

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	1
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.....	34

1.3)	Trình chỉnh sửa bố cục	Error! Bookmark not defined.
1.4)	Văn bản và các chế độ cuộn.....	Error! Bookmark not defined.
1.5)	Tài nguyên có sẵn	Error! Bookmark not defined.
Bài 2)	Activities	Error! Bookmark not defined.
2.1)	Activity và Intent	Error! Bookmark not defined.
2.2)	Vòng đời của Activity và trạng thái	Error! Bookmark not defined.
2.3)	Intent ngầm định	Error! Bookmark not defined.
Bài 3)	Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ...	Error! Bookmark not defined.
3.1)	Trình gỡ lỗi.....	Error! Bookmark not defined.
3.2)	Kiểm thử đơn vị	Error! Bookmark not defined.
3.3)	Thư viện hỗ trợ.....	Error! Bookmark not defined.
CHƯƠNG 2. Trải nghiệm người dùng		
Bài 1)	Tương tác người dùng.....	Error! Bookmark not defined.
1.1)	Hình ảnh có thể chọn	Error! Bookmark not defined.
1.2)	Các điều khiển nhập liệu	Error! Bookmark not defined.
1.3)	Menu và bộ chọn.....	Error! Bookmark not defined.
1.4)	Điều hướng người dùng	Error! Bookmark not defined.
1.5)	RecyclerView	Error! Bookmark not defined.
Bài 2)	Trải nghiệm người dùng thú vị	Error! Bookmark not defined.
2.1)	Hình vẽ, định kiểu và chủ đề.....	Error! Bookmark not defined.
2.2)	Thẻ và màu sắc.....	Error! Bookmark not defined.
2.3)	Bố cục thích ứng	Error! Bookmark not defined.
Bài 3)	Kiểm thử giao diện người dùng	Error! Bookmark not defined.
3.1)	Espresso cho việc kiểm tra UI.....	Error! Bookmark not defined.
CHƯƠNG 3. Làm việc trong nền		
Bài 1)	Các tác vụ nền	622
1.1)	AsyncTask	Error! Bookmark not defined.
1.2)	AsyncTask và AsyncTaskLoader	Error! Bookmark not defined.
1.3)	Broadcast receivers	Error! Bookmark not defined.

Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ	nền	Error! Bookmark not defined.
2.1) Thông báo.....	Error! Bookmark not defined.	
2.2) Trình quản lý cảnh báo		675
2.3) JobScheduler	Error! Bookmark not defined.	
CHƯƠNG 4. Lưu trữ dữ liệu người dùng		690
Bài 1) Tùy chọn và cài đặt		690
1.1) Shared preferences		690
1.2) Cài đặt ứng dụng		695
Bài 2) Lưu trữ dữ liệu với Room		713
2.1) Room, LiveData và ViewModel		713
2.2) Room, LiveData và ViewModel		739

CHƯƠNG I : LÀM QUEN

Bài 1: Tạo ứng dụng đầu tiên

1.1.Android Studio và Hello World

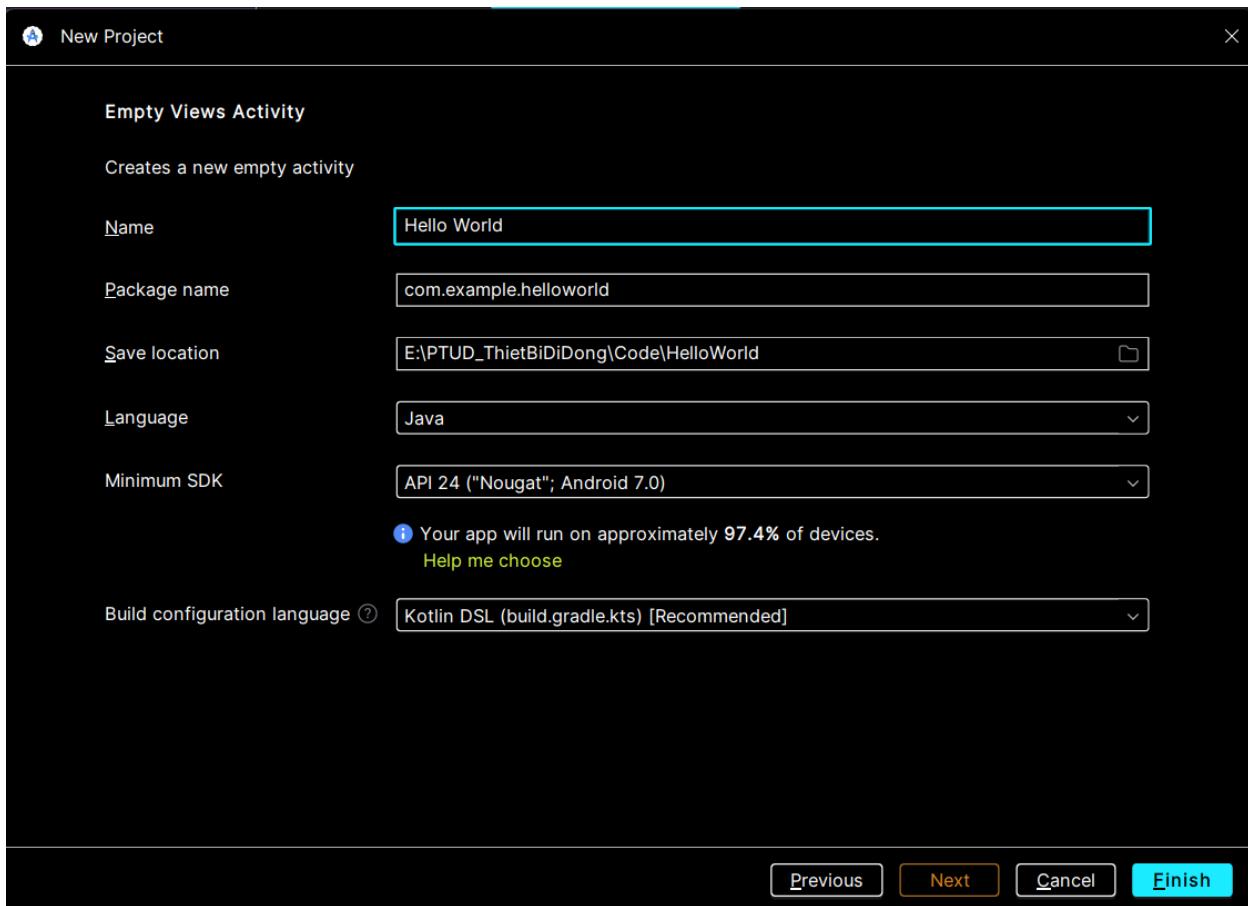
Giới thiệu

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.

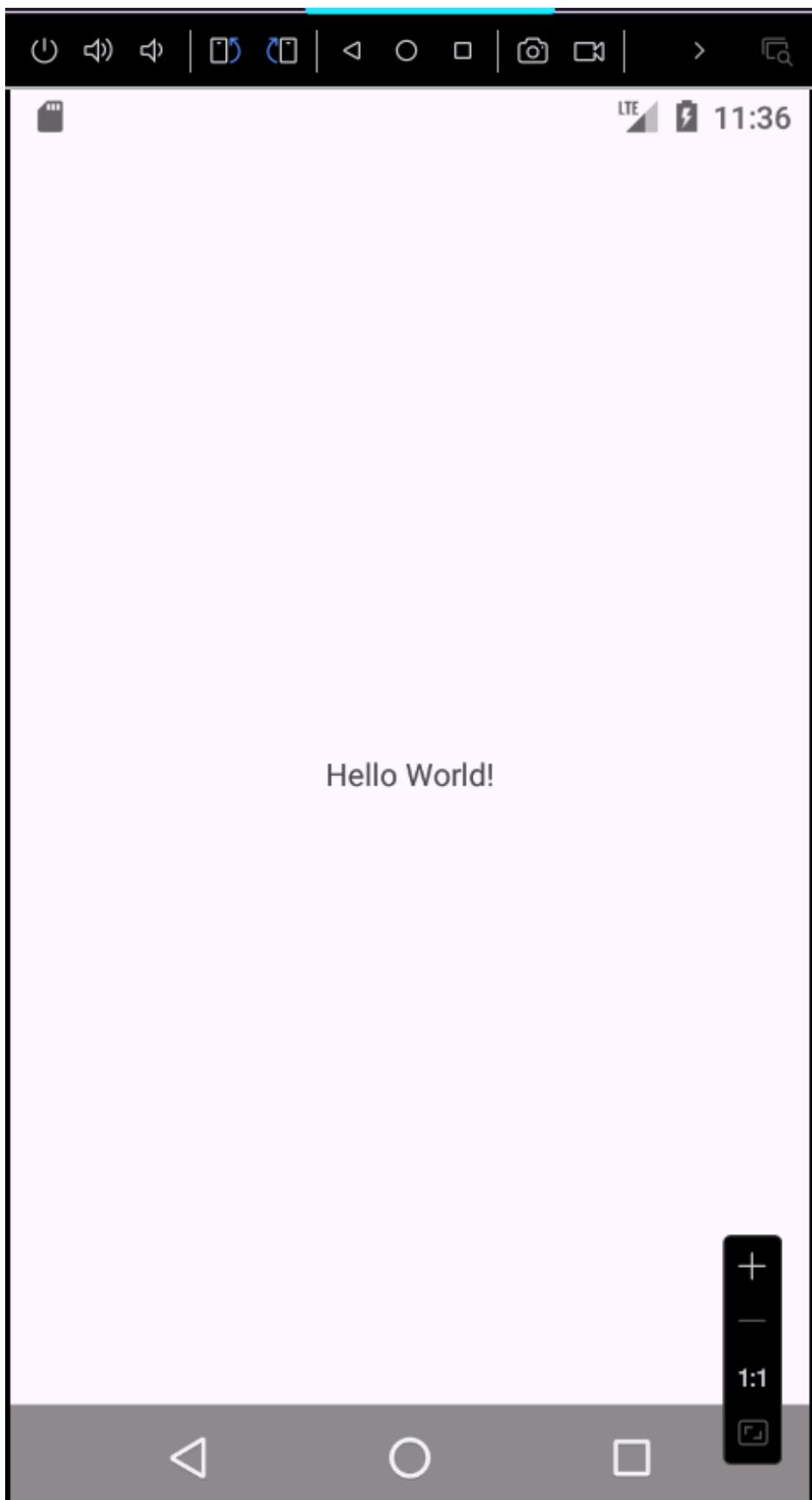
Những gì bạn sẽ làm

- 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**

Giới thiệu về ứng dụng

Sau khi cài đặt thành công Android Studio, bạn sẽ tạo một dự án mới từ mẫu có sẵn cho ứng dụng Hello World. Ứng dụng đơn giản này sẽ hiển thị dòng chữ “Hello World” trên màn hình của thiết bị Android (trên máy ảo và máy thật).

Hình ảnh của ứng dụng hoàn chỉnh sẽ trông như sau:



Bài 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển thích 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 hỗ trợ phát triển, gỡ lỗi, kiểm thử và tối ưu hiệu suất, giúp việc phát triển ứng dụng trở nên nhanh chóng 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 lên cửa hàng Google Play.

Lưu ý: Android Studio liên tục được cải tiến. Để biết thông tin mới nhất về yêu cầu hệ thống và hướng dẫn cài đặt, hãy xem tài liệu chính thức của Android Studio.

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

Để cài đặt và thiết lập Android Studio, trước tiên hãy kiểm tra yêu cầu hệ thống để đảm bảo thiết bị của bạn đáp ứng đủ điều kiện. Quy trình cài đặt tương tự trên tất cả các hệ điều hành, bất kỳ khác biệt nào sẽ được ghi chú riêng.

Các bước cài đặt:

1. Truy cập trang web chính thức của Android Developers và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
2. Chấp nhận các thiết lập mặc định trong suốt quá trình cài đặt, đồng thời đảm bảo rằng tất cả các thành phần cần thiết đều được chọn để cài đặt.
3. Sau khi quá trình cài đặt hoàn tất, Setup Wizard sẽ tiếp tục tải xuống và cài đặt một số thành phần bổ sung, bao gồm cả Android SDK. Hãy kiên nhẫn, vì quá trình này có thể mất thời gian tùy thuộc vào tốc độ Internet của bạn. Một số bước có thể trông giống nhau nhưng vẫn cần thiết.
4. Quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng để tạo dự án đầu tiên của mình.

Xử lý sự cố:

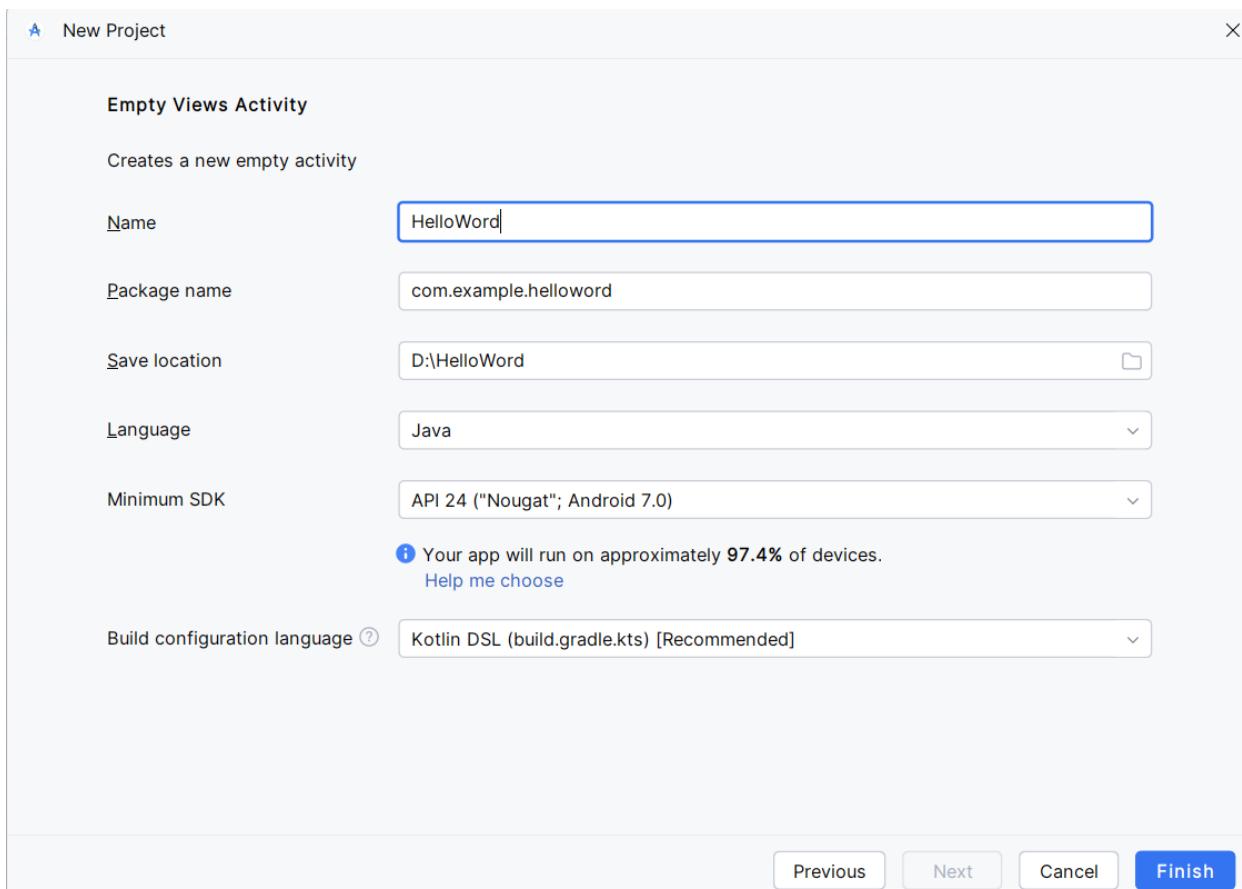
Nếu gặp sự cố khi cài đặt, hãy kiểm tra ghi chú phát hành của Android Studio hoặc tìm sự trợ giúp từ giảng viên của bạn.

Bài 2 : Tạo ứng dụng Hello Word

Trong bài này, bạn sẽ tạo một ứng dụng hiển thị ra dòng chữ “Hello Word” để xác minh rằng Android studio đã được cài đặt chính xác, và tìm hiểu những điều cơ bản của việc phát triển với Android Studio.

2.1. Tạo dự án ứng dụng

1. Mở Android Studio nếu nó chưa được mở
2. Trong cửa sổ chính Welcome to Android Studio, hãy nhấp vào Start a new Android Studio project.
3. Trong cửa sổ Create Android Project, hãy nhập Hello World cho tên ứng dụng



- 4 .Xác minh rằng vị trí Dự án mặc định là nơi bạn muốn lưu trữ ứng dụng Hello World và các dự án Android Studio khác hoặc thay đổi thành thư mục bạn muốn.

5.Chấp nhận android.example.com mặc định cho Tên miền công ty hoặc tạo một tên miền công ty duy nhất. Nếu bạn không có kế hoạch phát hành ứng dụng của mình, bạn có thể chấp nhận mặc định. Lưu ý rằng việc thay đổi tên gói ứng dụng của bạn sau này sẽ tốn thêm công sức.

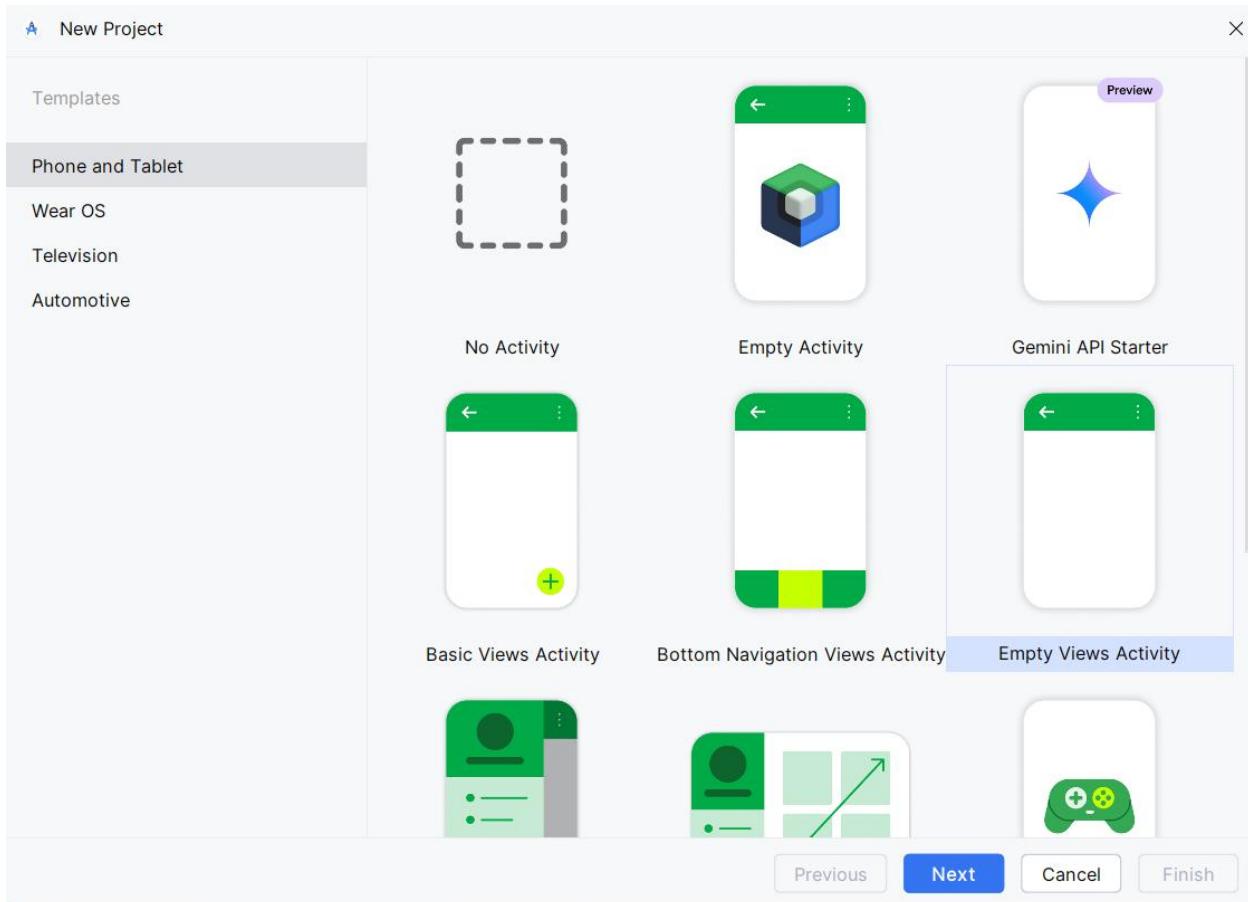
6. Bỏ chọn các tùy chọn Bao gồm hỗ trợ C++ và Bao gồm hỗ trợ Kotlin, rồi nhấp vào Tiếp theo.

7. Trên màn hình **Target Android Devices, Phone and Tablet** phải được chọn. Đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm SDK tối thiểu; nếu không, hãy sử dụng menu bật lên để đặt.

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

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

9. Cửa sổ **Add an Activity** xuất hiện. Hoạt động là một điều duy nhất, tập trung mà người dùng có thể thực hiện. Đây là thành phần quan trọng của bất kỳ ứng dụng Android nào. Hoạt động thường có bộ cục liên quan đến nó để xác định cách các thành phần UI xuất hiện trên màn hình. Android Studio cung cấp các mẫu Hoạt động để giúp bạn bắt đầu. Đối với dự án Hello World, hãy chọn **Empty Activity** như được hiển thị bên dưới và nhấp vào **Next**.



10. Màn hình **Configure Activity** xuất hiện (khác nhau tùy thuộc vào mẫu bạn đã chọn ở bước trước). Theo mặc định, Empty Activity do mẫu cung cấp được đặt tên là MainActivity. Bạn có thể thay đổi tên này nếu muốn, nhưng bài học này sử dụng MainActivity.

11. Đảm bảo rằng tùy chọn **Generate Layout file** được chọn. Tên bô cục theo mặc định là activity_main. Bạn có thể thay đổi tùy chọn này nếu muốn, nhưng bài học này sử dụng activity_main.

12. Đảm bảo rằng tùy chọn **Backwards Compatibility (App Compat)** được chọn. Điều này đảm bảo rằng ứng dụng của bạn sẽ tương thích ngược với các phiên bản Android trước đó.

13. Nhấp vào Finish.

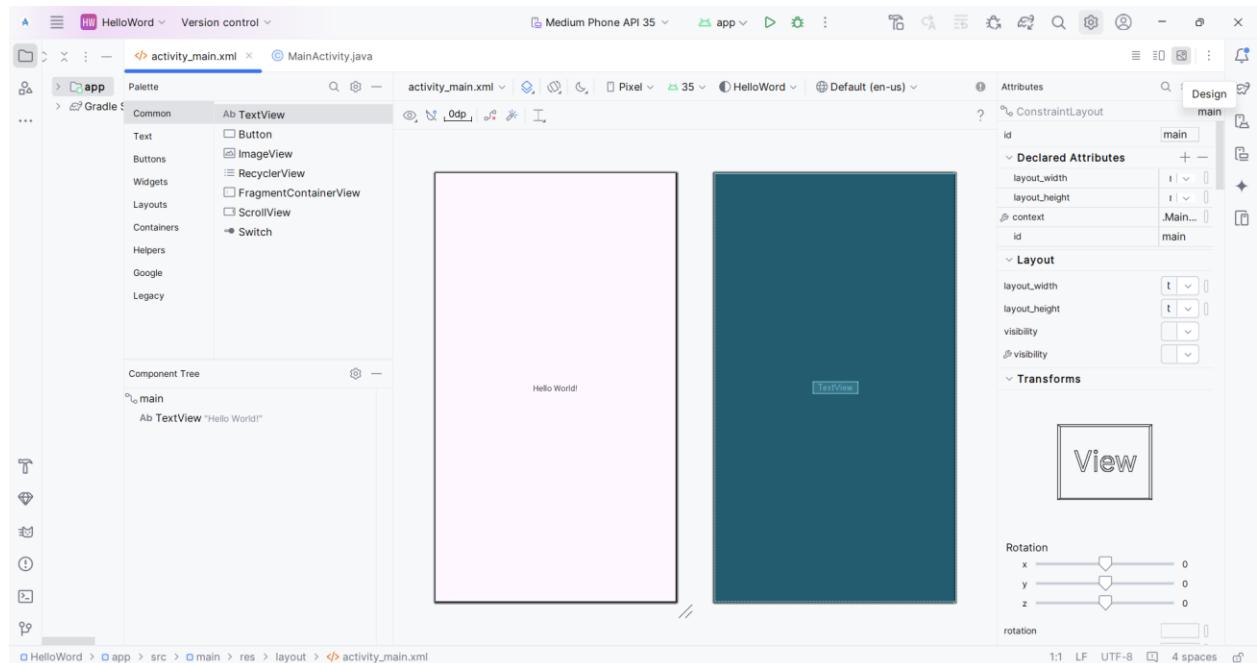
Android Studio tạo một thư mục cho các dự án của bạn và xây dựng dự án bằng Gradle (có thể mất vài phút).

Mẹo: Xem trang "Cấu hình bản dựng" (Configure your build) dành cho nhà phát triển để biết thông tin chi tiết. Bạn cũng có thể thấy thông báo "Mẹo trong

ngày"(Tip of the day) với các phím tắt và các mẹo hữu ích khác. Nhấp vào **Close** để đóng thông báo.

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

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



3. Nhấp vào tab MainActivity.java để mở trình chỉnh sửa mã (code editor), như được hiển thị bên dưới.

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** Shows "HelloWord" and "Version control".
- Toolbar:** Includes icons for file operations, search, and navigation.
- Code Editor:** Displays the Java code for `MainActivity.java`. The code handles window insets for a full-screen activity.

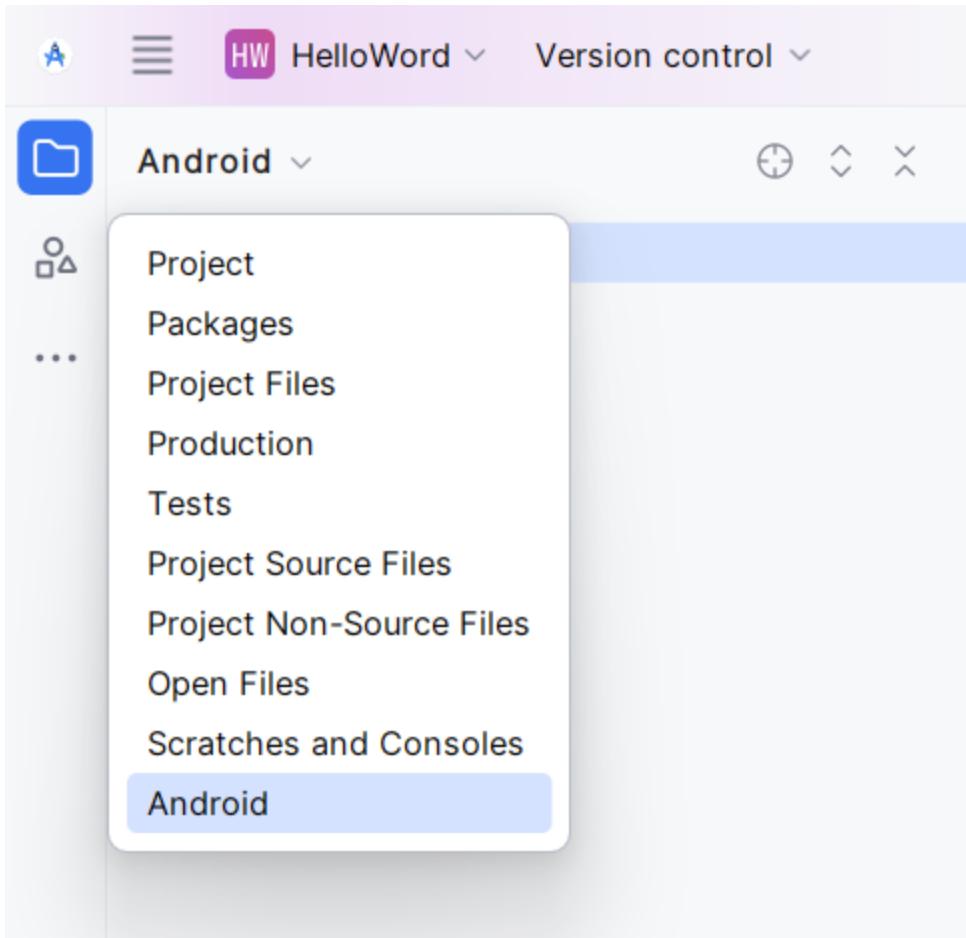
```
1 package com.example.helloworld;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        EdgeToEdge.enable($this$enableEdgeToEdge: this);
11     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

- Project Navigators:** On the left, there's a tree view of the project structure under "app".
- Bottom Navigation:** Shows the current file path: "HelloWord > app > src > main > java > com > example > helloworld > MainActivity".
- Status Bar:** Shows "1:1 LF UTF-8 4 spaces" and "0:07 DA".

2.2. Khám phá Project > Android pane

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

1. Nếu chưa được chọn, nhấp vào tab **Project** trong cột tab dọc ở phía bên trái cửa sổ Android Studio. **Project pane** sẽ xuất hiện.
2. Để xem dự án theo cấu trúc thư mục chuẩn của Android, chọn **Android** từ menu bật lên ở đầu **Project pane**, như minh họa bên dưới.

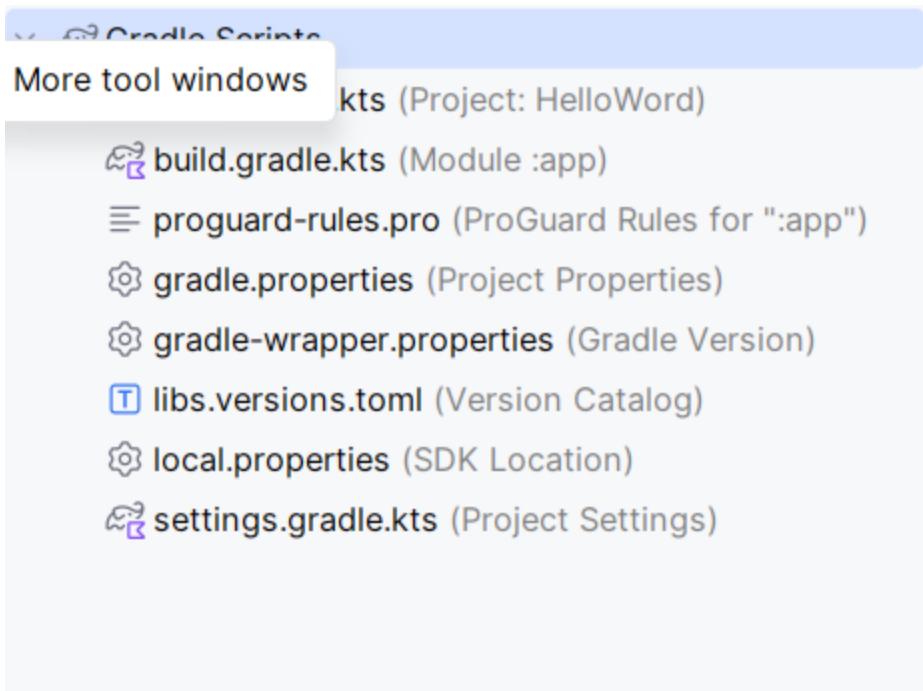


Lưu ý: Chương này và các chương khác đề cập đến **Project pane**, khi được đặt ở chế độ **Android**, sẽ được gọi là **Project > Android pane**.

2.3. Khám phá thư mục Gradle Scripts

Hệ thống build Gradle trong Android Studio giúp bạn dễ dàng thêm các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác làm dependencies vào quá trình build của bạn.

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



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

1. Mở rộng thư mục Gradle Scripts:

- Nếu thư mục **Gradle Scripts** chưa được mở rộng, nhấp vào biểu tượng tam giác để mở rộng nó.
- Thư mục này chứa tất cả các tệp cần thiết cho hệ thống build.

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

- Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô-đun trong dự án của mình.
- Mỗi dự án Android Studio đều chứa một tệp Gradle build cấp cao nhất. Thông thường, bạn không cần thay đổi tệp này, nhưng vẫn nên hiểu nội dung của nó.

Cấu trúc mặc định của tệp build cấp cao nhất:

- Tệp này sử dụng khôi buildscript để định nghĩa các kho lưu trữ (repositories) và dependencies chung cho tất cả các mô-đun trong dự án.

- Nếu dependency của bạn không phải là thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm kiếm các tệp trong các kho trực tuyến được chỉ định trong khôi repositories.

Mặc định, các dự án mới trong Android Studio sẽ khai báo các kho lưu trữ sau:

- **JCenter**
- **Google** (bao gồm Google Maven repository).

```

1 // Top-level build file where you can add configuration options common to all sub-projects and
2 plugins {
3     alias(libs.plugins.android.application) apply false
4 }
```

3. Tìm tệp build.gradle (Module: app)

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

Mục đích sử dụng

- Đây thường là tệp mà bạn sẽ chỉnh sửa nhiều nhất khi thay đổi các cấu hình ở cấp ứng dụng, chẳng hạn như khai báo dependencies trong phần dependencies.

Khai báo dependencies

- Bạn có thể khai báo một dependency thư viện bằng một trong nhiều cách khác nhau. Mỗi cách khai báo dependency cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện.
 - Ví dụ:
 - Lệnh implementation fileTree(dir: 'libs', include: ['*.jar']) thêm dependency cho tất cả các tệp ".jar" bên trong thư mục libs.

Ví dụ tệp build.gradle (Module: app)

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

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

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

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

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

    buildTypes {
        release {
            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)  
}
```

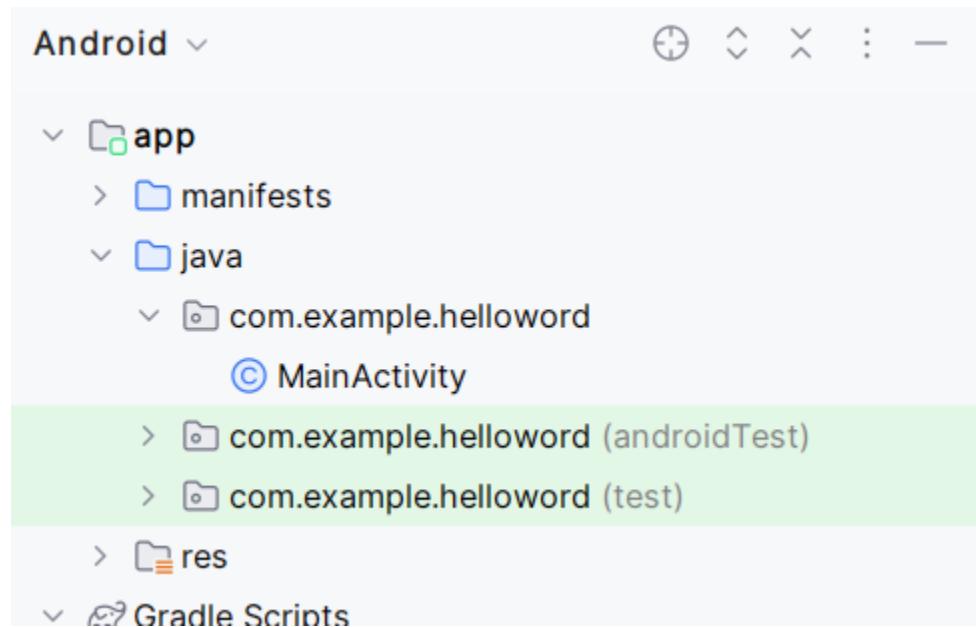
3. Nhập vào biểu tượng tam giác để đóng thư mục Gradle Scripts.

2.4 Khám phá thư mục app và res

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

1. Mở rộng thư mục **app**, sau đó mở rộng thư mục **java**, và tiếp tục mở rộng thư mục **com.example.android.helloworld** để xem tệp Java MainActivity.

- Nhấp đúp vào tệp để mở nó trong trình chỉnh sửa mã (code editor).



Thư mục java

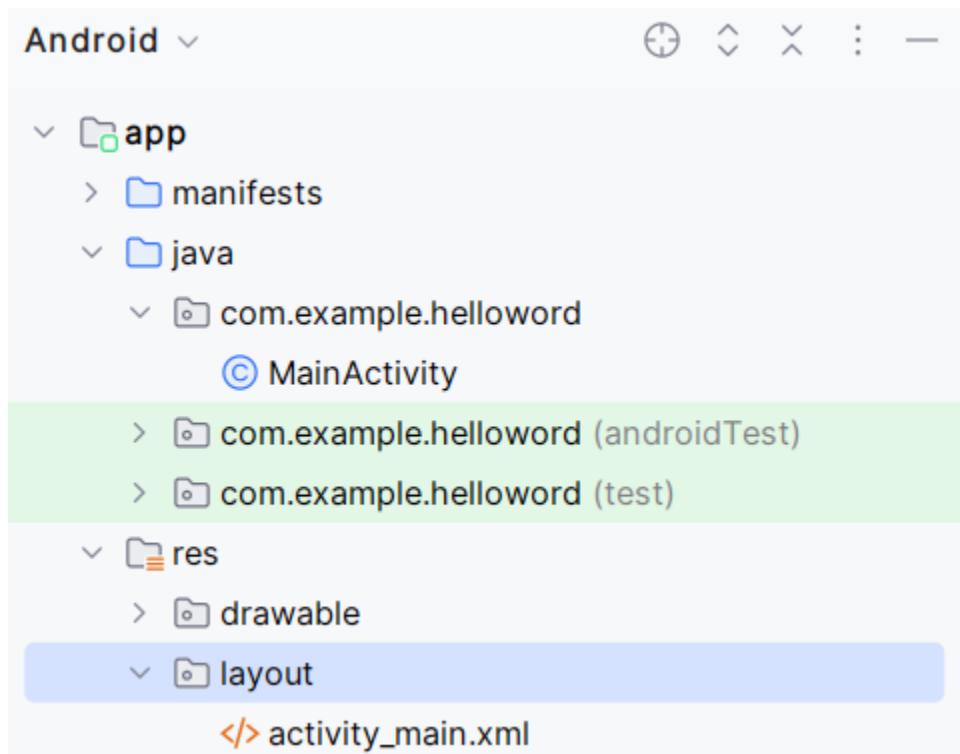
Thư mục **java** bao gồm các tệp lớp Java trong ba thư mục con, như minh họa trong hình trên:

- **Thư mục com.example.hello.helloworld** (hoặc tên miền mà bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng.
- **Hai thư mục khác** được sử dụng cho việc kiểm thử và sẽ được mô tả trong một bài học khác.

Trong ứng dụng Hello World, chỉ có một gói và nó chứa tệp MainActivity.java.

- Tệp này đại diện cho **Activity đầu tiên** (màn hình đầu tiên) mà người dùng nhìn thấy.
- Tệp MainActivity cũng khởi tạo các tài nguyên toàn cục của ứng dụng.
- Trong **Project > Android pane**, phần mở rộng tệp (.java) được bỏ qua để hiển thị ngắn gọn.

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



Thư mục **res** chứa các tài nguyên (resources) của ứng dụng, chẳng hạn như **Layouts**: Các bộ cục giao diện người dùng (UI), **Strings**: Các chuỗi văn bản, **Images**: Các hình ảnh. Mỗi **Activity** thường được liên kết với một bộ cục UI, được định nghĩa trong một tệp XML. Tệp bộ cục này thường được đặt tên theo tên của **Activity** tương ứng.

2.5 Khám phá thư mục manifests

Thư mục **manifests** chứa các tệp cung cấp thông tin quan trọng về ứng dụng của bạn cho hệ thống Android. Thông tin này cần thiết để hệ thống có thể chạy mã của ứng dụng.

Các bước thực hiện:

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

Thông tin về tệp AndroidManifest.xml

- Tệp **AndroidManifest.xml** mô tả tất cả các thành phần của ứng dụng Android, chẳng hạn như:
 - Các **Activity**.
 - Các **Service**, **BroadcastReceiver**, và **ContentProvider**.
- Mỗi thành phần của ứng dụng phải được khai báo trong tệp này.

Trong các bài học khác, bạn sẽ chỉnh sửa tệp này để:

- Thêm tính năng.
- Cấp quyền sử dụng tính năng cho ứng dụng.

Để tìm hiểu thêm, xem [App Manifest Overview](#).

1.1 Android Studio và Hello world

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

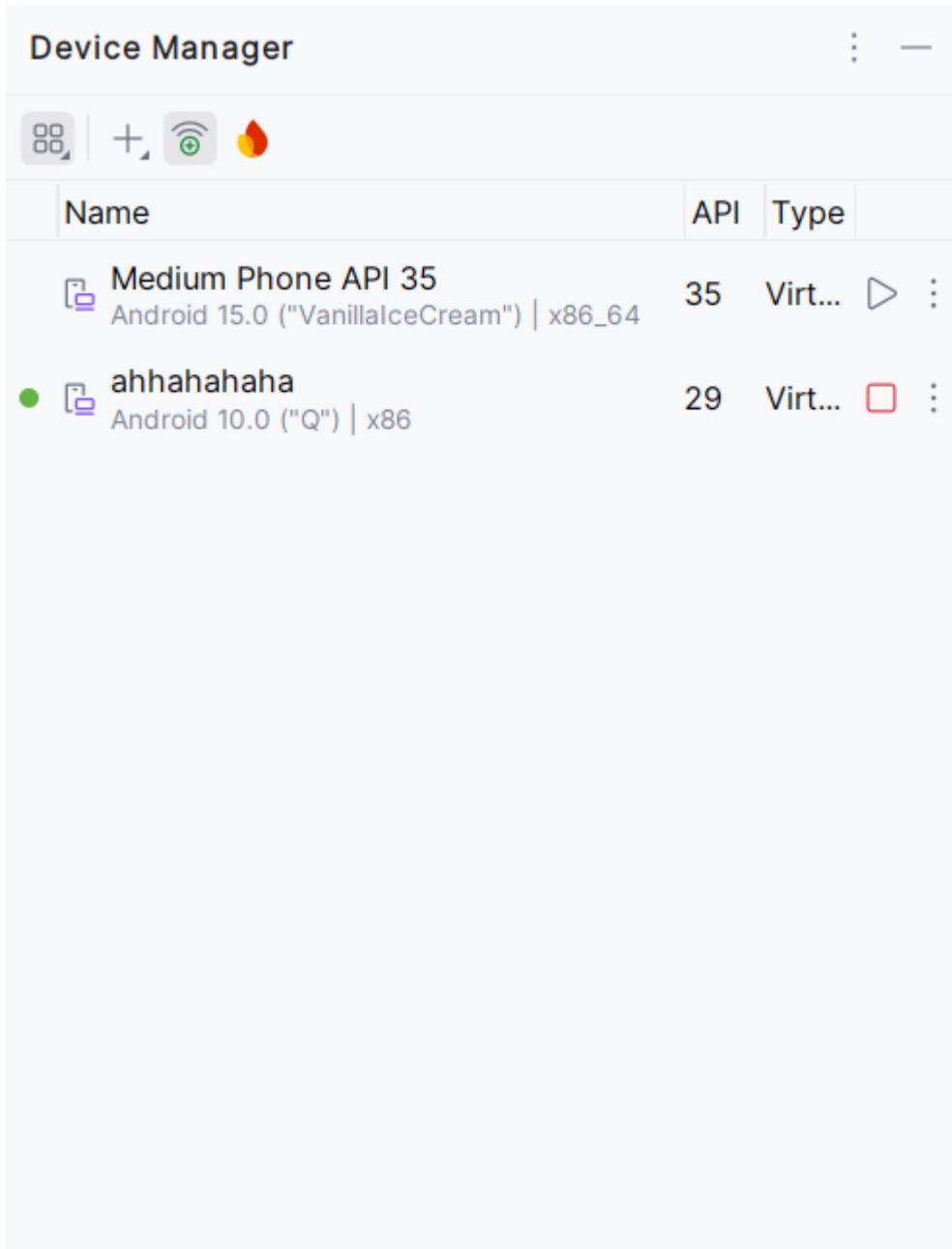
Sử dụng **AVD Manager**, bạn có thể xác định các đặc điểm phần cứng của thiết bị, cấp độ API, bộ nhớ, giao diện và các thuộc tính khác, sau đó lưu nó dưới dạng một

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

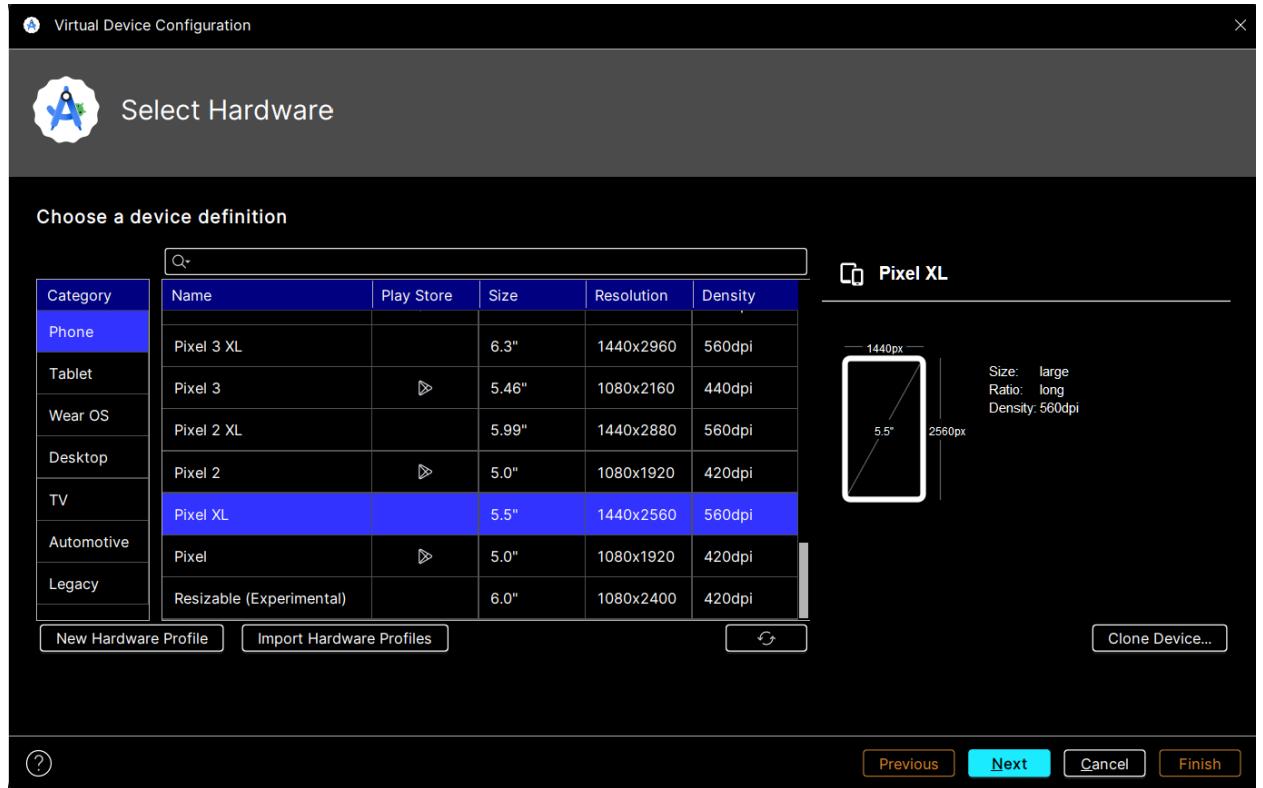
Giao diện người dùng tương tác đầu tiên

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

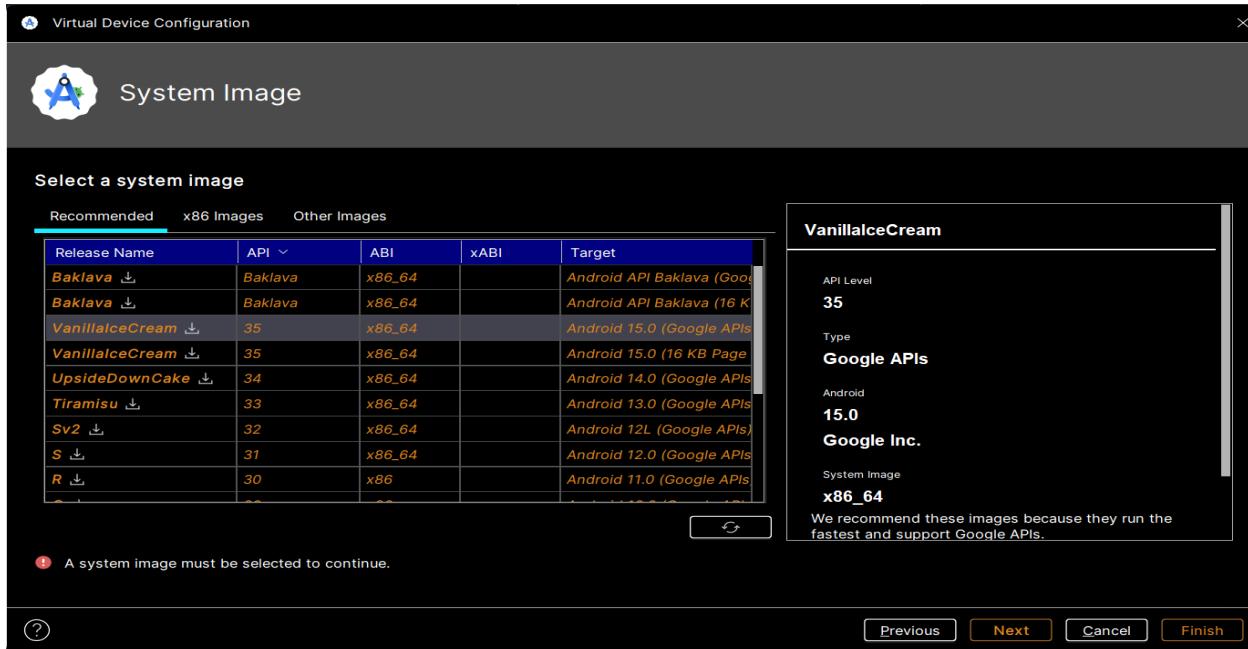
1. Trong **Android Studio**, chọn **Tools > Android > AVD Manager**, hoặc nhập vào biểu tượng **AVD Manager** trên thanh công cụ.



2. Nhấp vào + **Create Virtual Device**. Cửa sổ **Select Hardware** xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Mỗi thiết bị có các thông số như kích thước màn hình theo đường chéo (**Size**), độ phân giải màn hình tính theo pixel (**Resolution**), và mật độ điểm ảnh (**Density**).



3. Chọn một thiết bị, chẳng hạn như **Nexus 5X** hoặc **Pixel XL**, rồi nhấp vào **Next**. Màn hình **System Image** xuất hiện.
4. Nhấp vào tab **Recommended** (nếu chưa được chọn) và chọn phiên bản hệ điều hành Android mà bạn muốn chạy trên thiết bị ảo (chẳng hạn như **Oreo**).



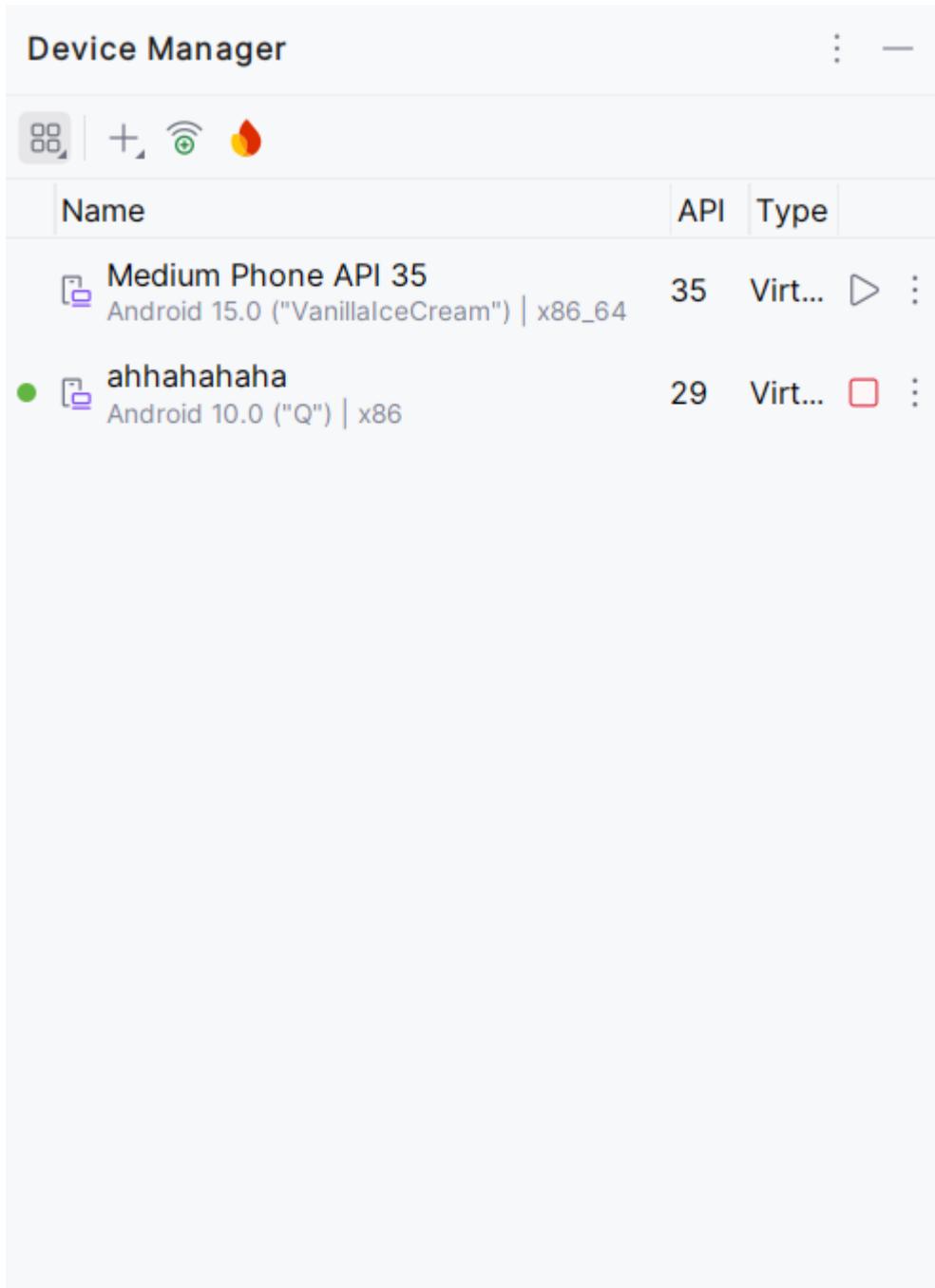
Có nhiều phiên bản hơn so với những gì được hiển thị trong tab **Recommended**. Bạn có thể xem thêm trong các tab **x86 Images** và **Other Images**. Nếu có liên kết **Download** bên cạnh hình ảnh hệ thống mà bạn muốn sử dụng, điều đó có nghĩa là nó chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống, sau đó nhấp **Finish** khi hoàn tất.

- Sau khi chọn xong hệ thống, nhấp vào **Next**. Cửa sổ **Android Virtual Device (AVD)** xuất hiện. Bạn có thể thay đổi tên của thiết bị ảo nếu muốn. Kiểm tra lại cấu hình và nhấp vào **Finish**.

3.2 Chạy ứng dụng trên thiết bị ảo

Trong nhiệm vụ này, 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.
- Cửa sổ **Select Deployment Target** xuất hiện. Trong phần **Available Virtual Devices**, chọn thiết bị ảo mà bạn vừa tạo và nhấp vào **OK**.



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

Bạn sẽ thấy ứng dụng **Hello World** hiển thị như trong hình minh họa bên dưới.



Mẹo: Khi kiểm thử trên thiết bị ảo, tốt nhất bạn nên khởi động nó một lần ngay từ đầu phiên làm việc và không đóng nó cho đến khi bạn hoàn thành việc kiểm thử. Điều này giúp tránh việc ứng dụng phải trải qua quá trình khởi động thiết bị nhiều lần.

