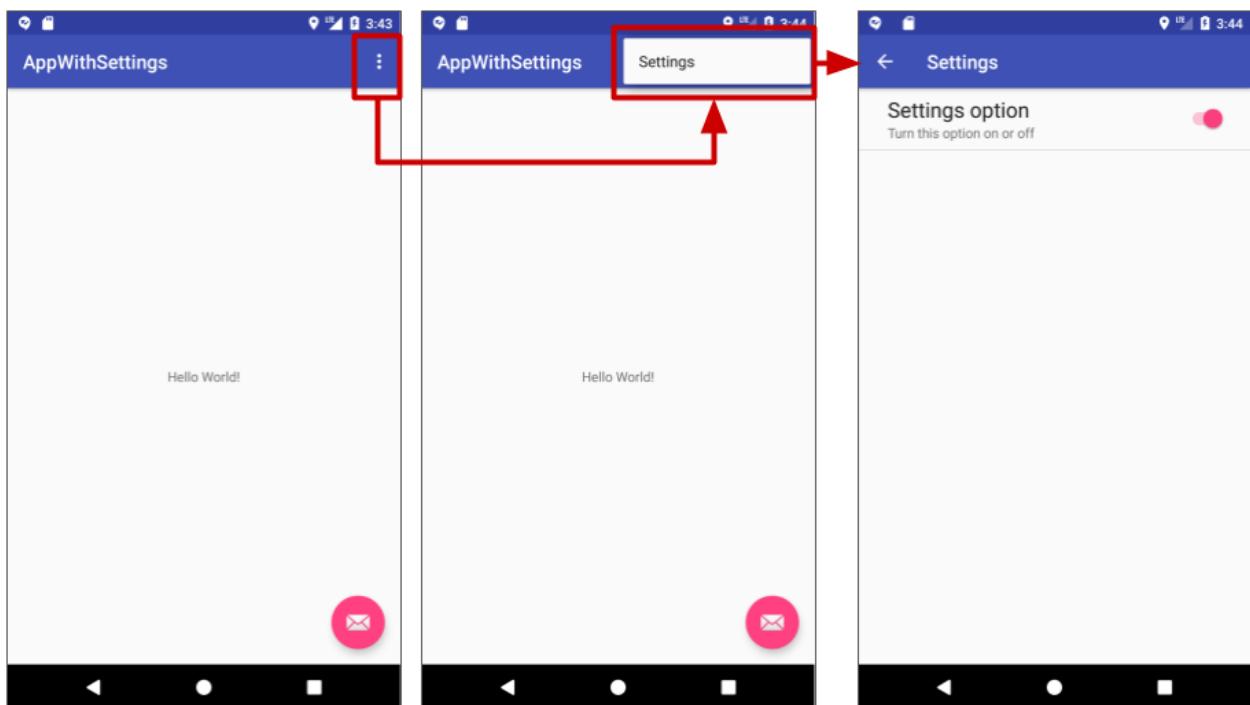
- Để thêm nút điều hướng **Up** trên thanh **app bar** cho **SettingsActivity**, bạn cần chỉnh sửa khai báo của nó trong tệp **AndroidManifest.xml** để xác định **SettingsActivity** có **MainActivity** làm activity cha. Mở **AndroidManifest.xml** và tìm khai báo **SettingsActivity**:

```
<activity android:name=".SettingsActivity"></activity>
```

Thay đổi khai báo thành:

```
<activity android:name=".SettingsActivity"
    android:label="Settings"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity"/>
</activity>
```

- Chạy ứng dụng. Nhấn vào biểu tượng menu (overflow icon) để mở menu tùy chọn, như trong hình bên trái. Chọn **Settings** để mở **SettingsActivity**, như trong hình trung tâm. Nhấn nút **Up** trên thanh **app bar** của **Settings Activity**, như trong hình bên phải, để quay lại **Main Activity**



