

**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.....	19
1.3) Trình chỉnh sửa bố cục .....	30
1.4) Văn bản và các chế độ cuộn .....	58
1.5) Tài nguyên có sẵn .....	71
Bài 2) Activities .....	80
2.1) Activity và Intent .....	80
2.2) Vòng đời của Activity và trạng thái.....	100
2.3) Intent ngầm định .....	107
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ .....	127
3.1) Trình gỡ lỗi .....	127
3.2) Kiểm thử đơn vị .....	135
3.3) Thư viện hỗ trợ.....	142
CHƯƠNG 2. Trải nghiệm người dùng .....	149
Bài 1) Tương tác người dùng .....	149
1.1) Hình ảnh có thể chọn.....	149
1.2) Các điều khiển nhập liệu .....	158
1.3) Menu và bộ chọn.....	158
1.4) Điều hướng người dùng .....	158
1.5) RecyclerView .....	158
Bài 2) Trải nghiệm người dùng thú vị .....	158
2.1) Hình vẽ, định kiểu và chủ đề .....	158
2.2) Thẻ và màu sắc.....	158

2.3)	Bố cục thích ứng .....	158
Bài 3)	Kiểm thử giao diện người dùng.....	158
3.1)	Espresso cho việc kiểm tra UI .....	158
CHƯƠNG 3. Làm việc trong nền.....		158
Bài 1)	Các tác vụ nền.....	158
1.1)	AsyncTask.....	158
1.2)	AsyncTask và AsyncTaskLoader .....	158
1.3)	Broadcast receivers .....	158
Bài 2)	Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	158
2.1)	Thông báo .....	158
2.2)	Trình quản lý cảnh báo .....	158
2.3)	JobScheduler .....	158
CHƯƠNG 4. Lưu dữ liệu người dùng .....		158
Bài 1)	Tùy chọn và cài đặt .....	158
1.1)	Shared preferences .....	158
1.2)	Cài đặt ứng dụng .....	159
Bài 2)	Lưu trữ dữ liệu với Room.....	159
2.1)	Room, LiveData và ViewModel .....	159
2.2)	Room, LiveData và ViewModel .....	159

## 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**

## Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp (IDE) hoàn chỉnh, bao gồm một trình chỉnh sửa mã nâng cao và một bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ để phát triển, gỡ lỗi, kiểm thử và tối ưu hiệu suất, giúp việc phát triển ứng dụng nhanh hơn và dễ dàng hơn. Bạn có thể kiểm thử ứng dụng trên nhiều trình giả lập được

cấu hình sẵn hoặc trên thiết bị di động của chính mình, xây dựng ứng dụng hoàn chỉnh và xuất bản trên cửa hàng Google Play.

Android Studio có sẵn cho máy tính chạy Windows hoặc Linux, và cho Mac chạy macOS. Phiên bản mới nhất của OpenJDK (Java Development Kit) được tích hợp sẵn trong Android Studio.

Để bắt đầu với Android Studio, trước tiên hãy kiểm tra yêu cầu hệ thống để đảm bảo rằng hệ thống của bạn đáp ứng đủ điều kiện. Quá trình cài đặt tương tự trên tất cả các nền tảng, mọi điểm khác biệt sẽ được ghi chú bên dưới.

Các bước cài đặt:

1. Truy cập trang web dành cho nhà phát triển Android và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
2. Chấp nhận các cấu hình mặc định cho tất cả các bước và đảm bảo rằng tất cả các thành phần đều được chọn để cài đặt.
3. Sau khi hoàn tất cài đặt, Trình hướng dẫn thiết lập sẽ tải xuống và cài đặt một số thành phần bổ sung, bao gồm cả Android SDK. Quá trình này có thể mất một chút thời gian tùy vào tốc độ Internet của bạn và có thể có một số bước trông có vẻ 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.

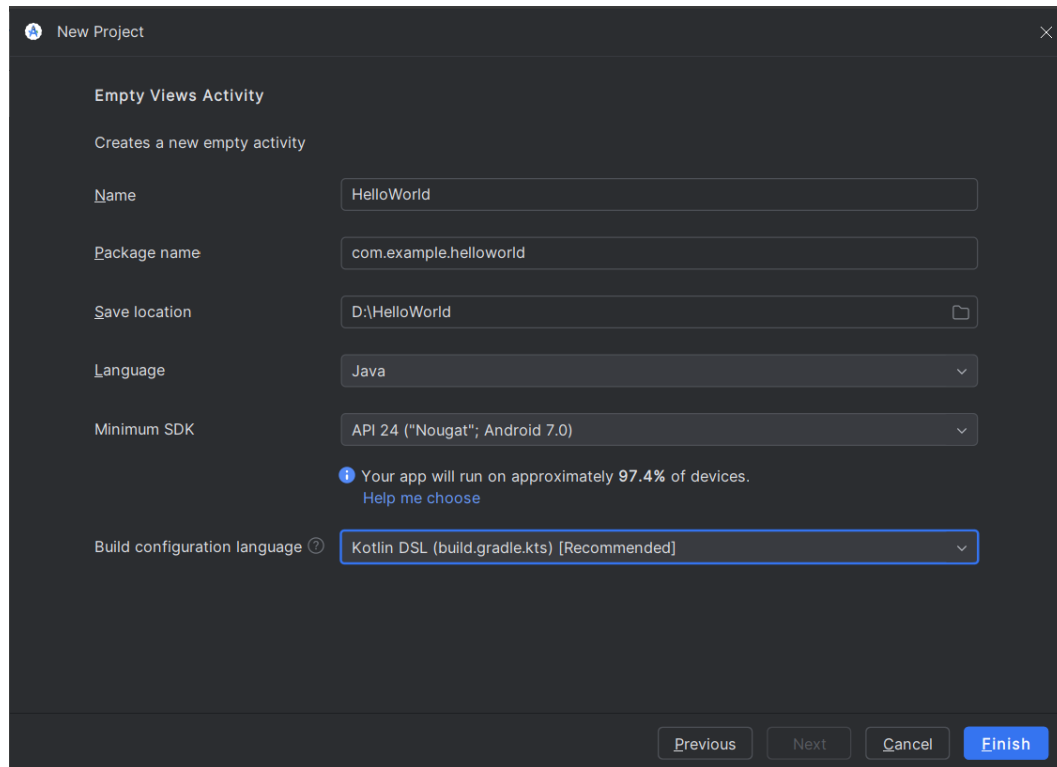
## **Nhiệm vụ 2: Tạo ứng dụng Hello World**

Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị "Hello World" để kiểm tra xem Android Studio đã được cài đặt đúng cách hay chưa, đồng thời làm quen với những kiến thức cơ bản khi phát triển ứng dụng bằng Android Studio.

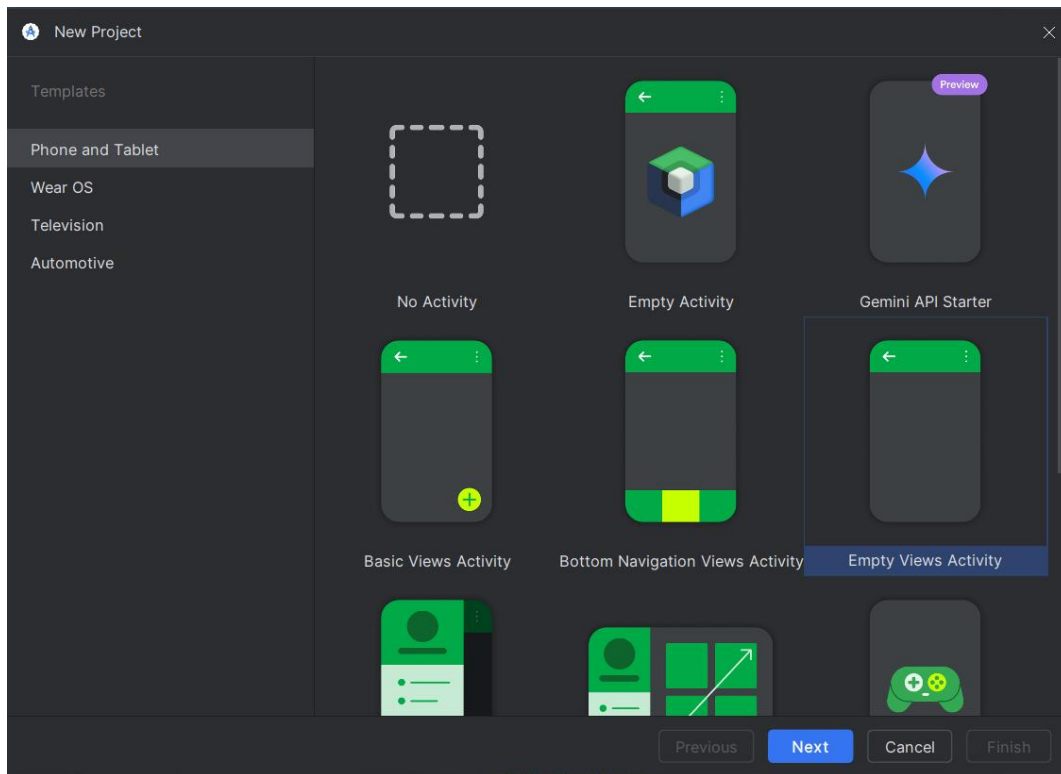
1. Tạo dự án ứng dụng
  - Mở Android Studio nếu nó chưa được mở.
  - Trong cửa sổ chính Welcome to Android Studio, nhấp vào Start a new Android Studio project.
  - Trong cửa sổ New Project, nhập Hello World vào ô Application name.
  - Kiểm tra xem vị trí lưu trữ mặc định của dự án có đúng với thư mục bạn muốn lưu ứng dụng Hello World và các dự án Android Studio khác không. Nếu không, hãy thay đổi thành thư mục bạn mong muốn.
  - Chấp nhận giá trị mặc định com.example.helloworld cho Package name, hoặc tạo một tên miền riêng cho công ty bạn. Nếu bạn không có kế hoạch phát hành

ứng dụng, bạn có thể giữ nguyên giá trị mặc định. Lưu ý rằng việc thay đổi tên gói (package name) sau này sẽ tốn thêm công sức.

- Minimum SDK defaults to API 24 ("Nougat"; Android 7.0).
- Nếu dự án của bạn yêu cầu các thành phần bổ sung cho SDK đã chọn, Android Studio sẽ tự động cài đặt chúng.



- Activity là một thành phần quan trọng của ứng dụng Android, đại diện cho một tác vụ cụ thể mà người dùng có thể thực hiện. Mỗi Activity thường có một layout đi kèm để xác định cách hiển thị các phần tử giao diện người dùng trên màn hình. Android Studio cung cấp các mẫu Activity để giúp bạn bắt đầu nhanh chóng. Đối với dự án Hello World, hãy chọn Empty View Activity, sau đó nhấn Next.
- Tên layout mặc định là activity\_main.

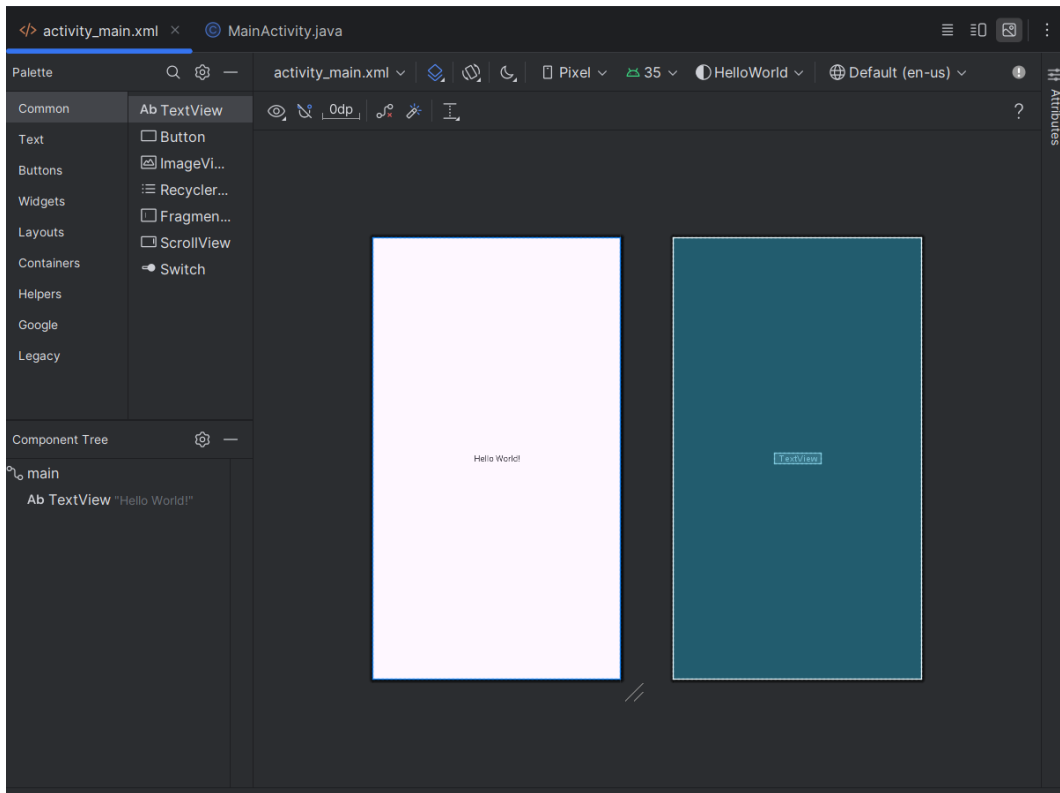


- Nhấn Finish.

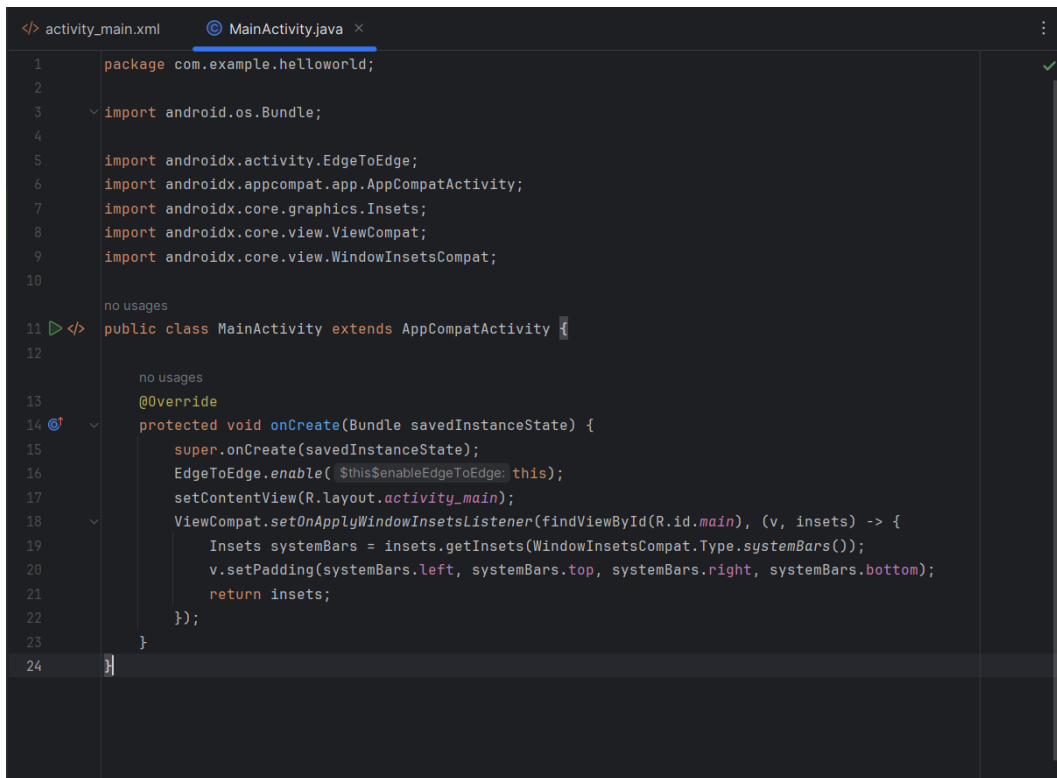
Android Studio sẽ tạo một thư mục chứa dự án của bạn và tiến hành build dự án bằng Gradle (quá trình này có thể mất một vài phút).

Sau khi quá trình tạo dự án hoàn tất, trình chỉnh sửa Android Studio sẽ xuất hiện. Hãy làm theo các bước sau:

- Nhấp vào tab `activity_main.xml` để mở trình chỉnh sửa layout.
- Nhấp vào tab Design (nếu chưa được chọn) để xem bố cục giao diện đồ họa.



- Nhấp vào tab MainActivity.java để xem trình chỉnh sửa mã nguồn.

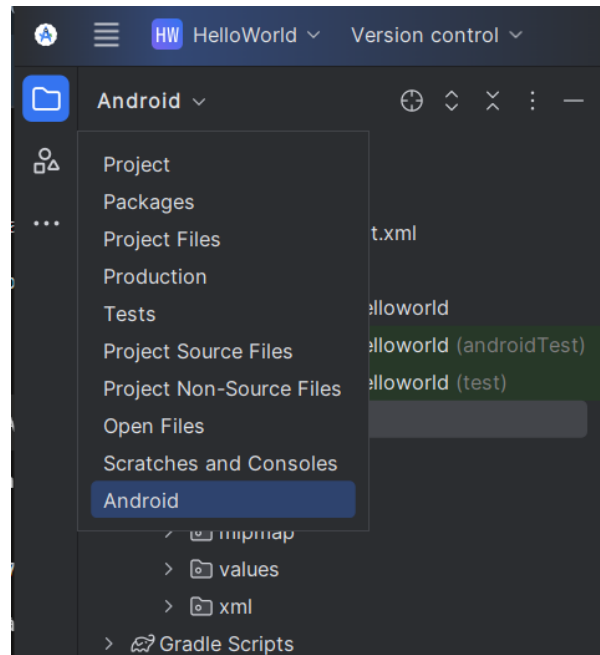




## 2. Khám phá bảng điều hướng Project > Android

Trong phần này, bạn sẽ tìm hiểu cách tổ chức dự án trong Android Studio.

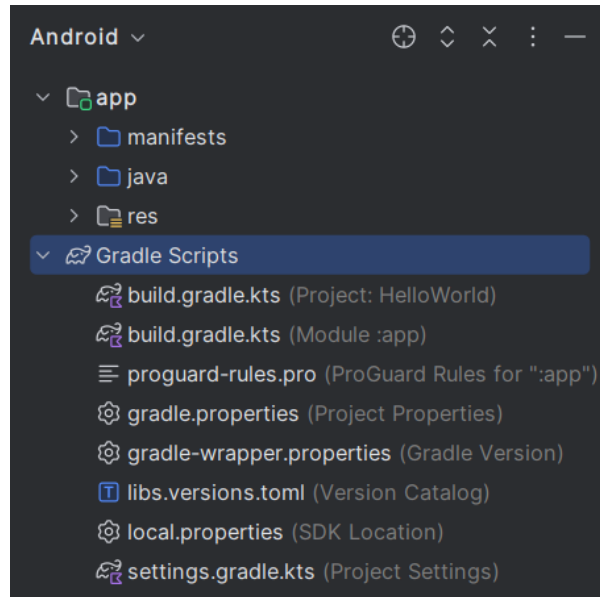
- Nếu tab Project chưa được chọn, hãy nhấp vào tab Project trong cột tab dọc ở bên trái cửa sổ Android Studio. Khi đó, bảng Project sẽ xuất hiện.
- Để xem dự án theo cấu trúc thư mục chuẩn của Android, hãy chọn Android từ menu thả xuống ở đầu bảng Project, như hình minh họa bên dưới.



## 3. Khám phá thư mục Gradle Scripts

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

Khi bạn tạo ứng dụng dự án lần đầu tiên, bảng Project > Android sẽ hiển thị với thư mục Gradle Scripts được mở rộng như hình minh họa bên dưới.



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

- Nếu thư mục Gradle Scripts không được mở rộng, hãy nhấp vào hình tam giác để mở rộng thư mục. Thư mục này chứa tất cả các tệp cần thiết cho hệ thống xây dựng.
- Tìm tệp build.gradle(Project: HelloWorld)
  - Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô-đun tạo nên dự án của bạn. Mỗi dự án Android Studio đều chứa một tệp xây dựng Gradle cấp cao nhất. Hầu hết thời gian, bạn sẽ không cần thực hiện bất kỳ thay đổi nào đối với tệp này, nhưng vẫn hữu ích khi hiểu nội dung của tệp.

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

- Tìm tệp build.gradle(Module:app).
  - Ngoài tệp build.gradle cấp dự án, mỗi module đều có tệp build.gradle riêng, cho phép bạn định cấu hình cài đặt bản dựng cho từng module cụ thể (ứng dụng HelloWorld chỉ có một module). Định cấu hình các cài đặt bản dựng này cho phép bạn cung cấp các tùy chọn đóng gói tùy chỉnh, chẳng hạn như các loại bản dựng bổ sung và hương vị sản phẩm.
  - Tệp này thường là tệp cần chỉnh sửa khi thay đổi cấu hình cấp ứng dụng, chẳng hạn như khai báo các phụ thuộc trong phần phụ thuộc. Bạn có thể khai báo phụ thuộc thư viện bằng một trong một số cấu hình phụ thuộc

khác nhau. Mỗi cấu hình phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện

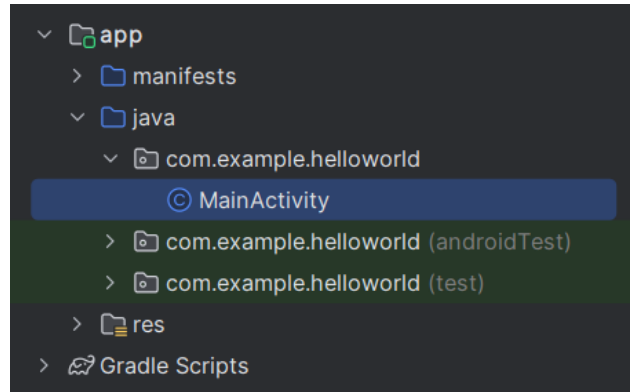
```
plugins {
    alias(libs.plugins.android.application)
}
android {
    namespace = "com.example.hellotoast"
    compileSdk = 35
    defaultConfig {
        applicationId = "com.example.hellotoast"
        minSdk = 24
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

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

- Đóng Gradle Scripts.
4. Khám phá các thư mục app và res

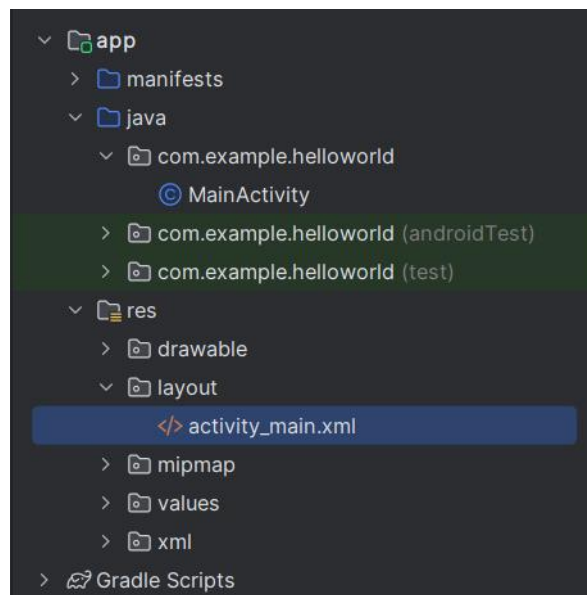
Tất cả mã và tài nguyên cho ứng dụng đều nằm trong các thư mục app và res.

- Mở rộng thư mục app, thư mục java và thư mục com.example.helloworld để xem tệp java MainActivity. Nhấp vào tệp để mở tệp trong trình soạn thảo mã.



Thư mục java bao gồm các tệp lớp Java trong ba thư mục con, như thể hiện trong hình trên. Thư mục com.example.helloworld (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục còn lại được sử dụng để thử nghiệm và được mô tả trong bài học khác. Đối với ứng dụng HelloWorld, chỉ có một gói và nó chứa MainActivity.java. Tên của Hoạt động đầu tiên mà người dùng nhìn thấy, cũng khởi tạo các tài nguyên trên toàn ứng dụng, thường được gọi là MainActivity.

- Mở thư mục res và thư mục layout, rồi nhấp đúp vào tệp activity\_main.xml để mở tệp đó trong trình soạn thảo layout



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

## 5. Khám phá thư mục manifests

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

- Mở thư mục manifests.
- Mở tệp AndroidManifest.xml.

Tệp AndroidManifest.xml mô tả tất cả các thành phần của ứng dụng Android của bạn. Tất cả các thành phần 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 truy cập tính năng.

## Nhiệm vụ 3: Sử dụng 1 thiết bị ảo (giả lập)

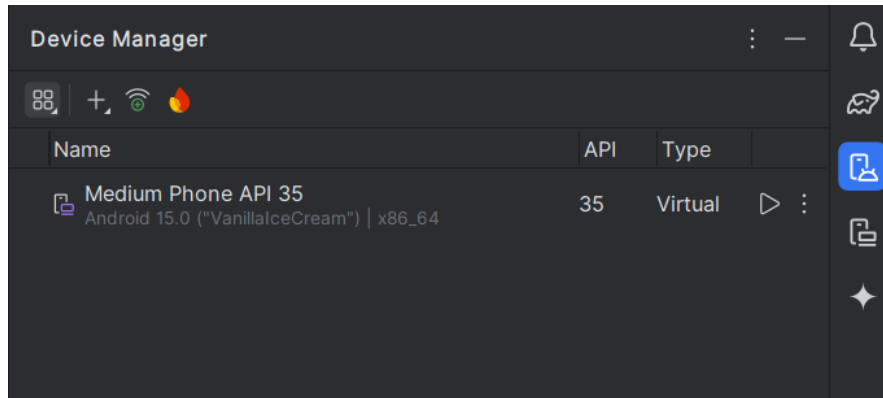
Trong nhiệm vụ này, bạn sẽ sử dụng trình quản lý Android Virtual Device(AVD) Manager để tạo một thiết bị ảo mô phỏng cấu hình cho một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Lưu ý rằng trình giả lập android có các yêu cầu bổ sung ngoài yêu cầu hệ thống cơ bản của Android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

Với AVD Manager, bạn có thể xác định các đặc điểm phần cứng của thiết bị, cấp API, dung lượng lưu trữ, giao diện và các thuộc tính khác rồi lưu lại dưới dạng một thiết bị ảo. Điều này giúp bạn kiểm tra ứng dụng trên các cấu hình thiết bị khác nhau (như điện thoại, máy tính bảng) với các phiên bản API khác nhau mà không cần sử dụng thiết bị vật lý.

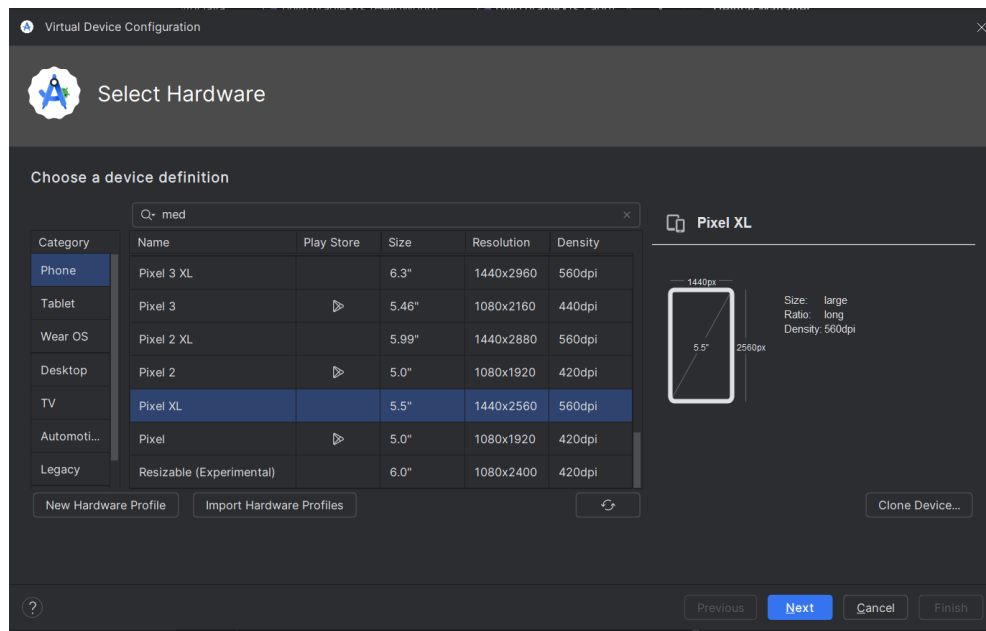
### 1. Các bước tạo thiết bị ảo Android (AVD)

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

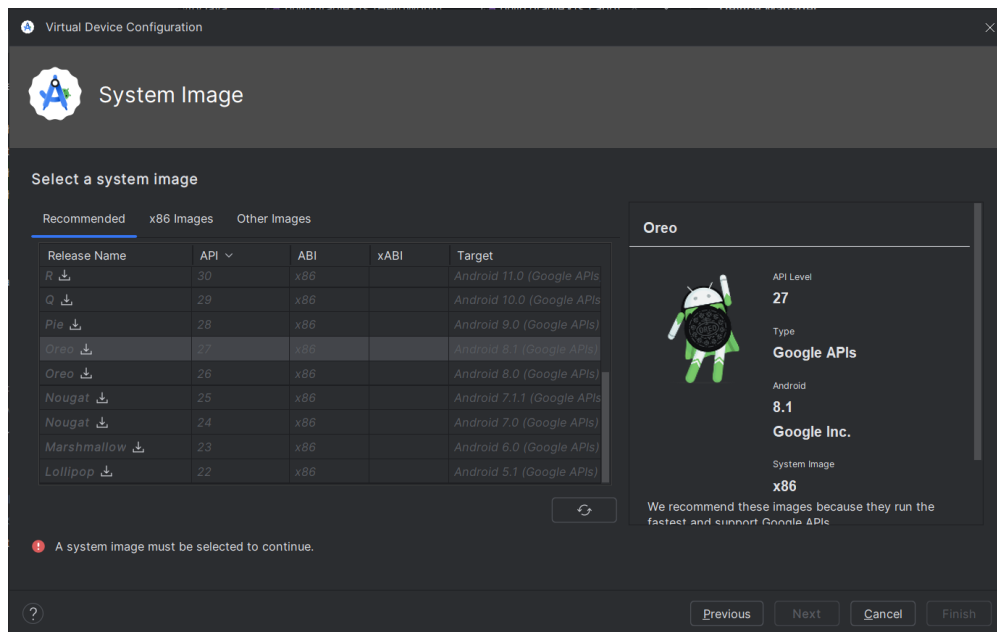
- Trong Android Studio, chọn Tools > Device Manager hoặc nhấp vào biểu tượng Device Manager ở thanh công cụ. Màn hình Thiết bị ảo của bạn sẽ xuất hiện. Nếu bạn đã tạo thiết bị ảo, màn hình sẽ hiển thị các thiết bị đó, nếu không bạn sẽ thấy danh sách trống.



- Nhấp vào “+” > Create Virtual Device. Cửa sổ Select Hardware xuất hiện hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng 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).



- Chọn một thiết bị như Nexus 5x hoặc Pixel XL, rồi nhấp vào Next. Màn hình Ảnh hệ thống sẽ xuất hiện.
- Nhấp vào tab Recommend nếu chưa chọn và chọn phiên bản hệ thống Android nào để chạy trên thiết bị ảo (như Oreo)



- Sau khi chọn ảnh hệ thống, hãy nhấp vào Next. Kiểm tra cấu hình của bạn và nhấp vào Finish.
- Chạy ứng dụng trên thiết bị ảo
    - Trong tác vụ này, cuối cùng bạn sẽ chạy ứng dụng Hello World của mình
    - Trong Android Studio, chọn Run > Run App hoặc nhấp vào biểu tượng Run trên thanh công cụ.
    - Trong cửa sổ Chọn mục tiêu triển khai, bên dưới Thiết bị ảo khả dụng, hãy chọn thiết bị ảo mà bạn vừa tạo và nhấp vào OK.

#### Nhiệm vụ 4: (Tùy chọn) Sử dụng thiết bị vật lý

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

Những gì bạn cần:

- Một thiết bị Android như điện thoại hoặc máy tính bảng.
- Cáp dữ liệu để kết nối thiết bị Android với máy tính qua cổng USB.
- Nếu bạn đang sử dụng hệ điều hành Linux hoặc Windows, bạn có thể cần thực hiện các bước bổ sung để chạy ứng dụng trên thiết bị vật lý. Xem tài liệu Sử dụng thiết bị phần cứng. Bạn có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị. Đối với Windows, xem trình điều khiển USB của OEM.

##### 1. Bật gỡ lỗi USB

Để Android Studio có thể giao tiếp với thiết bị của bạn, bạn cần bật tùy chọn gỡ lỗi USB trên thiết bị Android

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

- Trên thiết bị của bạn, mở Settings, tìm About phone và nhấn vào đó. Nhấn Build number 7 lần liên tiếp.
  - Quay lại màn hình trước và bạn sẽ thấy mục Developer options.
  - Bật gỡ lỗi USB.
2. Chạy ứng dụng của bạn trên thiết bị

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

- Kết nối thiết bị của bạn với máy phát triển bằng cáp USB.
- Nhấp vào nút Run trên thanh công cụ. Cửa sổ Select Deployment Target mở ra với danh sách các trình giả lập khả dụng và thiết bị được kết nối.
- Chọn thiết bị của bạn và nhấp vào OK.

Android Studio cài đặt và chạy ứng dụng trên thiết bị của bạn. Xử lý sự cố. Nếu Android Studio không nhận diện được thiết bị của bạn, hãy thử các cách sau.

- Rút cáp USB ra và cắm lại.
- Khởi động lại Android Studio.

Nếu máy tính vẫn không nhận diện thiết bị hoặc báo "Không được ủy quyền" hãy làm như sau:

- Rút cáp USB ra.
- Trên thiết bị Android, mở Developer Options trong Settings.
- Nhấn Thu hồi quyền gỡ lỗi USB (Revoke USB Debugging authorizations).
- Kết nối lại thiết bị với máy tính qua USB.
- Khi có thông báo trên màn hình điện thoại, hãy cấp quyền gỡ lỗi USB.

Nếu máy tính vẫn không nhận diện thiết bị, bạn có thể cần cài đặt trình điều khiển USB tương ứng. Xem tài liệu Sử dụng thiết bị phần cứng để biết thêm chi tiết.

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

Trong bài này, bạn sẽ thay đổi một số thiết lập trong tệp build.gradle (Module: app) để học cách chỉnh sửa và đồng bộ hóa các thay đổi với dự án trong Android Studio



## 1. Thay đổi phiên bản SDK tối thiểu của ứng dụng

- Mở rộng thư mục Gradle Scripts nếu thư mục này chưa mở và nhấp đúp vào tệp build.gradle(Module:app). Nội dung tệp sẽ hiển thị trong trình soạn thảo mã
- Tìm khối defaultConfig và thay đổi giá trị minSdkVersion. Khi chỉnh sửa xong, Android Studio sẽ hiển thị một thanh thông báo với liên kết Sync Now

## 2. Đồng bộ hóa cấu hình Gradle mới

- Khi thay đổi các tệp cấu hình Gradle, Android Studio yêu cầu bạn đồng bộ dự án để nhập các thay đổi và kiểm tra lỗi build.
- Để đồng bộ hóa cấu hình Gradle mới, bạn có thể nhấp vào Sync Now trong thanh thông báo hoặc chọn biểu tượng Sync Project with Gradle Files trên thanh công cụ
- Khi quá trình đồng bộ hóa hoàn tất, Android Studio sẽ hiển thị thông báo Gradle build finished ở góc dưới bên trái.
- Nếu bạn muốn tìm hiểu thêm về Gradle, bạn có thể tham khảo tài liệu chính thức về Hệ thống build Gradle

## Nhiệm vụ 6: Thêm câu lệnh log vào ứng dụng

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh Log vào ứng dụng của mình để hiển thị thông báo trong bảng Logcat. Các thông báo log là một công cụ gỡ lỗi mạnh mẽ, giúp bạn kiểm tra giá trị, đường đi thực thi của mã, cũng như báo cáo lỗi.

### 1. Xem bảng Logcat

Để xem bảng Logcat, hãy nhấp vào tab Logcat ở cuối cửa sổ Android Studio như được hiển thị trong hình bên dưới

Trong bảng Logcat:

- Tab Logcat giúp mở và đóng bảng hiển thị thông tin về ứng dụng khi nó đang chạy. Nếu bạn thêm câu lệnh Log vào mã, các thông báo sẽ xuất hiện tại đây.
- Mức độ Log mặc định là Verbose, hiển thị tất cả thông báo log. Bạn cũng có thể chọn các mức khác như Debug, Error, Info, và Warn để lọc thông tin hiển thị

### 2. Thêm câu lệnh log vào ứng dụng

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

Trong đó:

- Log: Lớp Log dùng để gửi thông báo log đến bảng Logcat.

- d: Mức độ Debug (có thể thay bằng e cho lỗi, w cho cảnh báo, i cho thông tin).
- "MainActivity": Tag (thẻ) giúp lọc thông báo trong Logcat. Thông thường, thẻ là tên của Activity, nhưng bạn có thể đặt tên khác tùy theo mục đích gỡ lỗi.
- "Hello world": Thông báo log thực tế.

### Các bước thực hiện

- Mở ứng dụng Hello World trong Android Studio và mở tệp MainActivity.java.
- Để tự động nhập các thư viện cần thiết (như android.util.Log), vào File > Settings (Windows) hoặc Android Studio > Preferences (macOS).
- Chọn Editor > General > Auto Import, đánh dấu tất cả checkbox và đặt Insert imports on paste thành All.
- Nhấn Apply rồi chọn OK.
- Trong phương thức onCreate() của MainActivity, thêm câu lệnh sau:

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

- Nếu bảng Logcat chưa mở, nhấp vào tab Logcat ở phía dưới Android Studio.
- Kiểm tra tên của thiết bị mục tiêu và tên gói ứng dụng có chính xác không.
- Chuyển mức Log trong Logcat sang Debug (hoặc để Verbose nếu số lượng log ít).
- Chạy ứng dụng.

2025-02-26 23:07:18.109 12981-12981 MainActivity com.example.helloworld D Hello World

### Tổng kết

- 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, hãy đặt API 24 ("Nougat"; Android 7.0)
- Để xem cấu trúc Android của ứng dụng trong Project pane, nhấp vào tab Project trong cột dọc, sau đó chọn Android trong menu bật lên ở trên cùng
- Chỉnh sửa tệp build.gradle (Module: app) khi cần thêm thư viện mới hoặc thay đổi phiên bản thư viện
- Tất cả mã và tài nguyên cho ứng dụng đều nằm trong thư mục app và res. Thư mục java bao gồm các hoạt động, thử nghiệm và các thành phần khác trong mã nguồn Java. Thư mục res chứa các tài nguyên, chẳng hạn như bố cục, chuỗi và hình ảnh.

- Chỉnh sửa tệp AndroidManifest.xml để thêm các thành phần và quyền (permissions) cho ứng dụng. Tất cả thành phần của ứng dụng, như nhiều Activity, phải được khai báo trong tệp này
- Sử dụng Android Virtual Device (AVD) Manager để tạo thiết bị ảo (emulator) để chạy ứng dụng
- Thêm Log statements vào ứng dụng để hiển thị thông báo trong Logcat, giúp gỡ lỗi cơ bản
- Để chạy ứng dụng của bạn trên thiết bị Android vật lý bằng Android Studio, hãy bật USB Debugging trên thiết bị. Mở Settings > About phone và nhấn vào Build number 7 lần. Quay lại màn hình trước đó và chọn vào Developer options. Chọn USB Debugging

## 1.2) Giao diện người dùng tương tác đầu tiên

### Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị Android bao gồm một hệ thống phân cấp các đối tượng gọi là View - mỗi phần tử trên màn hình là một View. Lớp View đại diện cho khối xây dựng cơ bản của tất cả các thành phần UI và là lớp cơ sở cho các lớp cung cấp thành phần UI tương tác như nút bấm (Button), ô kiểm (Checkbox), và trường nhập văn bản (Text Entry Field).

Các lớp con View thường được sử dụng, được đề cập trong nhiều bài học, bao gồm:

- TextView để hiển thị văn bản.
- EditText để người dùng nhập và chỉnh sửa văn bản.
- Button và các phần tử có thể nhấp khác (RadioButton, CheckBox, Spinner) để cung cấp hành vi tương tác.
- ScrollView và RecyclerView để hiển thị danh sách 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 UI được chứa trong một lớp mở rộng từ Activity. Một Activity thường được liên kết với một tệp XML định nghĩa bố cục của UI. Tệp XML này thường được đặt tên theo Activity của nó và định nghĩa bố cục của các thành phần View trên màn hình

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị một bố cục được định nghĩa trong tệp activity\_main.xml, trong đó có TextView hiển thị dòng chữ "Hello World".

Trong các ứng dụng phức tạp hơn, một Activity có thể thực hiện các hành động như xử lý tương tác của người dùng, vẽ nội dung đồ họa, hoặc yêu cầu dữ liệu từ cơ sở dữ liệu hoặc internet.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên - một ứng dụng cho phép người dùng tương tác. Bạn sẽ tạo ứng dụng sử dụng mẫu Empty Views Activity, sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế giao diện và chỉnh sửa bố cục trong XML.

## **Những gì bạn cần biết trước**

Bạn nên quen thuộc với:

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

## **Những gì bạn sẽ học**

- Cách tạo một ứng dụng có hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế giao diện.
- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới - hãy tham khảo từ điển thuật ngữ và khái niệm để hiểu rõ hơn.

## **Những gì bạn sẽ làm**

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

### **1. Tạo dự án Android Studio**

- Mở Android Studio và tạo một dự án mới với các thông số sau:

Thuộc tính	Giá trị
Name	HelloToast
Package Name	com.example.hellotoast
Minimum SDK	API 24 (“Nougat”; Android 7.0)
Template	Empty Views Activity

- Chọn Run > Run App hoặc nhấp vào biểu tượng Run trên thanh công cụ để xây dựng và thực thi ứng dụng trên trình giả lập hoặc thiết bị của bạn.

## 2. Khám phá trình chỉnh sửa bố cục (Layout Editor)

Android Studio cung cấp Layout Editor để giúp bạn xây dựng giao diện người dùng (UI) một cách nhanh chóng. Công cụ này cho phép bạn kéo các thành phần vào chế độ thiết kế hoặc bản thiết kế, định vị chúng, thêm ràng buộc, và thiết lập các thuộc tính. Ràng buộc giúp xác định vị trí của một phần tử trong bố cục. Ràng buộc có thể liên kết một phần tử với một phần tử khác, với bố cục cha, hoặc với một đường hướng dẫn vô hình.

Thực hành khám phá Layout Editor theo các bước sau:

- Trong Project > Android, điều hướng đến thư mục app > res > layout, sau đó nhấp đúp vào tệp activity\_main.xml để mở nó (nếu chưa mở).
- Chọn tab Design nếu chưa được chọn. Tab Design giúp bạn thao tác trực tiếp với bố cục, trong khi tab Text cho phép chỉnh sửa mã XML.
- Palettes hiển thị các thành phần UI có thể sử dụng trong bố cục của ứng dụng.
- Component Tree hiển thị cấu trúc cây của các phần tử UI. Mỗi phần tử UI là một View, có thể là cha hoặc con.
- Pane Design và Blueprint hiển thị các thành phần UI trong bố cục. Mặc định, ứng dụng sẽ có một TextView hiển thị "Hello World".
- Tab Attributes hiển thị thuộc tính của một phần tử UI và cho phép bạn chỉnh sửa chúng.

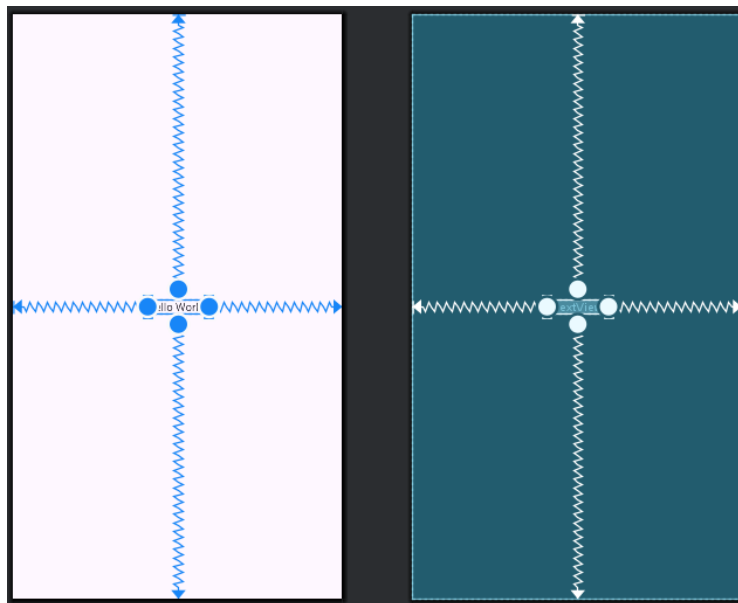
## Nhiệm vụ 2: Thêm các thành phần View trong trình chỉnh sửa bố cục

Trong nhiệm vụ này, bạn tạo bố cục UI cho ứng dụng HelloToast trong trình chỉnh sửa bố cục bằng các tính năng ConstraintLayout. Bạn có thể tạo các ràng buộc theo cách thủ công, hoặc tự động bằng công cụ Autoconnect

### 1. Kiểm tra ràng buộc của phần tử

Thực hiện các bước sau:

- Mở tệp `activity_main.xml` từ Project > Android, nếu chưa mở. Nếu chưa chọn tab Design, hãy chọn nó. Nếu không thấy chế độ bản thiết kế (Blueprint), nhấp vào Select Design Surface trên thanh công cụ và chọn Design + Blueprint.
- Đảm bảo công cụ Autoconnect không bị tắt. Công cụ này nằm trên thanh công cụ và mặc định được bật.
- Phóng to bố cục bằng cách nhấp vào nút Zoom In để nhìn rõ hơn bố cục và bản thiết kế.
- Chọn phần tử TextView trong Component Tree. TextView "Hello World" sẽ được đánh dấu trong cả Design và Blueprint, và các ràng buộc (constraints) của nó sẽ hiển thị.
- Nhấp vào tay cầm hình tròn ở bên phải của TextView để xóa ràng buộc theo chiều ngang liên kết chế độ xem với bên phải của bố cục. TextView nhảy sang bên trái vì nó không còn bị ràng buộc ở bên phải nữa. Để thêm lại ràng buộc theo chiều ngang, hãy nhấp vào cùng tay cầm đó và kéo một đường sang bên phải của bố cục



Các tay cầm điều chỉnh trên TextView:

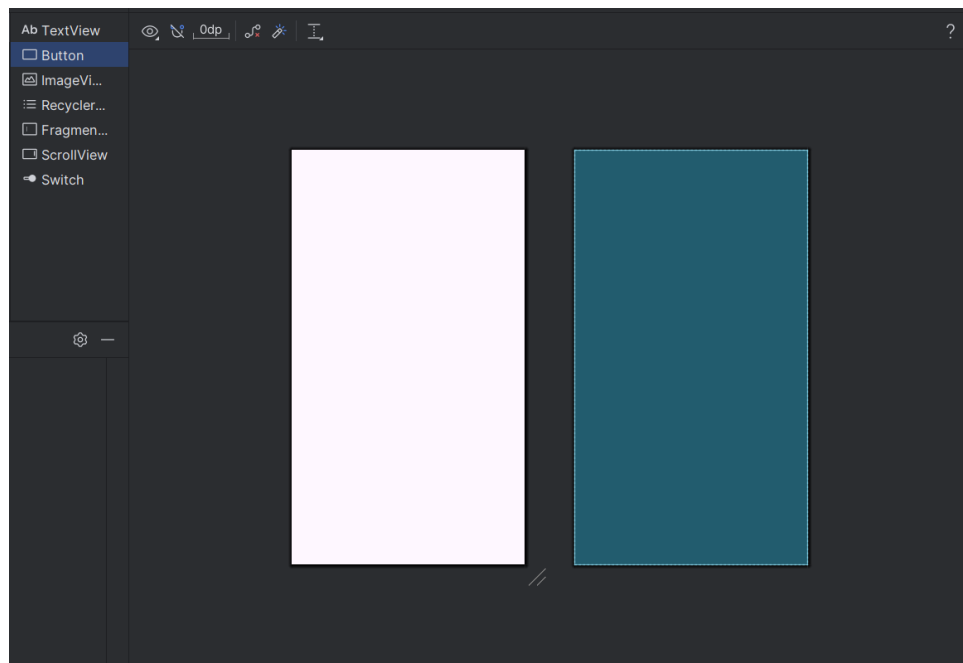
- Constraint Handle (Tay cầm ràng buộc):
  - Nhấp vào vòng tròn nhỏ trên cạnh của phần tử để tạo ràng buộc.
  - Kéo tay cầm này đến một phần tử khác hoặc cạnh của bố cục để thiết lập ràng buộc (hiển thị dưới dạng đường zigzag).
- Resizing Handle (Tay cầm thay đổi kích thước):
  - Nhấp và kéo các tay cầm hình vuông ở góc phần tử để thay đổi kích thước.
  - Trong khi kéo, tay cầm này sẽ thay đổi thành góc nghiêng

## 2. Thêm một Button vào bố cục

Khi được bật, công cụ Autoconnect sẽ tự động tạo hai hoặc nhiều ràng buộc (constraints) giữa một phần tử giao diện và bố cục cha. Khi bạn kéo một phần tử vào bố cục, nó sẽ tự động tạo các ràng buộc dựa trên vị trí của phần tử đó.

Thực hiện các bước sau để thêm một Button:

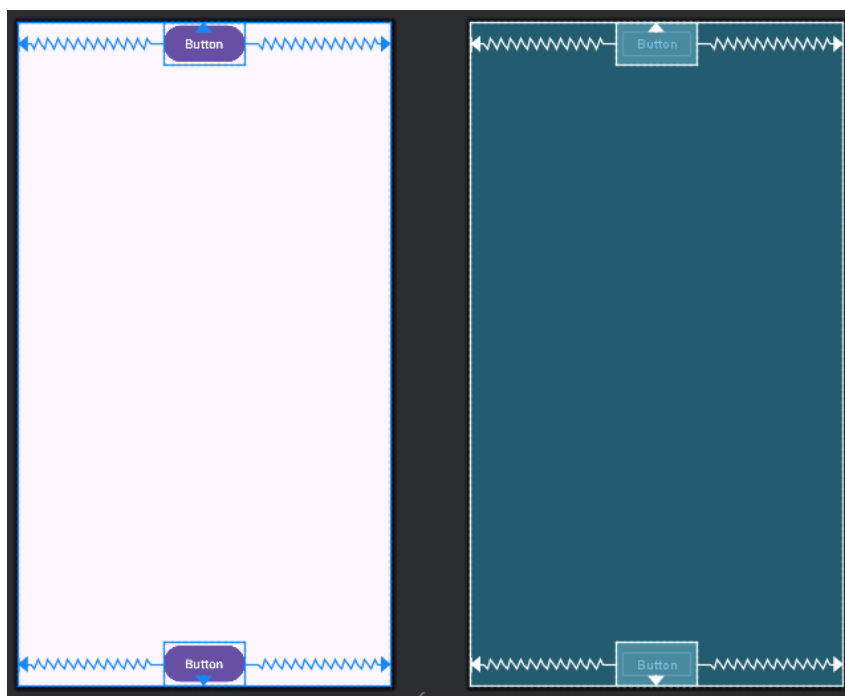
- Bắt đầu với một bố cục trống. Phần tử TextView không còn cần thiết, vì vậy hãy chọn nó và nhấn phím Delete hoặc vào Edit > Delete để xóa. Sau khi xóa, bố cục của bạn sẽ hoàn toàn trống.
- Tiếp theo, kéo một Button từ bảng Palette vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả Button vào khu vực giữa phía trên của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc thủ công đến mép trên, mép trái và mép phải của bố cục để đảm bảo Button được căn chỉnh chính xác.



## 3. Thêm một Button thứ hai vào bố cục

- Tiếp tục bằng cách kéo một Button thứ hai từ bảng Palette vào giữa bố cục. Công cụ Autoconnect có thể tự động tạo các ràng buộc ngang cho Button này. Nếu không, bạn có thể tự kéo các ràng buộc ngang để đảm bảo Button được căn chỉnh đúng cách.
- Sau đó, kéo một ràng buộc dọc từ Button thứ hai xuống mép dưới của bố cục để giữ cho nó cố định ở vị trí mong muốn.

- Nếu bạn muốn xóa ràng buộc khỏi một phần tử, hãy chọn phần tử đó và di chuyển chuột qua để hiển thị nút Clear Constraints, sau đó nhấp vào nút này để xóa tất cả ràng buộc của phần tử đã chọn. Nếu bạn chỉ muốn xóa một ràng buộc cụ thể, hãy nhấp vào tay cầm tạo ràng buộc đó để loại bỏ nó.
- Nếu bạn muốn xóa tất cả ràng buộc trong toàn bộ bố cục, hãy nhấp vào công cụ Clear All Constraints trên thanh công cụ. Công cụ này rất hữu ích khi bạn muốn thiết lập lại toàn bộ ràng buộc và bắt đầu căn chỉnh lại bố cục từ đầu.



### Nhiệm vụ 3: Thay đổi thuộc tính của phần tử giao diện

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

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

#### 1. Thay đổi kích thước của Button

Trình chỉnh sửa bố cục (Layout Editor) cung cấp các tay cầm thay đổi kích thước ở bốn góc của một View, giúp bạn có thể thay đổi kích thước nhanh chóng. Bạn có thể kéo các tay cầm ở mỗi góc của View để thay đổi kích thước, nhưng cách này sẽ gán cứng (hardcode) các kích thước chiều rộng và chiều cao. Việc gán cứng kích thước cho hầu hết



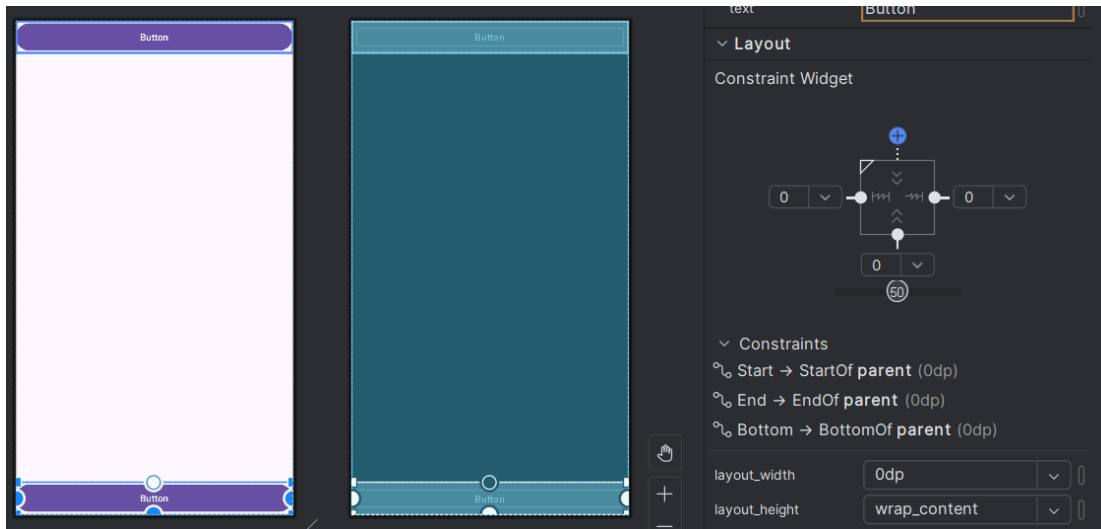
các phần tử View không được khuyến khích vì chúng không thể tự điều chỉnh theo nội dung hoặc kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng bảng Attributes ở bên phải trình chỉnh sửa bố cục để chọn một chế độ kích thước không sử dụng giá trị cố định. Bảng Attributes có một bảng kiểm tra kích thước (view inspector) ở phía trên cùng, hiển thị các biểu tượng đại diện cho các cài đặt chiều cao và chiều rộng như sau:

- Điều khiển chiều cao
  - Điều khiển này xác định thuộc tính `layout_height` và xuất hiện ở hai phần trên và dưới của hình vuông.
  - Các góc nghiêng cho biết giá trị đang được đặt là `wrap_content`, có nghĩa là view sẽ mở rộng theo chiều dọc khi cần để phù hợp với nội dung của nó.
  - Số "8" cho biết một lề tiêu chuẩn được đặt là 8dp.
- Điều khiển chiều rộng
  - Điều khiển này xác định thuộc tính `layout_width` và xuất hiện ở hai phần bên trái và bên phải của hình vuông.
  - Các góc nghiêng cho biết giá trị đang được đặt là `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần để phù hợp với nội dung, tối đa đến một lề 8dp.
- Nút đóng bảng Attributes
  - Nhấp vào nút này để đóng bảng Attributes.

Các bước thực hiện:

- Chọn Button đầu tiên trong bảng Component Tree.
- Nhấp vào tab Attributes ở bên phải cửa sổ trình chỉnh sửa bố cục.
- Nhấp vào điều khiển chiều rộng hai lần:
  - Lần nhấp đầu tiên thay đổi nó thành Fixed (đường thẳng).
  - Lần nhấp thứ hai thay đổi nó thành Match Constraints (hình lò xo)
  - Khi thay đổi điều khiển chiều rộng, thuộc tính `layout_width` trong bảng Attributes sẽ có giá trị `match_constraint`, và phần tử Button sẽ mở rộng theo chiều ngang để lấp đầy không gian giữa cạnh trái và cạnh phải của bố cục.
- Chọn Button thứ hai và thực hiện các thay đổi tương tự đối với thuộc tính `layout_width` như bước trước.



Khi bạn thay đổi điều khiển chiều rộng và chiều cao trong view inspector, các thuộc tính `layout_width` và `layout_height` trong bảng Attributes sẽ thay đổi tương ứng. Các thuộc tính này có thể nhận một trong ba giá trị sau trong `ConstraintLayout`:

- `match_constraint`:
  - Mở rộng phần tử View để lấp đầy bố cục cha theo chiều rộng hoặc chiều cao, tối đa đến phần lề nếu có.
  - Trong trường hợp này, `ConstraintLayout` là bố cục cha.
- `wrap_content`:
  - Thu nhỏ phần tử View sao cho vừa đủ để chứa nội dung của nó.
  - Nếu không có nội dung, View sẽ trở nên vô hình.
- Kích thước cố định (Fixed Size):
  - Nếu bạn muốn đặt kích thước cố định nhưng vẫn điều chỉnh phù hợp với màn hình thiết bị, hãy sử dụng đơn vị dp (density-independent pixels).
  - Ví dụ: 16dp có nghĩa là 16 pixel độc lập với mật độ màn hình.

## 2. Thay đổi thuộc tính của Button

Để xác định từng View một cách duy nhất 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, các phần tử Button cũng cần có văn bản hiển thị. Các phần tử View cũng có thể có nền là màu hoặc hình ảnh.

Bảng Attributes cung cấp quyền truy cập vào tất cả các thuộc tính mà bạn có thể gán cho một phần tử View. Bạn có thể nhập giá trị cho từng thuộc tính như:

- Sau khi chọn Button đầu tiên, hãy chỉnh sửa trường ID ở đầu ngăn Thuộc tính thành `button_toast` cho thuộc tính `android:id`, được sử dụng để xác định phần tử trong bố cục

- Đặt thuộc tính background thành `@color/colorPrimary`. (Khi bạn nhập `@c`, các lựa chọn sẽ xuất hiện để dễ dàng lựa chọn.)
- Đặt thuộc tính textColor thành `@android:color/white`
- Chỉnh sửa thuộc tính text thành Toast.
- Thực hiện các thay đổi thuộc tính tương tự cho Nút thứ hai, sử dụng `button_count` làm ID, Count cho thuộc tính văn bản và cùng màu cho nền và văn bản như các bước trước

`colorPrimary` là màu chủ đạo của giao diện ứng dụng, một trong những màu cơ bản được định nghĩa trong tệp tài nguyên `colors.xml`. Nó thường được sử dụng cho thanh ứng dụng (App Bar). Việc sử dụng màu cơ bản cho các phần tử giao diện khác giúp tạo ra sự đồng nhất trong giao diện người dùng. Bạn sẽ tìm hiểu thêm về chủ đề ứng dụng (App Themes) và Material Design trong bài học sau.

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

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

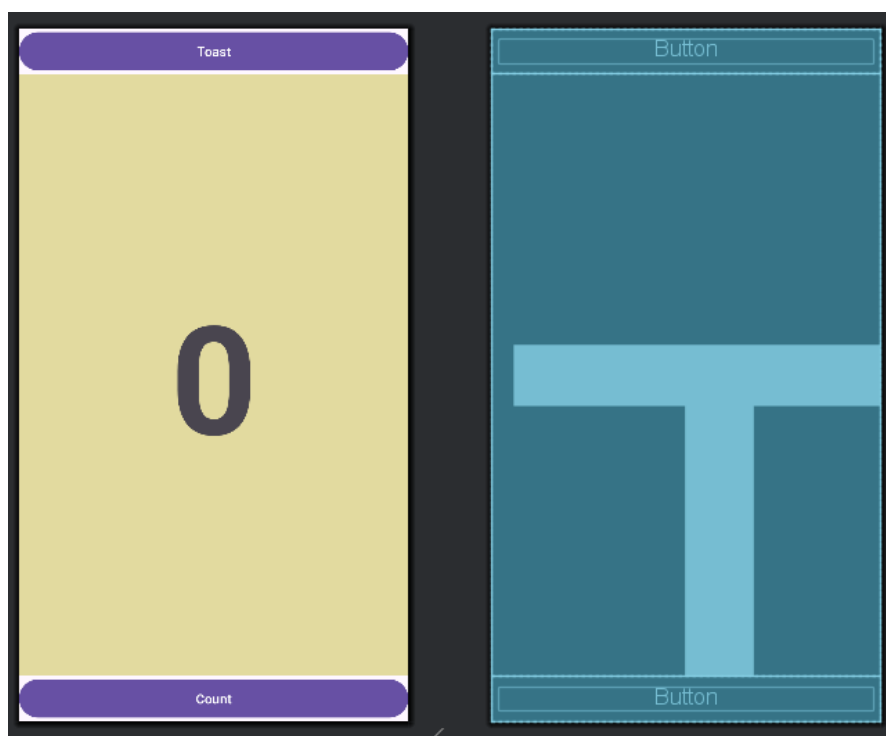
1. Thêm một `TextView` và thiết lập ràng buộc
  - Từ bảng `Palette`, kéo một `TextView` vào phần trên của bố cục. Kéo một constraint từ đỉnh của `TextView` đến chốt (handle) ở đáy của `Button Toast`. Điều này ràng buộc `TextView` nằm ngay bên dưới `Button Toast`.
  - Kéo một constraint từ đáy của `TextView` đến chốt trên của `Button Count`. Kéo thêm các ràng buộc từ hai bên của `TextView` đến hai bên của bố cục. Điều này giữ cho `TextView` nằm chính giữa bố cục, giữa hai nút `Button`.
2. Thiết lập thuộc tính cho `TextView`

Với `TextView` đã chọn, mở bảng `Attributes` (nếu chưa mở) và thực hiện các thay đổi sau:

- Đặt ID thành `show_count`.
- Đặt text thành "0".
- Đặt `textSize` thành 160sp.
- Đặt `textStyle` thành B (bold) và `textAlignment` thành `ALIGNCENTER` (căn giữa đoạn văn bản).
- Thay đổi `layout_width` và `layout_height` thành `match_constraint` để `TextView` mở rộng theo kích thước ràng buộc.
- Đặt `textColor` thành `@color/colorPrimary`.

- Cuộn xuống bảng Attributes, nhấn View all attributes. Tìm thuộc tính background, nhập #FFF00 để đặt nền màu vàng nhạt.
- Cuộn xuống gravity, mở rộng danh sách và chọn center\_ver để căn giữa theo chiều dọc.
  - textSize: Kích thước văn bản của TextView, trong bài này đặt là 160sp. Đơn vị sp (scale-independent pixel) giúp kích thước văn bản tự động điều chỉnh theo mật độ màn hình và tùy chọn kích thước phông chữ của người dùng. Tương tự như dp, nhưng chuyên dùng cho văn bản.
  - textStyle: Định dạng văn bản, đặt là B (bold) (chữ đậm).
  - textAlignment: Căn chỉnh văn bản, đặt là ALIGNCENTER (căn giữa).
  - gravity: Thuộc tính này căn chỉnh vị trí của View trong View cha (ConstraintLayout). Trong bước này, ta đặt TextView căn giữa theo chiều dọc trong ConstraintLayout.

Bạn có thể nhận thấy thuộc tính background nằm trên trang đầu tiên của bảng Attributes đối với Button, nhưng nằm trên trang thứ hai đối với TextView. Điều này xảy ra vì bảng Attributes thay đổi tùy theo loại View, trong đó: Các thuộc tính phổ biến nhất xuất hiện trên trang đầu tiên. Các thuộc tính ít phổ biến hơn nằm trên trang thứ hai. Để quay lại trang đầu tiên của bảng Attributes, hãy nhấn vào biểu tượng trên thanh công cụ ở đầu bảng.



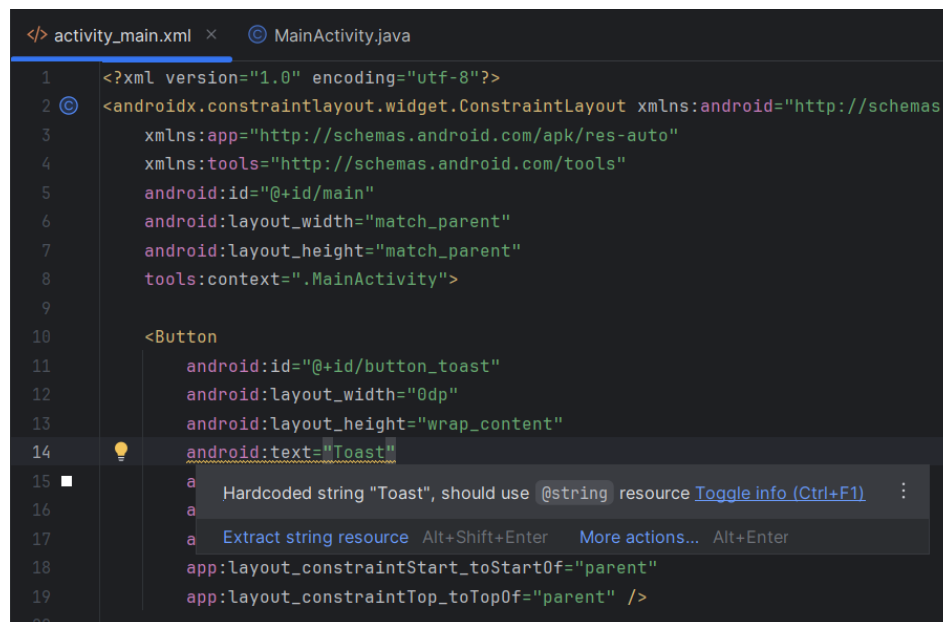
## Nhiệm vụ 5: Chỉnh sửa bố cục trong XML

Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử UI trong Cây thành phần. Di con trỏ qua các dấu chấm than này để xem các thông báo cảnh báo, như hiển thị bên dưới. Cảnh báo tương tự xuất hiện cho cả ba phần tử: các chuỗi được mã hóa cứng phải sử dụng tài nguyên.

Việc chỉnh sửa bố cục trong XML giúp giải quyết vấn đề này nhanh chóng. Trình chỉnh sửa layout rất mạnh mẽ, nhưng một số thay đổi sẽ dễ thực hiện hơn trực tiếp trong mã XML.

## 1. Mở mã XML của bố cục

- Mở file `activity_main.xml` (nếu chưa mở).
- Nhấn vào tab Text ở cuối trình chỉnh sửa bố cục.
- Kiểm tra cảnh báo:
- Trình chỉnh sửa XML sẽ hiển thị thay thế phần Design và Blueprint.
- Các cảnh báo xuất hiện trên các chuỗi mã hóa cứng như "Toast", "Count", và "0".
- Di chuột qua chuỗi "Toast" để xem thông báo cảnh báo.



## 2. Trích xuất tài nguyên chuỗi

Thay vì mã hóa cứng (hard-coding), ta nên sử dụng tài nguyên chuỗi (string resources). Điều này giúp dễ dàng quản lý khi sử dụng lại chuỗi nhiều lần. Bắt buộc khi dịch và bản địa hóa ứng dụng, vì bạn có thể tạo file tài nguyên chuỗi cho từng ngôn ngữ.

Các bước thực hiện:

- Nhấp một lần vào từ "Toast" (cảnh báo được tô sáng đầu tiên).
- Nhấn Alt-Enter trong Windows hoặc Option-Enter trong macOS và chọn Trích xuất tài nguyên chuỗi từ menu bật lên.
- Nhập button\_label\_toast cho tên Tài nguyên.
- Nhấp vào OK. Một tài nguyên chuỗi được tạo trong tệp values/res/string.xml và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: @string/button\_label\_toast
- Trích xuất các chuỗi còn lại: button\_label\_count cho "Count" và count\_initial\_value cho "0".
- Trong Project > Android, hãy mở rộng các giá trị trong res, sau đó nhấp đúp vào strings.xml để xem các tài nguyên chuỗi của bạn trong tệp strings.xml:
- Bạn cần một chuỗi khác để sử dụng trong tác vụ tiếp theo hiển thị thông báo. Thêm vào tệp strings.xml một tài nguyên chuỗi khác có tên toast\_message cho cụm từ "Hello Toast!":

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

## Task 6: Thêm xử lý sự kiện onClick cho các Button

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

### 1. Thêm thuộc tính onClick và trình xử lý vào mỗi Nút

Trình xử lý nhấp là phương thức được gọi khi người dùng nhấp hoặc chạm vào phần tử UI có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick trong ngăn Thuộc tính của tab Thiết kế. Bạn cũng có thể chỉ định tên của phương thức xử lý trong trình soạn thảo XML bằng cách thêm thuộc tính android:onClick vào Nút. Bạn sẽ sử dụng phương thức sau vì bạn vẫn chưa tạo các phương thức xử lý và trình soạn thảo XML cung cấp một cách tự động để tạo các phương thức đó.

- Khi trình soạn thảo XML mở (tab Văn bản), hãy tìm Nút có android:id được đặt thành button\_toast:
- Thêm thuộc tính android:onClick vào cuối phần tử button\_toast sau thuộc tính cuối cùng và trước chỉ báo kết thúc />:
- Nhấp vào biểu tượng bóng đèn màu đỏ xuất hiện bên cạnh thuộc tính. Chọn Create click handler, chọn MainActivity và nhấp vào OK.

- Nếu biểu tượng bóng đèn màu đỏ không xuất hiện, hãy nhấp vào tên phương thức ("showToast"). Nhấn Alt-Enter (Option-Enter trên máy Mac), chọn Create 'showToast(view)' trong MainActivity và nhấp vào OK.
- Hành động này tạo một stub phương thức giữ chỗ cho phương thức showToast() trong MainActivity, như được hiển thị ở cuối các bước này.
- Lặp lại hai bước cuối cùng với Button\_count Button: Thêm thuộc tính android:onClick vào cuối và thêm trình xử lý nhấp: Mã XML cho các thành phần UI trong ConstraintLayout bây giờ trông như thế này:

```
<Button
    android:id="@+id/button_label_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Toast"
    android:textColor="@color/white"
    android:textSize="14sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="showToast"/>
<Button
    android:id="@+id/button_label_count"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Count"
    android:textColor="@color/white"
    android:textSize="14sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:onClick="countUp"/>
```

- Mở rộng com.example.android.hellotoast, sau đó nhấp đúp vào MainActivity. Trình chỉnh sửa mã xuất hiện với mã trong MainActivity:

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

```

    });
}
public void countUp(View view) {
}
public void showToast(View view) {
}
}

```

## 2. Chỉnh sửa trình xử lý Button Toast

Bây giờ bạn sẽ chỉnh sửa phương thức `showToast()`—trình xử lý sự kiện khi nhấn vào nút Toast trong MainActivity—để hiển thị một thông báo. Toast cung cấp một cách để hiển thị một thông báo đơn giản trong một cửa sổ popup nhỏ. Nó chỉ chiếm không gian cần thiết cho thông báo, trong khi hoạt động hiện tại vẫn hiển thị và có thể tương tác. Toast có thể hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn—bạn có thể thêm một thông báo Toast để hiển thị kết quả khi nhấn vào Button hoặc thực hiện một hành động.

Thực hiện các bước sau để chỉnh sửa trình xử lý sự kiện Toast Button:

- Xác định phương thức `showToast()` mới được tạo.
- Để tạo một thể hiện của Toast, gọi phương thức `makeText()` của lớp Toast.
- Cung cấp context của Activity ứng dụng. Vì Toast hiển thị trên UI của Activity, hệ thống cần thông tin về Activity hiện tại. Khi đang ở trong Activity cần dùng context, có thể sử dụng `this` làm lỗi tắt.
- Cung cấp thông báo cần hiển thị, chẳng hạn như một string resource (chuỗi `toast_message` đã tạo ở bước trước). Chuỗi tài nguyên `toast_message` được xác định bởi `R.string.toast_message`.
- Cung cấp thời gian hiển thị. Ví dụ, `Toast.LENGTH_SHORT` sẽ hiển thị Toast trong thời gian tương đối ngắn.
  - `Toast.LENGTH_LONG` hiển thị khoảng 3.5 giây.
  - `Toast.LENGTH_SHORT` hiển thị khoảng 2 giây.
- Hiển thị Toast bằng cách gọi phương thức `show()`.

```

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

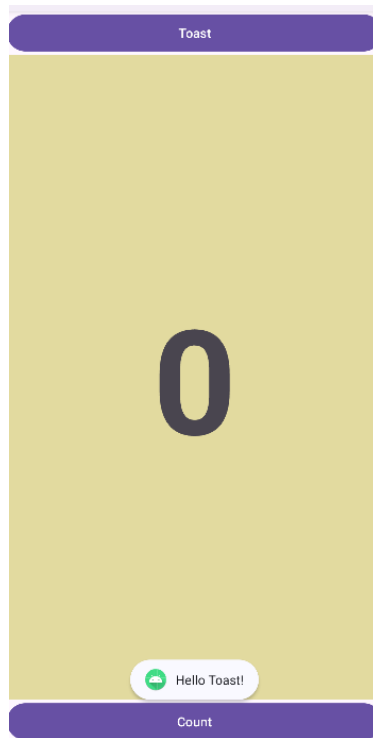
```



```

}
public void showToast(View view) {
    Toast toast = Toast.makeText(this, "Hello Toast!", Toast.LENGTH_SHORT);
    toast.show();
}
}

```



### 3. Chỉnh sửa trình xử lý Button Count

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

Mã xử lý sự kiện phải:

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

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

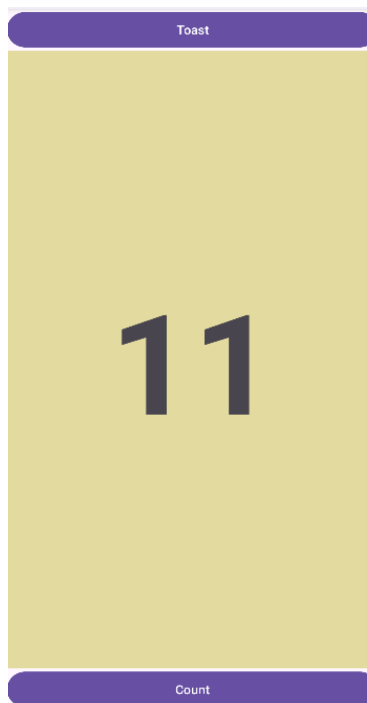
- Xác định phương thức `countUp()` mới được tạo.
- Để theo dõi số lần đếm, bạn cần một biến thành viên private. Mỗi lần nhấn nút Count, giá trị của biến này sẽ tăng lên. Nhập đoạn mã sau, nó sẽ được tô đỏ và hiển thị biểu tượng bóng đèn màu đỏ.

- Nếu biểu tượng bóng đèn không xuất hiện, hãy chọn biểu thức `mCount++`. Cuối cùng, bóng đèn sẽ xuất hiện.
- Nhấn vào biểu tượng bóng đèn đỏ và chọn Create field 'mCount' từ menu popup. Điều này sẽ tạo một biến thành viên private ở đầu MainActivity, và Android Studio sẽ mặc định đặt kiểu của nó là int.
- Thay đổi câu lệnh khai báo biến thành viên private để khởi tạo biến này với giá trị 0.
- Cùng với biến trên, bạn cũng cần một biến thành viên private để tham chiếu đến `show_count TextView`, biến này sẽ được thêm vào trình xử lý sự kiện. Đặt tên biến này là `mShowCount`.
- Khi đã có `mShowCount`, bạn có thể lấy tham chiếu đến TextView bằng ID đã đặt trong tệp layout. Để chỉ lấy tham chiếu này một lần, hãy xác định nó trong phương thức `onCreate()`. Như bạn sẽ học trong một bài học khác, phương thức `onCreate()` được sử dụng để inflate layout, có nghĩa là thiết lập content view của màn hình thành bố cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử UI khác trong bố cục, chẳng hạn như TextView. Xác định phương thức `onCreate()` trong MainActivity.
- Thêm câu lệnh `findViewById` vào cuối phương thức: Một View, giống như một chuỗi, là một tài nguyên có thể có id. Lệnh `findViewById` nhận ID của một View làm tham số và trả về View đó. Vì phương thức trả về một View, bạn phải ép kiểu kết quả về kiểu TextView.
- Khi đã gán TextView cho biến `mShowCount`, bạn có thể sử dụng biến này để thiết lập nội dung văn bản trong TextView bằng giá trị của biến `mCount`. Thêm đoạn mã sau vào phương thức `countUp()`.