Để đóng thiết bị ảo, bạn có thể:

- Nhấp vào nút **X** ở góc trên của trình giả lập.
- Chọn **Quit** từ menu.
- Nhấn **Control + Q** trên Windows hoặc **Command + Q** trên macOS.

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, chẳng hạn như điện thoại hoặc máy tính bảng. Bạn nên luôn kiểm thử ứng dụng của mình trên cả thiết bị ảo và thiết bị vật lý.

Bạn cần có:

- Một thiết bị Android, chẳng hạn 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 thống **Linux** hoặc **Windows**, có thể cần thực hiện thêm một số bước để chạy ứng dụng trên thiết bị phần cứng.
 - Kiểm tra tài liệu **Using Hardware Devices**.
 - Bạn cũng có thể cần cài đặt **trình điều khiển USB** phù hợp với thiết bị của mình.
 - Đối với trình điều khiển USB trên Windows, hãy xem **OEM USB Drivers**.

Bật chế độ Gỡ lỗi USB (USB Debugging)

Để **Android Studio** có thể giao tiếp với thiết bị của bạn, bạn phải bật **Gỡ lỗi USB** trên thiết bị Android. Tùy chọn này nằm trong phần **Developer options** của thiết bị.

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

1. Trên thiết bị, mở **Cài đặt (Settings)**, tìm **Giới thiệu về điện thoại (About phone)**, nhấn vào đó và chạm vào **Số bản dựng (Build number) 7 lần** liên tiếp.
2. Quay lại màn hình trước đó (**Cài đặt / Hệ thống (Settings / System)**). Mục **Developer options** sẽ xuất hiện trong danh sách. Nhấn vào **Developer options**.
3. Bật tùy chọn **USB Debugging**.

Chạy ứng dụng trên thiết bị

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

1. **Kết nối** thiết bị với máy tính bằng cáp USB.
2. Nhấp vào **Run** trên thanh công cụ.
 - o Cửa sổ **Select Deployment Target** sẽ mở ra, hiển thị danh sách trình giả lập và các thiết bị được kết nối.
3. **Chọn thiết bị** của bạn và nhấp **OK**.
 - o **Android Studio** sẽ cài đặt và chạy ứng dụng trên thiết bị của bạn.

Xử lý sự cố (Troubleshooting)

Nếu **Android Studio** không nhận diện được thiết bị, hãy thử các cách sau:

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

Nếu máy tính vẫn không tìm thấy thiết bị hoặc hiển thị trạng thái "**unauthorized**", hãy làm như sau:

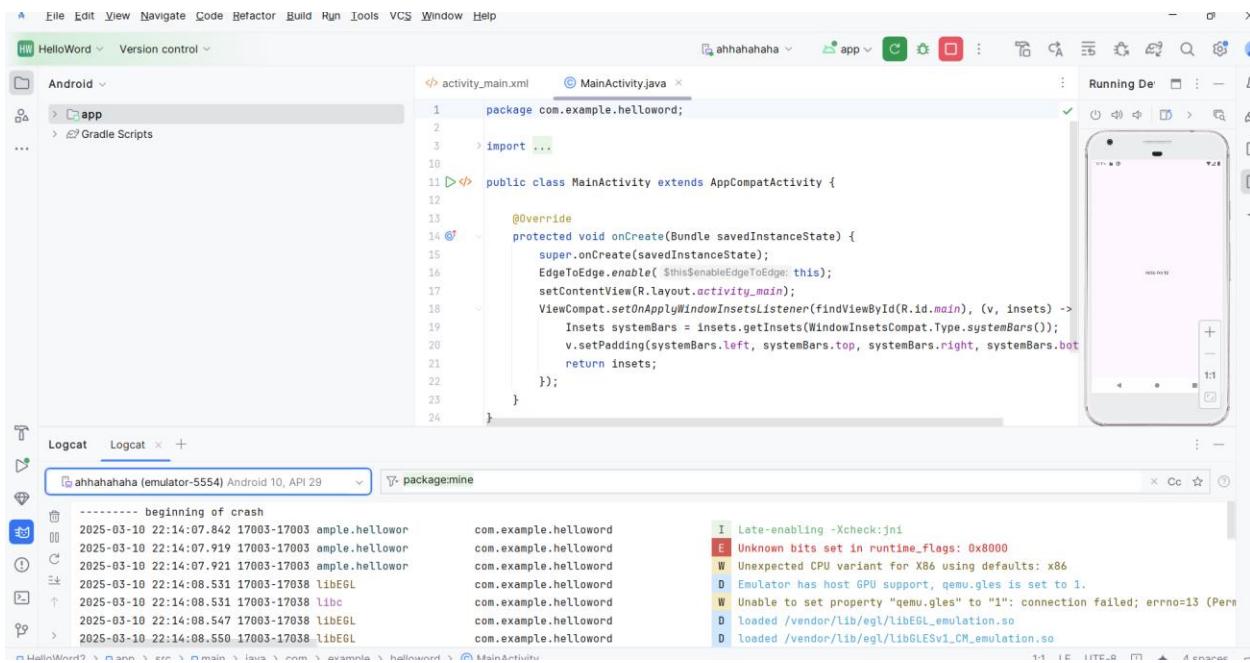
1. Rút kết nối thiết bị.
2. Trên thiết bị, mở **Developer Options** trong ứng dụng **Cài đặt (Settings)**.
3. Nhấn vào **Revoke USB Debugging authorizations**.
4. Kết nối lại thiết bị với máy tính.
5. Khi có thông báo yêu cầu cấp quyền, hãy chọn **Cho phép (Allow)**.

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

6.1 Xem Logcat pane

Để xem **Logcat pane**, hãy nhấp vào tab **Logcat** ở cuối cửa sổ **Android Studio**, như minh họa trong hình bên dưới

Trong hình minh họa:



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

Thêm câu lệnh log vào ứng dụng của bạn

Các câu lệnh **Log** trong mã nguồn của ứng dụng sẽ hiển thị thông báo trong **Logcat pane**. Ví dụ:

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

Các thành phần của thông báo:

- **Log**: Lớp **Log** dùng để gửi thông báo log đến **Logcat pane**.
- **d**: Cấp độ **Debug** của **Log**, dùng để lọc thông báo trong **Logcat pane**.

- Các cấp độ log khác:
 - e: Error (Lỗi)
 - w: Warn (Cảnh báo)
 - i: Info (Thông tin)
- "MainActivity": Đối số đầu tiên là **tag**, giúp lọc thông báo trong **Logcat pane**.
 - Thông thường, đây là tên của **Activity** chứa thông báo.
 - Tuy nhiên, bạn có thể đặt bất kỳ giá trị nào có ích cho việc gỡ lỗi.
 - Theo quy ước, **log tag** thường được định nghĩa là **hằng số** trong

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

Activity.

- "Hello world": Đối số thứ hai là nội dung của thông báo.

Các bước thực hiện:

1. Mở ứng dụng **Hello World** trong **Android Studio** và mở tệp **MainActivity**.
2. Để tự động thêm các **import** cần thiết vào dự án (chẳng hạn như **android.util.Log** để sử dụng **Log**), thực hiện:
 - Windows: Chọn **File > Settings**
 - macOS: Chọn **Android Studio > Preferences**
3. Chọn **Editor > General > Auto Import**.
 - Tích chọn **tất cả** các hộp kiểm và đặt **Insert imports on paste** thành **All**.
4. Nhập **Apply**, sau đó nhấp **OK**.
5. Trong phương thức **onCreate()** của **MainActivity**, thêm dòng lệnh sau:

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

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

```
public class MainActivity extends AppCompatActivity {

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

6. Nếu **Logcat pane** chưa mở, nhập vào tab **Logcat** ở cuối cửa sổ **Android Studio** để mở.
7. Kiểm tra xem **tên ứng dụng** và **tên gói (package name)** có chính xác không.
8. Thay đổi **Log level** trong **Logcat pane** thành **Debug** (hoặc giữ nguyên **Verbose** nếu có ít thông báo log).
9. Chạy ứng dụng của bạn.

Sau khi chạy, bạn sẽ thấy thông báo xuất hiện trong **Logcat pane**.

2025-03-10 22:26:36.151 17228-17228 MainActivity com.example.helloworld D Hello World!

Thử thách lập trình

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

Thử thách:

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

1. Tạo một dự án mới trong Android Studio.
2. Thay đổi dòng chữ "Hello World" thành "Happy Birthday to " kèm theo tên của một người vừa có sinh nhật gần đây.

3. (**Tùy chọn**) Chụp ảnh màn hình ứng dụng hoàn thành và gửi email cho ai đó mà bạn đã quên chúc mừng sinh nhật.
 4. Một cách sử dụng phổ biến của lớp Log là ghi lại các ngoại lệ (**exceptions**) trong chương trình Java khi chúng xảy ra. Có một số phương thức hữu ích như Log.e() mà bạn có thể dùng cho mục đích này. Hãy khám phá các phương thức cho phép bạn bao gồm một ngoại lệ trong một thông báo Log. Sau đó, viết mã trong ứng dụng của bạn để tạo ra và ghi lại một ngoại lệ.
-

Tóm tắt

- Để cài đặt Android Studio, truy cập [Android Studio](#) và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo một ứng dụng mới, đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm **Minimum SDK**.
- Để xem cấu trúc Android của ứng dụng trong **Project pane**, nhấp vào tab **Project** ở cột dọc và chọn **Android** trong menu thả xuống ở trên cùng.
- Chính sửa tệp **build.gradle(Module:app)** khi bạn cần thêm thư viện mới hoặc thay đổi phiên bản thư viện trong dự án.
- Tất cả mã nguồn và tài nguyên của ứng dụng đều nằm trong thư mục **app** và **res**. Thư mục **java** chứa các activity, test, và các thành phần khác trong mã nguồn Java. Thư mục **res** chứa các tài nguyên như layout, chuỗi ký tự (**strings**) và hình ảnh (**images**).
- Chính sửa tệp **AndroidManifest.xml** để thêm các thành phần (**components**) và quyền truy cập (**permissions**) cho ứng dụng Android của bạn. Mọi thành phần của ứng dụng, chẳng hạn như nhiều **activity**, phải được khai báo trong tệp XML này.
- Sử dụng **Android Virtual Device (AVD) manager** để tạo một thiết bị ảo (**emulator**) để chạy ứng dụng.
- Thêm các câu lệnh **Log** vào ứng dụng để hiển thị thông báo trong **Logcat pane**, giúp bạn gỡ lỗi (**debugging**) dễ dàng hơn.
- Để chạy ứng dụng trên thiết bị Android thực tế bằng Android Studio, bật **USB Debugging** trên thiết bị.
 - Mở **Settings > About phone** và nhấn vào **Build number** bảy lần.
 - Quay lại màn hình trước (**Settings**) và chọn **Developer options**.
 - Bật **USB Debugging**.

Các khái niệm liên quan

Tài liệu về các khái niệm liên quan có trong:

- **1.0: Giới thiệu về Android**
 - **1.1: Ứng dụng Android đầu tiên của bạn**
-

Tìm hiểu thêm

Tài liệu về Android Studio:

- [Trang tải xuống Android Studio](#)
- [Ghi chú phát hành Android Studio](#)
- [Giới thiệu về Android Studio](#)
- [Công cụ dòng lệnh Logcat](#)
- [Trình quản lý thiết bị ảo \(AVD Manager\)](#)
- [Tổng quan về Android Manifest](#)
- [Cấu hình Gradle](#)
- [Lớp Log trong Android](#)
- [Tạo và quản lý thiết bị ảo](#)

Khác:

- Làm thế nào để cài đặt Java?
 - [Cài đặt phần mềm JDK và thiết lập biến JAVA_HOME](#)
 - [Trang chủ Gradle](#)
 - Cú pháp Apache Groovy
 - [Trang Wikipedia về Gradle](#)
-

Bài tập về nhà

Xây dựng và chạy một ứng dụng

1. Tạo một dự án Android mới từ **Empty Template**.
 2. Thêm các câu lệnh Log cho nhiều cấp độ nhật ký khác nhau (**log levels**) trong phương thức `onCreate()` của activity chính.
 3. Tạo một trình giả lập (**emulator**) cho một thiết bị, chọn phiên bản Android tùy ý, và chạy ứng dụng trên đó.
 4. Sử dụng bộ lọc trong **Logcat** để tìm các câu lệnh Log của bạn và điều chỉnh mức hiển thị chỉ để hiển thị **debug** hoặc **error**.
-

Trả lời các câu hỏi

Câu hỏi 1

Tên của tệp layout cho activity chính là gì?

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

Câu hỏi 2

Tên của tài nguyên chuỗi xác định tên của ứng dụng là gì?

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

Câu hỏi 3

Công cụ nào được sử dụng để tạo trình giả lập mới?

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

Câu hỏi 4

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

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

Bạn sẽ thấy câu lệnh "MainActivity layout is complete" trong **Logcat pane** nếu menu **Log level** được đặt ở mức nào? (Có thể chọn nhiều câu trả lời)

- Verbose
- Debug
- Info
- Warn
- Error
- Assert

Kiểm tra để đảm bảo rằng ứng dụng có các yếu tố sau:

- Một **Activity** hiển thị "Hello World" trên màn hình.
- Các câu lệnh **Log** trong **onCreate()** của **MainActivity**.
- Mức **Log** trong **Logcat** chỉ hiển thị các thông báo **debug** hoặc **error**.

1.2 Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị Android bao gồm một hệ thống các đối tượng gọi là view — mọi thành phần trên màn hình đều là một View. Lớp View đại diện cho khối xây dựng cơ bản của tất cả các thành phần giao diện người dùng và là lớp cơ sở cho các lớp cung cấp các thành phần giao diện tương tác như nút bấm, hộp kiểm, và các trường nhập văn bản. Các lớp con View thường được sử dụng, được mô tả qua nhiều bài học, bao gồm:

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

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

Ví dụ: Mã **MainActivity** trong ứng dụng "Hello World" hiển thị một bộ cục được định nghĩa trong tệp bộ cục **activity_main.xml**, tệp này bao gồm một **TextView** với văn bản "Hello World".

Trong các ứng dụng phức tạp hơn, một **Activity** có thể thực hiện các hành động để phản hồi các thao tác chạm của người dùng, vẽ nội dung đồ họa, hoặc yêu cầu dữ

liệu từ cơ sở dữ liệu hoặc internet. Bạn sẽ tìm hiểu thêm về lớp **Activity** trong một bài học khác.

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

Những gì bạn cần biết trước

Bạn cần 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 với hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục để thiết kế một bố cục.
- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới. Hãy xem phần **Từ vựng và khái niệm** để tìm các định nghĩa dễ hiểu.

Những gì bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử **Button** và một **TextView** vào bố cục.
- Thao tác từng phần tử trong **ConstraintLayout** để ràng buộc chúng với lề và các phần tử khác.
- Thay đổi thuộc tính của các phần tử giao diện người dùng (UI).
- Chỉnh sửa bố cục của ứng dụng trong XML.
- Trích xuất các chuỗi mã cứng thành tài nguyên chuỗi (string resources).
- Triển khai các phương thức xử lý sự kiện khi nhấn (click-handler) để hiển thị thông báo trên màn hình khi người dùng chạm vào mỗi nút bấm (Button).

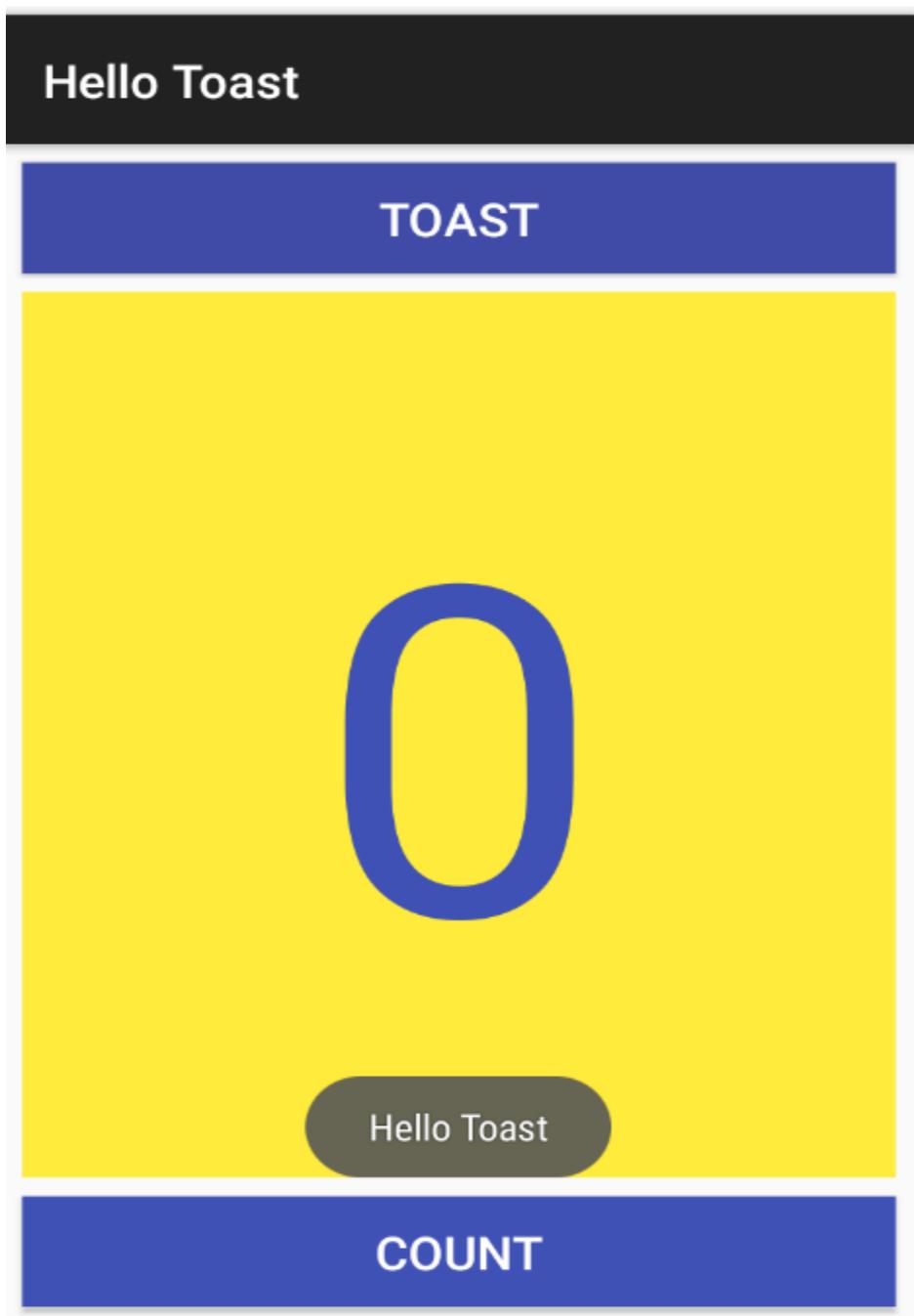
Tổng quan về ứng dụng

Ứng dụng Hellotoast bao gồm hai phần tử nút và một TextView. Khi người dùng chạm vào đầu tiên

Nút, nó hiển thị một tin nhắn ngắn (bánh mì nướng) trên màn hình. Nhấn vào nút thứ hai tăng một

"Nhấp vào" Bộ đếm được hiển thị trong TextView, bắt đầu từ 0.

Đây là những gì ứng dụng đã hoàn thành trông như thế nào:

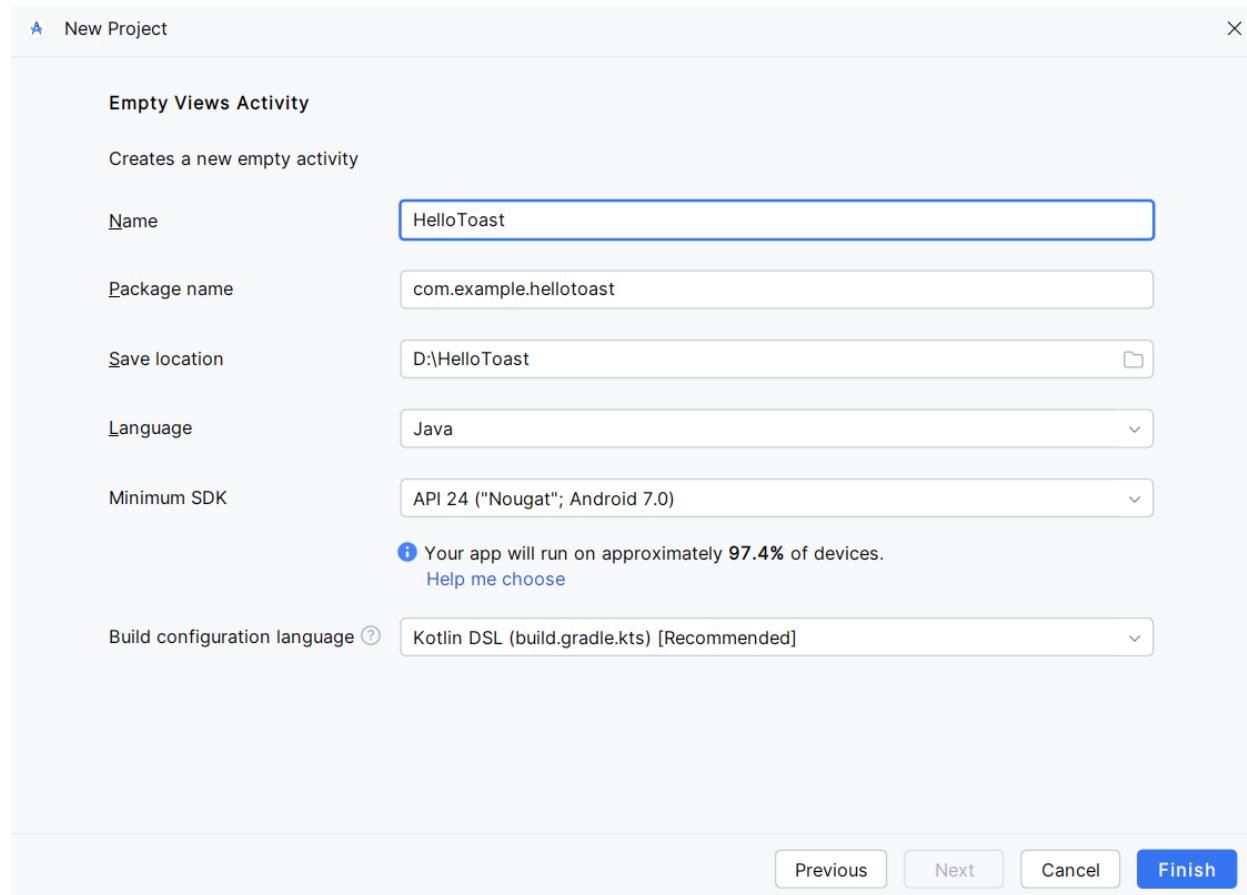


Nhiệm vụ 1: Tạo và khám phá một dự án mới

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

1.1 Tạo dự án Android Studio

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



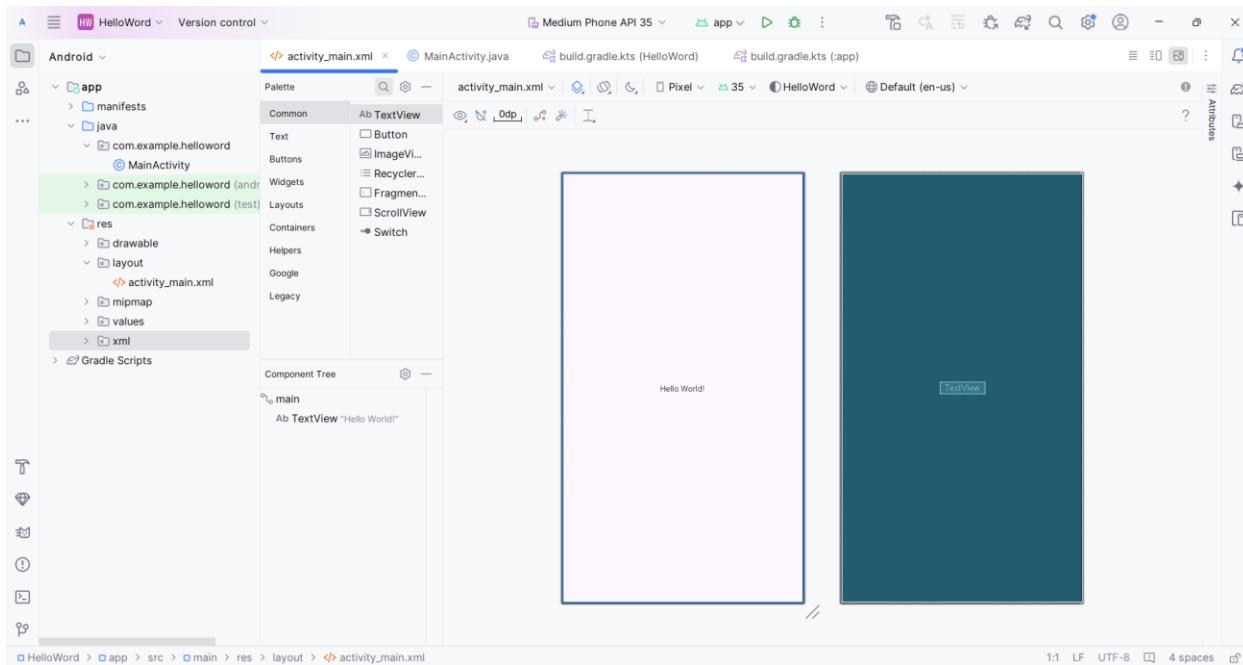
15. Chọn Run > Run app hoặc nhấp vào biểu tượng Run trên thanh công cụ để xây dựng và chạy ứng dụng trên trình giả lập hoặc thiết bị của bạn.

1.2 Khám phá trình chỉnh sửa bố cục (layout editor)

Android Studio cung cấp trình chỉnh sửa bố cục để nhanh chóng xây dựng bố cục giao diện người dùng (UI) cho ứng dụng. Nó cho phép bạn kéo các phần tử vào chế độ xem thiết kế trực quan và bắn thiết kế, định vị chúng trong bố cục, thêm các ràng buộc (constraints) và đặt thuộc tính. Ràng buộc xác định vị trí của một phần

tử UI trong bố cục. Một ràng buộc đại diện cho một kết nối hoặc sự căn chỉnh với một thành phần khác, bố cục cha, hoặc một hướng dẫn vô hình.

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



1. Trong thư mục app > res > layout trong ngăn Project > Android, nhấp đúp vào tệp activity_main.xml để mở, nếu tệp chưa được mở.
2. Nhấp vào tab Design nếu tab này chưa được chọn. Bạn sử dụng tab Design để thao tác các phần tử và bố cục, và tab Text để chỉnh sửa mã XML của bố cục.
3. Ngăn Palettes hiển thị các phần tử giao diện người dùng (UI) mà bạn có thể sử dụng trong bố cục ứng dụng của mình.
4. Ngăn Component tree hiển thị cấu trúc cây phân cấp của các phần tử UI. Các phần tử giao diện được tổ chức thành cấu trúc cây bao gồm cha và con, trong đó một phần tử con kế thừa các thuộc tính từ phần tử cha. Trong hình minh họa, TextView là phần tử con của ConstraintLayout. Bạn sẽ tìm hiểu thêm về các phần tử này trong bài học sau.

5. Các ngăn thiết kế và bản thiết kế trong trình chỉnh sửa bố cục hiển thị các phần tử UI trong bố cục. Trong hình minh họa, bố cục chỉ hiển thị một phần tử: một TextView hiển thị dòng chữ "Hello World".
6. Tab Attributes hiển thị ngăn Attributes dùng để thiết lập các thuộc tính cho một phần tử UI.

Mẹo: Tham khảo [Xây dựng giao diện người dùng với Layout Editor](#) để biết chi tiết về cách sử dụng trình chỉnh sửa bố cục, và [Giới thiệu về Android Studio](#) để xem toàn bộ tài liệu hướng dẫn về Android Studio.

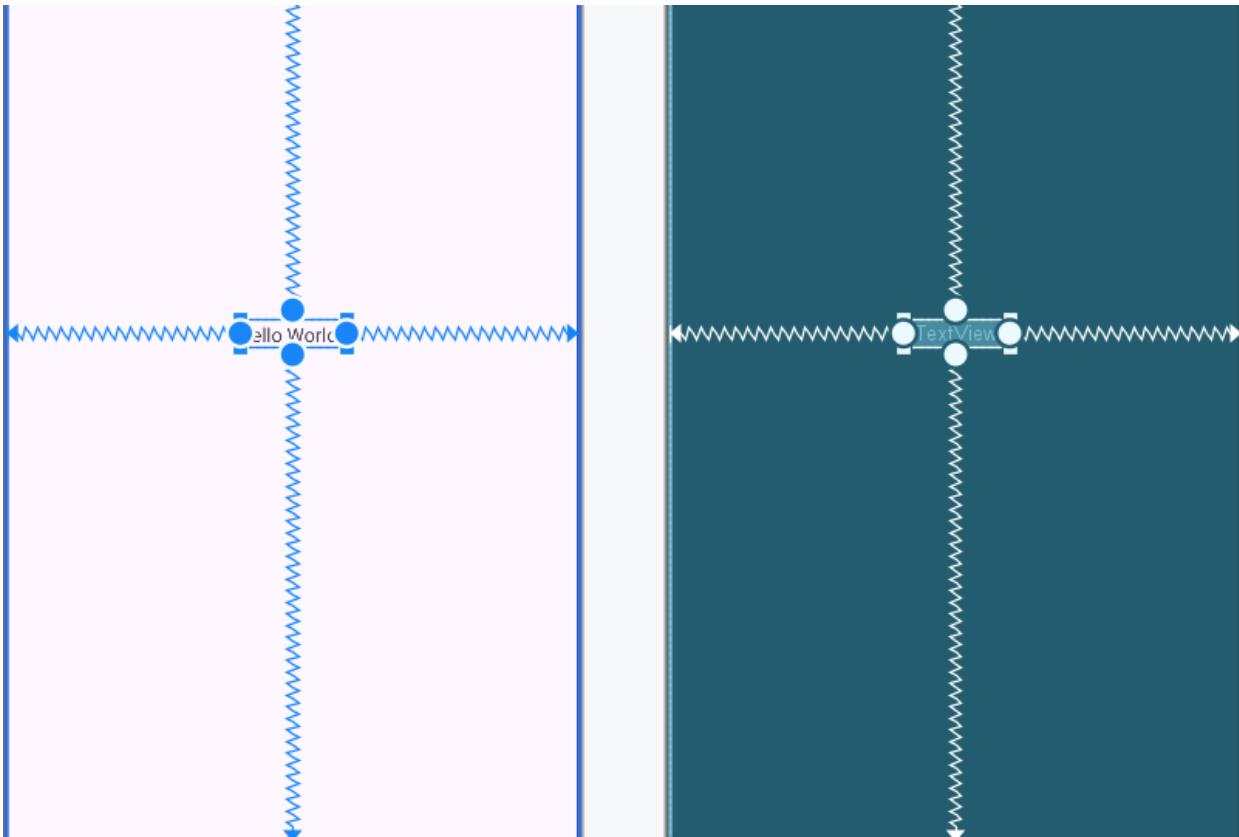
Nhiệm vụ 2: Thêm các phần tử View trong trình chỉnh sửa bố cục

Trong nhiệm vụ này, bạn tạo giao diện người dùng (UI) cho ứng dụng HelloToast trong trình chỉnh sửa bố cục bằng cách sử dụng các tính năng của ConstraintLayout. Bạn có thể tạo các ràng buộc (constraints) theo cách thủ công, như được mô tả bên dưới, hoặc tự động bằng cách sử dụng công cụ Autoconnect.

2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện theo các bước sau:

1. Mở tệp activity_main.xml từ ngăn Project > Android nếu nó chưa được mở. Nếu tab Design chưa được chọn, nhấp vào tab này.
Nếu không có bản thiết kế (blueprint), nhấp vào nút Select Design Surface trên thanh công cụ và chọn tùy chọn Design + Blueprint.
2. Công cụ Autoconnect cũng nằm trên thanh công cụ và được bật theo mặc định. Đảm bảo rằng công cụ này không bị tắt ở bước này.
3. Nhấp vào nút phóng to (zoom in) để phóng to các ngăn thiết kế và bản thiết kế, giúp bạn xem kỹ hơn.
4. Chọn TextView trong ngăn Component Tree. TextView với dòng chữ "Hello World" sẽ được đánh dấu trong các ngăn thiết kế và bản thiết kế, đồng thời các ràng buộc của phần tử sẽ hiển thị.
5. Tham khảo hình minh họa động dưới đây cho bước này. Nhấp vào chốt tròn (circular handle) ở bên phải của TextView để xóa ràng buộc ngang (horizontal constraint) liên kết phần tử này với cạnh phải của bố cục.
Khi xóa ràng buộc này, TextView sẽ nhảy sang phía bên trái vì nó không còn bị ràng buộc với cạnh phải.
Để thêm lại ràng buộc ngang, nhấp vào cùng chốt tròn và kéo một đường đến cạnh phải của bố cục.

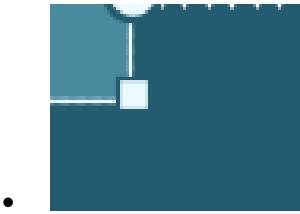


Trong các ngăn thiết kế hoặc bản thiết kế, các chốt sau xuất hiện trên phần tử TextView:

- **Chốt ràng buộc (Constraint handle):** Để tạo một ràng buộc như minh họa trong hình động ở trên, nhấp vào chốt ràng buộc (hiển thị dưới dạng hình tròn ở bên cạnh của một phần tử). Sau đó, kéo chốt này đến một chốt ràng buộc khác hoặc đến ranh giới của bố cục cha. Một đường zigzag biểu thị ràng buộc được tạo.



- **Chốt thay đổi kích thước (Resizing handle):** Để thay đổi kích thước của phần tử, kéo các chốt thay đổi kích thước hình vuông. Trong khi bạn kéo, chốt này sẽ chuyển thành góc nghiêng.



2.2 Thêm một nút (Button) vào bố cục

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

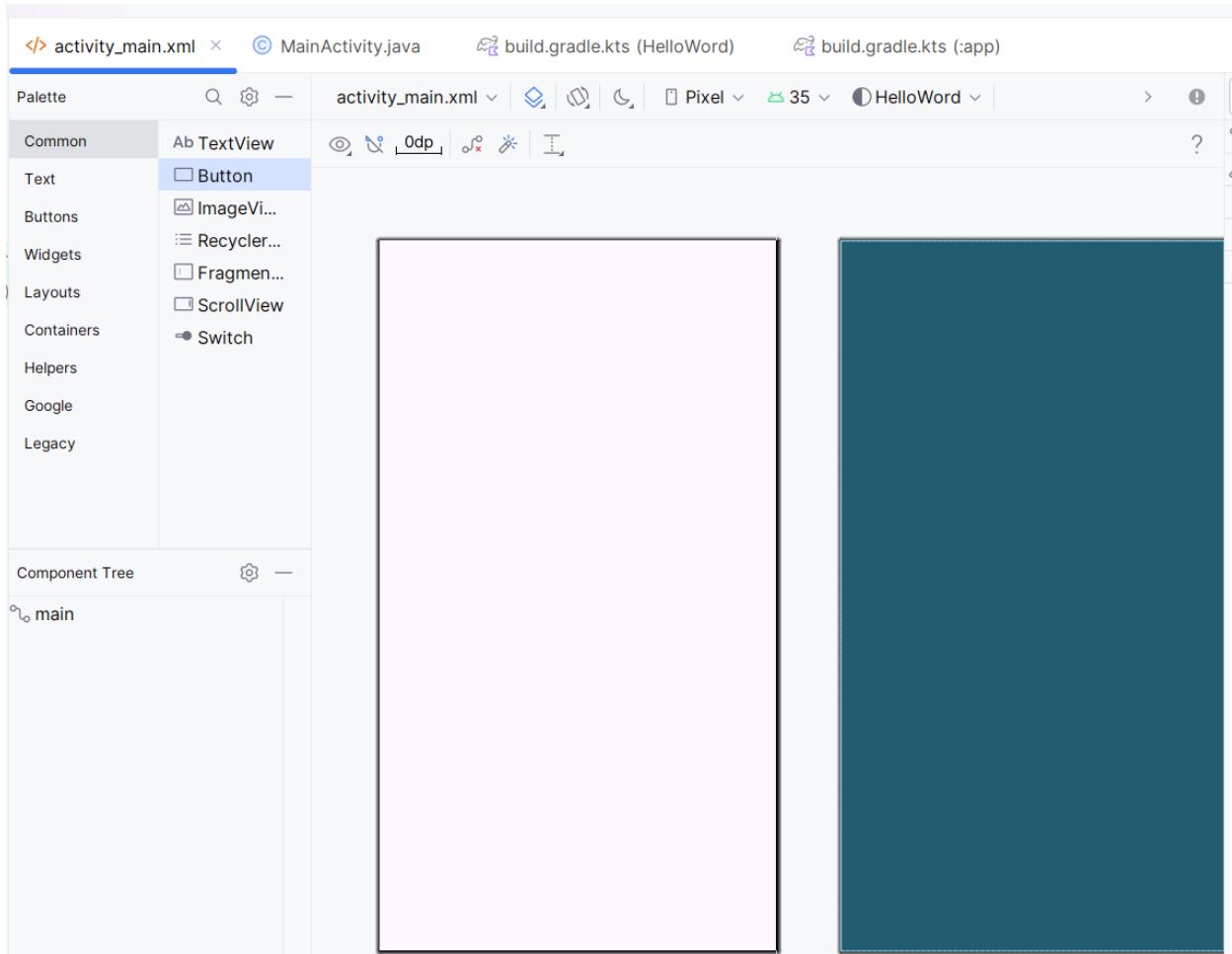
Thực hiện các bước sau để thêm một nút:

1. **Bắt đầu với một bố cục trống:**

Phần tử TextView không cần thiết, vì vậy khi nó vẫn đang được chọn, nhấn phím Delete hoặc chọn Edit > Delete. Lúc này, bạn sẽ có một bố cục hoàn toàn trống.

2. **Kéo một nút từ ngăn Palette vào bố cục:**

Kéo nút Button từ ngăn Palette vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả nút ở khu vực giữa trên cùng của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc đến cạnh trên, cạnh trái và cạnh phải của bố cục như minh họa trong hình động bên dưới.



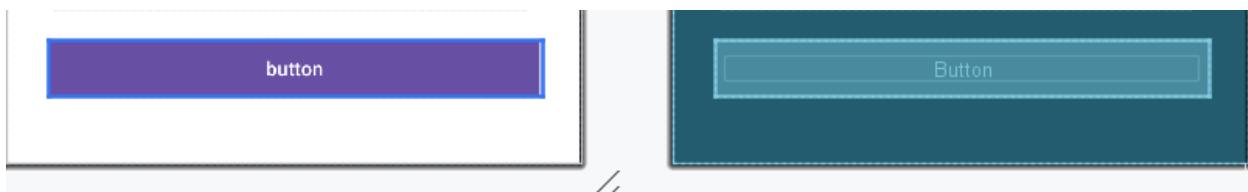
2.3 Thêm một nút thứ hai vào bố cục

1. Kéo một nút khác từ ngăn Palette:

Kéo một nút Button khác từ ngăn Palette vào giữa bố cục như minh họa trong hình động bên dưới. Công cụ Autoconnect có thể tự động tạo các ràng buộc ngang (horizontal constraints) cho bạn. Nếu không, bạn có thể tự kéo các ràng buộc này.

2. Thêm ràng buộc dọc:

Kéo một ràng buộc dọc (vertical constraint) từ nút này đến cạnh dưới của bố cục (tham khảo hình minh họa bên dưới).



Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách:

- Chọn phần tử và di chuyển con trỏ chuột qua nó để hiển thị nút **Clear Constraints**. Nhấp vào nút này để xóa tất cả ràng buộc của phần tử được chọn.
- Để xóa một ràng buộc cụ thể, hãy nhấp vào chốt cụ thể đã đặt ràng buộc đó.

Xóa tất cả các ràng buộc trong toàn bộ bố cục:

- Nhấp vào công cụ **Clear All Constraints** trong thanh công cụ. Công cụ này rất hữu ích nếu bạn muốn làm lại tất cả các ràng buộc trong bố cục của mình.

Nhiệm vụ 3: Thay đổi thuộc tính của phần tử giao diện người dùng (UI)

Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Bạn có thể tìm thấy các thuộc tính (được gọi là "properties") chung cho tất cả các chế độ xem (views) trong tài liệu của lớp View.

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

3.1 Thay đổi kích thước nút

Trình chỉnh sửa bố cục cung cấp các bộ điều khiển thay đổi kích thước ở cả bốn góc của View để bạn có thể nhanh chóng thay đổi kích thước View.

Bạn có thể kéo các bộ điều khiển ở mỗi góc của View để thay đổi kích thước, nhưng làm như vậy sẽ cố định kích thước chiều rộng và chiều cao (hardcoded).

Tránh mã hóa kích thước cố định cho hầu hết các phần tử View vì nó không thể thích nghi với nội dung và kích thước màn hình khác nhau.

Thay vào đó, sử dụng ngăn Attributes ở phía bên phải trình chỉnh sửa bố cục để chọn chế độ kích thước không dùng kích thước cố định. Ngăn Attributes bao gồm một bảng điều chỉnh kích thước hình vuông được gọi là View Inspector ở trên cùng.

Attributes

button	button_count
background	button_count
backgroundTint	#1565C0
id	button_count
text	COUNT

Layout

Constraint Widget

The diagram illustrates a constraint layout with three views: two horizontal buttons at the top and a vertical button below them. Horizontal constraints connect the left button to the center and the center to the right button, both with a value of 0. A vertical constraint connects the top of the left button to the bottom of the center button with a value of 48dp. A bottom constraint connects the bottom of the center button to the bottom of the right button with a value of 53.

Constraints

- Start → StartOf parent (0dp)
- End → EndOf parent (0dp)
- Bottom → BottomOf parent (48dp)
- Horizontal Bias (0.534)

layout_width	353dp
layout_height	40dp
visibility	
visibility	

Ý nghĩa của các ký hiệu trong hình vuông:

1. Điều khiển chiều cao (Height control):

- Điều khiển này xác định thuộc tính `layout_height` và xuất hiện ở hai đoạn trên và dưới của hình vuông.
- Các góc xiên cho biết điều khiển này được đặt thành `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều dọc khi cần để vừa với nội dung.
- Số "8" biểu thị khoảng cách margin tiêu chuẩn được đặt là 8dp.

2. Điều khiển chiều rộng (Width control):

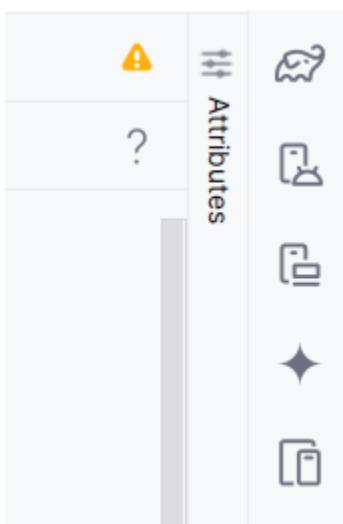
- Điều khiển này xác định thuộc tính `layout_width` và xuất hiện ở hai đoạn bên trái và bên phải của hình vuông.
- Các góc xiên cho biết điều khiển này được đặt thành `wrap_content`, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần để vừa với nội dung, đến giới hạn margin 8dp.

3. Nút đóng ngăn Attributes (Attributes pane close button):

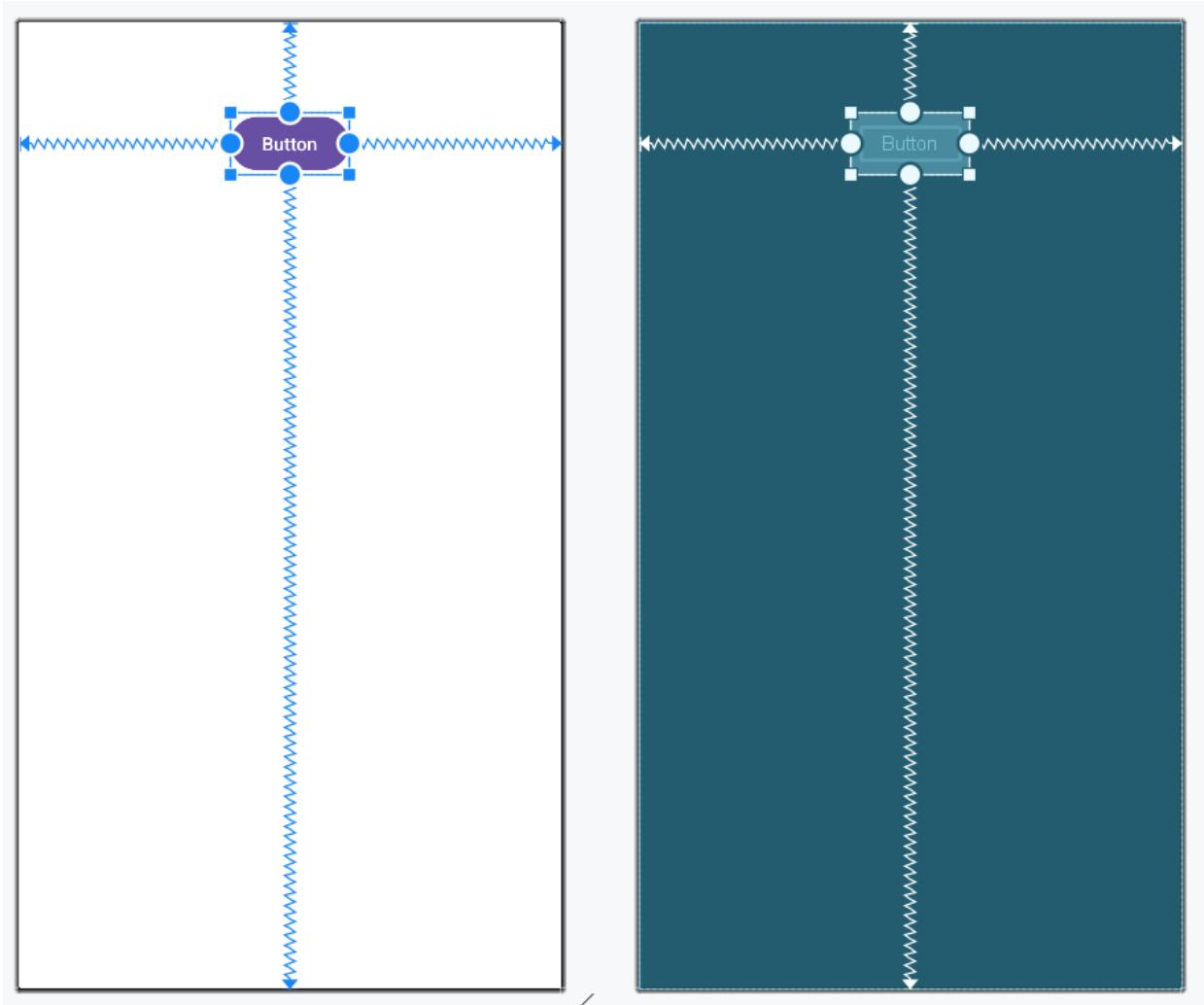
- Nhấp vào nút này để đóng ngăn Attributes.

Làm theo các bước sau:

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

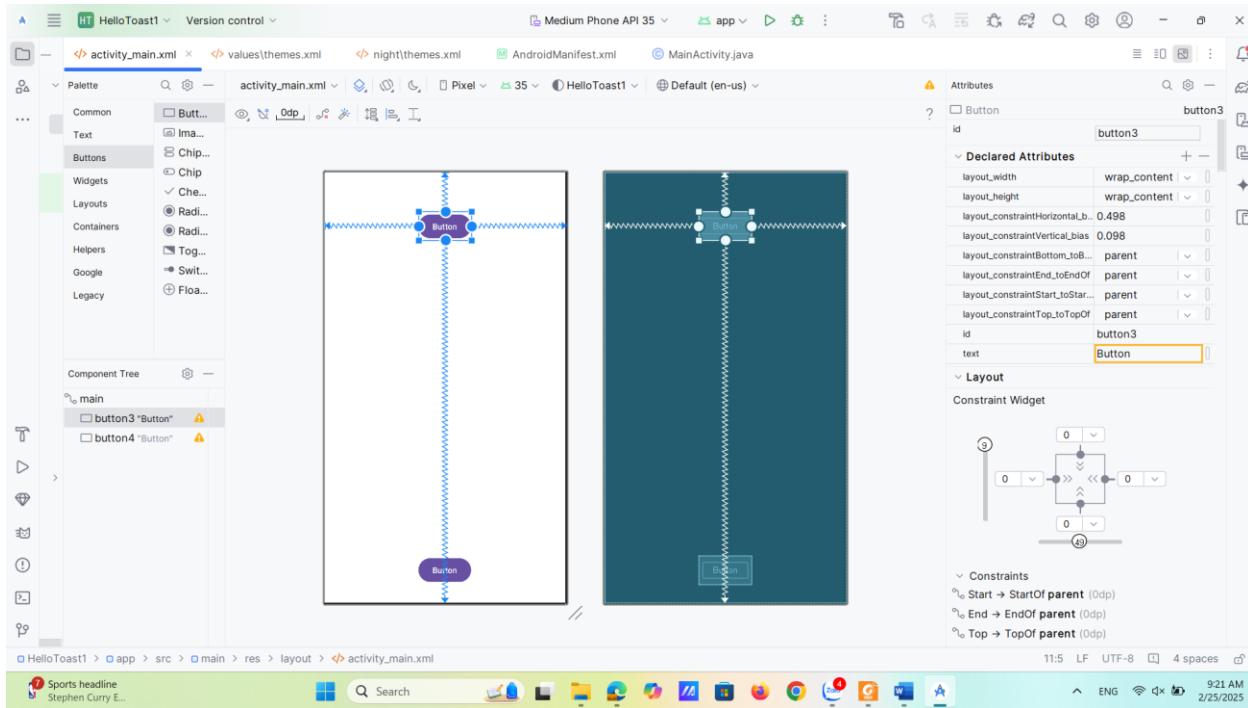


- Nhấp vào bộ điều khiển chiều rộng hai lần — lần nhấp đầu tiên thay đổi thành **Fixed** với các đường thẳng, và lần nhấp thứ hai thay đổi thành **Match Constraints** với các cuộn lò xo, như được minh họa trong hình động bên dưới.



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

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



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

- **match_constraint**: Thiết lập này mở rộng phần tử **View** để lấp đầy không gian của phần tử cha theo chiều rộng hoặc chiều cao—đến mức biên, nếu được đặt. Trong trường hợp này, phần tử cha là **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.
- **wrap_content**: Thiết lập này thu nhỏ kích thước của phần tử **View** sao cho chỉ vừa đủ để bao bọc nội dung của nó. Nếu không có nội dung, phần tử **View** sẽ trở nên không hiển thị.
- Để chỉ định một kích thước cố định phù hợp với kích thước màn hình của thiết bị, hãy sử dụng một số cố định của đơn vị điểm ảnh độc lập với mật độ (**dp units**). Ví dụ, **16dp** nghĩa là 16 điểm ảnh độc lập với mật độ.

Mẹo: Nếu bạn thay đổi thuộc tính **layout_width** bằng cách sử dụng menu bật lên của nó, thuộc tính **layout_width** sẽ được đặt thành **zero (0)** vì không có kích thước nào được đặt cụ thể. Cài đặt này giống với **match_constraint**—phần tử **View** có thể mở rộng hết mức có thể để đáp ứng các ràng buộc và cài đặt biên.

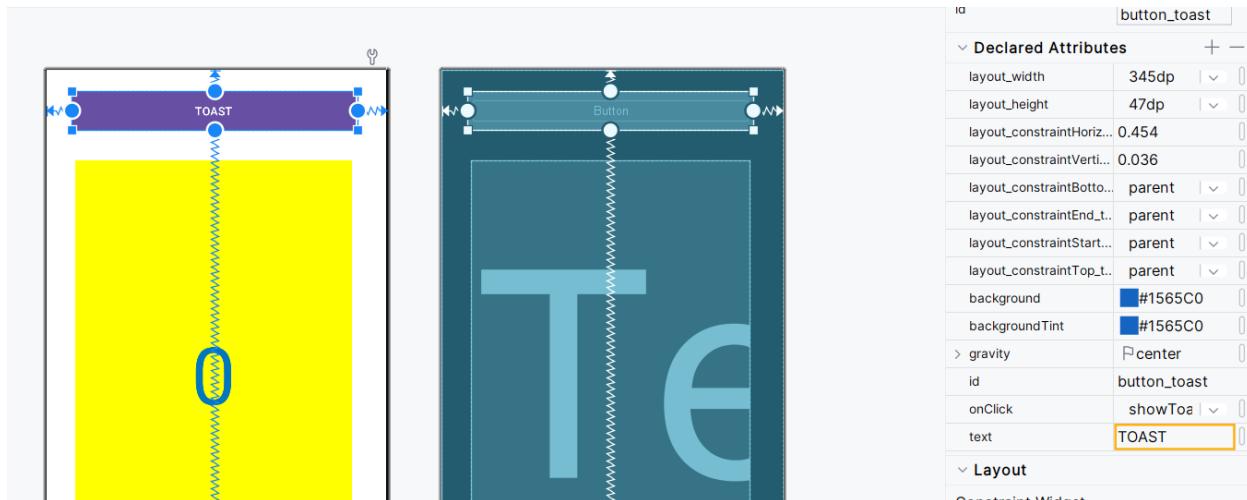
3.2 Thay đổi các thuộc tính của Button

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

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

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

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



5. Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng **button_count** làm ID, đặt **Count** cho thuộc tính text, và sử dụng cùng các màu cho background và text như các bước trước.

Lưu ý:

Màu **colorPrimary** là màu chủ đạo của theme, một trong những màu cơ sở được định nghĩa sẵn trong tệp tài nguyên **colors.xml**. Nó được sử dụng cho thanh ứng

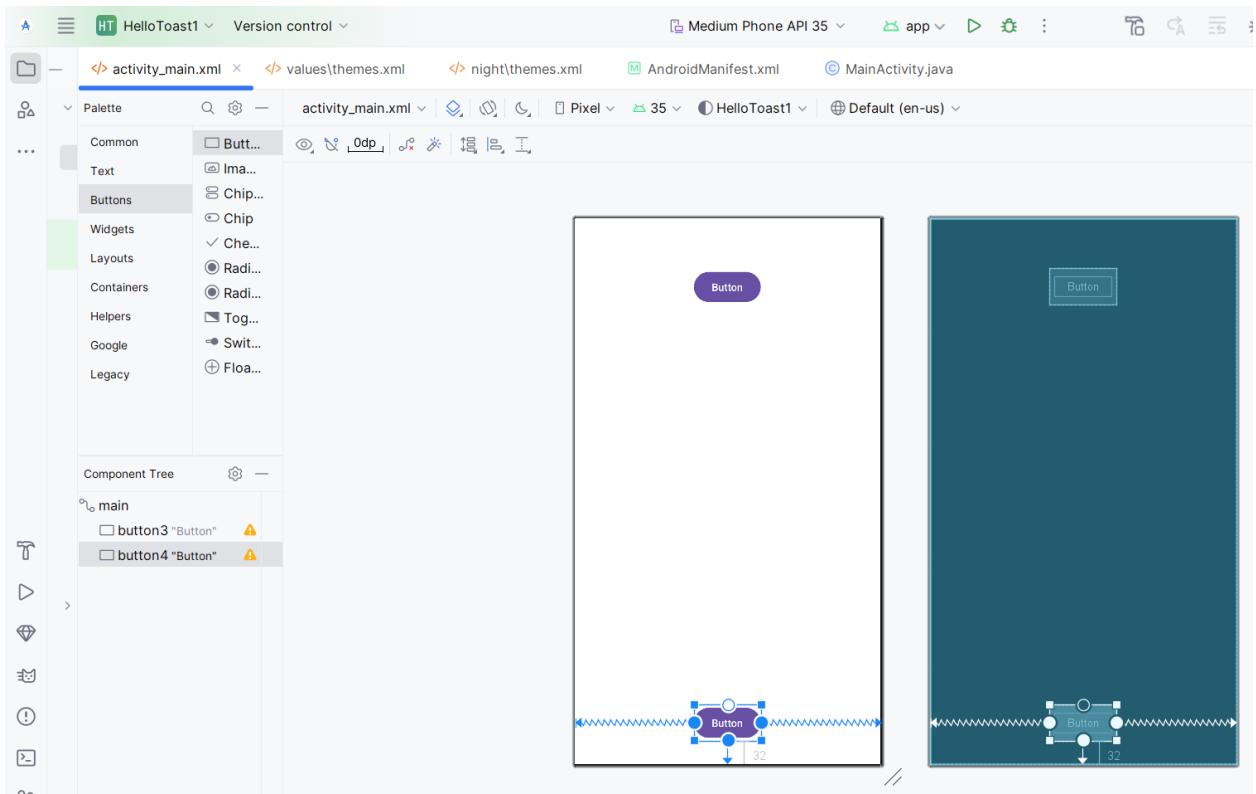
dụng (app bar). Việc sử dụng các màu cơ sở cho các phần tử UI khác sẽ tạo ra giao diện người dùng (UI) thống nhất. Bạn sẽ tìm hiểu thêm về theme ứng dụng và Material Design trong một bài học khác.

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

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

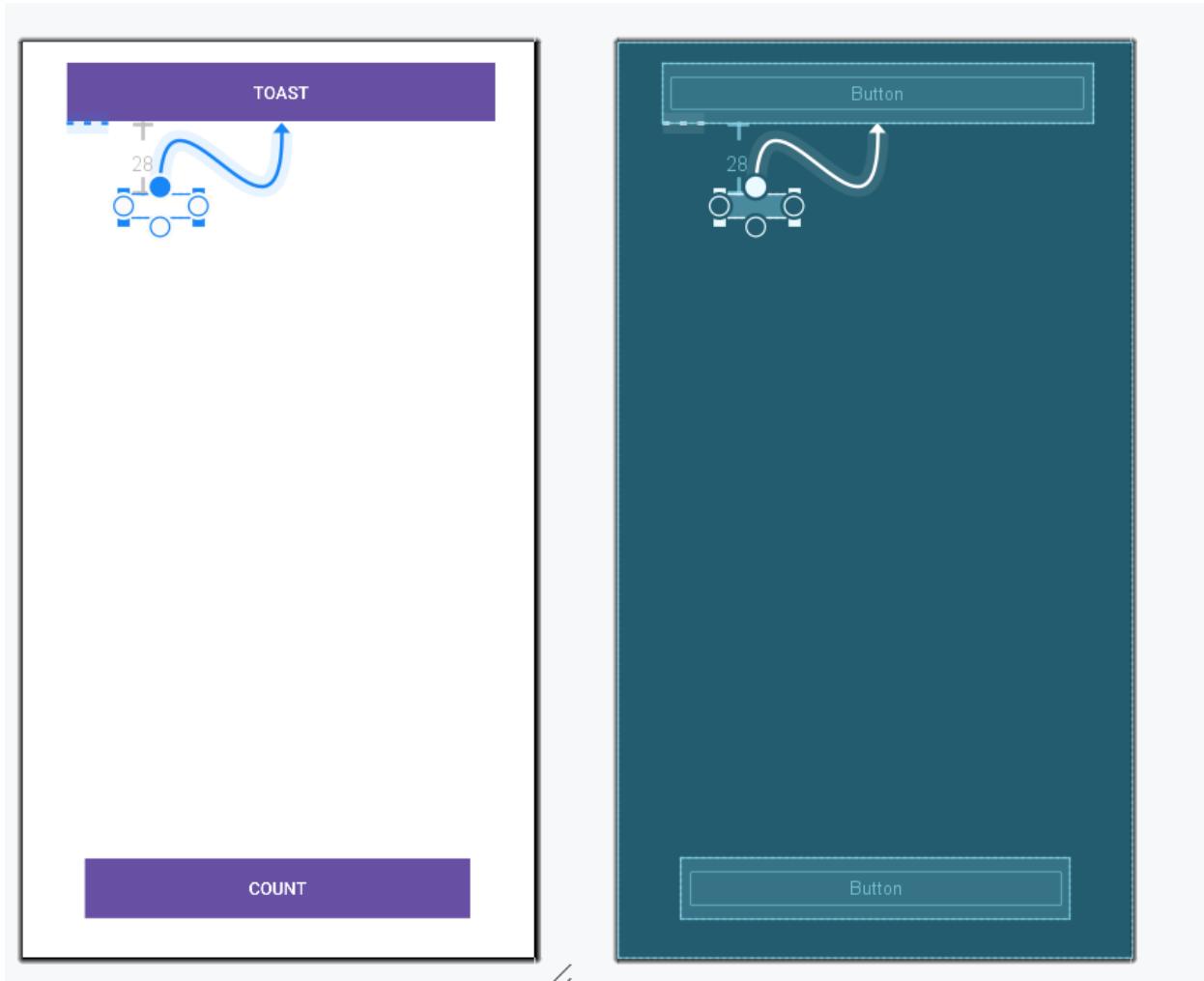
4.1 Thêm một TextView và thiết lập ràng buộc

1. Như minh họa trong hình động dưới đây, kéo một **TextView** từ ngăn **Palette** vào phần trên của bố cục. Kéo một ràng buộc từ phía trên của **TextView** đến chốt ở phía dưới của nút **Toast**. Việc này sẽ ràng buộc **TextView** nằm ngay bên dưới nút **Toast**.



2. Như minh họa trong hình động dưới đây, kéo một ràng buộc từ phía dưới của **TextView** đến chốt ở phía trên của nút **Count**. Sau đó, kéo các ràng

buộc từ hai bên của **TextView** đến hai cạnh của bố cục. Việc này ràng buộc **TextView** nằm giữa bố cục, ở giữa hai nút **Button**.



4.2 Đặt các thuộc tính cho TextView

Với **TextView** được chọn, hãy mở bảng thuộc tính (**Attributes pane**) nếu nó chưa mở. Đặt các thuộc tính cho **TextView** như hướng dẫn dưới đây. Những thuộc tính mới sẽ được giải thích sau:

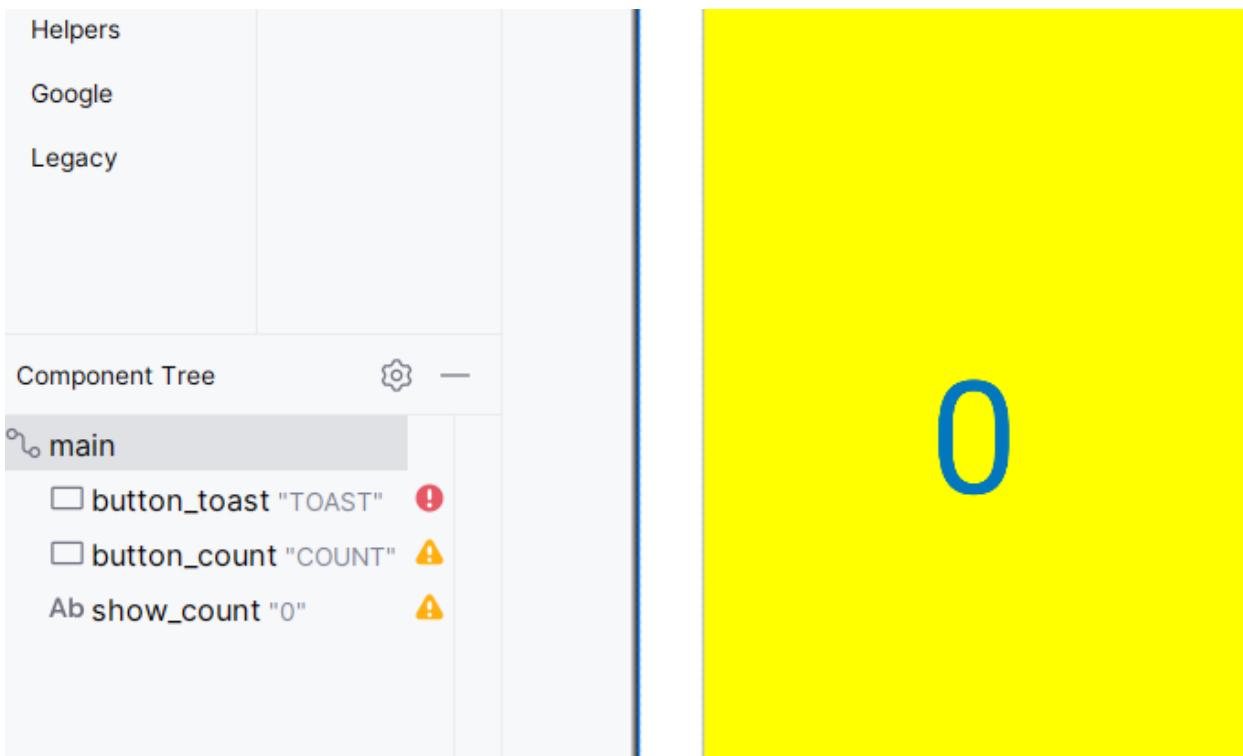
1. Đặt **ID** thành `show_count`.
2. Đặt **text** thành `0`.
3. Đặt **textSize** thành `160sp`.
4. Đặt **textStyle** thành **B** (in đậm) và **textAlignment** thành **ALIGNCENTER** (căn giữa đoạn văn bản).

5. Thay đổi chế độ kích thước ngang và dọc (**layout_width** và **layout_height**) thành **match_constraint**.
 6. Đặt **textColor** thành @color/colorPrimary.
 7. Cuộn xuống bảng thuộc tính và nhập vào "View all attributes", cuộn đến trang thứ hai của các thuộc tính, tìm thuộc tính "background" và nhập giá trị #FFF00 để chọn một sắc thái màu vàng.
 8. Cuộn xuống thuộc tính "gravity", mở rộng mục "gravity" và chọn "center_ver" (để căn giữa theo chiều dọc).
- **textSize**: Kích thước văn bản của TextView. Trong bài học này, kích thước được đặt là **160sp**. **sp** là viết tắt của **scale-independent pixel** (pixel không phụ thuộc tỷ lệ) và giống như **dp**, là đơn vị điều chỉnh theo mật độ màn hình và tùy chọn kích thước phông chữ của người dùng. Hãy sử dụng đơn vị **dp** khi bạn xác định kích thước phông chữ để đảm bảo kích thước được điều chỉnh cho cả mật độ màn hình và sở thích của người dùng.
 - **textStyle** và **textAlignment**: Kiểu văn bản, được đặt là **B (bold)** (in đậm) trong bài học này. Căn chỉnh văn bản, được đặt là **ALIGNCENTER** (căn giữa đoạn văn bản).
 - **gravity**: Thuộc tính *gravity* xác định cách một View được căn chỉnh trong *View* hoặc *ViewGroup* cha của nó. Trong bước này, bạn căn chỉnh *TextView* theo chiều dọc ở giữa *ConstraintLayout* cha.

Bạn có thể nhận thấy rằng thuộc tính *background* nằm ở trang đầu tiên của bảng *Attributes* (Thuộc tính) đối với một *Button*, nhưng lại nằm ở trang thứ hai đối với một *TextView*. Bảng *Attributes* thay đổi tùy thuộc vào từng loại *View*: các thuộc tính phổ biến nhất của loại *View* sẽ xuất hiện trên trang đầu tiên, và các thuộc tính khác được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của bảng *Attributes*, nhấn vào biểu tượng trên thanh công cụ ở phía trên cùng của bảng.

Task 5: Chính sửa bộ cục trong XML

Bộ cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, bên cạnh mỗi phần tử giao diện người dùng trong bảng Component Tree lại xuất hiện một dấu chấm than. Di chuyển con trỏ chuột qua các dấu chấm than này để xem thông báo cảnh báo, như hình minh họa bên dưới. Cùng một cảnh báo xuất hiện cho cả ba phần tử: chuỗi được mã hóa cứng nên được thay thế bằng tài nguyên.



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

5.1 Mở mã XML của bố cục

Trong nhiệm vụ này, hãy mở tệp activity_main.xml nếu nó chưa được mở, và nhấp vào tab Text ở dưới cùng của trình chỉnh sửa bố cục.

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