1.8 Lưu giá trị mặc định trong SharedPreferences

Mặc dù giá trị mặc định cho cài đặt **toggle switch** đã được thiết lập trong thuộc tính **android:defaultValue** (ở [Bước 1.2 của bài này](#)), ứng dụng vẫn cần lưu giá trị mặc định trong tệp **SharedPreferences** cho mỗi cài đặt khi người dùng mở ứng dụng lần đầu tiên. Hãy làm theo các bước sau để thiết lập giá trị mặc định cho **toggle switch**:

- Mở **MainActivity**.
- Thêm đoạn mã sau vào cuối phương thức **onCreate()**, ngay sau đoạn mã của **FloatingActionButton**:

```
        android.support.v7.preference.PreferenceManager  
            .setDefaultValues(this, R.xml.preferences, false);
```

Mã trên đảm bảo rằng các cài đặt được khởi tạo đúng với các giá trị mặc định.

Phương thức PreferenceManager.setDefaultValues() nhận ba tham số:

- **Ngữ cảnh của ứng dụng (app context)**, ví dụ như this.
- **ID tài nguyên (preferences)** của tệp XML chứa một hoặc nhiều cài đặt.
- **Một giá trị boolean** xác định xem có nên đặt lại các giá trị mặc định nhiều lần hay không.
 - Khi **false**, hệ thống chỉ đặt giá trị mặc định nếu phương thức này chưa từng được gọi trước đó. Nếu bạn luôn đặt tham số thứ ba là **false**, bạn có thể gọi phương thức này mỗi khi **MainActivity** khởi động mà không làm thay đổi các giá trị cài đặt mà người dùng đã lưu. Tuy nhiên, nếu bạn đặt nó là **true**, phương thức này sẽ ghi đè bất kỳ giá trị nào trước đó bằng giá trị mặc định.

1.9 Đọc giá trị cài đặt đã thay đổi từ SharedPreferences

Khi ứng dụng khởi động, phương thức **onCreate()** trong **MainActivity** có thể đọc các giá trị cài đặt đã được thay đổi và sử dụng chúng thay vì các giá trị mặc định.

Mỗi cài đặt được xác định bằng một cặp **key-value**. Hệ thống Android sử dụng cặp này để lưu hoặc truy xuất các cài đặt từ tệp **SharedPreferences** của ứng dụng. Khi người dùng thay đổi một cài đặt, hệ thống sẽ cập nhật giá trị tương ứng trong **SharedPreferences**. Để sử dụng giá trị cài đặt, ứng dụng có thể dùng **key** để truy xuất giá trị từ **SharedPreferences** bằng cách sử dụng phương thức **PreferenceManager.getDefaultSharedPreferences()**.

Hãy làm theo các bước sau để thêm đoạn mã này:

1. Mở **SettingsActivity** và tạo một biến **static String** để lưu khóa cho giá trị:

```
public static final String  
    KEY_PREF_EXAMPLE_SWITCH = "example_switch";
```

2. Mở **MainActivity** và thêm đoạn mã sau vào cuối phương thức **onCreate()**

```
SharedPreferences sharedPref =  
    android.support.v7.preference.PreferenceManager  
        .getDefaultSharedPreferences(this);  
Boolean switchPref = sharedPref.getBoolean  
    (SettingsActivity.KEY_PREF_EXAMPLE_SWITCH, false);
```