```
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 countUp(View view) {
        mCount++;
        if (mShowCount != null)
```

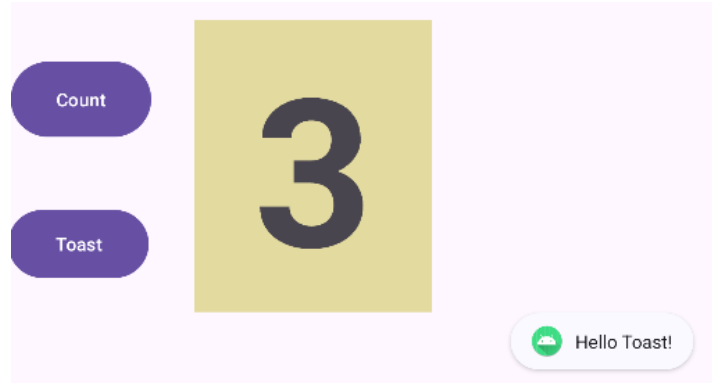
```
mShowCount.setText(Integer.toString(mCount));  
}  
}
```



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

Thay đổi bố cục để ứng dụng hiển thị tốt ở cả hai chế độ ngang và dọc:

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



## Tổng kết

Tóm tắt về View, ViewGroup và bố cục:

- Tất cả các phần tử giao diện người dùng (UI) là các lớp con của lớp View và 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 container. Mỗi quan hệ giữa chúng là cha-con, trong đó cha là ViewGroup và con là một View hoặc một ViewGroup khác.
- Phương thức onCreate() được sử dụng để inflate bố cục, nghĩa là thiết lập nội dung hiển thị trên màn hình theo tệp XML bố cục. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử UI khác trong bố cục.
- Một View, giống như một chuỗi ký tự (string), là một tài nguyên có thể có ID. Lệnh findViewById nhận ID của một View làm tham số và trả về View đó.

Sử dụng trình chỉnh sửa bố cục:

- Nhấp vào tab Design để thao tác với các phần tử và bố cục, hoặc vào tab Text để chỉnh sửa mã XML của bố cục.
- Trong tab Design, Palettes pane hiển thị các phần tử UI có thể sử dụng trong bố cục ứng dụng, Component tree pane hiển thị cây phân cấp của các phần tử UI.
- Các bảng thiết kế (design pane) và bản vẽ (blueprint pane) trong trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục.
- Tab Attributes hiển thị bảng thuộc tính, nơi bạn có thể đặt các thuộc tính cho một phần tử UI.
- Handle ràng buộc (constraint handle): Nhấp vào vòng tròn ở mỗi cạnh của phần tử để kéo và tạo ràng buộc với một handle khác hoặc với biên của phần tử cha. Ràng buộc được biểu diễn bằng đường gấp khúc.
- Handle thay đổi kích thước (resizing handle): Kéo các góc vuông để thay đổi kích thước phần tử. Khi kéo, handle sẽ thay đổi thành một góc nghiêng.
- Khi Autoconnect được bật, công cụ sẽ tự động tạo hai hoặc nhiều ràng buộc cho một phần tử UI với bố cục cha, dựa trên vị trí của phần tử.

- Bạn có thể xóa ràng buộc khỏi một phần tử bằng cách chọn phần tử đó, di chuyển con trỏ chuột lên để hiển thị nút Clear Constraints, sau đó nhấp vào nút này để xóa tất cả ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc cụ thể, nhấp vào handle đã tạo ràng buộc đó.
- Bảng Attributes cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Nó cũng bao gồm một bảng điều chỉnh kích thước hình vuông gọi là view inspector ở trên cùng. Các biểu tượng bên trong hình vuông đại diện cho cài đặt chiều cao và chiều rộng.

Thiết lập chiều rộng và chiều cao của bố cục:

- Các thuộc tính `layout_width` và `layout_height` thay đổi khi bạn điều chỉnh kích thước chiều cao và chiều rộng trong view inspector. Trong `ConstraintLayout`, các thuộc tính này có thể nhận một trong ba giá trị:
  - `match_constraint`: Mở rộng phần tử để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao, giới hạn ở phần margin nếu được đặt.
  - `wrap_content`: Thu nhỏ kích thước phần tử để vừa với nội dung của nó. Nếu không có nội dung, phần tử sẽ trở nên vô hình.
  - Sử dụng một số dp (density-independent pixels) cố định để chỉ định kích thước cố định, được điều chỉnh theo kích thước màn hình của thiết bị.

Trích xuất tài nguyên chuỗi: Thay vì mã hóa cứng (hardcoding) chuỗi, cách làm tốt nhất là sử dụng tài nguyên chuỗi (string resources). Làm theo các bước sau:

- Nhấp một lần vào chuỗi đã mã hóa cứng để trích xuất, nhấn `Alt+Enter` (hoặc `Option+Enter` trên Mac), và chọn `Extract string resources` từ menu bật lên.
- Đặt tên cho tài nguyên.
- Nhấn OK. Hành động này sẽ tạo một tài nguyên chuỗi trong tệp `values/res/string.xml` và thay thế chuỗi trong mã bằng một tham chiếu đến tài nguyên: `@string/button_label_toast`.

Xử lý sự kiện nhấp (click handler):

- Trình xử lý sự kiện nhấp (click handler) là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI.
- Chỉ định trình xử lý sự kiện nhấp cho một phần tử UI như `Button` bằng cách nhập tên phương thức vào trường `onClick` trong bảng Attributes của tab Design, hoặc trong trình chỉnh sửa XML bằng cách thêm thuộc tính `android:onClick` vào phần tử UI như `Button`.
- Tạo trình xử lý sự kiện nhấp trong Activity chính bằng cách sử dụng tham số `View`
- Bạn có thể tìm thấy thông tin về tất cả các thuộc tính của `Button` trong tài liệu của lớp `Button`, và tất cả các thuộc tính của `TextView` trong tài liệu của lớp `TextView`.

Hiển thị thông báo Toast: Toast cung cấp một cách hiển thị thông báo đơn giản trong một cửa sổ bật lên nhỏ, chỉ chiếm một lượng không gian cần thiết cho nội dung tin nhắn. Để tạo một đối tượng Toast, thực hiện các bước sau:

- Gọi phương thức `makeText()` trên lớp `Toast`.
- Cung cấp ngữ cảnh của `Activity` và thông điệp cần hiển thị (chẳng hạn như tài nguyên chuỗi).
- Cung cấp khoảng 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`.
- Gọi phương thức `show()` để hiển thị Toast.

### **1.3) Trình chỉnh sửa bố cục**

#### **Giới thiệu**

Trong 1.2 Giao diện tương tác đầu tiên của bạn, bạn có thể xây dựng giao diện người dùng (UI) bằng `ConstraintLayout` trong trình chỉnh sửa bố cục, bố trí các phần tử UI trong bố cục bằng cách sử dụng các kết nối ràng buộc với các phần tử khác và với các cạnh của bố cục. `ConstraintLayout` được thiết kế để giúp dễ dàng kéo các phần tử UI vào trình chỉnh sửa bố cục.

`ConstraintLayout` là một `ViewGroup`, một loại `View` đặc biệt có thể chứa các đối tượng `View` khác (gọi là con hoặc child views). Thực hành này sẽ giới thiệu thêm các tính năng của `ConstraintLayout` và trình chỉnh sửa bố cục.

Thực hành này cũng giới thiệu hai lớp con khác của `ViewGroup`:

**LinearLayout:** Một nhóm sắp xếp các phần tử `View` con bên trong nó theo chiều ngang hoặc chiều dọc.

**RelativeLayout:** Một nhóm các phần tử `View` con trong đó mỗi phần tử `View` được định vị và căn chỉnh tương đối với các phần tử `View` khác bên trong `ViewGroup`. Vị trí của các phần tử `View` con được mô tả liên quan đến nhau hoặc với `ViewGroup` cha.

#### **Những gì bạn cần biết trước**

Bạn cần có khả năng:

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

## Những gì bạn sẽ học

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



## Những gì bạn sẽ làm

- Tạo một biến thể bố cục cho màn hình ngang.
- Tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Chỉnh sửa bố cục để thêm các ràng buộc vào các phần tử UI.
- Sử dụng ràng buộc đường cơ sở 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 bố cục để sử dụng LinearLayout.
- Định vị các phần tử trong LinearLayout.
- Thay đổi bố cục để sử dụng RelativeLayout.
- Sắp xếp lại các views trong bố cục chính để chúng có quan hệ tương đối với nhau.














## Nhiệm vụ 1: Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu thay đổi bố cục của ứng dụng Hello Toast để phù hợp với chế độ hiển thị ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học một cách dễ dàng hơn để tạo các biến thể của bố cục cho chế độ ngang (còn gọi là landscape) và chế độ dọc (còn gọi là portrait) dành cho 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 bố cục. Thanh công cụ trên cùng cho phép bạn cấu hình giao diện xem trước bố cục trong trình chỉnh sửa bố cục:

- Chọn Bề mặt Thiết kế : Chọn Design để hiển thị bản xem trước màu của bố cục hoặc Blueprint để chỉ hiển thị đường viền của từng phần tử UI. Để xem cả hai gần cạnh nhau, chọn Design + Blueprint.
- Chế độ hiển thị trong Trình chỉnh sửa : Chọn Portrait hoặc Landscape để hiển thị bản xem trước theo chiều 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.

- Thiết bị trong Trình chỉnh sửa  Pixel ▾: Chọn loại thiết bị (điện thoại/máy tính bảng, Android TV hoặc Android Wear).
  - Phiên bản API trong Trình chỉnh sửa  35 ▾: Chọn phiên bản Android để hiển thị bản xem trước.
  - Chủ đề trong Trình chỉnh sửa  HelloToast ▾: Chọn một chủ đề (chẳng hạn như AppTheme) để áp dụng cho bản xem trước.
  - Ngôn ngữ trong Trình chỉnh sửa  Default (en-us) ▾: Chọn ngôn ngữ và khu vực cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có sẵn trong tài nguyên chuỗi (xem bài học về bản địa hóa để biết chi tiết cách thêm ngôn ngữ). Bạn cũng có thể chọn Preview as Right To Left để xem bố cục như thể một ngôn ngữ RTL đã được chọn.
  - Hiển thị : Chọn Show Constraints và Show Margins để hiển thị hoặc ẩn chúng trong bản xem trước.
  - Tự động kết nối : Bật hoặc tắt Autoconnect. Khi Autoconnect được bật, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như Button) đến bất kỳ phần nào của bố cục để tạo ràng buộc với bố cục cha.
  - Xóa tất cả ràng buộc : Xóa tất cả ràng buộc trong toàn bộ bố cục.
  - Suy luận ràng buộc : Tạo ràng buộc bằng cách suy luận.
  - Lề mặc định : Đặt lề mặc định.
  - Gói : Gói hoặc mở rộng các phần tử được chọn.
  - Căn chỉnh : Căn chỉnh các phần tử được chọn.
  - Đường hướng dẫn : Thêm đường hướng dẫn dọc hoặc ngang.
  - Điều khiển thu phóng/dịch chuyển : Thu phóng vào hoặc ra.
1. Xem trước bố cục ở chế độ ngang

Để xem trước bố cục của ứng dụng Hello Toast ở chế độ ngang, hãy làm theo các bước sau:

- Mở ứng dụng Hello Toast từ bài học trước.
- Mở tệp bố cục activity\_main.xml. Nhấp vào tab Design nếu nó chưa được chọn.



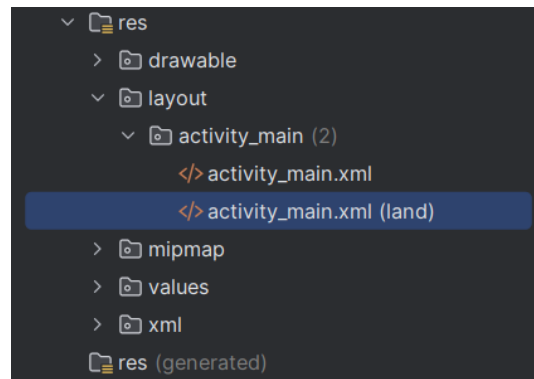
- Nhấp vào nút Orientation for Preview trên thanh công cụ trên cùng.
- Chọn Landscape trong menu thả xuống. Bố cục sẽ xuất hiện ở chế độ ngang như hình dưới đây. Để quay lại chế độ dọc, chọn Portrait.

## 2. Tạo một biến thể bố cục cho chế độ ngang

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

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

- Chọn Create Landscape Variation.
- Một cửa sổ trình chỉnh sửa mới sẽ mở với tab `land/activity_main.xml` hiển thị bố cục cho chế độ ngang. Bạn có thể thay đổi bố cục này mà không ảnh hưởng đến bố cục gốc dành cho chế độ dọc.
- Trong Project > Android, mở thư mục `res > layout`, bạn sẽ thấy rằng Android Studio đã tự động tạo biến thể cho bạn với tên `activity_main.xml (land)`.



## 3. Xem trước bố cục trên các thiết bị khác nhau

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

- Tab `land/activity_main.xml` vẫn đang mở trong trình chỉnh sửa bố cục; nếu không, nhấp đúp vào tệp `activity_main.xml (land)` trong thư mục `layout`.
- Nhấp vào nút Device for Preview trên thanh công cụ trên cùng.
- Chọn một thiết bị khác trong menu thả xuống. Ví dụ: chọn Nexus 4, Nexus 5, sau đó Pixel để thấy sự khác biệt trong các bản xem trước.

#### 4. Thay đổi bố cục cho hướng ngang

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

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

- Tab `land/activity_main.xml` vẫn phải mở trong trình chỉnh sửa bố cục; nếu chưa, nhấp đúp vào tệp `activity_main.xml` (land) trong thư mục bố cục.
- Nhấp vào tab Text và tab Preview (nếu chưa được chọn).
- Tìm phần tử `TextView` trong mã XML.
- Thay đổi thuộc tính `android:textSize="160sp"` thành `android:textSize="120sp"`. Bản xem trước bố cục sẽ hiển thị kết quả.



- Chọn các thiết bị khác nhau trong menu thả xuống Thiết bị trong Device for Preview để xem bố cục trông như thế nào trên các thiết bị khác nhau ở hướng ngang. Trong ngăn chỉnh sửa, tab `land/activity_main.xml` hiển thị bố cục cho hướng ngang. Tab `activity_main.xml` hiển thị bố cục không thay đổi cho hướng dọc. Bạn có thể chuyển qua lại bằng cách nhấp vào các tab.
  - Chạy ứng dụng trên trình giả lập hoặc thiết bị thật, sau đó chuyển đổi giữa hướng dọc và ngang để xem cả hai bố cục.
5. Tạo một biến thể bố cục cho máy tính bảng

Bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút Thiết bị trong Device for Preview trên thanh công cụ trên cùng. Nếu bạn chọn một thiết bị như Nexus 10 (máy tính bảng) từ menu, bạn sẽ thấy rằng bố cục không phù hợp với màn hình máy tính bảng—văn bản trên mỗi nút quá nhỏ, và cách sắp xếp các phần tử Button ở trên và dưới không phù hợp với màn hình lớn của máy tính bảng.

Để khắc phục điều này cho máy tính bảng mà không ảnh hưởng đến các bố cục ở kích thước điện thoại theo hướng ngang và dọc, bạn có thể tạo một biến thể bố cục hoàn toàn khác dành riêng cho máy tính bảng. Thực hiện các bước sau:

- Nhấp vào tab Design (nếu chưa được chọn) để hiển thị ngăn thiết kế và bản vẽ phác thảo.
- Nhấp vào nút Hướng trong Orientation for Preview trên thanh công cụ trên cùng.
- Chọn Tạo biến thể x-large.

Một cửa sổ chỉnh sửa mới sẽ mở với tab xlarge/activity\_main.xml hiển thị bố cục dành cho thiết bị có kích thước máy tính bảng. Trình chỉnh sửa cũng sẽ 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, dành riêng cho máy tính bảng, mà không ảnh hưởng đến các bố cục khác.

## 6. Thay đổi biến thể bố cục cho máy tính bảng

Bạn có thể sử dụng 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 các ràng buộc trong bố cục bằng cách nhấp vào nút Clear All Constraints trên thanh công cụ.  
Sau khi xóa ràng buộc, bạn có thể tự do di chuyển và thay đổi kích thước các phần tử trên bố cục.
- Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước ở cả bốn góc của một phần tử để thay đổi kích thước của nó. Trong Component Tree, chọn TextView có tên show\_count. Để di chuyển TextView ra khỏi đường đi và có thể kéo các phần tử Button tự do, hãy kéo một góc của nó để thay đổi kích thước, như hình minh họa bên dưới.  
Việc thay đổi kích thước một phần tử sẽ gán cố định các kích thước chiều rộng và chiều cao. Hạn chế sử dụng kích thước cố định cho hầu hết các phần tử, vì bạn không thể dự đoán được kích thước cố định sẽ hiển thị như thế nào trên các màn hình có kích thước và mật độ khác nhau. Ở bước này, bạn chỉ đang di chuyển phần tử ra khỏi đường đi và sẽ thay đổi kích thước sau.
- Chọn button\_toast trong Component Tree, nhấp vào tab Attributes để mở ngăn Thuộc tính, và thay đổi textSize thành 60sp và layout\_width thành wrap\_content. Bạn có thể nhấp vào điều khiển chiều rộng của trình kiểm tra chế độ xem xuất hiện ở hai đoạn 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.  
Bạn sử dụng wrap\_content để đảm bảo rằng nếu văn bản của Button được dịch sang một ngôn ngữ khác, nút sẽ rộng hơn hoặc hẹp hơn để phù hợp với từ trong ngôn ngữ khác.

- Chọn `button_count` trong Component Tree, thay đổi `textSize` thành 60sp và `layout_width` thành `wrap_content`, sau đó kéo Button lên trên TextView vào một khoảng trống trong bố cục.

#### 7. Sử dụng ràng buộc đường cơ sở (Baseline Constraint)

Bạn có thể căn chỉnh một phần tử giao diện người dùng (UI element) có chứa văn bản, chẳng hạn như TextView hoặc Button, với một phần tử giao diện người dùng khác cũng chứa văn bản. Một ràng buộc đường cơ sở (baseline constraint) cho phép bạn ràng buộc các phần tử sao cho các đường cơ sở của văn bản khớp nhau.

- Ràng buộc `button_toast` vào phía trên và bên trái của bố cục, kéo `button_count` vào một không gian gần `button_toast`, và ràng buộc `button_count` vào bên trái của `button_toast`
- Sử dụng một ràng buộc đường cơ sở, bạn có thể ràng buộc nút `button_count` sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của nút `button_toast`. Chọn phần tử `button_count`, sau đó di chuột qua phần tử cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.
- Nhấp vào nút ràng buộc đường cơ sở. Tay cầm đường cơ sở xuất hiện, nhấp nháy màu xanh lục như được hiển thị trong hình động. Nhấp và kéo một đường ràng buộc đường cơ sở đến đường cơ sở của phần tử `button_toast`.

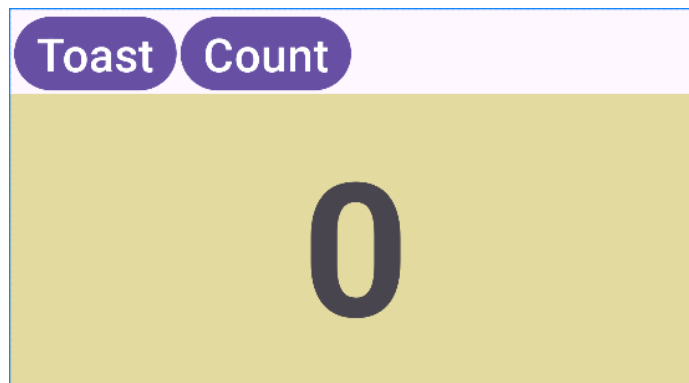
#### 8. Mở rộng các nút theo chiều ngang

Nút đóng gói trong thanh công cụ cung cấp các tùy chọn để đóng gói hoặc mở rộng các phần tử UI đã chọn. Bạn có thể sử dụng nó để sắp xếp các phần tử Button theo chiều ngang trên bố cục một cách đều nhau.

- Chọn nút `button_count` trong Component Tree, sau đó nhấn Shift và chọn nút `button_toast` để chọn cả hai.
- Nhấp vào nút đóng gói trong thanh công cụ và chọn Expand Horizontally như hiển thị trong hình dưới đây.
- Các phần tử Button mở rộng theo chiều ngang để lấp đầy bố cục như được hiển thị bên dưới.
- Để hoàn thành bố cục, ràng buộc TextView `show_count` vào phía dưới của nút `button_toast`, cũng như vào hai cạnh bên và cạnh dưới của bố cục, như được hiển thị trong hình động dưới đây.
- Các bước cuối cùng là thay đổi thuộc tính `layout_width` và `layout_height` của TextView `show_count` thành Match Constraints và `textSize` thành 200sp. Bố cục cuối cùng trông giống như hình dưới đây.

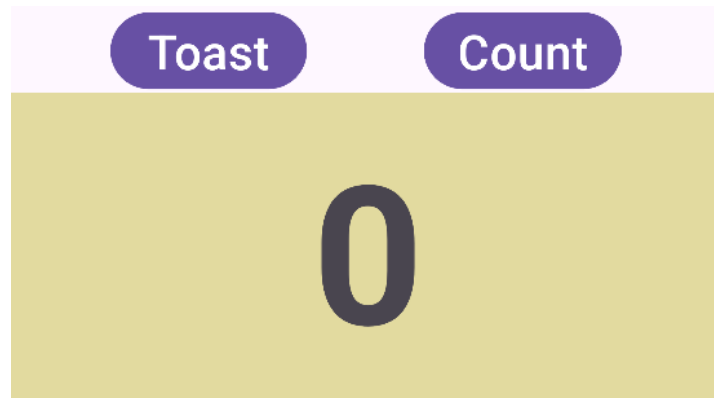


- Nhấp vào nút Orientation in Editor trong thanh công cụ trên cùng và chọn Switch to Landscape. Bố cục của máy tính bảng xuất hiện theo hướng ngang như hiển thị bên dưới. (Bạn có thể chọn Switch to Portrait để quay lại hướng dọc.)



- Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng sau khi chạy ứng dụng để xem nó trông như thế nào trên các loại thiết bị khác nhau. Bạn đã tạo thành công một ứng dụng có thể chạy với giao diện người dùng phù hợp trên điện thoại và máy tính bảng có kích thước màn hình và mật độ khác nhau.

Để phù hợp với hướng ngang (landscape) trên máy tính bảng, bạn có thể căn giữa các phần tử Button trong `activity_main.xml` (xlarge) sao cho chúng xuất hiện như trong hình dưới đây.



## Nhiệm vụ 2: Thay đổi bố cục thành LinearLayout

LinearLayout là một ViewGroup sắp xếp tập hợp các view của nó theo hàng ngang hoặc dọc. LinearLayout là một trong những bố cục phổ biến nhất vì nó đơn giản và nhanh chóng. Nó thường được sử dụng trong một nhóm view khác để sắp xếp các phần tử UI theo chiều ngang hoặc dọc.

LinearLayout yêu cầu có các thuộc tính sau:

- `layout_width`
- `layout_height`
- `orientation`

Giá trị của `layout_width` và `layout_height` có thể là:

- `match_parent`: Mở rộng view để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao. Khi LinearLayout là view gốc, nó mở rộng đến kích thước của màn hình (view cha).
- `wrap_content`: Thu nhỏ kích thước view để vừa đủ bao bọc nội dung của nó. Nếu không có nội dung, view sẽ trở nên vô hình.
- Số dp cố định (density-independent pixels): Chỉ định một kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ, 16dp có nghĩa là 16 pixel độc lập với mật độ.

Thuộc tính `orientation` có thể là:

- `horizontal`: Các view được sắp xếp từ trái sang phải.
- `vertical`: Các view được sắp xếp từ trên xuống dưới.

Trong nhiệm vụ này, bạn sẽ thay đổi nhóm view gốc `ConstraintLayout` của ứng dụng Hello Toast thành `LinearLayout` để thực hành sử dụng `LinearLayout`.

1. Thay đổi nhóm view gốc thành LinearLayout
  - Mở ứng dụng Hello Toast từ nhiệm vụ trước.
  - Mở tệp bố cục activity\_main.xml (nếu nó chưa được mở) và nhấp vào tab Text ở dưới cùng của khung chỉnh sửa để xem mã XML. Ở trên cùng của mã XML có dòng thẻ sau <android.constraintlayout.widget.ConstraintLayout ...
  - Thay đổi thẻ <android.support.constraint.ConstraintLayout> thành <LinearLayout>
  - Đảm bảo rằng thẻ đóng ở cuối mã đã được thay đổi thành </LinearLayout> (Android Studio tự động thay đổi thẻ đóng nếu bạn thay đổi thẻ mở). Nếu nó không thay đổi tự động, hãy thay đổi thủ công.
  - Dưới dòng thẻ <LinearLayout>, thêm thuộc tính sau sau thuộc tính android:layout\_height: android:orientation="vertical"

Sau khi thực hiện các thay đổi này, một số thuộc tính XML của các phần tử khác sẽ được gạch chân màu đỏ vì chúng được sử dụng với ConstraintLayout và không phù hợp với LinearLayout.

## 2. Thay đổi thuộc tính của các phần tử cho LinearLayout

Thực hiện theo các bước sau để thay đổi thuộc tính của các phần tử UI để chúng hoạt động với LinearLayout:

- Mở ứng dụng Hello Toast từ nhiệm vụ trước.
  - Mở tệp bố cục activity\_main.xml (nếu nó chưa được mở) và nhấp vào tab Text.
  - Tìm phần tử Button button\_toast và thay đổi layout\_width="match\_parent"
  - Xóa các thuộc tính sau khỏi phần tử button\_toast: layout\_constraintEnd\_toEndOf, layout\_constraintStart\_toStartOf, layout\_constraintTop\_toTopOf
  - Tìm phần tử Button button\_count và thay đổi thuộc tính layout\_width="match\_parent"
  - Xóa các thuộc tính sau khỏi phần tử button\_count: layout\_constraintEnd\_toEndOf, layout\_constraintStart\_toStartOf, layout\_constraintTop\_toTopOf
  - Tìm phần tử TextView show\_count và thay đổi các thuộc tính sau: layout\_width="match\_parent", layout\_height="wrap\_content"
  - Xóa các thuộc tính sau khỏi phần tử show\_count: layout\_constraintBottom\_toTopOf, layout\_constraintEnd\_toEndOf
  - Nhấp vào tab Preview ở bên phải cửa sổ Android Studio (nếu nó chưa được chọn) để xem bản xem trước của bố cục cho đến thời điểm này.
3. Thay đổi vị trí của các phần tử trong LinearLayout

LinearLayout sắp xếp các phần tử của nó theo hàng ngang hoặc dọc. Bạn đã thêm thuộc tính android:orientation="vertical" cho LinearLayout, vì vậy các phần tử được xếp chồng lên nhau theo chiều dọc như hiển thị trong hình trước đó.

Để thay đổi vị trí của chúng sao cho nút Count ở phía dưới, hãy thực hiện theo các bước sau:

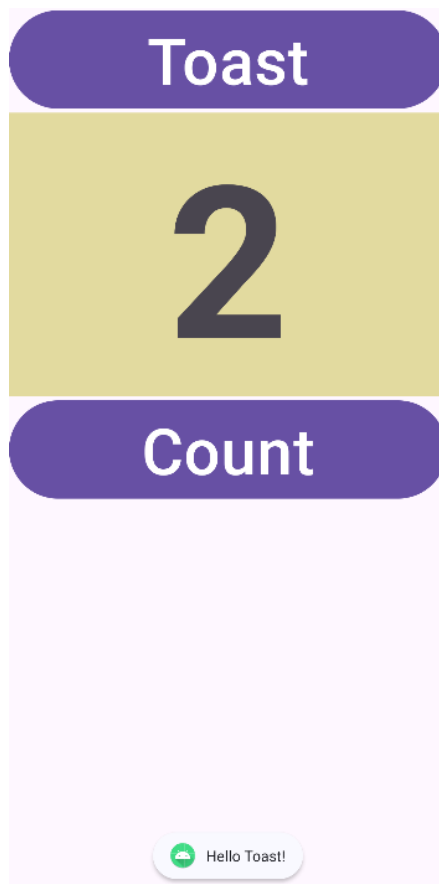
- Mở ứng dụng Hello Toast từ nhiệm vụ trước.
- Mở tệp bố cục activity\_main.xml (nếu nó chưa được mở) và nhấp vào tab Text.
- Chọn nút button\_count và tắt cả các thuộc tính của nó, từ thẻ <Button> cho đến và bao gồm cả thẻ đóng />, sau đó chọn Edit > Cut.
- Nhấp vào sau thẻ đóng /> của phần tử TextView, nhưng trước thẻ đóng </LinearLayout>, sau đó chọn Edit > Paste.
- (Tùy chọn) Để sửa bất kỳ lỗi thụt dòng hoặc khoảng cách nào cho mục đích thẩm mỹ, chọn Code > Reformat Code để định dạng lại mã XML với khoảng cách và thụt dòng phù hợp.

Mã XML của các phân tử UI bây giờ trông như sau:

```
<Button
    android:id="@+id/button_label_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="showToast"
    android:text="Toast"
    android:textColor="@color/white"
    android:textSize="60sp"/>
<TextView
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#E2DA9F"
    android:gravity="center"
    android:text="0"
    android:textAlignment="center"
    android:textSize="200sp"
    android:textStyle="bold" />
<Button
    android:id="@+id/button_label_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="countUp"
    android:text="Count"
    android:textColor="@color/white"
    android:textSize="60sp" />
```

Bằng cách di chuyển nút **button\_count** xuống dưới **TextView**, bố cục bây giờ gần giống với bố cục trước đó, với nút Count ở phía dưới. Bản xem trước của bố cục bây giờ trông như sau:





#### 4. Thêm trọng số (weight) vào phần tử TextView

Việc chỉ định các thuộc tính gravity và weight cung cấp cho bạn quyền kiểm soát bổ sung trong việc sắp xếp các View và nội dung trong một LinearLayout.

Thuộc tính android:gravity xác định cách căn chỉnh nội dung của một View trong chính View đó. Trong bài học trước, bạn đã đặt thuộc tính này cho phần tử TextView show\_count để căn giữa nội dung (số 0) vào giữa TextView:

Thuộc tính android:layout\_weight cho biết bao nhiêu không gian bổ sung trong LinearLayout sẽ được phân bổ cho View. Nếu chỉ có một View có thuộc tính này, nó sẽ nhận tất cả không gian trống còn lại. Đối với nhiều View, không gian sẽ được chia theo tỷ lệ. Ví dụ: nếu các phần tử Button có trọng số là 1 và TextView có trọng số là 2, tổng cộng là 4, thì các Button sẽ nhận  $\frac{1}{4}$  không gian mỗi cái, còn TextView sẽ nhận một nửa.

Trên các thiết bị khác nhau, bố cục có thể hiển thị phần tử TextView show\_count lấp đầy một phần hoặc hầu hết không gian giữa các Button Toast và Count. Để mở rộng TextView để lấp đầy không gian có sẵn bất kể thiết bị nào được sử dụng, hãy chỉ định thuộc tính android:gravity cho TextView. Thực hiện theo các bước sau:

- Mở ứng dụng Hello Toast từ nhiệm vụ trước.
- Mở tệp bố cục activity\_main.xml (nếu nó chưa mở), sau đó nhấp vào tab Text.
- Tìm phần tử TextView show\_count và thêm thuộc tính sau:

Bản xem trước bây giờ sẽ trông như hình minh họa sau.



Phần tử TextView show\_count chiếm toàn bộ không gian giữa các Button. Bạn có thể xem trước bố cục trên các thiết bị khác nhau, như đã làm trong nhiệm vụ trước, bằng cách nhấp vào Device for Preview trên thanh công cụ ở bảng xem trước và chọn một thiết bị khác. Dù chọn thiết bị nào, phần tử TextView show\_count vẫn chiếm toàn bộ không gian giữa các Button.

Mã XML trong activity\_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
android:orientation="vertical"
tools:context=".MainActivity">
<Button
    android:id="@+id/button_label_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="showToast"
    android:text="Toast"
    android:textColor="@color/white"
    android:textSize="60sp"/>
<TextView
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:background="#E2DA9F"
    android:gravity="center"
    android:text="0"
    android:textAlignment="center"
    android:textSize="200sp"
    android:textStyle="bold" />
<Button
    android:id="@+id/button_label_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="countUp"
    android:text="Count"
    android:textColor="@color/white"
    android:textSize="60sp" />
</LinearLayout>
```

### Nhiệm vụ 3: Chuyển bố cục sang RelativeLayout

RelativeLayout là một nhóm View trong đó mỗi View được định vị và căn chỉnh tương đối với các View khác trong nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục với RelativeLayout.

#### 1. Thay đổi LinearLayout thành RelativeLayout

Một cách dễ dàng để thay đổi LinearLayout thành RelativeLayout là thêm các thuộc tính XML trong tab Text.

- Mở tệp activity\_main.xml, sau đó nhấp vào tab Text ở dưới cùng của bảng chỉnh sửa để xem mã XML.

- Thay đổi <LinearLayout ở đầu tệp thành <RelativeLayout
- Cuộn xuống để đảm bảo rằng thẻ đóng </LinearLayout> cũng đã thay đổi thành </RelativeLayout>. Nếu chưa thay đổi tự động, hãy sửa đổi thủ công.

## 2. Sắp xếp lại các View trong RelativeLayout

Một cách dễ dàng để sắp xếp lại và định vị các View trong RelativeLayout là thêm các thuộc tính XML trong tab Text.

- Nhấp vào tab Preview bên cạnh trình chỉnh sửa (nếu chưa được chọn) để xem bản xem trước bố cục, bây giờ trông như hình minh họa bên dưới.
- Khi chuyển sang RelativeLayout, trình chỉnh sửa bố cục cũng thay đổi một số thuộc tính của View. Ví dụ:
  - Button Count (button\_count) đang chồng lên Button Toast, khiến bạn không thể nhìn thấy Button Toast.
  - Phần trên của TextView show\_count chồng lên các phần tử Button.



- Thêm thuộc tính android:layout\_below vào Button button\_count để đặt Button này ngay bên dưới TextView show\_count. Thuộc tính này là một trong số các thuộc tính giúp định vị View trong RelativeLayout—bạn đặt View dựa vào các View khác.
- Thêm thuộc tính android:layout\_centerHorizontal vào Button button\_count để căn giữa View theo chiều ngang trong View cha của nó, trong trường hợp này là RelativeLayout.

```
<Button
    android:id="@+id/button_label_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/show_count"
    android:layout_centerHorizontal="true"
```

```
android:onClick="countUp"  
android:text="Count"  
android:textColor="@color/white"  
android:textSize="60sp" />
```

- Thêm các thuộc tính sau vào TextView show\_count: android:layout\_below, android:layout\_alignParentLeft, android:layout\_alignParentStart
  - Thuộc tính android:layout\_alignParentLeft căn chỉnh View về phía trái của RelativeLayout (View cha).
  - Mặc dù thuộc tính này đủ để căn chỉnh View sang bên trái, nhưng bạn có thể muốn căn chỉnh View theo cạnh phải nếu ứng dụng đang chạy trên một thiết bị sử dụng ngôn ngữ từ phải sang trái (RTL). Vì vậy, thuộc tính android:layout\_alignParentStart đảm bảo rằng cạnh “bắt đầu” của View này khớp với cạnh bắt đầu của View cha. Cạnh bắt đầu là cạnh trái màn hình nếu ngôn ngữ là từ trái sang phải (LTR) hoặc là cạnh phải nếu ngôn ngữ là từ phải sang trái (RTL).

```
<TextView  
    android:id="@+id/show_count"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="#E2DA9F"  
    android:gravity="center"  
    android:text="0"  
    android:textAlignment="center"  
    android:textSize="200sp"  
    android:textStyle="bold"  
    android:layout_below="@+id/button_label_toast"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true"/>
```

- Xóa thuộc tính android:layout\_weight="1" khỏi TextView show\_count, vì nó không phù hợp với RelativeLayout. Bản xem trước bố cục bây giờ trông như hình minh họa sau:



Mã XML trong activity\_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button_label_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="showToast"
        android:text="Toast"
        android:textColor="@color/white"
        android:textSize="60sp" />

    <TextView
        android:id="@+id/show_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#E2DA9F"
        android:gravity="center"
        android:text="0"
        android:textAlignment="center"
        android:textSize="200sp"
```

```

        android:textStyle="bold"
        android:layout_below="@+id/button_label_toast"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>

<Button
    android:id="@+id/button_label_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/show_count"
    android:layout_centerHorizontal="true"
    android:onClick="countUp"
    android:text="Count"
    android:textColor="@color/white"
    android:textSize="60sp" />
</RelativeLayout>

```

## Tổng kết

Sử dụng trình chỉnh sửa bố cục để xem trước và tạo các biến thể:

- Để xem trước bố cục ứng dụng với hướng ngang trong trình chỉnh sửa bố cục, nhấp vào nút Orientation for Preview trên thanh công cụ phía trên và chọn Switch to Landscape. Chọn Switch to Portrait để quay lại hướng dọc.
- Để tạo một biến thể của bố cục khác cho hướng ngang, nhấp vào nút Orientation for Preview và chọn Create Landscape Variation. Một cửa sổ chỉnh sửa mới sẽ mở ra với tab land/activity\_main.xml hiển thị bố cục cho hướng ngang.
- Để xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị hoặc trình giả lập, nhấp vào nút Device in Editor trên thanh công cụ phía trên và chọn một thiết bị.
- Để tạo một biến thể của bố cục khác dành cho máy tính bảng (màn hình lớn hơn), nhấp vào nút Orientation for Preview và chọn Create layout x-large Variation. Một cửa sổ chỉnh sửa mới sẽ mở ra với tab xlarge/activity\_main.xml hiển thị bố cục dành cho thiết bị có kích thước máy tính bảng.

Sử dụng ConstraintLayout:

- Để xóa tất cả các ràng buộc trong một bố cục có gốc là ConstraintLayout, nhấp vào nút Clear All Constraints trên thanh công cụ.
- Bạn có thể căn chỉnh một phần tử giao diện người dùng có chứa văn bản, chẳng hạn như TextView hoặc Button, với một phần tử giao diện khác có chứa văn bản. Một baseline constraint cho phép bạn ràng buộc các phần tử để dòng cơ sở của văn bản khớp nhau.

- Để tạo baseline constraint, di chuột qua phần tử giao diện người dùng cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.
- Nút pack trên thanh công cụ cung cấp các tùy chọn để đóng gói hoặc mở rộng các phần tử giao diện người dùng đã chọn. Bạn có thể sử dụng nó để sắp xếp đều các phần tử Button theo chiều ngang trong bố cục.

#### Sử dụng LinearLayout:

- LinearLayout là một ViewGroup sắp xếp tập hợp các View của nó theo một hàng ngang hoặc dọc.
- Một LinearLayout bắt buộc phải có các thuộc tính layout\_width, layout\_height và orientation.
- match\_parent cho layout\_width hoặc layout\_height: Mở rộng View để lấp đầy parent theo chiều rộng hoặc chiều cao. Khi LinearLayout là root View, nó sẽ mở rộng theo kích thước màn hình (parent View).
- wrap\_content cho layout\_width hoặc layout\_height: Thu nhỏ kích thước để View vừa đủ bao bọc nội dung của nó. Nếu không có nội dung, View sẽ trở nên vô hình.
- Một số cố định dp (density-independent pixels) cho layout\_width hoặc layout\_height: Chỉ định một kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ, 16dp nghĩa là 16 điểm ảnh độc lập với mật độ.
- orientation của LinearLayout có thể là horizontal để sắp xếp các phần tử từ trái sang phải hoặc vertical để sắp xếp từ trên xuống dưới.
- Việc chỉ định các thuộc tính gravity và weight cung cấp thêm quyền kiểm soát đối với việc sắp xếp các View và nội dung trong LinearLayout.
- Thuộc tính android:gravity chỉ định căn chỉnh nội dung của một View bên trong chính nó.
- Thuộc tính android:layout\_weight chỉ ra lượng không gian thừa trong LinearLayout sẽ được phân bổ cho View. Nếu chỉ có một View có thuộc tính này, nó sẽ nhận toàn bộ không gian thừa. Nếu có nhiều View, không gian sẽ được chia theo tỷ lệ. Ví dụ, nếu hai Button có trọng số 1 và một TextView có trọng số 2, tổng cộng là 4, thì mỗi Button sẽ nhận  $\frac{1}{4}$  không gian, còn TextView nhận một nửa.

#### Sử dụng RelativeLayout:

- RelativeLayout là một ViewGroup trong đó mỗi View được định vị và căn chỉnh tương đối với các View khác trong nhóm.
- Sử dụng android:layout\_alignParentTop để căn chỉnh View lên trên cùng của parent.
- Sử dụng android:layout\_alignParentLeft để căn chỉnh View về phía bên trái của parent.
- Sử dụng android:layout\_alignParentStart để cạnh bắt đầu của View trùng với cạnh bắt đầu của parent. Thuộc tính này hữu ích nếu bạn muốn ứng dụng của mình hoạt



động trên các thiết bị sử dụng các ngôn ngữ hoặc tùy chọn ngôn ngữ khác nhau. Start là cạnh bên trái của màn hình nếu ngôn ngữ là từ trái sang phải hoặc là cạnh bên phải nếu ngôn ngữ là từ phải sang trái.

## Thay đổi một ứng dụng

Mở ứng dụng HelloToast.

- Đổi tên dự án thành HelloConstraint, và đổi tên các thành phần trong dự án thành Hello Constraint. (Để biết cách sao chép và đổi tên một dự án, hãy xem Phụ lục: Tiện ích).
- Chỉnh sửa bố cục trong `activity_main.xml` để căn chỉnh các nút Toast và Count dọc theo phía bên trái của TextView hiển thị số "0". Tham khảo các hình minh họa bên dưới để sắp xếp bố cục.
- Thêm một nút thứ ba có tên là Zero, đặt vị trí giữa các nút Toast và Count.
- Phân bố các nút theo chiều dọc, nằm giữa phần trên và phần dưới của TextView hiển thị số.
- Đặt màu nền ban đầu của nút Zero là màu xám.
- Đảm bảo rằng nút Zero cũng xuất hiện trong bố cục cho chế độ ngang (`activity_main.xml (land)`) và trong bố cục dành cho thiết bị có màn hình lớn (`activity_main.xml (xlarge)`).
- Khi nhấn vào nút Zero, giá trị hiển thị trong TextView phải trở về 0.
- Cập nhật trình xử lý sự kiện khi nhấn vào nút Count sao cho màu nền của nó thay đổi tùy thuộc vào giá trị lẻ hoặc chẵn của số đếm.
- Bạn có thể sử dụng các hằng số trong lớp Color để đặt hai màu nền khác nhau.
- Cập nhật trình xử lý sự kiện của nút Count để thay đổi màu nền của nút Zero (từ màu xám sang một màu khác) khi số đếm thay đổi, cho thấy nút này đã được kích hoạt.
- Cập nhật trình xử lý sự kiện của nút Zero để đặt lại màu nền của nó về màu xám khi giá trị trong TextView trở về 0.



## 1.4) Văn bản và các chế độ cuộn

### Giới thiệu

Lớp `TextView` là một lớp con của lớp `View`, hiển thị văn bản trên màn hình. Bạn có thể kiểm soát cách văn bản hiển thị bằng các thuộc tính của `TextView` trong tệp bố cục XML. Bài thực hành này hướng dẫn cách làm việc với nhiều phần tử `TextView`, bao gồm cả một phần tử mà người dùng có thể cuộn nội dung theo chiều dọc.

Nếu bạn có nhiều thông tin hơn mức có thể hiển thị trên màn hình thiết bị, bạn có thể tạo một chế độ xem cuộn để người dùng có thể cuộn theo chiều dọc bằng cách vuốt lên hoặc xuống, hoặc theo chiều ngang bằng cách vuốt sang phải hoặc trái.

Bạn thường sử dụng chế độ xem cuộn cho các bài báo, tin tức hoặc bất kỳ đoạn văn bản dài nào không thể hiển thị đầy đủ trên màn hình. Bạn cũng có thể sử dụng chế độ xem cuộn để cho phép người dùng nhập nhiều dòng văn bản hoặc kết hợp các phần tử giao diện người dùng (chẳng hạn như trường văn bản và nút) trong một chế độ xem cuộn.

Lớp `ScrollView` cung cấp bố cục cho chế độ xem cuộn. `ScrollView` là một lớp con của `FrameLayout`. Chỉ đặt một phần tử con trong đó—một phần tử con chứa toàn bộ nội dung cần cuộn. Phần tử con này có thể là một `ViewGroup` (chẳng hạn như `LinearLayout`) chứa các phần tử giao diện người dùng.

Các bố cục phức tạp có thể gặp vấn đề về hiệu suất với các phần tử con như hình ảnh. Một lựa chọn tốt cho một `View` bên trong `ScrollView` là `LinearLayout` được sắp xếp theo chiều dọc, trình bày các mục mà người dùng có thể cuộn qua (chẳng hạn như các phần tử `TextView`).

Với `ScrollView`, tất cả các phần tử giao diện người dùng đều được lưu trong bộ nhớ và trong hệ thống phân cấp chế độ xem ngay cả khi chúng không được hiển thị trên màn hình. Điều này làm cho `ScrollView` trở thành lựa chọn lý tưởng để cuộn các trang văn bản tự do một cách mượt mà, vì văn bản đã có sẵn trong bộ nhớ. Tuy nhiên, `ScrollView` có thể tiêu tốn nhiều bộ nhớ, điều này có thể ảnh hưởng đến hiệu suất của ứng dụng. Để hiển thị danh sách dài các mục mà người dùng có thể thêm, xóa hoặc chỉnh sửa, hãy cân nhắc sử dụng `RecyclerView`, được mô tả trong một bài học riêng.

### Những gì bạn cần biết trước

- Bạn cần có khả năng:
- Tạo ứng dụng Hello World với Android Studio.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị.
- Triển khai một `TextView` trong bố cục của ứng dụng.
- Tạo và sử dụng tài nguyên chuỗi.

## Những gì bạn sẽ học

- Cách sử dụng mã XML để thêm nhiều phần tử TextView.
- Cách sử dụng mã XML để xác định chế độ xem cuộn.
- Cách hiển thị văn bản tự do với một số thẻ định dạng HTML.
- Cách tạo kiểu nền và màu văn bản của TextView.
- Cách thêm liên kết web vào văn bản.

## Những gì bạn sẽ làm

- Tạo ứng dụng ScrollingText.
- Thay đổi ViewGroup ConstraintLayout thành RelativeLayout.
- Thêm hai phần tử TextView cho tiêu đề bài viết và tiêu đề phụ.
- Sử dụng các kiểu TextAppearance và màu sắc cho tiêu đề và tiêu đề phụ của bài viết.
- Sử dụng thẻ HTML trong chuỗi văn bản để kiểm soát định dạng.
- Sử dụng thuộc tính lineSpacingExtra để thêm khoảng cách dòng giúp dễ đọc hơn.
- Thêm ScrollView vào bố cục để bật tính năng cuộn cho phần tử TextView.
- Thêm thuộc tính autoLink để làm cho các URL trong văn bản có thể nhấp và hoạt động.

## Nhiệm vụ 1: Thêm và chỉnh sửa các phần tử TextView

Trong bài thực hành này, bạn sẽ tạo một dự án Android cho ứng dụng ScrollingText, thêm các phần tử TextView vào bố cục cho tiêu đề bài viết và tiêu đề phụ, và thay đổi phần tử TextView hiện có chứa văn bản "Hello World" thành một bài viết dài. Hình bên dưới là sơ đồ của bố cục. Bạn sẽ thực hiện tất cả những thay đổi này trong mã XML và tệp strings.xml.

Bạn sẽ chỉnh sửa mã XML cho bố cục trong bảng văn bản (Text pane), có thể hiển thị bằng cách nhấp vào tab Text, thay vì nhấp vào tab Design để hiển thị bảng thiết kế (Design pane). Một số thay đổi đối với các phần tử và thuộc tính giao diện người dùng sẽ dễ thực hiện hơn trực tiếp trong bảng văn bản bằng cách sử dụng mã nguồn XML.

### 1. Tạo dự án và các phần tử TextView

Trong nhiệm vụ này, bạn sẽ tạo dự án và các phần tử TextView, đồng thời sử dụng các thuộc tính của TextView để tạo kiểu cho văn bản và nền.

- Trong Android Studio, tạo một dự án mới với các tham số sau:

Thuộc tính	Giá trị
------------	---------

Name	Scrolling Text
Package Name	com.example.scrollingtext
Minimum SDK	API 24 (“Nougat”; Android 7.0)
Template	Empty Views Activity

- Trong thư mục app > res > layout ở khung Project > Android, mở tệp activity\_main.xml và nhấp vào tab Text để xem mã XML. Ở trên cùng, hoặc góc của hệ thống phân cấp View, là ConstraintLayout ViewGroup:
- Thay đổi ViewGroup này thành RelativeLayout. Dòng mã thứ hai bây giờ sẽ trông giống như sau:
- Xóa dòng mã XML sau, liên quan đến ConstraintLayout
- Thêm một phần tử TextView phía trên TextView “Hello World” bằng cách nhập <TextView. Một khối TextView sẽ xuất hiện kết thúc bằng /> và hiển thị các thuộc tính layout\_width và layout\_height, đây là các thuộc tính bắt buộc đối với TextView.
- Nhập các thuộc tính sau cho TextView. Khi bạn nhập từng thuộc tính và giá trị, các gợi ý sẽ xuất hiện để hoàn thành tên thuộc tính hoặc giá trị.