```
<?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"
    android:orientation="vertical"
```

```
tools:context=".MainActivity" >

<Button
    android:id="@+id/btnToast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:background="#3F4BA7"
    android:text="@string/button_label_toast"
    android:textColor="@color/white"
    android:textSize="20dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textToast"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="#FFEB3B"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAllCaps="true"
    android:textColor="#3F51B5"
    android:textSize="200dp"
    app:layout_constraintBottom_toTopOf="@+id/btnCount"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/btnToast"
    app:layout_constraintVertical_bias="0.0" />

<Button
```

```
    android:id="@+id/btnCount"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="#3F51B5"
    android:text="@string/button_label_count"
    android:textColor="@color/white"
    android:textSize="20dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

5.2 Tách các chuỗi thành tài nguyên

Thay vì mã hóa cứng các chuỗi, việc sử dụng tài nguyên chuỗi là một thực hành tốt, vì các chuỗi được lưu trữ trong một tệp riêng giúp quản lý dễ dàng hơn, đặc biệt khi bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và địa phương hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho mỗi ngôn ngữ.

Các bước thực hiện:

1. Nhập một lần vào từ "Toast" (chuỗi cảnh báo đầu tiên được đánh dấu).
2. Nhấn Alt-Enter trên Windows hoặc Option-Enter trên macOS và chọn Extract string resource từ menu bật lên.
3. Nhập "button_label_toast" cho tên tài nguyên.
4. Nhấp OK. Một tài nguyên chuỗi sẽ được tạo trong tệp values/res/values.xml, và chuỗi trong mã của bạn sẽ được thay thế bằng tham chiếu tới tài nguyên: @string/button_label_toast
5. Tách các chuỗi còn lại: sử dụng "button_label_count" cho "Count" và "count_initial_value" cho "0".

- Trong ngăn Project > Android, mở rộng mục values trong thư mục res, sau đó nhấp đúp vào tệp strings.xml để xem các tài nguyên chuỗi của bạn trong tệp strings.xml.

```
<resources>
    <string name="app_name">HelloToast1</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
</resources>
```

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

```
<resources>
    <string name="app_name">HelloToast1</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
    A
</resources>
```

Mẹo: Các tài nguyên chuỗi bao gồm tên ứng dụng, xuất hiện trên thanh ứng dụng ở đầu màn hình nếu bạn khởi tạo dự án bằng Mẫu Trống. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa tài nguyên app_name.

Nhiệm vụ 6: Thêm xử lý sự kiện onClick cho các nút

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

6.1 Thêm thuộc tính onClick và phương thức xử lý cho mỗi nút

Một click handler là phương thức được gọi khi người dùng nhấn hoặc chạm vào một phần tử UI có thể nhấn được. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick ở ngăn Attributes của tab Design. Bạn cũng có thể chỉ định tên của phương thức xử lý trong trình chỉnh sửa XML bằng cách thêm thuộc tính android:onClick vào nút. Ở đây, bạn sẽ sử dụng phương pháp thứ hai vì bạn chưa tạo các phương thức xử lý, và trình chỉnh sửa XML cung cấp cách tự động tạo các phương thức đó.

- Với trình chỉnh sửa XML (tab Text) đang mở, tìm nút có android:id được đặt là button_toast.

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="345dp"
    android:layout_height="47dp"
    android:background="#1565C0"
    android:backgroundTint="#1565C0"
```

```
    android:gravity="center"
    android:text="TOAST"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.454"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.036"
/>

```

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

```
    android:onClick="showToast"/>
```

3. Nhập vào biểu tượng bóng đèn đỏ xuất hiện bên cạnh thuộc tính. Chọn Create click handler, chọn MainActivity, và nhấp OK. Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy nhập vào tên phương thức ("showToast"). Nhấn Alt-Enter (trên Windows/Linux) hoặc Option-Enter (trên Mac), chọn Create 'showToast(view)' in MainActivity, và nhấp OK. Thao tác này sẽ tạo ra một phương thức mẫu (placeholder method stub) cho phương thức showToast() trong MainActivity, như được hiển thị ở cuối các bước này.
4. Lặp lại hai bước cuối với nút button_count: Thêm thuộc tính android:onClick vào cuối phần tử, và tạo click handler.

```
    android:onClick="countUp"/>
```

Mã XML của các phần tử giao diện người dùng bên trong ConstraintLayout bây giờ trông như sau:

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

    <Button
        android:id="@+id/btnToast"
        android:layout_width="0dp"
```

```
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:background="#3F4BA7"
    android:text="@string/button_label_toast"
    android:textColor="@color/white"
    android:textSize="20dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<TextView
    android:id="@+id/textToast"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="#FFEB3B"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAllCaps="true"
    android:textColor="#3F51B5"
    android:textSize="200dp"
    app:layout_constraintBottom_toTopOf="@+id(btnCount"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id	btnToast"
    app:layout_constraintVertical_bias="0.0" />
```

```
<Button
    android:id="@+id	btnCount"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
```

```
        android:layout_marginBottom="8dp"
        android:background="#3F51B5"
        android:text="@string/button_label_count"
        android:textColor="@color/white"
        android:textSize="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

5. Nếu MainActivity.java chưa được mở, mở rộng mục java trong ngăn Project > Android, mở rộng com.example.android.hellotoast, sau đó nhấp đúp vào MainActivity. Trình soạn thảo mã sẽ xuất hiện với mã của MainActivity.

```
package com.example.hellotoast1;

> import ...

› </> public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this );
        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            actionBar.setBackgroundDrawable(new ColorDrawable(Color.BLUE));
        }
        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;
        });
    }

    1 usage
    public void showToast(View view) {
    }

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

6.2 Chính sửa xử lý sự kiện của nút Toast

Bây giờ, bạn sẽ chỉnh sửa phương thức showToast() – xử lý sự kiện click của nút Toast trong MainActivity – để hiển thị một thông báo. Một Toast cung cấp cách hiển thị thông báo đơn giản trong một cửa sổ popup nhỏ. Toast chỉ chiếm không gian cần thiết để hiển thị thông báo, và Activity 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 – hãy thêm một thông báo Toast để hiển thị kết quả của thao tác nhấn nút hoặc thực hiện một hành động.

Thực hiện theo các bước sau để chỉnh sửa xử lý sự kiện nút Toast:

1. Tìm vị trí của phương thức showToast() mới được tạo trong MainActivity.

```
public void showToast(View view) {  
}
```

2. Để tạo một đối tượng Toast, hãy gọi phương thức makeText (factory method) trên lớp Toast.

```
public void showToast(View view) {  
    Toast toast = Toast.makeText(  
}
```

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

3. Cung cấp context của Activity của ứng dụng. Vì một Toast được hiển thị phía trên giao diện người dùng của Activity, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong context của Activity mà bạn cần, hãy sử dụng từ khóa "this" như một cách rút gọn.

```
Toast toast = Toast.makeText(this,
```

4. Cung cấp thông báo để hiển thị, chẳng hạn như một tài nguyên chuỗi (ví dụ: toast_message mà bạn đã tạo ở bước trước). Tài nguyên chuỗi toast_message được nhận diện bằng R.string.toast_message.

```
Toast toast= Toast.makeText(this,R.string.toast_message,
```

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

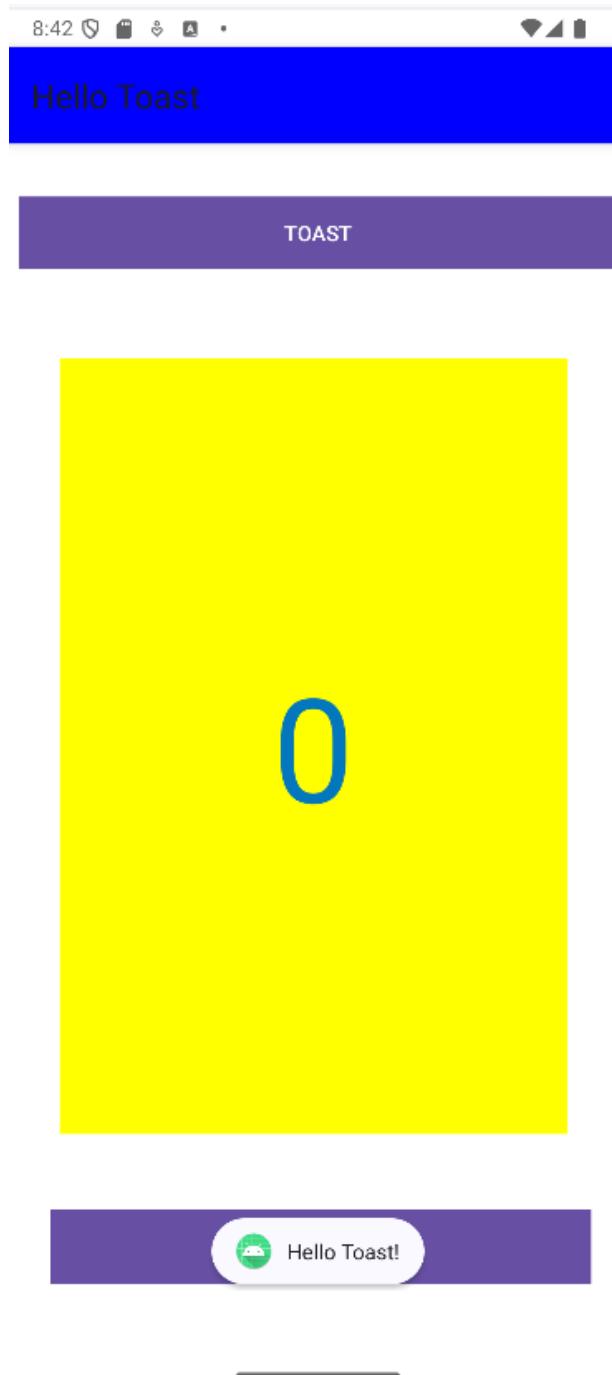
```
Toast toast = Toast.makeText(this,  
R.string.toast_message,Toast.LENGTH_SHORT);
```

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

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

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

Chạy ứng dụng và xác minh rằng thông báo Toast xuất hiện khi bạn nhấn nút Toast.



6.3 Chính sửa xử lý sự kiện của nút Count

Bây giờ, bạn sẽ chỉnh sửa phương thức `countUp()` – xử lý sự kiện click của nút Count trong `MainActivity` – sao cho nó hiển thị giá trị đếm hiện tại sau mỗi lần nhấn nút Count. Mỗi lần nhấn nút sẽ tăng giá trị đếm lên một đơn vị.

Mã xử lý sự kiện cần thực hiện các điều sau:

- Theo dõi giá trị đếm khi nó thay đổi.

- Gửi giá trị đếm được cập nhật đến TextView để hiển thị.

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

1. Tìm vị trí của phương thức countUp() mới được tạo trong MainActivity, có dạng:

```
public void countUp(View view) {  
}
```

2. Để theo dõi giá trị đếm, bạn cần một biến thành viên riêng (private member variable). Mỗi lần nhấn nút Count sẽ tăng giá trị của biến này. Nhập đoạn mã sau (đoạn mã này sẽ được đánh dấu màu đỏ và xuất hiện biểu tượng bóng đèn đỏ):

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

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

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn Create field 'mCount' từ menu bật lên. Thao tác này tạo ra một biến thành viên riêng tại đầu lớp MainActivity, và Android Studio mặc định kiểu của biến này là integer (int);
4. Thay đổi câu lệnh khai báo biến thành viên thành khởi tạo giá trị bằng 0:

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

5. Ngoài biến mCount, bạn cũng cần một biến thành viên riêng để lưu tham chiếu đến TextView hiển thị số đếm, gọi biến này là mShowCount:

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

.....}

6. Bây giờ, đã có mShowCount, bạn có thể lấy tham chiếu đến TextView thông qua ID đã đặt trong tệp bố cục. Để lấy tham chiếu này chỉ một lần, hãy thực hiện trong phương thức onCreate(). Như bạn đã học trong một bài học khác, phương thức onCreate() được dùng để inflate bố cục, tức là đặt content view của màn hình bằng tệp XML bố cục. Bạn cũng có thể sử dụng onCreate() để lấy tham chiếu đến các phần tử UI khác trong bố cục, chẳng hạn như TextView. Tìm phương thức onCreate() trong MainActivity:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable(this);  
    ActionBar actionBar = getSupportActionBar();  
    if (actionBar != null) {  
        actionBar.setBackgroundDrawable(new ColorDrawable(Color.BLUE));  
    }  
    setContentView(R.layout.activity_main);  
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {  
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());  
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
        return insets;  
    });  
}
```

7. Thêm câu lệnh findViewById vào cuối phương thức onCreate() để gán giá trị cho mShowCount:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable(this);  
    ActionBar actionBar = getSupportActionBar();  
    if (actionBar != null) {  
        actionBar.setBackgroundDrawable(new ColorDrawable(Color.BLUE));  
    }  
    setContentView(R.layout.activity_main);  
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {  
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());  
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
        return insets;  
    });  
    mShowCount = (TextView) findViewById(R.id.show_count);  
}
```

Một View, giống như một chuỗi, là một tài nguyên có thể có một ID. Gọi `findViewById` với ID của một view sẽ trả về đối tượng View. Vì phương thức này trả về một View, bạn cần ép kiểu kết quả về loại view mà bạn mong đợi, trong trường hợp này là (`TextView`).

8. Sau khi đã gán tham chiếu cho `mShowCount`, bạn có thể sử dụng biến này để cập nhật văn bản của `TextView` với giá trị của biến `mCount`. Thêm đoạn mã sau vào phương thức `countUp()`:

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

Toàn bộ phương thức `countUp()` bây giờ trông như sau:

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

9. Chạy ứng dụng để xác minh rằng giá trị đếm tăng lên mỗi khi bạn nhấn nút Count.

8:37 5G 🔋 ⚡ 🔋



Hello Toast

TOAST

1

COUNT

Mẹo: Để tìm hiểu chi tiết về cách sử dụng ConstraintLayout, hãy xem Codelab "Using ConstraintLayout to design your views".

Thách thức lập trình

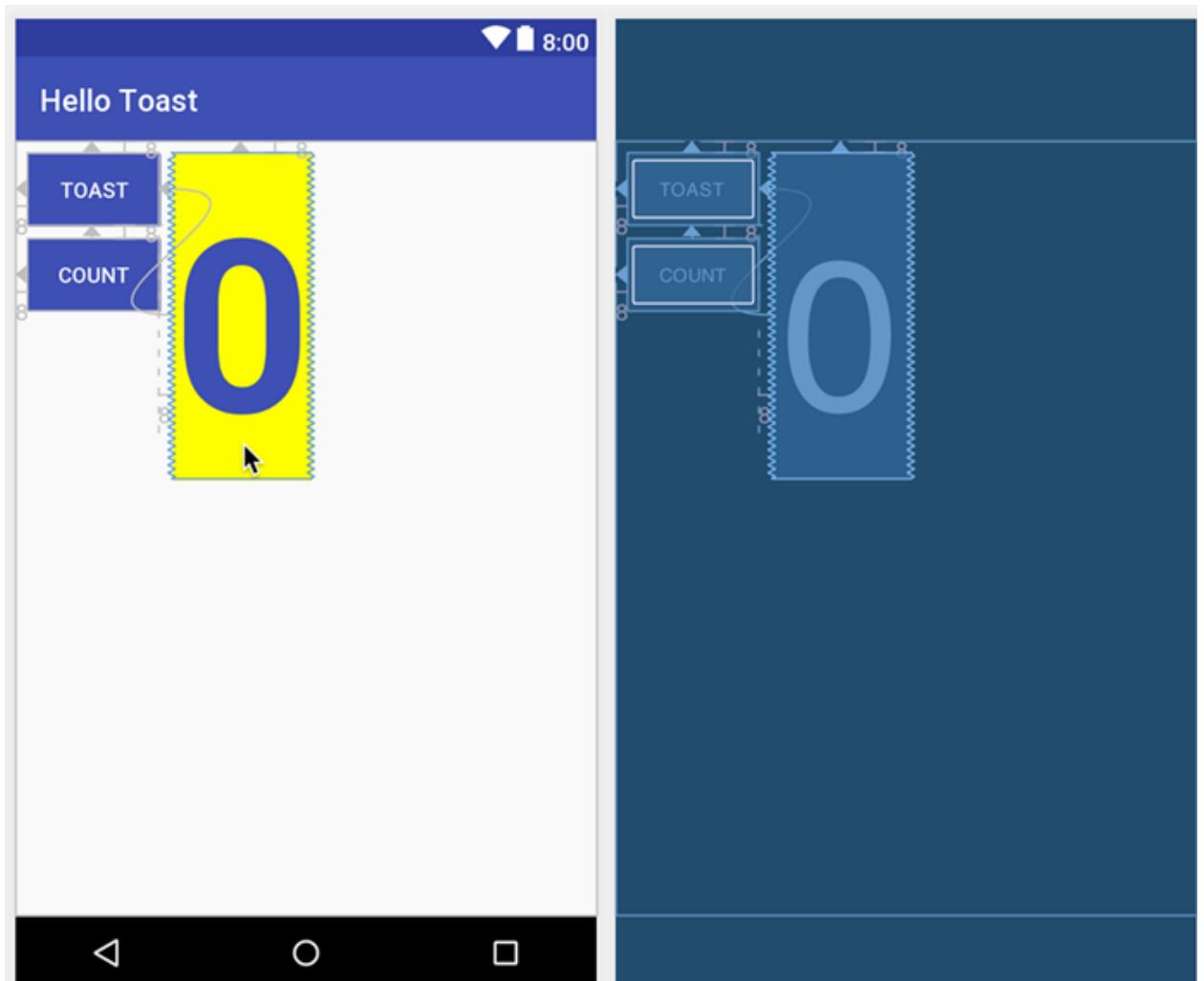
Lưu ý: Tất cả các thách thức lập trình đều không bắt buộc và không phải là điều kiện tiên quyết cho các bài học sau.

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



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

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





Tóm tắt:

View, ViewGroup và layouts:

- Tất cả các phần tử giao diện người dùng (UI) đều là các lớp con của lớp **View**, do đó kế thừa nhiều thuộc tính từ lớp **View**.
- Các phần tử **View** có thể được nhóm lại bên trong một **ViewGroup**, hoạt động như một container. Mỗi quan hệ này là mối quan hệ cha-con, trong đó **parent** là một **ViewGroup**, còn **child** là một **View** hoặc một **ViewGroup** khác.
- Phương thức **onCreate()** được sử dụng để **inflate layout**, nghĩa là thiết lập nội dung hiển thị của màn hình từ tệp XML layout. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện người dùng (UI) khác trong layout.
- Một **View**, giống như một chuỗi ký tự (string), là một tài nguyên có thể có ID. Phương thức **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 các phần tử và bố cục, hoặc tab **Text** để chỉnh sửa mã XML của bố cục.

- Trong tab **Design**, bảng **Palettes** hiển thị các phần tử giao diện người dùng (UI) có thể sử dụng trong bố cục ứng dụng, và bảng **Component tree** hiển thị cây phân cấp của các phần tử UI.
- Các phần tử UI trong bố cục được hiển thị trong các bảng **design** và **blueprint**. Tab **Attributes** hiển thị bảng **Attributes** để cài đặt các thuộc tính cho phần tử UI.
- Constraint handle: Nhấp vào một constraint handle (vòng tròn ở mỗi cạnh của phần tử), kéo đến một constraint khác hoặc đường viền cha để tạo ràng buộc. Ràng buộc được hiển thị bằng một đường gấp khúc.
- Resizing handle: Kéo tay nắm (hình vuông) để thay đổi kích thước phần tử. Trong khi kéo, tay nắm sẽ chuyển thành góc xiên.
- Autoconnect tool: Khi được bật, công cụ này tự động tạo hai hoặc nhiều ràng buộc cho phần tử UI với bố cục cha, dựa trên vị trí của phần tử.
- Bạn có thể xóa ràng buộc của một phần tử bằng cách chọn phần tử, di chuột qua để hiển thị nút **Clear Constraints**, và nhấp để xóa tất cả các ràng buộc. Để xóa một ràng buộc cụ thể, nhấp vào constraint handle tương ứng.
- Bảng **Attributes** cung cấp truy cập vào tất cả các thuộc tính XML có thể gán cho một phần tử UI. Nó cũng bao gồm một bảng định cỡ hình vuông gọi là **view inspector** ở trên cùng. Các biểu tượng trong hình vuông đại diện cho cài đặt chiều cao và chiều rộng.

Thuộc tính **layout_width** và **layout_height** thay đổi khi bạn thay đổi chiều cao và chiều rộng trong bảng điều khiển. Các giá trị có thể là:

1. **match_constraint**: Phần tử mở rộng để lấp đầy không gian bố cục cha (có tính đến lề nếu được đặt).
2. **wrap_content**: Phần tử co lại vừa với nội dung của nó. Nếu không có nội dung, phần tử sẽ trở nên vô hình.
3. **dp cố định**: Chỉ định kích thước cố định, phù hợp với kích thước màn hình của thiết bị.

Trích xuất tài nguyên chuỗi (String Resources):

Thay vì mã hóa cứng chuỗi, nên sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Thực hiện theo các bước sau:

1. Nhấp vào chuỗi mã hóa cứng để trích xuất, nhấn **Alt-Enter** (hoặc **Option-Enter** trên Mac) và chọn **Extract string resources** từ menu bật lên.
2. Đặt tên cho **Resource name**.

3. Nhấn **OK**. Điều này sẽ tạo một tài nguyên chuỗi trong tệp `values/res/string.xml`, và chuỗi trong mã sẽ được thay thế bằng tham chiếu tới tài nguyên: `@string/button_label_toast`.

Xử lý sự kiện nhấp (Handling Clicks):

- **Click handler** là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI.
- Để chỉ định một click handler cho một phần tử UI như **Button**, nhập tên phương thức trong trường **onClick** của bảng **Attributes** ở tab **Design**, hoặc trong trình chỉnh sửa XML bằng cách thêm thuộc tính `android:onClick` vào phần tử **Button**.
- Tạo click handler trong Activity chính bằng cách sử dụng tham số **View**. Ví dụ:

```
public void showToast(View view) {  
    // Xử lý khi nhấp  
}
```

- Có thể tìm thấy thông tin về tất cả các thuộc tính của **Button** trong tài liệu lớp **Button**, và tất cả các thuộc tính của **TextView** trong tài liệu lớp **TextView**.

Toast cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ popup nhỏ. Nó chỉ chiếm không gian đủ để chứa thông báo. Để tạo một instance của Toast, thực hiện các bước sau:

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

Bài học 1.2 Phần B: Trình chỉnh sửa bộ cục

Giới thiệu

Như đã học trong **1.2 Phần A: Giao diện tương tác đầu tiên**, bạn có thể xây dựng giao diện người dùng (UI) bằng **ConstraintLayout** trong trình chỉnh sửa bố cục. ConstraintLayout sắp xếp các phần tử UI bằng cách kết nối ràng buộc với các phần tử khác hoặc với các cạnh của bố cục. ConstraintLayout được thiết kế để dễ dàng kéo thả 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 (được gọi là **children** hoặc **child views**). Phần thực hành này giới thiệu nhiều tính năng hơn của **ConstraintLayout** và trình chỉnh sửa bố cục.

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

- **LinearLayout**: Một nhóm căn chỉnh các phần tử View con bên trong theo chiều ngang hoặc 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 dựa trên mối quan hệ giữa chúng với nhau hoặc với **ViewGroup** cha.

Những gì bạn cần biết trước:

Bạn cần có khả năng:

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

Những gì bạn sẽ học:

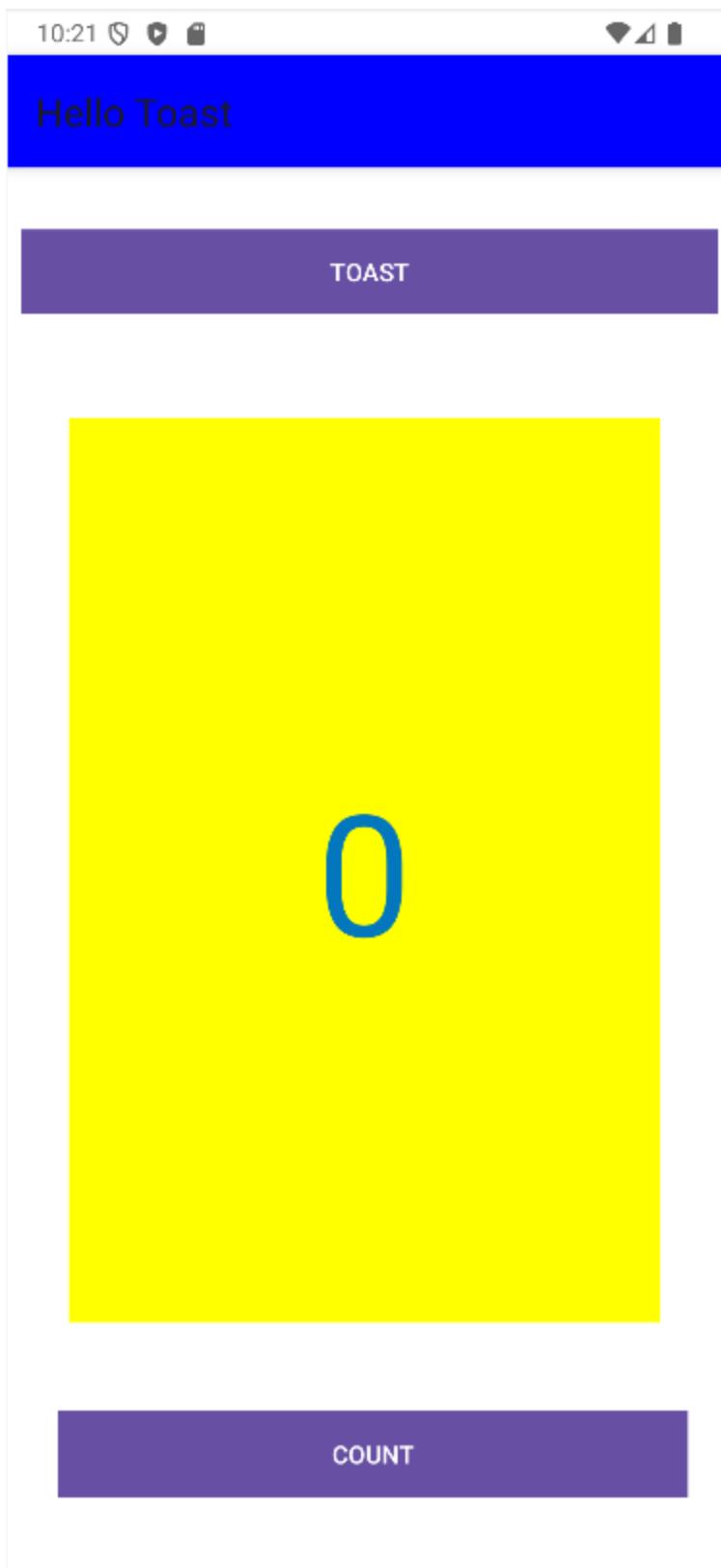
- Cách tạo một biến thể bố cục cho màn hình ngang (landscape).
- Cách tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Cách sử dụng ràng buộc đường cơ sở (baseline constraint) để căn chỉnh các phần tử UI với văn bản.
- Cách sử dụng các nút gói (pack) và căn chỉnh (align) để căn chỉnh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

Những gì bạn sẽ làm:

- Tạo một biến thể bố cục cho màn hình ngang.
- Tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Sửa đổi 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 gói và căn chỉnh 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 view trong bố cục chính để liên kết tương đối với nhau.

Tổng quan về ứng dụng:

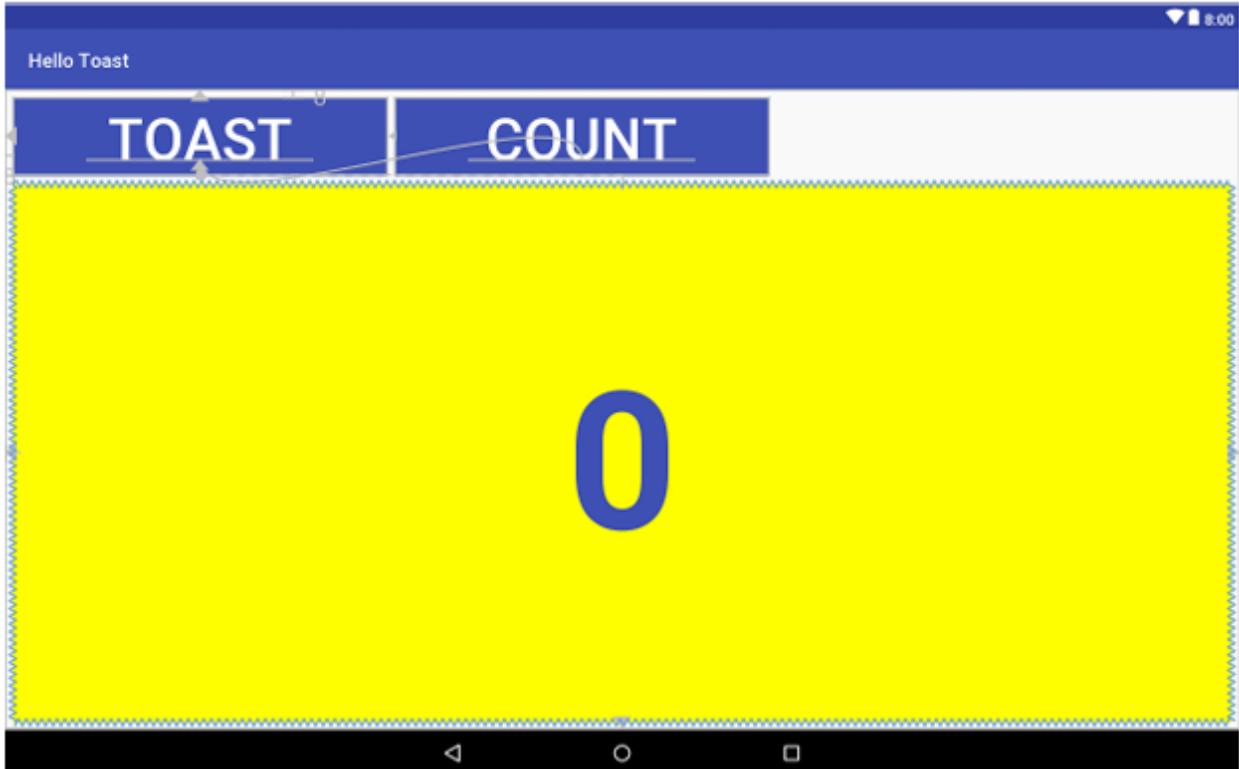
Ứng dụng **Hello Toast** từ bài học trước sử dụng **ConstraintLayout** để sắp xếp các phần tử UI trong bố cục của Activity, như minh họa trong hình dưới đây.



Để thực hành thêm với **ConstraintLayout**, bạn sẽ tạo một biến thể của bố cục này dành cho màn hình ngang (horizontal orientation), như minh họa trong hình dưới đây.



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



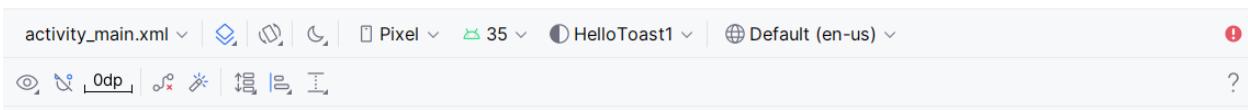
Bạn cũng sẽ tìm hiểu về các lớp con khác của **ViewGroup** như **LinearLayout** và **RelativeLayout**, đồng thời thay đổi bố cục của ứng dụng **Hello Toast** để sử dụng chúng.

Nhiệm vụ 1: Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu bạn thay đổi bố cục của ứng dụng **Hello Toast** để phù hợp với cả hai chế độ **dọc** và **ngang**. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo các biến thể bố cục cho các thiết bị di động ở chế độ **dọc** hoặc **ngang**, cũng như cho các thiết bị màn hình lớn như máy tính bảng.

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

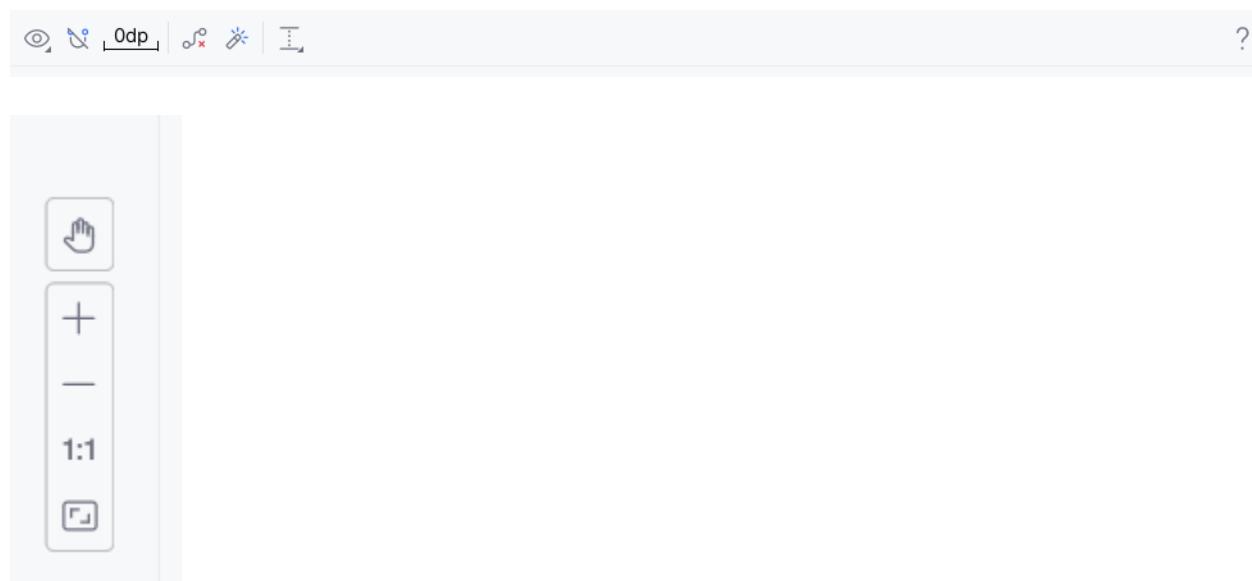
Thanh công cụ trên cùng giúp bạn cấu hình cách hiển thị bản xem trước của bố cục trong trình chỉnh sửa bố cục:



Các bước chính:

1. **Chọn bề mặt thiết kế (Design Surface):** Chọn **Design** để hiển thị bản xem trước màu sắc của bố cục, hoặc **Blueprint** để chỉ hiển thị các đường viền của từng phần tử giao diện người dùng (UI). Để xem cả hai chế độ cạnh nhau, chọn **Design + Blueprint**.
2. **Thay đổi hướng hiển thị trong trình chỉnh sửa (Orientation in Editor):** Chọn **Portrait** (chế độ dọc) hoặc **Landscape** (chế độ ngang) để xem trước bố cục theo chiều dọc hoặc ngang. Điều này rất 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 cho từng hướng, chọn **Create Landscape Variation** hoặc các biến thể khác.
3. **Chọn thiết bị trong trình chỉnh sửa (Device in Editor):** Chọn loại thiết bị (ví dụ: điện thoại/máy tính bảng, Android TV hoặc Android Wear).
4. **Chọn phiên bản API trong trình chỉnh sửa (API Version in Editor):** Chọn phiên bản Android để hiển thị bản xem trước.
5. **Chọn chủ đề trong trình chỉnh sửa (Theme in Editor):** Chọn một chủ đề (ví dụ: **AppTheme**) để áp dụng cho bản xem trước.
6. **Chọn ngôn ngữ và khu vực (Locale in Editor):** Chọn ngôn ngữ và khu vực cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có trong tài nguyên chuỗi văn bản (**string resources**). Bạn cũng có thể chọn **Preview as Right To Left** để xem bố cục như khi chọn một ngôn ngữ đọc từ phải sang trái (**RTL**).

Thanh công cụ thứ hai giúp bạn cấu hình cách hiển thị các phần tử giao diện người dùng trong **ConstraintLayout**, cũng như phóng to hoặc cuộn bản xem trước.



Trong hình trên:

1. **Hiển thị (Show):** Chọn **Show Constraints** và **Show Margins** để hiển thị hoặc ẩn các ràng buộc và khoảng cách trong bản xem trước.
2. **Autoconnect:** Bật hoặc tắt tính năng Autoconnect. Khi tính năng này được bật, bạn có thể kéo bất kỳ phần tử nào (như Button) đến bất kỳ phần nào trong bố cục để tự động tạo ràng buộc với bố cục cha.
3. **Xóa tất cả ràng buộc (Clear All Constraints):** Xóa toàn bộ ràng buộc trong bố cục.
4. **Suy luận ràng buộc (Infer Constraints):** Tạo ràng buộc bằng cách suy luận.
5. **Khoảng cách mặc định (Default Margins):** Thiết lập khoảng cách mặc định.
6. **Gói (Pack):** Gom nhóm hoặc mở rộng các phần tử được chọn.
7. **Căn chỉnh (Align):** Căn chỉnh các phần tử được chọn.
8. **Đường dẫn hướng (Guidelines):** Thêm đường dẫn hướng dọc hoặc ngang.
9. **Điều khiển phóng to/thu nhỏ (Zoom/pan controls):** Phóng to hoặc thu nhỏ bản xem trước.

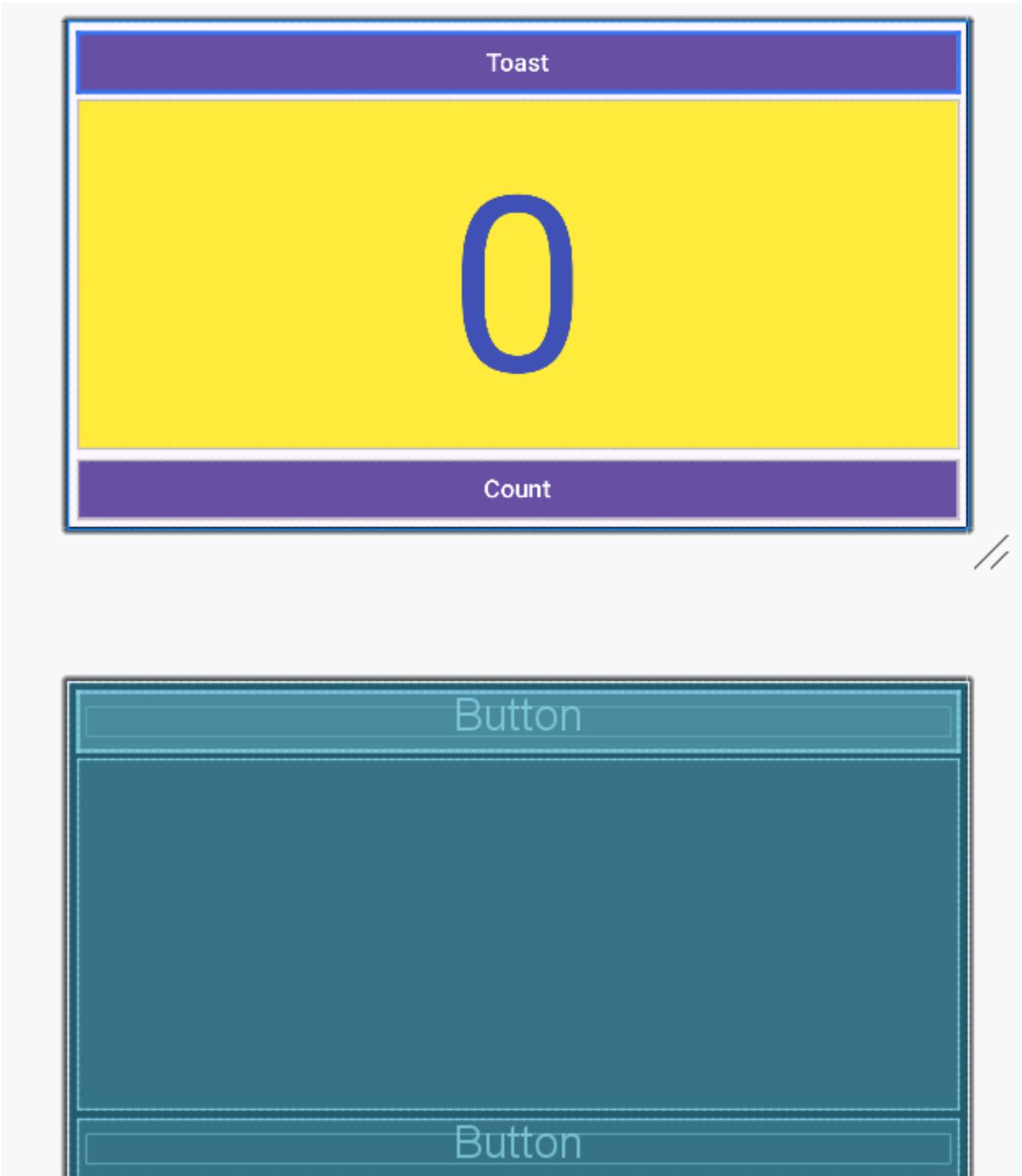
Mẹo: Để tìm hiểu thêm về cách sử dụng trình chỉnh sửa bố cục, xem **Build a UI with Layout Editor**. Để học cách xây dựng bố cục với **ConstraintLayout**, xem **Build a Responsive UI with ConstraintLayout**.

1.1 Xem trước bố cục ở chế độ ngang

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

1. Mở ứng dụng **Hello Toast** từ bài học trước.
 - **Lưu ý:** Nếu bạn đã tải xuống mã hoàn chỉnh cho HelloToast, hãy xóa các bố cục chế độ ngang (**landscape**) và màn hình cực lớn (**extra-large**) đã hoàn thành mà bạn sẽ tạo trong nhiệm vụ này.
 - Chuyển từ chế độ hiển thị **Project > Android** sang **Project > Project Files** trong bảng **Project**. Sau đó, mở rộng đường dẫn **app > src/main > res**, chọn cả hai thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**.
 - Sau khi hoàn tất, chuyển lại bảng **Project** về **Project > Android**.
2. Mở tệp **activity_main.xml** trong thư mục bố cục (**layout**). Nhấp vào tab **Design** nếu nó chưa được chọn.

3. Nhấp vào nút **Orientation in Editor** trên thanh công cụ ở phía trên để thay đổi hướng hiển thị.
4. Trong menu thả xuống, chọn **Switch to Landscape**. Bộ cục sẽ xuất hiện ở chế độ ngang, như minh họa bên dưới. Để quay lại chế độ dọc, chọn **Switch to Portrait**.