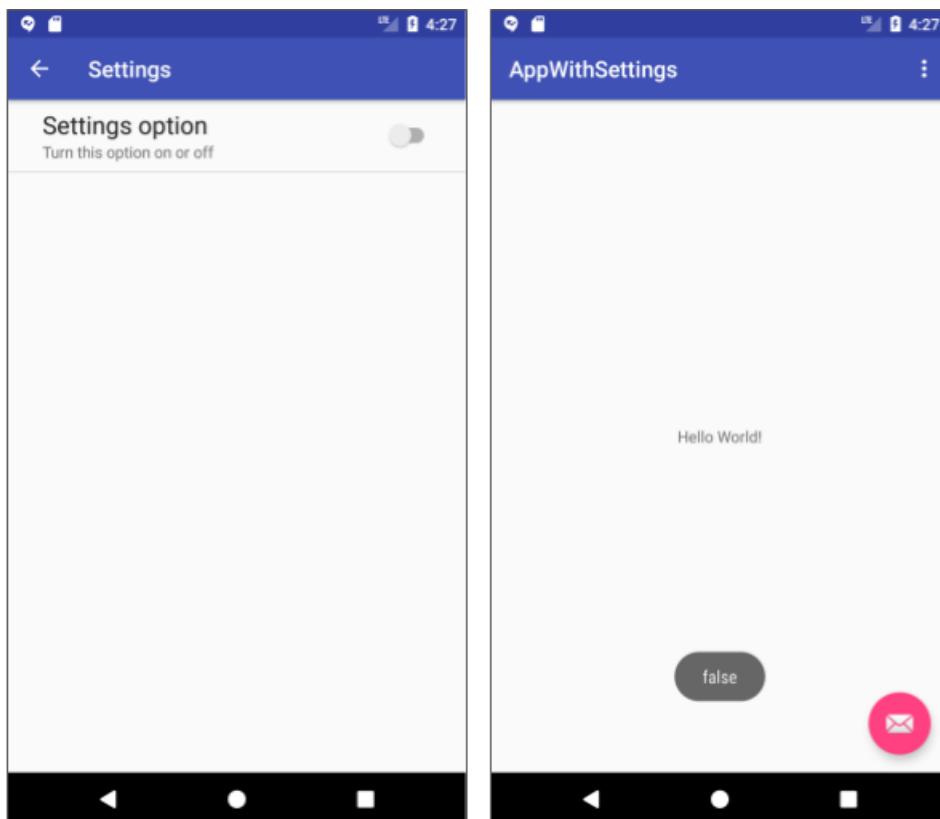
```
Toast.makeText(this, switchPref.toString(),  
    Toast.LENGTH_SHORT).show();
```

3. Chạy ứng dụng và nhấn **Settings** để mở **SettingsActivity**.

4. Nhấn vào tùy chọn cài đặt để thay đổi trạng thái **toggle** từ **on** sang **off**, như minh họa ở phía bên trái của hình dưới đây.

5. Nhấn nút **Up** trong **SettingsActivity** để quay lại **MainActivity**. Một thông báo **Toast** sẽ xuất hiện trong **MainActivity** với giá trị của cài đặt, như minh họa ở phía bên phải của hình dưới đây.

6. Lặp lại các bước trên để kiểm tra xem thông báo **Toast** có thay đổi khi bạn thay đổi cài đặt hay không.



Đoạn mã trên sử dụng các phương thức sau:

- android.support.v7.preference.PreferenceManager.getDefaultSharedPreferences(this) để lấy cài đặt dưới dạng đối tượng **SharedPreferences** (được lưu trong biến sharedPref).
- getBoolean() để lấy giá trị **Boolean** của cài đặt, sử dụng khóa (**key**) được định nghĩa là KEY_PREF_EXAMPLE_SWITCH trong **SettingsActivity** và gán giá trị này cho biến

switchPref. Nếu không có giá trị nào cho khóa này, phương thức getBoolean() sẽ gán giá trị mặc định **false** cho switchPref. Đối với các kiểu dữ liệu khác như chuỗi, số nguyên hoặc số thực dấu chấm động, bạn có thể sử dụng lần lượt các phương thức getString(), getInt(), hoặc getFloat().

- Toast.makeText() và show() được sử dụng để hiển thị giá trị của cài đặt switchPref.

Mỗi khi **MainActivity** khởi động hoặc khởi động lại, phương thức **onCreate()** cần đọc các giá trị cài đặt để sử dụng trong ứng dụng. Trong phiên bản thực tế, phương thức Toast.makeText() sẽ được thay thế bằng một phương thức khởi tạo cài đặt.