```
<TextView
    android:id="@+id/article_heading"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#3F51B5"
    android:padding="10dp"
    android:text="Article Title"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
    android:textColor="@color/white"
    android:textStyle="bold" />
```

- Trích xuất tài nguyên chuỗi cho thuộc tính android:text có chuỗi mã cứng "Article Title" trong TextView để tạo một mục cho nó trong strings.xml. Đặt con trỏ vào chuỗi mã cứng, nhấn Alt-Enter (Option-Enter trên Mac) và chọn Extract string resource. Đảm bảo rằng tùy chọn Create the resource in directories được chọn, sau đó chỉnh sửa tên tài nguyên cho giá trị chuỗi thành article\_title. Tài nguyên chuỗi được mô tả chi tiết trong String Resources.
- Trích xuất tài nguyên kích thước cho thuộc tính android:padding có chuỗi mã cứng "10dp" trong TextView để tạo dimens.xml và thêm một mục vào đó.

Đặt con trỏ vào chuỗi mã cứng, nhấn Alt-Enter (Option-Enter trên Mac) và chọn Extract dimension resource. Đảm bảo rằng tùy chọn Create the resource in directories được chọn, sau đó chỉnh sửa tên tài nguyên thành padding\_regular.

- Thêm một phần tử TextView khác phía trên TextView “Hello World” và bên dưới TextView bạn đã tạo trong các bước trước. Thêm các thuộc tính sau vào TextView:

```
<TextView
    android:id="@+id/article_subheading"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_heading"
    android:padding="@dimen/padding_regular"
    android:text="Article Subtitle"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault" />
```

Vì bạn đã trích xuất tài nguyên kích thước cho chuỗi "10dp" thành padding\_regular trong TextView đã tạo trước đó, bạn có thể sử dụng @dimen/padding\_regular cho thuộc tính android:padding trong TextView này.

- Trích xuất tài nguyên chuỗi cho thuộc tính android:text có chuỗi mã cứng "Article Subtitle" trong TextView thành article\_subtitle.
- Trong phần tử TextView “Hello World”, xóa các thuộc tính layout\_constraint
- Thêm các thuộc tính TextView sau vào phần tử TextView “Hello World” và thay đổi thuộc tính android:text:

```
<TextView
    android:id="@+id/article"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_subheading"
    android:lineSpacingExtra="5sp"
    android:padding="@dimen/padding_regular"
    android:text="Article text" />
```

- Trích xuất tài nguyên chuỗi cho "Article text" thành article\_text, và trích xuất tài nguyên kích thước cho "5sp" thành line\_spacing.
- Định dạng lại và căn chỉnh mã bằng cách chọn Code > Reformat Code. Việc định dạng và căn chỉnh mã là một thói quen tốt để giúp bạn và những người khác dễ dàng hiểu mã hơn.

## 2. Thêm văn bản của bài báo

Trong một ứng dụng thực tế truy cập vào các bài báo tạp chí hoặc báo, các bài báo xuất hiện có thể đến từ một nguồn trực tuyến thông qua một content provider hoặc có thể được lưu sẵn trong cơ sở dữ liệu trên thiết bị.

Trong bài thực hành này, bạn sẽ tạo bài báo dưới dạng một chuỗi dài duy nhất trong tài nguyên strings.xml.

- Trong thư mục app > res > values, mở strings.xml.
- Mở bất kỳ tệp văn bản nào có một lượng lớn nội dung hoặc mở tệp strings.xml của ứng dụng ScrollingText đã hoàn thành.
- Nhập các giá trị cho các chuỗi article\_title và article\_subtitle với tiêu đề và phụ đề tự tạo, hoặc sử dụng các giá trị trong tệp strings.xml của ứng dụng ScrollingText đã hoàn thành. Hãy đảm bảo rằng giá trị chuỗi là văn bản một dòng, không chứa thẻ HTML hoặc nhiều dòng.
- Nhập hoặc sao chép và dán nội dung cho chuỗi article\_text.

Bạn có thể sử dụng văn bản trong tệp của mình hoặc sử dụng văn bản được cung cấp cho chuỗi article\_text trong tệp strings.xml của ứng dụng ScrollingText đã hoàn thành. Yêu cầu duy nhất của nhiệm vụ này là văn bản phải đủ dài để không hiển thị hết trên màn hình. Lưu ý các điểm sau:

- Khi nhập hoặc dán văn bản vào strings.xml, các dòng văn bản sẽ không tự động xuống dòng mà sẽ kéo dài ra ngoài lề phải. Đây là hành vi đúng—mỗi dòng văn bản mới bắt đầu từ lề trái thể hiện một đoạn văn hoàn chỉnh. Nếu muốn văn bản trong strings.xml được xuống dòng, bạn có thể nhấn Return để nhập dấu ngắt dòng cứng hoặc định dạng văn bản trước trong trình chỉnh sửa văn bản có hỗ trợ ngắt dòng cứng.
- Nhập \n để đại diện cho ký tự xuống dòng và nhập thêm một \n nữa để tạo dòng trống. Bạn cần thêm các ký tự kết thúc dòng để tránh các đoạn văn bị nối liền nhau.
- Nếu có dấu nháy đơn (') trong văn bản, bạn phải thêm dấu gạch chéo ngược (\) trước nó để tránh lỗi ('). Nếu có dấu nháy kép ("), bạn cũng phải thêm gạch chéo ngược (\"). Bạn cũng phải thoát bất kỳ ký tự không thuộc ASCII nào. Xem phần Định dạng và kiểu dáng trong tài nguyên chuỗi để biết thêm chi tiết.
- Sử dụng thẻ HTML <b> và </b> để bôi đậm từ ngữ.
- Sử dụng thẻ HTML <i> và </i> để in nghiêng từ ngữ. Nếu bạn sử dụng dấu nháy đơn cong (') trong một cụm từ in nghiêng, hãy thay thế nó bằng dấu nháy đơn thẳng (').
- Bạn có thể kết hợp in đậm và in nghiêng bằng cách lồng các thẻ, ví dụ: <b><i>... từ ngữ ...</i></b>. Các thẻ HTML khác sẽ bị bỏ qua.
- Bao toàn bộ nội dung văn bản trong <string name="article\_text"> </string> trong tệp strings.xml.

- Bao gồm một liên kết web để kiểm tra, chẳng hạn như [www.google.com](http://www.google.com) (ví dụ minh họa sử dụng [www.rockument.com](http://www.rockument.com)). Không sử dụng thẻ HTML, vì bất kỳ thẻ HTML nào ngoài thẻ bold và italic đều bị bỏ qua và sẽ hiển thị dưới dạng văn bản thông thường, không phải liên kết.

### 3. Chạy ứng dụng

Chạy ứng dụng. Bài báo sẽ xuất hiện, nhưng người dùng không thể cuộn bài báo vì bạn chưa thêm ScrollView (bạn sẽ làm điều đó trong nhiệm vụ tiếp theo). Lưu ý rằng khi nhấn vào liên kết web, nó vẫn chưa hoạt động. Bạn cũng sẽ sửa lỗi này trong nhiệm vụ tiếp theo.

Tập `activity_main.xml` trông giống như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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/article_heading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3F51B5"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_title"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
        android:textColor="@color/white"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/article_subheading"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/article_heading"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_subtitle"
        android:textSize="30dp"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large" />
    <TextView
        android:id="@+id/article"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/article_subheading"
        android:lineSpacingExtra="@dimen/line_spacing"
```

```
android:textAppearance="@android:style/TextAppearance.DeviceDefault"
android:padding="@dimen/padding_regular"
android:textSize="16dp"
android:text="@string/article_text" />
</RelativeLayout>
```

## Nhiệm vụ 2: Thêm ScrollView và liên kết web có thể nhấp

Trong nhiệm vụ trước, bạn đã tạo ứng dụng ScrollingText với các phần tử TextView cho tiêu đề bài viết, phụ đề và nội dung bài viết dài. Bạn cũng đã thêm một liên kết web, nhưng liên kết này chưa hoạt động. Bạn sẽ thêm mã để kích hoạt liên kết.

Ngoài ra, TextView không thể tự cuộn nội dung để hiển thị toàn bộ bài viết. Bạn sẽ thêm một ViewGroup mới có tên là ScrollView vào bố cục XML để làm cho TextView có thể cuộn được.

### 1. Thêm thuộc tính autoLink để kích hoạt liên kết web

Thêm thuộc tính `android:autoLink="web"` vào TextView của bài viết. Mã XML cho TextView này bây giờ sẽ trông như sau:

```
<TextView
    android:id="@+id/article"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_subheading"
    android:lineSpacingExtra="@dimen/line_spacing"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault"
    android:padding="@dimen/padding_regular"
    android:autoLink="web"
    android:textSize="16dp"
    android:text="@string/article_text" />
```

### 2. Thêm ScrollView vào bố cục

Để làm cho một View (chẳng hạn như TextView) có thể cuộn, hãy đặt View đó bên trong một ScrollView.

- Thêm một ScrollView giữa TextView của phụ đề bài viết (`article_subheading`) và TextView của bài viết (`article TextView`). Khi bạn nhập `<ScrollView`, Android Studio sẽ tự động thêm `</ScrollView>` ở cuối và đề xuất các thuộc tính `android:layout_width` và `android:layout_height`.
- Chọn `wrap_content` từ danh sách đề xuất cho cả hai thuộc tính.



- Di chuyển thẻ đóng `</ScrollView>` xuống sau `TextView` của bài viết để đảm bảo rằng tất cả các thuộc tính của `TextView` này nằm hoàn toàn bên trong `ScrollView`.
- Xóa thuộc tính sau khỏi `TextView` của bài viết và thêm nó vào `ScrollView`:  
Với thuộc tính trên, phần tử `ScrollView` sẽ xuất hiện bên dưới phụ đề bài viết. Bài viết sẽ nằm bên trong phần tử `ScrollView`.
- Chọn `Code > Reformat Code` để định dạng lại mã XML sao cho `TextView` của bài viết xuất hiện thụt vào bên trong `<ScrollView>`.
- Nhấp vào tab `Preview` ở bên phải trình chỉnh sửa bố cục để xem trước bố cục.

### 3. Chạy ứng dụng

Để kiểm tra cách văn bản cuộn:

- Chạy ứng dụng trên thiết bị hoặc trình giả lập. Vuốt lên và xuống để cuộn bài viết. Thanh cuộn xuất hiện ở lề phải khi bạn cuộn. Nhấn vào liên kết web để truy cập trang web. Thuộc tính `android:autoLink` sẽ biến bất kỳ URL nào nhận diện được trong `TextView` (chẳng hạn như `www.rockument.com`) thành một liên kết web có thể nhấp.
- Xoay thiết bị hoặc trình giả lập trong khi chạy ứng dụng. Lưu ý rằng chế độ xem cuộn sẽ mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn được.
- Chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng. Lưu ý rằng chế độ xem cuộn sẽ mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn được.

Trong hình minh họa ở trên, có các thành phần sau:

## Article Title

### The Dead Sea Scrolls

opening on the side of a cliff and was surprised to hear a shattering sound. He and his companions later entered the cave and stumbled across a collection of large clay jars, seven of which contained scrolls with writing on them. The teenagers took the seven scrolls to a nearby town where they were sold for a small sum to a local antiquities dealer. Word of the find spread, and Bedouins and archaeologists eventually unearthed tens of thousands of additional scroll fragments from 10 nearby caves; together they make up between 800 and 900 manuscripts. It soon became clear that this was one of the greatest archaeological discoveries ever made. The origin of the **Dead Sea Scrolls**, which were written around 2,000 years ago between 150 BCE and 70 CE, is still the subject of scholarly debate even today. <https://www.google.com/>

According to the prevailing theory, they are the work of a population that inhabited the area until Roman troops destroyed the settlement around 70 CE. The area was known as Judea at that time, and the people are thought to have belonged to a group called the Essenes, a devout Jewish sect. The majority of the texts on the Dead Sea Scrolls are in Hebrew, with some fragments written in an ancient version of its alphabet thought to have fallen out of use in the fifth century BCE. But there are other languages as well.

Some scrolls are in *Aramaic*, the language spoken by many inhabitants of the region from the sixth century BCE to the siege of Jerusalem in 70 CE. In addition, several texts feature translations of the Hebrew Bible into Greek.

- Một liên kết web có thể nhấp được nhúng trong văn bản tự do
- Thanh cuộn xuất hiện khi cuộn văn bản

Mã XML cho bố cục với ScrollView như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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/article_heading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3F51B5"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_title"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large"
```

```

        android:textColor="@color/white"
        android:textStyle="bold" />
<TextView
    android:id="@+id/article_subheading"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/article_heading"
    android:padding="@dimen/padding_regular"
    android:text="@string/article_subtitle"
    android:textSize="30dp"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large" />
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@id/article_subheading">
    <TextView
        android:id="@+id/article"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault"
        android:padding="@dimen/padding_regular"
        android:autoLink="web"
        android:textSize="16dp"
        android:text="@string/article_text" />
    </ScrollView>
</RelativeLayout>

```

### Nhiệm vụ 3: Cuộn nhiều phần tử

Như đã đề cập trước đó, một ScrollView chỉ có thể chứa một View con (chẳng hạn như TextView của bài viết mà bạn đã tạo). Tuy nhiên, View đó có thể là một ViewGroup chứa các phần tử View khác, chẳng hạn như LinearLayout. Bạn có thể lồng một ViewGroup như LinearLayout vào bên trong ScrollView, từ đó cuộn tất cả những gì nằm bên trong LinearLayout.

Ví dụ: nếu bạn muốn phần phụ đề của bài viết cũng cuộn cùng với bài viết, hãy thêm một LinearLayout vào trong ScrollView và di chuyển cả phụ đề và bài viết vào trong LinearLayout. LinearLayout trở thành View con duy nhất trong ScrollView như trong hình minh họa bên dưới, và người dùng có thể cuộn toàn bộ LinearLayout: bao gồm cả phụ đề và bài viết.

#### 1. Thêm LinearLayout vào ScrollView

- Mở tệp activity\_main.xml của dự án ScrollingText và chọn tab Text để chỉnh sửa mã XML (nếu chưa được chọn).
- Thêm một LinearLayout phía trên TextView của bài viết bên trong ScrollView. Khi bạn nhập <LinearLayout, Android Studio sẽ tự động thêm </LinearLayout> vào cuối và hiển thị các gợi ý cho thuộc tính android:layout\_width và android:layout\_height. Chọn match\_parent và wrap\_content từ danh sách gợi ý tương ứng cho chiều rộng và chiều cao. Mã XML ở phần đầu của ScrollView bây giờ sẽ trông như sau:

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@id/article_subheading">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </LinearLayout>
</ScrollView>
```

Bạn sử dụng match\_parent để chiều rộng khớp với ViewGroup cha. Bạn sử dụng wrap\_content để điều chỉnh kích thước LinearLayout sao cho vừa đủ bao bọc nội dung bên trong.

- Di chuyển thẻ đóng </LinearLayout> xuống sau TextView của bài viết nhưng trước thẻ đóng </ScrollView>.
- LinearLayout bây giờ bao gồm TextView của bài viết và hoàn toàn nằm bên trong ScrollView.
- Thêm thuộc tính android:orientation="vertical" vào LinearLayout để đặt hướng của nó thành dọc.
  - Chọn Code > Reformat Code để căn chỉnh mã XML đúng cách.

LinearLayout bây giờ sẽ trông như sau:

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@id/article_subheading">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:id="@+id/article"
            android:layout_width="wrap_content"
```

```

        android:layout_height="wrap_content"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault"
        android:padding="@dimen/padding_regular"
        android:autoLink="web"
        android:textSize="16dp"
        android:text="@string/article_text" />
    </LinearLayout>
</ScrollView>

```

## 2. Di chuyển các phần tử giao diện người dùng vào bên trong LinearLayout

Hiện tại, LinearLayout chỉ có một phần tử giao diện người dùng—TextView của bài viết. Bạn cần thêm TextView của phụ đề bài viết (article\_subheading) vào trong LinearLayout để cả hai có thể cuộn cùng nhau.

- Để di chuyển TextView của phụ đề bài viết (article\_subheading), chọn đoạn mã, chọn Edit > Cut, nhấp vào phía trên TextView của bài viết bên trong LinearLayout, sau đó chọn Edit > Paste.
- Xóa thuộc tính android:layout\_below="@id/article\_heading" khỏi TextView của phụ đề bài viết. Vì TextView này bây giờ nằm trong LinearLayout, thuộc tính này sẽ gây xung đột với các thuộc tính của LinearLayout.
- Thay đổi thuộc tính android:layout\_below="@id/article\_subheading" của ScrollView thành android:layout\_below="@id/article\_heading". Vì phụ đề bây giờ là một phần của LinearLayout, ScrollView phải được đặt bên dưới tiêu đề, không phải bên dưới phụ đề. Mã XML của ScrollView bây giờ sẽ như sau:

```

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@id/article_heading">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:id="@+id/article_subheading"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="@dimen/padding_regular"
            android:text="@string/article_subtitle"
            android:textSize="30dp"
            android:textAppearance="@android:style/TextAppearance.DeviceDefault.Large" />
        <TextView
            android:id="@+id/article"

```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:textAppearance="@android:style/TextAppearance.DeviceDefault"
        android:padding="@dimen/padding_regular"
        android:autoLink="web"
        android:textSize="16dp"
        android:text="@string/article_text" />
    </LinearLayout>
</ScrollView>
```

- Chạy ứng dụng. Vuốt lên và xuống để cuộn bài viết, và nhận thấy rằng phụ đề bây giờ cũng cuộn cùng với bài viết trong khi tiêu đề vẫn cố định.

## Tóm tắt

- Sử dụng ScrollView để cuộn một View con duy nhất (chẳng hạn như TextView). Một ScrollView chỉ có thể chứa một View con hoặc một ViewGroup.
- Sử dụng một ViewGroup như LinearLayout làm View con trong ScrollView để cuộn nhiều phần tử View. Đặt tất cả các phần tử bên trong LinearLayout.
- Hiển thị văn bản tự do trong TextView với các thẻ định dạng HTML cho chữ in đậm và in nghiêng.
- Sử dụng `\n` như một ký tự xuống dòng trong văn bản tự do để ngăn đoạn văn này nối liền với đoạn văn tiếp theo.
- Sử dụng thuộc tính `android:autoLink="web"` để làm cho các liên kết web trong văn bản có thể nhấp được.

## Thay đổi ứng dụng

Mở ứng dụng ScrollingText2 mà bạn đã tạo trong bài học Làm việc với các phần tử TextView.

- Thay đổi phần phụ đề sao cho nó hiển thị trong một cột bên trái có chiều rộng 100 dp, như hình minh họa bên dưới.
- Đặt văn bản của bài viết ở bên phải phần phụ đề, như hình minh họa bên dưới.

Article Title	
The Dead Sea Scrolls	<p>In late 1946 or early 1947, three Bedouin teenagers were tending their goats and sheep near the ancient settlement of Qumran, located on the northwest shore of the Dead Sea in what is now known as the <b>West Bank</b>.</p> <p>One of these young shepherds tossed a rock into an opening on the side of a cliff and was surprised to hear a shattering sound. He and his companions later entered the cave and stumbled across a collection of large clay jars, seven of which contained scrolls with writing on them. The teenagers took the seven scrolls to a nearby town where they were sold for a small sum to a local antiquities dealer. Word of the find spread, and Bedouins and archaeologists eventually unearthed tens of thousands of additional scroll fragments from 10 nearby caves; together they make up between 800 and 900 manuscripts.</p>

## 1.5) Tài nguyên có sẵn

### Giới thiệu

### Những gì bạn nên biết trước

Bạn nên có khả năng:

- Hiểu quy trình làm việc cơ bản của Android Studio.
- Tạo một ứng dụng từ đầu bằng mẫu Empty Views Activity.
- Sử dụng trình chỉnh sửa bố cục (layout editor).

### Những gì bạn sẽ học

- Nơi tìm thông tin và tài nguyên dành cho nhà phát triển.
- Cách thêm biểu tượng khởi chạy (launcher icon) vào ứng dụng của bạn.
- Cách tìm kiếm sự trợ giúp khi phát triển ứng dụng Android.

### Những gì bạn sẽ làm

- Khám phá một số tài nguyên hữu ích dành cho các nhà phát triển Android ở mọi cấp độ.
- Thêm biểu tượng khởi chạy cho ứng dụng của bạn.

## **Nhiệm vụ 1: Thay đổi biểu tượng khởi chạy**

Mỗi ứng dụng mới bạn tạo với Android Studio sẽ bắt đầu với một biểu tượng khởi chạy tiêu chuẩn đại diện cho ứng dụng. Biểu tượng khởi chạy xuất hiện trong danh sách của cửa hàng Google Play. Khi người dùng tìm kiếm trên cửa hàng Google Play, biểu tượng của ứng dụng sẽ xuất hiện trong kết quả tìm kiếm.

Khi một người dùng đã cài đặt ứng dụng, biểu tượng khởi chạy xuất hiện trên thiết bị ở nhiều vị trí khác nhau, bao gồm màn hình chính và màn hình Tìm kiếm Ứng dụng. Ví dụ, ứng dụng HelloToast xuất hiện trong màn hình Tìm kiếm Ứng dụng của trình giả lập với biểu tượng tiêu chuẩn cho các dự án ứng dụng mới như được hiển thị bên dưới.

Thay đổi biểu tượng khởi chạy là một quá trình đơn giản từng bước, giúp bạn làm quen với các tính năng tài sản hình ảnh của Android Studio. Trong nhiệm vụ này, bạn cũng sẽ tìm hiểu thêm về cách truy cập tài liệu chính thức của Android.

### **1. Khám phá tài liệu chính thức của Android**

Bạn có thể tìm thấy tài liệu nhà phát triển chính thức của Android tại [developer.android.com](https://developer.android.com). Tài liệu này chứa một lượng lớn thông tin được Google cập nhật liên tục.

- Truy cập [developer.android.com/design/](https://developer.android.com/design/). Phần này nói về Material Design, một triết lý thiết kế khái niệm mô tả cách các ứng dụng nên trông như thế nào và hoạt động ra sao trên các thiết bị di động. Duyệt qua các liên kết để tìm hiểu thêm về Material Design. Ví dụ, truy cập phần "Style" để tìm hiểu thêm về cách sử dụng màu sắc và các chủ đề khác.
- Truy cập [developer.android.com/docs/](https://developer.android.com/docs/) để tìm thông tin về API, tài liệu tham khảo, hướng dẫn, công cụ và các mẫu mã nguồn.
- Truy cập [developer.android.com/distribute/](https://developer.android.com/distribute/) để tìm thông tin về cách đưa ứng dụng lên Google Play, hệ thống phân phối kỹ thuật số của Google dành cho các ứng dụng được phát triển bằng Android SDK. Sử dụng Google Play Console để mở rộng cơ sở người dùng và bắt đầu kiếm tiền.

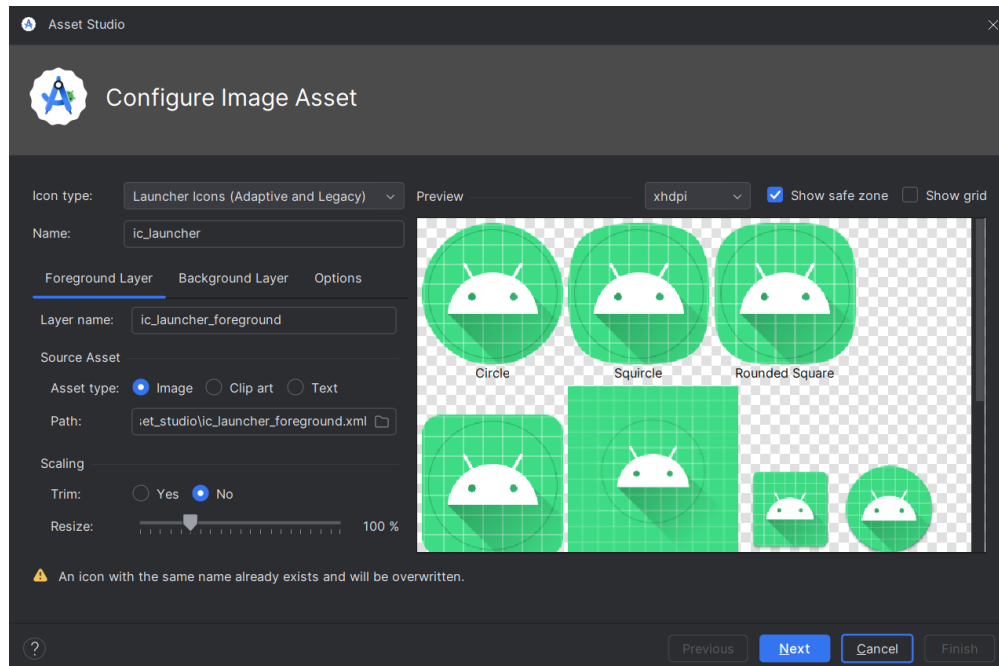
### **2. Thêm tài sản hình ảnh cho biểu tượng khởi chạy**

Để thêm một hình ảnh clip-art làm biểu tượng khởi chạy, hãy làm theo các bước sau:

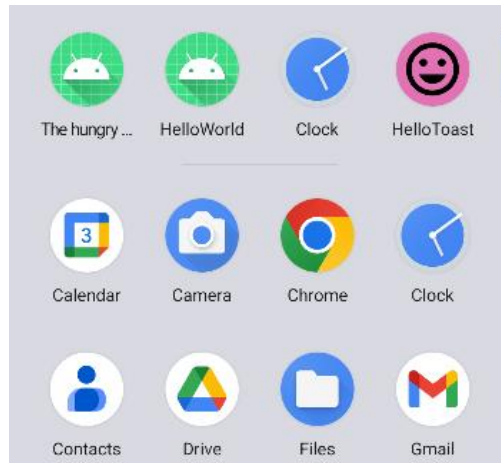
- Mở dự án ứng dụng HelloToast từ bài học trước về sử dụng trình chỉnh sửa bố cục, hoặc tạo một dự án ứng dụng mới.



- Trong bảng "Project > Android", nhấp chuột phải (hoặc nhấn Control và nhấp) vào thư mục res, sau đó chọn New > Image Asset. Cửa sổ "Configure Image Asset" xuất hiện.



- Trong trường "Icon Type", chọn Launcher Icons (Adaptive & Legacy) nếu chưa được chọn.
- Nhấp vào tab Foreground Layer, chọn Clip Art cho "Asset Type".
- Nhấp vào biểu tượng trong trường "Clip Art". Các biểu tượng từ bộ biểu tượng Material Design sẽ xuất hiện.
- Duyệt qua cửa sổ "Select Icon", chọn một biểu tượng phù hợp (chẳng hạn biểu tượng mood để gợi ý tâm trạng vui vẻ), sau đó nhấp OK.
- Nhấp vào tab Background Layer, chọn Color làm "Asset Type", sau đó nhấp vào hộp màu để chọn màu làm lớp nền.
- Nhấp vào tab Legacy và xem lại các cài đặt mặc định. Xác nhận rằng bạn muốn tạo các biểu tượng kế thừa, biểu tượng tròn và biểu tượng Google Play Store. Nhấp Next khi hoàn tất.
- Chạy ứng dụng.



Android Studio sẽ tự động thêm các hình ảnh khởi chạy vào thư mục mipmap cho các độ phân giải khác nhau. Kết quả là, biểu tượng khởi chạy của ứng dụng sẽ thay đổi thành biểu tượng mới sau khi bạn chạy ứng dụng, như minh họa bên dưới.

## Nhiệm vụ 2: Sử dụng mẫu dự án

Android Studio cung cấp các mẫu cho các thiết kế ứng dụng và hoạt động phổ biến, được khuyến nghị. Việc sử dụng các mẫu tích hợp giúp tiết kiệm thời gian và hỗ trợ bạn tuân theo các phương pháp thiết kế tốt nhất.

Mỗi mẫu bao gồm một hoạt động và giao diện người dùng khung sườn. Bạn đã sử dụng mẫu Empty Views Activity trước đó. Mẫu Basic Activity có nhiều tính năng hơn và tích hợp các tính năng ứng dụng được khuyến nghị, chẳng hạn như menu tùy chọn xuất hiện trên thanh ứng dụng.

### 1. Khám phá kiến trúc của Basic Activity

Mẫu Basic Activity là một mẫu linh hoạt do Android Studio cung cấp để hỗ trợ bạn khởi động nhanh quá trình phát triển ứng dụng.

- Trong Android Studio, tạo một dự án mới với mẫu Basic Activity.
- Biên dịch và chạy ứng dụng.
- Xác định các phần được gán nhãn trong hình và bảng bên dưới. Tìm các phần tương đương trên thiết bị hoặc trình giả lập của bạn. Kiểm tra mã Java và tệp XML tương ứng được mô tả trong bảng.

Làm quen với mã nguồn Java và tệp XML sẽ giúp bạn mở rộng và tùy chỉnh mẫu này theo nhu cầu riêng của mình.

Kiến trúc của mẫu Hoạt động Cơ bản

#	Mô tả UI	Tham chiếu mã
1	Thanh trạng thái Hệ thống Android cung cấp và kiểm soát thanh trạng thái.	Không hiển thị trong mã mẫu. Có thể truy cập từ hoạt động của bạn. Ví dụ, bạn có thể ẩn thanh trạng thái nếu cần.
2	AppBarLayout > Toolbar Thanh ứng dụng (còn gọi là thanh hành động) cung cấp cấu trúc trực quan, các phần tử giao diện tiêu chuẩn và điều hướng. Để đảm bảo khả năng tương thích ngược, AppBarLayout trong mẫu nhúng một Toolbar có cùng chức năng như một ActionBar.	Trong activity_main.xml, tìm android.support.v7.widget.Toolbar bên trong android.support.design.widget.AppBarLayout.  Thay đổi thanh công cụ để thay đổi giao diện của thanh ứng dụng. Xem ví dụ trong Hướng dẫn Thanh Ứng Dụng
3	Tên ứng dụng Tên này được lấy từ tên gói của bạn, nhưng có thể thay đổi theo ý muốn.	Trong AndroidManifest.xml: android:label="@string/app_name"
4	Nút menu tùy chọn Chứa các mục menu của hoạt động cũng như các tùy chọn chung, như Tìm kiếm và Cài đặt cho ứng dụng.	Trong MainActivity.java: onOptionsItemSelected() xác định hành động khi một mục menu được chọn.  Trong res > menu > menu_main.xml: Xác định các mục menu cho menu tùy chọn.
5	ViewPager Bố cục CoordinatorLayout là một ViewGroup giàu tính năng cung cấp cơ chế cho các phần tử giao diện người dùng (UI) tương tác với nhau. Giao diện ứng dụng nằm trong content_main.xml được bao gồm trong ViewPager này.	Trong activity_main.xml: Không có view nào được chỉ định; thay vào đó, nó bao gồm một bố cục khác với include layout để thêm @layout/content_main, nơi các view được chỉ định. Điều này giúp tách các view hệ thống khỏi các view dành riêng cho ứng dụng của bạn.
6	TextView Trong ví dụ, được sử dụng để hiển thị "Hello World". Bạn có thể thay thế nó bằng các phần tử UI cho ứng dụng của mình.	Trong content_main.xml: Tất cả các phần tử UI của ứng dụng được xác định trong tệp này.
7	Nút hành động nổi (FAB)	Trong activity_main.xml dưới dạng một phần tử UI sử dụng biểu tượng clip-art. MainActivity.java có một đoạn mã trong onCreate() để thiết lập trình nghe onClick() cho FAB.

## 2. Tùy chỉnh ứng dụng được tạo bởi mẫu

Thay đổi giao diện của ứng dụng được tạo bởi mẫu Hoạt động Cơ bản. Ví dụ, bạn có thể thay đổi màu thanh ứng dụng để phù hợp với thanh trạng thái (trên một số thiết bị, màu này có thể là phiên bản tối hơn của màu chính). Bạn cũng có thể xóa nút hành động nổi nếu không sử dụng nó.

- Thay đổi màu thanh ứng dụng (Toolbar) trong activity\_main.xml bằng cách thay đổi thuộc tính android:background thành "?attr/colorPrimaryDark", giúp thiết lập màu thanh ứng dụng thành một màu chính tối hơn để phù hợp với thanh trạng thái.

```
<com.google.android.material.appbar.MaterialToolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimaryDark"/>
```

- Để xóa nút hành động nổi, trước tiên hãy xóa đoạn mã trong onCreate() thiết lập trình nghe onClick() cho nút. Mở MainActivity.java và xóa đoạn mã sau:

```
binding.fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
            .setAnchorView(R.id.fab)
            .setAction("Action", null).show();
    }
});
```

- Để xóa nút hành động nổi khỏi bố cục, hãy xóa đoạn mã XML sau khỏi activity\_main.xml:

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_marginEnd="@dimen/fab_margin"
    android:layout_marginBottom="16dp"
    app:srcCompat="@android:drawable/ic_dialog_email" />
```

- Thay đổi tên của ứng dụng hiển thị trên thanh ứng dụng bằng cách chỉnh sửa tài nguyên chuỗi app\_name trong strings.xml thành giá trị mới.
  - Chạy ứng dụng. Nút hành động nổi không còn xuất hiện, tên đã thay đổi và màu nền của thanh ứng dụng cũng đã thay đổi.
3. Khám phá cách thêm hoạt động bằng các mẫu

Cho đến nay, bạn đã sử dụng các mẫu Hoạt động Rỗng (Empty Views Activity) và Hoạt động Cơ Bản (Basic Activity). Trong các bài học sau, các mẫu hoạt động bạn sử dụng sẽ thay đổi tùy theo nhiệm vụ.

Các mẫu hoạt động này cũng có sẵn trong dự án của bạn, vì vậy bạn có thể thêm nhiều hoạt động hơn vào ứng dụng sau khi thiết lập ban đầu. (Bạn sẽ học thêm về lớp Hoạt động - Activity trong một chương khác.)

- Tạo một dự án ứng dụng mới hoặc chọn một dự án hiện có.
- Trong Project > Android, nhấp chuột phải vào thư mục java.
- Chọn New > Activity > Gallery.
- Thêm một Hoạt động. Ví dụ, nhấp vào Navigation Drawer Activity để thêm một Hoạt động có ngăn điều hướng vào ứng dụng của bạn.
- Nhấp đúp vào các tệp bố cục của Hoạt động để hiển thị chúng trong trình chỉnh sửa bố cục.

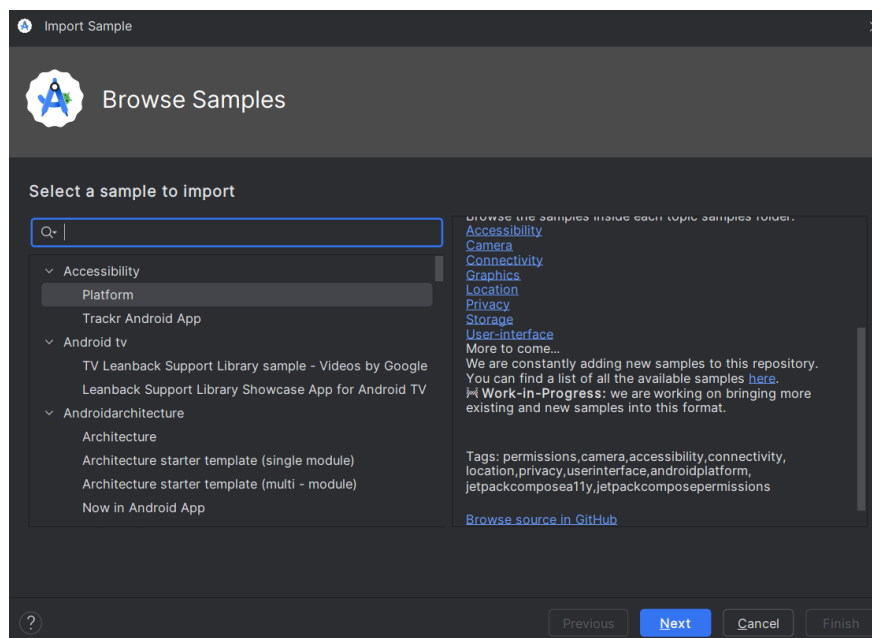
### Nhiệm vụ 3: Học từ mã nguồn ví dụ

Android Studio và tài liệu Android cung cấp nhiều mẫu mã nguồn mà bạn có thể nghiên cứu, sao chép và tích hợp vào các dự án của mình.

#### 1. Mẫu mã nguồn Android

Bạn có thể khám phá hàng trăm mẫu mã nguồn trực tiếp trong Android Studio.

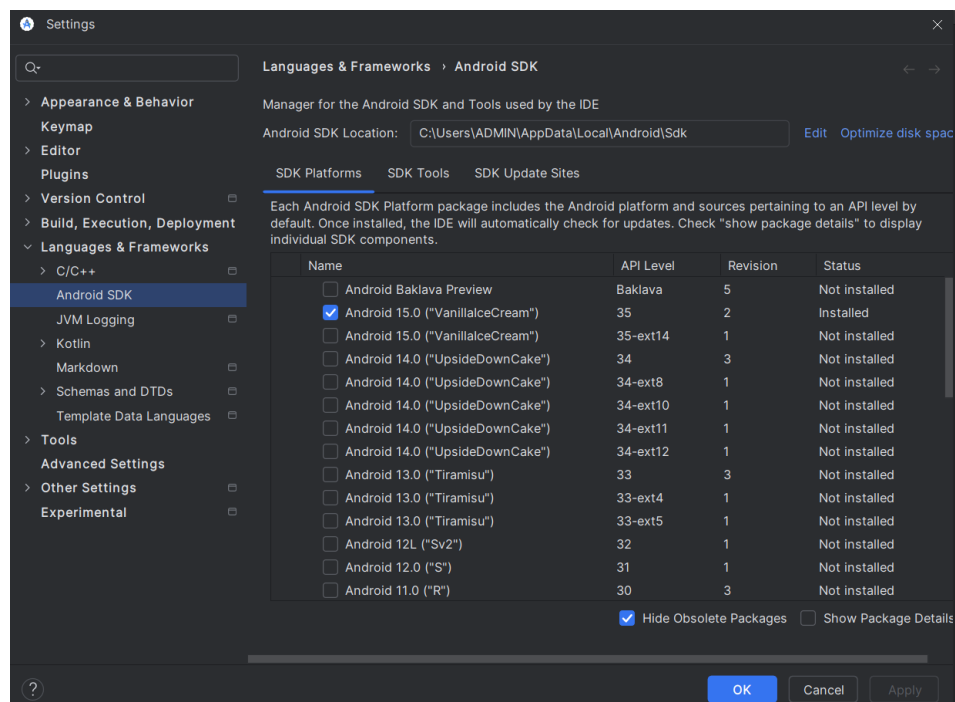
- Trong Android Studio, chọn File > New > Import Sample.



- Duyệt qua các mẫu.
  - Chọn một mẫu và nhấp vào Next.
  - Chấp nhận các giá trị mặc định và nhấp vào Finish.
2. Sử dụng SDK Manager để cài đặt tài liệu ngoại tuyến

Việc cài đặt Android Studio cũng bao gồm các thành phần thiết yếu của Android SDK (Bộ công cụ phát triển phần mềm). Tuy nhiên, các thư viện và tài liệu bổ sung có sẵn, và bạn có thể cài đặt chúng bằng SDK Manager.

- Chọn Tools > SDK Manager.
- Ở cột bên trái, nhấp vào Android SDK.
- Chọn và sao chép đường dẫn của Android SDK Location ở đầu màn hình, vì bạn sẽ cần nó để xác định vị trí tài liệu trên máy tính của mình.



- Nhấp vào tab SDK Platforms. Bạn có thể cài đặt các phiên bản bổ sung của hệ thống Android từ đây.
- Nhấp vào tab SDK Update Sites. Android Studio kiểm tra định kỳ các trang web được liệt kê và chọn để cập nhật.
- Nhấp vào tab SDK Tools. Bạn có thể cài đặt các công cụ SDK bổ sung chưa được cài đặt mặc định, cũng như phiên bản ngoại tuyến của tài liệu dành cho nhà phát triển Android.
- Chọn ô kiểm tra "Documentation for Android SDK" nếu nó chưa được cài đặt, sau đó nhấp vào Apply.
- Khi quá trình cài đặt hoàn tất, nhấp vào Finish.

- Điều hướng đến thư mục SDK mà bạn đã sao chép ở bước trên, sau đó mở thư mục docs.
- Tìm index.html và mở nó.

## Nhiệm vụ 4: Nhiều nguồn tài nguyên hơn

- Kênh YouTube Android Developer là một nguồn tài liệu tuyệt vời về hướng dẫn và mẹo lập trình.
  - Nhóm Android đăng tin tức và mẹo trên blog chính thức của Android.
  - Stack Overflow là một cộng đồng lập trình viên giúp đỡ lẫn nhau. Nếu bạn gặp vấn đề, khả năng cao là ai đó đã đăng câu trả lời trước đó. Hãy thử đăng câu hỏi như "Làm thế nào để thiết lập và sử dụng ADB qua WiFi?" hoặc "Các lỗi rò rỉ bộ nhớ phổ biến nhất trong lập trình Android là gì?"
  - Và cuối cùng, hãy nhập câu hỏi của bạn vào công cụ tìm kiếm Google, nó sẽ thu thập các kết quả liên quan từ tất cả các nguồn này. Ví dụ, "Phiên bản Android OS phổ biến nhất ở Ấn Độ là gì?"
1. Tìm kiếm trên Stack Overflow bằng thẻ

Truy cập Stack Overflow và nhập [android] vào hộp tìm kiếm. Dấu ngoặc vuông [] chỉ định rằng bạn muốn tìm kiếm các bài viết có gắn thẻ liên quan đến Android. Bạn có thể kết hợp thẻ và từ khóa để làm cho tìm kiếm cụ thể hơn. Hãy thử các tìm kiếm sau:

- [android] và [layout]
- [android] "hello world"

Để tìm hiểu thêm về các cách tìm kiếm trên Stack Overflow, hãy xem trung tâm trợ giúp Stack Overflow.

## Tóm tắt

- Tài liệu chính thức của Android Developer: [developer.android.com](https://developer.android.com)
- Material Design là triết lý thiết kế khái niệm xác định cách ứng dụng nên hiển thị và hoạt động trên thiết bị di động.
- Google Play Store là hệ thống phân phối kỹ thuật số của Google dành cho các ứng dụng được phát triển với Android SDK.
- Android Studio cung cấp các mẫu thiết kế cho ứng dụng và hoạt động phổ biến. Các mẫu này cung cấp mã nguồn hoạt động cho các trường hợp sử dụng phổ biến.
- Khi bạn tạo một dự án, bạn có thể chọn một mẫu cho hoạt động đầu tiên của mình.
- Khi tiếp tục phát triển ứng dụng, bạn có thể tạo thêm các thành phần ứng dụng từ các mẫu tích hợp sẵn.
- Android Studio chứa nhiều mã nguồn mẫu mà bạn có thể nghiên cứu, sao chép và tích hợp vào các dự án của mình.

## **Bài 2) Activities**

### **2.1) Activity và Intent**

#### **Giới thiệu**

Một Activity đại diện cho một màn hình trong ứng dụng của bạn, nơi người dùng có thể thực hiện một tác vụ cụ thể như chụp ảnh, gửi email hoặc xem bản đồ. Một activity thường được hiển thị cho người dùng dưới dạng một cửa sổ toàn màn hình.

Một ứng dụng thường bao gồm nhiều màn hình liên kết lỏng lẻo với nhau. Mỗi màn hình là một activity. Thông thường, một activity trong ứng dụng được chỉ định là "main" activity (MainActivity.java), được hiển thị cho người dùng khi ứng dụng khởi động. Main activity sau đó có thể khởi chạy các activity khác để thực hiện các hành động khác nhau.

Mỗi khi một activity mới được khởi chạy, activity trước đó sẽ bị dừng, nhưng hệ thống vẫn giữ activity đó trong một ngăn xếp (ngăn xếp "back stack"). Khi một activity mới bắt đầu, activity mới đó sẽ được đẩy vào ngăn xếp và chiếm quyền tập trung của người dùng. Ngăn xếp back stack tuân theo logic cơ bản của "vào sau, ra trước" (last in, first out). Khi người dùng hoàn tất hoạt động hiện tại và nhấn nút Quay lại (Back), activity đó sẽ bị loại bỏ khỏi ngăn xếp và bị hủy, sau đó activity trước đó sẽ tiếp tục hoạt động.

Một activity được khởi chạy hoặc kích hoạt bằng một intent. Intent là một thông điệp không đồng bộ mà bạn có thể sử dụng trong activity của mình để yêu cầu một hành động từ một activity khác hoặc từ một thành phần ứng dụng khác. Bạn sử dụng intent để khởi chạy một activity từ một activity khác và để truyền dữ liệu giữa các activity.

Intent có thể là intent tường minh (explicit) hoặc intent ẩn (implicit):

- Intent tường minh là intent mà bạn biết trước đích đến của nó. Nghĩa là, bạn đã biết tên lớp đầy đủ của activity cụ thể đó.
- Intent ẩn là intent mà bạn không có tên của thành phần đích, nhưng bạn có một hành động chung cần thực hiện. Trong bài thực hành này, bạn sẽ tạo intent tường minh. Bạn sẽ tìm hiểu cách sử dụng intent ẩn trong một bài thực hành sau.

#### **Những gì bạn cần biết trước**

Bạn cần có khả năng:

- Tạo và chạy ứng dụng trong Android Studio.



- Sử dụng trình chỉnh sửa bố cục (layout editor) để tạo bố cục trong ConstraintLayout.
- Chỉnh sửa mã XML của bố cục.
- Thêm chức năng onClick vào một Button.

## **Những gì bạn sẽ học**

- Cách tạo một Activity mới trong Android Studio.
- Cách xác định activity cha và activity con để điều hướng Up.
- Cách khởi chạy một Activity bằng Intent tường minh.
- Cách truyền dữ liệu giữa các Activity bằng Intent tường minh.

## **Những gì bạn sẽ làm**

- Tạo một ứng dụng Android mới với một main Activity và một second Activity.
- Truyền một số dữ liệu (chuỗi ký tự) từ main Activity đến second Activity bằng Intent và hiển thị dữ liệu đó trong second Activity.
- Gửi một dữ liệu khác trở lại main Activity, cũng bằng Intent.

## **Nhiệm vụ 1: Tạo dự án TwoActivities**

Trong nhiệm vụ này, bạn thiết lập dự án ban đầu với một main Activity, xác định bố cục và định nghĩa một phương thức skeleton cho sự kiện onClick của nút bấm.

### **1. Tạo dự án TwoActivities**

- Khởi động Android Studio và tạo một dự án Android Studio mới.
- Đặt tên ứng dụng của bạn là Two Activities và chọn cùng các cài đặt Phone và Tablet mà bạn đã sử dụng trong các bài thực hành trước đó. Thư mục dự án sẽ tự động được đặt tên là TwoActivities và tên ứng dụng hiển thị trên thanh ứng dụng sẽ là "Two Activities".
- Chọn Empty Views Activity làm mẫu Activity. Nhấp Next.
- Chấp nhận tên Activity mặc định (MainActivity). Đảm bảo đã chọn tùy chọn Generate Layout file và Backwards Compatibility (AppCompat).
- Nhấp Finish.

### **2. Xác định bố cục cho main Activity**

- Mở res > layout > activity\_main.xml trong Project > Android. Trình chỉnh sửa bố cục sẽ xuất hiện.
- Nhấp vào tab Design nếu nó chưa được chọn, sau đó xóa TextView (dòng chữ "Hello World") trong Component Tree.

- Với Autoconnect được bật (cài đặt mặc định), kéo một Button từ Palette vào góc dưới bên phải của bố cục. Autoconnect sẽ tạo các ràng buộc cho Button.
- Trong Attributes, đặt ID là `button_main`, đặt `layout_width` và `layout_height` thành `wrap_content`, và nhập Send vào trường Text.
- Nhấp vào tab Text để chỉnh sửa mã XML. Thêm thuộc tính sau vào Button: Giá trị thuộc tính sẽ bị gạch chân màu đỏ vì phương thức `launchSecondActivity()` chưa được tạo. Bỏ qua lỗi này ngay bây giờ; bạn sẽ sửa nó trong nhiệm vụ tiếp theo.
- Trích xuất chuỗi tài nguyên, như đã mô tả trong một bài thực hành trước đó, cho văn bản "Send" và sử dụng tên `button_main` cho tài nguyên.

Mã XML cho Button sẽ trông như sau:

```
<Button
    android:id="@+id/button_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="16dp"
    android:text="@string/button_main"
    android:textSize="20sp"
    android:onClick="launchSecondActivity"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent"/>
```

### 3. Định nghĩa hành động của Button

Trong nhiệm vụ này, bạn triển khai phương thức `launchSecondActivity()` mà bạn đã tham chiếu trong bố cục cho thuộc tính `android:onClick`.

- Nhấp vào "launchSecondActivity" trong mã XML của `activity_main.xml`.
- Nhấn Alt+Enter (hoặc Option+Enter trên Mac) và chọn Create 'launchSecondActivity(View)' in 'MainActivity'.  
Tập `MainActivity` sẽ mở ra và Android Studio sẽ tạo một phương thức cho trình xử lý `launchSecondActivity()`.
- Bên trong `launchSecondActivity()`, thêm một câu lệnh Log với nội dung "Button Clicked!". `LOG_TAG` sẽ hiển thị màu đỏ. Bạn sẽ thêm định nghĩa cho biến đó ở bước sau.
- Ở đầu lớp `MainActivity`, thêm một hằng số cho biến `LOG_TAG`: Hằng số này sử dụng chính tên lớp làm tag.
- Chạy ứng dụng của bạn. Khi bạn nhấp vào nút Send, bạn sẽ thấy thông báo "Button Clicked!" trong Logcat. Nếu có quá nhiều đầu ra trong trình giám sát,

nhập MainActivity vào hộp tìm kiếm và Logcat chỉ hiển thị các dòng khớp với tag đó.

Mã cho MainActivity sẽ trông như sau:

```
public class MainActivity extends AppCompatActivity {
    private static final String LOG_TAG = MainActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void launchSecondActivity(View view) {
        Log.d(LOG_TAG, "Button clicked!");
    }
}
```

## Nhiệm vụ 2: Tạo và khởi chạy Activity thứ hai

Mỗi Activity mới bạn thêm vào dự án sẽ có bố cục và tệp Java riêng, tách biệt với Activity chính. Chúng cũng có các phần tử <activity> riêng trong tệp AndroidManifest.xml.

Giống như Activity chính, các Activity mới mà bạn tạo trong Android Studio cũng kế thừa từ lớp AppCompatActivity. Mỗi Activity trong ứng dụng của bạn chỉ được kết nối lỏng lẻo với các Activity khác. Tuy nhiên, bạn có thể định nghĩa một Activity là cha của một Activity khác trong tệp AndroidManifest.xml. Mỗi quan hệ cha-con này cho phép Android thêm các gợi ý điều hướng như mũi tên quay lại ở thanh tiêu đề của mỗi Activity.

Một Activity giao tiếp với các Activity khác (trong cùng một ứng dụng và giữa các ứng dụng khác nhau) bằng một Intent. Intent có thể là Intent tường minh hoặc Intent ẩn:

- Intent tường minh là Intent mà bạn biết rõ đích đến của nó, nghĩa là bạn đã biết tên lớp đầy đủ của Activity cụ thể đó.
- Intent ẩn là Intent mà bạn không có tên của thành phần đích, nhưng có một hành động chung cần thực hiện.

Trong nhiệm vụ này, bạn sẽ thêm một Activity thứ hai vào ứng dụng với bố cục riêng. Bạn sẽ sửa đổi tệp AndroidManifest.xml để xác định Activity chính là cha của Activity thứ hai. Sau đó, bạn sẽ sửa đổi phương thức launchSecondActivity() trong MainActivity để bao gồm một Intent giúp khởi chạy Activity thứ hai khi nhấn vào nút.

## 1. Tạo Activity thứ hai

- Nhấp vào thư mục app của dự án và chọn File > New > Activity > Empty Views Activity.
- Đặt tên cho Activity mới là SecondActivity. Đảm bảo rằng các tùy chọn Generate Layout File và Backwards Compatibility (AppCompat) được chọn. Tên bố cục sẽ được điền tự động là activity\_second. Không chọn tùy chọn Launcher Activity.
- Nhấp vào Finish. Android Studio sẽ thêm một bố cục Activity mới (activity\_second.xml) và một tệp Java mới (SecondActivity.java) vào dự án của bạn cho Activity mới. Nó cũng sẽ cập nhật tệp AndroidManifest.xml để bao gồm Activity mới.

## 2. Sửa đổi tệp AndroidManifest.xml

- Mở manifests > AndroidManifest.xml.
- Tìm phần tử <activity> mà Android Studio đã tạo cho Activity thứ hai
- Thay thế toàn bộ phần tử <activity> bằng đoạn mã sau:

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

Thuộc tính label thêm tiêu đề của Activity vào thanh ứng dụng.

Thuộc tính parentActivityName xác định rằng MainActivity là cha của SecondActivity. Mối quan hệ này được sử dụng để điều hướng "lên trên" trong ứng dụng: thanh ứng dụng của SecondActivity sẽ có một mũi tên quay lại, cho phép người dùng quay về MainActivity.

Phần tử <meta-data> cung cấp thông tin bổ sung về Activity dưới dạng cặp khóa-giá trị. Trong trường hợp này, nó xác định mối quan hệ giữa hai Activity để hỗ trợ điều hướng trên các phiên bản Android cũ hơn, vì thuộc tính android:parentActivityName chỉ có sẵn từ API cấp 16 trở lên.

- Trích xuất một tài nguyên chuỗi cho "Second Activity" trong mã trên và đặt tên tài nguyên là activity2\_name
- ## 3. Xác định bố cục cho Activity thứ hai
- Mở activity\_second.xml và nhấp vào tab Design nếu nó chưa được chọn.

- Kéo một TextView từ bảng Palette vào góc trên bên trái của bố cục, sau đó thêm các ràng buộc (constraints) vào cạnh trên và cạnh trái của bố cục. Thiết lập các thuộc tính của nó trong bảng Attributes như sau:

Thuộc tính	Giá trị
id	text_header
Top margin	16
Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	Message Received
textAppearance	AppCompat.Medium
textStyle	B (bold)

Giá trị của textAppearance là một thuộc tính đặc biệt của giao diện Android, định nghĩa các kiểu chữ cơ bản.

- Nhấp vào tab Text để chỉnh sửa mã XML và trích xuất chuỗi "Message Received" vào một tài nguyên có tên text\_header.
- Thêm thuộc tính android:layout\_marginLeft="8dp" vào TextView để bổ sung cho layout\_marginStart, giúp hỗ trợ các phiên bản Android cũ hơn.

Mã XML cho activity\_second.xml sẽ như sau:

```
<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=".SecondActivity">

    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        android:text="Message Received"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
```

```
android:textStyle="bold"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

#### 4. Thêm một Intent vào MainActivity

Trong nhiệm vụ này, bạn sẽ thêm một Intent tường minh vào MainActivity. Intent này sẽ được sử dụng để kích hoạt SecondActivity khi nhấp vào nút Send.

- Mở MainActivity.
- Tạo một Intent mới trong phương thức launchSecondActivity().  
Constructor của Intent nhận hai đối số cho một Intent tường minh: một Context ứng dụng và một thành phần cụ thể sẽ nhận Intent đó.  
Ở đây, bạn nên sử dụng this làm Context và SecondActivity.class làm class cụ thể
- Gọi phương thức startActivity() với Intent vừa tạo làm đối số.

```
public void launchSecondActivity(View view) {
    Intent intent = new Intent(this, SecondActivity.class);
    startActivity(intent);
}
```

- Chạy ứng dụng.  
Khi bạn nhấp vào nút Send, MainActivity sẽ gửi Intent và hệ thống Android sẽ khởi chạy SecondActivity, hiển thị nó trên màn hình. Để quay lại MainActivity, bạn có thể nhấp vào nút Up (mũi tên trái trên thanh ứng dụng) hoặc nút Back ở cuối màn hình.

### Nhiệm vụ 3: Gửi dữ liệu từ MainActivity đến SecondActivity

Ở nhiệm vụ trước, bạn đã thêm một Intent tường minh vào MainActivity để khởi chạy SecondActivity. Bạn cũng có thể sử dụng một Intent để gửi dữ liệu từ một Activity sang một Activity khác khi khởi chạy nó.

Đối tượng Intent có thể truyền dữ liệu sang Activity đích theo hai cách: Trường dữ liệu (Data field): Là một URI chỉ định dữ liệu cụ thể cần xử lý. Intent extras: Nếu thông tin bạn muốn truyền không phải là một URI, hoặc bạn cần gửi nhiều hơn một phần thông tin, bạn có thể đặt thông tin bổ sung vào extras.

Intent extras là các cặp khóa/giá trị trong một Bundle. Bundle là một tập hợp dữ liệu được lưu trữ dưới dạng cặp khóa/giá trị. Để truyền thông tin từ một Activity sang một Activity khác, bạn sẽ đặt các khóa và giá trị vào Intent extra Bundle trong Activity gửi và lấy chúng ra trong Activity nhận.

Trong nhiệm vụ này, bạn sẽ sửa đổi Intent tương minh trong MainActivity để bao gồm dữ liệu bổ sung (cụ thể là một chuỗi do người dùng nhập vào). Sau đó, bạn sẽ chỉnh sửa SecondActivity để lấy dữ liệu đó từ Intent extra Bundle và hiển thị nó trên màn hình.

### 1. Thêm EditText vào bố cục MainActivity

- Mở activity\_main.xml.
- Kéo một phần tử Plain Text (EditText) từ bảng Palette xuống cuối bố cục, sau đó thêm các ràng buộc (constraints) vào cạnh trái của bố cục, cạnh dưới của bố cục, và cạnh trái của nút Send. Thiết lập các thuộc tính của nó trong bảng Attributes như sau:

Thuộc tính	Giá trị
id	editText_main
Right margin	8
Left margin	8
Bottom margin	16
layout_width	match_constraint
layout_height	wrap_content
inputType	textLongMessage
hint	Enter Your Message Here
text	(Delete any text in this field)

- Nhấp vào tab Text để chỉnh sửa mã XML, sau đó trích xuất chuỗi "Enter Your Message Here" vào một tài nguyên có tên editText\_main.

Mã XML cho bố cục sẽ trông giống 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">

    <Button
        android:id="@+id/button_main"
        android:layout_width="wrap_content"
```

```

        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:text="@string/button_main"
        android:textSize="20sp"
        android:onClick="launchSecondActivity"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

```

<EditText

```

        android:id="@+id/editText_main"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="16dp"
        android:ems="10"
        android:hint="@string/editText_main"
        android:inputType="textLongMessage"
        android:minHeight="48dp"
        android:padding="12dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button_main"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintHorizontal_bias="1.0"/>

```

</androidx.constraintlayout.widget.ConstraintLayout>

## 2. Thêm một chuỗi vào Intent extras

Intent extras là các cặp khóa/giá trị trong một Bundle. Để truyền thông tin từ MainActivity sang SecondActivity, bạn cần đặt dữ liệu vào Intent extra Bundle từ Activity gửi và lấy nó ra ở Activity nhận.

- Mở MainActivity.
- Thêm một hằng số công khai ở đầu lớp để định nghĩa khóa cho Intent extra:

```

public static final String EXTRA_MESSAGE =
    "com.example.android.twoactivities.extra.MESSAGE";

```

- Thêm một biến private ở đầu lớp để lưu tham chiếu đến EditText:

```

private EditText mMessageEditText;

```



- Trong phương thức onCreate(), sử dụng findViewById() để lấy tham chiếu đến EditText và gán nó cho biến private
- Trong phương thức launchSecondActivity(), ngay dưới new Intent, lấy văn bản từ EditText dưới dạng chuỗi
- Thêm chuỗi đó vào Intent như một extra, sử dụng hằng số EXTRA\_MESSAGE làm khóa và chuỗi làm giá trị

Phương thức onCreate() trong MainActivity sẽ như sau:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mMessageEditText = findViewById(R.id.editText_main);
}
```

Phương thức launchSecondActivity() trong MainActivity sẽ như sau:

```
public void launchSecondActivity(View view) {
    Log.d(LOG_TAG, "Button clicked!");
    Intent intent = new Intent(this, SecondActivity.class);
    String message = mMessageEditText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

### 3. Thêm một TextView vào SecondActivity để hiển thị tin nhắn

- Mở activity\_second.xml.
- Kéo một TextView khác vào bố cục, đặt bên dưới TextView text\_header, sau đó thêm ràng buộc vào cạnh trái của bố cục và cạnh dưới của text\_header.
- Thiết lập các thuộc tính của TextView trong bảng Attributes như sau:

Thuộc tính	Giá trị
id	text_message
Top margin	8
Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	(Delete any text in this field)
textAppearance	AppCompat.Medium

Bố cục mới vẫn giống như trước đó, vì TextView mới chưa chứa bất kỳ văn bản nào, do đó nó chưa xuất hiện trên màn hình.

Mã XML cho bố cục activity\_second.xml sẽ 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=".SecondActivity">

    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        android:text="Message Received"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/text_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/text_header" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

#### 4. Chỉnh sửa SecondActivity để nhận extras và hiển thị tin nhắn

- Mở SecondActivity để thêm mã vào phương thức onCreate().
- Nhận Intent đã kích hoạt Activity này
- Lấy chuỗi chứa tin nhắn từ Intent extras bằng cách sử dụng biến tĩnh MainActivity.EXTRA\_MESSAGE làm khóa

- Sử dụng findViewById() để lấy tham chiếu đến TextView hiển thị tin nhắn từ bố cục
- Đặt nội dung của TextView thành chuỗi từ Intent extra
- Chạy ứng dụng. Khi nhập tin nhắn trong MainActivity và nhấn Send, SecondActivity sẽ khởi chạy và hiển thị tin nhắn.

Phương thức onCreate() trong SecondActivity sẽ giống như sau:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
    TextView textView = findViewById(R.id.text_message);
    textView.setText(message);
}
```

#### Nhiệm vụ 4: Trả dữ liệu về MainActivity

Bây giờ bạn đã có một ứng dụng có thể mở một Activity mới và gửi dữ liệu đến đó, bước cuối cùng là trả dữ liệu từ SecondActivity về MainActivity. Bạn cũng sử dụng Intent và Intent extras cho nhiệm vụ này.

##### 1. Thêm EditText và Button vào bố cục SecondActivity

- Mở strings.xml và thêm tài nguyên chuỗi cho văn bản của Button và gợi ý của EditText mà bạn sẽ thêm vào SecondActivity:

```
<string name="button_second">Reply</string>
<string name="editText_second">Enter Your Reply Here</string>
```

- Mở activity\_main.xml và activity\_second.xml.
- Sao chép EditText và Button từ tệp bố cục activity\_main.xml và dán chúng vào activity\_second.xml.
- Trong activity\_second.xml, chỉnh sửa các thuộc tính của Button như sau:

Thuộc tính cũ	Thuộc tính mới
android:id="@+id/button_main"	android:id="@+id/button_second"
android:onClick="launchSecondActivity"	android:onClick="returnReply"
android:text="@string/button_main"	android:text="@string/button_second"

- Trong activity\_second.xml, chỉnh sửa các thuộc tính của EditText như sau:

Thuộc tính cũ	Thuộc tính mới
android:id="@+id/editText_main"	android:id="@+id/editText_second"
app:layout_constraintEnd_toStartOf="@+id/button"	app:layout_constraintEnd_toStartOf="@+id/button_second"
android:hint="@string/editText_main"	android:hint="@string/editText_second"

- Trong trình chỉnh sửa bố cục XML, nhấp vào returnReply, nhấn Alt+Enter (hoặc Option+Return trên Mac), và chọn Create 'returnReply(View)' in 'SecondActivity'. Android Studio sẽ tạo một phương thức returnReply() khung xương. Bạn sẽ triển khai phương thức này trong nhiệm vụ tiếp theo.

Mã XML của activity\_second.xml 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=".SecondActivity">

    <TextView
        android:id="@+id/text_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        android:text="Message Received"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/text_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/text_header" />
```

```

<Button
    android:id="@+id/button_second"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="16dp"
    android:text="@string/button_second"
    android:textSize="20sp"
    android:onClick="returnReply"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"/>

<EditText
    android:id="@+id/editText_second"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="16dp"
    android:ems="10"
    android:hint="@string/editText_second"
    android:inputType="textLongMessage"
    android:minHeight="48dp"
    android:padding="12dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button_second"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintHorizontal_bias="1.0"/>
</androidx.constraintlayout.widget.ConstraintLayout>

```

## 2. Tạo Intent phản hồi trong SecondActivity

Dữ liệu phản hồi từ SecondActivity trở lại MainActivity sẽ được gửi trong một Intent extra. Bạn tạo Intent này và đưa dữ liệu vào nó theo cách tương tự như khi gửi Intent ban đầu.

- Mở SecondActivity.
- Ở đầu lớp, thêm một hằng số công khai để xác định khóa cho Intent extra:

```
public static final String EXTRA_REPLY = "com.example.android.twoactivities.extra.REPLY";
```

- Thêm một biến riêng tư ở đầu lớp để lưu EditText.

```
private EditText mReply;
```

- Trong phương thức onCreate(), trước mã Intent, sử dụng findViewById() để lấy tham chiếu đến EditText và gán nó vào biến riêng tư
- Trong phương thức returnReply(), lấy nội dung EditText dưới dạng chuỗi
- Trong phương thức returnReply(), tạo một Intent mới để phản hồi—không sử dụng lại đối tượng Intent nhận được từ yêu cầu ban đầu.
- Thêm chuỗi phản hồi từ EditText vào Intent mới dưới dạng Intent extra. Vì extras là cặp khóa/giá trị, ở đây khóa là EXTRA\_REPLY và giá trị là chuỗi phản hồi
- Đặt kết quả thành RESULT\_OK để chỉ ra rằng phản hồi đã thành công. Lớp Activity định nghĩa các mã kết quả, bao gồm RESULT\_OK và RESULT\_CANCELED.
- Gọi finish() để đóng Activity và quay lại MainActivity.

Mã trong SecondActivity bây giờ sẽ như sau:

```
public class SecondActivity extends AppCompatActivity {
    public static final String EXTRA_REPLY =
"com.example.android.twoactivities.extra.REPLY";
    private EditText mReply;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        mReply = findViewById(R.id.editText_second);
        Intent intent = getIntent();
        String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
        TextView textView = findViewById(R.id.text_message);
        textView.setText(message);
    }
    public void returnReply(View view) {
        String reply = mReply.getText().toString();
        Intent replyIntent = new Intent();
        replyIntent.putExtra(EXTRA_REPLY, reply);
        setResult(RESULT_OK, replyIntent);
        finish();
    }
}
```

### 3. Thêm TextView để hiển thị phản hồi

MainActivity cần một cách để hiển thị phản hồi mà SecondActivity gửi. Trong nhiệm vụ này, bạn thêm TextView vào bố cục activity\_main.xml để hiển thị phản hồi trong MainActivity.

Để làm cho nhiệm vụ này dễ dàng hơn, bạn sao chép các TextView đã sử dụng trong SecondActivity.

- Mở strings.xml và thêm một tài nguyên chuỗi cho tiêu đề phản hồi:

```
<string name="text_header_reply">Reply Received</string>
```

- Mở activity\_main.xml và activity\_second.xml.
- Sao chép hai TextView từ tập bố cục activity\_second.xml và dán chúng vào activity\_main.xml ngay phía trên Button.
- Trong activity\_main.xml, chỉnh sửa các thuộc tính của TextView đầu tiên như sau:

Thuộc tính cũ	Thuộc tính mới
android:id="@+id/text_header"	android:id="@+id/text_header_reply"
android:text="@string/text_header"	android:text="@string/text_header_reply"

- Trong activity\_main.xml, chỉnh sửa các thuộc tính của TextView thứ hai như sau:

Thuộc tính cũ	Thuộc tính mới
android:id="@+id/text_message"	android:id="@+id/text_message_reply"
app:layout_constraintTop_toBottomOf="@+id/text_header"	app:layout_constraintTop_toBottomOf="@+id/text_header_reply"

- Thêm thuộc tính android:visibility cho cả hai TextView để chúng ban đầu bị ẩn. (Nếu hiển thị nhưng không có nội dung, người dùng có thể bị nhầm lẫn.) Bạn sẽ làm cho các TextView này hiển thị sau khi dữ liệu phản hồi được truyền từ SecondActivity về. Bố cục activity\_main.xml vẫn sẽ trông giống như trước—mặc dù bạn đã thêm hai TextView mới vào bố cục. Vì bạn đặt TextView thành ẩn, chúng sẽ không xuất hiện trên màn hình.

Mã XML của activity\_main.xml 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/text_header_reply"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="16dp"
    android:text="@string/text_header_reply"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium"
    android:textStyle="bold"
    android:visibility="invisible"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

#### <TextView

```
    android:id="@+id/text_message_reply"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text_header_reply" />
```

#### <Button

```
    android:id="@+id/button_main"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="16dp"
    android:text="@string/button_main"
    android:textSize="20sp"
    android:onClick="launchSecondActivity"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"/>
```

#### <EditText

```
    android:id="@+id/editText_main"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="16dp"
    android:ems="10"
    android:hint="@string/editText_main"
```



```

        android:inputType="textLongMessage"
        android:minHeight="48dp"
        android:padding="12dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/button_main"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintHorizontal_bias="1.0"/>
</androidx.constraintlayout.widget.ConstraintLayout>

```

#### 4. Nhận phản hồi từ Intent extra và hiển thị nó

Khi bạn sử dụng một Intent tường minh để khởi chạy một Activity khác, bạn có thể không mong đợi nhận lại bất kỳ dữ liệu nào—bạn chỉ đơn giản là kích hoạt Activity đó. Trong trường hợp này, bạn sử dụng `startActivity()` để khởi chạy Activity mới, như bạn đã làm trước đó trong thực hành này. Tuy nhiên, nếu bạn muốn nhận lại dữ liệu từ Activity được kích hoạt, bạn cần khởi chạy nó bằng `startActivityForResult()`.

Trong nhiệm vụ này, bạn sẽ sửa đổi ứng dụng để khởi chạy `SecondActivity` với mong đợi nhận kết quả, trích xuất dữ liệu trả về từ Intent và hiển thị dữ liệu đó trong các phần tử `TextView` mà bạn đã tạo ở nhiệm vụ trước.

- Mở `MainActivity`.
- Thêm một hằng số công khai ở đầu lớp để xác định khóa cho loại phản hồi cụ thể mà bạn quan tâm:

```
public static final int TEXT_REQUEST = 1;
```

- Thêm hai biến riêng tư để lưu trữ phần tử `TextView` tiêu đề phản hồi và phản hồi:

```
private TextView mReplyHeadTextView;
private TextView mReplyTextView;
```

- Trong phương thức `onCreate()`, sử dụng `findViewById()` để lấy tham chiếu từ bố cục đến phần tử `TextView` tiêu đề phản hồi và phản hồi. Gán các thể hiện này vào các biến riêng tư:

Phương thức `onCreate()` đầy đủ bây giờ sẽ trông như sau:

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mMessageEditText = findViewById(R.id.editText_main);
    mReplyHeadTextView = findViewById(R.id.text_header_reply);
}

```

```
mReplyTextView = findViewById(R.id.text_message_reply);
}
```

- Trong phương thức launchSecondActivity(), thay đổi lời gọi startActivity() thành startActivityForResult(), và bao gồm khóa TEXT\_REQUEST làm đối số
- Ghi đè phương thức onActivityResult():

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {}
```

Ba đối số của onActivityResult() chứa tất cả thông tin bạn cần để xử lý dữ liệu phản hồi: requestCode bạn đặt khi khởi chạy Activity với startActivityForResult(), resultCode được đặt trong Activity được khởi chạy (thường là RESULT\_OK hoặc RESULT\_CANCELED), và dữ liệu Intent chứa dữ liệu được trả về từ Activity được khởi chạy.

- Bên trong onActivityResult(), gọi super.onActivityResult()
- Thêm mã để kiểm tra TEXT\_REQUEST để đảm bảo bạn xử lý đúng kết quả Intent, trong trường hợp có nhiều Intent khác nhau. Cũng kiểm tra RESULT\_OK để đảm bảo rằng yêu cầu đã thành công  
Lớp Activity định nghĩa các mã kết quả. Mã có thể là RESULT\_OK (yêu cầu thành công), RESULT\_CANCELED (người dùng đã hủy thao tác), hoặc RESULT\_FIRST\_USER (để xác định mã kết quả tùy chỉnh của bạn).
- Bên trong khối if bên trong, lấy Intent extra từ Intent phản hồi (data). Ở đây, khóa cho extra là hằng số EXTRA\_REPLY từ SecondActivity
- Đặt thuộc tính hiển thị (visibility) của tiêu đề phản hồi thành true
- Đặt văn bản cho TextView phản hồi và đặt thuộc tính hiển thị của nó thành true  
Phương thức onActivityResult() đầy đủ bây giờ sẽ trông như sau:

```
@Override
public void onActivityResult(int requestCode,
                             int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == TEXT_REQUEST) {
        if (resultCode == RESULT_OK) {
            String reply =
                data.getStringExtra(SecondActivity.EXTRA_REPLY);
            mReplyHeadTextView.setVisibility(View.VISIBLE);
            mReplyTextView.setText(reply);
            mReplyTextView.setVisibility(View.VISIBLE);
        }
    }
}
```

- Chạy ứng dụng.  
Bây giờ, khi bạn gửi một tin nhắn đến SecondActivity và nhận phản hồi, MainActivity sẽ cập nhật để hiển thị phản hồi.

## Tổng kết

Tổng quan:

- Một Activity là một thành phần ứng dụng cung cấp một màn hình duy nhất tập trung vào một tác vụ cụ thể của người dùng.
- Mỗi Activity có một tệp bố cục giao diện người dùng riêng.
- Bạn có thể gán mối quan hệ cha/con cho các Activity của mình để kích hoạt điều hướng Up trong ứng dụng.
- Một View có thể được hiển thị hoặc ẩn bằng thuộc tính android:visibility.

Để triển khai một Activity:

- Chọn File > New > Activity để bắt đầu từ một mẫu và thực hiện các bước tự động.
- Nếu không bắt đầu từ một mẫu, hãy tạo một lớp Activity bằng Java, triển khai giao diện người dùng cơ bản cho Activity trong một tệp bố cục XML liên kết và khai báo Activity mới trong AndroidManifest.xml.

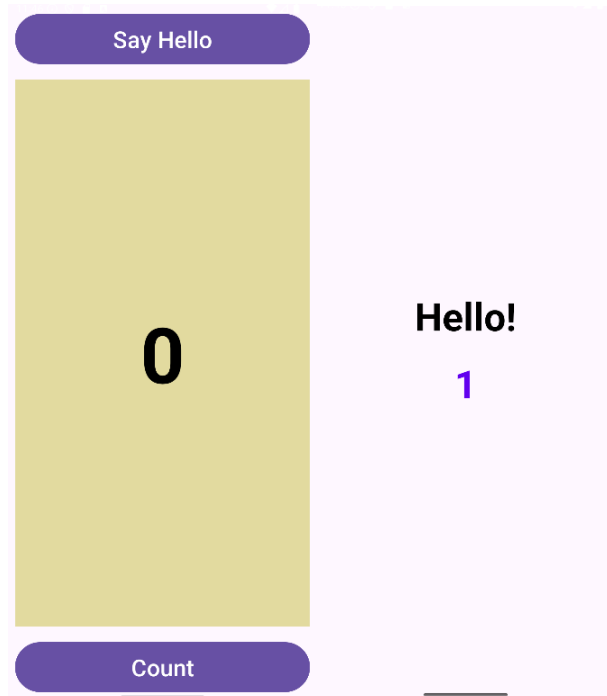
Intent:

- Một Intent cho phép bạn yêu cầu một hành động từ một thành phần khác trong ứng dụng, ví dụ: để khởi chạy một Activity từ một Activity khác. Intent có thể là tường minh (explicit) hoặc ngầm định (implicit).
- Với Intent tường minh, bạn chỉ định rõ thành phần đích nhận dữ liệu.
- Với Intent ngầm định, bạn chỉ định chức năng mong muốn mà không chỉ định thành phần đích.
- Một Intent có thể bao gồm dữ liệu để thực hiện hành động (dưới dạng URI) hoặc thông tin bổ sung dưới dạng Intent extras.
- Intent extras là các cặp key/value trong một Bundle, được gửi kèm với Intent.

## Xây dựng và chạy ứng dụng

Mở ứng dụng HelloToast mà bạn đã tạo trong một bài thực hành trước đó.

- Chỉnh sửa nút Toast để khởi chạy một Activity mới, hiển thị từ "Hello!" và số đếm hiện tại, như hình minh họa bên dưới.
- Thay đổi văn bản trên nút Toast thành Say Hello.



## 2.2) Vòng đời của Activity và trạng thái

### Giới thiệu

Trong bài thực hành này, bạn tìm hiểu thêm về vòng đời của Activity. Vòng đời là tập hợp các trạng thái mà một Activity có thể trải qua trong suốt thời gian tồn tại của nó, từ khi được tạo cho đến khi bị hủy và tài nguyên của nó được hệ thống thu hồi. Khi người dùng điều hướng giữa các Activity trong ứng dụng (cũng như khi vào hoặc thoát khỏi ứng dụng), các Activity sẽ chuyển đổi giữa các trạng thái khác nhau trong vòng đời của chúng.

Mỗi giai đoạn trong vòng đời của Activity có một phương thức callback tương ứng: `onCreate()`, `onStart()`, `onPause()`, v.v. Khi một Activity thay đổi trạng thái, phương thức callback liên quan sẽ được gọi. Bạn đã thấy một trong các phương thức này: `onCreate()`. Bằng cách ghi đè bất kỳ phương thức callback nào của vòng đời trong lớp Activity, bạn có thể thay đổi hành vi mặc định của Activity để phản ứng với hành động của người dùng hoặc hệ thống.

Trạng thái của Activity cũng có thể thay đổi khi có sự thay đổi cấu hình của thiết bị, ví dụ khi người dùng xoay màn hình từ dọc sang ngang. Khi những thay đổi cấu hình này xảy ra, Activity sẽ bị hủy và tạo lại ở trạng thái mặc định, và người dùng có thể mất thông tin đã nhập. Để tránh gây nhầm lẫn cho người dùng, điều quan trọng là bạn phải phát triển ứng dụng để ngăn chặn việc mất dữ liệu ngoài ý muốn. Trong phần sau của bài thực hành, bạn sẽ thử nghiệm với các thay đổi cấu hình và tìm hiểu cách bảo toàn trạng thái

của Activity khi có thay đổi cấu hình thiết bị hoặc các sự kiện vòng đời khác của Activity.

Trong bài thực hành này, bạn sẽ thêm các câu lệnh ghi log vào ứng dụng TwoActivities và quan sát các thay đổi của vòng đời Activity khi sử dụng ứng dụng. Sau đó, bạn sẽ làm việc với các thay đổi này và khám phá cách xử lý đầu vào của người dùng trong các điều kiện này.

## **Những gì bạn nên biết trước**

Bạn nên có khả năng:

- Tạo và chạy một dự án ứng dụng trong Android Studio.
- Thêm các câu lệnh log vào ứng dụng và xem nhật ký trong bảng Logcat.
- Hiểu và làm việc với Activity và Intent, đồng thời thoải mái tương tác với chúng.

## **Những gì bạn sẽ học**

- Cách hoạt động của vòng đời Activity.
- Khi nào một Activity bắt đầu, tạm dừng, dừng và bị hủy.
- Các phương thức callback của vòng đời liên quan đến thay đổi Activity.
- Ảnh hưởng của các hành động (chẳng hạn như thay đổi cấu hình) dẫn đến các sự kiện vòng đời Activity.
- Cách bảo toàn trạng thái của Activity qua các sự kiện vòng đời.

## **Những gì bạn sẽ làm**

- Thêm mã vào ứng dụng TwoActivities từ bài thực hành trước để triển khai các phương thức callback của vòng đời Activity, bao gồm các câu lệnh ghi log.
- Quan sát sự thay đổi trạng thái khi ứng dụng chạy và khi bạn tương tác với mỗi Activity trong ứng dụng.
- Chỉnh sửa ứng dụng để giữ nguyên trạng thái của một Activity khi bị tạo lại ngoài ý muốn do hành vi của người dùng hoặc do thay đổi cấu hình trên thiết bị.

## **Nhiệm vụ 1: Thêm callback vòng đời vào TwoActivities**

Trong nhiệm vụ này, bạn sẽ triển khai tất cả các phương thức callback vòng đời của Activity để in thông báo ra logcat khi các phương thức này được gọi. Các thông báo log này sẽ giúp bạn thấy khi nào vòng đời của Activity thay đổi trạng thái và cách các thay đổi trạng thái đó ảnh hưởng đến ứng dụng của bạn khi chạy.

1. (Tùy chọn) Sao chép dự án TwoActivities

Đối với các nhiệm vụ trong bài thực hành này, bạn sẽ sửa đổi dự án TwoActivities hiện có mà bạn đã xây dựng trong bài thực hành trước. Nếu bạn muốn giữ nguyên dự án TwoActivities trước đó, hãy làm theo các bước trong Phụ lục: Tiện ích để sao chép dự án.

## 2. Triển khai callback vào MainActivity

- Mở dự án TwoActivities trong Android Studio và mở MainActivity trong Project > Android
- Trong phương thức onCreate(), thêm các câu lệnh log sau:

```
Log.d(LOG_TAG, "-----");  
Log.d(LOG_TAG, "onCreate");
```

- Ghi đè phương thức callback onStart(), với một câu lệnh log cho sự kiện này:

```
@Override  
public void onStart(){  
    super.onStart();  
    Log.d(LOG_TAG, "onStart");  
}
```

Để sử dụng phím tắt, chọn Code > Override Methods trong Android Studio. Một hộp thoại sẽ xuất hiện với tất cả các phương thức có thể ghi đè trong lớp của bạn. Chọn một hoặc nhiều phương thức callback từ danh sách này sẽ chèn một mẫu hoàn chỉnh cho các phương thức đó, bao gồm cả lời gọi bắt buộc đến lớp cha.

- Sử dụng phương thức onStart() làm mẫu để triển khai các phương thức callback onPause(), onRestart(), onResume(), onStop(), và onDestroy().  
Tất cả các phương thức callback có cùng chữ ký (trừ tên). Nếu bạn sao chép và dán onStart() để tạo các phương thức callback khác, đừng quên cập nhật nội dung để gọi đúng phương thức trong lớp cha và ghi log đúng phương thức.
- Chạy ứng dụng của bạn.
- Nhấp vào tab Logcat ở cuối Android Studio để hiển thị bảng Logcat.

## 3. Triển khai callback vòng đời trong SecondActivity

Bây giờ bạn đã triển khai các phương thức callback vòng đời cho MainActivity, hãy làm điều tương tự cho SecondActivity.

- Mở SecondActivity.
- Ở đầu lớp, thêm một hằng số cho biến LOG\_TAG.

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

- Thêm các phương thức callback vòng đời và câu lệnh log vào Activity thứ hai. (Bạn có thể sao chép và dán các phương thức callback từ MainActivity).
- Thêm một câu lệnh log vào phương thức returnReply() ngay trước phương thức finish().