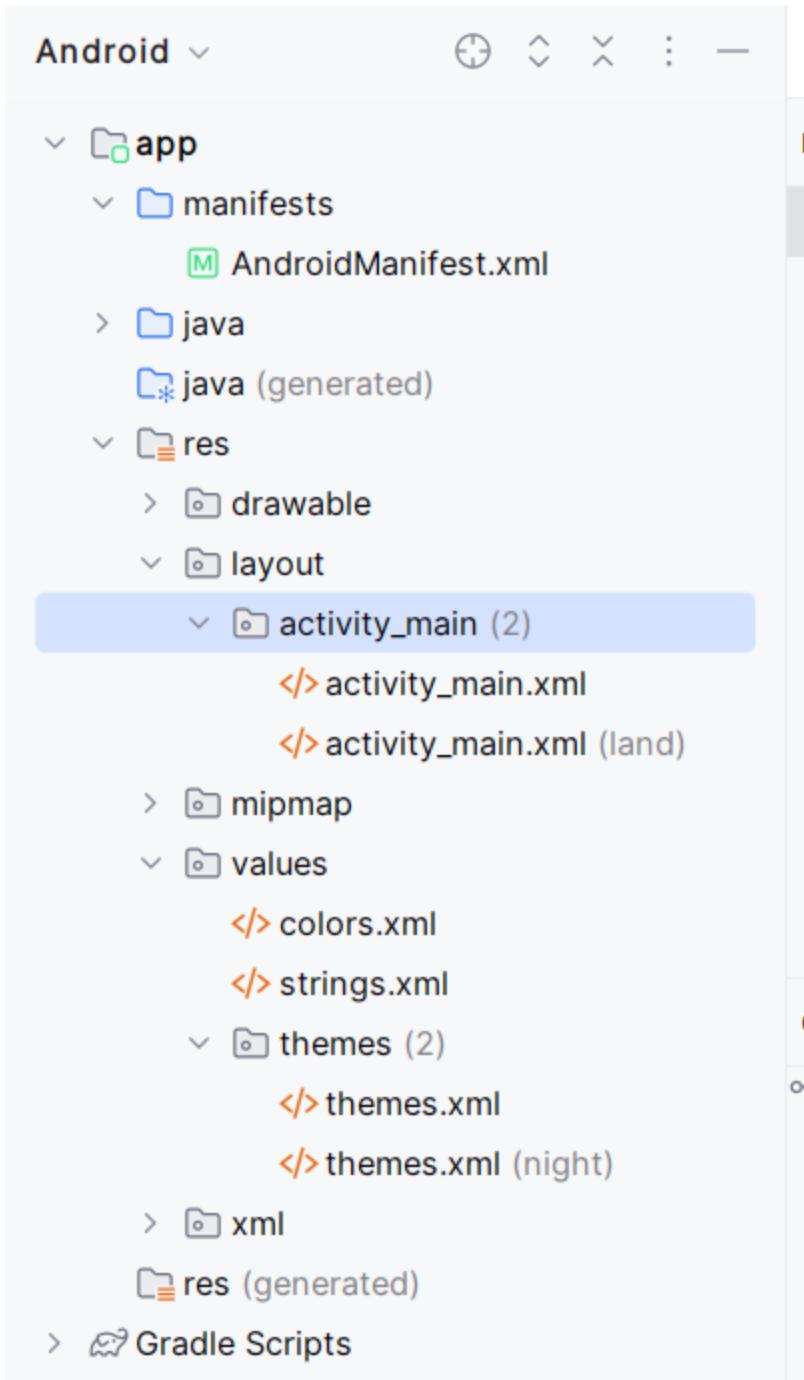
1.3 Trình chỉnh sửa bố cục

Sự khác biệt về mặt hình ảnh giữa chế độ dọc và ngang trong bố cục này là chữ số **0** trong phần tử **show_count** (**TextView**) nằm quá thấp đối với chế độ ngang, quá gần với nút **Count** và kích thước cố định của **TextView** (160sp) có thể khiến bố cục trong chế độ ngang trông không cân đối, với chữ số **0** quá lớn hoặc không được căn giữa. Để giải quyết vấn đề này trong chế độ ngang mà vẫn giữ nguyên bố cục cho chế độ dọc, bạn có thể tạo một biến thể bố cục cho ứng dụng **Hello Toast**. Thực hiện theo các bước sau:

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

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

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



1.3 Xem trước bố cục trên các thiết bị khác nhau

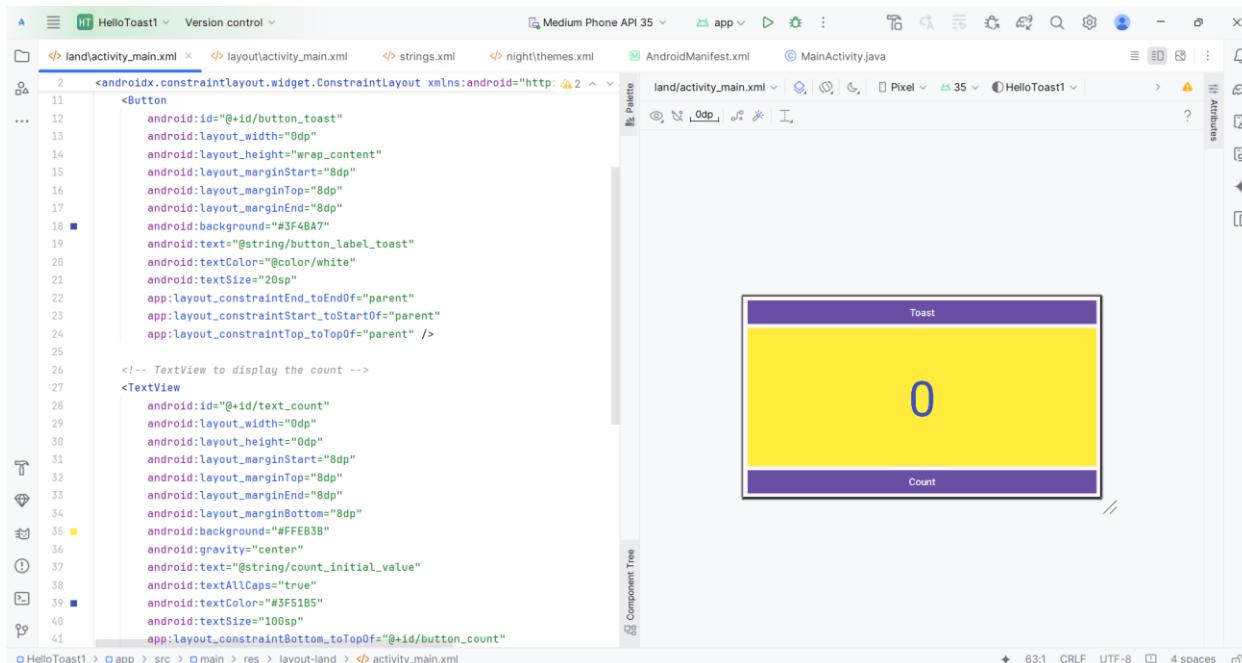
Bạn có thể xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thực hoặc trình giả lập. Thực hiện các bước sau:

1. Tab **land/activity_main.xml** vẫn đang mở trong trình chỉnh sửa bố cục. Nếu không, hãy nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục **layout**.

- Nhấp vào nút **Device in Editor** trên thanh công cụ phía trên.
- Trong menu thả xuống, chọn một thiết bị khác. Ví dụ, chọn **Nexus 4, Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong các bản xem trước. Những khác biệt này xảy ra do kích thước văn bản cố định của **TextView**.

Thay đổi bố cục cho chế độ ngang

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



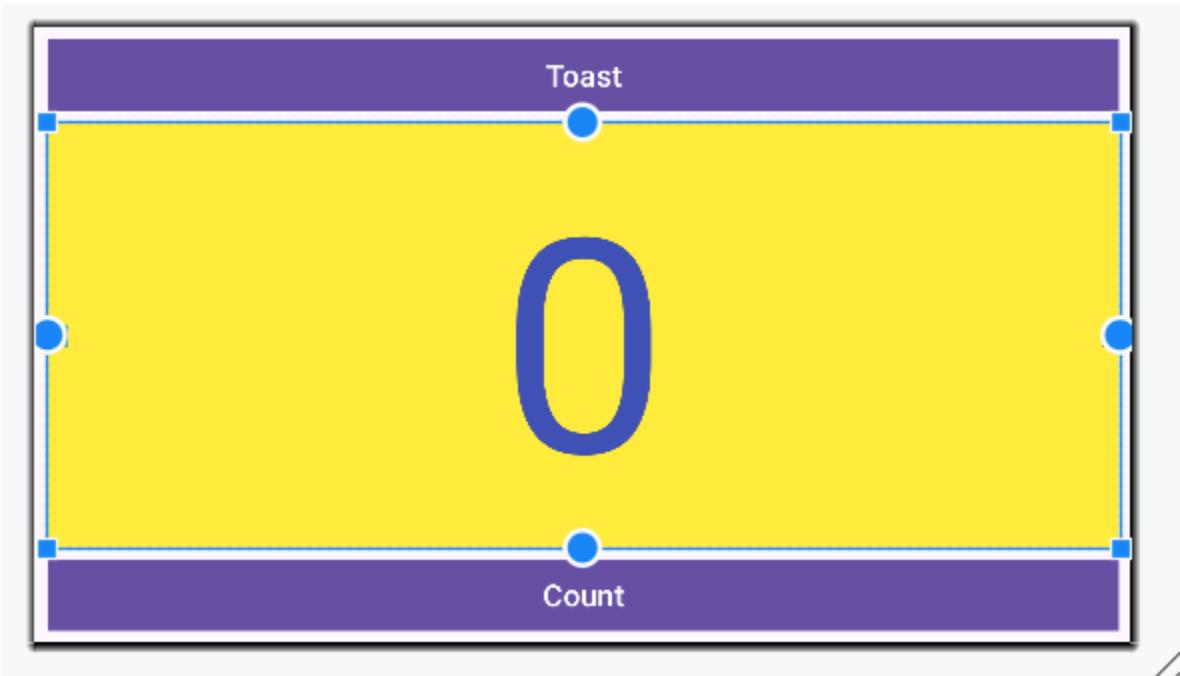
Thay đổi bố cục với các bước chi tiết

Hình minh họa:

- Tab Preview:** Sử dụng để hiển thị bảng xem trước bố cục.
- Preview Pane:** Hiển thị cách bố cục trông trên thiết bị.
- XML Code:** Phần mã nguồn của bố cục trong tab **Text**.

Các bước thay đổi bố cục:

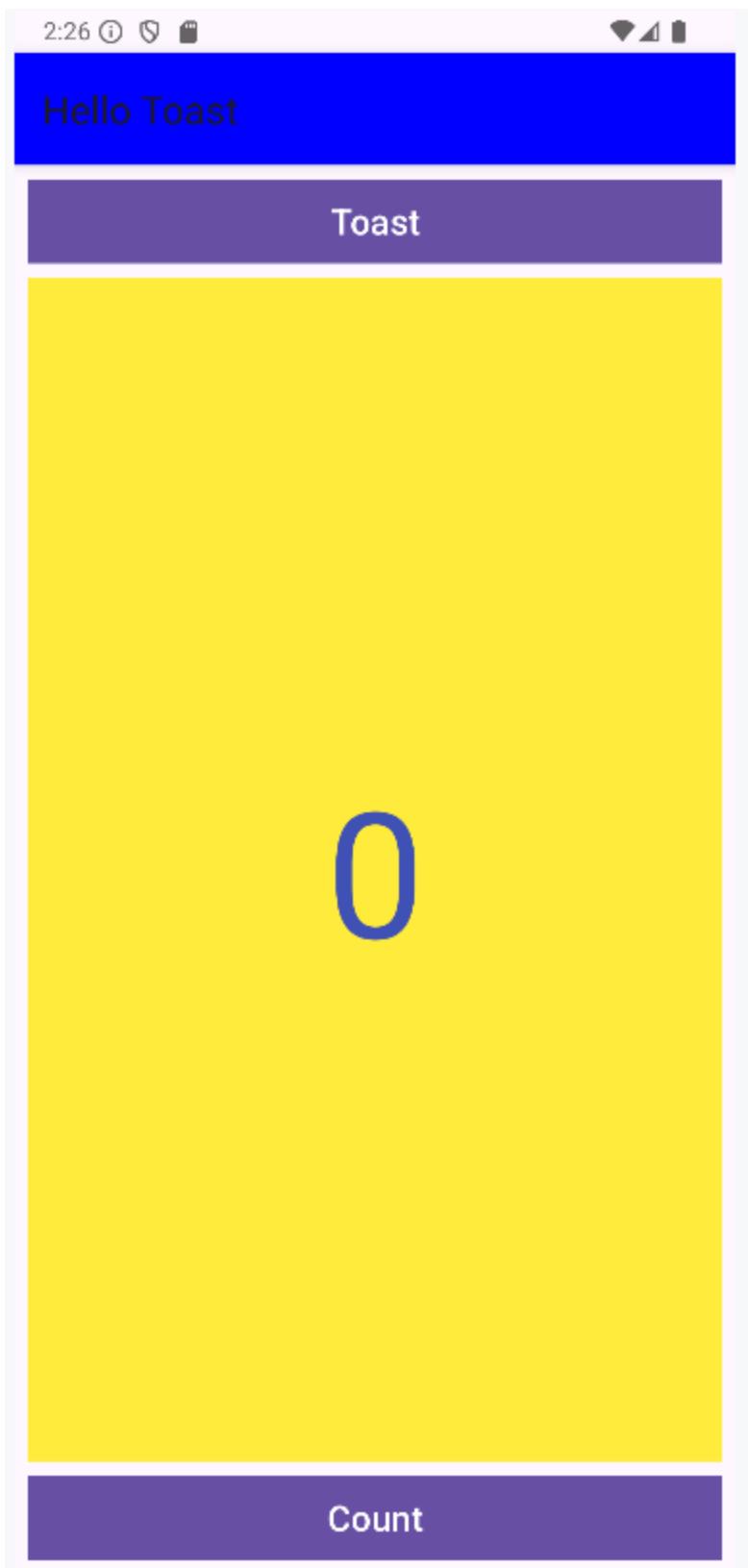
1. Đảm bảo tab **land/activity_main.xml** vẫn mở trong trình chỉnh sửa bố cục. Nếu không, nhấp đúp vào tệp **activity_main.xml (land)** trong thư mục **layout**.
2. Chuyển sang tab **Text** và bật tab **Preview** (nếu chưa được chọn).
3. Trong mã XML, tìm phần tử **TextView**.
4. Thay đổi thuộc tính `android:textSize="100sp"` thành `android:textSize="120sp"`. Kết quả thay đổi sẽ hiển thị ngay lập tức trong bản xem trước.



5. Sử dụng menu thả xuống **Device in Editor** để chọn các thiết bị khác nhau và xem cách bố cục hiển thị ở chế độ ngang trên các thiết bị đó.

Lưu ý trong trình chỉnh sửa:

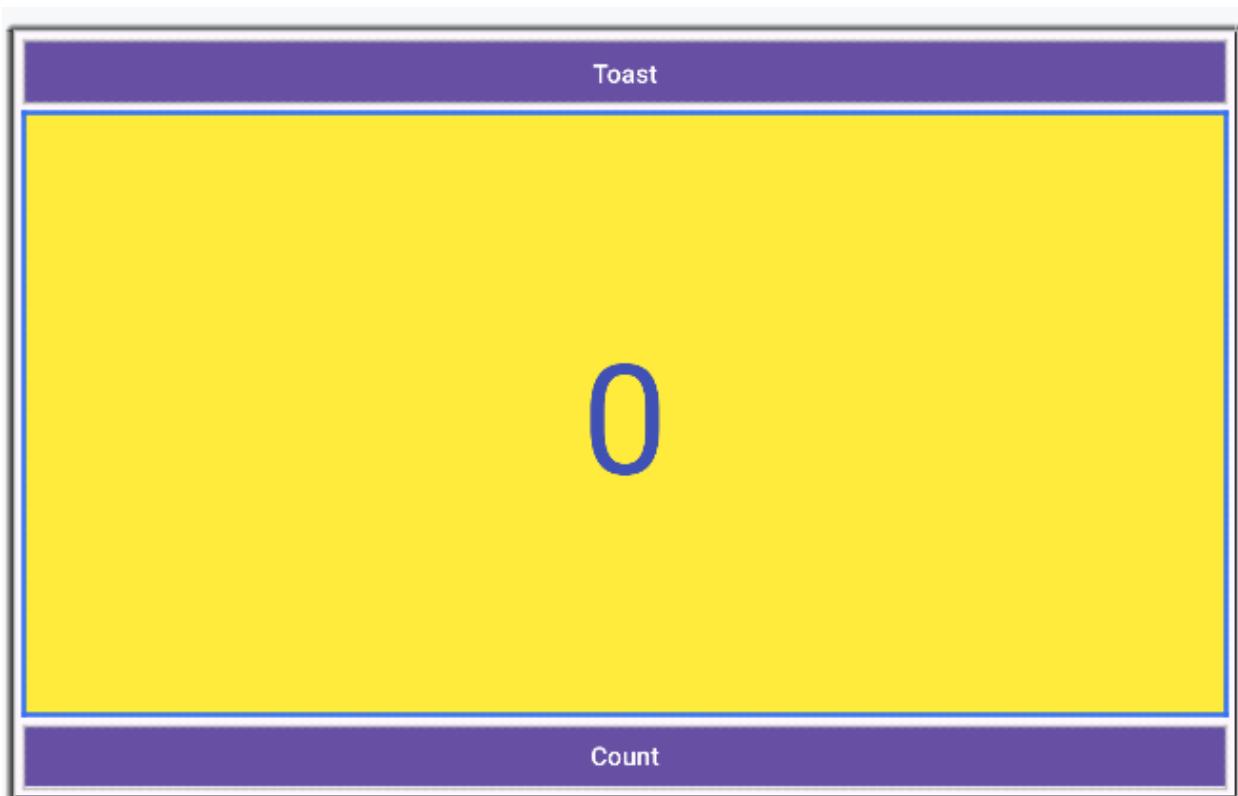
- Tab **land/activity_main.xml** hiển thị bố cục cho chế độ ngang.
 - Tab **activity_main.xml** hiển thị bố cục không thay đổi cho chế độ dọc. Bạn có thể chuyển đổi qua lại giữa hai tab để so sánh.
6. Chạy ứng dụng trên trình giả lập hoặc thiết bị thực. Chuyển hướng hiển thị từ dọc sang ngang để xem cách cả hai bố cục hoạt động.





1.5 Tạo một biến thể bố cục cho máy tính bảng

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



Để điều chỉnh bố cục phù hợp với máy tính bảng mà không ảnh hưởng đến bố cục cho các thiết bị điện thoại ở chế độ ngang hoặc dọc, hãy thực hiện các bước sau:

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị các bảng thiết kế và bản vẽ.
2. Nhấp vào nút **Orientation in Editor** trên thanh công cụ phía trên.
3. Chọn **Create layout x-large Variation**.

Sau khi thực hiện:

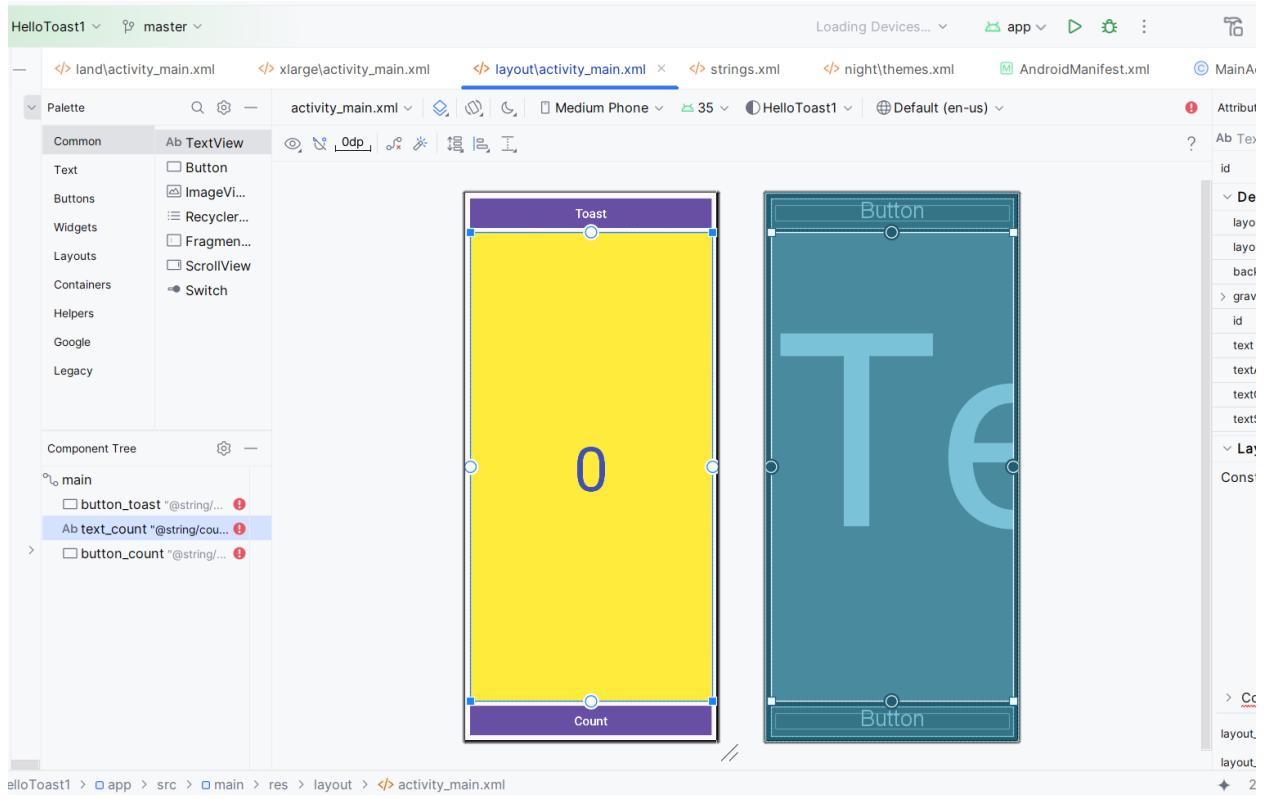
- Một cửa sổ chỉnh sửa mới sẽ mở ra với tab **xlarge/activity_main.xml**, hiển thị bố cục dành riêng cho thiết bị có kích thước màn hình máy tính bảng.
- Trình chỉnh sửa cũng tự động chọn một thiết bị máy tính bảng, chẳng hạn như **Nexus 9** hoặc **Nexus 10**, để hiển thị bản xem trước.

Bạn có thể thay đổi bố cục này, dành riêng cho máy tính bảng, mà không làm thay đổi các bố cục khác.

1.6 Thay đổi biến thể bố cục cho máy tính bảng

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

1. **Tắt công cụ Autoconnect** trên thanh công cụ. Ở bước này, hãy đảm bảo rằng công cụ đã bị vô hiệu hóa.
2. Xóa tất cả các ràng buộc (constraints) trong bố cục bằng cách nhấp vào nút **Clear All Constraints** trên thanh công cụ.
 - Khi các ràng buộc đã bị xóa, bạn có thể tự do di chuyển và thay đổi kích thước các phần tử trên bố cục.
3. Trình chỉnh sửa bố cục cung cấp các tay cầm (handles) ở bốn góc của mỗi phần tử để thay đổi kích thước chúng. Trong **Component Tree**, chọn **TextView** có tên **show_count**. Để dời **TextView** ra ngoài đường đi, giúp bạn dễ dàng kéo các phần tử **Button**, hãy kéo một góc của nó để thay đổi kích thước, như được minh họa trong hình động dưới đây.



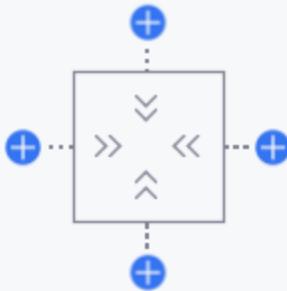
Khi thay đổi kích thước một phần tử, nó sẽ cố định các giá trị chiều rộng và chiều cao. Tránh cố định kích thước cho hầu hết các phần tử, vì điều này có thể dẫn đến giao diện không nhất quán trên các màn hình có kích thước và mật độ khác nhau. Trong bước này, việc thay đổi kích thước chỉ được thực hiện để di chuyển phần tử tạm thời, và các kích thước sẽ được điều chỉnh lại trong bước khác.

Bước 4:

Chọn nút **button_toast** trong **Component Tree**, nhấp vào tab **Attributes** để mở bảng thuộc tính, và thay đổi **textSize** thành **60sp** (#1 trong hình bên dưới), **layout_width** thành **wrap_content** (#2 trong hình bên dưới)

Button		button_toa
id	button_toast	
Declared Attributes		+ -
layout_width	wrap_content	▾
layout_height	wrap_content	▾
background	#3F4BA7	
id	button_toast	
text	@string/button_label_...	
textColor	@color/white	
textSize	60sp	▾
Layout		

Constraint Widget



> Constraints (0)

layout_width

wrap_content

layout_height

wrap_content

visibility

visibility

Như được hiển thị ở phía bên phải của hình trên (2), bạn có thể nhập vào điều khiển chiều rộng của trình kiểm tra chế độ xem, xuất hiện dưới dạng hai đoạn ở hai bên trái và 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 nếu văn bản của **Button** được bản địa hóa sang một ngôn ngữ khác, nút **Button** sẽ hiển thị rộng hơn hoặc hẹp hơn để phù hợp với từ trong ngôn ngữ khác.

5. Chọn nút **button_count** trong **Component Tree**, thay đổi **textSize** thành **60sp** và **layout_width** thành **wrap_content**, sau đó kéo nút **Button** lên trên **TextView** vào một không gian trống trong bố cục.

1.7 Sử dụng ràng buộc đường cơ sở (baseline constraint)

Bạn có thể căn chỉnh một phần tử giao diện người dùng (UI) chứa văn bản, như **TextView** hoặc **Button**, với một phần tử giao diện khác cũng chứa văn bản.

Ràng buộc đường cơ sở (**baseline constraint**) cho phép bạn ràng buộc các phần tử sao cho các đường cơ sở của văn bản được căn thẳng hàng.

1. Ràng buộc nút **button_toast** vào phía trên và bên trái của bố cục. Kéo nút **button_count** đến một vị trí gần nút **button_toast**. Sau đó, ràng buộc nút **button_count** vào phía bên trái của nút **button_toast**, như minh họa trong hình động.



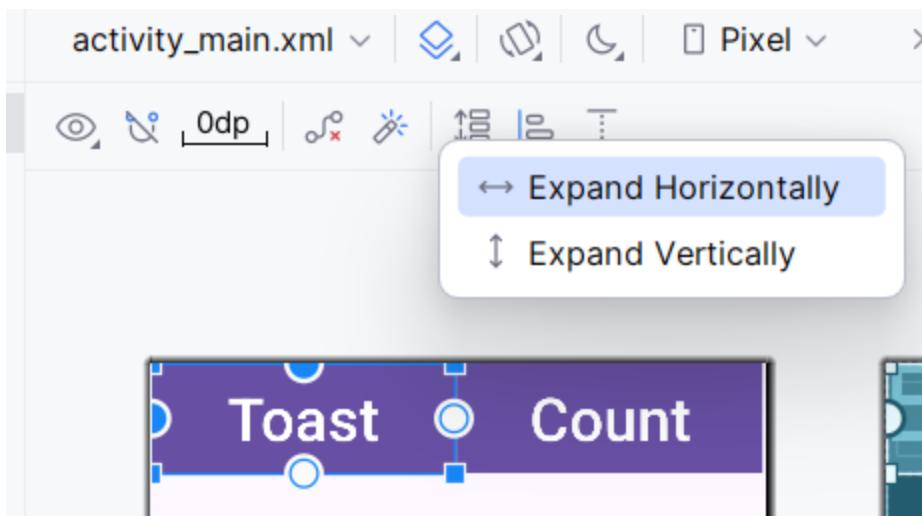
2. **Sử dụng ràng buộc đường cơ sở (baseline constraint)**, bạn có thể ràng buộc nút **button_count** sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của nút **button_toast**. Chọn phần tử **button_count**, sau đó di chuyển con trỏ chuột qua phần tử cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.
3. Nhấp vào nút ràng buộc đường cơ sở. Tay cầm ràng buộc đường cơ sở sẽ xuất hiện, nhấp nháy màu xanh lá. Nhấp và kéo một đường ràng buộc đường cơ sở đến đường cơ sở của phần tử **button_toast** để hoàn tất.



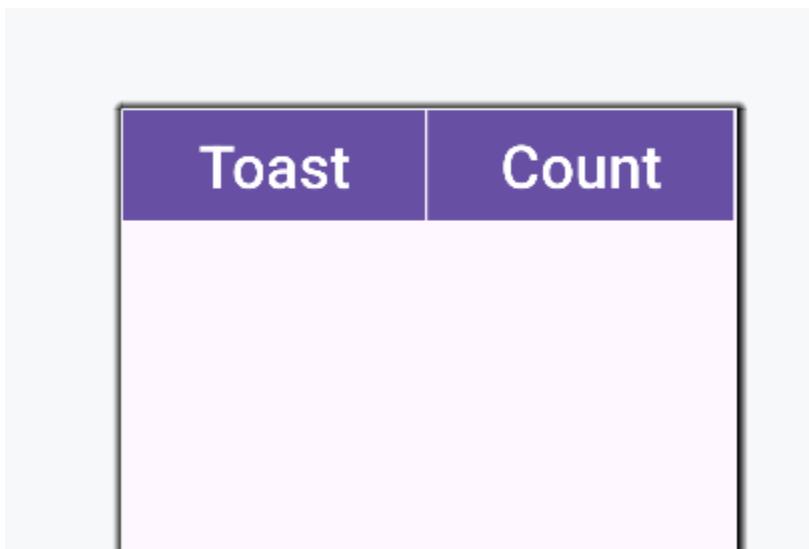
1.8 Mở rộng các nút theo chiều ngang

Nút **pack** trên thanh công cụ cung cấp các tùy chọn để sắp xếp hoặc mở rộng các phần tử giao diện người dùng đã chọn. Bạn có thể sử dụng nó để sắp xếp đều các nút **Button** theo chiều ngang trên giao diện.

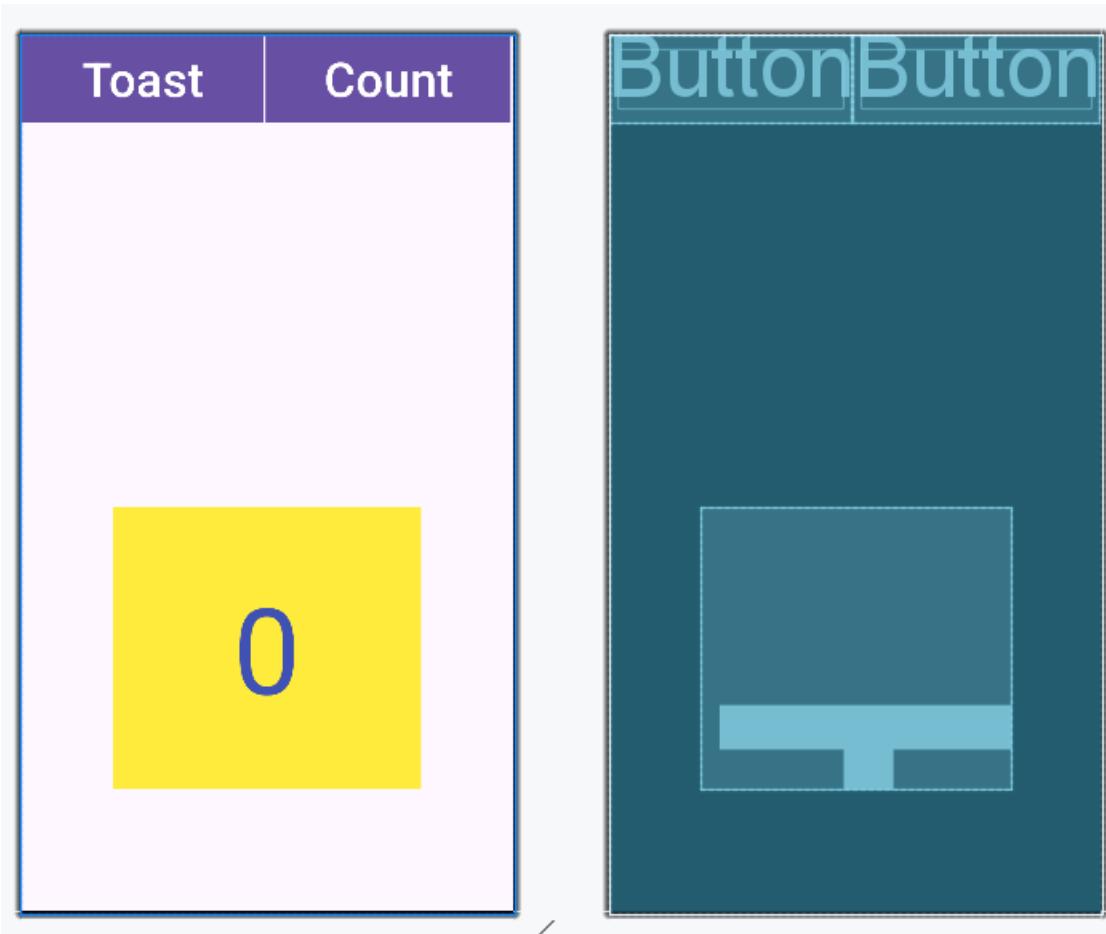
1. Chọn nút **button_count** trong **Component Tree**, sau đó nhấn giữ **Shift** và chọn thêm nút **button_toast** để cả hai nút đều được chọn.
2. Nhấp vào nút **pack** trên thanh công cụ, sau đó chọn tùy chọn **Expand Horizontally** như minh họa trong hình.



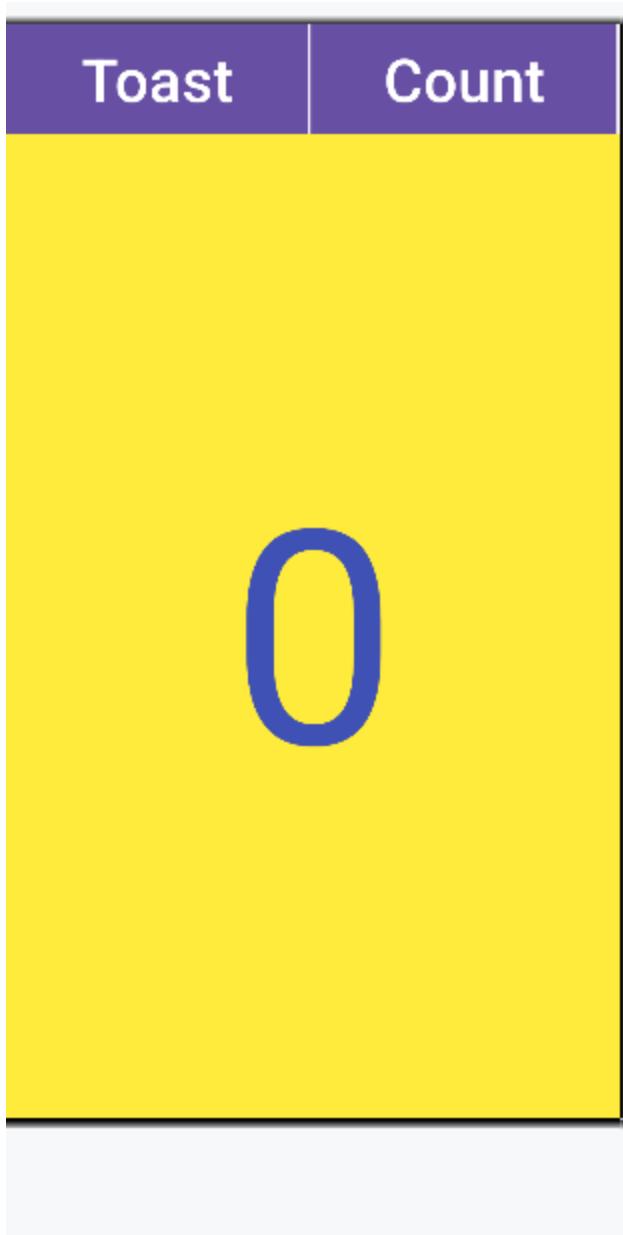
Các phần tử **Button** sẽ mở rộng theo chiều ngang để lấp đầy toàn bộ giao diện như minh họa bên dưới.



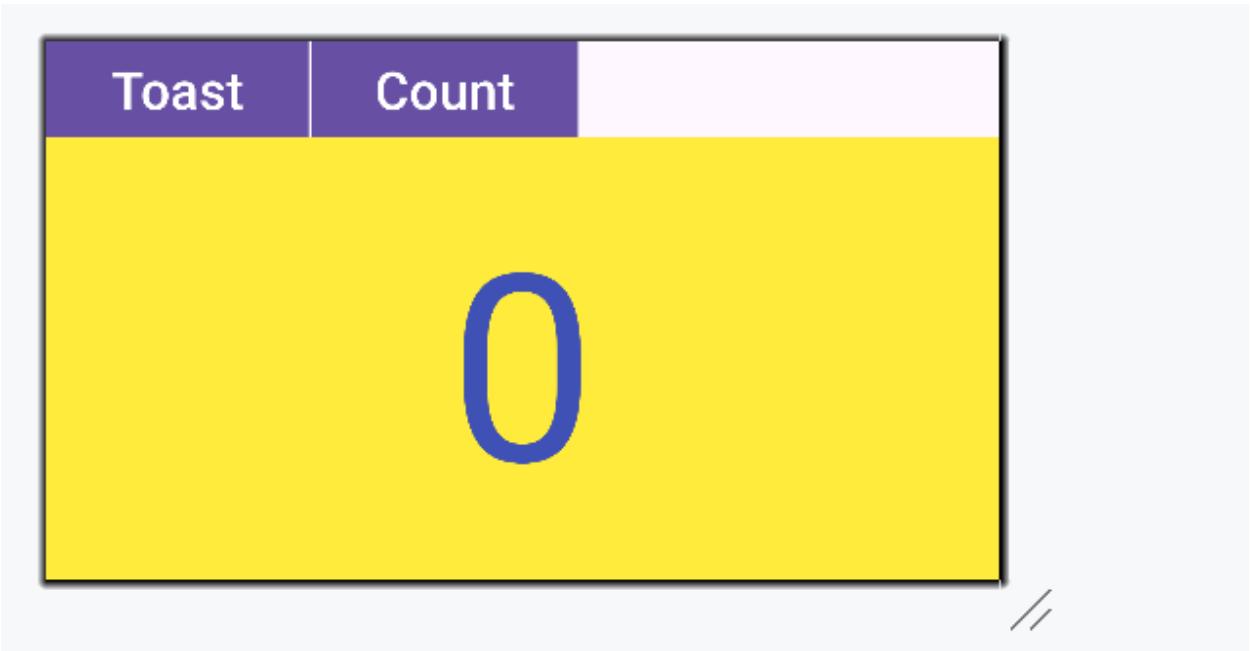
3. Để hoàn thiện bố cục, hãy ràng buộc **show_count TextView** vào phía dưới của nút **button_toast** và ràng buộc nó với các cạnh bên cũng như cạnh dưới của bố cục, như được minh họa trong hình động bên dưới.



4. Các bước cuối cùng là thay đổi **layout_width** và **layout_height** của **show_count TextView** thành **Match Constraints** và đặt **textSize** thành **200sp**. Bố cục cuối cùng sẽ trông như hình minh họa bên dưới.



5. Nhấn vào nút **Orientation in Editor** trên thanh công cụ ở phía trên và chọn **Switch to Landscape** (Chuyển sang chế độ ngang). Giao diện của bố cục trên máy tính bảng sẽ xuất hiện với hướng nằm ngang như hình dưới đây. (Bạn có thể chọn **Switch to Portrait** để quay lại chế độ dọc).

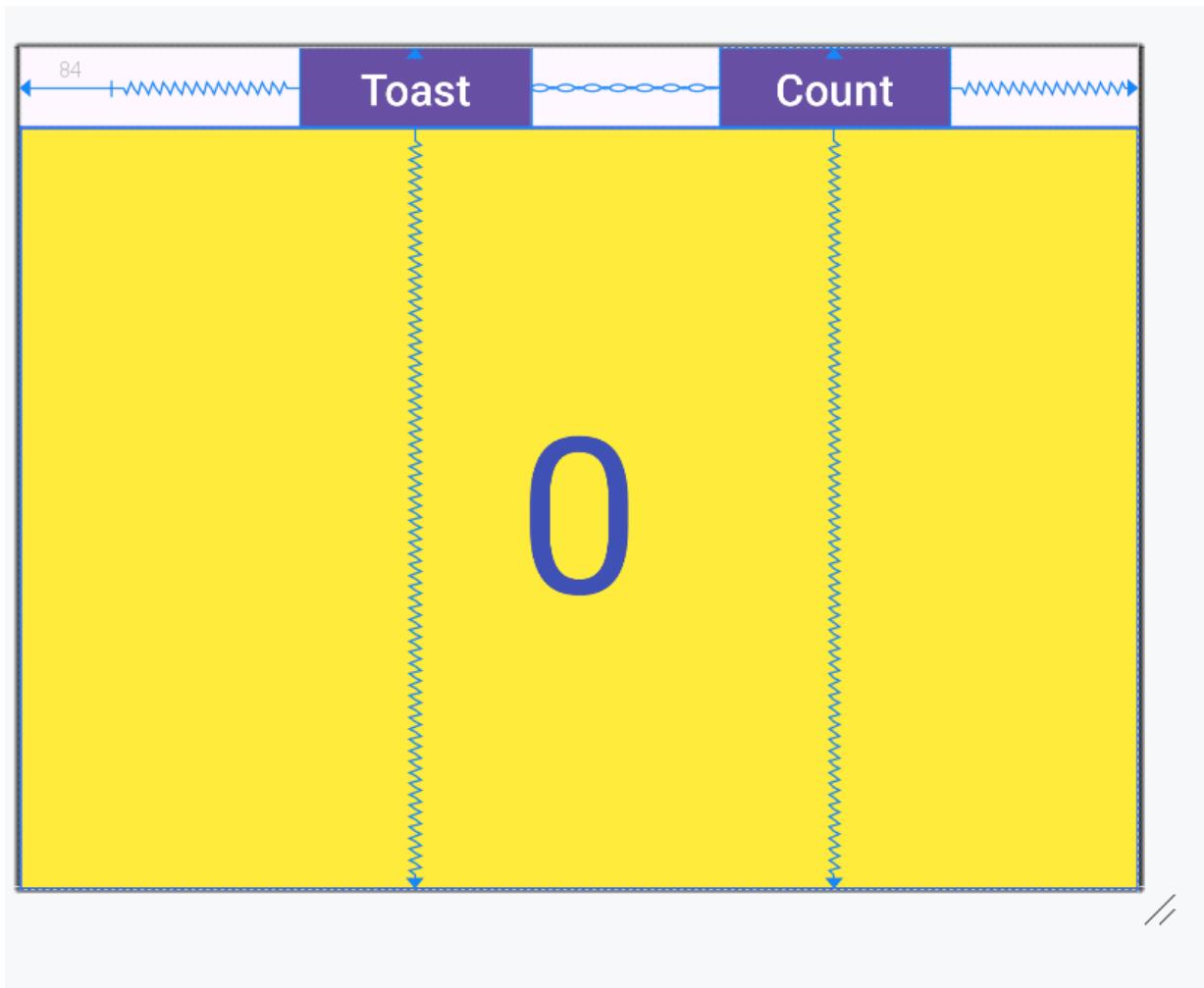


6. Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng màn hình sau khi chạy ứng dụng để xem giao diện trông như thế nào trên các loại thiết bị khác nhau. Bạn đã thành công trong việc tạo một ứng dụng có giao diện người dùng (UI) hoạt động đúng trên điện thoại và máy tính bảng với các kích thước và mật độ màn hình khác nhau.

Mẹo: Để có hướng dẫn chi tiết về cách sử dụng ConstraintLayout, hãy xem **Using ConstraintLayout to design your views**.

Thử thách: Để hỗ trợ giao diện ngang (landscape) cho máy tính bảng, bạn có thể căn giữa các nút (Button) trong tệp activity_main.xml (xlarge) để chúng xuất hiện như hình minh họa bên dưới.

Gợi ý: Chọn các phần tử, nhấp vào nút căn chỉnh trong thanh công cụ, và chọn **Căn giữa theo chiều ngang (Center Horizontally)**.



Bài 2: Thay đổi bố cục thành LinearLayout

LinearLayout là một **ViewGroup** sắp xếp tập hợp các **view** theo hàng ngang hoặc hàng dọc.

LinearLayout là một trong những bố cục phổ biến nhất vì nó đơn giản và nhanh. Nó thường được sử dụng trong một **view group** khác để sắp xếp các phần tử giao diện người dùng theo chiều ngang hoặc dọc.

LinearLayout yêu cầu có các thuộc tính sau:

- **layout_width**
- **layout_height**
- **orientation**

layout_width và **layout_height** có thể nhận một trong các giá trị sau:

- **match_parent**: 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 của màn hình (**parent view**).
- **wrap_content**: Thu nhỏ kích thước view sao cho nó vừa đủ chứa nội dung. Nếu không có nội dung, view sẽ trở nên vô hình.
- **Số dp cố định (density-independent pixels)**: Xác định 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.

orientation có thể là:

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

Thay đổi root view group thành **LinearLayout**

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), và nhấp vào tab **Text** ở cuối khung chỉnh sửa để xem mã XML. Ở dòng đầu tiên của mã XML, bạn sẽ thấy dòng sau:

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Thay đổi thẻ **<android.support.constraint.ConstraintLayout** thành **<LinearLayout** để mã XML trông như sau:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

4. Đảm bảo rằng thẻ đóng ở cuối mã đã được thay đổi thành **</LinearLayout>** (Android Studio sẽ tự động thay đổi thẻ đóng nếu bạn thay đổi thẻ mở). Nếu nó không thay đổi tự động, hãy chỉnh sửa thủ công.
5. Ngay dưới dòng thẻ **<LinearLayout**, thêm thuộc tính sau vào sau thuộc tính **android:layout_height**:

```
    android:layout_height="match_parent"  
    android:orientation="vertical"
```

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

Thay đổi thuộc tính của các phần tử để phù hợp với **LinearLayout**

Thực hiện các bước sau để thay đổi thuộc tính của các phần tử giao diện người dùng sao cho chúng hoạt động với **LinearLayout**:

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), và nhấp vào tab **Text**.
3. Tìm phần tử **Button** có id là **button_toast**, và thay đổi thuộc tính sau:

```
    android:id="@+id/button_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#3E/RA7"
```

4. Xóa các thuộc tính sau khỏi phần tử **button_toast**.

```
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
```

5. Tìm phần tử **Button** có id là **button_count**, và thay đổi thuộc tính sau.

```
    android:id="@+id/button_count"
    android:layout_width="match_parent"
```

6. Xóa các thuộc tính sau khỏi phần tử **button_count**.

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
```

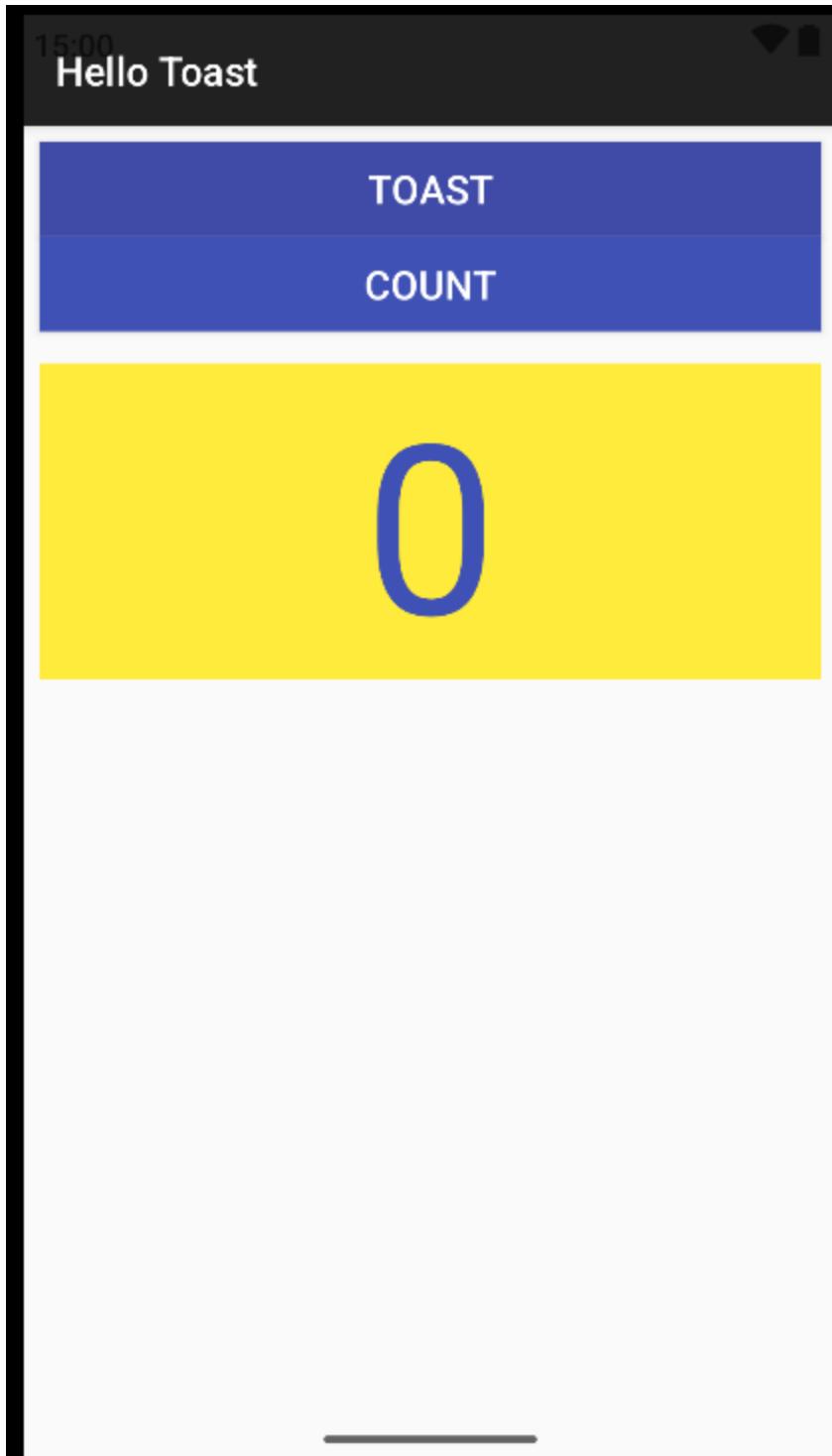
7. Tìm phần tử **TextView** có id là **show_count**, và thay đổi các thuộc tính sau.

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

8. Xóa các thuộc tính sau khỏi phần tử **show_count**.

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button_count"
```

9. Nhấp vào tab **Preview** ở phía bên phải cửa sổ Android Studio (nếu chưa được chọn) để xem trước bố cục hiện tại.



Thay đổi vị trí của các phần tử trong LinearLayout

LinearLayout sắp xếp các phần tử của nó theo một 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 đã hiển thị trong hình trước đó.

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

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp bố cục **activity_main.xml** (nếu chưa mở), sau đó nhấp vào tab **Text**.
3. Chọn nút **button_count** và tất cả các thuộc tính của nó, từ thẻ `<Button` đến hết dấu đóng `>`, sau đó chọn **Edit > Cut (Cắt)**.
4. Nhấp chuột sau thẻ đóng `>` của phần tử **TextView**, nhưng trước thẻ đóng `</LinearLayout>`, sau đó chọn **Edit > Paste (Dán)**.
5. (Tùy chọn) Để chỉnh sửa lại khoảng cách hoặc thụt dòng cho đẹp mắt, chọn **Code > Reformat Code** để định dạng lại mã XML với khoảng cách và thụt dòng phù hợp.

Bây giờ, mã XML cho các phần tử giao diện sẽ trông giống như sau:

```
<?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 for showing toast -->

    <!-- TextView to display the count -->

    <!-- Button for incrementing the count -->

    <Button
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3F4BA7"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
```

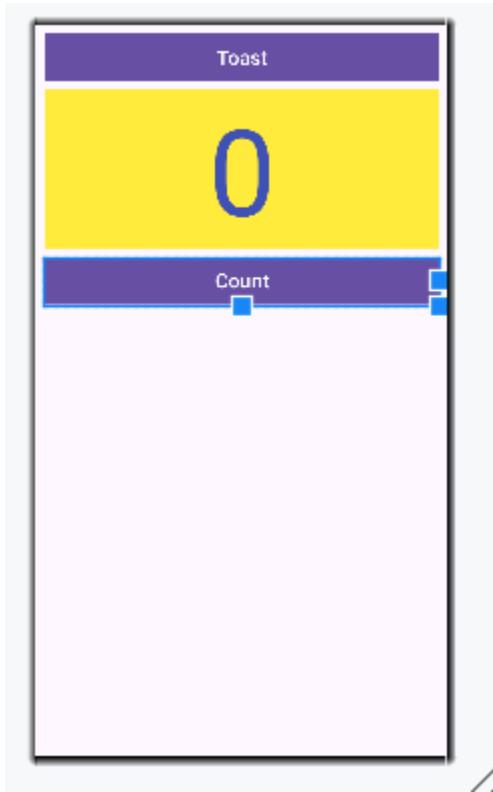
```
    android:layout_marginEnd="8dp"
    android:text="@string/button_label_toast"
    android:textColor="@color/white"
    android:textSize="20sp" />

<TextView
    android:id="@+id/text_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#FFEB3B"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAllCaps="true"
    android:textColor="#3F51B5"
    android:textSize="120dp" />

<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#3F51B5"
    android:text="@string/button_label_count"
    android:layout_marginStart="8dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:textColor="@color/white"
    android:textSize="20dp" />

</LinearLayout>
```

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



Thêm thuộc tính weight cho phần tử TextView

Việc xác định các thuộc tính **gravity** và **weight** giúp bạn kiểm soát tốt hơn cách sắp xếp các **View** và nội dung trong **LinearLayout**.

- Thuộc tính `android:gravity` xác định cách căn chỉnh nội dung bên trong một **View**. Ở bài trước, bạn đã đặt thuộc tính này cho **show_count** (**TextView**) để căn giữa nội dung (chữ số **0**) trong khung **TextView**.

```
    android:gravity="center"
```

- Thuộc tính `android:layout_weight` xác định lượng không gian thừa trong **LinearLayout** mà View đó sẽ chiếm. Nếu chỉ có một View có thuộc tính này, nó sẽ chiếm toàn bộ không gian trống. Nếu có nhiều View có **weight**, không gian sẽ được chia theo tỷ lệ.
 - Ví dụ: Nếu mỗi nút **Button** có `weight="1"` và **TextView** có `weight="2"`, tổng cộng là 4, thì mỗi nút **Button** sẽ nhận $\frac{1}{4}$ không gian, còn **TextView** sẽ nhận $\frac{1}{2}$ không gian.

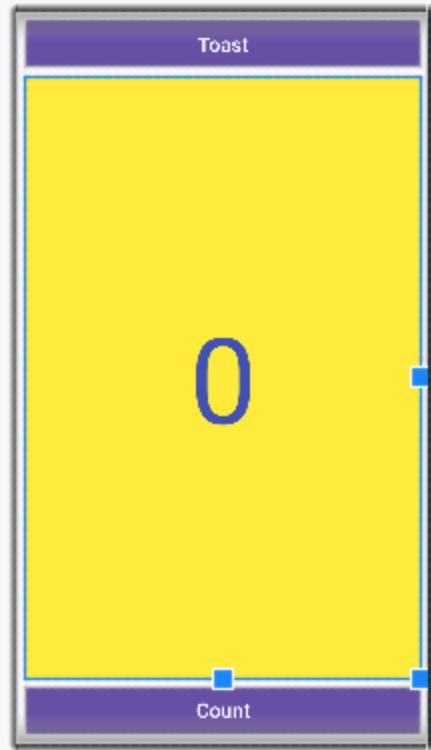
Trên các thiết bị khác nhau, **show_count** (**TextView**) có thể chiếm một phần hoặc phần lớn không gian giữa hai nút **Toast** và **Count**. Để đảm bảo **TextView** mở rộng và lấp đầy không gian trống trên mọi thiết bị, hãy thêm thuộc tính `android:layout_weight`.

Các bước thực hiện

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp bố cục **activity_main.xml** (nếu chưa mở), sau đó nhập vào tab **Text**.
3. Tìm phần tử **show_count** (**TextView**) và thêm thuộc tính sau:

```
    android:layout_weight="1"
```

Bản xem trước bây giờ trông như hình sau.



Bây giờ, phần tử **show_count** (**TextView**) sẽ chiếm toàn bộ không gian giữa các nút bấm. Bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ của cửa sổ xem trước và chọn một thiết bị khác.

Dù chọn thiết bị nào, **show_count** (**TextView**) vẫn sẽ lấp đầy không gian giữa các nút.