Bạn đã có một **Settings Activity** hoạt động trong ứng dụng của mình.

Task 1: Mã giải pháp

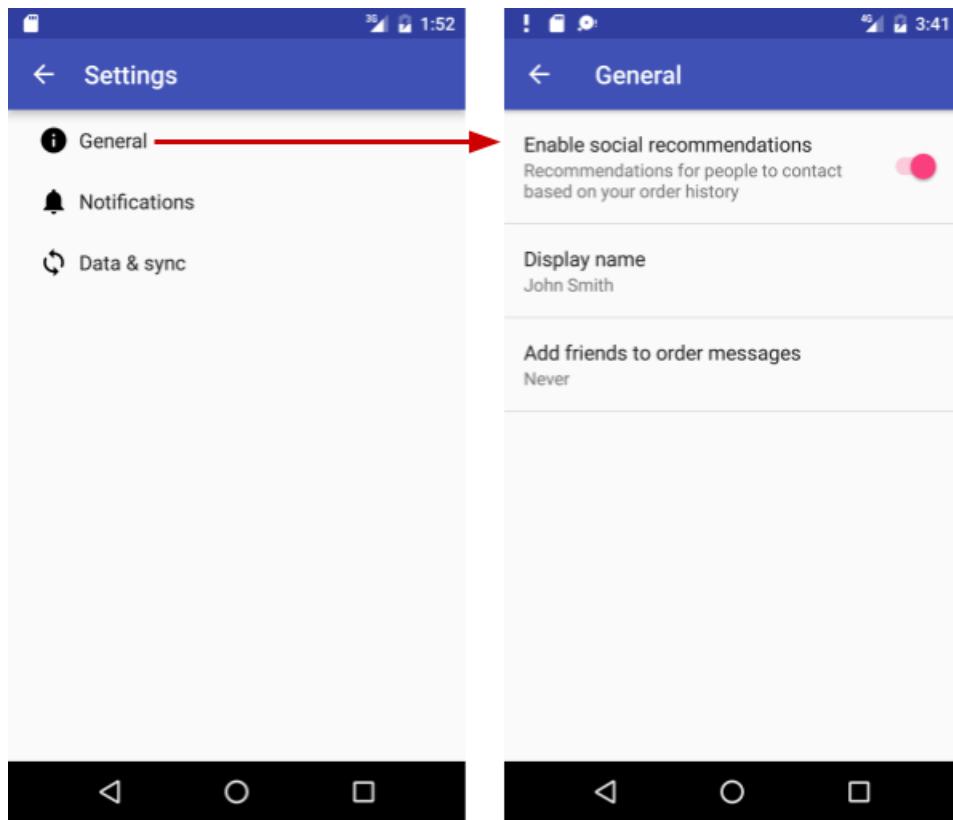
Dự án Android Studio: **AppWithSettings**