```
Log.d(LOG_TAG, "End SecondActivity");
```

#### 4. Quan sát log khi ứng dụng chạy

- Chạy ứng dụng của bạn.
- Nhấp vào tab Logcat ở cuối Android Studio để hiển thị bảng Logcat.
- Nhập "Activity" vào hộp tìm kiếm.  
Logcat của Android có thể rất dài và lộn xộn. Vì biến LOG\_TAG trong mỗi lớp chứa MainActivity hoặc SecondActivity, bạn có thể sử dụng từ khóa này để lọc log chỉ hiển thị thông tin mà bạn quan tâm.

Thử nghiệm với ứng dụng của bạn và lưu ý các sự kiện vòng đời xảy ra khi thực hiện các hành động khác nhau. Đặc biệt, hãy thử:

- Sử dụng ứng dụng bình thường (gửi tin nhắn, trả lời bằng tin nhắn khác).
- Sử dụng nút Back để quay lại từ SecondActivity về MainActivity.
- Sử dụng mũi tên Up trên thanh ứng dụng để quay lại từ SecondActivity về MainActivity.
- Xoay thiết bị trên cả MainActivity và SecondActivity vào những thời điểm khác nhau trong ứng dụng và quan sát những gì xảy ra trong log và trên màn hình.
- Nhấn nút Overview (nút hình vuông bên phải nút Home) và đóng ứng dụng (nhấn vào dấu X).
- Quay lại màn hình chính và khởi động lại ứng dụng.

Mã code nhiệm vụ 1:

#### MainActivity

- Phương thức onCreate():

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mMessageEditText = findViewById(R.id.editText_main);
    mReplyHeadTextView = findViewById(R.id.text_header_reply);
    mReplyTextView = findViewById(R.id.text_message_reply);
```

```

    Log.d(LOG_TAG, "-----");
    Log.d(LOG_TAG, "onCreate");
}

```

- Các phương thức khác:

```

@Override
protected void onStart() {
    super.onStart();
    Log.d(LOG_TAG, "onStart");
}

@Override
protected void onPause() {
    super.onPause();
    Log.d(LOG_TAG, "onPause");
}

@Override
protected void onRestart() {
    super.onRestart();
    Log.d(LOG_TAG, "onRestart");
}

@Override
protected void onResume() {
    super.onResume();
    Log.d(LOG_TAG, "onResume");
}

@Override
protected void onStop() {
    super.onStop();
    Log.d(LOG_TAG, "onStop");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d(LOG_TAG, "onDestroy");
}

```

## SecondActivity

Các đoạn mã sau đây hiển thị mã đã được thêm vào SecondActivity, nhưng không phải toàn bộ lớp.



Ở đầu lớp SecondActivity:

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

Phương thức returnReply():

```
public void returnReply(View view) {  
    String reply = mReply.getText().toString();  
    Intent replyIntent = new Intent();  
    replyIntent.putExtra(EXTRA_REPLY, reply);  
    setResult(RESULT_OK, replyIntent);  
    Log.d(LOG_TAG, "End SecondActivity");  
    finish();  
}
```

Các phương thức khác: Tương tự như MainActivity ở trên.

## Nhiệm vụ 2: Lưu và khôi phục trạng thái phiên của Activity

Tùy thuộc vào tài nguyên hệ thống và hành vi của người dùng, mỗi Activity trong ứng dụng của bạn có thể bị hủy và tái tạo thường xuyên hơn bạn nghĩ.

Bạn có thể đã nhận thấy hành vi này trong phần trước khi xoay thiết bị hoặc trình giả lập. Xoay thiết bị là một ví dụ về thay đổi cấu hình thiết bị. Mặc dù xoay màn hình là trường hợp phổ biến nhất, nhưng tất cả các thay đổi cấu hình đều dẫn đến việc Activity hiện tại bị hủy và tái tạo như thể nó mới. Nếu bạn không xử lý hành vi này trong mã, khi xảy ra thay đổi cấu hình, giao diện của Activity có thể trở về trạng thái mặc định và người dùng có thể mất dữ liệu hoặc trạng thái tiến trình trong ứng dụng.

Trạng thái của mỗi Activity được lưu trữ dưới dạng tập hợp các cặp khóa/giá trị trong một đối tượng Bundle gọi là trạng thái phiên của Activity. Hệ thống sẽ tự động lưu thông tin trạng thái mặc định vào Bundle trước khi Activity bị dừng và truyền Bundle đó đến Activity mới để khôi phục.

Để tránh mất dữ liệu trong Activity khi nó bị hủy và tái tạo bất ngờ, bạn cần triển khai phương thức onSaveInstanceState(). Hệ thống gọi phương thức này trên Activity (giữa onPause() và onStop()) khi có khả năng Activity sẽ bị hủy và tái tạo.

Dữ liệu bạn lưu vào Bundle chỉ tồn tại trong phiên hiện tại của Activity. Khi bạn dừng và khởi động một phiên ứng dụng mới, trạng thái phiên của Activity sẽ bị mất và Activity sẽ trở về giao diện mặc định. Nếu bạn cần lưu dữ liệu người dùng giữa các phiên, hãy sử dụng SharedPreferences hoặc cơ sở dữ liệu (bạn sẽ học về điều này trong bài thực hành sau).

## 1. Lưu trạng thái phiên của Activity với onSaveInstanceState()

Bạn có thể nhận thấy rằng việc xoay thiết bị không ảnh hưởng đến trạng thái của Activity thứ hai. Điều này là do bố cục và trạng thái của Activity thứ hai được tạo từ layout và Intent đã kích hoạt nó. Ngay cả khi Activity được tạo lại, Intent vẫn tồn tại và dữ liệu trong Intent vẫn được sử dụng mỗi khi phương thức onCreate() của Activity thứ hai được gọi.

Ngoài ra, bạn có thể nhận thấy rằng trong mỗi Activity, bất kỳ văn bản nào bạn nhập vào các phần tử EditText (ô nhập liệu) đều được giữ lại ngay cả khi xoay thiết bị. Điều này xảy ra vì trạng thái của một số thành phần View trong bố cục được tự động lưu khi có sự thay đổi cấu hình, và giá trị hiện tại của một EditText là một trong những trường hợp đó.

Như vậy, trạng thái Activity mà bạn cần quan tâm là hai TextView trong Activity chính hiển thị tiêu đề phản hồi và nội dung phản hồi. Cả hai TextView này mặc định bị ẩn và chỉ xuất hiện khi bạn gửi một tin nhắn phản hồi từ Activity thứ hai về Activity chính.

Trong bước này, bạn sẽ thêm mã để bảo toàn trạng thái của hai TextView này bằng onSaveInstanceState().

- Mở MainActivity.
- Thêm phương thức onSaveInstanceState() vào Activity. Bạn có thể tự thêm hoặc sử dụng Code > Override Methods để tự động tạo phương thức này.

```
@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
}
```

- Kiểm tra xem tiêu đề phản hồi có đang hiển thị không. Nếu có, lưu trạng thái hiển thị vào Bundle bằng phương thức putBoolean() với khóa (key) "reply\_visible".

```
if (mReplyHeadTextView.getVisibility() == View.VISIBLE) {
    outState.putBoolean("reply_visible", true);
}
```

Hãy nhớ rằng tiêu đề phản hồi và nội dung phản hồi mặc định bị ẩn cho đến khi có phản hồi từ Activity thứ hai. Nếu tiêu đề hiển thị, điều đó có nghĩa là có dữ liệu phản hồi cần được lưu. Lưu ý: Bạn không cần lưu nội dung của tiêu đề vì nội dung của tiêu đề không thay đổi.

- Bên trong cùng một điều kiện kiểm tra đó, lưu nội dung phản hồi (TextView chứa phản hồi) vào Bundle.

```
outState.putString("reply_text", mReplyTextView.getText().toString());
```

Nếu tiêu đề hiển thị, có thể chắc chắn rằng nội dung phản hồi cũng hiển thị. Vì vậy, bạn chỉ cần lưu nội dung văn bản vào Bundle bằng khóa "reply\_text".

Bạn chỉ cần lưu trạng thái của các View có thể thay đổi sau khi Activity được tạo. Các View khác trong ứng dụng (chẳng hạn như EditText, Button) có thể được khởi tạo lại từ layout mặc định bất cứ lúc nào.

Lưu ý rằng hệ thống sẽ tự động lưu trạng thái của một số phần tử View, chẳng hạn như nội dung của EditText.

## 2. Khôi phục trạng thái phiên của Activity trong onCreate()

Sau khi lưu trạng thái của Activity, bạn cần khôi phục nó khi Activity được tạo lại. Bạn có thể thực hiện điều này trong onCreate() hoặc sử dụng phương thức onRestoreInstanceState(), phương thức này sẽ được gọi sau onStart() sau khi Activity được tạo.

Thông thường, cách tốt nhất để khôi phục trạng thái Activity là trong onCreate(), vì điều này đảm bảo rằng giao diện người dùng (UI) được hiển thị ngay lập tức với trạng thái đã lưu. Đôi khi, có thể thuận tiện hơn khi thực hiện trong onRestoreInstanceState() để đảm bảo rằng tất cả các khởi tạo khác đã hoàn tất, hoặc để cho phép các lớp con quyết định có sử dụng triển khai mặc định hay không.

- Trong phương thức onCreate(), sau khi các biến View được khởi tạo bằng findViewById(), hãy thêm một bài kiểm tra để đảm bảo rằng savedInstanceState không phải là null.

```
mMessageEditText = findViewById(R.id.editText_main);
mReplyHeadTextView = findViewById(R.id.text_header_reply);
mReplyTextView = findViewById(R.id.text_message_reply);

if (savedInstanceState != null) {
}
```

Khi Activity của bạn được tạo, hệ thống sẽ truyền trạng thái Bundle cho onCreate() làm đối số duy nhất của nó. Lần đầu tiên onCreate() được gọi và ứng dụng của bạn khởi động, Bundle sẽ là null—không có trạng thái nào hiện có khi ứng dụng của bạn khởi động lần đầu tiên. Các lệnh gọi tiếp theo đến onCreate() có một bundle được điền dữ liệu bạn đã lưu trữ trong onSaveInstanceState().

- Bên trong kiểm tra đó, lấy khả năng hiển thị hiện tại (đúng hoặc sai) ra khỏi Bundle với khóa "reply\_visible".

```
if (savedInstanceState != null) {
    boolean isVisible = savedInstanceState.getBoolean("reply_visible");
}
```

- Thêm một bài kiểm tra bên dưới dòng trước đó cho biến isVisible. Nếu có khóa reply\_visible trong Bundle trạng thái (và do đó isVisible là đúng), bạn sẽ cần khôi phục trạng thái.

```
if (isVisible) {
}
```

- Bên trong isVisible test, hãy làm cho tiêu đề hiển thị.

```
mReplyHeadTextView.setVisibility(View.VISIBLE);
```

- Lấy tin nhắn trả lời văn bản từ Bundle với khóa "reply\_text" và đặt TextView trả lời để hiển thị chuỗi đó

```
mReplyTextView.setText(savedInstanceState.getString("reply_text"));
```

- Làm cho TextView trả lời cũng hiển thị

```
mReplyTextView.setVisibility(View.VISIBLE);
```

- Chạy ứng dụng. Thử xoay thiết bị hoặc trình giả lập để đảm bảo rằng tin nhắn trả lời (nếu có) vẫn hiển thị trên màn hình sau khi Hoạt động được tạo lại

Mã code nhiệm vụ 2

MainActivity

- Phương thức onSaveInstanceState():

```
@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    if (mReplyHeadTextView.getVisibility() == View.VISIBLE) {
        outState.putBoolean("reply_visible", true);
        outState.putString("reply_text",
            mReplyTextView.getText().toString());
    }
}
```

- Phương thức onCreate():

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mMessageEditText = findViewById(R.id.editText_main);
    mReplyHeadTextView = findViewById(R.id.text_header_reply);
    mReplyTextView = findViewById(R.id.text_message_reply);

    Log.d(LOG_TAG, "-----");
    Log.d(LOG_TAG, "onCreate");

    if (savedInstanceState != null) {
        boolean isVisible = savedInstanceState.getBoolean("reply_visible");
        if (isVisible) {
            mReplyHeadTextView.setVisibility(View.VISIBLE);
            mReplyTextView.setText(savedInstanceState.getString("reply_text"));
            mReplyTextView.setVisibility(View.VISIBLE);
        }
    }
}

```

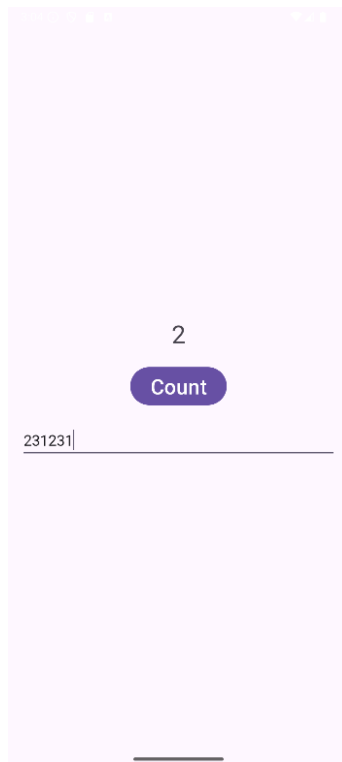
## Tổng kết

- Vòng đời của Activity là tập hợp các trạng thái mà một Activity trải qua, bắt đầu từ khi nó được tạo và kết thúc khi hệ thống Android thu hồi tài nguyên của Activity đó.
- Khi người dùng điều hướng giữa các Activity trong và ngoài ứng dụng, mỗi Activity sẽ di chuyển giữa các trạng thái trong vòng đời của nó.
- Mỗi trạng thái trong vòng đời Activity có một phương thức gọi lại (callback method) tương ứng mà bạn có thể ghi đè trong lớp Activity của mình.
- Các phương thức vòng đời bao gồm: onCreate(), onStart(), onPause(), onRestart(), onResume(), onStop(), onDestroy().
- Việc ghi đè các phương thức vòng đời cho phép bạn thêm hành vi khi Activity chuyển sang trạng thái tương ứng.
- Bạn có thể thêm các phương thức ghi đè skeleton vào lớp trong Android Studio bằng cách vào Code > Override.
- Thay đổi cấu hình thiết bị (chẳng hạn như xoay màn hình) sẽ khiến Activity bị hủy và tạo lại như thể nó mới được khởi tạo.
- Một phần trạng thái của Activity được bảo toàn khi có thay đổi cấu hình, bao gồm giá trị hiện tại của các phần tử EditText. Đối với các dữ liệu khác, bạn cần lưu trữ thủ công.
- Lưu trạng thái Activity trong phương thức onSaveInstanceState().

- Dữ liệu trạng thái phiên hoạt động (instance state) được lưu dưới dạng cặp key/value trong một Bundle. Sử dụng các phương thức của Bundle để lưu và lấy dữ liệu.
- Khôi phục trạng thái phiên hoạt động trong onCreate(), đây là cách ưu tiên, hoặc trong onRestoreInstanceState().

## Xây dựng và chạy ứng dụng

- Tạo một ứng dụng với bố cục chứa một TextView để hiển thị bộ đếm, một Button để tăng bộ đếm và một EditText. Xem ảnh minh họa bên dưới để tham khảo. Bạn không cần sao chép chính xác bố cục.
- Thêm sự kiện click cho Button để tăng giá trị bộ đếm.
- Chạy ứng dụng, tăng giá trị bộ đếm và nhập văn bản vào EditText.
- Xoay thiết bị và quan sát: giá trị bộ đếm sẽ bị đặt lại, nhưng nội dung EditText vẫn giữ nguyên.
- Cài đặt phương thức onSaveInstanceState() để lưu trạng thái hiện tại của ứng dụng.
- Cập nhật onCreate() để khôi phục trạng thái của ứng dụng.
- Kiểm tra lại: khi xoay thiết bị, trạng thái của ứng dụng phải được bảo toàn.



## 2.3) Intent ngầm định

### Giới thiệu

Trong phần trước, bạn đã tìm hiểu về explicit intent. Với một explicit intent, bạn thực hiện một hoạt động trong ứng dụng của mình hoặc một ứng dụng khác bằng cách gửi một intent có tên lớp đầy đủ của activity. Trong phần này, bạn sẽ tìm hiểu thêm về implicit intent và cách sử dụng chúng để thực hiện các hoạt động.

Với implicit intent, bạn khởi tạo một hoạt động mà không cần biết ứng dụng hoặc activity nào sẽ xử lý tác vụ đó. Ví dụ, nếu bạn muốn ứng dụng của mình chụp ảnh, gửi email hoặc hiển thị vị trí trên bản đồ, bạn thường không quan tâm ứng dụng nào thực hiện tác vụ đó.

Ngược lại, một activity có thể khai báo một hoặc nhiều intent filter trong tệp AndroidManifest.xml để thông báo rằng nó có thể chấp nhận implicit intent và xác định các loại intent mà nó có thể xử lý.

Để khớp yêu cầu của bạn với một ứng dụng đã cài đặt trên thiết bị, hệ thống Android sẽ tìm kiếm implicit intent khớp với một activity có intent filter tương ứng. Nếu có nhiều ứng dụng phù hợp, người dùng sẽ được hiển thị một trình chọn ứng dụng để chọn ứng dụng xử lý intent.

Trong bài thực hành này, bạn sẽ xây dựng một ứng dụng gửi implicit intent để thực hiện các tác vụ sau:

- Mở một URL trong trình duyệt web.
- Mở một vị trí trên bản đồ.
- Chia sẻ văn bản.

Chia sẻ – gửi một phần thông tin đến người khác qua email hoặc mạng xã hội – là một tính năng phổ biến trong nhiều ứng dụng. Để thực hiện hành động chia sẻ, bạn sử dụng lớp `ShareCompat.IntentBuilder`, giúp dễ dàng tạo implicit intent để chia sẻ dữ liệu.

Cuối cùng, bạn sẽ tạo một trình nhận intent đơn giản để chấp nhận implicit intent cho một hành động cụ thể.

### Những gì bạn cần biết trước

Bạn cần có khả năng:

- Sử dụng trình chỉnh sửa bố cục (layout editor) để chỉnh sửa giao diện.
- Chỉnh sửa mã XML của bố cục.
- Thêm Button và trình xử lý sự kiện click.

- Tạo và sử dụng một Activity.
- Tạo và gửi một Intent giữa các Activity.

## Những gì bạn sẽ học

- Cách tạo một implicit intent, sử dụng các hành động và danh mục của nó.
- Cách sử dụng lớp hỗ trợ `ShareCompat.IntentBuilder` để tạo một implicit intent chia sẻ dữ liệu.
- Cách khai báo rằng ứng dụng của bạn có thể chấp nhận implicit intent bằng cách thêm intent filter vào tệp `AndroidManifest.xml`.

## Những gì bạn sẽ làm

- Tạo một ứng dụng mới để thực nghiệm với implicit intent.
- Triển khai một implicit intent để mở một trang web và một intent khác để mở vị trí trên bản đồ.
- Triển khai một hành động chia sẻ một đoạn văn bản.
- Tạo một ứng dụng mới có thể chấp nhận implicit intent để mở một trang web.

## Nhiệm vụ 1: Tạo dự án và bố cục

Trong nhiệm vụ này, bạn sẽ tạo một dự án mới và ứng dụng có tên Implicit Intents, đồng thời thiết lập bố cục cho ứng dụng.

1. Tạo dự án
  - Khởi động Android Studio và tạo một dự án Android Studio mới. Đặt tên ứng dụng là Implicit Intents.
  - Chọn Empty Views Activity làm mẫu dự án. Nhấn Next.
  - Giữ nguyên tên Activity mặc định (`MainActivity`). Đảm bảo hộp kiểm Generate Layout file được chọn. Nhấn Finish.
2. Tạo bố cục

Trong bước này, bạn sẽ tạo bố cục cho ứng dụng. Sử dụng `LinearLayout`, ba `Button` và ba `EditText`, như sau:

- Mở `app > res > values > strings.xml` trong Project > Android pane và thêm các tài nguyên chuỗi cần thiết.

```
<resources>
  <string name="app_name">Implicit Intents</string>
  <string name="edittext_uri">http://developer.android.com</string>
  <string name="button_uri">Open Website</string>
  <string name="edittext_loc">Golden Gate Bridge</string>
```



```
<string name="button_loc">Open Location</string>
<string name="edittext_share">\'Twas brillig and the slithy toves</string>
<string name="button_share">Share This Text</string>
</resources>
```

- Mở res > layout > activity\_main.xml trong Project > Android. Nhấn vào tab Text để chuyển sang chế độ mã XML.
- Thay đổi android.support.constraint.ConstraintLayout thành LinearLayout.
- Thêm thuộc tính android:orientation với giá trị "vertical", và thuộc tính android:padding với giá trị "16dp".
- Xóa TextView hiển thị "Hello World".
- Thêm một nhóm phần tử giao diện người dùng cho nút Open Website, bao gồm một EditText và một Button. Sử dụng các giá trị thuộc tính thích hợp.

Thuộc tính EditText	Giá trị
android:id	"@+id/website_edittext"
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:text	"@string/edittext_uri"
Thuộc tính Button	Giá trị
android:id	"@+id/open_website_button"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginBottom	"24dp"
android:text	"@string/button_uri"
android:onClick	"openWebsite"

Thuộc tính android:onClick của Button sẽ bị gạch đỏ cho đến khi bạn định nghĩa phương thức callback trong bước tiếp theo.

- Thêm một nhóm phần tử giao diện (EditText và Button) cho nút Open Location. Sử dụng cùng thuộc tính như bước trước, nhưng điều chỉnh giá trị phù hợp.

Thuộc tính EditText	Giá trị
android:id	"@+id/location_edittext"
android:text	"@string/edittext_loc"
Thuộc tính Button	Giá trị
android:id	"@+id/open_location_button"

android:text	"@string/button_loc"
android:onClick	"openLocation"

Thuộc tính android:onClick vẫn sẽ bị gạch đỏ cho đến khi bạn định nghĩa phương thức callback.

- Thêm một nhóm phần tử giao diện (EditText và Button) cho nút Share This. Sao chép các giá trị từ nút Open Website và chỉnh sửa theo yêu cầu.

Thuộc tính EditText	Giá trị
android:id	"@+id/share_edittext"
android:text	"@string/edittext_share"
Thuộc tính Button	Giá trị
android:id	"@+id/share_text_button"
android:text	"@string/button_share"
android:onClick	"shareText"

Tùy thuộc vào phiên bản Android Studio của bạn, mã activity\_main.xml sẽ trông giống như ví dụ sau. Các giá trị của thuộc tính android:onClick sẽ tiếp tục bị gạch đỏ cho đến khi bạn định nghĩa các phương thức callback trong bước tiếp theo.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/website_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/edittext_uri"/>
    <Button
        android:id="@+id/open_website_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="24dp"
        android:text="@string/button_uri"
        android:onClick="openWebsite"/>
```

```

<EditText
    android:id="@+id/location_edittext"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/edittext_uri"/>
<Button
    android:id="@+id/open_location_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="@string/button_loc"
    android:onClick="openLocation"/>
<EditText
    android:id="@+id/share_edittext"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/edittext_share"/>
<Button
    android:id="@+id/share_text_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="24dp"
    android:text="@string/button_share"
    android:onClick="shareText"/>

</LinearLayout>

```

## Nhiệm vụ 2: Triển khai nút Open Website

Trong bài tập này, bạn sẽ triển khai phương thức xử lý sự kiện khi nhấn nút Open Website. Hành động này sử dụng một Implicit Intent để gửi URI được nhập đến một Activity có thể xử lý Intent đó (chẳng hạn như trình duyệt web).

### 1. Định nghĩa phương thức openWebsite()

- Nhấn vào "openWebsite" trong mã XML của activity\_main.xml.
  - Nhấn Alt + Enter (Option + Enter trên Mac) và chọn Create 'openWebsite(View)' in 'MainActivity'.
- MainActivity.java sẽ mở ra và Android Studio sẽ tự động tạo một phương thức openWebsite() trống.
- Trong MainActivity, thêm một biến private ở đầu lớp để giữ đối tượng EditText chứa URI của trang web.

```
private EditText mWebsiteEditText;
```

- Trong phương thức onCreate() của MainActivity, sử dụng findViewById() để tham chiếu đến EditText và gán vào biến private vừa tạo.

```
protected void onCreate(Bundle savedInstanceState){
    mWebsiteEditText = findViewById(R.id.website_edittext);
}
```

## 2. Thêm mã vào phương thức openWebsite()

- Thêm câu lệnh lấy giá trị chuỗi từ EditText.

```
String url = mWebsiteEditText.getText().toString();
```

- Mã hóa và phân tích chuỗi đó thành một đối tượng Uri.

```
Uri webpage = Uri.parse(url);
```

- Tạo một Intent mới với Intent.ACTION\_VIEW làm hành động và Uri làm dữ liệu:

```
Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
```

- Bộ khởi tạo Intent này khác với bộ khởi tạo được sử dụng để tạo Intent tường minh (explicit Intent). Trong bộ khởi tạo trước, bạn chỉ định context hiện tại và một thành phần cụ thể (lớp Activity) để gửi Intent. Trong bộ khởi tạo này, bạn chỉ định một hành động (action) và dữ liệu (data) cho hành động đó. Các hành động được xác định bởi lớp Intent và có thể bao gồm ACTION\_VIEW (để xem dữ liệu đã cho), ACTION\_EDIT (để chỉnh sửa dữ liệu), hoặc ACTION\_DIAL (để quay số điện thoại). Trong trường hợp này, hành động là ACTION\_VIEW vì bạn muốn hiển thị trang web được chỉ định bởi biến webpage.
- Sử dụng phương thức resolveActivity() kết hợp với Android Package Manager để kiểm tra xem có Activity nào có thể xử lý Intent này không.

```
if (intent.resolveActivity(getPackageManager()) != null) {
}
```

- Yêu cầu này sẽ khớp hành động và dữ liệu trong Intent của bạn với các bộ lọc Intent của các ứng dụng đã cài đặt trên thiết bị. Bạn sử dụng nó để đảm bảo có ít nhất một Activity có thể xử lý yêu cầu của bạn.
- Bên trong câu lệnh if, gọi startActivity() để gửi Intent.

```
startActivity(intent);
```

- Thêm một khối else để ghi lại thông báo Log nếu Intent không thể được xử lý.

```
else {  
    Log.d("ImplicitIntents", "Can't handle this intent!");  
}
```

Phương thức `openWebsite()` bây giờ sẽ trông giống như sau (các chú thích được thêm vào để làm rõ):

```
public void openWebsite(View view) {  
    String url = mWebsiteEditText.getText().toString();  
    Uri webpage = Uri.parse(url);  
    Intent intent = new Intent(Intent.ACTION_VIEW, webpage);  
    if (intent.resolveActivity(getPackageManager()) != null) {  
        startActivity(intent);  
    } else {  
        Log.d("ImplicitIntents", "Can't handle this intent!");  
    }  
}
```

### Nhiệm vụ 3: Triển khai nút Open Location

Trong nhiệm vụ này, bạn sẽ triển khai phương thức xử lý sự kiện khi nhấn vào nút thứ hai trong giao diện người dùng, Open Location. Phương thức này gần như giống hệt với phương thức `openWebsite()`. Điểm khác biệt là sử dụng URI geo để chỉ định vị trí bản đồ. Bạn có thể sử dụng URI geo với vĩ độ và kinh độ hoặc sử dụng chuỗi truy vấn để tìm kiếm vị trí chung. Trong ví dụ này, chúng ta sử dụng phương thức thứ hai.

#### 1. Định nghĩa `openLocation()`

- Nhập vào "openLocation" trong mã XML của `activity_main.xml`.
- Nhấn Alt+Enter (hoặc Option+Enter trên Mac) và chọn Create 'openLocation(View)' in MainActivity.  
Android Studio sẽ tạo một phương thức khung trong MainActivity cho trình xử lý `openLocation()`.
- Thêm một biến riêng tư ở đầu MainActivity để giữ đối tượng EditText cho URI vị trí.

```
private EditText mLocationEditText;
```

- Trong phương thức `onCreate()`, sử dụng `findViewById()` để lấy tham chiếu đến EditText và gán nó vào biến riêng tư đó.

```
mLocationEditText = findViewById(R.id.location_edittext);
```

#### 2. Thêm mã vào `openLocation()`

- Trong phương thức `openLocation()` mới, thêm một câu lệnh để lấy giá trị chuỗi của `EditText` chứa vị trí (`mLocationEditText`).

```
String loc = mLocationEditText.getText().toString();
```

- Phân tích chuỗi đó thành một đối tượng `Uri` với truy vấn tìm kiếm bản đồ:

```
Uri addressUri = Uri.parse("geo:0,0?q=" + loc);
```

- Tạo một `Intent` mới với hành động `Intent.ACTION_VIEW` và dữ liệu là `loc`.

```
Intent intent = new Intent(Intent.ACTION_VIEW, addressUri);
```

- Xác định `Intent` và kiểm tra xem `Intent` có được xử lý thành công hay không. Nếu có, gọi `startActivity()`, nếu không, ghi lại một thông báo lỗi.

```
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
} else {
    Log.d("ImplicitIntents", "Can't handle this intent!");
}
```

Phương thức `openLocation()` giờ sẽ trông giống như sau (các chú thích được thêm để làm rõ):

```
public void openLocation(View view) {
    String loc = mLocationEditText.getText().toString();
    Uri addressUri = Uri.parse("geo:0,0?q=" + loc);
    Intent intent = new Intent(Intent.ACTION_VIEW, addressUri);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    } else {
        Log.d("ImplicitIntents", "Can't handle this intent!");
    }
}
```

## Nhiệm vụ 4: Triển khai nút Share This Text

Một hành động chia sẻ là cách dễ dàng để người dùng chia sẻ nội dung trong ứng dụng với mạng xã hội và các ứng dụng khác. Mặc dù bạn có thể tự xây dựng chức năng chia sẻ trong ứng dụng bằng cách sử dụng `Intent` ngầm định, Android cung cấp lớp trợ giúp `ShareCompat.IntentBuilder` để làm cho việc triển khai tính năng chia sẻ trở nên dễ dàng hơn. Bạn có thể sử dụng `ShareCompat.IntentBuilder` để tạo một `Intent` và khởi chạy hộp thoại lựa chọn ứng dụng đích để chia sẻ.

## 1. Định nghĩa shareText()

- Nhấp vào "shareText" trong mã XML của activity\_main.xml.
- Nhấn Alt+Enter (hoặc Option+Enter trên Mac) và chọn Create 'shareText(View)' in MainActivity.  
Android Studio sẽ tạo một phương thức khung trong MainActivity cho trình xử lý shareText().
- Thêm một biến riêng tư ở đầu MainActivity để giữ đối tượng EditText.

```
private EditText mShareTextEditText;
```

- Trong onCreate(), sử dụng findViewById() để lấy tham chiếu đến EditText và gán nó vào biến riêng tư đó.

```
mShareTextEditText = findViewById(R.id.share_edittext);
```

## 2. Thêm mã vào shareText()

- Trong phương thức shareText() mới, thêm một câu lệnh để lấy giá trị chuỗi của EditText (mShareTextEditText).

```
String txt = mShareTextEditText.getText().toString();
```

- Xác định kiểu MIME của văn bản cần chia sẻ:

```
String mimeType = "text/plain";
```

- Gọi ShareCompat.IntentBuilder với các phương thức sau:
- Trích xuất giá trị của .setChooserTitle thành một tài nguyên chuỗi.

Lệnh gọi ShareCompat.IntentBuilder sử dụng các phương thức sau:

Phương thức	Mô tả
from()	Hoạt động khởi chạy Intent chia sẻ này
setType()	Kiểu MIME của mục được chia sẻ
setChooserTitle()	Tiêu đề xuất hiện trên trình chọn ứng dụng hệ thống.
setText()	Văn bản thực tế được chia sẻ
startChooser()	Hiển thị trình chọn ứng dụng hệ thống và gửi Intent.

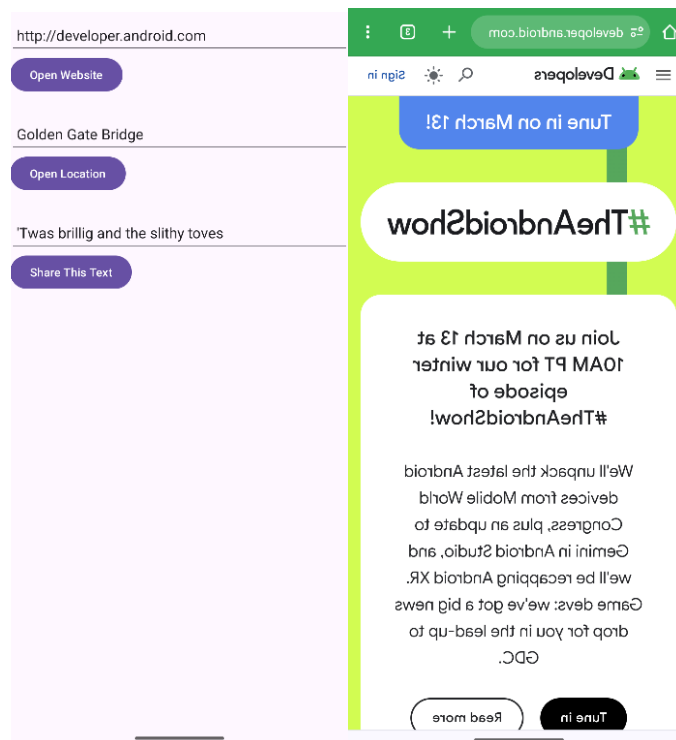
Cấu trúc này, với tất cả các phương thức thiết lập của trình tạo (builder) được liên kết trong một câu lệnh, là một cách dễ dàng để tạo và khởi chạy Intent. Bạn có thể thêm bất kỳ phương thức bổ sung nào vào danh sách này.

Phương thức `shareText()` giờ sẽ trông như sau:

```
public void shareText(View view) {
    String txt = mShareTextEditText.getText().toString();
    String mimeType = "text/plain";
    ShareCompat.IntentBuilder
        .from(this)
        .setType(mimeType)
        .setChooserTitle("Share this text with: ")
        .setText(txt)
        .startChooser();
}
```

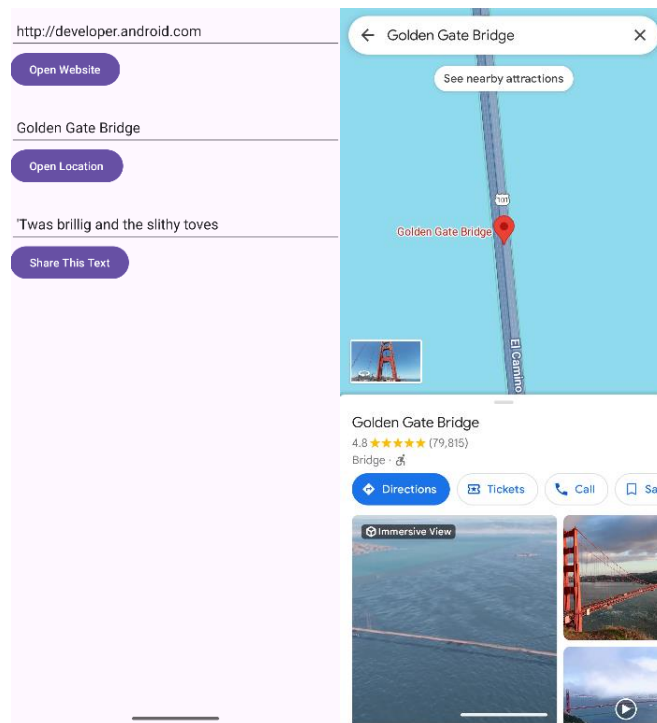
### 3. Chạy ứng dụng

- Chạy ứng dụng.
- Nhấn nút Open Website để mở trình duyệt với URL trang web trong EditText phía trên nút. Trình duyệt và trang web sẽ hiển thị như hình bên dưới.

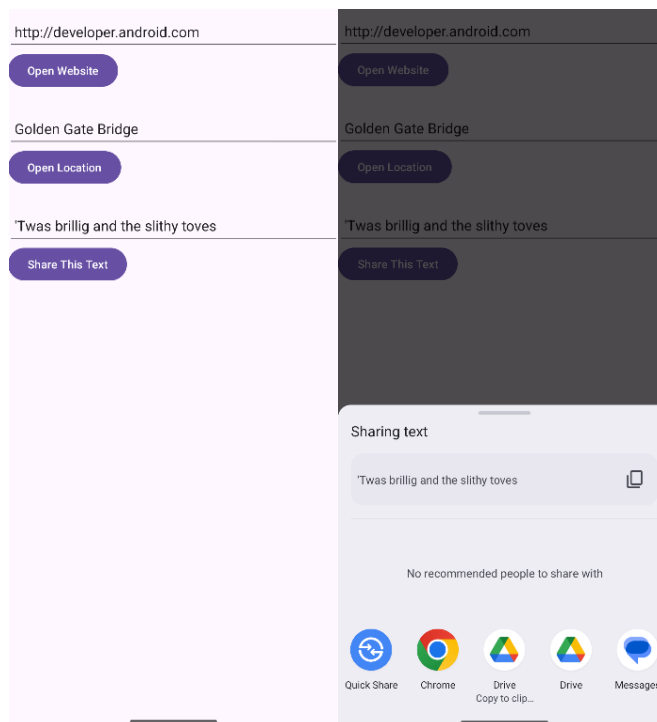




- Nhấn nút Open Location để mở bản đồ với vị trí trong EditText phía trên nút. Bản đồ với vị trí sẽ hiển thị như hình bên dưới.



- Nhấn nút Share This Text để mở hộp thoại với các tùy chọn chia sẻ văn bản. Hộp thoại với các lựa chọn sẽ hiển thị như hình bên dưới.



## Nhiệm vụ 5: Nhận một Intent ẩn danh

Cho đến nay, bạn đã tạo một ứng dụng sử dụng Intent ẩn danh để khởi chạy Activity của một ứng dụng khác. Trong nhiệm vụ này, bạn sẽ xem xét vấn đề theo hướng ngược lại: cho phép một Activity trong ứng dụng của bạn phản hồi một Intent ẩn danh được gửi từ một ứng dụng khác.

Một Activity trong ứng dụng của bạn luôn có thể được kích hoạt từ bên trong hoặc bên ngoài ứng dụng của bạn bằng một Intent tường minh. Để cho phép một Activity nhận Intent ẩn danh, bạn phải xác định một Intent filter trong tệp `AndroidManifest.xml` của ứng dụng để chỉ ra loại Intent ẩn danh mà Activity quan tâm xử lý.

Để khớp yêu cầu của bạn với một ứng dụng cụ thể đã được cài đặt trên thiết bị, hệ thống Android sẽ khớp Intent ẩn danh của bạn với một Activity có Intent filter chỉ ra rằng chúng có thể thực hiện hành động đó. Nếu có nhiều ứng dụng được cài đặt khớp với yêu cầu, người dùng sẽ được hiển thị một trình chọn ứng dụng để chọn ứng dụng họ muốn sử dụng để xử lý Intent đó.

Khi một ứng dụng trên thiết bị gửi một Intent ẩn danh, hệ thống Android sẽ khớp hành động và dữ liệu của Intent đó với bất kỳ Activity nào có sẵn bao gồm các Intent filter phù hợp. Khi các Intent filter của một Activity khớp với Intent:

- Nếu chỉ có một Activity phù hợp, Android sẽ cho phép Activity tự xử lý Intent.
- Nếu có nhiều Activity phù hợp, Android sẽ hiển thị một trình chọn ứng dụng để cho phép người dùng chọn ứng dụng họ muốn sử dụng để thực hiện hành động đó.

Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng rất đơn giản để nhận một Intent ẩn danh nhằm mở URI của một trang web. Khi được kích hoạt bởi một Intent ẩn danh, ứng dụng đó sẽ hiển thị URI được yêu cầu dưới dạng chuỗi trong một TextView.

### 1. Tạo dự án và bố cục

- Tạo một dự án Android Studio mới với tên ứng dụng là `Implicit Intents Receiver` và chọn `Empty Views Activity` làm mẫu dự án.
- Chấp nhận tên Activity mặc định (`MainActivity`). Nhấp vào `Next`.
- Đảm bảo rằng hộp `Generate Layout file` được chọn. Nhấp vào `Finish`.
- Mở `activity_main.xml`.
- Trong TextView hiện có ("`Hello World`"), xóa thuộc tính `android:text`. Không có văn bản nào trong TextView này theo mặc định, nhưng bạn sẽ thêm URI từ Intent trong `onCreate()`.
- Giữ nguyên các thuộc tính `layout_constraint`, nhưng thêm các thuộc tính sau:

Thuộc tính	Giá trị
android:id	"@+id/text_uri_message"
android:textSize	"18sp"
android:textStyle	"bold"

## 2. Sửa đổi AndroidManifest.xml để thêm Intent filter

- Mở tệp AndroidManifest.xml.
- Lưu ý rằng MainActivity đã có Intent filter sau:

```
<intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

- Intent filter này, là một phần của tệp manifest mặc định của dự án, cho biết rằng Activity này là điểm nhập chính của ứng dụng (với hành động Intent là "android.intent.action.MAIN") và rằng Activity này sẽ xuất hiện dưới dạng một mục cấp cao nhất trong trình khởi chạy (với danh mục là "android.intent.category.LAUNCHER").
- Thêm một thẻ <intent-filter> thứ hai bên trong <activity> và bao gồm các phần tử sau:

```
<intent-filter>
  <action android:name="android.intent.action.VIEW" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  <data android:scheme="http"
        android:host="developer.android.com" />
</intent-filter>
```

- Những dòng này xác định một Intent filter cho Activity, tức là loại Intent mà Activity có thể xử lý. Intent filter này khai báo các phần tử sau:

Loại bộ lọc	Giá trị	Ý nghĩa
action	"android.intent.action.VIEW"	Bất kỳ Intent nào có hành động xem (View).
category	"android.intent.category.DEFAULT"	Bất kỳ Intent nào là implicit (không chỉ định rõ component). Danh mục này phải được bao gồm để Activity của bạn có thể nhận bất kỳ Intent implicit nào.

category	"android.intent.category.BROWSABLE"	Yêu cầu mở các liên kết duyệt web từ trang web, email hoặc các nguồn khác.
data	android:scheme="http" android:host="developer.android.com"	Các URI có scheme là http và host là developer.android.com.

- Lưu ý rằng bộ lọc dữ liệu có một giới hạn đối với loại liên kết mà nó sẽ chấp nhận và tên máy chủ cho các URI đó. Nếu bạn muốn ứng dụng nhận Intent từ bất kỳ liên kết nào, bạn có thể bỏ qua phần tử <data>.

Phần application của AndroidManifest.xml bây giờ sẽ trông như sau:

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.ImplicitIntentsReceiver"
    tools:targetApi="31">
    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT" />
            <category android:name="android.intent.category.BROWSABLE" />
            <data android:scheme="http"
                android:host="developer.android.com" />
        </intent-filter>
    </activity>
</application>
```

### 3. Xử lý Intent

Trong phương thức onCreate() của Activity, xử lý Intent đến để lấy dữ liệu hoặc các extras mà Intent đó bao gồm. Trong trường hợp này, Intent ẩn danh đến có URI được lưu trữ trong dữ liệu của Intent.