Mã giải pháp cho **Task 2**

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 for showing toast -->

<!-- TextView to display the count -->

<!-- Button for incrementing the count -->

<Button
    android:id="@+id/button_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#3F4BA7"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="@string/button_label_toast"
    android:textColor="@color/white"
    android:textSize="20sp" />

<TextView
    android:id="@+id/text_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:background="#FFEB3B"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAllCaps="true"
    android:textColor="#3F51B5"
    android:textSize="120dp" />

<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#3F51B5"
```

```
    android:text="@string/button_label_count"
    android:layout_marginStart="8dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:textColor="@color/white"
    android:textSize="20dp" />

</LinearLayout>
```

Thay đổi bố cục sang RelativeLayout

RelativeLayout là một nhóm **View** trong đó mỗi **View** được định vị và căn chỉnh dựa trên các **View** khác trong cùng nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục bằng **RelativeLayout**.

Thay đổi LinearLayout thành RelativeLayout

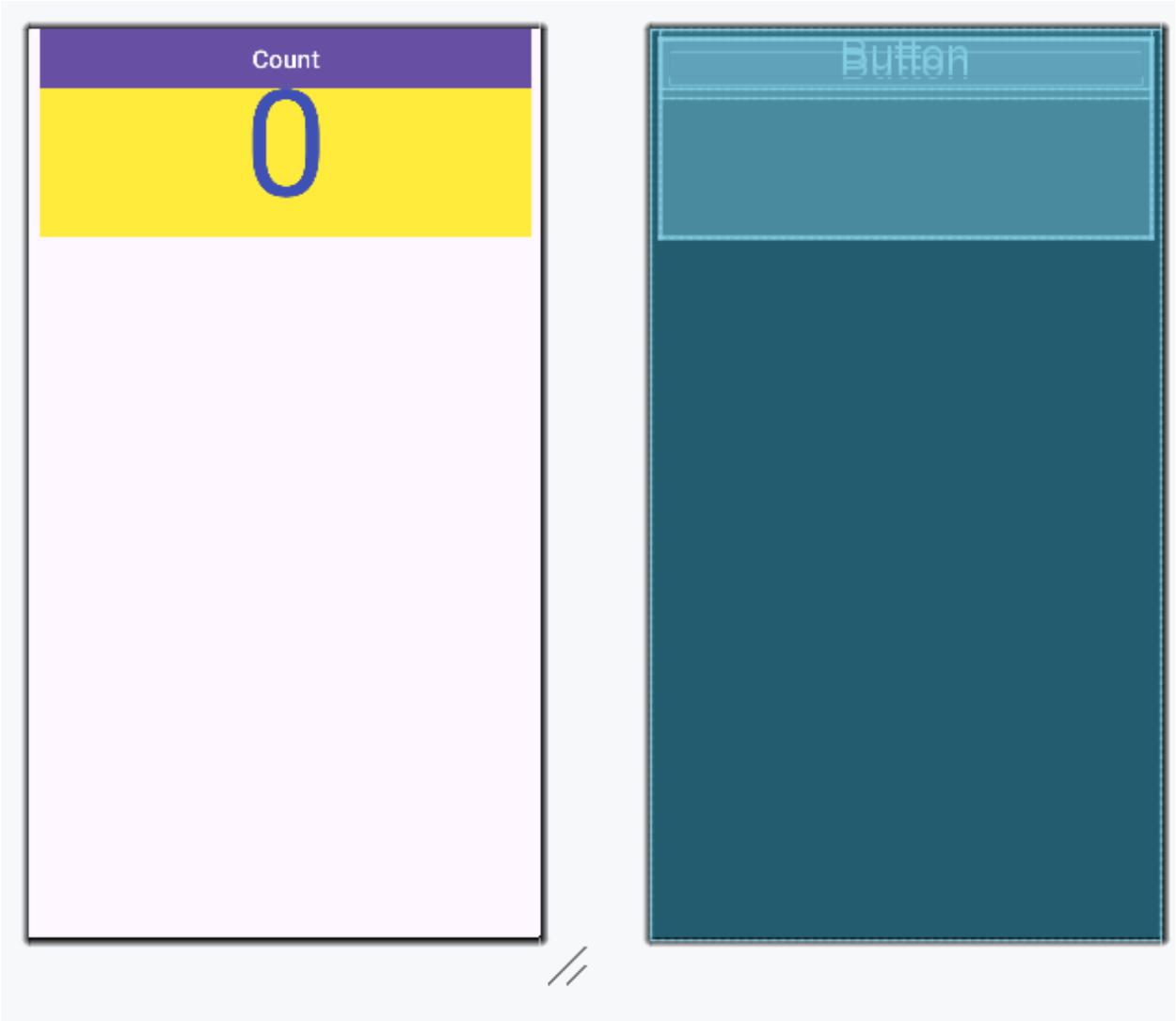
Một cách dễ dàng để thay đổi **LinearLayout** thành **RelativeLayout** là chỉnh sửa trực tiếp trong tab **Text** của tệp XML.

1. Mở tệp **activity_main.xml**, sau đó nhấp vào tab **Text** ở cuối trình chỉnh sửa để xem mã XML.
2. Thay đổi `<LinearLayout` ở đầu tệp thành `<RelativeLayout`, để câu lệnh trông như sau:

- ```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android">
```
3. Cuộn xuống để đảm bảo thẻ kết thúc `</LinearLayout>` cũng được thay đổi thành `</RelativeLayout>`; nếu chưa, hãy chỉnh sửa thủ công.
  4. **Sắp xếp lại các View trong RelativeLayout**

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

1. Nhấp vào tab **Preview** bên cạnh trình chỉnh sửa (nếu chưa được chọn) để xem bản xem trước bố cục, bây giờ trông giống như hình bên dưới.



**Lưu ý:** Khi thay đổi sang **RelativeLayout**, trình chỉnh sửa bố cục cũng thay đổi một số thuộc tính của View. Ví dụ:

- Nút **Count** (button\_count) **đè lên** nút **Toast** (button\_toast), khiến bạn không nhìn thấy nút **Toast**.
  - Phần trên của **TextView** (show\_count) **đè lên** các nút **Button**.
2. Thêm thuộc tính `android:layout_below` vào nút **button\_count** để đặt nút này ngay bên dưới **TextView** (show\_count). Đây là một trong nhiều thuộc tính giúp định vị View trong **RelativeLayout**, cho phép bạn đặt View dựa trên View khác.

```
 android:layout_below="@+id/show_count"
```

3. Thêm thuộc tính android:layout\_centerHorizontal vào **button\_count** để căn giữa nút này theo chiều ngang trong **RelativeLayout** (là nhóm cha của View này).

```
 android:layout_centerHorizontal="true".
```

→ Mã XML đầy đủ cho nút **button\_count** như sau:

```
<Button
 android:id="@+id/button_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:background="#3F51B5"
 android:text="Count"
 android:layout_below="@+id/show_count"
 android:layout_marginStart="8dp"
 android:layout_marginBottom="8dp"
 android:layout_marginEnd="8dp"
 android:textColor="@color/white"
 android:textSize="20dp"
 android:layout_centerHorizontal="true"/>
```

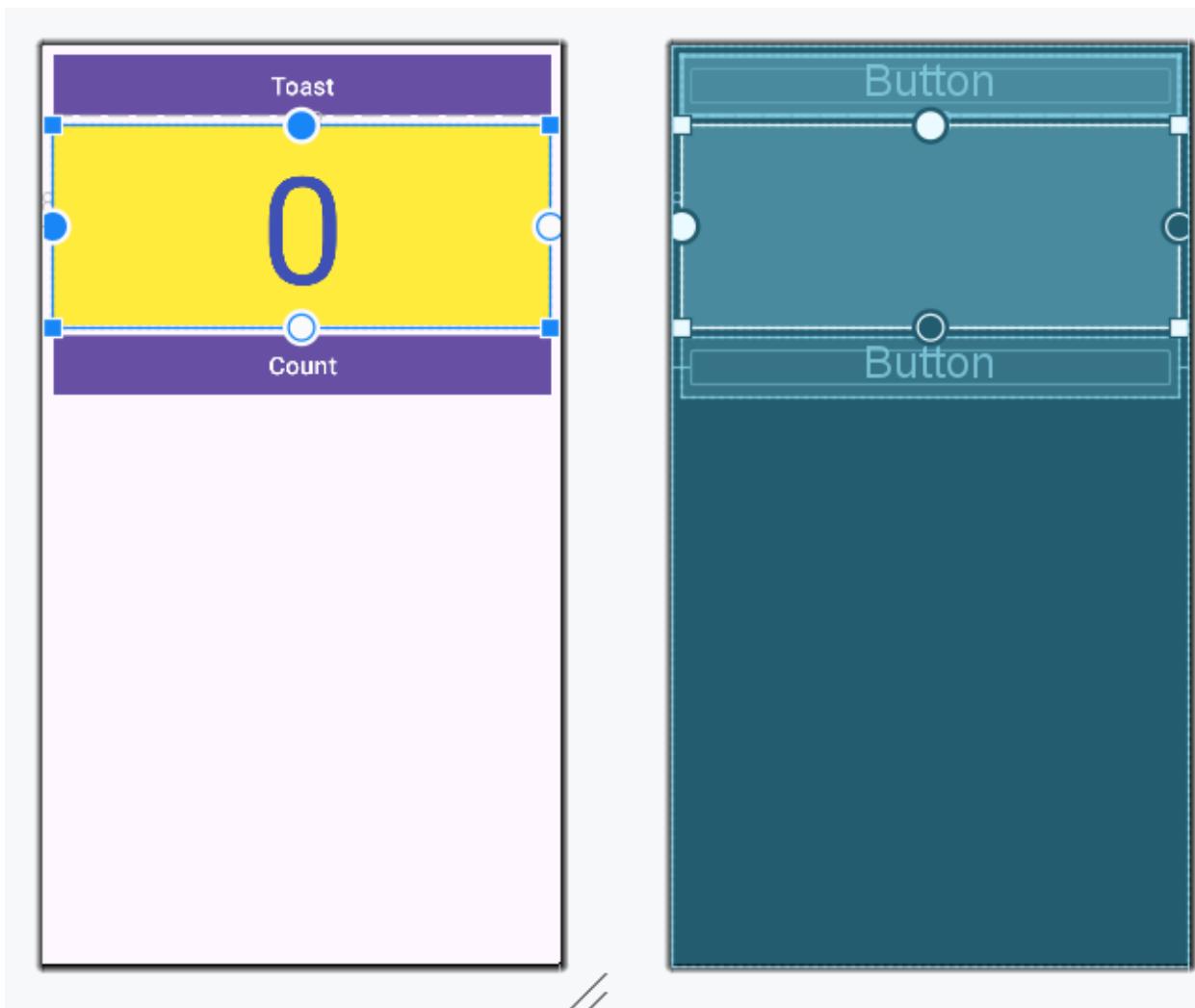
4. Thêm các thuộc tính sau vào **show\_count** (TextView):

```
 android:layout_below="@+id/button_toast"
 android:layout_alignParentLeft="true"
 android:layout_alignParentStart="true".
```

- android:layout\_alignParentLeft → Căn View về phía **trái** của **RelativeLayout** (nhóm cha).
- android:layout\_alignParentStart → Căn View theo **cạnh bắt đầu** của nhóm cha.
  - Thuộc tính này giúp hỗ trợ các thiết bị sử dụng ngôn ngữ **phải sang trái (RTL)**.
  - Nếu ứng dụng chạy trên thiết bị có ngôn ngữ **trái sang phải (LTR)**, nó sẽ căn về bên **trái**.

- Nếu chạy trên thiết bị có ngôn ngữ **phải sang trái (RTL)**, nó sẽ căn về bên **phải**.
5. **Xóa thuộc tính** `android:layout_weight="1"` của **show\_count** (`TextView`), vì thuộc tính này không có tác dụng trong **RelativeLayout**.

Bản xem trước bố cục bây giờ trông giống như hình bên dưới.



Mẹo:

**RelativeLayout** giúp bạn dễ dàng sắp xếp nhanh chóng các phần tử UI trong bố cục. Để tìm hiểu thêm về cách định vị View trong **RelativeLayout**, hãy tham khảo tài liệu "**Positioning Views**" trong chủ đề "**RelativeLayout**" của **API Guide**.

Mã giải pháp cho Task 3

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 for showing toast -->

 <!-- TextView to display the count -->

 <!-- Button for incrementing the count -->

 <Button
 android:id="@+id/button_toast"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:background="#3F4BA7"
 android:layout_marginStart="8dp"
 android:layout_marginTop="8dp"
 android:layout_marginEnd="8dp"
 android:text="@string/button_label_toast"
 android:textColor="@color/white"
 android:textSize="20sp" />

 <TextView
 android:id="@+id/show_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:background="#FFEB3B"
 android:layout_marginStart="8dp"
 android:layout_marginTop="8dp"
 android:layout_marginEnd="8dp"
```

```

 android:layout_marginBottom="8dp"
 android:gravity="center"
 android:text="@string/count_initial_value"
 android:layout_below="@+id/button_toast"
 android:layout_alignParentLeft="true"
 android:layout_alignParentStart="true"
 android:textAllCaps="true"
 android:textColor="#3F51B5"
 android:textSize="120dp" />

<Button
 android:id="@+id/button_count"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:background="#3F51B5"
 android:text="@string/button_label_count"
 android:layout_below="@+id/show_count"
 android:layout_marginStart="8dp"
 android:layout_marginBottom="8dp"
 android:layout_marginEnd="8dp"
 android:textColor="@color/white"
 android:textSize="20dp"
 android:layout_centerHorizontal="true"/>

</RelativeLayout>

```

Tóm tắt

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

- Để xem trước bô cục ứng dụng theo hướng **ngang**, nhấp vào nút **Orientation in Editor** trên thanh công cụ và chọn **Switch to Landscape**.  
→ Chọn **Switch to Portrait** để quay lại hướng dọc.
- Để tạo một biến thể bô cục khác cho **hướng ngang**, nhấp vào nút **Orientation in Editor** và chọn **Create Landscape Variation**.  
→ Một cửa sổ chỉnh sửa mới sẽ mở với tab **land/activity\_main.xml**, hiển thị bô cục theo hướng ngang.

- Để xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thật hoặc trình giả lập, nhấp vào nút **Device in Editor** trên thanh công cụ và chọn một thiết bị.
- Để tạo một biến thể bố cục khác dành cho **máy tính bảng (màn hình lớn hơn)**, nhấp vào nút **Orientation in Editor** và chọn **Create layout x-large Variation**.  
→ Một cửa sổ chỉnh sửa mới sẽ mở với tab **xlarge/activity\_main.xml**, hiển thị bố cục cho thiết bị có màn hình lớn.

## Sử dụng ConstraintLayout

- Để xóa tất cả ràng buộc trong **ConstraintLayout**, nhấp vào nút **Clear All Constraints** trên thanh công cụ.
- Bạn có thể căn chỉnh một phần tử giao diện người dùng chứa văn bản (chẳng hạn như **TextView** hoặc **Button**) với một phần tử khác bằng **ràng buộc đường cơ sở (baseline constraint)**.
- Để tạo ràng buộc đường cơ sở, di chuột qua phần tử giao diện người dùng cho đến khi nút **baseline constraint** xuất hiện bên dưới phần tử.
- Nút **pack** trên thanh công cụ cung cấp tùy chọn để sắp xếp hoặc mở rộng các phần tử UI đã chọn.  
→ Bạn có thể sử dụng nó để **căn đều các nút (Button) theo chiều ngang trong bố cục**.

## Sử dụng LinearLayout

- LinearLayout** là một **ViewGroup** sắp xếp các **View** theo hàng **ngang** hoặc **dọc**.
- Một **LinearLayout** cần có các thuộc tính sau:
  - layout\_width
  - layout\_height
  - orientation
- match\_parent** cho layout\_width hoặc layout\_height:
  - Mở rộng **View** để lấp đầy phần tử cha.
  - Nếu **LinearLayout** là phần tử gốc, nó sẽ mở rộng theo kích thước màn hình.
- wrap\_content** cho layout\_width hoặc layout\_height:
  - Thu nhỏ **View** để vừa với nội dung của nó.
  - Nếu không có nội dung, **View** sẽ trở nên vô hình.

- **Số dp (density-independent pixels) cố định** cho layout\_width hoặc layout\_height:
  - Xác định kích thước cố định, được điều chỉnh theo mật độ màn hình.
  - Ví dụ: 16dp có nghĩa là 16 pixel độc lập với mật độ.
- **Hướng (orientation) của LinearLayout:**
  - horizontal → Sắp xếp phần tử **trái sang phải**.
  - vertical → Sắp xếp phần tử **trên xuống dưới**.
- **Sử dụng gravity và weight để kiểm soát bố cục:**
  - **android:gravity** → Căn chỉnh nội dung bên trong **View**.
  - **android:layout\_weight** → Xác định tỷ lệ không gian bổ sung được phân bổ cho **View**.
    - Nếu chỉ có một **View** có thuộc tính này, nó sẽ nhận toàn bộ không gian trống.
    - Nếu có nhiều **View**, không gian sẽ được chia theo tỷ lệ.
    - Ví dụ: Nếu hai **Button** có **weight="1"** và một **TextView** có **weight="2"** (tổng cộng 4), mỗi **Button** sẽ nhận  $\frac{1}{4}$  không gian và **TextView** sẽ nhận  $\frac{1}{2}$ .

## Sử dụng RelativeLayout

- **RelativeLayout** là một **ViewGroup**, trong đó mỗi **View** được căn chỉnh tương đối với các **View** khác trong cùng nhóm.
- Các thuộc tính quan trọng:
  - **android:layout\_alignParentTop** → Căn **View** lên **đỉnh** của phần tử cha.
  - **android:layout\_alignParentLeft** → Căn **View** về **bên trái** của phần tử cha.
  - **android:layout\_alignParentStart** → Căn **View** theo **cạnh bắt đầu** của phần tử cha.
    - **Start** là **bên trái** nếu ngôn ngữ từ **trái sang phải (LTR)**.
    - **Start** là **bên phải** nếu ngôn ngữ từ **phải sang trái (RTL)**.

Tài liệu liên quan

**Tài liệu về các khái niệm liên quan có trong:**

**1.2: Bố cục và tài nguyên cho UI (Layouts and resources for the UI).**

**Tìm hiểu thêm:**

**Tài liệu Android Studio:**

- Giới thiệu về Android Studio
- Tạo biểu tượng ứng dụng với Image Asset Studio

**Tài liệu Android Developer:**

- Layouts
- Xây dựng UI với Layout Editor
- Xây dựng giao diện đáp ứng với ConstraintLayout
- LinearLayout
- RelativeLayout
- View
- Button
- TextView
- Hỗ trợ nhiều mật độ pixel khác nhau

**Khác:**

- Codelabs: **Sử dụng ConstraintLayout để thiết kế giao diện Android**
- **Từ vựng và khái niệm quan trọng**

Bài tập về nhà

Thay đổi một ứng dụng

Mở ứng dụng **HelloToast**.

1. **Đổi tên** dự án thành **HelloConstraint**, sau đó **refactor** toàn bộ dự án thành **HelloConstraint**.  
→ (Hướng dẫn về cách sao chép và refactor một dự án có trong **Phụ lục: Tiện ích**.)
2. **Chỉnh sửa bố cục** trong **activity\_main.xml** để căn chỉnh các nút **Toast** và **Count** **dọc theo bên trái** của **TextView show\_count** (hiển thị số "0").  
→ Xem hình minh họa về bố cục.
3. **Thêm một nút thứ ba** có tên **Zero**, xuất hiện **giữa** các nút **Toast** và **Count**.
4. **Sắp xếp** các nút theo **chiều dọc**, phân bổ khoảng cách đều từ trên xuống dưới của **TextView show\_count**.

5. **Đặt nền ban đầu của nút Zero thành màu xám.**
6. **Đảm bảo rằng nút Zero xuất hiện trong các bố cục khác**, bao gồm:
  - **activity\_main.xml (land)** (giao diện ngang).
  - **activity\_main.xml (xlarge)** (giao diện trên màn hình máy tính bảng).
7. **Cập nhật xử lý sự kiện của nút Zero** để khi nhấn vào, nó sẽ đặt giá trị trong **show\_count** về 0.
8. **Cập nhật xử lý sự kiện của nút Count:**
  - Khi nhấn, nút Count sẽ đổi màu nền tùy thuộc vào số hiện tại là **chẵn hay lẻ**.
  - **Gợi ý:** Không sử dụng **findViewById** để tìm nút Count. Hãy tìm cách khác!
  - Có thể sử dụng các hằng số trong lớp **Color** để đặt màu nền.
9. **Cập nhật xử lý sự kiện của nút Count** để thay đổi màu nền của nút Zero thành một màu khác ngoài xám, nhằm thể hiện rằng nút Zero đã được kích hoạt.
  - **Gợi ý:** Lần này, có thể sử dụng **findViewById** để tìm nút Zero.
10. **Cập nhật xử lý sự kiện của nút Zero** để **đặt lại màu nền của chính nó về xám** khi số đếm bằng 0.

Trả lời các câu hỏi

Câu hỏi 1:

Hai thuộc tính **ràng buộc bố cục (layout constraint)** nào trên nút Zero giúp nó **được căn giữa theo chiều dọc**, nằm cách đều hai nút còn lại? (Chọn 2 đáp án)  
**app:layout\_constraintBottom\_toTopOf="@+id/button\_count"**  
**app:layout\_constraintTop\_toBottomOf="@+id/button\_toast"**

Câu hỏi 2:

Thuộc tính ràng buộc bố cục nào trên **nút Zero** giúp nó **căn chỉnh ngang hàng** với hai nút còn lại?

**app:layout\_constraintLeft\_toLeftOf="parent"**

Câu hỏi 3:

Đâu là chữ ký (signature) đúng của một phương thức được dùng với thuộc tính **android:onClick** trong XML?

**public void callMethod(View view)**

Câu hỏi 4:

Trong trình xử lý sự kiện của nút **Count**, phương pháp nào **hiệu quả hơn** để thay đổi màu nền của nút?

**Sử dụng tham số view được truyền vào trình xử lý sự kiện, với `setBackgroundColor()`:**

### Bài hc 1.3: Văn bản và chế độ xem cuộn

#### Giới thiệu

Lớp **TextView** là một lớp con của lớp **View**, dùng để hiển thị văn bản trên màn hình. Bạn có thể kiểm soát cách văn bản xuất hiện bằng cách sử dụng 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 của nó theo chiều dọc.

Nếu có nhiều thông tin hơn so với khả năng hiển thị của 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 cuộn ngang bằng cách vuốt sang phải hoặc trái.

Thông thường, bạn sẽ sử dụng chế độ xem cuộn cho các bài báo, tin tức hoặc bất kỳ đoạn văn bản dài nào không thể hiển thị hoàn toàn 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 thành phần giao diện người dùng (chẳng hạn như trường nhập 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**. Hãy chỉ đặt **một** phần tử con bên trong nó—phần tử con này chứa toàn bộ nội dung cần cuộn. Phần tử con này có thể là một **ViewGroup** (chẳng hạn như **LinearLayout**) chứa các thành phần 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, hiển thị các mục mà người dùng có thể cuộn qua (chẳng hạn như các phần tử **TextView**).

Với **ScrollView**, tất cả các thành phần giao diện người dùng đều được tải vào bộ nhớ và có trong cây cấu trúc xem ngay cả khi chúng không hiển thị trên màn

hình. Điều này làm cho **ScrollView** lý tưởng để cuộn qua các trang văn bản tự do một cách mượt mà, vì văn bản đã được tải sẵn vào bộ nhớ. Tuy nhiên, **ScrollView** có thể tiêu tốn nhiều bộ nhớ, ảnh hưởng đến hiệu suất của phần còn lại 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 bài học khác.

## Những gì bạn cần biết trước

Bạn cần có khả năng:

- Tạo ứng dụng Hello World bằng Android Studio.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị.
- Triển khai **TextView** trong bố cục của ứng dụng.
- Tạo và sử dụng tài nguyên chuỗi văn bản.

## Những gì bạn sẽ học

- Cách sử dụng mã XML để thêm nhiều phần tử **TextView**.
- Cách sử dụng mã XML để định nghĩa một **View** cuộn.
- Cách hiển thị văn bản tự do với một số thẻ định dạng HTML.
- Cách tạo kiểu cho màu 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 **ConstraintLayout** **ViewGroup** thành **RelativeLayout**.
- Thêm hai phần tử **TextView** cho tiêu đề bài viết và tiêu đề phụ.
- Sử dụng kiểu dáng và màu sắc **TextAppearance** cho tiêu đề bài viết và tiêu đề phụ.

### Sử dụng thẻ HTML trong chuỗi văn bản để kiểm soát định dạng:

- Sử dụng thuộc tính **lineSpacingExtra** để thêm khoảng cách giữa các dòng, cải thiện khả năng đọc.
- Thêm **ScrollView** vào bố cục để cho phép cuộn qua phần tử **TextView**.

- Sử dụng thuộc tính **autoLink** để các URL trong văn bản trở nên hoạt động và có thể nhấp vào.

## Tổng quan về ứng dụng

Ứng dụng **Scrolling Text** minh họa cách sử dụng thành phần giao diện người dùng **ScrollView**. **ScrollView** là một **ViewGroup**, trong ví dụ này, chứa một **TextView**. Nó hiển thị một trang văn bản dài—trong trường hợp này là một bài đánh giá album nhạc—mà người dùng có thể cuộn theo chiều dọc để đọc bằng cách vuốt lên hoặc xuống. Thanh cuộn xuất hiện ở lề bên phải. Ứng dụng này cho thấy cách bạn có thể sử dụng văn bản được định dạng với các thẻ HTML tối thiểu để thiết lập văn bản in đậm hoặc in nghiêng, và sử dụng ký tự xuống dòng để ngăn cách các đoạn văn. Bạn cũng có thể thêm các liên kết web hoạt động trong văn bản.

Trong hình trên, các yếu tố sau xuất hiện:

1. Một liên kết web hoạt động được nhúng trong văn bản tự do.
2. Thanh cuộn xuất hiện khi cuộn văn bản.

## 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 để hiển thị tiêu đề và tiêu đề phụ của bài viết, đồng thời thay đổi phần tử **TextView** "Hello World" hiện tại để hiển thị một bài viết dài. Hình minh họa bên dưới là sơ đồ của bố cục.

Bạn sẽ thực hiện tất cả các thay đổi này trong mã XML và tệp strings.xml. Bạn sẽ chỉnh sửa mã XML cho bố cục trong tab **Text**, mà bạn có thể hiển thị bằng cách nhấp vào tab **Text**, thay vì nhấp vào tab **Design** để mở giao diện thiết kế. Một số thay đổi đối với các phần tử giao diện người dùng (UI) và thuộc tính sẽ dễ dàng thực hiện hơn trực tiếp trong tab **Text** bằng cách sử dụng mã nguồn XML.

### 1.1 Tạo dự án và các phần tử TextView

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

**Mẹo:** Để tìm hiểu thêm về các thuộc tính này, xem tài liệu tham khảo về **TextView**.

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

Thuộc tính	Giá trị
<b>Application Name</b>	Scrolling Text
<b>Company Name</b>	android.example.com (hoặc tên miền của bạn)
<b>Phone and Tablet Minimum SDK</b>	API15: Android 4.0.3 IceCreamSandwich
<b>Template</b>	Empty Activity
<b>Generate Layout File checkbox</b>	Selected
<b>Backwards Compatibility (AppCompat) checkbox</b>	Selected

- Trong thư mục **app > res > layout** ở ngăn **Project > Android**, mở tệp **activity\_main.xml** và nhấp vào tab **Text** để xem mã XML.  
Ở phần trên cùng, hoặc gốc, của cấu trúc phân cấp View là **ViewGroup ConstraintLayout**:  
`<androidx.constraintlayout.widget.ConstraintLayout>`
- Thay đổi **ViewGroup** này thành **RelativeLayout**. Dòng mã thứ hai bây giờ sẽ trông giống như sau:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

**RelativeLayout** cho phép bạn đặt các phần tử giao diện người dùng (UI) tương đối với nhau hoặc tương đối với chính **RelativeLayout** cha.

Phần tử **TextView** "Hello World" mặc định được tạo bởi mẫu Bố cục Trống (Empty Layout) vẫn có các thuộc tính ràng buộc (chẳng hạn như `app:layout_constraintBottom_toBottomOf="parent"`). Đừng lo lắng—you sẽ xóa chúng trong bước tiếp theo.

- Xóa dòng mã XML sau đây, dòng này liên quan đến **ConstraintLayout**:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

Khôi mã XML ở phần đầu bây giờ trông như thế này:

```
<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">
```

5. 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, là những thuộc tính bắt buộc đối với TextView.
6. 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 #1 attribute	Value
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:id	"@+id/article_heading"
android:background	"@color/colorPrimary"
android:textColor	"@android:color/white"
android:padding	"10dp"
android:textAppearance	"@android:style/TextAppearance.DeviceDefault.Large"
android:textStyle	"bold"
android:text	"Article Title"

7. Trích xuất tài nguyên chuỗi cho thuộc tính android:text với chuỗi cứng "Article Title" trong TextView để tạo một mục trong **strings.xml**.

Đặt con trỏ vào chuỗi cứng, nhấn Alt-Enter (hoặc Option-Enter trên Mac), chọn **Extract string resource**, đảm bảo tùy chọn **Create the resource in directories** được chọn, sau đó chỉnh sửa tên tài nguyên thành **article\_title**.

Tài nguyên chuỗi được mô tả chi tiết trong mục **String Resources**.

8. Trích xuất tài nguyên kích thước cho thuộc tính android:padding với chuỗi cứng "10dp" trong TextView để tạo tệp dimens.xml và thêm một mục vào đó.

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

9. Thêm một phần tử TextView khác phía trên TextView "Hello World" và bên dưới TextView bạn đã tạo trong các bước trước đó. Thêm các thuộc tính sau cho TextView.

TextView #2 Attribute	Value
layout_width	"match_parent"

layout_height	"wrap_content"
android:id	"@+id/article_subheading"
android:layout_below	"@+id/article_heading"
android:padding	"@dimen/padding_regular"
android:textAppearance	"@android:style/TextAppearance.DeviceDefault"
android:text	"Article Subtitle"

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

10. Trích xuất tài nguyên chuỗi cho chuỗi cứng "Article Subtitle" của thuộc tính android:text trong TextView thành article\_subtitle.

11. Trong phần tử TextView "Hello World", xóa các thuộc tính layout\_constraint:

```
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"
```

12. Thêm các thuộc tính TextView sau vào phần tử "Hello World" TextView và thay đổi thuộc tính android:text:

<b>TextView Attribute</b>	<b>Value</b>
android:id	"@+id/article"
android:layout_below	"@+id/article_subheading"
android:lineSpacingExtra	"5sp"
android:padding	"@dimen/padding_regular"
android:text	Change to "Article text"

13. Trích xuất tài nguyên chuỗi cho "Article text" thành article\_text và trích xuất tài nguyên kích thước cho "5sp" thành line\_spacing.

14. Định dạng lại và căn chỉnh mã bằng cách chọn **Code > Reformat Code**.

Đây là một thực hành tốt để định dạng và căn chỉnh mã của bạn sao cho dễ hiểu hơn đối với bạn và người khác.

## 1.2 Thêm nội dung bài viết

Trong một ứng dụng thực tế truy cập các bài báo từ tạp chí hoặc báo, các bài viết hiển thị có thể sẽ đến từ một nguồn trực tuyến thông qua nhà cung cấp nội dung hoặc có thể được lưu sẵn trong cơ sở dữ liệu trên thiết bị.

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

1. Trong thư mục **app > res > values**, mở **strings.xml**.
2. Mở bất kỳ tệp văn bản nào có một lượng lớn văn bản, hoặc mở tệp **strings.xml** của ứng dụng **ScrollingText** đã hoàn thiện.
3. Nhập giá trị cho các chuỗi article\_title và article\_subtitle với tiêu đề và phụ đề tùy ý, hoặc sử dụng các giá trị trong tệp strings.xml của ứng dụng **ScrollingText** đã hoàn thiện. Đảm bảo rằng các giá trị chuỗi là văn bản trên một dòng và không có thẻ HTML hoặc nhiều dòng.
4. Nhập hoặc sao chép-dán văn bản cho chuỗi article\_text.

Bạn có thể sử dụng văn bản trong tệp văn bản của bạn, hoặc sử dụng văn bản được cung cấp cho chuỗi article\_text trong tệp strings.xml của ứng dụng **ScrollingText** đã hoàn thiện. Yêu cầu duy nhất cho nhiệm vụ này là văn bản phải đủ dài để không hiển thị hết trên màn hình.

Lưu ý các điểm sau (tham khảo hình minh họa bên dưới để có ví dụ):

- Khi bạn nhập hoặc dán văn bản vào tệp strings.xml, các dòng văn bản sẽ không tự động xuống dòng mà sẽ kéo dài vượt ra ngoài lề phải. Đây là hành vi đúng—mỗi dòng văn bản mới bắt đầu từ lề trái sẽ đại diện cho toàn bộ một đoạn văn. Nếu bạn muốn văn bản trong strings.xml được xuống dòng, bạn có thể nhấn **Return** để thêm dấu kết thúc dòng cứng, hoặc định dạng văn bản trước trong một trình soạn thảo văn bản với các dấu kết thúc dòng cứng.

- Nhập \n để đại diện cho kết thúc một dòng và thêm một \n khác để đại diện cho một dòng trống. Bạn cần thêm ký tự kết thúc dòng để các đoạn văn không nối liền với nhau.
- Nếu bạn có dấu nháy đơn (') trong văn bản, bạn phải thoát nó bằng cách thêm một dấu gạch chéo ngược (\') phía trước. Nếu bạn có dấu nháy kép trong văn bản, bạn cũng phải thoát nó ("). Bạn cũng phải thoát bất kỳ ký tự không thuộc ASCII nào khác. Xem phần **Định dạng và phong cách** của Tài nguyên chuỗi để biết thêm chi tiết.
- Thêm thẻ HTML <b> và </b> xung quanh các từ cần in đậm.
- Thêm thẻ HTML <i> và </i> xung quanh các từ cần in nghiêng. Nếu bạn sử dụng dấu nháy đơn cong trong một cụm từ in nghiêng, hãy thay chúng bằng dấu nháy đơn thẳng.
- Bạn có thể kết hợp in đậm và in nghiêng bằng cách kết hợp các thẻ, như trong <b><i>... từ ...</i></b>.
- Các thẻ HTML khác sẽ bị bỏ qua.
- Bao toàn bộ 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 dưới đây sử dụng [www.rockument.com](http://www.rockument.com).) Đừng sử dụng thẻ HTML, vì bất kỳ thẻ HTML nào ngoài thẻ in đậm và in nghiêng sẽ bị bỏ qua và hiển thị dưới dạng văn bản, điều này không phải là điều bạn muốn.

```

<resources>
 <string name="app_name">Scrolling Text</string>
 <string name="article_title">Beatles Anthology Vol. 1</string>
 <string name="article_subtitle">Behind That Locked Door: Beatles Rarities!</string>
 <string name="article_text">
 <![CDATA[
 In a vault deep inside Abbey Road Studios in London – protected by an unmarked, triple-locked door – lies one of the rarest collections of Be
 For more information, see The Beatles Time Capsule at www.rockument.com.

 This volume starts with the first new Beatle song, "Free as a Bird"; (based on a John Lennon demo, found only on <i>The Lost Tapes</i>)
 Highlights include:

 • Cry for a Shadow: – Many a Beatle fanatic started down the outtake road, like I did, with a first listen to this track.
 • My Bonnie: and #39;t She Sweet: – At the same session, the Beatles played "My Bonnie"; (the first single
 • Searching: – A Jerry Leiber – Mike Stoller comedy song that was a hit for the Coasters in 1957, and a popular part of i
 • Love Me Do: – An early version of the song, played a bit slower and with more of a blues feeling, and a cool harmonica part.<
 • She Loves You: – Till There Was You – Twist and Shout: – Live at the Princess Wales Theatre by Leicester Square.
 • Leave My Kitten Alone: – One of the lost Beatle songs recorded during the "Beatles For Sale"; sessions but never rel
 • One After 909: – A song recorded for the <i>Let It Be</i> album was actually worked on way back in the beginning.

]]>
 </string>
 <color name="colorPrimary">#6200EE</color>
</resources>

```

### 1.3 Chạy ứng dụng

Chạy ứng dụng. Bài viết xuất hiện, nhưng người dùng không thể cuộn bài viết vì bạn chưa thêm một ScrollView (việc này bạn sẽ thực hiện trong nhiệm vụ tiếp theo). Lưu ý rằng khi nhấn vào một liên kết web, hiện tại sẽ không có tác dụng gì. Bạn cũng sẽ sửa điều đó trong nhiệm vụ tiếp theo.

The screenshot shows a mobile application interface. At the top, there is a light purple header bar with three small black dots at the top left. Below it is a white section containing the text "Scrolling Text". Underneath that is a dark blue header bar with the text "Beatles Anthology Vol. 1" in white. The main content area is white and contains the following text:

Behind That Locked Door: Beatles  
Rarities!

In a vault deep inside Abbey Road  
Studios in London – protected by  
an unmarked, triple-locked door –  
lies one of the rarest collections  
of Beatles recordings.  
> For more information, see  
The Beatles Time Capsule at  
[www.rockument.com](http://www.rockument.com).  
> This volume starts with the first  
new Beatle song, "Free as  
a Bird" (based on a John  
Lennon demo, found only on  
*The Lost Tapes*).  
> **Highlights include:**  
<ul> <li> "Cry for a Shadow"  
– Many a Beatle fanatic started  
down the outtake road, like I did,

## Mã giải pháp nhiệm vụ 1

Tệp bô cục activity\_main.xml trông 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:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context="com.example.android.scrollingtext.MainActivity">

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:id="@+id/article_heading"
 android:background="@color/colorPrimary"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_title"
 android:textAppearance=
 "@android:style/TextAppearance.DeviceDefault.Large"
 android:textColor="@android:color/white"
 android:textStyle="bold" />

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

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/article"
 android:layout_below="@+id/article_subheading"
 android:lineSpacingExtra="@dimen/line_spacing"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_text" />
```

```
</RelativeLayout>
```

## Nhiệm vụ 2: Thêm một ScrollView và một liên kết web hoạt động

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 đó chưa hoạt động. Bạn sẽ thêm mã để làm cho nó hoạt động.

Ngoài ra, TextView tự nó không thể cho phép người dùng cuộn văn bản bài viết để xem toàn bộ nội dung. Bạn sẽ thêm một ViewGroup mới gọi là ScrollView vào bộ cục XML để làm cho TextView có thể cuộn được.

### 2.1 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:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/article"
 android:autoLink="web"
 android:layout_below="@+id/article_subheading"
 android:lineSpacingExtra="@dimen/line_spacing"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_text" />
```

### 2.2 Thêm ScrollView vào bộ cục

Để làm cho Chế độ xem (chẳng hạn như TextView) có thể cuộn được, hãy nhúng Chế độ xem bên trong ScrollView

1. Thêm một ScrollView giữa TextView có ID article\_subheading và TextView có ID article. Khi bạn nhập <ScrollView>, Android Studio sẽ tự động thêm </ScrollView> ở cuối, và hiển thị các gợi ý cho thuộc tính android:layout\_width và android:layout\_height.
2. Chọn **wrap\_content** từ danh sách gợi ý cho cả hai thuộc tính.  
Mã cho hai phần tử TextView và ScrollView bây giờ trông như sau:

```

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

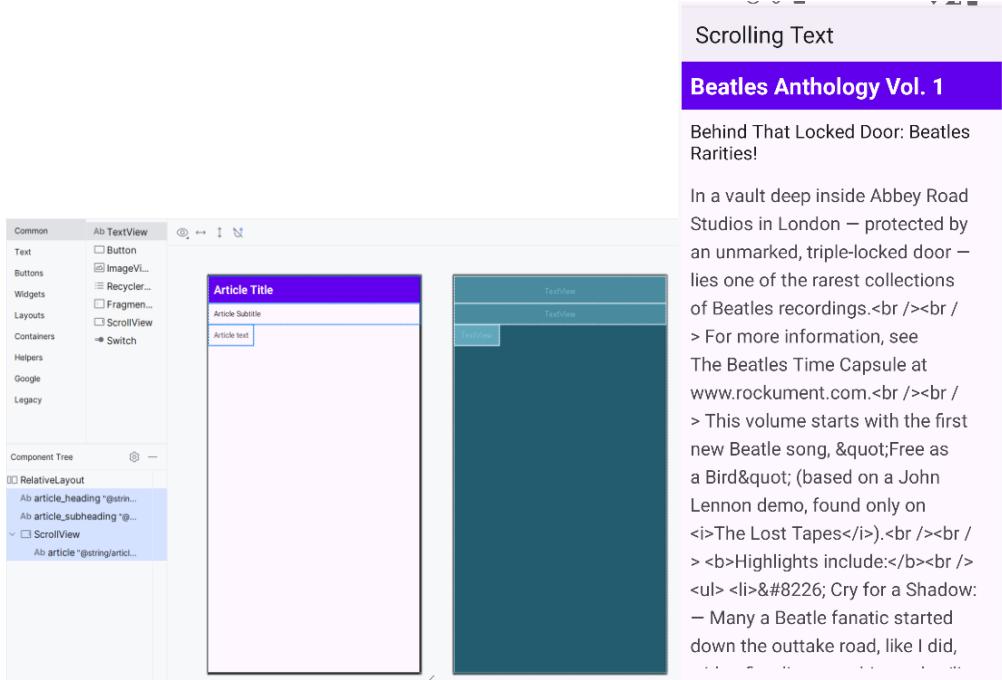
```

3. Di chuyển đoạn mã kết thúc </ScrollView> xuống sau TextView có ID article để các thuộc tính của article nằm hoàn toàn bên trong ScrollView.
4. Xóa thuộc tính sau từ TextView có ID article và thêm nó vào ScrollView: android:layout\_below="@+id/article\_subheading">

Bài viết nằm bên trong phần tử ScrollView.

5. Chọn **Code > Reformat Code** để định dạng lại mã XML, sao cho TextView của bài viết xuất hiện được thụt vào bên trong đoạn mã <ScrollView>.
6. Nhấp vào tab **Preview** ở phía bên phải của trình chỉnh sửa bô cục để xem trước bô cục.

Bô cục giờ đây trông giống như phần bên phải của hình minh họa sau:



## 2.3 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 qua bài viết. Thanh cuộn xuất hiện ở lề phải khi bạn cuộn.

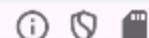
Nhấn vào liên kết web để truy cập trang web. Thuộc tính android:autoLink tự động biến bất kỳ URL nào có thể nhận dạng được trong TextView (chẳng hạn như [www.rockument.com](http://www.rockument.com)) thành một liên kết web.

- Xoay thiết bị hoặc trình giả lập khi ứng dụng đang chạy.

Quan sát cách chế độ cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.

- Chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng.

Quan sát cách chế độ cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.



## Scrolling Text

### Beatles Anthology Vol. 1

Behind That Locked Door: Beatles Rarities!

In a vault deep inside Abbey Road Studios in London – protected by an unmarked, triple-locked door – lies one of the rarest collections of Beatles recordings.<br /><br />

> For more information, see

The Beatles Time Capsule at

[www.rockument.com](http://www.rockument.com).<br /><br />

> This volume starts with the first new Beatle song, "Free as a Bird" (based on a John Lennon demo, found only on

*The Lost Tapes*).<br /><br />

> **Highlights include:**<br />

<ul> <li> "Cry for a Shadow:"

– Many a Beatle fanatic started down the outtake road, like I did,  
with a first listen to this track </li>

Trong hình trên, các yếu tố sau đây xuất hiện:

1. Một liên kết web hoạt động được nhúng trong văn bản tự do.
2. Thanh cuộn xuất hiện khi cuộn qua văn bản.

Mã giải pháp nhiệm vụ 2

Mã XML cho bộ cục với chế độ xem cuộn 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:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context="com.example.android.scrollingtext.MainActivity"
 android:id="@+id/main">

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:id="@+id/article_heading"
 android:background="@color/colorPrimary"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_title"
 android:textAppearance=
 "@android:style/TextAppearance.DeviceDefault.Large"
 android:textColor="@android:color/white"
 android:textStyle="bold" />

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

```
<ScrollView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/article_subheading">
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/article"
 android:autoLink="web"
 android:lineSpacingExtra="@dimen/line_spacing"
 android:padding="@dimen/padding_regular"
 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 (ví dụ như TextView bạn đã tạo cho bài viết). Tuy nhiên, View đó có thể là một ViewGroup khác chứa các phần tử View, chẳng hạn như LinearLayout.

Bạn có thể *lồng* một ViewGroup, chẳng hạn như LinearLayout, vào trong ScrollView, từ đó cho phép cuộn toàn bộ nội dung bên trong LinearLayout.

Ví dụ, nếu bạn muốn phần tiêu đề phụ của bài viết cùng cuộn với nội dung bài viết, bạn cần thêm một LinearLayout vào bên trong ScrollView, sau đó di chuyển tiêu đề phụ và bài viết vào trong LinearLayout.

LinearLayout sẽ trở thành phần tử con duy nhất của ScrollView, như minh họa trong hình dưới đây. Người dùng sẽ có thể cuộn toàn bộ LinearLayout, bao gồm cả tiêu đề phụ và bài viết.

#### 3.1 Thêm một LinearLayout vào ScrollView

1. Mở tệp activity\_main.xml trong dự án ứng dụng ScrollingText và chọn tab **Text** để chỉnh sửa mã XML (nếu tab này chưa được chọn).
2. Thêm một LinearLayout phía trên TextView bài viết (article) trong ScrollView. Khi bạn nhập <LinearLayout, Android Studio sẽ tự động thêm

</LinearLayout> ở cuối, đồng thời gợi ý các thuộc tính android:layout\_width và android:layout\_height. Chọn match\_parent cho chiều rộng và wrap\_content cho chiều cao từ các gợi ý. Mã ở phần đầu của ScrollView bây giờ sẽ trông như sau:

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

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

3. Di chuyển mã kết thúc </LinearLayout> xuống sau TextView bài viết (article) nhưng trước thẻ đóng </ScrollView>. LinearLayout bây giờ sẽ bao gồm TextView bài viết (article) và hoàn toàn nằm trong ScrollView.
4. Thêm thuộc tính android:orientation="vertical" vào LinearLayout để thiết lập hướng của nó thành dọc.
5. Chọn **Code > Reformat Code** để thuần hóa mã chính xác.

LinearLayout bây giờ trông như thế này:

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

 <LinearLayout
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">

 <TextView
 android:id="@+id/article"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:autoLink="web"
 android:lineSpacingExtra="@dimen/line_spacing"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_text" />
 </LinearLayout>
</ScrollView>
```

### 3.2 Di chuyển các thành phần giao diện vào trong LinearLayout

Hiện tại, LinearLayout chỉ chứa một thành phần giao diện là TextView bài viết (article). Bạn muốn đưa TextView tiêu đề phụ (article\_subheading) vào LinearLayout để cả hai có thể cuộn.

- Để di chuyển TextView tiêu đề phụ (article\_subheading), hãy chọn đoạn mã của nó, chọn **Edit > Cut**, nhấp vào phía trên TextView bài viết (article) bên trong LinearLayout, và chọn **Edit > Paste**.
- Xóa thuộc tính android:layout\_below="@+id/article\_subheading" khỏi TextView tiêu đề phụ (article\_subheading). Vì TextView này hiện đang nằm trong

LinearLayout, thuộc tính này sẽ xung đột với các thuộc tính của LinearLayout.

3. Thay đổi thuộc tính bô cục của ScrollView từ android:layout\_below="@+id/article\_subheading" thành android:layout\_below="@+id/article\_heading".  
Bây giờ, vì tiêu đề phụ là một phần của LinearLayout, ScrollView phải được đặt bên dưới tiêu đề, không phải tiêu đề phụ.

Mã XML cho ScrollView bây giờ sẽ như sau:

```
<ScrollView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_below="@+id/article_heading">

<LinearLayout
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">
 <TextView
 android:id="@+id/article_subheading"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_subtitle"
 android:textAppearance="@android:style/TextAppearance.DeviceDefault"
 />

<TextView
 android:id="@+id/article"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:autoLink="web"
 android:lineSpacingExtra="@dimen/line_spacing"
 android:padding="@dimen/padding_regular"
 android:text="@string/article_text" />
```

```
</LinearLayout>
</ScrollView>
```

#### 4. Chạy ứng dụng.

Vuốt lên và xuống để cuộn bài viết, và hãy chú ý rằng phần tiêu đề phụ giờ đây di chuyển cùng bài viết, trong khi phần tiêu đề chính vẫn cố định tại chỗ.

#### Thử thách mã hóa

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

Thử thách: Thêm một thành phần UI khác—một Nút —vào LinearLayout bên trong ScrollView để nó cuộn cùng với văn bản.tử giao diện người dùng khác—một **nút bấm (Button)**—vào **LinearLayout** bên trong **ScrollView** sao cho nút bấm này cuộn cùng với văn bản.

- Làm cho **nút bấm (Button)** xuất hiện ở phía dưới bài viết. Người dùng sẽ cuộn đến cuối bài viết để thấy **nút bấm**.
- Sử dụng dòng chữ **Thêm bình luận** làm nội dung cho **nút bấm**. Trong thử thách này, không cần tạo phương thức xử lý nút bấm; bạn chỉ cần đặt phần tử **Button** vào đúng vị trí trong bố cục.

#### Tóm tắt:

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

## Các khái niệm liên quan

Tài liệu về các khái niệm liên quan nằm trong **1.3 Text and scrolling views**.

### Tìm hiểu thêm

#### Tài liệu Android Studio:

- [Trang tải xuống Android Studio](#)
- [Tìm hiểu về Android Studio](#)

#### Tài liệu dành cho nhà phát triển Android:

- **ScrollView**
- **LinearLayout**
- **RelativeLayout**
- **View**
- **Button**
- **TextView**
- **String resources**
- **Relative Layout**

#### Khác:

- **Blog nhà phát triển Android:** Liên kết văn bản của bạn! (Linkify your Text!)
- **Codepath:** Làm việc với TextView (Working with a TextView)

### Bài tập về nhà

#### Thay đổi một ứng dụng

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

1. Thay đổi phần tiêu đề phụ (**subheading**) sao cho nó được gói gọn trong một cột bên trái có chiều rộng 100 dp, như minh họa bên dưới.
2. Đặt nội dung bài viết ở bên phải của phần tiêu đề phụ, như minh họa bên dưới.

### Trả lời các câu hỏi:

#### Câu hỏi 1:

**Bạn có thể sử dụng bao nhiêu View trong một ScrollView?**

Chọn một:

- Chỉ một View

- Một View hoặc một nhóm View
- Nhiều View tùy ý

**Câu hỏi 2:**

**Thuộc tính XML nào bạn sử dụng trong LinearLayout để hiển thị các View cạnh nhau?**

Chọn một:

- android:orientation="horizontal"
- android:orientation="vertical"
- android:layout\_width="wrap\_content"

**Câu hỏi 3:**

**Thuộc tính XML nào bạn sử dụng để xác định chiều rộng của LinearLayout bên trong ScrollView?**

Chọn một:

- android:layout\_width="wrap\_content"
- android:layout\_width="match\_parent"
- android:layout\_width="200dp"

**Nộp ứng dụng của bạn để được chấm điểm**

**Hướng dẫn cho người chấm bài:**

- Kiểm tra xem ứng dụng có các tính năng sau:  
Giao diện hiển thị tiêu đề phụ ở cột bên trái và văn bản bài viết ở cột bên phải, như minh họa trong hình trên.
- ScrollView bao gồm một LinearLayout với hai phần tử TextView.
- Thuộc tính orientation của LinearLayout được đặt là horizontal.

## **Bài học 1.4: Học cách tự giúp bản thân**

### **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 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** để giải quyết vấn đề hoặc bổ sung tính năng.
- **Cách thêm biểu tượng khởi chạy (launcher icon)** cho ứng dụng của bạn.
- **Cách tìm kiếm sự trợ giúp** khi gặp khó khăn trong quá trình phát triển ứng dụng Android.

#### **Những gì bạn sẽ làm**

- **Khám phá các tài nguyên dành cho nhà phát triển Android** ở mọi trình độ, bao gồm tài liệu, hướng dẫn và cộng đồng.
- **Thêm biểu tượng khởi chạy (launcher icon)** cho ứng dụng HelloToast hoặc bất kỳ ứng dụng nào bạn đã tạo.

## Tổng quan về ứng dụng

Bạn sẽ cài tiến ứng dụng của mình bằng cách thêm một **biểu tượng khởi chạy** – biểu tượng nhỏ đại diện cho ứng dụng của bạn trên màn hình chính hoặc trong danh sách ứng dụng. Nhiệm vụ này sẽ dạy bạn cách tùy chỉnh ứng dụng, cải thiện giao diện và tính tiện dụng của nó.

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

Mỗi ứng dụng mới bạn tạo bằng 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 ứng dụng trên Google Play Store. Khi người dùng tìm kiếm trên Google Play Store, 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 sẽ xuất hiện trên thiết bị ở nhiều nơi khác nhau, bao gồm màn hình chính và màn hình Tìm kiếm ứng dụng. Ví dụ, ứng dụng HelloToast xuất hiện trên màn hình Tìm kiếm ứng dụng của trình giả lập với biểu tượng tiêu chuẩn cho các dự án ứng dụng mới, như hình minh họa bên dưới.

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

### 1.1 Khám phá tài liệu chính thức của Android

Bạn có thể tìm thấy tài liệu chính thức dành cho nhà phát triển Android tại

[developer.android.com](http://developer.android.com).

Tài liệu này chứa rất nhiều thông tin và được Google cập nhật thường xuyên.

## 16. Truy cập [developer.android.com/design/](http://developer.android.com/design/).

Phần này nói về Material Design, một triết lý thiết kế mang tính khái niệm, định hướng cách các ứng dụng nên trông như thế nào và hoạt động ra sao trên các thiết bị di động. Hãy điều hướng 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.

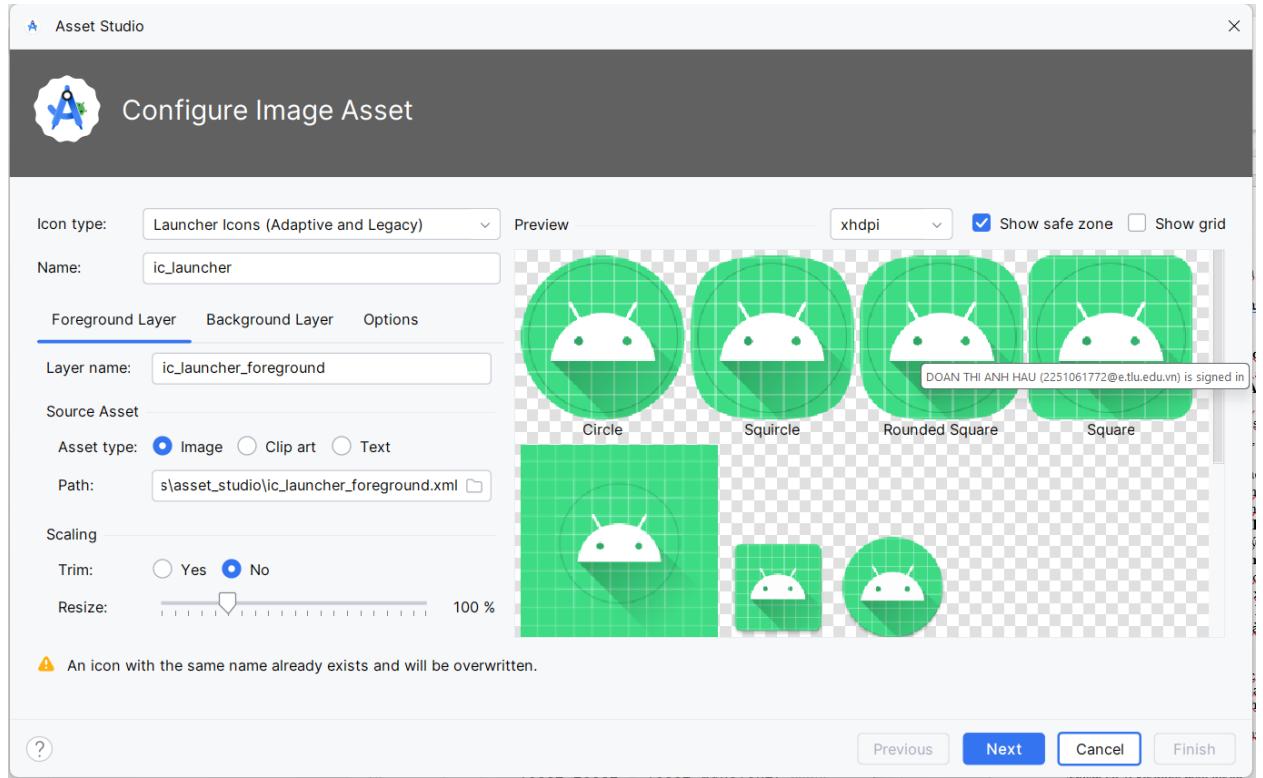
17. Truy cập [developer.android.com/docs/](http://developer.android.com/docs/) để tìm thông tin về API, tài liệu tham khảo, hướng dẫn, công cụ, và các mẫu mã.