Task 2: Sử dụng mẫu (template) Settings Activity

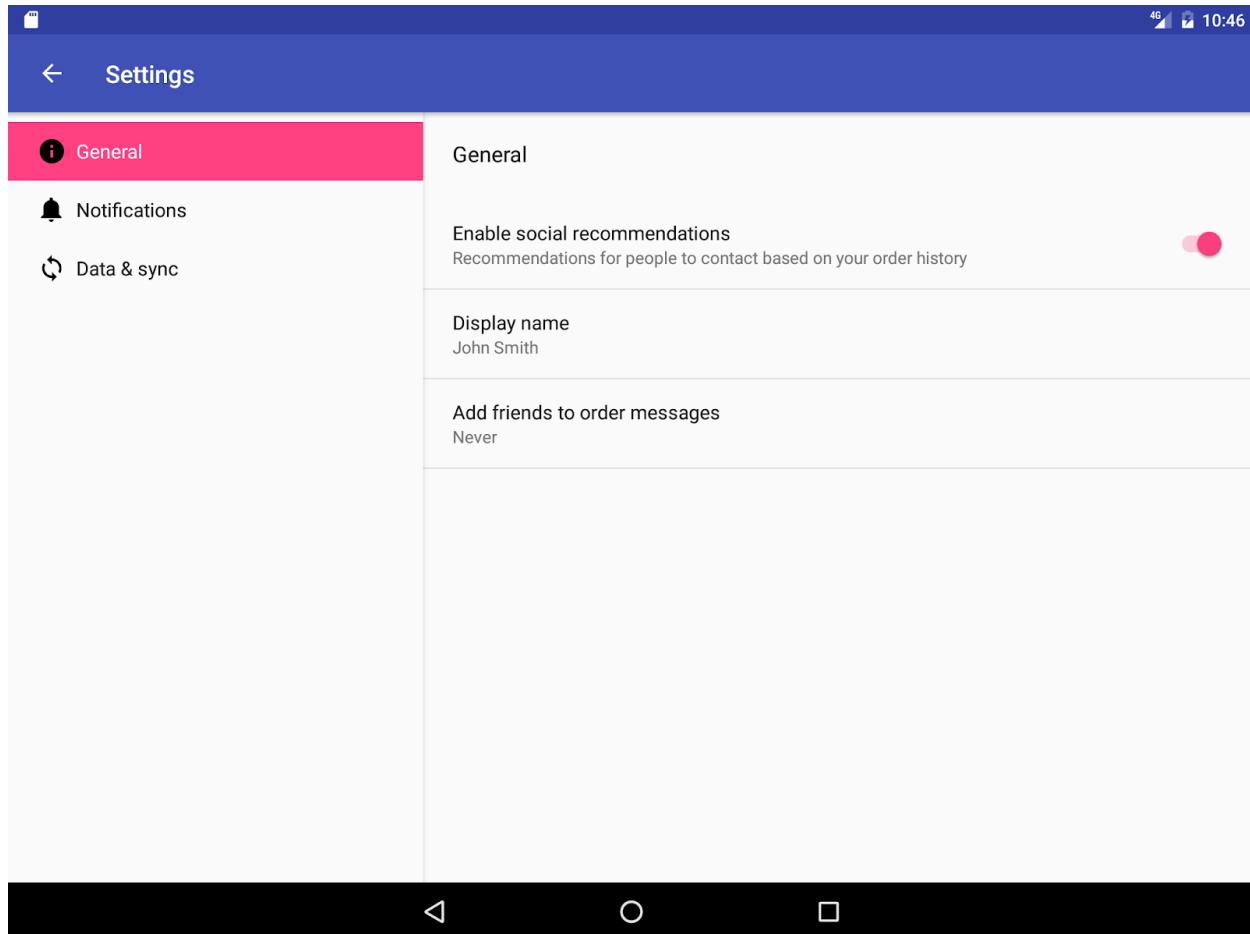
Nếu bạn cần tạo nhiều màn hình con cho cài đặt và muốn tận dụng không gian màn hình của các thiết bị tablet cũng như đảm bảo tính tương thích với các phiên bản Android cũ hơn dành cho tablet, Android Studio cung cấp một giải pháp rút gọn: mẫu Settings Activity.

Trong bài trước, bạn đã học cách sử dụng một Settings Activity trống và một Fragment trống để thêm cài đặt vào ứng dụng. Task 2 sẽ hướng dẫn bạn cách sử dụng mẫu Settings Activity có sẵn trong Android Studio để:

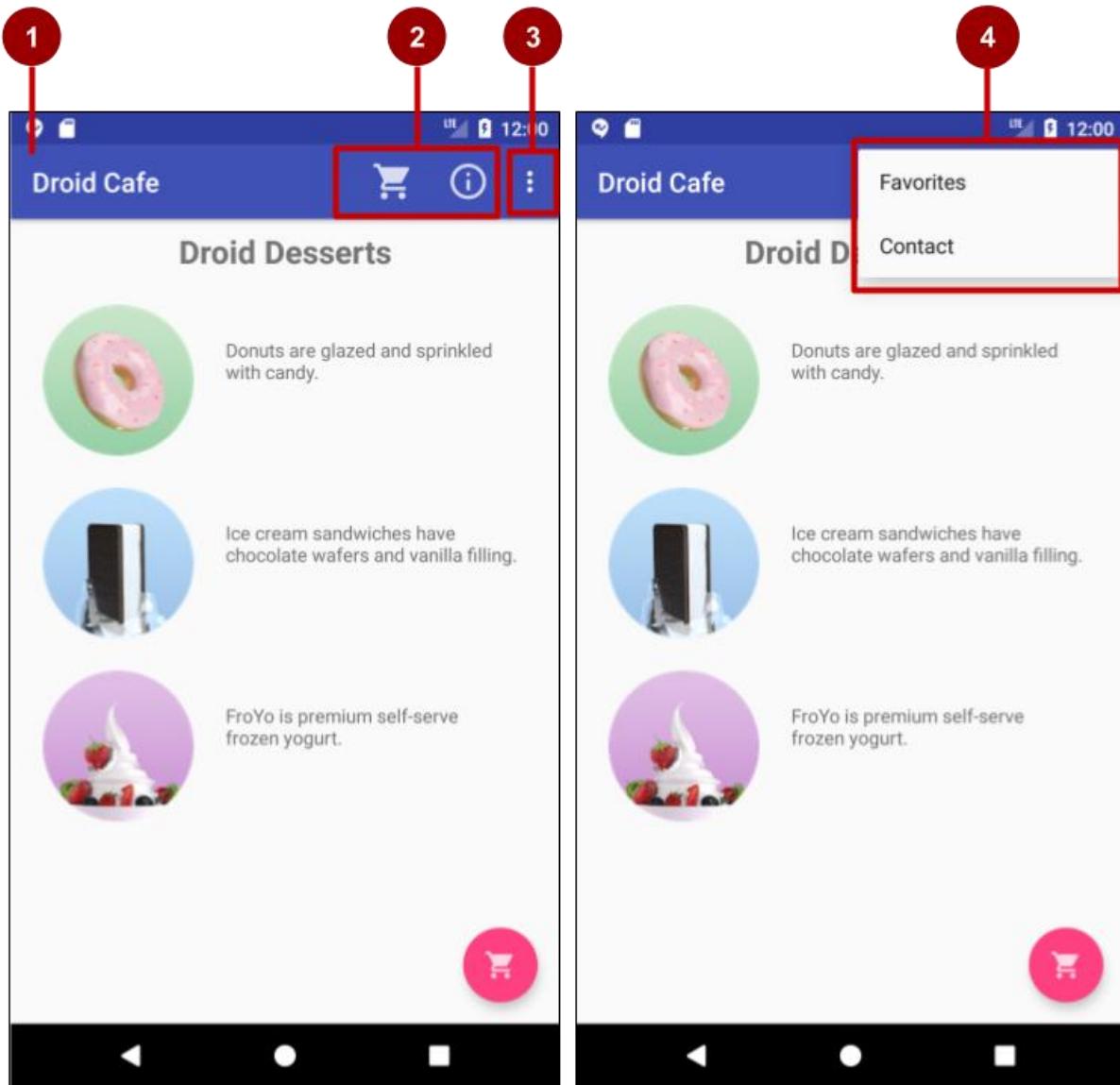
- **Phân chia** các cài đặt thành nhiều nhóm.
- **Tùy chỉnh** các cài đặt và giá trị của chúng.
- **Hiển thị** màn hình cài đặt chính với một tiêu đề liên kết cho mỗi nhóm cài đặt, chẳng hạn như nhóm cài đặt chung (General settings), như minh họa trong hình bên dưới.



Hiển thị giao diện màn hình master/detail với một liên kết tiêu đề cho mỗi nhóm ở phía bên trái (master) và nhóm cài đặt ở phía bên phải (detail), như minh họa trong hình bên dưới.