- Mở MainActivity.
- Trong phương thức onCreate(), lấy Intent đến được sử dụng để kích hoạt Activity.

```
Intent intent = getIntent();
```

- Lấy dữ liệu từ Intent. Dữ liệu Intent luôn là một đối tượng URI.

```
Uri uri = intent.getData();
```

- Kiểm tra xem biến uri có null hay không. Nếu kiểm tra này thành công, tạo một chuỗi từ đối tượng URI:

```
if (uri != null) {  
    String uri_string = "URI: " + uri.toString();  
}
```

- Trích xuất phần "URI: " từ trên thành một tài nguyên chuỗi (uri\_label).
- Bên trong cùng một khối if, lấy TextView để hiển thị thông báo.

```
TextView textView = findViewById(R.id.text_uri_message);
```

- Cũng bên trong khối if, đặt văn bản của TextView thành URI.

```
textView.setText(uri_string);
```

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

#### 4. Chạy cả hai ứng dụng

Để hiển thị kết quả của việc nhận một Intent ẩn danh, bạn sẽ chạy cả ứng dụng Implicit Intents Receiver và Implicit Intents trên trình giả lập hoặc thiết bị của bạn.

- Chạy ứng dụng Implicit Intents Receiver.  
Chạy ứng dụng riêng lẻ sẽ hiển thị một Activity trống mà không có văn bản nào. Điều này là do Activity đã được kích hoạt từ trình khởi chạy hệ thống, chứ không phải bằng một Intent từ một ứng dụng khác.
- Chạy ứng dụng Implicit Intents và nhấp vào nút Open Website với URI mặc định.  
Một trình chọn ứng dụng xuất hiện hỏi bạn có muốn sử dụng trình duyệt mặc định (Chrome trong ví dụ dưới) hoặc ứng dụng Implicit Intents Receiver. Chọn Implicit

Intents Receiver, sau đó nhấp vào Just Once. Ứng dụng Implicit Intents Receiver sẽ khởi chạy và hiển thị URI từ yêu cầu ban đầu.

- Nhấn nút Back, nhập một URI khác, rồi nhấp vào Open Website. Ứng dụng nhận có một Intent filter rất hạn chế, chỉ khớp chính xác với giao thức URI (http) và máy chủ (developer.android.com). Bất kỳ URI nào khác sẽ mở trình duyệt web mặc định.

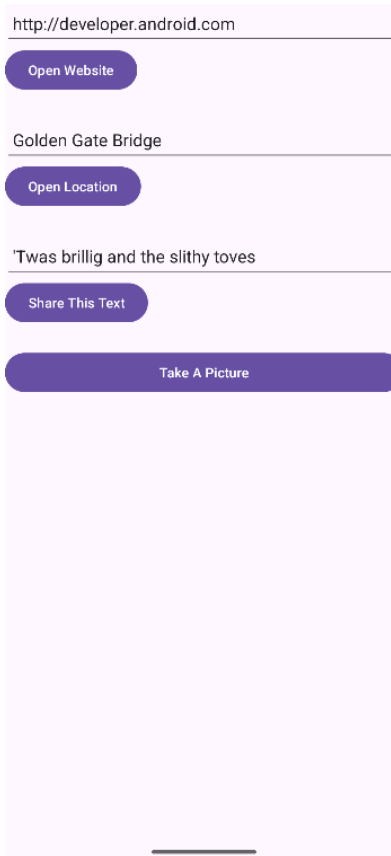
## Tổng kết

- Một Intent ẩn danh cho phép bạn kích hoạt một Activity nếu bạn biết hành động, nhưng không biết chính xác ứng dụng hoặc Activity nào sẽ xử lý hành động đó.
- Một Activity có thể nhận Intent ẩn danh phải xác định các Intent filter trong tệp AndroidManifest.xml để khớp với một hoặc nhiều hành động và danh mục Intent.
- Hệ thống Android sẽ khớp nội dung của một Intent ẩn danh với các Intent filter của bất kỳ Activity nào có sẵn để xác định Activity nào sẽ được kích hoạt. Nếu có nhiều Activity phù hợp, hệ thống sẽ cung cấp một trình chọn ứng dụng để người dùng chọn.
- Lớp `ShareCompat.IntentBuilder` giúp dễ dàng tạo một Intent ẩn danh để chia sẻ dữ liệu lên mạng xã hội hoặc email.

## Xây dựng và chạy ứng dụng

Mở ứng dụng `ImplicitIntents` mà bạn đã tạo.

- Thêm một nút khác ở cuối màn hình.
- Khi nút được nhấp, hãy khởi chạy ứng dụng máy ảnh để chụp ảnh. (Bạn không cần trả về ảnh cho ứng dụng gốc.)



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

### 3.1) Trình gỡ lỗi

#### Giới thiệu

Trong các bài thực hành trước, bạn đã sử dụng lớp Log để in thông tin ra nhật ký hệ thống, hiển thị trong ngăn Logcat của Android Studio khi ứng dụng chạy. Việc thêm các câu lệnh ghi nhật ký vào ứng dụng là một cách để tìm lỗi và cải thiện hoạt động của ứng dụng. Một cách khác là sử dụng trình gỡ lỗi tích hợp trong Android Studio.

Trong bài thực hành này, bạn sẽ học cách gỡ lỗi ứng dụng trên trình giả lập và thiết bị, đặt và xem các điểm dừng (breakpoint), từng bước thực thi mã và kiểm tra các biến.

#### Những gì bạn cần biết trước

Bạn cần có khả năng:

- Tạo một dự án Android Studio.
- Sử dụng trình chỉnh sửa bố cục để làm việc với các phần tử EditText và Button.

- Xây dựng và chạy ứng dụng trong Android Studio trên cả trình giả lập và thiết bị.
- Đọc và phân tích dấu vết ngăn xếp (stack trace), bao gồm nguyên tắc "vào sau, ra trước" (last in, first out).
- Thêm câu lệnh ghi nhật ký và xem nhật ký hệ thống (ngăn Logcat) trong Android Studio.

## **Những gì bạn sẽ học**

- Cách chạy ứng dụng ở chế độ gỡ lỗi trên trình giả lập hoặc thiết bị.
- Cách từng bước thực thi ứng dụng.
- Cách đặt và tổ chức các điểm dừng (breakpoint).
- Cách kiểm tra và sửa đổi biến trong trình gỡ lỗi.

## **Những gì bạn sẽ làm**

- Xây dựng ứng dụng SimpleCalc.
- Đặt và xem các điểm dừng trong mã của SimpleCalc.
- Từng bước thực thi mã khi ứng dụng chạy.
- Kiểm tra biến và đánh giá biểu thức.
- Xác định và sửa lỗi trong ứng dụng mẫu.

## **Nhiệm vụ 1: Khám phá dự án và ứng dụng SimpleCalc**

Trong bài thực hành này, bạn sẽ không tự xây dựng ứng dụng SimpleCalc. Dự án hoàn chỉnh có sẵn tại SimpleCalc. Trong nhiệm vụ này, bạn sẽ mở dự án SimpleCalc trong Android Studio và khám phá một số tính năng chính của ứng dụng.

### **1. Tải xuống và mở dự án SimpleCalc**

- Tải xuống SimpleCalc và giải nén tệp.
- Mở Android Studio và chọn File > Open.
- Điều hướng đến thư mục chứa SimpleCalc, chọn thư mục đó và nhấn OK. Dự án SimpleCalc sẽ được xây dựng.
- Mở ngăn Project > Android nếu nó chưa được mở.

### **2. Khám phá bố cục**

- Mở tệp activity\_main.xml.
- Nhấp vào tab Text để xem mã XML.
- Nhấp vào tab Preview để xem bản xem trước bố cục.

Xem xét mã XML và thiết kế bố cục, lưu ý những điểm sau:



- Bố cục chứa hai phần tử EditText để nhập dữ liệu, bốn phần tử Button để thực hiện phép toán và một phần tử TextView để hiển thị kết quả.
- Mỗi nút tính toán có trình xử lý sự kiện android:onClick riêng (onAdd, onSub, v.v.).
- Phần tử TextView hiển thị kết quả không có văn bản mặc định.
- Hai phần tử EditText có thuộc tính android:inputType với giá trị "numberDecimal". Thuộc tính này chỉ định rằng EditText chỉ chấp nhận số làm đầu vào. Bàn phím xuất hiện trên màn hình sẽ chỉ chứa các số. Bạn sẽ tìm hiểu thêm về kiểu đầu vào cho phần tử EditText trong một bài thực hành sau.

### 3. Khám phá mã ứng dụng

- Mở rộng thư mục app > java trong ngăn Project > Android. Ngoài lớp MainActivity, dự án này còn bao gồm lớp tiện ích Calculator.
- Mở lớp Calculator và kiểm tra mã. Lưu ý rằng các phép toán mà máy tính có thể thực hiện được định nghĩa bởi enum Operator và tất cả các phương thức toán học đều ở mức truy cập public.
- Mở lớp MainActivity và kiểm tra mã cùng các chú thích.  
Lưu ý những điểm sau:
  - Tất cả các trình xử lý sự kiện android:onClick được định nghĩa đều gọi phương thức compute() riêng tư, với tên phép toán là một trong các giá trị từ enum Calculator.Operator.
  - Phương thức compute() gọi phương thức riêng tư getOperand() (phương thức này tiếp tục gọi getOperandText()) để lấy giá trị số từ các phần tử EditText.
  - Sau đó, phương thức compute() sử dụng câu lệnh switch dựa trên tên toán tử để gọi phương thức tương ứng trong thể hiện Calculator (mCalculator).
  - Các phương thức tính toán trong lớp Calculator thực hiện các phép toán số học thực tế và trả về kết quả.
  - Phần cuối cùng của phương thức compute() cập nhật TextView với kết quả của phép tính.

### 4. Chạy ứng dụng

Chạy ứng dụng và thực hiện các bước sau:

- Nhập cả số nguyên và số thực vào ô tính toán.
- Nhập số thực có phần thập phân dài (ví dụ: 1.6753456).
- Chia một số cho 0.
- Để trống một hoặc cả hai trường EditText và thử thực hiện phép tính bất kỳ.
- Nhấp vào tab Logcat ở cuối cửa sổ Android Studio để mở ngăn Logcat (nếu chưa mở). Kiểm tra dấu vết ngăn xếp tại thời điểm ứng dụng báo lỗi.

Nếu một hoặc cả hai phần tử EditText trong SimpleCalc trống, ứng dụng sẽ báo lỗi ngoại lệ, như minh họa trong hình dưới đây, và nhật ký hệ thống hiển thị trạng thái của ngăn xếp thực thi tại thời điểm xảy ra lỗi. Dấu vết ngăn xếp thường cung cấp thông tin quan trọng về nguyên nhân gây ra lỗi.

## Nhiệm vụ 2: Chạy SimpleCalc trong trình gỡ lỗi

Trong nhiệm vụ này, bạn sẽ được giới thiệu về trình gỡ lỗi trong Android Studio và học cách đặt một điểm dừng (breakpoint) cũng như chạy ứng dụng của bạn ở chế độ gỡ lỗi.

### 1. Khởi động và chạy ứng dụng của bạn ở chế độ gỡ lỗi

- Trong Android Studio, chọn Run > Debug app hoặc nhấp vào biểu tượng Debug trên thanh công cụ.
- Nếu ứng dụng của bạn đã chạy, bạn sẽ được hỏi liệu bạn có muốn khởi động lại ứng dụng ở chế độ gỡ lỗi hay không. Nhấp vào Restart app. Android Studio sẽ biên dịch và chạy ứng dụng của bạn trên trình giả lập hoặc thiết bị. Quá trình gỡ lỗi sẽ giống nhau trong cả hai trường hợp. Khi Android Studio đang khởi tạo trình gỡ lỗi, bạn có thể thấy thông báo "Waiting for debugger" trên thiết bị trước khi có thể sử dụng ứng dụng.
- Nhấp vào tab Debug ở cuối cửa sổ Android Studio để hiển thị bảng Debug (hoặc chọn View > Tool Windows > Debug). Tab Debugger trong bảng này sẽ tự động được chọn, hiển thị bảng Debugger.

### 2. Đặt một điểm dừng (breakpoint)

Một điểm dừng là một vị trí trong mã nguồn nơi bạn muốn tạm dừng quá trình thực thi bình thường của ứng dụng để thực hiện các hành động khác như kiểm tra biến, đánh giá biểu thức hoặc thực thi từng dòng mã để xác định nguyên nhân của lỗi thời gian chạy. Bạn có thể đặt một điểm dừng trên bất kỳ dòng mã thực thi nào.

- Mở MainActivity, và nhấp vào dòng thứ tư trong phương thức compute() (dòng ngay sau câu lệnh try).
- Nhấp vào phần lề bên trái của trình chỉnh sửa mã tại dòng đó, bên cạnh số dòng. Một chấm đỏ xuất hiện trên dòng đó, cho biết đã đặt một điểm dừng. Nếu ứng dụng đang chạy ở chế độ gỡ lỗi, chấm đỏ sẽ có thêm dấu kiểm. Ngoài ra, bạn cũng có thể chọn Run > Toggle Line Breakpoint hoặc nhấn Control-F8 (Command-F8 trên Mac) để đặt hoặc xóa một điểm dừng trên một dòng. Nếu bạn nhấp vào điểm dừng nhầm, bạn có thể hoàn tác bằng cách nhấp lại vào điểm dừng. Nếu bạn nhấp vào một dòng không thể thực thi, chấm đỏ sẽ có thêm dấu "x" và một cảnh báo xuất hiện cho biết dòng mã đó không thể thực thi.
- Trong ứng dụng SimpleCalc, nhập số vào các trường EditText và nhấp vào một trong các nút tính toán.

Việc thực thi ứng dụng của bạn sẽ dừng lại khi đến điểm dừng mà bạn đã đặt, và trình gỡ lỗi sẽ hiển thị trạng thái hiện tại của ứng dụng tại điểm dừng đó, như minh họa trong hình bên dưới.

Bảng Debug hiển thị các thông tin sau:

- **Frames tab:** Nhấp vào để hiển thị bảng Frames, nơi chứa các stack frame thực thi hiện tại của một luồng nhất định. Stack frame hiển thị mỗi lớp và phương thức đã được gọi trong ứng dụng và trong môi trường chạy Android, với phương thức gần đây nhất nằm ở đầu danh sách.  
Nhấp vào tab Threads để thay thế bảng Frames bằng bảng Threads. Ứng dụng của bạn hiện đang chạy trong luồng chính (main thread) và đang thực thi phương thức `compute()` trong `MainActivity`.
- **Watches button:** Nhấp vào để hiển thị bảng Watches trong bảng Variables, nơi hiển thị các giá trị của bất kỳ biến nào bạn đã đặt theo dõi. Theo dõi biến giúp bạn theo dõi giá trị của một biến cụ thể trong chương trình và xem cách biến đó thay đổi trong quá trình chạy chương trình.
- **Variables pane:** Hiển thị các biến trong phạm vi hiện tại và giá trị của chúng. Ở giai đoạn thực thi này của ứng dụng, các biến có sẵn bao gồm: `this` (đối tượng Activity), `operator` (tên toán tử từ `Calculator.Operator` mà phương thức được gọi), cùng với các biến toàn cục cho các phần tử `EditText` và `TextView`. Mỗi biến trong bảng này có một biểu tượng mở rộng để mở rộng danh sách thuộc tính của đối tượng đó. Hãy thử mở rộng một biến để khám phá các thuộc tính của nó

### 3. Tiếp tục thực thi ứng dụng

Tiếp tục thực thi ứng dụng của bạn bằng cách chọn **Run > Resume Program**, hoặc nhấp vào biểu tượng Resume ở bên trái cửa sổ trình gỡ lỗi.

Ứng dụng SimpleCalc sẽ tiếp tục chạy và bạn có thể tương tác với ứng dụng cho đến khi mã nguồn chạy đến điểm dừng tiếp theo.

### 4. Gỡ lỗi một ứng dụng đang chạy

Nếu ứng dụng của bạn đã chạy trên thiết bị hoặc trình giả lập, và bạn quyết định muốn gỡ lỗi ứng dụng đó, bạn có thể chuyển ứng dụng đang chạy sang chế độ gỡ lỗi.

- Chạy ứng dụng SimpleCalc như bình thường bằng biểu tượng Run.
- Chọn **Run > Attach debugger to Android process** hoặc nhấp vào biểu tượng Attach trên thanh công cụ.
- Chọn tiến trình của ứng dụng của bạn từ hộp thoại xuất hiện và nhấp vào OK.

## Nhiệm vụ 3: Khám phá các tính năng của trình gỡ lỗi

Trong nhiệm vụ này, chúng ta sẽ khám phá các tính năng khác nhau của trình gỡ lỗi trong Android Studio, bao gồm thực thi từng dòng mã, làm việc với breakpoint và kiểm tra biến.

## 1. Thực thi từng bước mã của ứng dụng

Sau khi đặt một breakpoint, bạn có thể sử dụng trình gỡ lỗi để thực thi từng dòng mã trong ứng dụng một cách tuần tự và kiểm tra trạng thái của các biến trong khi ứng dụng đang chạy.

- Chạy ứng dụng trong chế độ debug trên Android Studio, với breakpoint đã đặt ở nhiệm vụ trước.
- Trong ứng dụng, nhập số vào cả hai ô EditText và nhấn nút Add. Quá trình thực thi của ứng dụng dừng lại tại breakpoint mà bạn đã đặt, và cửa sổ Debugger sẽ hiển thị trạng thái hiện tại của ứng dụng. Dòng hiện tại trong mã sẽ được đánh dấu.
- Nhấp vào nút Step Over ở đầu cửa sổ trình gỡ lỗi. Trình gỡ lỗi thực thi dòng mã hiện tại trong phương thức compute() (dòng chứa phép gán cho operandOne), sau đó di chuyển đến dòng tiếp theo (phép gán cho operandTwo). Trong Variables pane, trạng thái thực thi mới được cập nhật và các giá trị hiện tại của biến cũng xuất hiện trong mã nguồn dưới dạng chữ nghiêng. Bạn cũng có thể dùng Run > Step Over, hoặc nhấn F8, để thực hiện bước này.
- Ở dòng tiếp theo (phép gán cho operandTwo), nhấp vào biểu tượng Step Into. Step Into sẽ nhảy vào phương thức được gọi trong dòng hiện tại (thay vì chỉ thực thi và vẫn giữ nguyên dòng). Vì dòng này gọi getOperand(), trình gỡ lỗi sẽ chuyển đến phần khai báo của phương thức này trong MainActivity. Khi bước vào một phương thức, Frames pane cập nhật để hiển thị khung gọi mới (ở đây là getOperand()), và Variables pane hiển thị các biến trong phạm vi mới.
- Bạn cũng có thể dùng Run > Step Into, hoặc nhấn F7, để bước vào phương thức.
  - Nhấp vào Step Over để thực thi từng dòng trong getOperand(). Khi phương thức hoàn tất, trình gỡ lỗi đưa bạn trở lại vị trí ban đầu khi bạn bước vào phương thức và tất cả các bảng điều khiển cập nhật để hiển thị thông tin mới.
  - Nhấp vào Step Over hai lần để di chuyển đến dòng đầu tiên trong khối case cho ADD.
  - Nhấp vào Step Into. Trình gỡ lỗi thực thi phương thức thích hợp trong lớp Calculator, mở tệp Calculator.java và cuộn đến vị trí thực thi trong lớp đó. Các bảng điều khiển tiếp tục cập nhật trạng thái mới.
  - Sử dụng Step Out để thực thi phần còn lại của phương thức tính toán và quay trở lại phương thức compute() trong MainActivity. Bạn cũng có thể sử dụng Run > Step Out, hoặc nhấn Shift + F8 để thoát khỏi một phương thức.

## 2. Làm việc với Breakpoints

Bạn có thể sử dụng breakpoint để xác định vị trí trong mã mà bạn muốn dừng thực thi ứng dụng để gỡ lỗi phần đó.

- Tìm breakpoint mà bạn đã đặt trong nhiệm vụ trước—ở đầu phương thức `compute()` trong `MainActivity`.
- Thêm một breakpoint vào đầu câu lệnh `switch`.
- Nhấp chuột phải vào breakpoint mới đó để nhập điều kiện, như minh họa trong hình dưới, và nhập biểu thức kiểm tra sau vào trường `Condition`:
- Nhấp vào `Done`. Breakpoint thứ hai này là một breakpoint có điều kiện. Việc thực thi ứng dụng sẽ chỉ dừng lại tại breakpoint này nếu biểu thức điều kiện trả về giá trị `true`. Trong trường hợp này, biểu thức chỉ đúng nếu một trong hai toán hạng mà bạn nhập là 42. Bạn có thể nhập bất kỳ biểu thức Java nào làm điều kiện miễn là nó trả về giá trị boolean.
- Chạy ứng dụng của bạn ở chế độ gỡ lỗi (`Run > Debug`) hoặc nhấp vào `Resume` nếu ứng dụng đã đang chạy. Trong ứng dụng, nhập hai số khác 42 và nhấn nút `Add`. Việc thực thi sẽ tạm dừng tại breakpoint đầu tiên trong phương thức `compute()`.
- Nhấp vào `Resume` để tiếp tục gỡ lỗi ứng dụng. Quan sát rằng việc thực thi không dừng tại breakpoint thứ hai vì điều kiện không được đáp ứng.
- Trong ứng dụng, nhập 42 vào ô `EditText` đầu tiên và nhấp vào bất kỳ nút nào. Nhấp vào `Resume` để tiếp tục thực thi sau breakpoint đầu tiên. Quan sát rằng breakpoint thứ hai trong câu lệnh `switch`—breakpoint có điều kiện—đã dừng việc thực thi vì điều kiện được đáp ứng.
- Nhấp chuột phải (hoặc `Control-click`) vào breakpoint đầu tiên trong `compute()` và bỏ chọn `Enabled`. Nhấp vào `Done`. Quan sát rằng biểu tượng breakpoint bây giờ có một chấm xanh với viền đỏ. Việc vô hiệu hóa breakpoint cho phép bạn tạm thời “tắt” breakpoint đó mà không thực sự xóa nó khỏi mã nguồn. Nếu bạn xóa hoàn toàn một breakpoint, bạn cũng sẽ mất bất kỳ điều kiện nào mà bạn đã tạo cho nó, vì vậy việc vô hiệu hóa thường là lựa chọn tốt hơn. Bạn cũng có thể tắt tất cả các breakpoint trong ứng dụng bằng biểu tượng `Mute Breakpoints`.
- Nhấp vào `View Breakpoints` ở cạnh trái của cửa sổ trình gỡ lỗi. Cửa sổ `Breakpoints` xuất hiện. Cửa sổ `Breakpoints` cho phép bạn xem tất cả các breakpoint trong ứng dụng, bật hoặc tắt từng breakpoint, và thêm các tính năng bổ sung cho breakpoint, bao gồm điều kiện, phụ thuộc vào breakpoint khác, và ghi log.

## 3. Kiểm tra và chỉnh sửa biến

Trình gỡ lỗi của Android Studio cho phép bạn kiểm tra trạng thái của các biến trong ứng dụng khi ứng dụng chạy.

- Chạy ứng dụng `SimpleCalc` ở chế độ gỡ lỗi nếu nó chưa chạy.

- Trong ứng dụng, nhập hai số, một trong số đó là 42, và nhấn nút Add. Breakpoint đầu tiên trong `compute()` vẫn bị tắt. Việc thực thi sẽ dừng tại breakpoint thứ hai (breakpoint có điều kiện ở câu lệnh `switch`), và trình gỡ lỗi sẽ xuất hiện.
- Quan sát trong bảng Variables rằng các biến `operandOne` và `operandTwo` có giá trị bạn đã nhập vào ứng dụng.
- Biến `this` là một đối tượng MainActivity. Nhấp vào biểu tượng mở rộng để xem danh sách các biến thành viên của đối tượng đó. Nhấp vào biểu tượng mở rộng một lần nữa để đóng danh sách.
- Nhấp chuột phải (hoặc Control-click) vào biến `operandOne` trong bảng Variables và chọn Set Value.
- Thay đổi giá trị của `operandOne` thành 10 và nhấn Return.
- Thay đổi giá trị của `operandTwo` thành 10 theo cách tương tự và nhấn Return.
- Quan sát rằng kết quả trong ứng dụng bây giờ dựa trên các giá trị biến đã thay đổi trong trình gỡ lỗi.
- Nhấp vào Resume để tiếp tục chạy ứng dụng.
- Trong ứng dụng, các giá trị ban đầu (bao gồm 42) vẫn được giữ trong ô EditText.
- Nhấp vào Evaluate Expression để kiểm tra trạng thái biến.
- Nhập `mOperandOneEditText.getHint();` và nhấn Evaluate.
- Kết quả hiển thị chuỗi "Type Operand 1".
- Nhấp vào Close để đóng cửa sổ Evaluate Code Fragment.

## Tổng kết

- Xem thông tin log trong Android Studio bằng cách nhấp vào tab Logcat.
- Chạy ứng dụng ở chế độ gỡ lỗi bằng cách nhấp vào biểu tượng Debug hoặc chọn Run > Debug app.
- Nhấp vào tab Debug để hiển thị ngăn Debug. Nhấp vào tab Debugger trong ngăn Debug để hiển thị bảng Debugger (nếu chưa được chọn).
- Bảng Debugger hiển thị (ngăn xếp) Frames, các biến trong một frame cụ thể, và Watches (theo dõi biến khi chương trình chạy).
- Breakpoint là một điểm trong mã nguồn nơi bạn muốn tạm dừng việc thực thi bình thường của ứng dụng để thực hiện các hành động khác. Đặt hoặc xóa breakpoint bằng cách nhấp vào lề bên trái của cửa sổ trình chỉnh sửa ngay bên cạnh dòng mã cần đặt breakpoint.

## Xây dựng và chạy ứng dụng

Mở ứng dụng SimpleCalc.

- Trong MainActivity, đặt một breakpoint tại dòng đầu tiên của phương thức `onAdd()`.

- Chạy ứng dụng trong trình gỡ lỗi. Thực hiện một phép cộng trong ứng dụng. Việc thực thi sẽ dừng lại tại breakpoint.
- Nhấp vào Step Into để theo dõi từng bước thực thi của ứng dụng. Lưu ý rằng Step Into sẽ mở và thực thi các tệp từ Android framework, cho phép bạn xem cách Android vận hành mã của bạn.
- Quan sát cách ngăn Debug thay đổi khi bạn bước qua mã, bao gồm stack frame hiện tại và các biến cục bộ.
- Quan sát cách mã trong trình chỉnh sửa được chú thích khi từng dòng được thực thi.
- Nhấp vào Step Out để quay lại ứng dụng nếu ngăn xếp thực thi trở nên quá sâu và khó hiểu.

### **3.2) Kiểm thử đơn vị**

#### **Giới thiệu**

Kiểm thử mã nguồn có thể giúp bạn phát hiện lỗi sớm trong quá trình phát triển, khi chi phí sửa lỗi còn thấp nhất. Khi ứng dụng của bạn ngày càng lớn và phức tạp hơn, kiểm thử giúp cải thiện độ ổn định của mã nguồn. Với các bài kiểm thử trong mã nguồn, bạn có thể kiểm tra từng phần nhỏ của ứng dụng một cách độc lập, đồng thời có thể kiểm thử theo cách tự động hóa và lặp lại dễ dàng.

Android Studio và Android Testing Support Library hỗ trợ nhiều loại kiểm thử và framework kiểm thử khác nhau. Trong bài thực hành này, bạn sẽ khám phá tính năng kiểm thử tích hợp sẵn của Android Studio, đồng thời học cách viết và chạy kiểm thử đơn vị (unit test) cục bộ.

Kiểm thử đơn vị cục bộ là các bài kiểm thử được biên dịch và chạy hoàn toàn trên máy cục bộ bằng Java Virtual Machine (JVM). Bạn sử dụng kiểm thử đơn vị cục bộ để kiểm thử các phần của ứng dụng không cần truy cập vào Android framework hoặc thiết bị/mô phỏng Android, chẳng hạn như logic nội bộ. Bạn cũng có thể sử dụng kiểm thử đơn vị cục bộ để kiểm thử các phần của ứng dụng mà bạn có thể tạo các đối tượng giả (mock hoặc stub) có hành vi tương tự như các thành phần trong framework.

Kiểm thử đơn vị được viết bằng JUnit, một framework kiểm thử phổ biến trong Java.

#### **Những kiến thức bạn cần biết trước**

Bạn nên có khả năng:

- Tạo một dự án Android Studio.
- Xây dựng và chạy ứng dụng trong Android Studio, cả trên trình mô phỏng và thiết bị thật.

- Điều hướng trong mục Project > Android của Android Studio.
- Xác định các thành phần chính của một dự án Android Studio, bao gồm AndroidManifest.xml, tài nguyên (resources), tệp Java và tệp Gradle.

## Những gì bạn sẽ học

- Cách tổ chức và chạy kiểm thử trong Android Studio.
- Hiểu kiểm thử đơn vị là gì.
- Viết kiểm thử đơn vị cho mã nguồn của bạn.

## Những gì bạn sẽ làm

- Chạy các bài kiểm thử ban đầu trong ứng dụng SimpleCalc.
- Thêm nhiều bài kiểm thử hơn vào ứng dụng SimpleCalc.
- Chạy kiểm thử đơn vị để xem kết quả.

## Nhiệm vụ 1: Khám phá và chạy CalculatorTest

Bạn sẽ viết và chạy các bài kiểm thử (bao gồm cả kiểm thử đơn vị và kiểm thử có công cụ hỗ trợ) trực tiếp trong Android Studio, cùng với mã nguồn của ứng dụng. Mỗi dự án Android mới đều bao gồm các lớp kiểm thử mẫu cơ bản mà bạn có thể mở rộng hoặc thay thế theo nhu cầu của mình.

Trong nhiệm vụ này, bạn sẽ quay lại ứng dụng SimpleCalc, ứng dụng này đã có sẵn một lớp kiểm thử đơn vị cơ bản.

### 1. Khám phá source sets và CalculatorTest

Source sets là tập hợp mã nguồn trong dự án của bạn, phục vụ cho các mục tiêu xây dựng khác nhau hoặc các "flavor" khác nhau của ứng dụng. Khi Android Studio tạo dự án của bạn, nó cũng tạo ba source sets chính:

- Main source set: Chứa mã nguồn và tài nguyên chính của ứng dụng.
- Test source set: Chứa kiểm thử đơn vị cục bộ của ứng dụng. Source set này sẽ hiển thị "(test)" sau tên package.
- AndroidTest source set: Chứa các kiểm thử có công cụ hỗ trợ (instrumented tests). Source set này sẽ hiển thị "(androidTest)" sau tên package.

Trong nhiệm vụ này, bạn sẽ tìm hiểu cách hiển thị source sets trong Android Studio, kiểm tra cấu hình Gradle cho kiểm thử, và chạy các kiểm thử đơn vị của ứng dụng SimpleCalc.

- Mở dự án SimpleCalc trong Android Studio (nếu chưa mở).



- Mở Project > Android, sau đó mở rộng thư mục app và java.  
Trong chế độ xem Android, thư mục java liệt kê tất cả các source sets của ứng dụng theo tên package. Mã nguồn ứng dụng nằm trong source set `com.android.example.SimpleCalc`. Mã nguồn kiểm thử nằm trong source set có chữ (test) sau tên package: `com.android.example.SimpleCalc (test)`.
- Mở rộng thư mục `com.android.example.SimpleCalc (test)`. Đây là nơi chứa các kiểm thử đơn vị cục bộ của ứng dụng.  
Android Studio tạo sẵn một lớp kiểm thử mẫu cho bạn trong thư mục này. Trong ứng dụng `SimpleCalc`, lớp kiểm thử có tên là `CalculatorTest`.
- Mở `CalculatorTest` và kiểm tra mã nguồn, chú ý các điểm sau:
  - Các mục nhập duy nhất là từ các gói `org.junit`, `org.hamcrest` và `android.test`. Không có sự phụ thuộc nào vào các lớp khung Android.
  - Chú thích `@RunWith(JUnit4.class)` chỉ ra trình chạy sẽ được sử dụng để chạy các bài kiểm tra trong lớp này. Trình chạy kiểm tra là một thư viện hoặc bộ công cụ cho phép thực hiện kiểm tra và in kết quả vào nhật ký. Đối với các bài kiểm tra có yêu cầu về thiết lập hoặc cơ sở hạ tầng phức tạp hơn (như Espresso), bạn sẽ sử dụng các trình chạy thử nghiệm khác nhau. Đối với ví dụ này, chúng tôi đang sử dụng trình chạy thử nghiệm JUnit4 cơ bản.
  - Chú thích `@SmallTest` chỉ ra rằng tất cả các bài kiểm tra trong lớp này đều là các bài kiểm tra đơn vị không có phụ thuộc và chạy trong mili giây. Các chú thích `@SmallTest`, `@MediumTest` và `@LargeTest` là các quy ước giúp dễ dàng đóng gói các nhóm bài kiểm tra thành các bộ có chức năng tương tự.
  - Phương thức `setUp()` được sử dụng để thiết lập môi trường trước khi kiểm tra và bao gồm chú thích `@Before`. Trong trường hợp này, thiết lập sẽ tạo một phiên bản mới của lớp `Calculator` và gán phiên bản đó cho biến thành viên `mCalculator`.
  - Phương thức `addTwoNumbers()` là một bài kiểm tra thực tế và được chú thích bằng `@Test`. Chỉ các phương thức trong lớp kiểm tra có chú thích `@Test` mới được coi là các bài kiểm tra đối với trình chạy thử nghiệm. Lưu ý rằng theo quy ước, các phương thức kiểm tra không bao gồm từ "test".
  - Dòng đầu tiên của `addTwoNumbers()` gọi phương thức `add()` từ lớp `Calculator`. Bạn chỉ có thể kiểm tra các phương thức công khai hoặc được bảo vệ bằng gói. Trong trường hợp này, `Calculator` là một lớp công khai với các phương thức công khai, vì vậy mọi thứ đều ổn.
  - Dòng thứ hai là khẳng định cho bài kiểm tra. Khẳng định là các biểu thức phải đánh giá và trả về giá trị true để bài kiểm tra vượt qua. Trong trường hợp này, khẳng định là kết quả bạn nhận được từ phương thức `add(1 + 1)` khớp với số đã cho là 2. Bạn sẽ tìm hiểu thêm về cách tạo khẳng định sau trong bài thực hành này.

## 2. Chạy kiểm thử trong Android Studio

Trong nhiệm vụ này, bạn sẽ chạy các kiểm thử đơn vị trong thư mục test và xem kết quả của cả kiểm thử thành công lẫn kiểm thử thất bại.

- Trong Project > Android, nhấp chuột phải (hoặc Control-click) vào CalculatorTest, sau đó chọn Run 'CalculatorTest'.  
Nếu cần, dự án sẽ được biên dịch trước khi chạy kiểm thử. Cửa sổ CalculatorTest xuất hiện ở cuối màn hình, hiển thị kết quả chạy kiểm thử. Ở trên cùng của cửa sổ, danh sách cấu hình thực thi (execution configurations) sẽ thay đổi thành CalculatorTest. Sau khi hoàn thành bước này, bạn sẽ thấy kết quả của các kiểm thử được hiển thị. Nếu có kiểm thử nào thất bại, bạn có thể xem chi tiết lỗi để tìm cách sửa.  
Tất cả các bài kiểm tra trong lớp CalculatorTest sẽ chạy và nếu các bài kiểm tra đó thành công, thanh tiến trình ở đầu giao diện sẽ chuyển sang màu xanh lá cây. (Trong trường hợp này, hiện tại chỉ có một bài kiểm tra.) Một thông báo trạng thái ở phần chân trang cũng báo "Tests Passed".
- Mở CalculatorTest nếu chưa mở và thay đổi câu lệnh kiểm tra trong phương thức addTwoNumbers() thành:
- Trong menu cấu hình chạy (run configurations) ở đầu màn hình, chọn CalculatorTest (nếu chưa được chọn) và nhấn Run.  
Bài kiểm tra sẽ chạy lại như trước, nhưng lần này câu lệnh kiểm tra sẽ thất bại (3 không bằng 1 + 1). Thanh tiến trình trong cửa sổ chạy thử nghiệm sẽ chuyển sang màu đỏ và nhật ký kiểm thử sẽ chỉ ra bài kiểm tra thất bại ở đâu và vì sao.
- Đổi lại câu lệnh kiểm tra trong addTwoNumbers() thành bài kiểm tra đúng và chạy lại kiểm thử để đảm bảo tất cả đều thành công.
- Trong menu cấu hình chạy, chọn app để chạy ứng dụng bình thường.

### Nhiệm vụ 2: Thêm nhiều bài kiểm thử đơn vị vào CalculatorTest

Với kiểm thử đơn vị, bạn sẽ lấy một phần nhỏ của mã trong ứng dụng, chẳng hạn như một phương thức hoặc một lớp, và tách biệt nó khỏi phần còn lại của ứng dụng. Các bài kiểm thử đảm bảo rằng phần mã đó hoạt động đúng như mong đợi. Thông thường, một bài kiểm thử đơn vị sẽ gọi một phương thức với nhiều đầu vào khác nhau, kiểm tra xem phương thức đó có hoạt động đúng và trả về kết quả như mong muốn hay không.

Trong nhiệm vụ này, bạn sẽ tìm hiểu thêm về cách xây dựng bài kiểm thử đơn vị, viết thêm kiểm thử đơn vị cho các phương thức tiện ích Calculator trong ứng dụng SimpleCalc, và chạy các bài kiểm thử để đảm bảo chúng tạo ra kết quả như mong đợi.

Lưu ý: Kiểm thử đơn vị, phát triển theo hướng kiểm thử (TDD) và API JUnit 4 đều là các chủ đề rộng và phức tạp, nằm ngoài phạm vi của khóa học này.

## 1. Thêm nhiều bài kiểm thử cho phương thức add()

Mặc dù không thể kiểm thử mọi giá trị mà phương thức add() có thể nhận được, nhưng bạn nên kiểm tra các đầu vào bất thường. Ví dụ:

- Các toán hạng âm
- Các số dấu chấm động (floating-point)
- Các số cực lớn
- Các toán hạng khác loại (ví dụ: một số float và một số double)
- Một toán hạng bằng 0
- Một toán hạng là vô cùng (infinity)

Trong nhiệm vụ này, chúng ta sẽ thêm các bài kiểm thử đơn vị cho phương thức add() để kiểm tra các loại đầu vào khác nhau.

- Thêm một phương thức mới vào CalculatorTest có tên addTwoNumbersNegative(). Sử dụng cấu trúc sau:  
Phương thức kiểm thử này có cấu trúc tương tự như addTwoNumbers(): nó là một phương thức công khai (public), không có tham số, và trả về void. Nó được chú thích bằng @Test, cho biết đây là một bài kiểm thử đơn vị.  
Tại sao không chỉ thêm nhiều câu lệnh kiểm tra vào addTwoNumbers()? Nhóm nhiều câu lệnh kiểm tra vào một phương thức có thể khiến quá trình gỡ lỗi khó khăn hơn nếu chỉ một câu lệnh thất bại. Điều này cũng làm lu mờ các bài kiểm thử đã thành công. Quy tắc chung là cung cấp một phương thức kiểm thử riêng cho từng câu lệnh kiểm tra riêng lẻ.
- Chạy tất cả các bài kiểm thử trong CalculatorTest như trước.  
Trong cửa sổ kiểm thử, cả addTwoNumbers và addTwoNumbersNegative đều được liệt kê là các bài kiểm thử có sẵn (và đã vượt qua) ở bảng bên trái. Bài kiểm thử addTwoNumbersNegative vẫn thành công mặc dù chưa có mã nào bên trong – một bài kiểm thử không làm gì vẫn được coi là thành công.
- Thêm một dòng vào addTwoNumbersNegative() để gọi phương thức add() trong lớp Calculator với một toán hạng âm.  
Ký hiệu d sau mỗi toán hạng cho biết đây là số kiểu double. Vì phương thức add() được định nghĩa với các tham số kiểu double, nên một số float hoặc int cũng có thể hoạt động. Việc chỉ rõ kiểu giúp bạn kiểm tra các kiểu dữ liệu khác biệt nếu cần thiết.
- Thêm một câu lệnh kiểm tra với assertTrue().  
Phương thức assertTrue() là một câu lệnh kiểm tra trong JUnit4, khẳng định rằng biểu thức trong đôi số đầu tiên bằng với giá trị trong đôi số thứ hai. Các phiên bản cũ hơn của JUnit sử dụng các phương thức kiểm tra cụ thể hơn (assertEquals(), assertNull(), hoặc assertTrue()), nhưng assertTrue() linh hoạt hơn, dễ gỡ lỗi hơn và dễ đọc hơn.

`assertThat()` được sử dụng với `matchers`. `Matchers` là các chuỗi gọi phương thức trong toán hạng thứ hai của câu lệnh kiểm tra, chẳng hạn như `is(equalTo())`. Thư viện `Hamcrest` định nghĩa các `matchers` có sẵn mà bạn có thể sử dụng để xây dựng câu lệnh kiểm tra. (Tên "`Hamcrest`" là một anagram của "`matchers`"). `Hamcrest` cung cấp nhiều `matchers` cơ bản cho hầu hết các kiểm tra đơn giản. Bạn cũng có thể định nghĩa `matchers` tùy chỉnh cho các kiểm tra phức tạp hơn.