18. Truy cập [developer.android.com/distribute/](http://developer.android.com/distribute/) để tìm thông tin về việc đưa ứng dụng lên **Google Play**, hệ thống phân phối kỹ thuật số của Google dành cho các ứng dụng phát triển bằng Android SDK. Sử dụng Google Play Console để phát triển cơ sở người dùng của bạn và bắt đầu **kiếm tiền**.

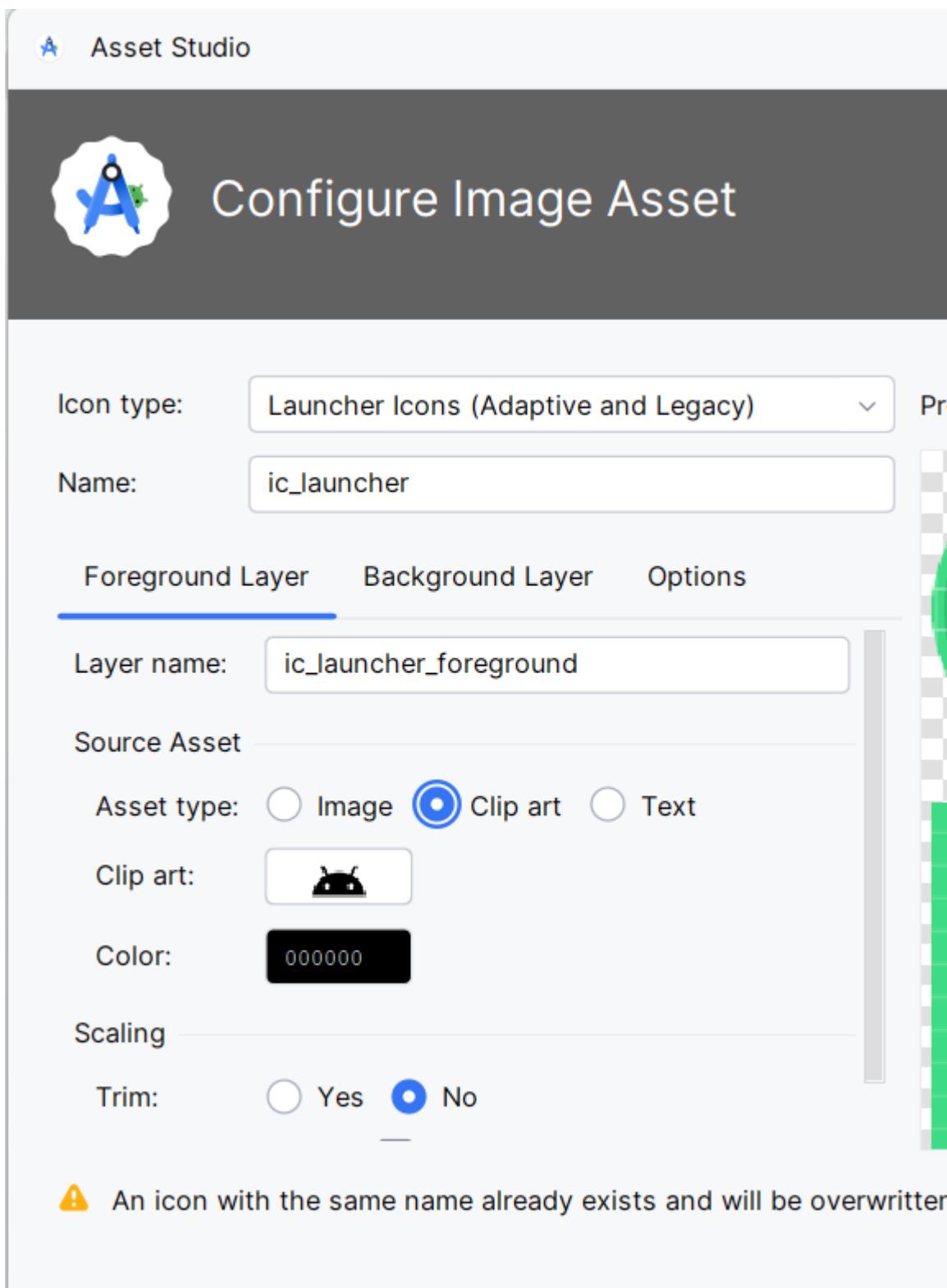
## 1.2 Thêm một tài nguyên hình ảnh cho biểu tượng khởi động

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

1. Mở dự án ứng dụng **HelloToast** từ bài học trước về cách sử dụng trình chỉnh sửa bộ cục, hoặc tạo một dự án ứng dụng mới.
2. Trong khung **Project > Android**, nhấp chuột phải (hoặc nhấn **Control** và nhấp chuột) vào thư mục **res** và chọn **New > Image Asset**. Cửa sổ **Configure Image Asset** sẽ xuất hiện.

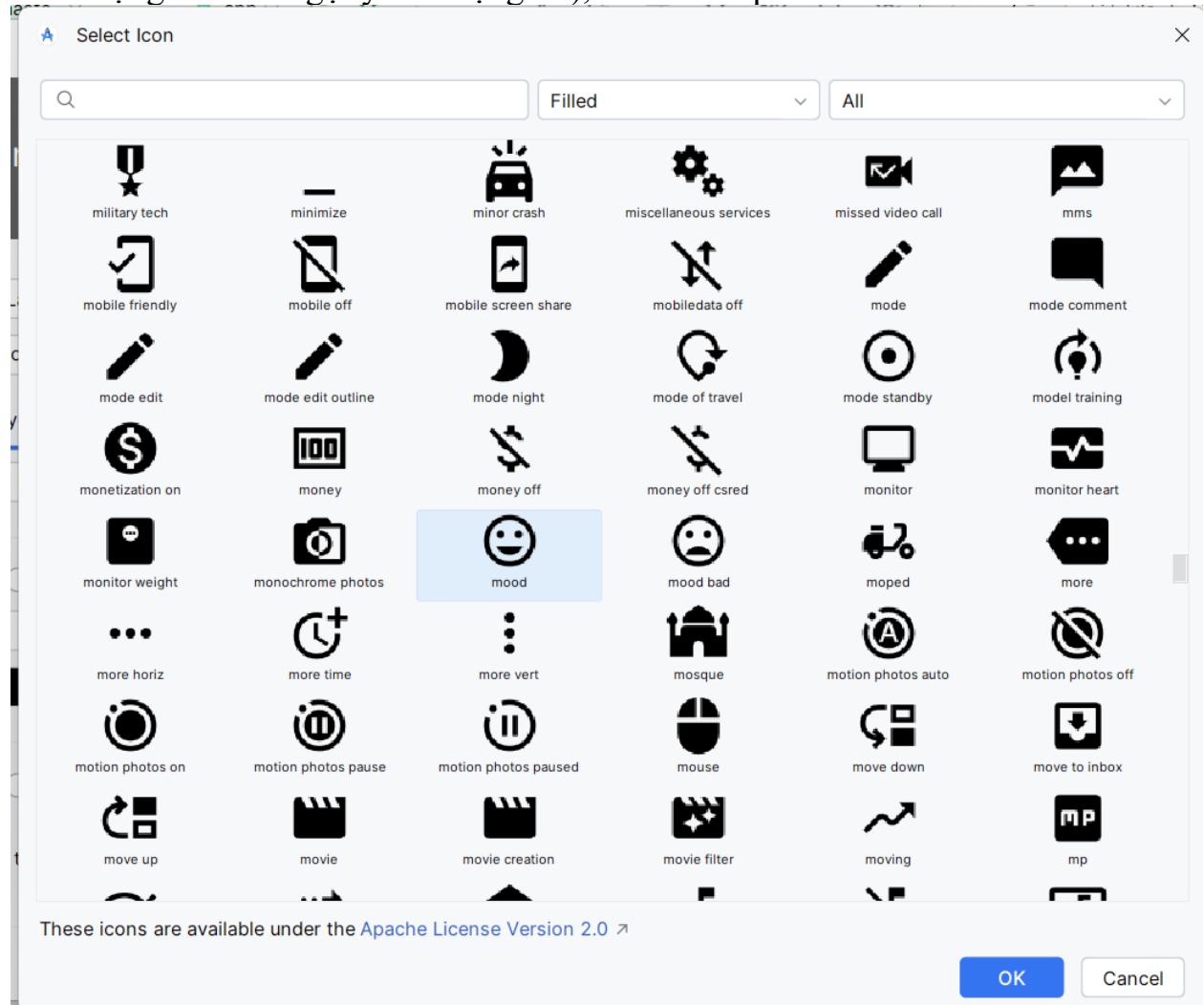


3. Trong trường **Icon Type**, chọn **Launcher Icons (Adaptive & Legacy)** nếu chưa được chọn.
4. Nhấp vào tab **Foreground Layer**, chọn **Clip Art** trong mục **Asset Type**.



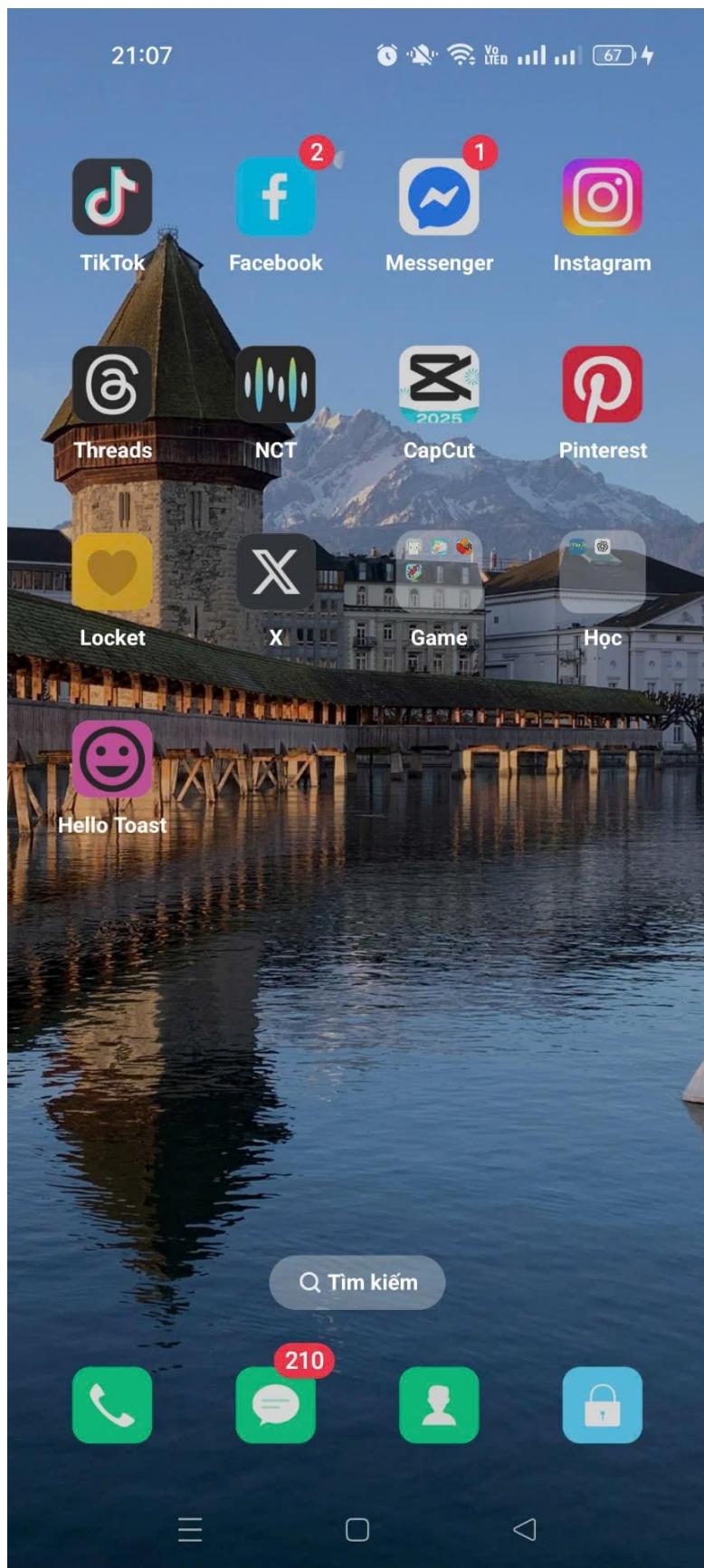
- Nhập vào biểu tượng trong trường **Clip Art**. Các biểu tượng từ bộ icon Material Design sẽ xuất hiện.

6. Duyệt qua cửa sổ **Select Icon**, chọn một biểu tượng phù hợp (chẳng hạn như biểu tượng mood để gợi ý tâm trạng tốt), sau đó nhấp vào **OK**.



7. Nhấp vào tab **Background Layer**, chọn **Color** trong mục **Asset Type**, sau đó nhấp vào ô màu để chọn màu sử dụng làm lớp nền.  
 8. Nhấp vào tab **Legacy** và xem lại các cài đặt mặc định. Xác nhận rằng bạn muốn tạo các biểu tượng **legacy**, **round**, và biểu tượng cho **Google Play Store**. Nhấp **Next** khi hoàn tất.  
 9. Chạy ứng dụng.

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



**Mẹo:** Xem **Launcher Icons** để tìm hiểu thêm về cách thiết kế các biểu tượng khởi chạy hiệu quả.

## Nhiệm vụ 2: Sử dụng mẫu dự án

**Android Studio** cung cấp các mẫu (templates) cho các thiết kế ứng dụng và hoạt động (activity) phổ biến cũng như được khuyến nghị. Việc sử dụng các mẫu tích hợp sẵn giúp tiết kiệm thời gian và đảm bảo bạn tuân theo các phương pháp thiết kế tốt nhất.

- Mỗi mẫu bao gồm một **bộ khung hoạt động (skeleton activity)** và giao diện người dùng cơ bản.
- Bạn đã sử dụng mẫu **Empty Activity**.
- 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 (options menu)** hiển thị trên **app bar**.

### 2.1 Khám phá kiến trúc của Basic Activity

Mẫu **Basic Activity** là một mẫu đa năng do **Android Studio** cung cấp để hỗ trợ bạn bắt đầu phát triển ứng dụng.

1. Trong **Android Studio**, tạo một dự án mới bằng mẫu **Basic Activity**.
2. Xây dựng (**Build**) và chạy (**Run**) ứng dụng.
3. Xác định các phần được gắn nhãn trong hình và bảng bên dưới. Tìm các phần tương ứng trên thiết bị hoặc màn hình trình giả lập của bạn. Kiểm tra mã nguồn Java và các tệp XML liên quan được mô tả trong bảng.

Hiểu rõ mã nguồn **Java** và các tệp **XML** sẽ giúp bạn mở rộng và tùy chỉnh mẫu này theo nhu cầu của mình.



## First Fragment

:

[Next](#)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam in scelerisque sem. Mauris volutpat, dolor id interdum ullamcorper, risus dolor egestas lectus, sit amet mattis purus dui nec risus. Maecenas non sodales nisi, vel dictum dolor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Suspendisse blandit eleifend diam, vel rutrum tellus vulputate quis. Aliquam eget libero aliquet, imperdiet nisl a, ornare ex. Sed rhoncus est ut libero porta lobortis. Fusce in dictum tellus.

Suspendisse interdum ornare ante. Aliquam nec cursus lorem. Morbi id magna felis. Vivamus egestas, est a condimentum egestas, turpis nisl iaculis ipsum, in dictum tellus dolor sed neq' Morbi tellus erat, dapibus ut s  a, iaculis tincidunt dui. Interdum et malesuada fames ac ante ipsum

#	Mô tả giao diện người dùng	Tham chiếu trong mã
1	<b>Thanh trạng thái</b> Hệ thống Android cung cấp và kiểm soát thanh trạng thái	Không hiển thị trong mã mẫu. Bạn có thể truy cập nó từ Activity của mình. Ví dụ, bạn có thể ẩn thanh trạng thái nếu cần thiết.
2	<b>AppBarLayout &gt; Toolbar</b> Thanh ứng dụng (còn gọi là <b>Action Bar</b> ) cung cấp cấu trúc giao diện, các thành phần giao diện tiêu chuẩn và điều hướng. Để đảm bảo tính tương thích ngược, <b>AppBarLayout</b> trong mẫu chứa một <b>Toolbar</b> với chức năng tương tự như <b>ActionBar</b>	Trong tệp <b>activity_main.xml</b> , tìm <b>android.support.v7.widget.Toolbar</b> bên trong <b>android.support.design.widget.AppBarLayout</b> . Thay đổi <b>Toolbar</b> để thay đổi giao diện của thành phần cha, <b>App Bar</b> . Để xem ví dụ, hãy tham khảo <b>App Bar Tutorial</b> .
3	<b>Tên ứng dụng</b> Tên này được lấy từ tên gói của bạn, nhưng có thể tùy chỉnh theo ý muốn	Trong tệp <b>AndroidManifest.xml</b> :  <code>android:label="@string/app_name"</code>
4	<b>Nút menu tùy chọn (Options menu overflow button)</b> Chứa các mục menu cho <b>Activity</b> , cũng như các tùy chọn chung như <b>Tìm kiếm (Search)</b> và <b>Cài đặt (Settings)</b> của ứng dụng. Các mục menu của ứng dụng sẽ được thêm vào menu này.	Trong <b>MainActivity.java</b> : <ul style="list-style-type: none"> <li>• <b>onOptionsItemSelected()</b> triển khai hành động xảy ra khi một mục menu được chọn.</li> </ul> Trong <b>res &gt; menu &gt; menu_main.xml</b> : <ul style="list-style-type: none"> <li>• Đây là tệp tài nguyên xác định các mục menu cho menu tùy chọn.</li> </ul>
5	<b>Bố cục ViewGroup CoordinatorLayout</b> là một <b>ViewGroup</b> giàu tính năng, cung cấp cơ chế để các phần tử <b>View (UI)</b> tương tác. Giao diện	Trong <b>activity_main.xml</b> Không có <b>View</b> nào được chỉ định trực tiếp trong bố cục này; thay vào đó, nó bao gồm một bố cục khác bằng lệnh <b>include layout</b> , để đưa <b>@layout/content_main</b> vào, nơi chứa các <b>View</b> .

	người dùng của ứng dụng được đặt trong tệp <b>content_main.xml</b> , tệp này được bao gồm bên trong <b>ViewGroup</b> này.	Điều này giúp tách biệt các <b>View</b> của hệ thống khỏi các <b>View</b> riêng của ứng dụng.
6	<b>TextView</b> Trong ví dụ, <b>TextView</b> được sử dụng để hiển thị " <b>Hello World</b> ". Hãy thay thế nó bằng các <b>phản tử giao diện người dùng (UI elements)</b> phù hợp với ứng dụng của bạn.	<b>Trong content_main.xml</b> Tất cả <b>các phản tử giao diện người dùng (UI elements)</b> của ứng dụng đều được định nghĩa trong tệp này.
7	Nút hành động nổi (Floating Action Button – FAB)	Trong <b>activity_main.xml</b> được thêm vào dưới dạng một phản tử giao diện người dùng (UI element) sử dụng biểu tượng clip-art.  Trong <b>MainActivity.java</b> bao gồm một đoạn mã mẫu (stub) trong <code>onCreate()</code> , thiết lập trình nghe sự kiện ( <code>onClick()</code> listener) cho FAB.

## 2.2 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 của thanh ứng dụng** để khớp với **thanh trạng thái** (trên một số thiết bị, thanh trạng thái có màu tối hơn của cùng một màu chính).

Bạn cũng có thể **xóa nút hành động nổi** nếu không sử dụng nó.

### 1. Thay đổi màu của thanh appbar(Toolbar) trong activity\_main.xml

- o Bằng cách thay đổi giá trị `android:background` thành:

```
 android:background="?attr/colorPrimaryDark"/>
```

- o Điều này sẽ đặt màu của thanh ứng dụng thành màu chính tối hơn, giúp khớp với thanh trạng thái.

## 2. Để xóa nút hành động nổi, bắt đầu bằng cách xóa đoạn mã trong `onCreate()`

- Thiết lập trình nghe sự kiện (`onClick()` listener) cho nút.
- Mở MainActivity và xóa khỏi mã sau.

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

## 3. Để xóa nút hành động nổi khỏi bộ cục, xóa khỏi mã XML sau

- Trong `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="16dp"
 android:layout_marginBottom="16dp"
 app:srcCompat="@android:drawable/ic_dialog_email" />
```

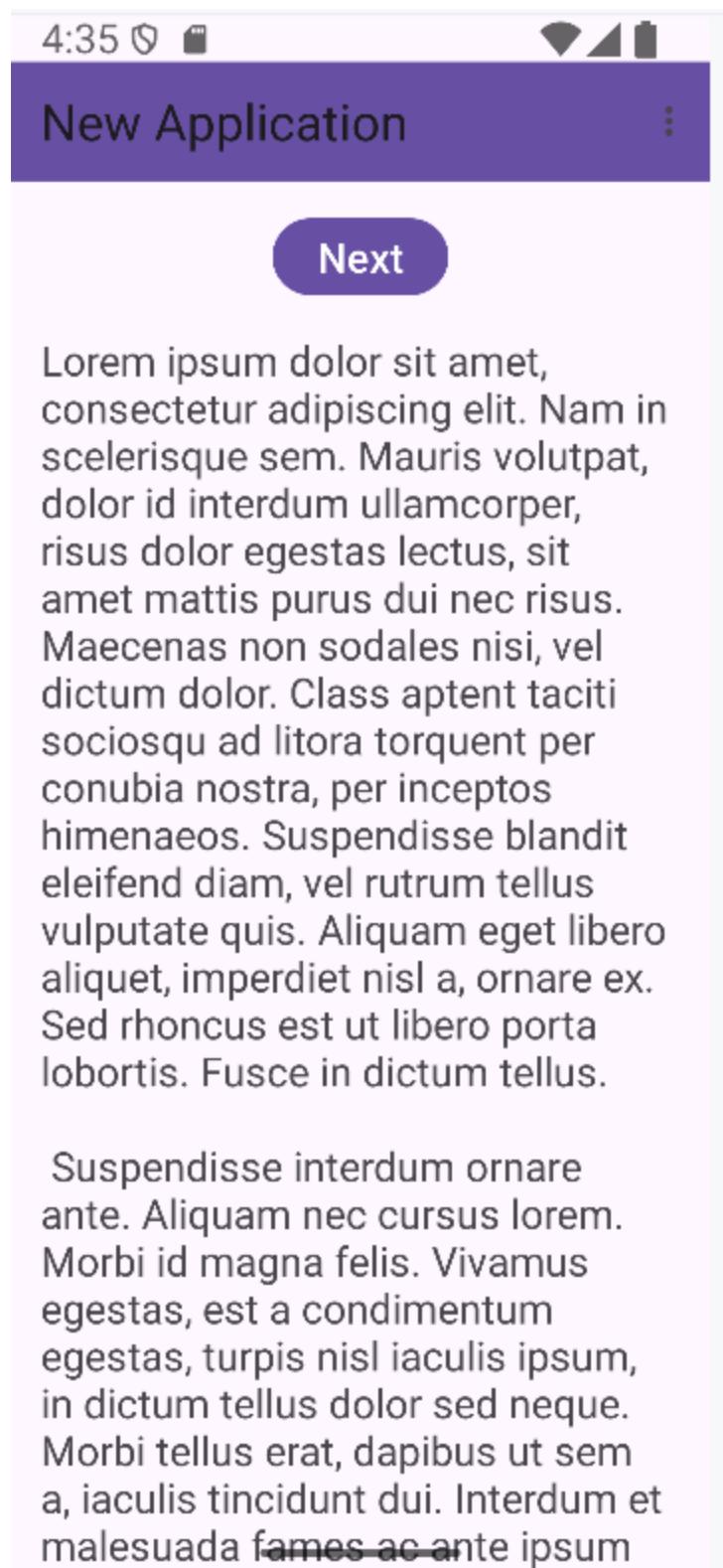
## 4. Thay đổi tên của ứng dụng hiển thị trên thanh ứng dụng

- Bằng cách thay đổi giá trị chuỗi `app_name` trong `strings.xml` thành nội dung sau.

```
<string name="app_name">New Application</string>
```

## 5. Chạy ứng dụng

- Nút hành động nổi không còn xuất hiện, tên ứng dụng đã thay đổi, và màu nền của thanh ứng dụng đã thay đổi.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam in scelerisque sem. Mauris volutpat, dolor id interdum ullamcorper, risus dolor egestas lectus, sit amet mattis purus dui nec risus. Maecenas non sodales nisi, vel dictum dolor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Suspendisse blandit eleifend diam, vel rutrum tellus vulputate quis. Aliquam eget libero aliquet, imperdiet nisl a, ornare ex. Sed rhoncus est ut libero porta lobortis. Fusce in dictum tellus.

Suspendisse interdum ornare ante. Aliquam nec cursus lorem. Morbi id magna felis. Vivamus egestas, est a condimentum egestas, turpis nisl iaculis ipsum, in dictum tellus dolor sed neque. Morbi tellus erat, dapibus ut sem a, iaculis tincidunt dui. Interdum et malesuada fames ac ante ipsum

**Mẹo:** Xem Truy cập tài nguyên để biết chi tiết về cú pháp XML khi truy cập tài nguyên.

## 1.4 Văn bản và các chế độ cuộn

Trong các bài thực hành trước, bạn đã sử dụng các mẫu **Empty Activity** và **Basic Activity**.

Trong các bài học sau, các mẫu bạn sử dụng sẽ thay đổi tùy theo nhiệm vụ.

Các mẫu **Activity** này cũng có sẵn trong dự án của bạn, cho phép bạn thêm **Activity** mới vào ứng dụng ngay cả sau khi thiết lập dự án ban đầu.

(Bạn sẽ tìm hiểu thêm về lớp Activity trong một chương khác.)

1. **Tạo một dự án ứng dụng mới hoặc chọn một dự án hiện có.**
2. **Trong ngăn Project > Android, nhấp chuột phải vào thư mục java.**
3. **Chọn New > Activity > Gallery.**
4. **Thêm một Activity.** Ví dụ: Nhấp vào **Navigation Drawer Activity** để thêm một Activity có **ngăn điều hướng (navigation drawer)** vào ứng dụng của bạn.
5. **Nhấp đúp vào các tệp bố cục của Activity để hiển thị chúng trong trình chỉnh sửa bố cục (layout editor).**

## Bài 3: Học từ mã nguồn mẫu

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

Mã nguồn mẫu Android

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

1. Trong Android Studio, chọn **File > New > Import Sample.**
2. Duyệt qua các mẫu có sẵn.
3. Chọn một mẫu và nhấp vào **Next.**
4. Chấp nhận các thiết lập mặc định và nhấp vào **Finish.**

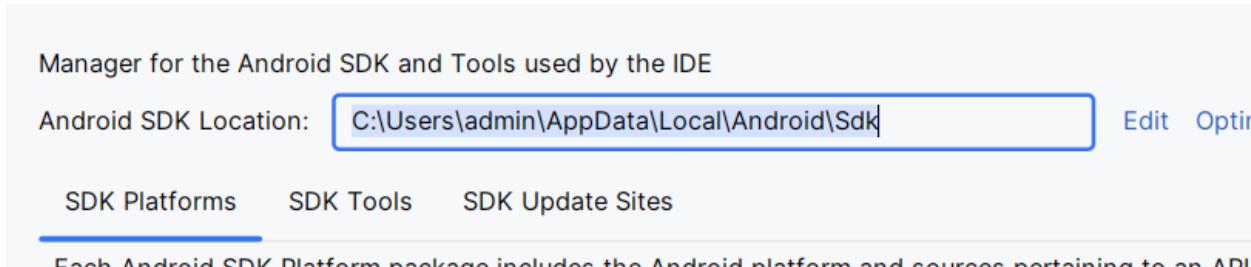
**Lưu ý:** Các mẫu này chỉ là điểm khởi đầu để phát triển thêm. Chúng tôi khuyến khích bạn tự thiết kế và xây dựng ý tưởng của riêng mình.

## Sử dụng SDK Manager để cài đặt tài liệu ngoại tuyến

Khi cài đặt Android Studio, các thành phần thiết yếu của Android SDK (Software Development Kit) cũng được cài đặt. Tuy nhiên, bạn có thể cài đặt thêm thư viện và tài liệu bằng SDK Manager.

1. Chọn **Tools > Android > SDK Manager.**
2. Ở cột bên trái, nhấp vào **Android SDK.**

- Sao chép đường dẫn trong phần **Android SDK Location** (bạn sẽ cần nó để tìm tài liệu trên máy tính).



- Nhấp vào tab **SDK Platforms** để cài đặt các phiên bản Android bổ sung.
- Nhấp vào tab **SDK Update Sites** để xem các trang cập nhật mà Android Studio kiểm tra thường xuyên.
- Nhấp vào tab **SDK Tools** để cài đặt các công cụ SDK bổ sung và tài liệu nhà phát triển Android ngoại tuyến.
- Chọn hộp kiểm **Documentation for Android SDK** nếu nó chưa được cài đặt, rồi nhấp vào **Apply**.
- Khi quá trình cài đặt hoàn tất, nhấp vào **Finish**.
- Điều hướng đến thư mục **sdk** đã sao chép trước đó, mở thư mục **docs**.
- Tìm tệp **index.html** và mở nó.

#### Bài 4: Nhiều nguồn tài nguyên hơn

- Kênh YouTube **Android Developer** là một nguồn tuyệt vời để học các hướng dẫn và mẹo hay.
- Đội ngũ Android thường xuyên đăng tin tức và mẹo trên **blog chính thức của Android**.
- Stack Overflow** là một cộng đồng lập trình viên hỗ trợ lẫn nhau. Nếu gặp vấn đề, rất có thể ai đó đã đăng câu trả lời. Bạn có thể thử đặt câu hỏi như:
  - "Làm thế nào để thiết lập và sử dụng ADB qua WiFi?"
  - "Những lỗi rò rỉ bộ nhớ phổ biến nhất trong lập trình Android là gì?"
- Cuối cùng nhưng không kém phần quan trọng, hãy tìm kiếm trên Google. Công cụ tìm kiếm sẽ thu thập kết quả từ tất cả các nguồn trên. Ví dụ: "Phiên bản Android OS phổ biến nhất ở Ấn Độ là gì?"

#### 1.5 Tài nguyên có sẵn

Truy cập **Stack Overflow** và nhập **[android]** vào ô tìm kiếm. Dấu **[]** giúp bạn tìm các bài viết có thẻ liên quan đến Android.

Bạn có thể kết hợp nhiều thẻ và từ khóa để tìm kiếm chính xác hơn. Hãy thử các tìm kiếm sau:

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

Để tìm hiểu thêm về cách tìm kiếm hiệu quả trên Stack Overflow, hãy xem **trung tâm trợ giúp của Stack Overflow**.

Tóm tắt

- **Tài liệu chính thức về Android Developer:** [developer.android.com](http://developer.android.com)
- **Material Design** là một triết lý thiết kế giúp ứng dụng hoạt động và hiển thị tốt trên thiết bị di động.
- **Google Play Store** là nền tảng phân phối ứng dụng của Google dành cho các ứng dụng phát triển bằng Android SDK.
- **Android Studio** cung cấp các mẫu thiết kế cho ứng dụng và hoạt động (activity) phổ biến, được khuyến nghị. Những mẫu này bao gồm mã nguồn hoạt động cho các trường hợp sử dụng phổ biến.
- Khi tạo một dự án, bạn có thể chọn một mẫu cho activity đầu tiên của mình. Trong quá trình phát triển ứng dụng, bạn có thể tạo thêm các activity và thành phần khác từ các mẫu có sẵn.
- **Android Studio** chứa nhiều đoạn mã mẫu mà bạn có thể nghiên cứu, sao chép và tích hợp vào dự án của mình.

## Khái niệm liên quan

Tài liệu về khái niệm liên quan có trong **Mục 1.4: Các tài nguyên giúp bạn học tập.**

Tìm hiểu thêm

Tài liệu về Android Studio

- **Giới thiệu về Android Studio**
- **Những bước cơ bản trong quy trình phát triển**

Tài liệu dành cho lập trình viên Android

- **Trang web dành cho lập trình viên Android**
- **Khóa đào tạo của Google Developers**
- **Bố cục (Layouts)**

- **Tổng quan về tài nguyên ứng dụng**
- **Bố cục (Layouts)**
- **Menu**
- **TextView**
- **Tài nguyên chuỗi (String resources)**
- **Tệp khai báo ứng dụng (App Manifest)**

Mã nguồn mẫu

- **Mã nguồn bài tập trên GitHub**
- **Mã mẫu dành cho lập trình viên Android**

Video

- **Kênh YouTube Android Developer**
- **Các khóa học trực tuyến trên Udacity**

Khác

- **Blog chính thức của Android**
- **Blog của Android Developers**
- **Google Developers Codelabs**
- **Stack Overflow**
- **Từ vựng Android**

Bài tập về nhà

## **Tải một ứng dụng mẫu và khám phá tài nguyên**

1. Tải một ứng dụng mẫu vào **Android Studio**.
2. Mở một tệp **Java activity** trong ứng dụng. Tìm một lớp, kiểu dữ liệu hoặc thủ tục mà bạn chưa quen thuộc và tra cứu trong tài liệu **Android Developer**.
3. Truy cập **Stack Overflow** và tìm các câu hỏi về chủ đề tương tự.
4. Thay đổi biểu tượng khởi chạy. Sử dụng một biểu tượng có sẵn trong mục **image assets** của **Android Studio**.

## Bài học 2.1: Activities và Intents

### Giới thiệu

Một **Activity** đại diện cho một màn hình đơn lẻ trong ứng dụng của bạn, nơi người dùng có thể thực hiện một tác vụ cụ thể, chẳng hạn như chụp ảnh, gửi email hoặc xem bản đồ. Một hoạt động (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 được kết nối lồng léo với nhau. Mỗi màn hình là một activity. Thông thường, một activity trong ứng dụng được chỉ định là "**main activity**" (**MainActivity.java**), và đây là activity được hiển thị khi ứng dụng được khởi chạy. Activity chính có thể khởi động các activity khác để thực hiện các hành động khác nhau.

Mỗi khi một activity mới được khởi động, activity trước đó sẽ dừng lại, nhưng hệ thống sẽ lưu activity đó trong một ngăn xếp (ngăn xếp "back stack"). Khi một activity mới được khởi động, activity mới này sẽ được đẩy vào ngăn xếp và nhận quyền điều khiển từ người dùng. Ngăn xếp "back stack" tuân theo nguyên tắc cơ bản "vào sau, ra trước" (**last in, first out**). Khi người dùng hoàn thành với activity hiện tại và nhấn nút **Back**, activity đó sẽ được xóa khỏi ngăn xếp và bị hủy, và activity trước đó sẽ được tiếp tục.

Một activity được khởi động hoặc kích hoạt bằng cách sử dụng một **intent**. Một **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 khác trong ứng dụng. Bạn sử dụng intent để khởi động một activity từ một activity khác và truyền dữ liệu giữa các activity.

**Intent** có thể là **explicit** hoặc **implicit**:

- Một **explicit intent** (intent rõ ràng) là intent trong đó bạn biết rõ mục tiêu của intent đó. Nghĩa là, bạn đã biết tên lớp đầy đủ (fully qualified class name) của activity cụ thể.
- Một **implicit intent** (intent không rõ ràng) là intent trong đó bạn không có tên của thành phần mục tiêu, nhưng bạn có một hành động chung để thực hiện.

Trong bài thực hành này, bạn sẽ tạo các **explicit intents**. Bạn sẽ tìm hiểu cách sử dụng **implicit intents** trong bài thực hành sau.

### Những gì bạn nên biết trước:

Bạn cần có khả năng:

- Tạo và chạy các ứng dụng trong Android Studio.
- Sử dụng trình chỉnh sửa bố cục (**layout editor**) để tạo bố cục trong một **ConstraintLayout**.
- Chỉnh sửa mã XML của bố cục.
- Thêm chức năng **onClick** cho một **Button**.

### Những gì bạn sẽ học được:

- Cách tạo một **Activity** mới trong Android Studio.
- Cách định nghĩa **parent** (cha) và **child activities** (hoạt động con) để sử dụng điều hướng **Up navigation**.
- Cách khởi động một **Activity** bằng một **explicit Intent**.
- Cách truyền dữ liệu giữa các **Activity** bằng một **explicit Intent**.

### Những gì bạn sẽ làm:

- Tạo một ứng dụng Android mới với một **main Activity** (activity chính) và một **second Activity** (activity thứ hai).
- Truyền một số dữ liệu (chuỗi) từ **main Activity** sang **second Activity** bằng một **Intent**, và hiển thị dữ liệu đó trong **second Activity**.
- Gửi một dữ liệu khác (dạng chuỗi) ngược lại từ **second Activity** về **main Activity**, cũng bằng một **Intent**.

### Tổng quan ứng dụng:

Trong chương này, bạn sẽ tạo và xây dựng một ứng dụng gọi là **Two Activities**, đúng như tên gọi, ứng dụng này chứa hai **Activity**. Bạn sẽ xây dựng ứng dụng qua ba giai đoạn:

- Ở giai đoạn đầu tiên, bạn tạo một ứng dụng với **main activity** chứa một nút **Send**. Khi người dùng nhấn nút này, **main activity** sẽ sử dụng một **intent** để khởi động **second activity**.
- Ở giai đoạn thứ hai, bạn thêm một thành phần **EditText** vào **main activity**. Người dùng nhập một thông điệp và nhấn nút **Send**. **Main activity** sử dụng

một **intent** để khởi động **second activity** và gửi thông điệp của người dùng đến **second activity**. **Second activity** hiển thị thông điệp mà nó nhận được.

- Ở giai đoạn cuối cùng của việc tạo ứng dụng **Two Activities**, bạn thêm một thành phần **EditText** và một nút **Reply** vào **second activity**. Bây giờ, người dùng có thể nhập một thông điệp phản hồi và nhấn nút **Reply**, và thông điệp phản hồi sẽ được hiển thị trên **main activity**. Trong giai đoạn này, bạn sử dụng một **intent** để chuyển thông điệp phản hồi từ **second activity** trở lại **main activity**.

## Nhiệm vụ 1: Tạo dự án TwoActivities

Trong nhiệm vụ này, bạn thiết lập dự án ban đầu với một **main Activity**, định nghĩa bối cảnh, và tạo một phương thức cơ bản cho sự kiện nhấn nút **onClick**.

## Bài 2: Activities

1. Mở **Android Studio** và tạo một **dự án Android Studio mới**.
  - Đặt tên ứng dụng là **Two Activities** và chọn cài đặt **Phone and Tablet** giống như các bài thực hành trước.
  - Thư mục dự án sẽ tự động được đặt tên là **TwoActivities**, và tên ứng dụng hiển thị trên **App Bar** sẽ là "**Two Activities**".
2. Chọn **Empty Activity** làm mẫu **Activity**. Nhấn **Next**.
3. Chấp nhận tên **Activity** mặc định là **MainActivity**.
  - Đảm bảo rằng **Generate Layout file** và **Backwards Compatibility (AppCompat)** đã được chọn.
4. Nhấn **Finish** để hoàn tất tạo dự án.

### 2.1 Activity và intent

1. Mở **res > layout > activity\_main.xml** trong **Project > Android**. Trình chỉnh sửa bối cảnh (**Layout Editor**) sẽ xuất hiện.
2. Nhấn vào tab **Design** (nếu chưa được chọn) và xóa **TextView** mặc định (có nội dung "**Hello World**") trong **Component Tree**.
3. Khi chế độ **Autoconnect** đang bật (mặc định), kéo một **Button** từ **Palette** vào góc dưới bên phải của bối cảnh. **Autoconnect** sẽ tự động tạo các ràng buộc (**constraints**) cho **Button**.
4. Trong **Attributes**, đặt các thuộc tính sau:
  - **ID:** button\_main
  - **layout\_width** và **layout\_height:** wrap\_content
  - **Text:** Send

Lúc này, bộ cục sẽ trông như sau:



5. Nhập vào tab **Text** để chỉnh sửa mã XML. Thêm thuộc tính sau vào Button:

```
android:onClick="launchSecondActivity"
```

Giá trị của thuộc tính được gạch dưới màu đỏ vì phương thức launchSecondActivity() chưa được tạo. Bỏ qua lỗi này tạm thời; bạn sẽ sửa trong nhiệm vụ tiếp theo.

**6. Trích xuất tài nguyên chuỗi**, như đã mô tả trong bài thực hành trước, cho văn bản "Send" và sử dụng tên tài nguyên là button\_main.

Mã XML cho nút Button sẽ trông như sau:

```
<Button
 android:id="@+id/button_main"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginRight="16dp"
 android:layout_marginBottom="16dp"
 android:onClick="launchSecondActivity"
 android:text="@string/button_main"
 app:layout_constraintBaseline_toBottomOf="parent"
 app:layout_constraintRight_toRightOf="parent"
 tools:ignore="OnClick" />
```

### 1.3 Xác định hành động của Button

Trong tác vụ này, bạn sẽ triển khai phương thức launchSecondActivity() mà bạn đã tham chiếu trong bộ cục thông qua thuộc tính android:onClick.

1. Nhập vào "launchSecondActivity" trong mã XML của activity\_main.xml.
2. Nhấn Alt+Enter (hoặc Option+Enter trên Mac) và chọn **Create 'launchSecondActivity(View)' in 'MainActivity'**.
  - o File MainActivity sẽ mở ra, và Android Studio sẽ tự động tạo một phương thức khung cho trình xử lý launchSecondActivity().
3. Bên trong launchSecondActivity(), thêm một câu lệnh Log với nội dung "Button Clicked!".

```
public void launchSecondActivity(View view) {
 Log.d(LOG_TAG, msg: "Button clicked!");
}
```

LOG\_TAG sẽ hiển thị màu đỏ. Bạn sẽ thêm định nghĩa cho biến đó trong một bước tiếp theo

4. Ở đầu lớp MainActivity, thêm hằng số cho biến LOG\_TAG:

```
1 usage
private static final String LOG_TAG = MainActivity.class.getSimpleName();
@Override
```

Hằng số này sử dụng tên của chính lớp làm thê.

5. Chạy ứng dụng của bạn. Khi bạn nhấn nút Send, bạn sẽ thấy thông báo "Button Clicked!" trong bảng Logcat. Nếu có quá nhiều đầu ra trong màn hình, hãy nhập **MainActivity** vào hộp tìm kiếm, và bảng Logcat sẽ chỉ hiển thị các dòng khớp với thẻ đó.

Mã cho MainActivity sẽ trông như sau:

```
package com.example.twoactivities;

import ...

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

 1 usage
 public void launchSecondActivity(View view) {
 Log.d(LOG_TAG, msg: "Button clicked!");
 }
}
```

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

Mỗi activity mới mà bạn thêm vào dự án sẽ có tệp layout và tệp Java riêng, tách biệt với những tệp của activity chính. Các activity này 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 mở rộng 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 **parent** (cha) của một activity khác trong tệp **AndroidManifest.xml**. Mỗi quan hệ cha-con này cho phép Android thêm các gợi ý điều hướng, chẳng hạn như mũi tên quay lại hướng trái trong thanh tiêu đề của mỗi activity.

Một activity giao tiếp với các activity khác (trong cùng một ứng dụng hoặc giữa các ứng dụng khác nhau) bằng một **intent**. Một **Intent** có thể là:

- **Intent rõ ràng (explicit intent)**: Là loại mà bạn biết rõ mục tiêu của intent đó; nghĩa là bạn đã biết tên lớp đầy đủ của activity cụ thể đó.
- **Intent không rõ ràng (implicit intent)**: Là loại mà bạn không có tên của thành phần mục tiêu, nhưng có một hành động tổng quát để thực hiện.

Trong nhiệm vụ này, bạn thêm một activity thứ hai vào ứng dụng, với layout riêng của nó. Bạn chỉnh sửa tệp **AndroidManifest.xml** để định nghĩa activity chính là **parent** của activity thứ hai. Sau đó, bạn chỉnh sửa phương thức **launchSecondActivity()** trong **MainActivity** để bao gồm một intent giúp khởi chạy activity thứ hai khi bạn nhấn nút.

### 2.1 Tạo Activity thứ hai

1. Nhấp vào thư mục **app** của dự án và chọn **File > New > Activity > Empty Activity**.
2. Đặt tên cho Activity mới là **SecondActivity**. Đảm bảo rằng các tùy chọn **Generate Layout File** và **Backwards Compatibility (AppCompat)** được chọn. Tên layout sẽ được tự động điền là **activity\_second**.  
**Không chọn** tùy chọn **Launcher Activity**.
3. Nhấn **Finish**. Android Studio sẽ thêm cả một tệp layout mới (**activity\_second.xml**) và một tệp Java mới (**SecondActivity.java**) vào dự

án của bạn cho Activity mới. Đồng thời, nó cũng cập nhật tệp **AndroidManifest.xml** để bao gồm Activity mới.

## 2.2 Sửa đổi tệp AndroidManifest.xml

1. Mở **manifests > AndroidManifest.xml**.
2. Tìm phần tử `<activity>` mà Android Studio đã tạo cho **SecondActivity**:

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

```
<activity
 android:name=".SecondActivity"
 android:exported="false" />
<activity
```

3. Thay thế toàn bộ phần tử `<activity>` bằng nội dung sau:

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

**Thuộc tính label** thêm tiêu đề của **Activity** vào thanh ứng dụng (app bar). VỚI thuộc tính `parentActivityName`, bạn chỉ định rằng **MainActivity** là cha (parent) của **SecondActivity**. Mỗi quan hệ này được sử dụng cho điều hướng **Up navigation** trong ứng dụng của bạn: thanh ứng dụng của **SecondActivity** sẽ có mũi tên quay trái để người dùng có thể điều hướng "lên trên" (quay lại) **MainActivity**.

VỚI phần tử `<meta-data>`, bạn cung cấp thông tin tùy ý bổ sung về Activity dưới dạng các cặp key-value. Trong trường hợp này, các thuộc tính metadata thực hiện cùng một chức năng như thuộc tính `android:parentActivityName`—chúng định nghĩa một mối quan hệ giữa hai Activity cho điều hướng "lên trên". Các thuộc tính metadata này là bắt buộc

đối với các phiên bản Android cũ hơn, vì thuộc tính android:parentActivityName chỉ khả dụng cho API từ cấp 16 trở lên.

4. Trích xuất tài nguyên chuỗi (string resource) cho "Second Activity" trong đoạn mã trên, và sử dụng tên tài nguyên là activity2\_name.

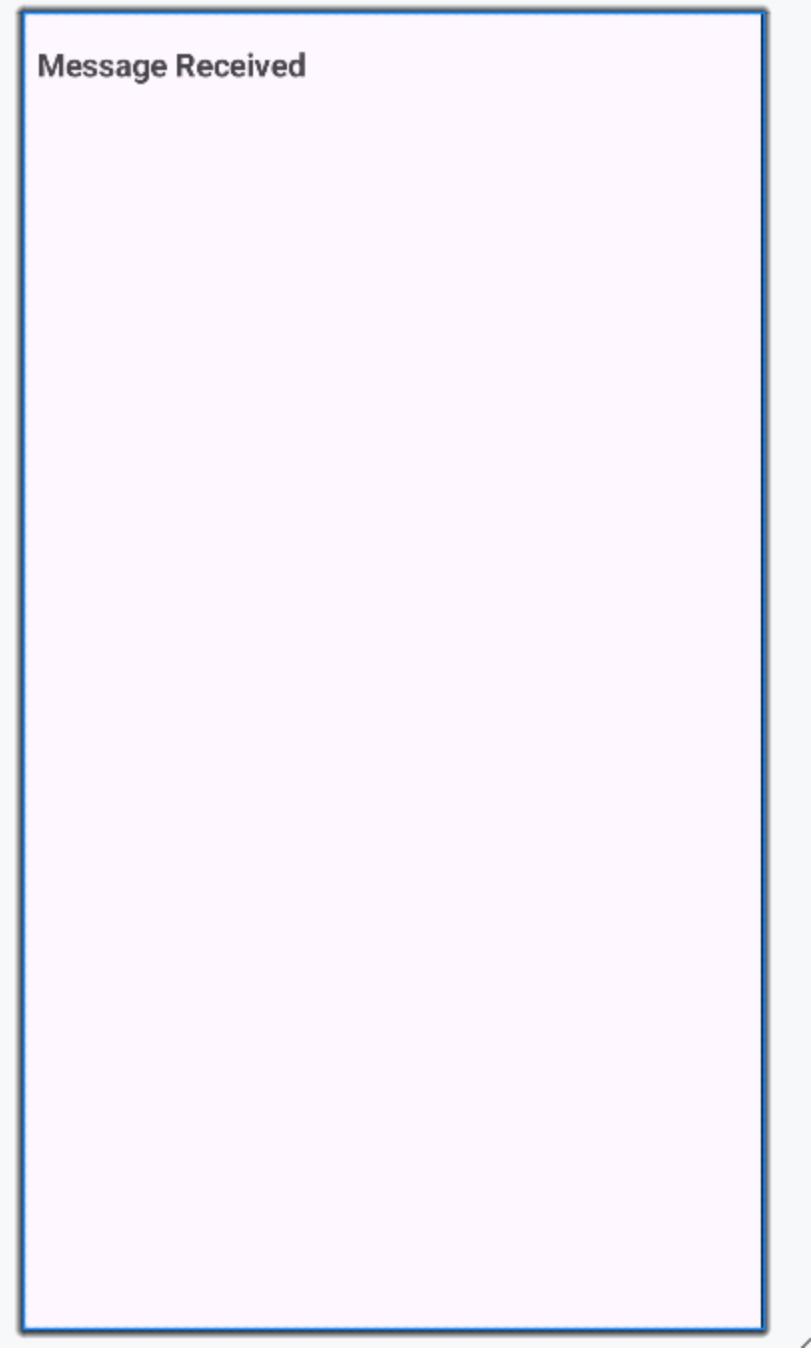
### 2.3 Định nghĩa bố cục cho Activity thứ hai

1. Mở tệp **activity\_second.xml** và nhấp vào tab **Design** nếu nó chưa được chọn.
2. Kéo một **TextView** từ bảng **Palette** vào góc trên bên trái của bố cục và thêm các ràng buộc (**constraints**) vào các cạnh trên và trái của bố cục.
  - o Thiết lập các thuộc tính của nó trong bảng **Attributes** như sau:

Attribute	Value
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 Android theme, định nghĩa các kiểu phông chữ cơ bản. Bạn sẽ tìm hiểu thêm về các theme trong một bài học sau.

Bố cục bây giờ sẽ trông như thế này:



Message Received

3. Nhập vào tab **Text** để chỉnh sửa mã XML, sau đó trích xuất chuỗi "**Message Received**" thành một tài nguyên có tên là **text\_header**.
4. Thêm thuộc tính **android:layout\_marginLeft="8dp"** vào **TextView** để bổ sung cho thuộc tính **layout\_marginStart** dành cho các phiên bản Android cũ hơn.

Mã XML cho tệp **activity\_second.xml** nên 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="@string/text_header"
 android:textAppearance="@style/TextAppearance.AppCompat.Medium"
 android:textStyle="bold"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 />
</androidx.constraintlayout.widget.ConstraintLayout>

```

## 2.4 Thêm một Intent vào Main Activity

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

### Các bước thực hiện:

- Mở MainActivity.java.**
- Tạo một Intent mới** trong phương thức launchSecondActivity().

Constructor của **Intent** yêu cầu hai đối số cho một **explicit Intent**:

- Context:** Ngữ cảnh ứng dụng (sử dụng this để tham chiếu đến **MainActivity**).
- Component:** Thành phần cụ thể sẽ nhận Intent (sử dụng SecondActivity.class).

```
Intent intent = new Intent(packageContext: this, SecondActivity.class);
```

- Gọi phương thức **startActivity()** với đối tượng **Intent** mới làm đối số.

```
startActivity(intent);
```

#### 4. Chạy ứng dụng.

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

### Task 3: Gửi dữ liệu từ **MainActivity** sang **SecondActivity**

Trong nhiệm vụ trước, bạn đã thêm một intent rõ ràng vào **MainActivity** để khởi chạy **SecondActivity**. Bạn cũng có thể sử dụng một intent để **gửi dữ liệu** từ một activity này sang một activity khác trong quá trình khởi chạy.

Đối tượng intent có thể truyền dữ liệu tới activity mục tiêu theo hai cách: trong trường **data**, hoặc trong phần **intent extras**.

- Dữ liệu trong intent là một **URI** chỉ định **dữ liệu cụ thể cần xử lý**.
- Nếu thông tin bạn muốn truyền tới activity thông qua intent không phải là một **URI**, hoặc nếu bạn muốn gửi nhiều thông tin, bạn có thể đưa thông tin bổ sung vào phần **extras**.

**Extras của intent** là các cặp key/value được lưu trữ trong một **Bundle**. **Bundle** là một tập hợp dữ liệu được lưu trữ dưới dạng các cặp key/value.

Để truyền thông tin từ một activity này sang một activity khác, bạn đưa các key và value vào phần **extras** của intent từ activity gửi và sau đó lấy chúng ra trong activity nhận.

Trong nhiệm vụ này, bạn sẽ chỉnh sửa intent rõ ràng trong **MainActivity** để thêm dữ liệu bổ sung (trong trường hợp này là một chuỗi do người dùng nhập vào) vào **extras** của intent. Sau đó, bạn chỉnh sửa **SecondActivity** để lấy dữ liệu này từ **extras** của intent và hiển thị trên màn hình.

#### 3.1 Thêm một **EditText** vào bố cục của **MainActivity**

1. Mở tệp **activity\_main.xml**.
2. Kéo một thành phần **Plain Text (EditText)** từ **Palette** vào phía dưới của bố cục:
  - Thêm ràng buộc (constraints) vào cạnh trái của bố cục, cạnh dưới của bố cục, và cạnh trái của nút **Send**.

- Đặt các thuộc tính trong bảng **Attributes** như sau:

Attribute	Value
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)

Bố cục mới trong tệp activity\_main.xml sẽ trông như thế này:

5:58



## TwoActivities

Enter Your Message H

Send

3. Nhấn vào tab **Text** để chỉnh sửa mã XML và trích xuất chuỗi "**Enter Your Message Here**" vào một tài nguyên (resource) với tên là **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_marginRight="16dp"
 android:onClick="launchSecondActivity"
 android:text="@string/button_main"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintRight_toRightOf="parent" />

 <EditText
 android:id="@+id/editText_main"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginRight="8dp"
 android:layout_marginLeft="8dp"
 android:layout_marginBottom="16dp"
 android:ems="10"
 android:inputType="textLongMessage"
 android:hint="@string/editText_main"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toStartOf="@+id/button_main" />
```

```
 app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

### 3.2 Thêm một chuỗi vào Intent extras

Các **Intent extras** là các cặp key/value được lưu trong một **Bundle**. Một **Bundle** là một tập hợp dữ liệu được lưu trữ dưới dạng cặp key/value. Để truyền thông tin từ một **Activity** sang một **Activity** khác, bạn cần đưa các cặp key/value vào **Intent extras** từ **Activity** gửi, và sau đó lấy chúng ra từ **Intent extras** trong **Activity** nhận.