Trong bài thực hành trước, bạn đã tạo một ứng dụng có tên **DroidCafeOptionsUp** bằng cách sử dụng mẫu **Basic Activity**, mẫu này cung cấp một menu tùy chọn trên thanh ứng dụng như hình bên dưới.



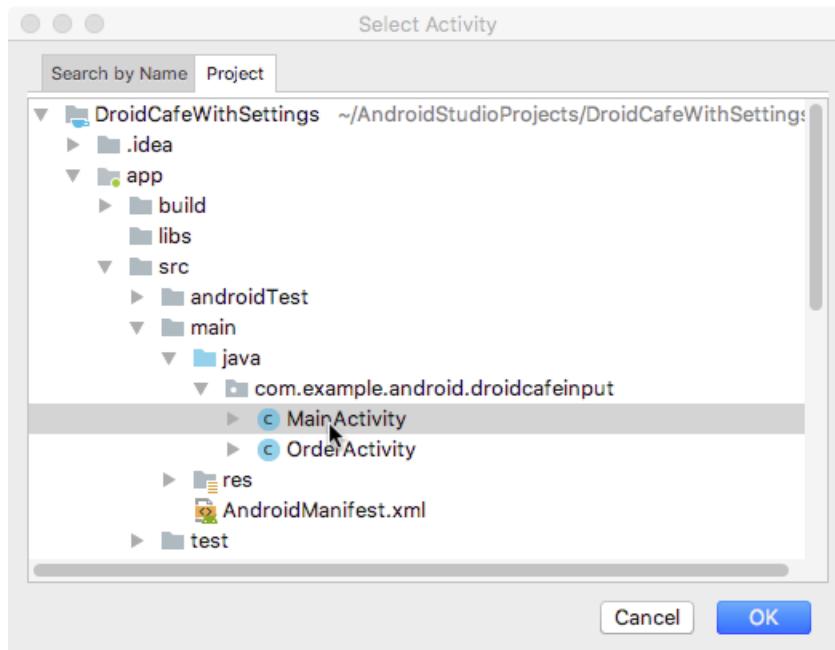
Chú giải cho hình trên:

1. Thanh ứng dụng
2. Biểu tượng hành động trong menu tùy chọn
3. Nút Overflow
4. Menu tùy chọn overflow

2.1 Khám phá mẫu Settings Activity

Để thêm mẫu Settings Activity vào dự án ứng dụng trong Android Studio, hãy làm theo các bước sau:

1. Sao chép thư mục dự án **DroidCafeOptionsUp** và đổi tên thành **DroidCafeWithSettings**. Chạy ứng dụng để đảm bảo rằng nó hoạt động bình thường.
2. Trong **Project > Android** (ở bảng điều hướng bên trái), chọn **app**, sau đó chọn **New > Activity > Settings Activity**.
3. Trong hộp thoại xuất hiện, chấp nhận tên Activity mặc định (**SettingsActivity** là tên được đề xuất) và tiêu đề (**Settings**).
4. Nhập vào **ba dấu chấm** ở cuối menu **Hierarchical Parent**, sau đó trong hộp thoại **Select Activity**, chọn tab **Project** (xem hình bên dưới).
5. Mở rộng đường dẫn **DroidCafeWithSettings > app > src > main > java > com.example.android.droidcafeinput**, sau đó chọn **MainActivity** làm activity cha, như minh họa trong hình bên dưới. Nhấn **OK**.



Bạn chọn **MainActivity** làm activity cha để nút **Up** trên thanh ứng dụng trong **SettingsActivity** đưa người dùng quay lại **MainActivity**. Việc chọn activity cha sẽ tự động cập nhật tệp **AndroidManifest.xml** để hỗ trợ điều hướng bằng nút **Up**.

6. Nhấn **Finish**.
7. Trong bảng điều hướng **Project > Android**, mở rộng đường dẫn **app > res > xml** để xem các tệp XML được tạo bởi mẫu **Settings Activity**.

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