- Thêm một bài kiểm thử đơn vị mới vào `CalculatorTest` cho số dấu chấm động: Đây là một bài kiểm thử rất giống với bài trước, nhưng có một tham số là kiểu `float` thay vì `double`. Vì phương thức `add()` được định nghĩa với tham số kiểu `double`, nên bạn có thể gọi nó bằng `float`, và số đó sẽ được chuyển đổi thành `double`.
- Nhấn `Run` để chạy tất cả các bài kiểm thử lại.  
Lần này, bài kiểm thử thất bại và thanh tiến trình chuyển sang màu đỏ. Phần quan trọng của thông báo lỗi là:  
Phép toán số dấu chấm động không chính xác, và việc chuyển đổi đã dẫn đến hiệu ứng phụ là độ chính xác bổ sung. Câu lệnh kiểm tra trong bài kiểm thử về mặt kỹ thuật là sai: giá trị mong đợi không bằng giá trị thực tế.  
Câu hỏi đặt ra là: Khi gặp vấn đề về độ chính xác trong việc chuyển đổi kiểu `float`, đó là lỗi của mã hay lỗi của bài kiểm thử? Trong trường hợp này, cả hai đầu vào của phương thức `add()` trong ứng dụng `SimpleCalc` luôn là kiểu `double`, nên đây là một bài kiểm thử không thực tế. Tuy nhiên, nếu ứng dụng của bạn được viết sao cho đầu vào có thể là `double` hoặc `float`, và bạn chỉ quan tâm đến một mức độ chính xác nhất định, bạn cần thêm một khoảng sai số vào bài kiểm thử để "đủ gần" cũng được coi là thành công.
- Thay đổi phương thức `assertThat()` để sử dụng `matcher closeTo()`.
- Nhấn `Run` để chạy lại tất cả các bài kiểm thử.  
Lần này bài kiểm thử thành công.  
Với `matcher closeTo()`, thay vì kiểm tra bằng chính xác, bạn có thể kiểm tra trong một phạm vi sai số nhất định.

## 2. Thêm kiểm thử đơn vị cho các phương thức tính toán khác

Sử dụng những gì bạn đã học để viết các bài kiểm thử cho lớp `Calculator`.

- Thêm bài kiểm thử `subTwoNumbers()` để kiểm tra phương thức `sub()`.
- Thêm bài kiểm thử `subWorksWithNegativeResults()` để kiểm tra `sub()` với kết quả là số âm.
- Thêm bài kiểm thử `mulTwoNumbers()` để kiểm tra `mul()`.
- Thêm bài kiểm thử `mulTwoNumbersZero()` để kiểm tra `mul()` với một toán hạng bằng 0.
- Thêm bài kiểm thử `divTwoNumbers()` để kiểm tra `div()` với hai toán hạng khác 0.
- Thêm bài kiểm thử `divTwoNumbersZero()` để kiểm tra `div()` với mẫu số bằng 0.

Tất cả các bài kiểm thử này sẽ thành công, trừ `divTwoNumbersZero()` sẽ gây ra ngoại lệ `IllegalArgumentException` vì phép chia cho 0

## Tổng kết

Android Studio có các tính năng tích hợp để chạy các bài kiểm tra đơn vị cục bộ:

- Kiểm tra đơn vị cục bộ sử dụng JVM của máy cục bộ mà không cần đến Android framework.
- JUnit là framework phổ biến được sử dụng để viết các bài kiểm tra đơn vị.
- Các bài kiểm tra được đặt trong thư mục (test) trong Android Studio > Android pane.
- Các kiểm tra đơn vị chỉ cần các gói: `org.junit`, `org.hamcrest`, và `android.test`.
- `@RunWith(JUnit4.class)` giúp trình chạy kiểm tra thực thi các bài kiểm tra trong lớp đó.
- Các chú thích `@SmallTest`, `@MediumTest`, `@LargeTest` giúp nhóm các bài kiểm tra theo mức độ phụ thuộc.
- `@SmallTest` chỉ ra rằng tất cả các bài kiểm tra trong lớp là kiểm tra đơn vị, không có phụ thuộc và chạy trong mili giây.
- Instrumented tests chạy trên thiết bị Android thật hoặc trình giả lập và có quyền truy cập vào Android framework.
- Test runner là thư viện hoặc công cụ giúp thực thi kiểm tra và hiển thị kết quả trong log.

## Xây dựng và chạy ứng dụng

Mở ứng dụng SimpleCalc từ bài thực hành sử dụng trình gỡ lỗi. Bạn sẽ thêm một nút POW vào giao diện để tính toán lũy thừa. Ví dụ, với toán hạng là 5 và 4, ứng dụng sẽ tính  $5^4 = 625$ .

Trước khi triển khai chức năng của nút POW, hãy suy nghĩ về các trường hợp đặc biệt có thể xảy ra trong phép toán này.

- Cập nhật lớp Calculator để thêm phương thức `pow()`. Gợi ý: Tham khảo tài liệu về lớp `java.lang.Math`.
- Cập nhật lớp MainActivity để kết nối nút POW với phép tính.

Viết các bài kiểm tra cho phương thức `pow()`:

- Kiểm tra với toán hạng là số nguyên dương.
- Kiểm tra với toán hạng đầu tiên là số nguyên âm.
- Kiểm tra với toán hạng thứ hai là số nguyên âm.

- Kiểm tra với toán hạng đầu tiên là 0 và toán hạng thứ hai là số nguyên dương.
- Kiểm tra với toán hạng thứ hai là 0.
- Kiểm tra với toán hạng đầu tiên là 0 và toán hạng thứ hai là -1.
- Kiểm tra với toán hạng đầu tiên là -0 và toán hạng thứ hai là số âm.

### **3.3) Thư viện hỗ trợ**

#### **Giới thiệu**

Android SDK bao gồm Android Support Library, một tập hợp nhiều thư viện cung cấp các tính năng không có sẵn trong nền tảng Android, bao gồm:

- Các phiên bản tương thích ngược của các thành phần trong nền tảng, giúp ứng dụng chạy trên các phiên bản Android cũ hỗ trợ các tính năng mới.
- Các phần tử giao diện và bố cục bổ sung.
- Hỗ trợ nhiều kiểu thiết bị khác nhau, như TV hoặc thiết bị đeo (wearables).
- Các thành phần hỗ trợ Material Design.
- Các tính năng khác như hỗ trợ màu sắc (palette), chú thích (annotations), bố cục theo tỷ lệ phần trăm (percentage-based layout dimensions), và quản lý tùy chọn (preferences).

#### **Kiến thức cần có trước khi bắt đầu**

Bạn nên có khả năng:

- Tạo một dự án trong Android Studio.
- Sử dụng Layout Editor để làm việc với các phần tử EditText và Button.
- Biên dịch và chạy ứng dụng trên Android Studio, cả trên trình giả lập và thiết bị thật.
- Điều hướng trong mục Project > Android trong Android Studio.
- Xác định các thành phần chính của một dự án Android Studio, bao gồm AndroidManifest.xml, tài nguyên, các tệp Java, và các tệp Gradle.

#### **Những gì bạn sẽ học**

- Cách kiểm tra xem Android Support Library đã có sẵn trong Android Studio chưa.
- Cách sử dụng các lớp thư viện hỗ trợ trong ứng dụng của bạn.
- Sự khác biệt giữa compileSdkVersion, targetSdkVersion và minSdkVersion.
- Cách nhận biết API bị loại bỏ (deprecated) hoặc không khả dụng trong mã nguồn của bạn.
- Tìm hiểu thêm về các thư viện hỗ trợ trong Android.

## Những gì bạn sẽ làm

- Tạo một ứng dụng mới với một TextView và một Button.
- Xác minh rằng Android Support Repository đã được cài đặt trong Android Studio.
- Khám phá các tệp build.gradle trong dự án.
- Quản lý các phương thức hoặc lớp không khả dụng trên các phiên bản Android mà ứng dụng của bạn hỗ trợ.
- Sử dụng các lớp tương thích từ thư viện hỗ trợ để đảm bảo ứng dụng có thể chạy trên nhiều phiên bản Android khác nhau.

### Nhiệm vụ 1: Thiết lập dự án để sử dụng thư viện hỗ trợ

Bạn sẽ tạo ứng dụng HelloCompat và triển khai giao diện cũng như hành vi cơ bản.

#### 1. Xác minh rằng Android Support Repository đã được cài đặt

Thư viện hỗ trợ Android được tải xuống cùng với Android SDK và có sẵn trong SDK Manager. Bạn cần kiểm tra xem Android Support Repository đã được cài đặt chưa.

- Trong Android Studio, chọn Công cụ > Android > Trình quản lý SDK hoặc nhấp vào biểu tượng Trình quản lý SDK. Ngăn Tùy chọn mặc định của Android SDK sẽ xuất hiện.
- Nhấp vào tab Công cụ SDK và mở rộng Kho lưu trữ hỗ trợ, như minh họa trong hình bên dưới.
- Tìm Kho lưu trữ hỗ trợ Android trong danh sách.
  - Nếu Đã cài đặt xuất hiện trong cột Trạng thái, bạn đã hoàn tất. Nhấp vào Hủy.
  - Nếu Chưa cài đặt hoặc Có sẵn bản cập nhật xuất hiện, hãy nhấp vào hộp kiểm bên cạnh Kho lưu trữ hỗ trợ Android. Biểu tượng tải xuống sẽ xuất hiện bên cạnh hộp kiểm. Nhấp vào OK.
- Nhấp vào OK một lần nữa, sau đó là Hoàn tất khi kho lưu trữ hỗ trợ đã được cài đặt.

#### 2. Thiết lập dự án và kiểm tra build.gradle

- Tạo một dự án mới có tên là HelloCompat.  
Trên trang Thiết bị Android mục tiêu, API 15: Android 4.0.3 (IceCreamSandwich) được chọn cho SDK tối thiểu. Như bạn đã tìm hiểu trong các bài học trước, đây là phiên bản cũ nhất của nền tảng Android mà ứng dụng của bạn sẽ hỗ trợ.
- Nhấp vào Tiếp theo và chọn mẫu Hoạt động trống.

- Nhấp vào Tiếp theo và đảm bảo rằng các tùy chọn Tạo tệp bố cục và Tương thích ngược (Tương thích ứng dụng) được chọn. Tùy chọn sau đảm bảo rằng ứng dụng của bạn sẽ tương thích ngược với các phiên bản Android trước đó.
- Nhấp vào Hoàn tất.

#### Khám phá build.gradle (Module:app)

- Trong Android Studio, đảm bảo rằng ngăn Project > Android đang mở.
- Mở rộng phần Gradle Scripts và mở tệp build.gradle (Module: app).  
Lưu ý rằng build.gradle của toàn bộ dự án (build.gradle (Project: helpcompat)) là một tệp khác so với build.gradle của mô-đun ứng dụng. Trong tệp build.gradle (Module: app):
- Xác định dòng compileSdkVersion gần đầu tệp. Ví dụ:  
Phiên bản biên dịch là phiên bản framework Android mà ứng dụng của bạn được biên dịch với trong Android Studio. Đối với các dự án mới, phiên bản biên dịch là tập hợp API framework mới nhất mà bạn đã cài đặt. Giá trị này chỉ ảnh hưởng đến Android Studio và các cảnh báo hoặc lỗi bạn nhận được nếu sử dụng API cũ hơn hoặc mới hơn.
- Xác định dòng minSdkVersion trong phần defaultConfig ở vài dòng dưới.  
Phiên bản tối thiểu là phiên bản API Android cũ nhất mà ứng dụng của bạn có thể chạy. Đây là số bạn đã chọn ở Bước 1 khi tạo dự án. Google Play sử dụng số này để đảm bảo ứng dụng của bạn có thể chạy trên thiết bị của người dùng. Android Studio cũng sử dụng số này để cảnh báo bạn về việc sử dụng các API đã lỗi thời.
- Xác định dòng targetSdkVersion trong phần defaultConfig. Ví dụ:  
Phiên bản mục tiêu chỉ ra phiên bản API mà ứng dụng của bạn được thiết kế và kiểm tra. Nếu API của nền tảng Android cao hơn số này (tức là ứng dụng của bạn đang chạy trên thiết bị mới hơn), nền tảng có thể bật các hành vi tương thích để đảm bảo rằng ứng dụng của bạn tiếp tục hoạt động như đã thiết kế. Ví dụ, Android 6.0 (API 23) cung cấp mô hình quyền runtime mới. Nếu ứng dụng của bạn nhắm đến API thấp hơn, nền tảng sẽ quay lại mô hình quyền cài đặt thời gian cũ.
- Xác định phần dependencies trong build.gradle, gần cuối tệp. Ví dụ:  
Phần dependencies của một dự án mới bao gồm một số phụ thuộc để kiểm thử với Espresso và JUnit, cũng như thư viện hỗ trợ appcompat v7. Số phiên bản của các thư viện trong dự án của bạn có thể khác.
- Cập nhật số phiên bản nếu cần thiết.  
Nếu số phiên bản hiện tại của thư viện thấp hơn phiên bản thư viện hiện có, Android Studio sẽ đánh dấu dòng đó và cảnh báo rằng có phiên bản mới hơn. Chỉnh sửa số phiên bản để cập nhật.
- Cập nhật số compileSdkVersion nếu cần thiết.  
Số phiên bản chính của thư viện hỗ trợ (số đầu tiên) phải khớp với compileSdkVersion của bạn. Khi cập nhật phiên bản thư viện hỗ trợ, bạn có thể cần cập nhật compileSdkVersion để khớp.



- Nhấp vào Sync Now để đồng bộ hóa các tệp Gradle đã cập nhật với dự án.
- Cài đặt các tệp nền tảng SDK bị thiếu nếu cần.  
Nếu bạn cập nhật compileSdkVersion, có thể bạn cần cài đặt các thành phần nền tảng SDK tương ứng. Nhấp vào Install missing platform(s) and sync project để bắt đầu quá trình này.

## Nhiệm vụ 2: Triển khai bố cục và MainActivity

Trong nhiệm vụ này, bạn sẽ triển khai bố cục và hành vi cơ bản cho lớp MainActivity.

### 1. Thay đổi bố cục và màu sắc

Trong nhiệm vụ này, bạn sẽ sửa đổi bố cục activity\_main.xml cho ứng dụng.

- Mở activity\_main.xml trong Project > Android pane.
- Nhấn vào tab Design (nếu chưa được chọn) để hiển thị trình chỉnh sửa bố cục.
- Chọn TextView có nội dung "Hello World" trong bố cục và mở bảng Attributes.
- Thay đổi các thuộc tính của TextView như sau:

Trường thuộc tính	Nhập thông tin sau
ID	hello_textview
textStyle	B (bold)
textAlignment	Center the paragraph icon
textSize	100sp

Điều này sẽ thêm thuộc tính android:id vào TextView với id được đặt là hello\_textview, thay đổi căn chỉnh văn bản, làm cho văn bản đậm, và đặt kích thước văn bản lớn hơn là 100sp.

- Xóa ràng buộc kéo dài từ cạnh dưới của hello\_textview đến cạnh dưới của bố cục để TextView gắn vào phần trên của bố cục và chọn giá trị 8dp cho top margin như hình bên dưới.
- Kéo một Button vào phần dưới cùng của bố cục, thêm các ràng buộc vào cạnh trái, phải và cạnh dưới của bố cục như hình minh họa.
- Thay đổi thuộc tính layout\_width trong bảng Attributes của Button thành match\_constraint.
- Thay đổi các thuộc tính khác trong bảng Attributes của Button như sau:

Trường thuộc tính	Nhập thông tin sau
ID	color_button

text	"Change Color"
------	----------------

- Trong bài học trước, bạn đã học cách trích xuất tài nguyên chuỗi từ một chuỗi văn bản cố định.
- Nhấn vào tab Text để chuyển sang mã XML, trích xuất các chuỗi "Hello Text!" và "Change Color" trong TextView và Button, đồng thời nhập tên tài nguyên chuỗi cho chúng.
- Thêm thuộc tính android:onClick sau vào Button:
- Để thêm màu sắc, mở rộng res và values trong Project > Android pane, sau đó mở colors.xml.
- Thêm các tài nguyên màu sau vào tệp:

Những giá trị màu này đến từ bảng màu được khuyến nghị cho ứng dụng Android được xác định trong Material Design - Style - Color. Các mã chỉ định giá trị RGB của màu dưới dạng thập lục phân.

## 2. Thêm hành vi vào MainActivity

Trong nhiệm vụ này, bạn sẽ hoàn tất việc thiết lập dự án bằng cách thêm các biến riêng tư và triển khai onCreate() và onSaveInstanceState().

- Mở MainActivity.
- Thêm một biến riêng tư ở đầu lớp để giữ đối tượng TextView.
- Thêm mảng màu sau ngay sau biến riêng tư:  
Mỗi tên màu tương ứng với tên của một tài nguyên màu trong colors.xml.
- Trong phương thức onCreate(), sử dụng findViewById() để lấy tham chiếu đến thể hiện TextView và gán nó cho biến riêng tư đó:
  - Cũng trong onCreate(), khôi phục trạng thái đã lưu (nếu có):
  - Thêm phương thức onSaveInstanceState() vào MainActivity.

## Nhiệm vụ 3: Triển khai hành vi của Button

Nút "Change Color" trong ứng dụng HelloCompat chọn ngẫu nhiên một trong 20 màu từ tệp tài nguyên colors.xml và đặt màu văn bản thành màu đó. Trong nhiệm vụ này, bạn sẽ triển khai hành vi cho trình xử lý sự kiện onClick của Button.

### 1. Thêm trình xử lý sự kiện click cho Button

- Mở activity\_main.xml (nếu chưa mở). Nhấn vào tab Text để hiển thị mã XML.
- Nhấn vào "changeColor" trong thuộc tính android:onClick bên trong phần tử Button.

- Nhấn Alt+Enter (hoặc Option+Enter trên Mac), sau đó chọn Create onClick event handler.
- Chọn MainActivity và nhấn OK.  
Điều này tạo một phương thức trống `changeColor()` trong MainActivity:

## 2. Triển khai hành động của Button

- Chuyển sang MainActivity.
- Trong phương thức `changeColor()`, tạo một đối tượng số ngẫu nhiên bằng cách sử dụng lớp `Random` (một lớp trong Java) để tạo các số ngẫu nhiên đơn giản.
- Sử dụng thẻ `random` để chọn một màu ngẫu nhiên từ mảng `mColorArray`. Phương thức `nextInt()` với đối số 20 sẽ lấy một số nguyên ngẫu nhiên từ 0 đến 19. Bạn sử dụng số nguyên đó làm chỉ mục của mảng để lấy tên màu.
- Lấy mã định danh tài nguyên (một số nguyên) cho tên màu từ tài nguyên: Khi ứng dụng của bạn được biên dịch, hệ thống Android chuyển các định nghĩa trong tệp XML thành các tài nguyên với ID số nguyên nội bộ. Có các ID riêng biệt cho cả tên và giá trị. Dòng này khớp chuỗi tên màu từ `colorName` với ID tên màu tương ứng trong tệp tài nguyên XML. Phương thức `getResources()` lấy tất cả tài nguyên của ứng dụng. Phương thức `getIdentifier()` tìm kiếm tên màu (chuỗi) trong tài nguyên màu ("color") cho tên gói hiện tại.
- Lấy ID số nguyên cho màu thực tế từ tài nguyên và gán nó cho biến `colorRes`, sau đó sử dụng phương thức `getTheme()` để lấy giao diện hiện tại của ứng dụng: Phương thức `getResources()` lấy tập hợp tài nguyên cho ứng dụng của bạn, còn phương thức `getColor()` truy xuất một màu cụ thể từ các tài nguyên bằng ID của tên màu. Tuy nhiên, `getColor()` bị gạch chân màu đỏ.  
Nếu bạn di chuột vào `getColor()`, Android Studio sẽ báo: "Call requires API 23 (current min is 15)". Vì `minSdkVersion` của bạn là 15, bạn sẽ nhận được thông báo này nếu cố gắng sử dụng bất kỳ API nào được giới thiệu sau API 15. Bạn vẫn có thể biên dịch ứng dụng, nhưng vì phiên bản `getColor()` này không có sẵn trên thiết bị chạy API thấp hơn 23, ứng dụng của bạn sẽ bị lỗi khi người dùng nhấn vào nút "Change Color".  
Ở giai đoạn này, bạn có thể kiểm tra phiên bản nền tảng và sử dụng phiên bản `getColor()` phù hợp tùy thuộc vào nơi ứng dụng đang chạy. Một cách tốt hơn để hỗ trợ cả API Android cũ và mới mà không có cảnh báo là sử dụng một trong các lớp tương thích trong thư viện hỗ trợ.
- Thay đổi dòng gán `colorRes` để sử dụng lớp `ContextCompat`:  
`ContextCompat` cung cấp nhiều phương thức tương thích để giải quyết các khác biệt API trong ngữ cảnh ứng dụng và tài nguyên ứng dụng. Phương thức `getColor()` trong `ContextCompat` nhận hai đối số: ngữ cảnh hiện tại (`this`, tức thể hiện `Activity`), và tên màu.  
Việc triển khai phương thức này trong thư viện hỗ trợ ẩn đi các khác biệt trong các

phiên bản API khác nhau. Bạn có thể gọi phương thức này bất kể phiên bản compile SDK hay minimum SDK mà không có cảnh báo, lỗi hoặc sự cố.

- Đặt màu của TextView thành ID màu tài nguyên:
- Chạy ứng dụng trên thiết bị hoặc trình giả lập, sau đó nhấn vào nút "Change Color".

Nút này bây giờ sẽ thay đổi màu văn bản trong ứng dụng như hình bên dưới.

## Tổng kết

Cài đặt thư viện hỗ trợ Android:

- Sử dụng SDK Manager để cài đặt Android Support Repository. Chọn Tools > Android > SDK Manager, nhấn vào tab SDK Tools và mở rộng Support Repository.
- Nếu Installed xuất hiện trong cột Status, nhấn Cancel; nếu Not installed hoặc Update Available xuất hiện, đánh dấu chọn ô kiểm. Biểu tượng tải xuống sẽ xuất hiện bên cạnh ô kiểm. Nhấn OK.

Android sử dụng ba chỉ thị để chỉ ra cách ứng dụng của bạn hoạt động đối với các phiên bản API khác nhau:

- minSdkVersion: phiên bản API tối thiểu mà ứng dụng của bạn hỗ trợ.
- compileSdkVersion: phiên bản API mà ứng dụng của bạn phải được biên dịch.
- targetSdkVersion: phiên bản API mà ứng dụng của bạn được thiết kế cho.

Để quản lý các phụ thuộc trong dự án của bạn:

- Mở rộng Gradle Scripts trong ngăn Dự án > Android và mở tệp build.gradle (Module:app).
- Bạn có thể thêm các phụ thuộc vào phần phụ thuộc.

Lớp ContextCompat cung cấp các phương thức để tương thích với các phương thức liên quan đến ngữ cảnh và tài nguyên cho cả cấp độ API cũ và mới.

## Chạy ứng dụng

Mở ứng dụng HelloCompat mà bạn đã tạo trong phần thực hành về cách sử dụng các thư viện hỗ trợ.

- Đặt điểm dừng trình gỡ lỗi trên dòng trong phương thức changeColor() thực sự thay đổi màu:

- Chạy ứng dụng ở chế độ gỡ lỗi trên thiết bị hoặc trình giả lập đang chạy phiên bản API 23 trở lên. Nhấp vào Bước vào để bước vào phương thức getColor() và theo các lệnh gọi phương thức sâu hơn vào ngăn xếp. Kiểm tra cách lớp ContextCompat xác định cách lấy màu từ các tài nguyên và các lớp khung khác mà nó sử dụng.  
Một số lớp có thể tạo ra cảnh báo rằng "mã nguồn không khớp với mã byte". Nhấp vào Bước ra để quay lại tệp nguồn đã biết hoặc tiếp tục nhấp vào Bước vào cho đến khi trình gỡ lỗi tự trả về.
- Lặp lại bước trước đó cho thiết bị hoặc trình giả lập chạy phiên bản API cũ hơn 23. Lưu ý các đường dẫn khác nhau mà khuôn khổ thực hiện để hoàn thành việc lấy màu

## 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 thiết bị chạy Android bao gồm một hệ thống các đối tượng được gọi là views. Mỗi phần tử trên màn hình là một View.

Lớp View đại diện cho khối xây dựng cơ bản cho tất cả các thành phần giao diện người dùng. View là lớp cơ sở cho các lớp cung cấp các thành phần giao diện người dùng tương tác, chẳng hạn như các phần tử Button. Một Button là một phần tử giao diện người dùng 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 một ImageView, thành một phần tử giao diện người dùng có thể được nhấn hoặc bấm. Bạn phải lưu hình ảnh cho ImageView trong thư mục drawables của dự án.

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

##### Những gì bạn nên biết trước

Bạn nên có thể:

- Tạo một dự án Android Studio từ một mẫu và tạo giao diện chính.
- Chạy các ứng dụng trên trình giả lập hoặc thiết bị được kết nối.

- Tạo và chỉnh sửa các phần tử giao diện người dùng bằng trình chỉnh sửa giao diện và mã XML.
- Truy cập các phần tử giao diện người dùng từ mã của bạn bằng cách sử dụng findViewById().
- Xử lý sự kiện nhấp chuột của 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 như một phần tử tương tác để thực hiện một hành động.
- Cách thiết lập thuộc tính cho các phần tử ImageView trong trình chỉnh sửa giao diện.
- 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 như các phần tử tương tác.
- Thiết lập các trình xử lý onClick() cho các hình ảnh để hiển thị các thông báo Toast khác nhau.
- Thay đổi nút hành động nổi được cung cấp bởi mẫu để nó hiển thị một biểu tượng khác và khởi chạy một Activity khác.

### 1. Bắt đầu dự án mới

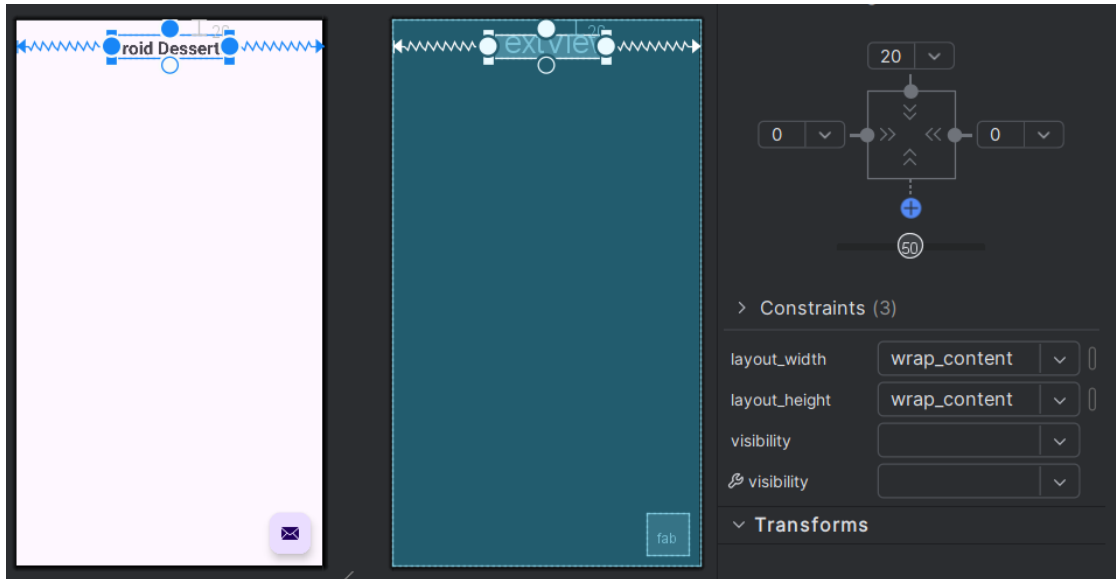
- Tạo một dự án mới trên Android Studio với tên ứng dụng là Droid Cafe.
- Chọn mẫu Basic Views Activity và chấp nhận tên Activity mặc định (MainActivity). Đảm bảo rằng các tùy chọn Generate Layout file và Backwards Compatibility (AppCompat) được chọn.
- Nhấp vào Finish.  
Dự án sẽ mở với hai bố cục trong thư mục res > layout: activity\_main.xml cho thanh công cụ và nút hành động nổi (không thay đổi trong nhiệm vụ này), và content\_main.xml cho tất cả những gì khác trong bố cục.
- Mở file content\_main.xml và nhấp vào tab Design (nếu chưa được chọn) để hiển thị trình chỉnh sửa bố cục.
- Chọn TextView "Hello World" trong bố cục và mở bảng Attributes.
- Thay đổi các thuộc tính textintro như sau:

Trường thuộc tính	Nhập thông tin sau
ID	textintro
text	Change Hello World to Droid Desserts

textStyle	B (bold)
textSize	24sp

Thêm thuộc tính android:id cho TextView với id được đặt là textintro, thay đổi văn bản, làm cho văn bản đậm, và đặt kích thước văn bản lớn hơn là 24sp.

- Xóa ràng buộc kéo dài từ dưới của TextView textintro đến cuối bố cục, để TextView được gắn vào phần trên của bố cục và chọn 8 (8dp) cho lề trên.



- Trong bài học trước, bạn đã học cách trích xuất một tài nguyên chuỗi từ một chuỗi văn bản gốc. Nhấp vào tab Text để chuyển sang mã XML và trích xuất chuỗi "Droid Desserts" thành tài nguyên chuỗi.

## 2. Thêm các 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 hoặc sử dụng hình ảnh của riêng bạn dưới dạng các tệp PNG, nhưng chúng phải có kích thước khoảng 113 x 113 pixel để sử dụng trong ví dụ này.

- Để sao chép các hình ảnh vào dự án của bạn, trước tiên hãy đóng dự án.
- Sao chép các tệp hình ảnh vào thư mục drawable của dự án. Tìm thư mục drawable theo đường dẫn: project\_name > app > src > main > res > drawable.
- Mở lại dự án.
- Mở file activity\_main.xml và nhấp vào tab Design (nếu chưa được chọn).

- Kéo một ImageView vào bố cục, chọn hình ảnh donut\_circle cho nó, và ràng buộc nó với TextView ở trên và cạnh trái của bố cục với khoảng cách 24 (24dp) cho cả hai ràng buộc.
- Trong bảng Attributes, nhập các giá trị thuộc tính cần thiết:

Trường thuộc tính	Nhập thông tin sau
ID	donut
contentDescription	Donuts are glazed and sprinkled with candy. (You can copy/paste the text into the field.)

- Kéo một ImageView thứ hai vào bố cục, chọn hình ảnh icecream\_circle cho nó, và ràng buộc nó với phần dưới của ImageView đầu tiên và cạnh trái của bố cục với khoảng cách 24 (24dp).
- Trong bảng Attributes, nhập các giá trị thuộc tính cần thiết:

Trường thuộc tính	Nhập thông tin sau
ID	ice_cream
contentDescription	Ice cream sandwiches have chocolate wafers and vanilla filling. (You can copy/paste the text into the field.)

- Kéo ImageView thứ ba vào bố cục, chọn hình ảnh froyo\_circle cho nó và giới hạn nó ở dưới cùng của ImageView thứ hai và ở phía bên trái của bố cục với lề của 24 (24dp) cho cả hai ràng buộc.
- Trong bảng Attributes, nhập các giá trị sau cho các thuộc tính:

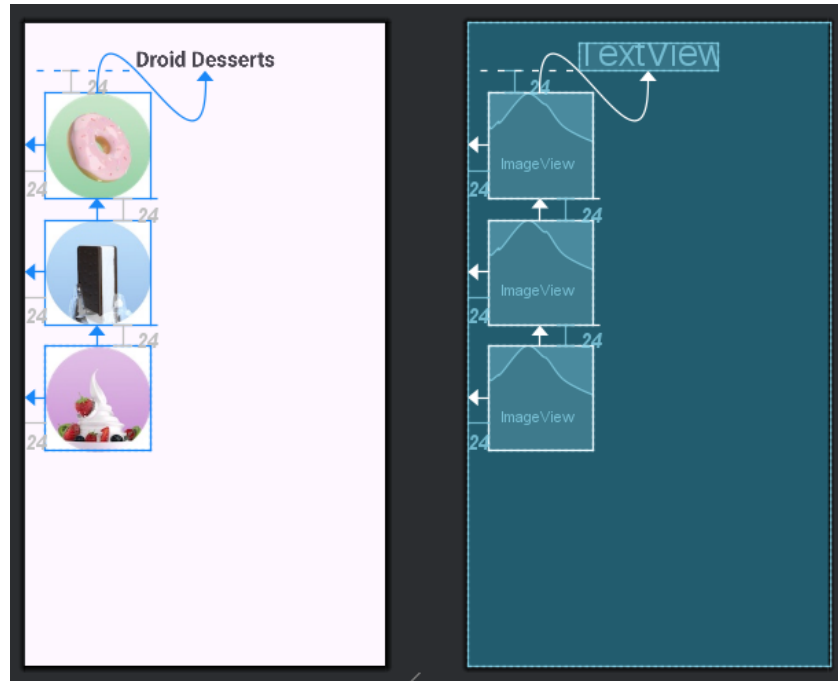
Trường thuộc tính	Nhập thông tin sau
ID	froyo
contentDescription	FroYo is premium self-serve frozen yogurt. (You can copy/paste the text into the field.)

- Nhấp vào biểu tượng cảnh báo ở góc trên bên trái của trình chỉnh sửa bố cục để mở ngăn cảnh báo, ngăn này sẽ hiển thị các cảnh báo về văn bản được mã hóa cứng
- Mở rộng từng cảnh báo Văn bản được mã hóa cứng, cuộn xuống cuối thông báo cảnh báo và nhấp vào nút Sửa  
Bản sửa cho mỗi cảnh báo văn bản được mã hóa cứng sẽ trích xuất tài nguyên chuỗi cho chuỗi. Hộp thoại Trích xuất tài nguyên xuất hiện và bạn có thể nhập tên cho tài nguyên chuỗi. Nhập các tên sau cho tài nguyên chuỗi

Chuỗi	Nhập tên sau
-------	--------------



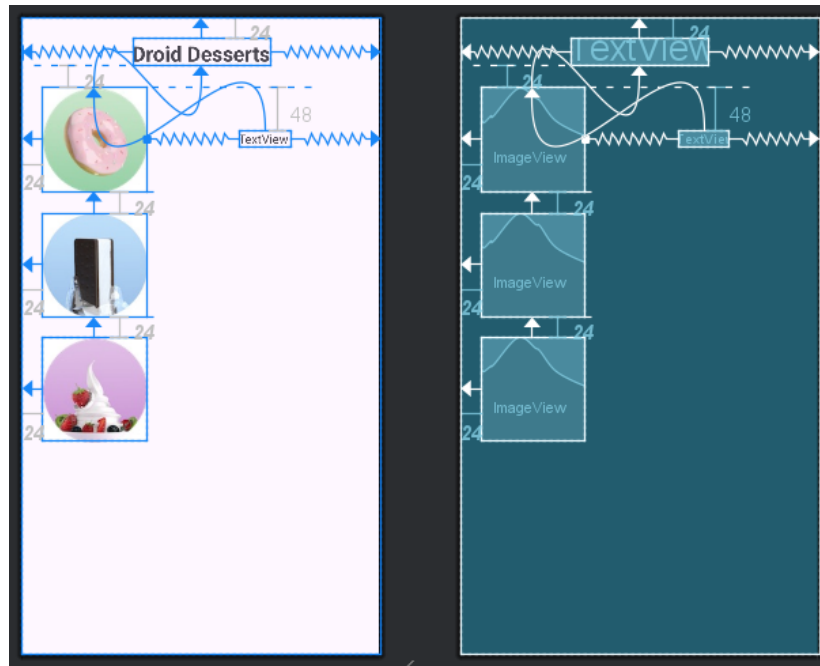
Donuts are glazed and sprinkled with candy.	donuts
Ice cream sandwiches have chocolate wafers and vanilla filling.	ice_cream_sandwiches
FroYo is premium self-serve frozen yogurt.	froyo



### 3. Thêm mô tả văn bản

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

- Kéo một phần tử `TextView` vào bố cục.
- Giới hạn phần bên trái của phần tử vào bên phải của `ImageView` hình bánh rán và phần trên của nó vào phần trên của `ImageView` hình bánh rán, cả hai đều có lẽ là 24 (24dp).
- Giới hạn phần bên phải của phần tử vào bên phải của bố cục và sử dụng cùng một ID là 24 (24dp). Nhập `donut_description` cho trường ID trong ngăn Thuộc tính. `TextView` mới sẽ xuất hiện bên cạnh hình bánh rán như trong hình bên dưới.



- Trong ngăn Thuộc tính, hãy thay đổi chiều rộng trong ngăn kiểm tra thành Match Constraints
- Trong ngăn Thuộc tính, hãy bắt đầu nhập tài nguyên chuỗi cho trường văn bản bằng cách thêm ký hiệu @ vào trước: @d. Nhập vào tên tài nguyên chuỗi (@string/donuts) xuất hiện dưới dạng
- Lặp lại các bước trên để thêm TextView thứ hai được giới hạn ở bên phải và trên cùng của ImageView hình kem và bên phải của nó ở bên phải của bố cục. Nhập nội dung sau vào ngăn Thuộc tính:

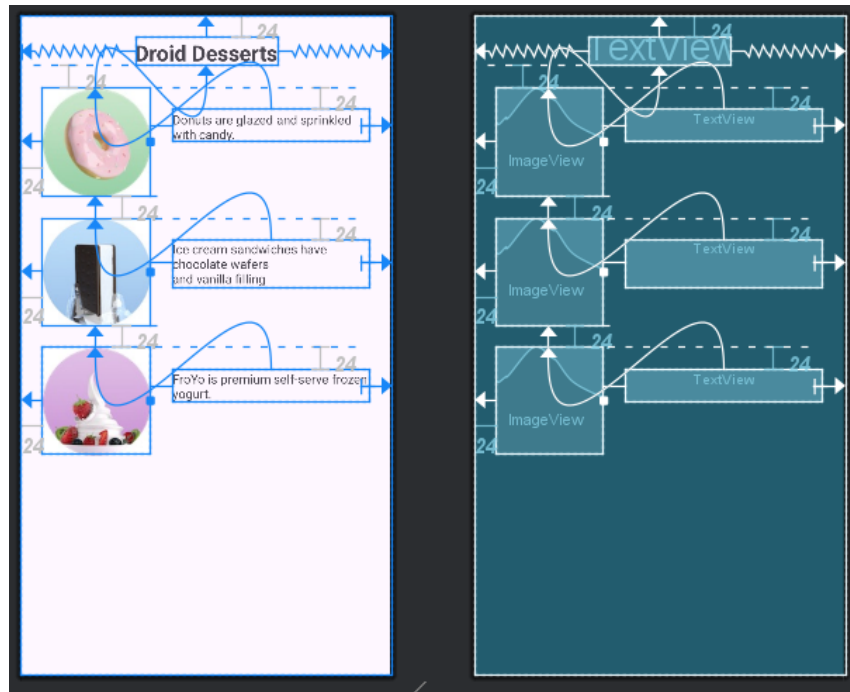
Trường thuộc tính	Nhập thông tin sau
ID	ice_cream_description
Left, right, and top margins	24
layout_width	match_constraint
text	@string/ice_cream_sandwiches

- Lặp lại các bước trên để thêm TextView thứ ba bị ràng buộc vào bên phải và trên cùng của ImageView froyo và bên phải của nó vào bên phải của bố cục. Nhập nội dung sau vào ngăn Thuộc tính:

Trường thuộc tính	Nhập thông tin sau
ID	froyo_description
Left, right, and top margins	24

layout_width	match_constraint
text	@string/froyo

Bố cục bây giờ sẽ trông như sau:



Mã code nhiệm vụ 1

Bố cục XML cho tệp 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:layout_width="match_parent"
android:layout_height="match_parent"
app:layout_behavior="@string/appbar_scrolling_view_behavior">
<TextView
    android:id="@+id/textintro"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/margin_regular"
    android:text="@string/intro_text"
    android:textSize="@dimen/text_heading"
    android:textStyle="bold"
```

```

app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
<ImageView
    android:id="@+id/donut"
    android:layout_width="120dp"
    android:layout_height="120dp"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    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="120dp"
    android:layout_height="120dp"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    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="120dp"
    android:layout_height="120dp"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    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_marginEnd="@dimen/margin_wide"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    android:text="@string/donuts"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/donut"
    app:layout_constraintTop_toTopOf="@+id/donut" />
<TextView

```

```
    android:id="@+id/ice_cream_description"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="@dimen/margin_wide"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    android:text="@string/ice_cream_sandwiches"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/ice_cream"
    app:layout_constraintTop_toTopOf="@+id/ice_cream" />
<TextView
    android:id="@+id/froyo_description"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="@dimen/margin_wide"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    android:text="@string/froyo"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/froyo"
    app:layout_constraintTop_toTopOf="@+id/froyo" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**1.2) Các điều khiển nhập liệu**

**1.3) Menu và bộ chọn**

**1.4) Điều hướng người dùng**

**1.5) RecyclerView**

**Bài 2) Trải nghiệm người dùng thú vị**

**2.1) Hình vẽ, định kiểu và chủ đề**

**2.2) Thẻ và màu sắc**

**2.3) Bộ cục thích ứng**

**Bài 3) Kiểm thử giao diện người dùng**

**3.1) Espresso cho việc kiểm tra UI**

## **CHƯƠNG 3. LÀM VIỆC TRONG NỀN**

**Bài 1) Các tác vụ nền**

**1.1) AsyncTask**

**1.2) AsyncTask và AsyncTaskLoader**

**1.3) Broadcast receivers**

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

**2.1) Thông báo**

**2.2) Trình quản lý cảnh báo**

**2.3) JobScheduler**

## **CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG**

**Bài 1) Tùy chọn và cài đặt**

**1.1) Shared preferences**

## **1.2) Cài đặt ứng dụng**

### **Bài 2) Lưu trữ dữ liệu với Room**

#### **2.1) Room, LiveData và ViewModel**

#### **2.2) Room, LiveData và ViewModel**