1. Mở **MainActivity**.
2. Thêm một hằng số **public** ở đầu lớp để định nghĩa key cho **Intent extra**:

```
no usages
public static final String EXTRA_MESSAGE = "com.example.android.twoactivities.extra.MESSAGE";
^
```

3. Thêm một biến riêng tư ở đầu lớp để chứa **EditText**.

```
private EditText mMessageEditText;
```

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

```
mMessageEditText = findViewById(R.id.editText_main);|
```

5. Trong phương thức **launchSecondActivity()**, ngay dưới Intent mới, lấy văn bản từ **EditText** dưới dạng chuỗi.

```
String message = mMessageEditText.getText().toString();
```

6. Thêm chuỗi đó vào Intent dưới dạng một extra với hằng số **EXTRA\_MESSAGE** làm khóa và chuỗi làm giá trị

```
intent.putExtra(EXTRA_MESSAGE, message);
```

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

```

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

```

Phương thức launchSecondActivity() trong MainActivity bây giờ sẽ trông như sau:

```

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

```

### 3.3 Thêm một TextView vào SecondActivity để hiển thị thông điệp

1. Mở tệp activity\_second.xml.
2. Kéo một TextView khác vào giao diện bên dưới TextView có ID là text\_header, và thêm các ràng buộc (constraints) vào cạnh trái của giao diện và cạnh dưới của text\_header.
3. Đặt các thuộc tính cho TextView mới trong bảng Attributes như sau:

Attribute	Value
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

Giao diện mới trông giống như trong bài tập trước, vì TextView mới chưa (chưa có) chứa bất kỳ văn bản nào, vì vậy nó không xuất hiện trên màn hình. Mã XML cho giao diện activity\_second.xml sẽ trông giống như sau

```

<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="@string/text_header"
 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_marginLeft="8dp"
 android:layout_marginTop="8dp"
 android:textAppearance="AppCompat.Medium"
 app:layout_constraintBottom_toBottomOf="@+id/text_header"
 app:layout_constraintStart_toStartOf="parent" />

```

### 3.4 Sửa đổi SecondActivity để lấy dữ liệu extras và hiển thị thông điệp

1. Mở SecondActivity để thêm mã vào phương thức onCreate().
2. Lấy Intent kích hoạt Activity này:

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

3. Lấy chuỗi chứa thông điệp từ extras của Intent bằng cách sử dụng biến tĩnh MainActivity.EXTRA\_MESSAGE làm khóa:

```
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

4. Sử dụng findViewById() để tham chiếu đến TextView cho thông điệp từ giao diện:

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

- Đặt văn bản của TextView thành chuỗi từ extra của Intent:

```
textView.setText(message);
```

- Chạy ứng dụng. Khi bạn nhập thông điệp trong MainActivity và nhấn Send, SecondActivity sẽ được mở và hiển thị thông điệp.

Phương thức onCreate() của SecondActivity sẽ trông như sau:

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

Nhiệm vụ 4: Trả lại dữ liệu cho Activity chính

Bây giờ bạn đã có một ứng dụng khởi chạy một Activity mới và gửi dữ liệu đến đó, bước cuối cùng là trả lại dữ liệu từ Activity thứ hai về Activity chính. Bạn cũng sẽ sử dụng Intent và intent extras cho nhiệm vụ này.

4.1 Thêm một EditText và một Button vào giao diện SecondActivity

- Mở tệp strings.xml và thêm các tài nguyên chuỗi cho văn bản Button và gợi ý (hint) cho EditText mà bạn sẽ thêm vào SecondActivity:

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

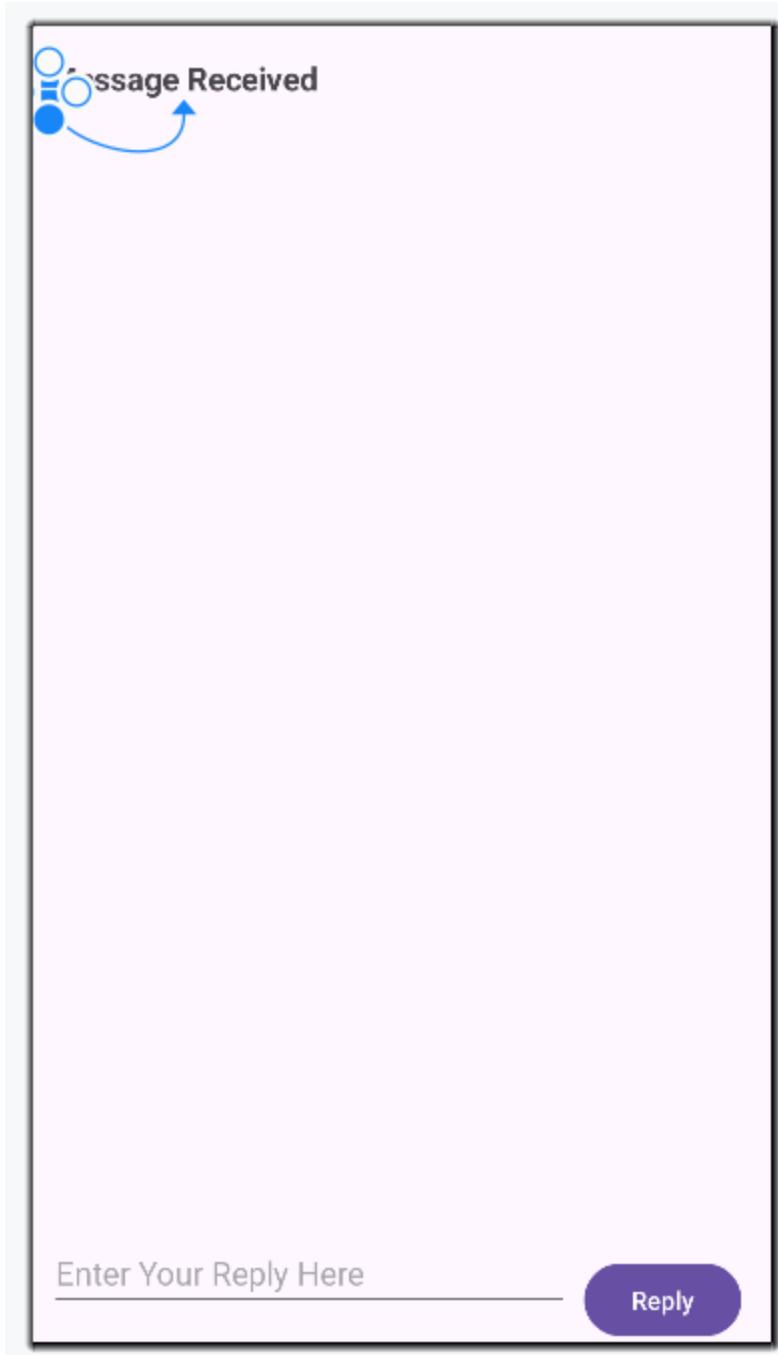
- Mở tệp activity\_main.xml và activity\_second.xml.
- Sao chép EditText và Button từ tệp layout activity\_main.xml và dán chúng vào layout activity\_second.xml.
- Trong activity\_second.xml, sửa đổi giá trị thuộc tính cho Button như sau:

<b>Old attribute value</b>	<b>New attribute value</b>
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"

5. Trong activity\_second.xml, sửa đổi giá trị thuộc tính cho EditText như sau:

<b>Old attribute value</b>	<b>New attribute value</b>
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"

6. Trong trình chỉnh sửa XML layout, nhấp vào returnReply, nhấn Alt+Enter (Option+Return trên Mac), và chọn Tạo 'returnReply(View)' trong 'SecondActivity'. Android Studio sẽ tạo ra một phương thức khung cho trình xử lý returnReply(). Bạn sẽ triển khai phương thức này trong nhiệm vụ tiếp theo. Giao diện mới cho activity\_second.xml trông như sau:



Mã XML cho tệp layout 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="@string/text_header"
 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_marginLeft="8dp"
 android:layout_marginTop="8dp"
 android:textAppearance="App.AppCompat.Medium"
 app:layout_constraintBottom_toBottomOf="@+id/text_header"
 app:layout_constraintStart_toStartOf="parent" />

 <Button
 android:id="@+id/button_second"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginRight="16dp"
 android:onClick="returnReply"
 android:text="@string/button_second"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintRight_toRightOf="parent" />

 <EditText
 android:id="@+id/editText_second"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginRight="8dp"
 android:layout_marginLeft="8dp"
 android:layout_marginBottom="16dp"
 android:ems="10"
 android:inputType="textLongMessage"
 android:hint="@string/editText_second"
```

```
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toStartOf="@+id/button_second"
 app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 4.2 Tạo một Intent phản hồi trong Activity thứ hai

Dữ liệu phản hồi từ Activity thứ hai về Activity chính được gửi trong một Intent extra.

Bạn tạo ra Intent trả lại này và đưa dữ liệu vào đó theo cách tương tự như cách bạn làm với Intent gửi đi.

1. Mở SecondActivity.

2. Ở đầu lớp, thêm một hằng số public để định nghĩa khóa cho Intent extra:

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

3. Thêm một biến riêng tư ở đầu lớp để chứa EditText.

```
private EditText mReply;
```

4. Trong phương thức onCreate(), trước mã Intent, sử dụng findViewById() để lấy tham chiếu đến EditText và gán nó cho biến riêng tư đó:

```
mReply = findViewById(R.id.editText_second);
```

5. Trong phương thức returnReply(), lấy văn bản của EditText dưới dạng chuỗi:

```
String reply = mReply.getText().toString();
```

7. Trong phương thức returnReply(), tạo một intent mới cho phản hồi — không tái sử dụng đối tượng Intent mà bạn nhận được từ yêu cầu ban đầu.

```
Intent replyIntent = new Intent();
```

7. Thêm chuỗi phản hồi từ EditText vào intent mới như một Intent extra. Vì extras là cặp khóa/giá trị, ở đây khóa là EXTRA\_REPLY và giá trị là reply:

```
replyIntent.putExtra(EXTRA_REPLY, reply);
```

8. 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\_CANCELLED.

```
setResult(RESULT_OK, replyIntent);
```

9. Gọi finish() để đóng Activity và quay lại MainActivity.

```
finish();
```

Mã cho SecondActivity bây giờ sẽ trông như sau:

```
package com.example.twoactivities;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

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

public class 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);
 EdgeToEdge.enable(this);
 setContentView(R.layout.activity_second);
 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;
 });
 Intent intent = getIntent();
 String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

```

 TextView textView = findViewById(R.id.text_message);
 textView.setText(message);
 mReply = findViewById(R.id.editText_second);
 }

 public void returnReply(View view) {
 String reply = mReply.getText().toString();
 Intent replyIntent = new Intent();
 replyIntent.putExtra(EXTRA_REPLY, reply);
 setResult(RESULT_OK, replyIntent);
 finish();
 }
}

```

#### 4.3 Thêm các phần tử TextView để hiển thị phản hồi

MainActivity cần một cách để hiển thị phản hồi mà SecondActivity gửi. Trong nhiệm vụ này, bạn sẽ thêm các phần tử TextView vào giao diện activity\_main.xml để hiển thị phản hồi trong MainActivity.

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

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

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

2. Mở tệp activity\_main.xml và activity\_second.xml.
3. Sao chép hai phần tử TextView từ tệp layout activity\_second.xml và dán chúng vào giao diện activity\_main.xml phía trên Button.
4. Trong activity\_main.xml, sửa đổi giá trị thuộc tính cho TextView đầu tiên như sau:

Old attribute value	New attribute value
android:id="@+id/text_header"	android:id="@+id/text_header_reply"
android:text= "@string/text_header"	android:text= "@string/text_header_reply"

5. Trong activity\_main.xml, sửa đổi giá trị thuộc tính cho TextView thứ hai như sau:

Old attribute value	New attribute value
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"

6. Thêm thuộc tính android:visibility vào mỗi TextView để làm chúng ban đầu vô hình. (Việc để chúng hiển thị trên màn hình mà không có nội dung có thể gây nhầm lẫn cho người dùng.)

`android:visibility="invisible",`

Bạn sẽ làm cho các phần tử TextView này hiển thị sau khi dữ liệu phản hồi được truyền lại từ Activity thứ hai.

Giao diện activity\_main.xml trông giống như trong nhiệm vụ trước — mặc dù bạn đã thêm hai phần tử TextView mới vào giao diện. Vì bạn đã đặt các phần tử này thành vô hình, nên chúng không hiển thị trên màn hình.

Dưới đây là mã XML cho tệp activity\_main.xml:

```
<?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_marginTop="16dp"
 android:text="@string/text_header_reply"
 android:textAppearance="@style/TextAppearance.AppCompat.Medium"
 android:textStyle="bold">
```

```
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"
 android:visibility="invisible"/>

<TextView
 android:id="@+id/text_message_reply"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginLeft="8dp"
 android:layout_marginTop="8dp"

 android:textAppearance="@style/TextAppearance.AppCompat.Medium"
 app:layout_constraintTop_toBottomOf="@+id/text_header_reply"
 app:layout_constraintStart_toStartOf="parent"
 android:visibility="invisible"/>

<Button
 android:id="@+id/button_main"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginRight="16dp"
 android:onClick="launchSecondActivity"
 android:text="@string/button_main"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintRight_toRightOf="parent" />

<EditText
 android:id="@+id/editText_main"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginRight="8dp"
 android:layout_marginLeft="8dp"
 android:layout_marginBottom="16dp"
 android:ems="10"
 android:inputType="textLongMessage"
 android:hint="@string/editText_main"
 app:layout_constraintBottom_toBottomOf="parent"
```

```
 app:layout_constraintEnd_toStartOf="@+id/button_main"
 app:layout_constraintStart_toStartOf="parent" />

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

#### 4.4 Lấy phản hồi từ Intent extra và hiển thị nó

Khi bạn sử dụng một Intent rõ ràng để bắt đầu một Activity khác, bạn có thể không mong đợi nhận lại bất kỳ dữ liệu nào — bạn chỉ đang kích hoạt Activity đó. Trong trường hợp này, bạn sử dụng `startActivity()` để bắt đầu Activity mới, như bạn đã làm trước đó trong bài thực hành này. Tuy nhiên, nếu bạn muốn lấy dữ liệu từ Activity đã được kích hoạt, bạn cần bắt đầu nó với `startActivityForResult()`.

Trong nhiệm vụ này, bạn sẽ chỉnh sửa ứng dụng để bắt đầu SecondActivity và mong đợi một kết quả, lấy dữ liệu trả về từ Intent và hiển thị dữ liệu đó trong các phần tử `TextView` mà bạn đã tạo trong nhiệm vụ trước.

1. Mở `MainActivity`.
2. Thêm một hằng số `public` ở đầu lớp để định nghĩa khóa cho một loại phản hồi mà bạn quan tâm:

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

3. Thêm hai biến riêng tư để chứa phần tử tiêu đề phản hồi và phần tử `TextView` phản hồi:

```
private TextView mReplyHeadTextView;
```

```
no usages
```

```
private TextView mReplyTextView;
```

4. Trong phương thức `onCreate()`, sử dụng `findViewById()` để lấy tham chiếu từ giao diện đến phần tử tiêu đề phản hồi và phần tử `TextView` phản hồi. Gán các đối tượng view này cho các biến riêng tư:

```
mReplyHeadTextView = findViewById(R.id.text_header_reply);
```

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

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

```

protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 EdgeToEdge.enable($this$enableEdgeToEdge: this);
 setContentView(R.layout.activity_main);
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
 return insets;
 });
 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 từ startActivity() thành startActivityForResult(), và bao gồm khóa TEXT\_REQUEST làm đối số:

startActivityForResult(intent, TEXT\_REQUEST);

- Ghi đè phương thức callback onActivityResult() với chữ ký sau:**

```

public void onActivityResult(int requestCode, int resultCode, Intent data)

```

Ba tham số trong onActivityResult() chưa tất cả thông tin mà bạn cần để xử lý dữ liệu trả về: requestCode là mã yêu cầu mà bạn đã đặt khi khởi động Activity với startActivityForResult(), resultCode là mã kết quả được đặt trong Activity đã được khởi động (thường là một trong RESULT\_OK hoặc RESULT\_CANCELED), và Intent data chứa dữ liệu trả về từ Activity đã được khởi động.

- Bên trong onActivityResult(), gọi super.onActivityResult():**

```

super.onActivityResult(requestCode, resultCode, data);

```

- Thêm mã để kiểm tra TEXT\_REQUEST để đảm bảo bạn xử lý đúng kết quả của Intent, trong trường hợp có nhiều kết quả. Cũng kiểm tra RESULT\_OK để đảm bảo rằng yêu cầu đã thành công:**

```
if (requestCode == TEXT_REQUEST)
{
 if (resultCode == RESULT_OK)
 {
 ...
 }
}
```

Lớp Activity định nghĩa các mã kết quả. Mã có thể là RESULT\_OK (yêu cầu thành công), RESULT\_CANCELED (người dùng hủy bỏ thao tác), hoặc RESULT\_FIRST\_USER (dành cho mã kết quả do người dùng tự định nghĩa).

9. **Bên trong khối if con, lấy Intent extra từ phản hồi Intent (data):** Ở đây, khóa cho extra là hằng số EXTRA\_REPLY từ SecondActivity. Bạn sẽ sử dụng data.getStringExtra() để lấy giá trị trả về:

```
String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);
```

10. **Đặt độ hiển thị của tiêu đề phản hồi (reply header) thành true:**

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

11. **Đặt nội dung cho TextView phản hồi (reply TextView) với reply và đặt độ hiển thị của nó thành true:**

```
mReplyTextView.setText(reply);
```

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

**Phương thức onActivityResult() đầy đủ:**

```
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);
 }
 }
}
```

## 12. Chạy ứng dụng.

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

11:17



## TwoActivities

**Reply Received**

Thauhs sds



Hi

Send

## Bài học 2.2: Vòng đời và trạng thái của Activity

### Giới thiệu

Trong bài thực hành này, bạn sẽ 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 ra 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 của bạn (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 một Activity có một phương thức callback tương ứng: **onCreate()**, **onStart()**, **onPause()**, và nhiều hơn nữa. Khi một Activity thay đổi trạng thái, phương thức callback tương ứng sẽ được gọi. Bạn đã từng thấy một trong những phương thức này: **onCreate()**. Bằng cách ghi đè bất kỳ phương thức callback nào của vòng đời trong các lớp **Activity** của bạn, bạn có thể thay đổi hành vi mặc định của Activity để phản hồi các hành động của người dùng hoặc hệ thống.

Trạng thái của Activity cũng có thể thay đổi do các thay đổi cấu hình của thiết bị, ví dụ như khi người dùng xoay thiết bị từ chế độ dọc sang ngang. Khi những thay đổi cấu hình này xảy ra, Activity sẽ bị hủy và được tạo lại trong trạng thái mặc định, và người dùng có thể mất thông tin mà họ đã nhập vào Activity. Để tránh làm người dùng bối rối, điều quan trọng là bạn cần phát triển ứng dụng của mình để ngăn ngừa việc mất dữ liệu ngoài ý muốn. Trong phần sau của bài thực hành này, bạn sẽ thử nghiệm với các thay đổi cấu hình và học cách bảo toàn trạng thái của Activity để phản hồi những thay đổi cấu hình thiết bị và các sự kiện khác trong vòng đời của Activity.

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

### Những gì bạn cần biết trước

Bạn cần có khả năng:

- Tạo và chạy một dự án ứng dụng trong Android Studio.
- Thêm các câu lệnh ghi nhật ký (log statements) vào ứng dụng của bạn và xem các nhật ký đó trong bảng **Logcat**.

- Hiểu và làm việc với **Activity** và **Intent**, đồng thời thoải mái khi 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 lại và bị hủy.
- Các phương thức callback của vòng đời liên quan đến những thay đổi của **Activity**.
- Tác động của các hành động (chẳng hạn như thay đổi cấu hình) có thể dẫn đến các sự kiện trong vòng đời **Activity**.
- Cách giữ 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 vòng đời khác nhau của **Activity**, bao gồm cả các câu lệnh ghi nhật ký.
- Quan sát các thay đổi trạng thái khi ứng dụng của bạn chạy và khi bạn tương tác với từng **Activity** trong ứng dụng.
- Sửa đổi ứng dụng của bạn để giữ trạng thái phiên bản của một **Activity** khi nó được tạo lại một cách không mong muốn do hành vi của người dùng hoặc thay đổi cấu hình trên thiết bị.

### Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ bổ sung vào ứng dụng **TwoActivities**. Ứng dụng sẽ trông và hoạt động gần giống như trong bài codelab trước. Nó chứa hai triển khai **Activity** và cung cấp cho người dùng khả năng gửi dữ liệu giữa chúng. Những thay đổi bạn thực hiện trong bài thực hành này sẽ không ảnh hưởng đến hành vi giao diện người dùng có thể thấy được của ứng dụng.

### Nhiệm vụ 1: Thêm các 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 vào **logcat** khi các phương thức này được gọi. Các thông báo log này sẽ cho phép bạn thấy khi nào vòng đời **Activity** thay đổi trạng thái và cách những thay đổi trạng thái vòng đời này ảnh hưởng đến ứng dụng khi nó chạy.

#### 1.1 (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 tạo một bản sao của dự án.

## 1.2 Thêm các callback vào MainActivity

1. Mở dự án **TwoActivities** trong Android Studio, sau đó mở **MainActivity** từ **Project > Android pane**.
2. Trong phương thức **onCreate()**, thêm các câu lệnh log sau đây:

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

3. Thêm một phương thức ghi đè cho callback **onStart()**, với một câu lệnh ghi nhật ký cho sự kiện đó:

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

Để tiết kiệm thời gian, 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 sẽ chèn một mẫu hoàn chỉnh cho các phương thức đó, bao gồm cả lời gọi bắt buộc đến lớp cha (superclass).

4. Sử dụng phương thức **onStart()** làm mẫu để triển khai các callback vòng đời **onPause()**, **onRestart()**, **onResume()**, **onStop()**, và **onDestroy()**.  
Tất cả các phương thức callback có chữ ký (signature) giống nhau (ngoại trừ tên). Nếu bạn Copy và Paste phương thức **onStart()** để tạo các phương thức callback khác, đừng quên cập nhật nội dung để gọi đúng phương thức trong lớp cha, và ghi nhật ký đúng phương thức.

1. Chạy ứng dụng của bạn.
2. Nhấp vào tab **Logcat** ở dưới cùng của Android Studio để hiển thị bảng Logcat. Bạn sẽ thấy ba thông báo nhật ký hiển thị ba trạng thái vòng đời mà Activity đã chuyển qua khi nó bắt đầu:

```
D onCreat
D onStart
D onResume
```

### 1.3 Triển khai các phương thức callback vòng đời trong SecondActivity

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

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

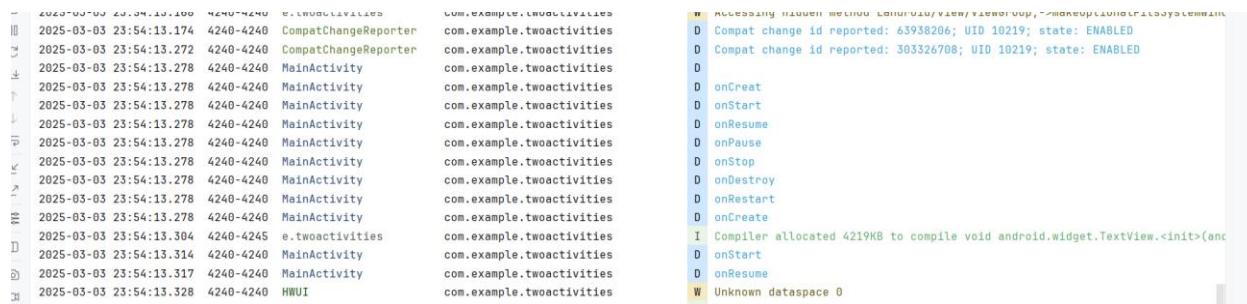
3. Thêm các phương thức callback vòng đời và câu lệnh log vào SecondActivity. (Bạn có thể sao chép và dán các phương thức callback từ MainActivity).
4. Thêm một câu lệnh log vào phương thức returnReply() ngay trước khi gọi finish().

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

### 1.4 Quan sát log khi ứng dụng chạy

1. Chạy ứng dụng của bạn.
2. Nhấp vào tab **Logcat** ở dưới cùng trong Android Studio để hiển thị cửa sổ Logcat.
3. Nhập **Activity** vào ô tìm kiếm.

Logcat của Android có thể rất dài và lộn xộn. Vì biến **LOG\_TAG** trong mỗi lớp chứa từ **MainActivity** hoặc **SecondActivity**, từ khóa này giúp bạn lọc log chỉ hiển thị những thông tin bạn quan tâm.



Thử nghiệm sử dụng ứng dụng của bạn và ghi chú lại các sự kiện vòng đời xảy ra khi thực hiện các hành động khác nhau. Cụ thể, thử các điều sau:

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

**MẸO:** Nếu bạn đang chạy ứng dụng trong giả lập, bạn có thể mô phỏng việc xoay màn hình bằng cách nhấn **Control+F11** hoặc **Control+Function+F11**.

### Mã giải pháp cho tác vụ 1

Dưới đây là đoạn mã thêm vào **MainActivity**, nhưng không phải là toàn bộ lớp.

Phương thức `onCreate()`:

```
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 EdgeToEdge.enable($this$enableEdgeToEdge: this);
 setContentView(R.layout.activity_main);
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
 return insets;
 });
 mMessageEditText = findViewById(R.id.editText_main);
 mReplyHeadTextView = findViewById(R.id.text_header_reply);
 mReplyTextView = findViewById(R.id.text_message_reply);
 Log.d(LOG_TAG, msg: "");
 Log.d(LOG_TAG, msg: "onCreate");
 Log.d(LOG_TAG, msg: "onStart");
 Log.d(LOG_TAG, msg: "onResume");
 Log.d(LOG_TAG, msg: "onPause");
 Log.d(LOG_TAG, msg: "onStop");
 Log.d(LOG_TAG, msg: "onDestroy");
 Log.d(LOG_TAG, msg: "onRestart");
 Log.d(LOG_TAG, msg: "onCreate");
```

Các phương thức vòng đời khác:

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

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

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

1 usage
@Override
protected void onRestart() {
 super.onRestart();
 Log.d(LOG_TAG, msg: "onRestart");
}
```

## SecondActivity

Dưới đây là đoạn mã thêm vào **SecondActivity**, nhưng không phải là 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, msg: "End SecondActivity");
 finish();
}
```

Các phương thức vòng đời khác:  
Giống như đối với **MainActivity**, ở trên.

## Tác vụ 2: Lưu và phục hồi trạng thái của Activity

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

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

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

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

Dữ liệu bạn lưu trong trạng thái phiên bản chỉ áp dụng cho chính phiên bản **Activity** này trong phiên làm việc hiện tại của ứng dụng. Khi bạn dừng và khởi động lại một phiên làm việc mới, trạng thái phiên bản **Activity** bị mất và **Activity** trở lại giao diện mặc định. Nếu bạn cần lưu trữ dữ liệu người dùng giữa các phiên làm việc của ứng dụng, hãy sử dụng **shared preferences** hoặc cơ sở dữ liệu. Bạn sẽ học về cả hai trong một bài thực hành sau.

## 2.1 Lưu trạng thái phiên bản Activity với onSaveInstanceState()

Bạn có thể đã nhận thấy rằng việc xoay thiết bị không ảnh hưởng đến trạng thái của **SecondActivity** chút nào. Điều này là vì giao diện và trạng thái của **SecondActivity** được tạo ra từ giao diện và **Intent** kích hoạt nó. Ngay cả khi **Activity** bị tái tạo, **Intent** vẫn còn và dữ liệu trong **Intent** đó vẫn được sử dụng mỗi khi phương thức **onCreate()** trong **SecondActivity** được gọi.

Ngoài ra, bạn có thể nhận thấy rằng trong mỗi **Activity**, bất kỳ văn bản nào bạn đã nhập vào các phần tử **EditText** của tin nhắn hoặc trả lời đều được giữ lại ngay cả khi thiết bị bị xoay. Điều này là vì thông tin trạng thái của một số phần tử **View** trong giao diện của bạn được tự động lưu trữ qua các thay đổi cấu hình, và giá trị hiện tại của **EditText** là một trong những trường hợp đó.

Vì vậy, trạng thái **Activity** mà bạn quan tâm chỉ là các phần tử **TextView** cho tiêu đề trả lời và văn bản trả lời trong **MainActivity**. Cả hai phần tử **TextView** này đều vô hình mặc định; chúng chỉ hiển thị khi bạn gửi tin nhắn trở lại **MainActivity** từ **SecondActivity**.

Trong tác vụ này, bạn sẽ thêm mã để bảo vệ trạng thái phiên bản của hai phần tử **TextView** này bằng cách sử dụng **onSaveInstanceState()**.

1. Mở **MainActivity**.
2. Thêm cấu trúc triển khai của **onSaveInstanceState()** vào **Activity**, hoặc sử dụng **Code > Override Methods** để chèn một phương thức ghi đè cấu trúc.

**@Override**

```
protected void onSaveInstanceState(@NotNull Bundle outState) {
 super.onSaveInstanceState(outState);
}
```

3. Kiểm tra xem tiêu đề có đang hiển thị không, và nếu có, hãy đưa trạng thái hiển thị đó vào trong **Bundle** trạng thái bằng phương thức **putBoolean()** với khóa "**reply\_visible**":

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

Hãy nhớ rằng tiêu đề và văn bản trả lời sẽ được đánh dấu là vô hình cho đến khi có một câu trả lời từ **SecondActivity**. Nếu tiêu đề đang hiển thị, điều này có nghĩa là có dữ liệu trả lời cần được lưu lại.

Lưu ý rằng chúng ta chỉ quan tâm đến trạng thái hiển thị đó — văn bản thực tế của tiêu đề không cần phải lưu lại, vì văn bản đó không bao giờ thay đổi.

4. Trong cùng một kiểm tra đó, thêm văn bản trả lời vào trong **Bundle**:

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

Ở đây, **mReplyTextView** là phần tử **TextView** chứa văn bản trả lời. Phương thức **putString()** sẽ lưu lại văn bản trả lời vào trong **Bundle**.

Nếu tiêu đề đang hiển thị, bạn có thể giả định rằng tin nhắn trả lời cũng đang hiển thị. Bạn không cần phải kiểm tra hay lưu trạng thái hiển thị hiện tại của tin nhắn trả lời. Chỉ cần lưu lại văn bản thực tế của tin nhắn vào trong **Bundle** trạng thái với khóa "**reply\_text**".

Bạn chỉ lưu trạng thái của những phần tử **View** có thể thay đổi sau khi **Activity** được tạo. Các phần tử **View** khác trong ứng dụng của bạn (như **EditText**, **Button**) có thể được tái tạo từ giao diện mặc định bất kỳ 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**.

## 2.2 Khôi phục trạng thái phiên bản Activity trong onCreate()

Sau khi bạn đã lưu trạng thái phiên bản **Activity**, bạn cũng cần phải phục hồi nó khi **Activity** được tái tạo. Bạn có thể làm điều này trong **onCreate()**, hoặc bằng cách triển khai phương thức **onRestoreInstanceState()**, phương thức này sẽ được gọi sau **onStart()** sau khi **Activity** được tạo.

Hầu hết thời gian, việc phục hồi trạng thái **Activity** trong **onCreate()** là lựa chọn tốt hơn, để đảm bảo rằng giao diện người dùng, bao gồm cả trạng thái, có sẵn càng sớm càng tốt. Tuy nhiên, đôi khi cũng tiện lợi khi làm điều này trong **onRestoreInstanceState()** sau khi tất cả việc khởi tạo đã hoàn tất, hoặc để cho các lớp con quyết định có sử dụng triển khai mặc định của bạn hay không.

- Trong phương thức **onCreate()**, sau khi các biến **View** được khởi tạo với **findViewById()**, thêm một 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 **Bundle** trạng thái vào **onCreate()** như là đối số duy nhất. Lần đầu tiên **onCreate()** được gọi và ứng dụng của bạn khởi động, **Bundle** sẽ là **null** — không có trạng thái tồn tại khi ứng dụng của bạn khởi động lần đầu tiên. Những lần gọi sau của **onCreate()** sẽ có một **Bundle** được điền dữ liệu từ những gì bạn đã lưu trong **onSaveInstanceState()**.

2. Bên trong kiểm tra đó, lấy trạng thái hiển thị hiện tại (đúng hoặc sai) từ **Bundle** với khóa "reply\_visible".

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

3. Thêm một kiểm tra bên dưới dòng trước đó cho biến **isVisible**.

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

Nếu có một khóa **reply\_visible** trong **Bundle** trạng thái (và do đó **isVisible** là true), bạn sẽ cần phải khôi phục trạng thái.

4. Bên trong kiểm tra **isVisible**, làm cho tiêu đề hiển thị.

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

5. Lấy văn bản trả lời từ **Bundle** với khóa "reply\_text", và đặt **TextView** trả lời để hiển thị chuỗi đó.

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

6. Làm cho **TextView** trả lời hiển thị nữa:

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

7. Chạy ứng dụng. Thủ 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 **Activity** được tái tạo.

Mã giải pháp cho nhiệm vụ 2

## MainActivity

Các đoạn mã sau đây hiển thị mã đã thêm vào **MainActivity**, nhưng không phải là toàn bộ lớp.

Phương thức **onSaveInstanceState()**:

```

@Override
protected void onSaveInstanceState(@NonNull Bundle outState) {
 super.onSaveInstanceState(outState);
 if(mReplyHeadTextView.getVisibility() == View.VISIBLE)
 {

 outState.putBoolean("reply_visible", true);
 outState.putString("reply_text", mReplyTextView.getText().toString());
 }
}

```

Phuong thuc onCreate():

```

protected void onSaveInstanceState(@NonNull Bundle outState) {
 super.onSaveInstanceState(outState);
 if(mReplyHeadTextView.getVisibility() == View.VISIBLE)
 {

 outState.putBoolean("reply_visible", true);
 outState.putString("reply_text", mReplyTextView.getText().toString());
 }
}

```

@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;
});
mMessageEditText = findViewById(R.id.editText_main);
mReplyHeadTextView = findViewById(R.id.text_header_reply);

```

```

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

```

Dự án hoàn chỉnh:

Dự án Android Studio: TwoActivitiesLifecycle

**Thách thức lập trình Lưu ý:** Tất cả các thách thức lập trình là tùy chọn và không phải là yêu cầu bắt buộc cho các bài học sau.

**Thách thức:** Tạo một ứng dụng danh sách mua sắm đơn giản với một hoạt động chính cho danh sách mà người dùng đang xây dựng, và một hoạt động thứ hai cho danh sách các mặt hàng mua sắm thông thường.

- Hoạt động chính nên chứa danh sách để xây dựng, bao gồm mười phần tử TextView trống.
- Một nút **Thêm mục** trên hoạt động chính sẽ mở một hoạt động thứ hai chứa danh sách các mặt hàng mua sắm thông thường (Phô mai, Gạo, Táo, v.v.). Sử dụng các phần tử Button để hiển thị các mục.
- Việc chọn một mục sẽ trả người dùng về hoạt động chính và cập nhật một TextView trống để bao gồm mục đã chọn.

Sử dụng một Intent để truyền thông tin từ một hoạt động sang hoạt động khác. Đảm bảo rằng trạng thái hiện tại của danh sách mua sắm được lưu lại khi người dùng xoay thiết bị.

### Tóm tắt

- Vòng đời của Activity là một tập hợp các trạng thái mà một Activity chuyển qua, bắt đầu khi nó được tạo ra và kết thúc khi hệ thống Android thu hồi tài nguyên cho Activity đó.
- Khi người dùng điều hướng từ một Activity sang một Activity khác, và trong và ngoài ứng dụng của bạn, mỗi Activity di chuyển giữa các trạng thái trong vòng đời Activity.
- Mỗi trạng thái trong vòng đời Activity có một phương thức callback tương ứng mà bạn có thể ghi đè trong lớp Activity của bạn.
- Các phương thức vòng đời bao gồm: onCreate(), onStart(), onPause(),

onRestart(), onResume(), onStop(), onDestroy().

- Việc ghi đè một phương thức callback của vòng đời cho phép bạn thêm hành vi xảy ra khi Activity của bạn chuyển sang trạng thái đó.
- Bạn có thể thêm các phương thức ghi đè khung vào các lớp của mình trong Android Studio bằng cách vào **Code > Override**.

Dưới đây là bản dịch tiếng Việt của đoạn mô tả:

- Các thay đổi cấu hình thiết bị như xoay màn hình dẫn đến Activity bị hủy và tái tạo lại như thế nào là mới.
- Một phần trạng thái của Activity được lưu lại khi có sự thay đổi cấu hình, bao gồm các giá trị hiện tại của các phần tử EditText. Đối với tất cả các dữ liệu khác, bạn phải tự lưu trữ dữ liệu đó.
- Lưu trạng thái của Activity trong phương thức onSaveInstanceState().
- Dữ liệu trạng thái của instance được lưu trữ dưới dạng các cặp khóa/giá trị đơn giản trong một Bundle. Sử dụng các phương thức của Bundle để đưa dữ liệu vào và lấy dữ liệu ra khỏi Bundle.
- Khôi phục trạng thái của instance trong onCreate(), đó là cách được ưu tiên, hoặc onRestoreInstanceState().

### **Khái niệm liên quan**

Tài liệu về khái niệm liên quan có trong mục 2.2: Vòng đời và trạng thái của Activity.

### **Tìm hiểu thêm**

Tài liệu của Android Studio:

- Làm quen với Android Studio

Tài liệu của nhà phát triển Android:

- Các nguyên lý cơ bản của ứng dụng
- Hoạt động (Activities)
- Hiểu về vòng đời của Activity
- Intents và Bộ lọc Intent
- Xử lý các thay đổi cấu hình
- Activity
- Intent

### **Bài tập về nhà**

Xây dựng và chạy một ứng dụng

1. Tạo một ứng dụng với một layout chứa một TextView đếm số, một nút Button để tăng giá trị của bộ đếm, và một EditText. Xem ảnh chụp màn hình dưới đây làm ví dụ. Bạn không cần phải sao chép chính xác layout.
2. Thêm một trình xử lý sự kiện khi nhấn nút Button để tăng bộ đếm.

3. Chạy ứng dụng và tăng bộ đếm. Nhập một số văn bản vào EditText.
4. Xoay thiết bị. Lưu ý rằng bộ đếm bị đặt lại, nhưng EditText thì không.
5. Triển khai phương thức onSaveInstanceState() để lưu lại trạng thái hiện tại của ứng dụng.
6. Cập nhật phương thức onCreate() để khôi phục trạng thái của ứng dụng.
7. Đảm bảo rằng khi bạn xoay thiết bị, trạng thái của ứng dụng vẫn được bảo lưu.

### Câu hỏi 1

Nếu bạn chạy ứng dụng bài tập trước khi triển khai phương thức onSaveInstanceState(), điều gì sẽ xảy ra khi bạn xoay thiết bị? Chọn một đáp án:

- EditText không còn chứa văn bản bạn đã nhập, nhưng bộ đếm vẫn được bảo lưu.
- Bộ đếm bị đặt lại về 0, và EditText không còn chứa văn bản bạn đã nhập.
- Bộ đếm bị đặt lại về 0, nhưng nội dung của EditText được bảo lưu.
- Bộ đếm và nội dung của EditText đều được bảo lưu.

### Câu hỏi 2

Các phương thức vòng đời của Activity nào được gọi khi có thay đổi cấu hình thiết bị (chẳng hạn như xoay màn hình)? Chọn một đáp án:

- Android ngay lập tức tắt Activity của bạn bằng cách gọi onStop(). Mã của bạn phải khởi động lại Activity.
- Android tắt Activity của bạn bằng cách gọi onPause(), onStop(), và onDestroy(). Mã của bạn phải khởi động lại Activity.
- Android tắt Activity của bạn bằng cách gọi onPause(), onStop(), và onDestroy(), sau đó khởi động lại Activity và gọi onCreate(), onStart(), và onResume().
- Android ngay lập tức gọi onResume().

### Câu hỏi 3

Khi nào phương thức onSaveInstanceState() được gọi trong vòng đời của Activity? Chọn một đáp án:

- onSaveInstanceState() được gọi trước phương thức onStop().
- onSaveInstanceState() được gọi trước phương thức onResume().
- onSaveInstanceState() được gọi trước phương thức onCreate().
- onSaveInstanceState() được gọi trước phương thức onDestroy().

## Câu hỏi 4

Các phương thức vòng đời của Activity nào là tốt nhất để lưu dữ liệu trước khi Activity kết thúc hoặc bị hủy? Chọn một đáp án:

- onPause() hoặc onStop()
- onResume() hoặc onCreate()
- onDestroy()
- onStart() hoặc onRestart()

## Bài học 2.3: Intents gián tiếp

### Giới thiệu

Trong một phần trước, bạn đã học về **explicit intents** (intents rõ ràng). Trong một explicit intent, bạn thực hiện một hoạt động trong ứng dụng của bạn, hoặc trong một ứng dụng khác, bằng cách gửi một intent với tên lớp đầy đủ của hoạt động. Trong bài học này, bạn sẽ tìm hiểu thêm về **implicit intents** (intents gián tiếp) và cách sử dụng chúng để thực hiện các hoạt động.

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

Ngược lại, hoạt động của bạn có thể khai báo một hoặc nhiều **intent filters** trong tệp AndroidManifest.xml để thông báo rằng hoạt động đó có thể chấp nhận implicit intents và định nghĩa các loại intents mà hoạt động sẽ chấp nhận.

Để khớp yêu cầu của bạn với một ứng dụng đã cài đặt trên thiết bị, hệ thống Android sẽ so khớp implicit intent của bạn với một hoạt động có intent filters chỉ ra rằng chúng có thể thực hiện hành động đó. Nếu có nhiều ứng dụng khớp, người dùng sẽ được hiển thị một trình chọn ứng dụng cho phép họ chọn ứng dụng mà họ muốn sử dụng để xử lý intent.

Trong bài thực hành này, bạn sẽ xây dựng một ứng dụng gửi một 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 mảnh thông tin cho những 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 tạo một implicit intent để chia sẻ dữ liệu một cách dễ dàng.

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

### Những gì bạn đã biết

Bạn sẽ có khả năng:

- Sử dụng trình chỉnh sửa layout để thay đổi một layout.
- Chính sửa mã XML của một layout.
- Thêm một Button và một trình xử lý sự kiện nhấn nút.
- Tạo và sử dụng một Activity.
- Tạo và gửi một Intent giữa hai Activity.

### Những gì bạn sẽ học

- Cách tạo một Implicit Intent, và sử dụng các hành động và danh mục của nó.
- Cách sử dụng lớp trợ giúp ShareCompat.IntentBuilder để tạo một Implicit Intent cho việc chia sẻ dữ liệu.
- Cách quảng cáo ứng dụng của bạn có thể chấp nhận một Implicit Intent bằng cách khai báo các Intent filters trong tệp AndroidManifest.xml.

### Những gì bạn sẽ làm

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

### Tổng quan về ứng dụng

Trong phần này, bạn sẽ tạo một ứng dụng mới với một Activity và ba tùy chọn hành động: mở một trang web, mở một vị trí trên bản đồ, và chia sẻ một đoạn văn bản. Tất cả các trường văn bản đều có thể chỉnh sửa (EditText), nhưng chứa các giá trị mặc định.

#### Nhiệm vụ 1: Tạo dự án và layout

Trong bài tập này, bạn sẽ tạo một dự án và ứng dụng mới có tên là **Implicit Intents**, với một layout mới.

##### 1.1 Tạo dự án

1. Mở Android Studio và tạo một dự án Android Studio mới. Đặt tên ứng dụng của bạn là **Implicit Intents**.
2. Chọn **Empty Activity** làm mẫu cho dự án. Nhấn **Next**.
3. Chấp nhận tên **Activity** mặc định là **MainActivity**. Đảm bảo ô **Generate Layout file** được chọn. Nhấn **Finish**.

## 1.2 Tạo layout

Trong nhiệm vụ này, bạn sẽ tạo layout cho ứng dụng. Sử dụng một **LinearLayout**, ba phần tử **Button**, và ba phần tử **EditText**, như sau:

1. Mở ứng dụng > res > values > strings.xml trong bảng **Project > Android**, và thêm các tài nguyên chuỗi sau:

```
<string name="button_uri">Open Website</string>
<string name="edittex_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>
```

2. Mở res > layout > activity\_main.xml trong bảng **Project > Android**. Nhấp vào tab **Text** để chuyển sang mã XML.
3. Thay đổi **android.support.constraint.ConstraintLayout** thành **LinearLayout**, như bạn đã học trong bài thực hành trước.
4. Thêm thuộc tính **android:orientation** với giá trị "vertical". Thêm thuộc tính **android:padding** với giá trị "16dp".

```
<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">
```

5. Xóa **TextView** hiển thị "Hello World".
6. Thêm một bộ các phần tử giao diện người dùng cho nút **Open Website**. Bạn cần một phần tử **EditText** và một phần tử **Button**. Sử dụng các giá trị thuộc tính sau:

<b>EditText attribute</b>	<b>Value</b>
android:id	"@+id/website_edittext"
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:text	"@string/edittext_uri"
<b>Button attribute</b>	<b>Value</b>
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"

Giá trị cho thuộc tính android:onClick sẽ vẫn được gạch chân đỏ cho đến khi bạn định nghĩa phương thức callback trong một nhiệm vụ sau.

7. **Thêm một bộ các phần tử giao diện người dùng (EditText và Button) vào layout cho nút Open Location.** Sử dụng các thuộc tính giống như trong bước trước, nhưng sửa đổi chúng như sau (Bạn có thể sao chép các giá trị từ nút Open Website và chỉnh sửa chúng)

<b>EditText attribute</b>	<b>Value</b>
android:id	"@+id/location_edittext"
android:text	"@string/edittext_loc"
<b>Button attribute</b>	<b>Value</b>
android:id	"@+id/open_location_button"
android:text	"@string/button_loc"
android:onClick	"openLocation"

Giá trị cho thuộc tính android:onClick sẽ vẫn được gạch chân đỏ cho đến khi bạn định nghĩa phương thức callback trong một nhiệm vụ sau.

8. **Thêm một bộ các phần tử giao diện người dùng** (EditText và Button) vào layout cho nút **Share This**. Sử dụng các thuộc tính như dưới đây. (Bạn có thể sao chép các giá trị từ nút **Open Website** và chỉnh sửa chúng.)

<b>EditText attribute</b>	<b>Value</b>
android:id	"@+id/share_edittext"
android:text	"@string/edittext_share"
<b>Button attribute</b>	<b>Value</b>
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 của bạn sẽ trông giống như sau. Các giá trị cho thuộc tính android:onClick sẽ vẫn được gạch chân đỏ cho đến khi bạn định nghĩa các phương thức callback trong một nhiệm vụ sau:

```
<?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="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/edittex_loc" />

 <Button
 android:id="@+id/open_location_button"
 android:layout_width="match_parent"
 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="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/edittext_share" />

 <Button
 android:id="@+id/share_text_button"
 android:layout_width="match_parent"
 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 nhiệm vụ này, bạn triển khai phương thức xử lý sự kiện nhấn nút cho nút đầu tiên trong layout, **Open Website**. Hành động này sử dụng một **Implicit Intent** để gửi URI đã cho đến một **Activity** có thể xử lý **Implicit Intent** đó (chẳng hạn như trình duyệt web).

### 2.1 Định nghĩa openWebsite()

1. Nhấp vào "openWebsite" trong mã XML activity\_main.xml.
2. Nhấn Alt+Enter (hoặc Option+Enter trên Mac) và chọn **Create 'openWebsite(View)' in 'MainActivity'**.

Tệp **MainActivity** sẽ mở, và Android Studio sẽ tạo một phương thức khung cho trình xử lý **openWebsite()**.

```
public void openWebsite(View view) {
}
```

3. Trong **MainActivity**, thêm một biến private ở đầu lớp để lưu đối tượng **EditText** cho URI của trang web.

```
private EditText mWebsiteEditText;
```

4. Trong phương thức **onCreate()** của **MainActivity**, sử dụng **findViewById()** để lấy tham chiếu đến đối tượng **EditText** và gán nó cho biến private đó:

```
mWebsiteEditText = findViewById(R.id.website_edittext);
```

### 2.2 Thêm mã cho openWebsite()

- 1.Thêm một câu lệnh vào phương thức **openWebsite()** mới để lấy giá trị chuỗi từ **EditText**:

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

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

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

3. Tạo một **Intent** mới với **Intent.ACTION\_VIEW** làm hành động và URI làm dữ liệu:

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

Trình tạo **Intent** này khác với cái bạn đã sử dụng để tạo một **explicit Intent**. Trong trình tạo trước, bạn chỉ định ngữ cảnh hiện tại và một thành phần cụ thể (lớp **Activity**) để gửi **Intent**. Trong trình tạo này, bạn chỉ định một hành động và dữ liệu cho hành động đó. Các hành động được định nghĩa bởi lớp **Intent** và có thể bao gồm **ACTION\_VIEW** (để xem dữ liệu đã cho), **ACTION\_EDIT** (để chỉnh sửa dữ liệu đã cho), hoặc **ACTION\_DIAL** (để gọi một số điện thoại). Trong trường hợp này, hành động là **ACTION\_VIEW** vì bạn muốn hiển thị trang web được chỉ định bởi URI trong biến **webpage**.

4. Sử dụng phương thức **resolveActivity()** và trình quản lý gói Android để tìm một **Activity** có thể xử lý **Implicit Intent** của bạn. Đảm bảo rằng yêu cầu đã được giải quyết thành công:

```
if (intent.resolveActivity(getApplicationContext()) != null) {
```

Yêu cầu này sẽ so khớp hành động **Intent** và dữ liệu 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.

5. Bên trong câu lệnh **if**, gọi **startActivity()** để gửi **Intent**:

```
startActivity(intent);
```

6. Thêm một khối **else** để in một thông báo Log nếu Intent không thể được giải quyết.

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

Phương thức openWebsite() bây giờ nên trông như sau. (Các chú thích đã 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(getApplicationContext()) != null) {
 startActivity(intent);
 }
 else {
 Log.d(tag: "ImplicitIntents", msg: "Can't handle this!");
 }
}
```

### Nhiệm vụ 3: Triển khai nút Mở Vị trí

Trong nhiệm vụ này, bạn sẽ triển khai phương thức xử lý sự kiện khi nhấp vào nút thứ hai trong giao diện người dùng, **Mở Vị trí**. Phương thức này gần như giống hệt với phương thức openWebsite(). Điểm khác biệt là việc sử dụng URI dạng geo để chỉ định một vị trí bản đồ. Bạn có thể sử dụng URI dạng geo với vĩ độ và kinh độ, hoặc sử dụng chuỗi truy vấn để chỉ định một vị trí chung. Trong ví dụ này, chúng tôi đã sử dụng cách thứ hai.

#### 3.1 Định nghĩa openLocation()

1. Nhấp vào "openLocation" trong mã XML của **activity\_main.xml**.
2. Nhấn **Alt+Enter** (hoặc **Option+Enter** trên Mac) và chọn **Create 'openLocation(View)' in MainActivity**.
  - o Android Studio sẽ tự động tạo một phương thức khung xương trong **MainActivity** dành cho trình xử lý openLocation().

```
public void openLocation(View view) {
 |
}
|
```

3. Thêm một biến riêng (private variable) ở đầu lớp **MainActivity** để lưu đối tượng **EditText** cho URI vị trí:

```
private EditText mLocationEditText;
```

- Trong phương thức **onCreate()**, sử dụng **findViewById()** để lấy tham chiếu đến đối tượng **EditText** và gán nó vào biến riêng:

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

### 3.2 Thêm mã vào openLocation()

- Trong phương thức mới **openLocation()**, thêm một lệnh để lấy giá trị chuỗi từ đối tượng **mLocationEditText**:

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

- Phân tích chuỗi đó thành một đối tượng **Uri** với một truy vấn tìm kiếm geo:

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

- Tạo một đối tượng **Intent** mới với **Intent.ACTION\_VIEW** làm hành động và **addressUri** làm dữ liệu:

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

- Giải quyết **Intent** và kiểm tra để đảm bảo rằng **Intent** được giải quyết thành công. Nếu có, gọi **startActivity()**, nếu không, ghi lại một thông báo lỗi:

```
if (intent.resolveActivity(getApplicationContext()) != null)
 startActivity(intent);
else
 Log.d(tag: "ImplicitIntents", msg: "Can't handle this intent!");
```

Khi hoàn thành, phương thức **openLocation()** đầy đủ sẽ trông như sau:

```

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

}

```

#### Nhiệm vụ 4: Triển khai nút Chia sẻ Văn bản này

Hành động chia sẻ là một cách đơn giản để người dùng chia sẻ nội dung trong ứng dụng của bạn lên mạng xã hội và các ứng dụng khác. Mặc dù bạn có thể tự xây dựng một hành động chia sẻ trong ứng dụng bằng cách sử dụng một **Intent** ngầm định, Android cung cấp lớp trợ giúp **ShareCompat.IntentBuilder** để việc triển khai chia sẻ trở nên dễ dàng hơn.

Bạn có thể sử dụng **ShareCompat.IntentBuilder** để tạo một **Intent** và khởi chạy trình chọn (chooser), cho phép người dùng chọn ứng dụng đích để chia sẻ.

Trong nhiệm vụ này, bạn sẽ triển khai chức năng chia sẻ một đoạn văn bản trong ô nhập văn bản (text edit), bằng cách sử dụng lớp **ShareCompat.IntentBuilder**.

##### 4.1 Định nghĩa shareText()

1. Nhập vào "**shareText**" trong mã XML của **activity\_main.xml**.
2. Nhấn **Alt+Enter** (hoặc **Option+Enter** trên Mac) và chọn **Create 'shareText(View)' in MainActivity**.
  - o Android Studio sẽ tự động tạo một phương thức khung xương trong **MainActivity** cho trình xử lý **shareText()**.

```

public void shareText(View view) {
}

```

3. Thêm một biến riêng (private variable) ở đầu lớp **MainActivity** để lưu đối tượng **EditText**:

```
private EditText mShareEditText;
```

- Trong phương thức **onCreate()**, sử dụng **findViewById()** để lấy tham chiếu đến đối tượng **EditText** và gán nó vào biến riêng:

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

## 4.2 Thêm mã vào shareText()

- Trong phương thức **shareText()**, thêm một lệnh để lấy giá trị chuỗi từ đối tượng **mShareTextEdit**:

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

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

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

- Gọi **ShareCompat.IntentBuilder** với các phương thức sau:

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

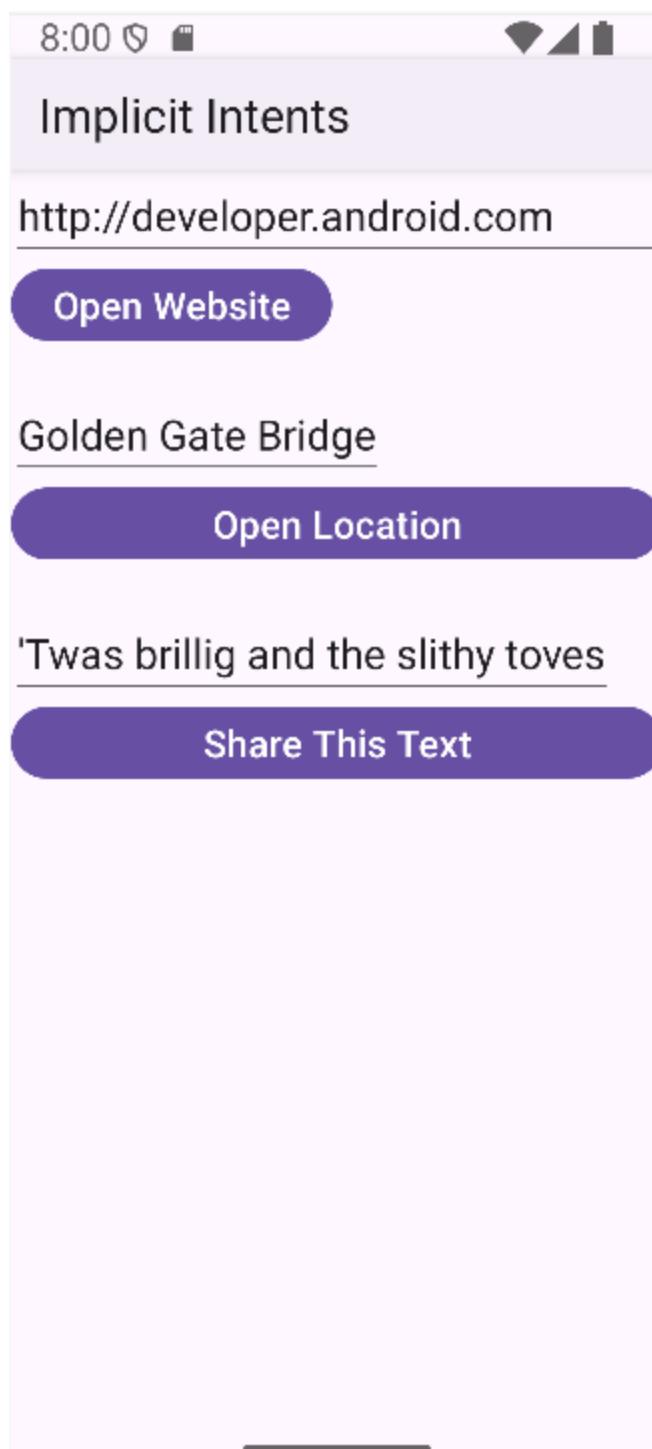
- Trích xuất giá trị của **.setChooserTitle** thành một tài nguyên chuỗi (string resource).

Lời gọi đến **ShareCompat.IntentBuilder** sử dụng các phương thức sau:

Method	Description
from()	The Activity that launches this share Intent (this).
setType()	The MIME type of the item to be shared.
setChooserTitle()	The title that appears on the system app chooser.
setText()	The actual text to be shared
startChooser()	Show the system app chooser and send the Intent.

Định dạng này, với tất cả các phương thức setter của builder được kết nối với nhau trong một câu lệnh, là một cách viết tắt 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() bây giờ nên trông như sau:

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



### 4.3 Chạy ứng dụng

1. Chạy ứng dụng.

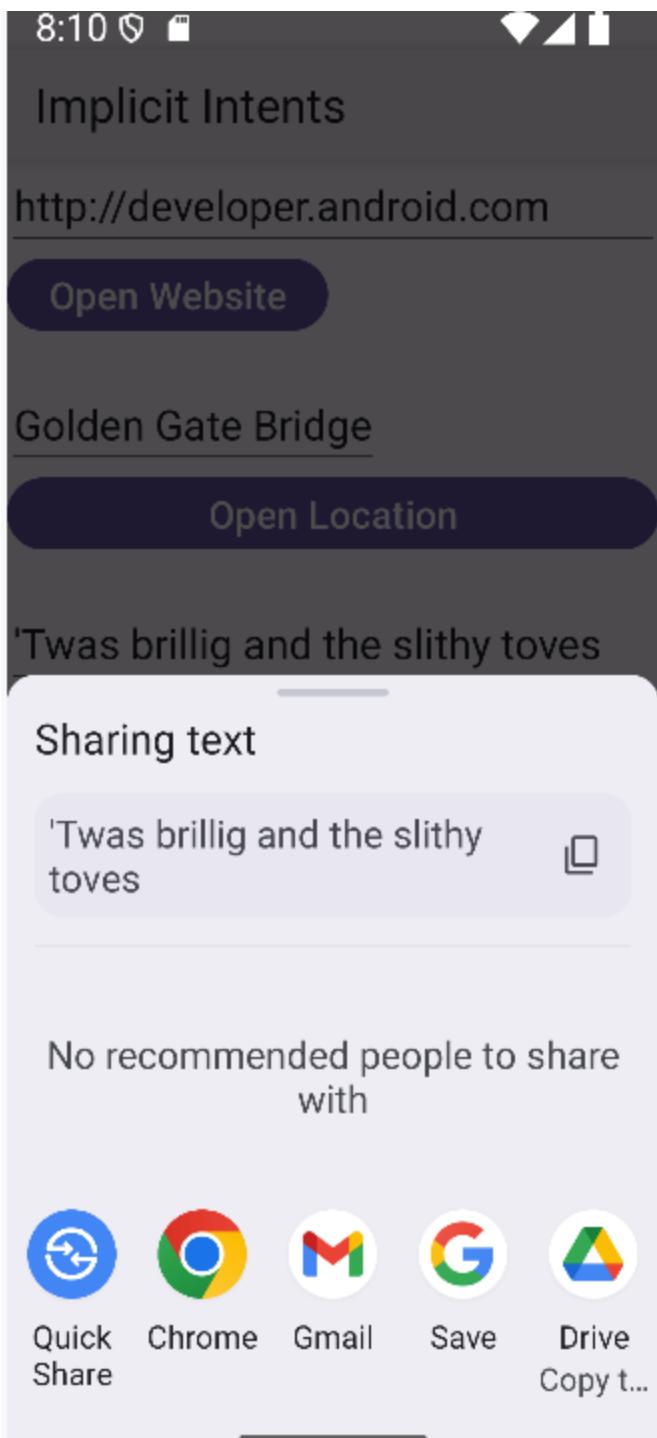
- Nhấp vào nút **Open Website** để mở trình duyệt với URL trang web được nhập trong ô **EditText** phía trên nút **Button**. Trình duyệt và trang web sẽ hiển thị như hình minh họa bên dưới.



- Nhấn nút **Open Location** để mở bản đồ với vị trí được nhập trong **EditText** phía trên nút. Bản đồ hiển thị vị trí sẽ xuất hiện như hình dưới đây.



4. Nhấn nút **Share This Text** để mở hộp thoại với các tùy chọn chia sẻ văn bản. Hộp thoại hiển thị các tùy chọn sẽ xuất hiện như hình dưới đây.



### Nhiệm vụ 5: Nhận một Intent ngầm định

Đến nay, bạn đã tạo một ứng dụng sử dụng một **Intent ngầm định** để khởi chạy **Activity** của ứng dụng khác. Trong nhiệm vụ này, bạn sẽ xem xét vấn đề từ góc độ ngược lại: cho phép một **Activity** trong ứng dụng của bạn phản hồi một **Intent ngầm định** được gửi từ ứng dụng khác.

**Một Activity trong ứng dụng của bạn luôn có thể được kích hoạt từ bên trong hoặc bên ngoài ứng dụng bằng một Intent tương minh (explicit Intent).**

Để cho phép một Activity nhận một Intent ngầm định (implicit Intent), bạn cần định nghĩa một **Intent filter** trong tệp **AndroidManifest.xml** của ứng dụng để chỉ ra loại Intent ngầm định nào mà Activity của bạn quan tâm xử lý.

Để kết nối yêu cầu của bạn với một ứng dụng cụ thể đã cài đặt trên thiết bị, hệ thống Android sẽ khớp Intent ngầm định của bạn với một Activity có Intent filter cho biết rằng Activity đó có thể thực hiện hành động đó. Nếu có nhiều ứng dụng cài đặt phù hợp, hệ thống sẽ hiển thị một bộ chọn ứng dụng (app chooser) để người dùng chọn ứng dụng họ muốn sử dụng để xử lý Intent.

Khi một ứng dụng trên thiết bị gửi một Intent ngầm định, hệ thống Android sẽ khớp hành động (action) và dữ liệu (data) của Intent đó với bất kỳ Activity nào có Intent filter phù hợp. Khi các Intent filter của một Activity khớp với Intent:

- Nếu chỉ có một Activity phù hợp, Android sẽ để Activity đó xử lý Intent.
- Nếu có nhiều Activity phù hợp, Android sẽ hiển thị một bộ chọn ứng dụng để người dùng chọn ứng dụng họ muốn thực thi hành động đó.

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

## 5.1 Tạo dự án và bối cảnh

1. Tạo một dự án Android Studio mới với tên ứng dụng là **Implicit Intents Receiver** và chọn mẫu dự án **Empty Activity**.
2. Chấp nhận tên **Activity** mặc định (**MainActivity**). Nhấn **Next**.
3. Đảm bảo hộp kiểm **Generate Layout file** đã được chọn. Nhấn **Finish**.
4. Mở tệp **activity\_main.xml**.
5. Trong **TextView** hiện có (hiển thị "Hello World"), xóa thuộc tính `android:text`. Mặc định **TextView** này sẽ không có văn bản, nhưng bạn sẽ thêm URI từ **Intent** trong phương thức `onCreate()`.
6. Giữ nguyên các thuộc tính `layout_constraint`, nhưng thêm các thuộc tính sau:

<b>Attribute</b>	<b>Value</b>
android:id	"@+id/text_uri_message"
android:textSize	"18sp"
android:textStyle	"bold"

## 5.2 Sửa đổi AndroidManifest.xml để thêm một Intent filter

1. Mở tệp **AndroidManifest.xml**.
2. 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 manifest mặc định trong dự án, cho biết rằng Activity này là điểm khởi đầu chính của ứng dụng (với Intent action là "android.intent.action.MAIN") và rằng Activity này sẽ xuất hiện dưới dạng một mục cấp cao nhất trong trình khởi chạy (category là "android.intent.category.LAUNCHER").

3. Thêm một thẻ `<intent-filter>` thứ hai bên trong thẻ `<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:host="developer.android.com"
 android:scheme="http" />

<intent-filter>

```

Các dòng này định nghĩa 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 (Type)	Giá trị (Value)	Khớp với (Matches)
action	"android.intent.action.VIEW"	Bất kỳ Intent nào có hành động là "view" (xem).
category	"android.intent.category.DEFAULT"	Bất kỳ Intent ngầm định nào. Category này phải được bao gồm để Activity của bạn nhận bất kỳ Intent ngầm định nào.
category	"android.intent.category.BROWSABLE"	Yêu cầu cho các liên kết duyệt web từ các trang web, email, hoặc nguồn khác.
data	android:scheme="http" android:host="developer.android.com"	Các URI chứa giao thức là http và tên máy chủ là developer.android.com .

Lưu ý rằng bộ lọc **data** có một giới hạn đối với cả loại liên kết mà nó sẽ chấp nhận và tên máy chủ cho các URI đó. Nếu bạn muốn bộ nhận (receiver) của mình có thể chấp nhận bất kỳ liên kết nào, bạn có thể bỏ qua phần tử <data>.

Phần **application** trong tệp **AndroidManifest.xml** bây giờ sẽ trông như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools">

 <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>
 <action android:name="android.intent.action.VIEW" />

 <category android:name="android.intent.category.DEFAULT" />
 <category android:name="android.intent.category.BROWSABLE" />

 <data
 android:host="developer.android.com"
 android:scheme="http" />

 <intent-filter>
 </intent-filter>
 </activity>
 </application>

</manifest>
```

### 5.3 Xử lý Intent

Trong phương thức onCreate() của Activity, xử lý Intent đến để lấy bất kỳ dữ liệu hoặc thông tin bổ sung (extras) nào mà nó chứa. Trong trường hợp này, Intent ngầm định đến sẽ có URI được lưu trữ trong dữ liệu của Intent.

1. Mở tệp **MainActivity**.
2. Trong phương thức onCreate(), lấy Intent đến được sử dụng để kích hoạt Activity:

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

3. Lấy dữ liệu từ Intent. Dữ liệu trong Intent luôn là một đối tượng URI:

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

4. Kiểm tra để đảm bảo rằng biến uri không phải là null. Nếu kiểm tra đó thành công, tạo một chuỗi từ đối tượng URI:

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

5. Trích xuất phần "URI: " vào một tài nguyên chuỗi (string resource) với tên **uri\_label**.
6. Trong cùng khái if, lấy TextView để hiển thị thông báo:

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

7. Cũng trong khái if, đặt nội dung văn bản cho TextView thành URI:

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

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

```
package com.example.implicitintentsreceiver;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.TextView;

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

public class MainActivity extends AppCompatActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 EdgeToEdge.enable(this);
 setContentView(R.layout.activity_main);
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
 Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
 v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
 return insets;
 });
 Intent intent = getIntent();
 Uri uri = intent.getData();
 if (uri != null) {
 String uri_string = "URI: " + uri.toString();
 TextView textView = findViewById(R.id.text_uri_message);
 textView.setText(uri_string);
 }
 }
}
```

## 5.4 Chạy cả hai ứng dụng

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

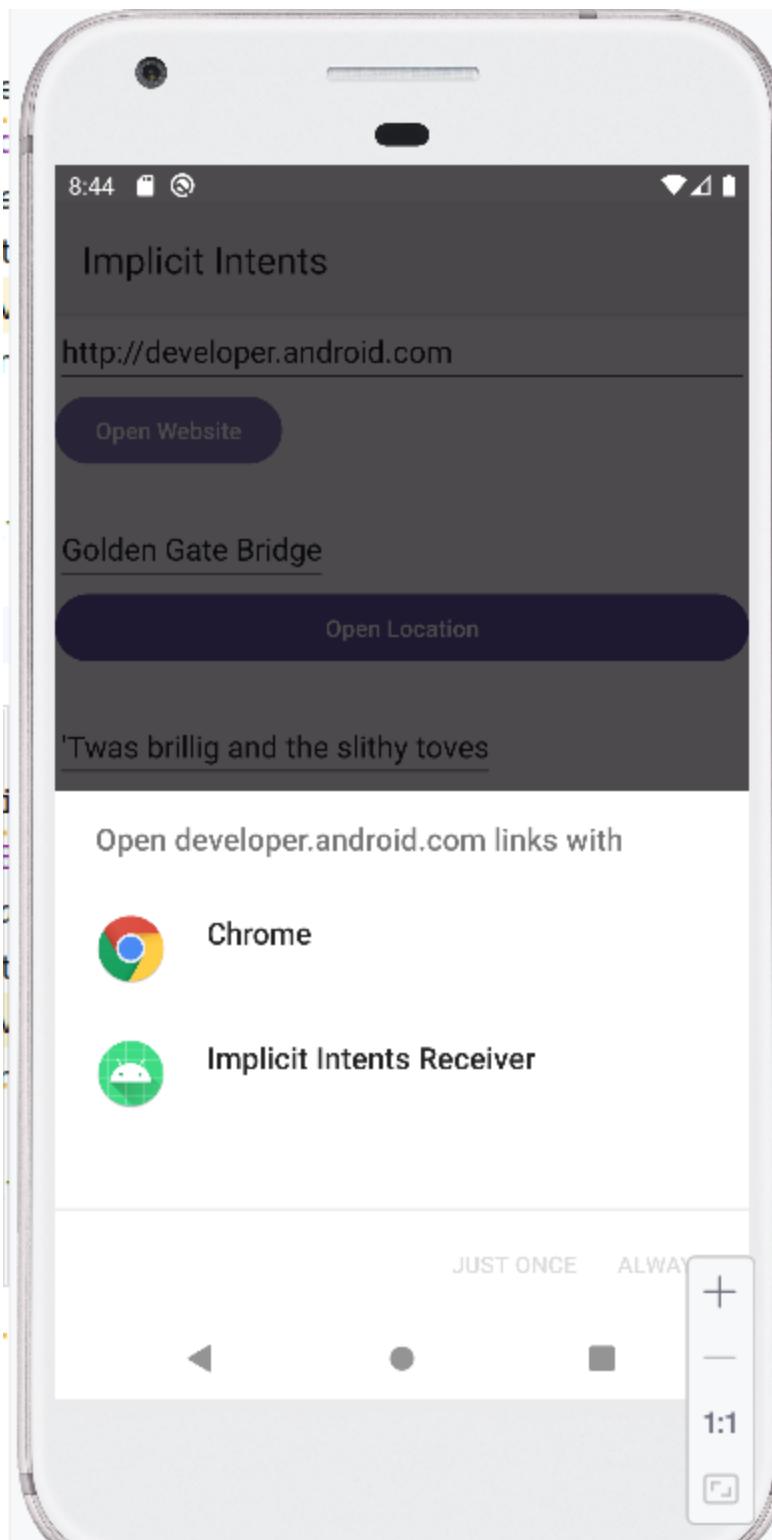
### 1. Chạy ứng dụng Implicit Intents Receiver.

Khi chạy ứng dụng này riêng lẻ, nó sẽ hiển thị một Activity trống mà không có văn bản. Điều này là do Activity được kích hoạt từ trình khởi chạy hệ thống, không phải bằng một Intent từ ứng dụng khác.

### 2. Chạy ứng dụng Implicit Intents và nhấn nút Open Website với URI mặc định.

Một hộp thoại chọn ứng dụng xuất hiện, hỏi bạn có muốn sử dụng trình duyệt mặc định (Chrome trong hình minh họa) hay ứng dụng **Implicit Intents Receiver**.

- Chọn **Implicit Intents Receiver**, sau đó nhấn **Just Once**.
- Ứng dụng **Implicit Intents Receiver** sẽ được khởi chạy, và thông báo sẽ hiển thị URI từ yêu cầu ban đầu



3. Nhấn nút Back và nhập một URI khác. Nhấn nút Open Website.

**Ứng dụng receiver có bộ lọc Intent rất hạn chế**, chỉ khớp với giao thức URI chính xác (<http://>) và host ([developer.android.com](http://developer.android.com)). Bất kỳ URI nào khác sẽ được mở bằng trình duyệt web mặc định.

## Mã giải pháp cho Task 5

Dự án Android Studio: **ImplicitIntentsReceiver**

### Thử thách lập trình

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

#### Thử thách:

Trong một thử thách thực hành trước, bạn đã tạo một ứng dụng danh sách mua sắm với một Activity để hiển thị danh sách và một Activity khác để chọn một mục.

Hãy thêm một **EditText** và một **Button** vào Activity danh sách mua sắm để định vị một cửa hàng cụ thể trên bản đồ.

### Tóm tắt

- **Intent ngầm định** cho phép bạn kích hoạt một Activity nếu bạn biết hành động nhưng không biết ứng dụng hoặc Activity cụ thể nào sẽ xử lý hành động đó.
- Một Activity có thể nhận Intent ngầm định phải định nghĩa bộ lọc Intent trong tệp **AndroidManifest.xml** để khớp với một hoặc nhiều hành động (action) và danh mục (category) của Intent.
- Hệ thống Android sẽ so khớp nội dung của Intent ngầm định và bộ lọc Intent của bất kỳ Activity có sẵn nào để xác định Activity nào sẽ được kích hoạt. Nếu có nhiều hơn một Activity có sẵn, hệ thống sẽ cung cấp một hộp chọn để người dùng chọn một.
- Lớp **ShareCompat.IntentBuilder** giúp việc tạo một Intent ngầm định để chia sẻ dữ liệu lên mạng xã hội hoặc email trở nên dễ dàng.

### Liên quan đến khái niệm

Tài liệu về khái niệm liên quan nằm trong **2.3: Implicit intents (Intent ngầm định)**.

### Tìm hiểu thêm

Tài liệu phát triển Android:

- **Application Fundamentals** (**Nguyên tắc cơ bản của ứng dụng**)
- **Activities** (**Hoạt động**)
- **Understand the Activity Lifecycle** (**Hiểu vòng đời của Activity**)
- **Intents and Intent Filters** (**Intent và Bộ lọc Intent**)
- **Allowing Other Apps to Start Your Activity** (**Cho phép ứng dụng khác khởi chạy Activity của bạn**)
- **Google Maps Intents for Android** (**Intent Google Maps cho Android**)
- **Activity**
- **Intent**
- **<intent-filter>**
- **<activity>**
- **Uri**
- **ShareCompat.IntentBuilder**

### Bài tập về nhà

#### Xây dựng và chạy một ứng dụng

Mở ứng dụng **ImplicitIntents** mà bạn đã tạo.

1. Thêm một nút khác ở cuối màn hình.
2. Khi nhấn vào nút này, hãy khởi chạy một ứng dụng camera để chụp ảnh.  
*(Bạn không cần trả lại ảnh về ứng dụng ban đầu).*

#### Lưu ý:

Nếu bạn sử dụng trình giả lập Android để kiểm tra camera, hãy mở cấu hình trình giả lập trong **Android AVD Manager**, chọn **Advanced Settings** và sau đó chọn **Emulated** cho cả camera trước và sau. Khởi động lại trình giả lập nếu cần thiết.

### Trả lời các câu hỏi

#### Câu hỏi 1

**Phương thức constructor nào bạn sử dụng để tạo một implicit Intent để mở ứng dụng camera?**

- new Intent()
- new Intent(Context context, Class<?> class)
- new Intent(String action, Uri uri)
- new Intent(String action)

---

## Câu hỏi 2

**Khi bạn tạo một đối tượng implicit Intent, điều nào sau đây là đúng?**

- Không cần chỉ định Activity hoặc thành phần cụ thể nào để mở.
  - Thêm một hành động Intent hoặc các danh mục Intent (hoặc cả hai).
  - Giải quyết Intent với hệ thống trước khi gọi startActivity() hoặc startActivityForResult().
  - Tất cả các đáp án trên.
- 

## Câu hỏi 3

**Hành động Intent nào bạn sử dụng để chụp ảnh bằng ứng dụng camera?**

- Intent takePicture = new Intent(Intent.ACTION\_VIEW);
  - Intent takePicture = new Intent(Intent.ACTION\_MAIN);
  - Intent takePicture = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);
  - Intent takePicture = new Intent(Intent.ACTION\_GET\_CONTENT);
- 

**Nộp ứng dụng của bạn để được chấm điểm**

**Hướng dẫn cho người chấm điểm:**

Kiểm tra rằng ứng dụng có các tính năng sau:

- Ứng dụng hiển thị nút **Take a Picture** ở cuối màn hình.
- Khi được nhấn, nút này mở ứng dụng camera trên thiết bị.
- Trước khi gửi intent, phương thức onClick() dành cho nút **Take a Picture** đảm bảo rằng trên thiết bị có sẵn một ứng dụng có thể xử lý intent này. Điều này được thực hiện bằng cách sử dụng các phương thức resolveActivity() và getPackageManager().

**Bài 3.1: Trình gõ lỗi**

**Giới thiệu**

Trong các bài thực hành trước, bạn đã sử dụng lớp Log để in thông tin ra nhật ký hệ thống, thông tin này sẽ xuất hiện ở bảng Logcat trong Android Studio khi ứng dụng của bạn chạy. Thêm các câu lệnh ghi log 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 (debugger) được tích hợp trong Android Studio.

Trong bài thực hành này, bạn sẽ học cách gỡ lỗi ứng dụng trên trình giả lập hoặc thiết bị thật, đặt và xem điểm dừng (breakpoints), từng bước thực hiện mã lệnh 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 giao diện (layout editor) để làm việc với các thành phần EditText và Button.
  - Xây dựng và chạy ứng dụng của bạn trong Android Studio, trên trình giả lập và thiết bị thật.
  - Đọc và phân tích dấu vết ngăn xếp (stack trace), bao gồm cách hiểu thứ tự **last on, first off**.
  - Thêm các câu lệnh ghi log và xem nhật ký hệ thống (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 thật.
  - Cách thực hiện từng bước mã lệnh khi ứng dụng chạy.
  - Cách đặt và tổ chức các điểm dừng (breakpoints).
  - Cách kiểm tra và thay đổi giá trị của các biến trong trình gỡ lỗi.
- 

## Những gì bạn sẽ làm

- Xây dựng ứng dụng SimpleCalc.
- Đặt và xem các điểm dừng trong mã nguồn của ứng dụng SimpleCalc.

- Từng bước thực hiện mã lệnh khi ứng dụng chạy.
  - Kiểm tra các biến và đánh giá các biểu thức.
  - Xác định và sửa lỗi trong ứng dụng mẫu.
- 

## Tổng quan ứng dụng

Ứng dụng SimpleCalc có hai thành phần EditText và bốn nút Button. Khi bạn nhập hai số và nhấn một nút, ứng dụng sẽ thực hiện phép tính tương ứng với nút đó và hiển thị kết quả.

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

Trong thực hành này, bạn sẽ không tự xây dựng ứng dụng SimpleCalc. Dự án hoàn chỉnh đã sẵn sàng 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.

---

#### 1.1 Tải xuống và mở dự án SimpleCalc

1. **Tải xuống SimpleCalc** và giải nén file.
2. Mở **Android Studio** và chọn **File > Open**.
3. Điều hướng đến thư mục chứa SimpleCalc, chọn thư mục đó và nhấn **OK**.  
Dự án SimpleCalc sẽ được xây dựng.
4. Mở **Project > Android** nếu nó chưa được mở.

**Cảnh báo:** Ứng dụng này có chứa lỗi mà bạn sẽ tìm và sửa chữa. Nếu bạn chạy ứng dụng trên thiết bị hoặc trình giả lập, có thể gặp phải hành vi bất thường, bao gồm cả sự cố ứng dụng.

---

#### 1.2 Khám phá bố cục (Layout)

1. Mở file **activity\_main.xml**.
2. Nhấp vào tab **Text** để xem mã XML.
3. Nhấp vào tab **Preview** để xem trước bố cục.

## **Khám phá mã XML bố cục và thiết kế, lưu ý các điểm sau:**

- Bố cục chứa hai phần tử **EditText** để nhập liệu, bốn nút **Button** cho các phép tính và một **TextView** để hiển thị kết quả.
- Mỗi nút tính toán (**Button**) có trình xử lý sự kiện nhấp riêng thông qua thuộc tính **android:onClick** (như `onAdd`, `onSub`, v.v.).
- **TextView** dùng để hiển thị kết quả không có bất kỳ văn bản nào theo mặc định.
- Hai phần tử **EditText** có thuộc tính **android:inputType** với giá trị "`numberDecimal`". Thuộc tính này cho biết **EditText** chỉ chấp nhận đầu vào là số. Bàn phím hiển thị trên màn hình sẽ chỉ chứa các số. Bạn sẽ tìm hiểu thêm về kiểu nhập liệu (**input types**) cho phần tử **EditText** trong các thực hành sau.

### **1.3 Khám phá mã nguồn ứng dụng**

1. Mở rộng thư mục **app > java** trong **Project > Android**. Ngoài lớp **MainActivity**, dự án này còn bao gồm lớp tiện ích **Calculator**.
2. Mở **Calculator** và xem xét mã nguồn. 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 thực hiện phép toán đều có phạm vi **public**.
3. Mở **MainActivity** và xem xét mã nguồn 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 riêng tư `compute()`, với tên phép toán là một trong các giá trị của `Calculator.Operator` enumeration.
- Phương thức `compute()` gọi phương thức riêng tư `getOperand()` (mà tiếp tục gọi `getOperandText()`) để lấy giá trị số từ các thành phần **EditText**.
- Phương thức `compute()` sau đó sử dụng câu lệnh `switch` trên tên phép toán để gọi phương thức thích hợp trong đối tượng `Calculator` (được biểu diễn bởi `mCalculator`).
- Các phương thức tính toán trong lớp `Calculator` thực hiện các phép toán số học thực tế và trả về một giá trị.
- Phần cuối cùng của phương thức `compute()` cập nhật **TextView** với kết quả của phép tính.

---

## 1.4 Chạy ứng dụng

1. Chạy ứng dụng và làm theo các bước sau:
  - Nhập cả giá trị số nguyên và số thực cho phép tính.
  - Nhập các giá trị số thực với các phần thập phân lớn (ví dụ: 1.6753456).
  - Thực hiện phép chia một số cho 0.
  - Để trống một hoặc cả hai thành phần EditText và thử bất kỳ phép tính nào.
2. Nhấp vào tab **Logcat** ở cuối cửa sổ Android Studio để mở khung **Logcat** (nếu chưa được mở).
3. Kiểm tra dấu vết ngăn xếp (stack trace) tại điểm ứng dụng báo lỗi.
  - Nếu một hoặc cả hai thành phần EditText trong SimpleCalc để trống, ứng dụng sẽ báo lỗi **exception**, như minh họa trong hình dưới đây.
  - Nhật ký hệ thống (log) sẽ hiển thị trạng thái ngăn xếp thực thi (execution stack) tại thời điểm ứng dụng tạo ra lỗi.
  - Stack trace thường cung cấp thông tin quan trọng về lý do xảy ra lỗi.

### 2.1 Bắt đầu và chạy ứng dụng trong chế độ gỡ lỗi

1. Trong Android Studio, chọn **Run > Debug app** hoặc nhấp vào biểu tượng **Debug** trên thanh công cụ.
2. Nếu ứng dụng của bạn đang chạy, hệ thống sẽ hỏi bạn có muốn khởi động lại ứng dụng trong chế độ gỡ lỗi hay không. Nhấp vào **Restart app** (Khởi động lại ứng dụng).

Android Studio sẽ biên dịch và chạy ứng dụng của bạn trên trình giả lập hoặc trên thiết bị.

- Gỡ lỗi (debugging) đều hoạt động giống nhau trên cả hai trường hợp.
  - Trong khi Android Studio đang khởi tạo debugger, bạn có thể thấy một thông báo trên thiết bị hoặc trình giả lập có nội dung: "**Waiting for debugger**" trước khi bạn có thể sử dụng ứng dụng.
3. Nhấp vào tab **Debug** ở cuối cửa sổ Android Studio để mở bảng điều khiển **Debug** (hoặc chọn **View > Tool Windows > Debug**). Tab **Debugger** sẽ tự động được chọn và hiển thị bảng điều khiển Debugger.

## 2.2 Đặt một điểm dừng (breakpoint)

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

1. Mở tệp **MainActivity**, nhấp vào dòng thứ tư của phương thức compute() (ngay sau câu lệnh try).
2. Nhấp vào phần lề bên trái cạnh số dòng. Một chấm đỏ xuất hiện, biểu thị rằng một breakpoint đã được đặt. Nếu ứng dụng đang chạy ở chế độ debug, chấm đỏ sẽ có thêm dấu kiểm (✓).

Ngoài ra, bạn có thể chọn **Run > Toggle Line Breakpoint** hoặc nhấn **Control-F8** (hoặc **Command-F8** trên Mac) để đặt hoặc xóa breakpoint.

- o Đặt nhầm có thể được sửa bằng cách nhấp lại vào chấm đỏ.
  - o Nếu dòng mã không thể thực thi, chấm đỏ sẽ có thêm dấu "x" và hiển thị cảnh báo.
3. Trong ứng dụng **SimpleCalc**, nhập số vào các ô **EditText** và nhấn một nút tính toán.

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

Hình minh họa hiển thị **ngăn Debug** với các tab **Debugger** và **Console**. Tab **Debugger** được chọn, hiển thị ngăn Debugger với các tính năng sau:

1. **Tab Frames:** Nhấp để hiển thị ngăn **Frames** với các khung ngăn xếp thực thi (execution stack frames) hiện tại cho một luồng (thread) cụ thể. Ngăn xếp thực thi hiển thị từng lớp (class) và phương thức (method) đã được gọi trong ứng dụng của bạn và trong thời gian chạy Android (Android runtime), với phương thức được gọi gần nhất ở trên cùng.
  - o Nhấp vào tab **Threads** để thay thế ngăn **Frames** bằng ngăn **Threads**. Ứng dụng của bạn hiện đang chạy trong luồng chính (main thread) và đang thực thi phương thức **compute()** trong lớp **MainActivity**.
2. **Nút Watches:** Nhấp để hiển thị ngăn **Watches** bên trong ngăn **Variables**, nơi hiển thị giá trị của bất kỳ biến nào mà bạn đã đặt để theo dõi (watches).

Tính năng Watches cho phép bạn theo dõi một biến cụ thể trong chương trình và xem cách biến đó thay đổi khi chương trình chạy.

3. **Ngăn Variables:** Hiển thị các biến trong phạm vi hiện tại và giá trị của chúng. Tại giai đoạn này của việc thực thi ứng dụng, các biến có sẵn bao gồm: **this** (đại diện cho Activity), **operator** (tên toán tử từ **Calculator.Operator** mà phương thức được gọi từ đó), cũng như các biến toàn cục cho các phần tử **EditText** và **TextView**. Mỗi biến trong ngăn này có biểu tượng mở rộng để bạn có thể mở rộng danh sách các thuộc tính của đối tượng cho biến đó. Hãy thử mở rộng một biến để khám phá các thuộc tính của nó.

### 2.3 Tiếp tục thực thi ứng dụng của bạn

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

Ứng dụng **SimpleCalc** sẽ tiếp tục chạy, và bạn có thể tương tác với ứng dụng cho đến khi quá trình thực thi mã tiếp theo đạt đến điểm dừng (breakpoint).

### 2.4 Gõ lỗi một ứng dụng đang chạy

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

1. Chạy ứng dụng **SimpleCalc** bình thường bằng cách nhấp vào biểu tượng **Run**.
2. Chọn **Run > Attach debugger to Android process**, hoặc nhấp vào biểu tượng **Attach** trên thanh công cụ.
3. Chọn tiến trình (process) của ứng dụng bạn từ hộp thoại xuất hiện (như hình minh họa bên dưới). Nhấp vào **OK**.

Cửa sổ Debug xuất hiện với phần Debugger được mở, và bây giờ bạn có thể gõ lỗi ứng dụng của mình như thể bạn đã khởi chạy nó trong chế độ debug.

**Lưu ý:** Nếu cửa sổ Debug không tự động xuất hiện, nhấp vào tab **Debug** ở cuối màn hình. Nếu tab **Debugger** chưa được chọn, nhấp vào tab **Debugger** trong cửa sổ **Debug** để hiển thị phần Debugger.

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

Trong nhiệm vụ này, chúng ta sẽ khám phá các tính năng khác nhau trong trình gõ lỗi của Android Studio, bao gồm thực thi ứng dụng từng dòng, làm việc với các điểm dừng (breakpoints), và kiểm tra các biến.

### 3.1 Thực hiện từng bước khi ứng dụng chạy

Sau khi dừng tại điểm dừng (breakpoint), bạn có thể sử dụng trình gỡ lỗi để thực thi từng dòng mã trong ứng dụng và kiểm tra trạng thái của các biến khi ứng dụng chạy.

1. Gỡ lỗi ứng dụng trong Android Studio với điểm dừng mà bạn đã đặt ở nhiệm vụ trước.
2. Trong ứng dụng, nhập số vào cả hai trường **EditText**, sau đó nhấp vào nút **Add**.
  - Việc thực thi ứng dụng sẽ dừng lại tại điểm dừng mà bạn đã đặt trước đó, và cửa sổ **Debugger** sẽ hiển thị trạng thái hiện tại của ứng dụng. Dòng mã hiện tại sẽ được đánh dấu nổi bật trong mã nguồn của bạn.
3. Nhấp vào nút **Step Over** ở đầu cửa sổ trình gỡ lỗi.
  - Trình gỡ lỗi sẽ thực thi dòng mã hiện tại trong phương thức **compute()** (nơi bạn đặt điểm dừng, dòng gán giá trị cho biến `operandOne`), và điểm đánh dấu sẽ chuyển sang dòng tiếp theo trong mã nguồn (dòng gán giá trị cho biến `operandTwo`).
  - Cửa sổ **Variables** sẽ cập nhật để phản ánh trạng thái thực thi mới, và các giá trị hiện tại của biến sẽ xuất hiện dưới dạng chữ nghiêng sau mỗi dòng mã trong mã nguồn.
  - Bạn cũng có thể sử dụng **Run > Step Over** hoặc nhấn **F8** để thực hiện lệnh này.
4. Tại dòng tiếp theo (dòng gán giá trị cho `operandTwo`), nhấp vào biểu tượng **Step Into**.
  - **Step Into** nhảy vào việc thực thi một lời gọi phương thức trong dòng hiện tại (so với việc chỉ thực thi phương thức đó và giữ nguyên tại dòng hiện tại). Trong trường hợp này, do dòng lệnh bao gồm lời gọi tới phương thức **getOperand()**, trình gỡ lỗi sẽ cuộn tới định nghĩa của phương thức **getOperand()** trong mã nguồn của **MainActivity**.
  - Khi bạn nhảy vào một phương thức, cửa sổ **Frames** sẽ cập nhật để hiển thị khung mới trong ngăn xếp lệnh gọi (ở đây là **getOperand()**), và cửa sổ **Variables** hiển thị các biến có sẵn trong phạm vi phương thức mới. Bạn có thể nhấp vào bất kỳ dòng nào trong cửa sổ **Frames** để xem điểm trong khung lệnh gọi trước đó nơi phương thức được gọi.

Bạn cũng có thể sử dụng **Run > Step Into** hoặc phím **F7** để nhảy vào một phương thức.

- Nhấp vào **Step Over** để thực thi từng dòng trong phương thức **getOperand()**.
  - Lưu ý rằng khi phương thức hoàn thành, trình gõ lỗi sẽ đưa bạn quay lại điểm mà bạn đã nhảy vào phương thức ban đầu, và tất cả các bảng điều khiển sẽ được cập nhật để hiển thị thông tin mới.
- Nhấp vào **Step Over** hai lần để di chuyển điểm thực thi đến dòng đầu tiên bên trong câu lệnh **case** cho **ADD**.
- Nhấp vào **Step Into**.
  - Trình gõ lỗi thực thi phương thức phù hợp được định nghĩa trong lớp **Calculator**, mở tệp **Calculator.java**, và cuộn đến điểm thực thi trong lớp đó. Một lần nữa, các bảng điều khiển khác nhau sẽ được cập nhật để phản ánh trạng thái mới.
- Sử dụng biểu tượng **Step Out** để thực thi phần còn lại của phương thức tính toán đó và quay trở lại phương thức **compute()** trong **MainActivity**. Bạn có thể tiếp tục gõ lỗi phương thức **compute()** từ điểm mà bạn đã dừng trước đó.
  - Bạn cũng có thể sử dụng **Run > Step Out** hoặc nhấn **Shift-F8** để thoát khỏi việc thực thi một phương thức.

### 3.2 Làm việc với Breakpoint

Sử dụng breakpoint để chỉ định vị trí trong mã mà bạn muốn tạm dừng thực thi ứng dụng để gõ lỗi phần đó của ứng dụng.

- Tìm breakpoint mà bạn đã đặt trong tác vụ trước—ở phần bắt đầu của phương thức **compute()** trong **MainActivity**.
- Thêm một breakpoint vào dòng bắt đầu của câu lệnh **switch**.
- Nhấp chuột phải vào breakpoint mới đó để nhập một điều kiện, như trong hình minh họa bên dưới, và nhập thử nghiệm sau vào trường **Condition**:

CopyEdit

(operandOne == 42)||(operandTwo == 42)

- Nhấn **Done** (Hoàn tất).

Breakpoint thứ hai này là một **breakpoint có điều kiện** (*conditional breakpoint*). Việc thực thi ứng dụng của bạn chỉ dừng lại tại breakpoint này nếu kiểm tra điều kiện trong biểu thức là **true**. Trong trường hợp này, biểu thức chỉ đúng nếu một trong hai toán hạng bạn nhập là **42**. Bạn có thể nhập bất kỳ biểu thức Java nào làm điều kiện miễn là nó trả về một giá trị **boolean**.

5. Chạy ứng dụng của bạn ở chế độ gỡ lỗi (**Run > Debug**) hoặc nhấn **Resume** nếu ứng dụng đã chạy. Trong ứng dụng, nhập hai số khác **42** và nhấp vào nút **Add**. Việc thực thi sẽ dừng lại tại breakpoint đầu tiên trong phương thức **compute()**.
6. Nhấn **Resume** để tiếp tục gỡ lỗi ứng dụng. Quan sát rằng việc thực thi không dừng lại ở breakpoint thứ hai của bạn vì điều kiện không được đáp ứng.
7. Trong ứng dụng, nhập **42** vào ô **EditText** đầu tiên và nhấn bất kỳ nút nào. Nhấn **Resume** để tiếp tục thực thi sau breakpoint đầu tiên. Quan sát rằng breakpoint thứ hai tại câu lệnh **switch**—breakpoint có điều kiện—dừng việc thực thi vì điều kiện đã được đáp ứng.
8. Nhấp chuột phải (hoặc **Control-click**) vào breakpoint đầu tiên trong phương thức **compute()** và bỏ chọn **Enabled**. Nhấn **Done**. Quan sát rằng biểu tượng breakpoint giờ có một chấm xanh với viền đỏ.  
Việc vô hiệu hóa một breakpoint cho phép bạn tạm thời "tắt tiếng" breakpoint đó mà không thực sự xóa nó khỏi mã của bạn. Nếu bạn xóa hoàn toàn một breakpoint, bạn cũng sẽ mất bất kỳ điều kiện nào đã tạo cho breakpoint đó, vì vậy việc vô hiệu hóa thường là lựa chọn tốt hơn.  
Bạn cũng có thể tắt tất cả các breakpoint trong ứng dụng cùng lúc bằng biểu tượng **Mute Breakpoints**.
9. Nhấp vào **View Breakpoints** ở cạnh trái của cửa sổ debugger. 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 các điều kiện, phụ thuộc vào các breakpoint khác và ghi nhật ký (*logging*).

Để đóng cửa sổ **Breakpoints**, nhấn **Done**.

### 3.3 Kiểm tra và thay đổi biến

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

1. Chạy ứng dụng **SimpleCalc** ở chế độ debug nếu nó chưa chạy.
2. Trong ứng dụng, nhập hai số, một trong số đó là **42**, sau đó nhấn nút **Add**. Breakpoint đầu tiên trong phương thức **compute()** hiện đang bị tắt. Việc thực thi dừng lại tại breakpoint thứ hai (breakpoint có điều kiện ở câu lệnh **switch**), và trình gỡ lỗi sẽ xuất hiện.
3. Quan sát trong bảng **Variables** rằng các biến **operandOne** và **operandTwo** đã nhận giá trị mà bạn đã nhập vào ứng dụng.

4. Biến **this** là một đối tượng thuộc lớp **MainActivity**. Nhấp vào biểu tượng mở rộng để xem danh sách các biến thành viên của đối tượng đó. Sau đó, nhấp lại vào biểu tượng mở rộng để đóng danh sách.
5. Nhấp chuột phải (hoặc **Control-click**) vào biến **operandOne** trong bảng **Variables**, và chọn **Set Value**.
6. Thay đổi giá trị của biến **operandOne** thành **10** và nhấn **Return**.
7. Thay đổi giá trị của biến **operandTwo** thành **10** theo cách tương tự và nhấn **Return**.
8. Quan sát rằng kết quả trong ứng dụng bây giờ dựa trên các giá trị biến mà bạn đã thay đổi trong trình gõ lỗi; ví dụ, vì bạn đã nhấn nút **Add** ở Bước 2, kết quả trong ứng dụng giờ là **20**.
9. Nhấp vào biểu tượng **Resume** để tiếp tục chạy ứng dụng của bạn.
10. Trong ứng dụng, các mục nhập ban đầu (bao gồm **42**) vẫn được giữ nguyên trong các trường **EditText**. (Các giá trị này chỉ được thay đổi trong trình gõ lỗi.) Nhấn lại nút **Add**. Quá trình thực thi sẽ dừng lại tại điểm breakpoint một lần nữa.
11. Nhấn vào biểu tượng **Evaluate Expression**, hoặc chọn **Run > Evaluate Expression**. Bạn cũng có thể nhấp chuột phải (hoặc **Control-click**) vào bất kỳ biến nào và chọn **Evaluate Expression**.

Cửa sổ **Evaluate Code Fragment** xuất hiện. Sử dụng nó để khám phá trạng thái của các biến và đối tượng trong ứng dụng của bạn, bao gồm cả việc gọi các phương thức trên các đối tượng đó. Bạn có thể nhập bất kỳ đoạn mã nào vào cửa sổ này.

12. Nhập câu lệnh **mOperandOneEditText.getHint()**; vào trường phía trên của cửa sổ **Evaluate Code Fragment** (như minh họa trong hình trên) và nhấn **Evaluate**.
13. Trường **Result** hiển thị kết quả của biểu thức đó. Gợi ý (hint) cho trường **EditText** này là chuỗi "**Type Operand 1**", như đã được định nghĩa ban đầu trong tệp XML cho trường **EditText** này.

Kết quả bạn nhận được khi đánh giá một biểu thức dựa trên trạng thái hiện tại của ứng dụng. Tùy thuộc vào giá trị của các biến trong ứng dụng tại thời điểm bạn đánh giá biểu thức, bạn có thể nhận được kết quả khác nhau.

Cũng cần lưu ý rằng nếu bạn sử dụng **Evaluate Expression** để thay đổi giá trị của biến hoặc thuộc tính của đối tượng, bạn sẽ thay đổi trạng thái đang chạy của ứng dụng.

#### 14. Nhấn **Close** để đóng cửa sổ **Evaluate Code Fragment**.

Thách thức lập trình

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

**Thách thức:** Ở cuối Bài 1, bạn đã thử chạy ứng dụng **SimpleCalc** mà không nhập giá trị vào một trong các trường **EditText**, dẫn đến lỗi. Sử dụng trình gỡ lỗi để từng bước thực thi mã và xác định chính xác lý do gây ra lỗi này. Khắc phục lỗi gây ra sự cố này.

**Tóm tắt:**

- Xem thông tin ghi log trong Android Studio bằng cách nhấp vào tab **Logcat**.
- Chạy ứng dụng của bạn trong chế độ gỡ lỗi bằng cách nhấp vào biểu tượng **Debug** hoặc chọn **Run > Debug app**.
- Nhấp vào tab **Debug** để hiển thị khung **Debug**. Nhấp vào tab **Debugger** trong khung **Debug** để hiển thị khung **Debugger** (nếu nó chưa được chọn).
- Khung **Debugger** hiển thị các **Frames** (ngăn xếp), **Variables** trong một frame cụ thể và **Watches** (theo dõi hoạt động của một biến khi chương trình chạy).
- **Breakpoint** là một điểm trong mã của bạn nơi bạn muốn tạm dừng quá trình thực thi bình thường của ứng dụng để thực hiện các hành động khác. Đặt hoặc xóa một điểm dừng gỡ lỗi bằng cách nhấp vào lề bên trái của cửa sổ trình soạn thảo ngay cạnh dòng mục tiêu.

**Khái niệm liên quan:**

Tài liệu khái niệm liên quan nằm trong mục **3.1: Trình gỡ lỗi của Android Studio**.

**Tìm hiểu thêm**

**Tài liệu Android Studio:**

- Hướng dẫn sử dụng Android Studio
- Gỡ lỗi ứng dụng của bạn
- Ghi và xem log
- Phân tích Stack Trace
- Cầu nối gỡ lỗi Android (ADB)

- Công cụ phân tích Android Profiler
- Trình phân tích mạng (Network Profiler)
- Trình phân tích CPU (CPU Profiler)
- Traceview

## Khác:

- Video: Gỡ lỗi và kiểm thử trong Android Studio

## Bài tập về nhà

### Xây dựng và chạy ứng dụng

Mở ứng dụng SimpleCalc.

1. Trong **MainActivity**, đặt một breakpoint ở dòng đầu tiên của phương thức **onAdd()**.
2. Chạy ứng dụng trong chế độ gỡ lỗi. Thực hiện một phép tính cộng trong ứng dụng. Quá trình thực thi dừng lại tại breakpoint.
3. Nhấp vào **Step Into** để theo dõi quá trình thực thi của ứng dụng từng bước. Lưu ý rằng **Step Into** sẽ mở và thực thi các tệp từ framework Android, cho phép bạn xem cách Android hoạt động với mã của bạn.
4. Quan sát cách khung **Debug** thay đổi khi bạn bước qua mã cho frame ngăn xếp hiện tại và các biến cục bộ.
5. Quan sát cách mã trong khung trình soạn thảo được chú thích khi mỗi dòng được thực thi.
6. Nhấp vào **Step Out** để quay lại ứng dụng của bạn nếu ngăn xếp thực thi trở nên quá sâu để hiểu.

## Câu hỏi 1

Chạy ứng dụng **SimpleCalc** mà không sử dụng trình gỡ lỗi. Để trống một hoặc cả hai trường **EditText**, sau đó thử thực hiện phép tính bất kỳ. Tại sao lỗi xảy ra?

- **java.lang.NumberFormatException: empty String**
- **W/OpenGLOutputSurface: Failed to choose config with EGL\_SWAP\_BEHAVIOR\_PRESERVED**
- **Ứng dụng có thể đang thực hiện quá nhiều công việc trên luồng chính.**
- **Dung lượng bộ nhớ cache của mã đã được tăng lên 128KB.**

## Câu hỏi 2

Chức năng nào bạn thực hiện trong bảng **Debug** để thực thi dòng hiện tại nơi đặt breakpoint, sau đó dừng ở dòng tiếp theo trong mã? Hãy chọn một:

- **Step Into**
- **Step Over**
- **Step Out**
- **Resume**

### Câu hỏi 3

Chức năng nào bạn thực hiện trong bảng **Debug** để nhảy tới thực thi một phương thức được gọi từ dòng hiện tại nơi đặt breakpoint? Hãy chọn một:

- **Step Into**
- **Step Over**
- **Step Out**
- **Resume**

### Gửi ứng dụng của bạn để chấm điểm

#### Hướng dẫn cho người chấm điểm

Không có ứng dụng nào cần gửi cho bài tập về nhà này.

## Bài học 3.2: Unit tests

### Giới thiệu

Kiểm thử mã của bạn có thể giúp bạn phát hiện lỗi sớm trong quá trình phát triển, khi chi phí xử lý lỗi là thấp nhất. Khi ứng dụng của bạn trở nên lớn hơn và phức tạp hơn, việc kiểm thử cải thiện độ tin cậy của mã.

Với các bài kiểm thử trong mã, bạn có thể kiểm tra từng phần nhỏ trong ứng dụng một cách độc lập, và bạn có thể kiểm thử theo cách tự động hóa và lặp lại được.

Android Studio và **Android Testing Support Library** hỗ trợ nhiều loại kiểm thử và khung kiểm thử khác nhau. Trong bài thực hành này, bạn sẽ khám phá chức năng kiểm thử tích hợp sẵn của Android Studio và học cách viết và chạy các bài kiểm thử đơn vị cục bộ.

**Kiểm thử đơn vị cục bộ** là các bài kiểm thử được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với **Java Virtual Machine (JVM)**. Bạn sử dụng kiểm thử đơn vị cục bộ để kiểm tra các phần của ứng dụng không cần truy cập vào **Android framework** hoặc thiết bị/emulator Android, chẳng hạn như logic nội bộ.

Bạn cũng có thể sử dụng kiểm thử đơn vị cục bộ để kiểm tra các phần của ứng dụng mà bạn có thể tạo ra các đối tượng giả ("mock" hoặc "stub") để bắt chước hành vi của các thành phần tương đương trong framework.

Kiểm thử đơn vị được viết bằng **JUnit**, một khung kiểm thử đơn vị phổ biến dành cho Java.

---

## Những gì bạn cần biết trước

Bạn cần phải:

- Tạo một dự án Android Studio.
- Xây dựng và chạy ứng dụng của bạn trong Android Studio, trên cả trình giả lập và thiết bị thực.
- Điều hướng trong **Project > Android** pane của Android Studio.

### • Tìm các thành phần chính của một dự án Android Studio, bao gồm:

- **AndroidManifest.xml**,
  - Các tài nguyên (resources),
  - Các tệp Java,
  - Các tệp Gradle.
- 

## Bạn sẽ học được gì?

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

## Bạn sẽ làm gì?

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

## Tổng quan về ứng dụng

Bài thực hành này sử dụng ứng dụng **SimpleCalc** từ bài codelab thực hành trước (**Android fundamentals 3.1: The debugger**). Bạn có thể chỉnh sửa ứng dụng này tại chỗ hoặc tạo một bản sao thư mục dự án trước khi tiếp tục.

---

## 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) trong **Android Studio**, cùng với mã 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 để sử dụng cho mục đích riêng của mình.

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

### 1.1 Khám phá bộ mã nguồn (source sets) và CalculatorTest

**Source sets** là các tập hợp mã trong dự án của bạn dùng cho các mục tiêu build khác nhau hoặc các "phiên bản" khác nhau của ứng dụng. Khi Android Studio tạo dự án của bạn, nó sẽ tạo ra ba bộ mã nguồn:

- **Bộ mã nguồn chính (main source set)**: Chứa mã và tài nguyên chính của ứng dụng.
  - **Bộ mã nguồn (test)**: Chứa các bài kiểm thử đơn vị cục bộ của ứng dụng. Bộ mã nguồn này hiển thị (**test**) sau tên gói.
  - **Bộ mã nguồn (androidTest)**: Chứa các bài kiểm thử có công cụ (**instrumented tests**) dành cho Android. Bộ mã nguồn này hiển thị (**androidTest**) sau tên gói.
- 

**Trong nhiệm vụ này, bạn sẽ khám phá cách các bộ mã nguồn được hiển thị trong Android Studio, kiểm tra cấu hình Gradle dành cho việc kiểm thử, và chạy các bài kiểm thử đơn vị cho ứng dụng SimpleCalc.**

**Lưu ý:** Bộ mã nguồn (**androidTest**) đã được loại bỏ khỏi ví dụ này để đơn giản hóa. Nó sẽ được giải thích chi tiết hơn trong một bài học khác.

---

## Các bước thực hiện

1. Mở dự án **SimpleCalc** trong Android Studio nếu bạn chưa làm điều này.
2. Mở bảng điều hướng **Project > Android**, sau đó mở rộng các thư mục **app** và **java**.
  - o Thư mục **java** trong chế độ xem Android hiển thị tất cả các bộ mã nguồn trong ứng dụng theo tên gói.
  - o Trong trường hợp này (như minh họa bên dưới), mã của ứng dụng nằm trong bộ mã nguồn **com.android.example.SimpleCalc**.
  - o Mã kiểm thử nằm trong bộ mã nguồn có chữ **test** xuất hiện trong ngoặc đơn sau tên gói: **com.android.example.SimpleCalc (test)**.

## 2. Mở rộng thư mục com.android.example.SimpleCalc (test)

Thư mục này là nơi bạn đặt các bài kiểm thử đơn vị cục bộ của ứng dụng. Android Studio tạo sẵn một lớp kiểm thử mẫu trong thư mục này cho các dự án mới, nhưng với ứng dụng **SimpleCalc**, lớp kiểm thử được gọi là **CalculatorTest**.

---

### 1. Mở tệp CalculatorTest

Hãy kiểm tra mã nguồn và lưu ý những điểm sau:

- **Các thư viện được nhập:**  
Chỉ có các thư viện từ các gói **org.junit**, **org.hamcrest**, và **android.test** được nhập. Không có sự phụ thuộc nào vào các lớp của framework Android.
- **Chú thích @RunWith(JUnit4.class):**  
Chú thích này chỉ định **runner** sẽ được sử dụng để chạy các bài kiểm thử trong lớp này. **Test runner** là một thư viện hoặc bộ công cụ cho phép thực hiện kiểm thử và in kết quả ra log. Với các bài kiểm thử cần thiết lập hoặc hàn tầng phức tạp hơn (như Espresso), bạn sẽ sử dụng các test runner khác.  
Trong ví dụ này, chúng ta sử dụng **JUnit4 test runner** cơ bản.
- **Chú thích @SmallTest:**  
Chú thích này cho biết tất cả các bài kiểm thử trong lớp là kiểm thử đơn vị, không có phụ thuộc nào và chạy trong thời gian rất ngắn (tính bằng mili giây). Các chú thích **@SmallTest**, **@MediumTest**, và **@LargeTest** là các quy ước giúp dễ dàng nhóm các bài kiểm thử thành các bộ với chức năng tương tự nhau.
- **Phương thức setUp():**  
Phương thức này được dùng để thiết lập môi trường trước khi kiểm thử và

có chú thích **@Before**. Trong trường hợp này, phương thức **setUp()** tạo một instance mới của lớp **Calculator** và gán nó vào biến thành viên **mCalculator**.

- **Phương thức addTwoNumbers():**

Đây là một bài kiểm thử thực tế và được chú thích bằng **@Test**. Chỉ những phương thức trong lớp kiểm thử có chú thích **@Test** mới được trình kiểm thử (test runner) xem là bài kiểm thử. Theo quy ước, tên của phương thức kiểm thử không bao gồm từ "test."

- **Dòng lệnh đầu tiên của addTwoNumbers():**

Dòng lệnh này gọi phương thức **add()** từ lớp **Calculator**. Bạn chỉ có thể kiểm thử các phương thức có phạm vi truy cập là **public** hoặc **package-protected**. Trong trường hợp này, lớp **Calculator** là lớp **public** với các phương thức **public**, nên không có vấn đề gì.

- **Dòng lệnh thứ hai:**

Đây là một biểu thức khẳng định (assertion) cho bài kiểm thử. **Assertion** là các biểu thức phải đánh giá và trả về **true** để bài kiểm thử được thông qua. Trong trường hợp này, **assertion** kiểm tra xem kết quả nhận được từ phương thức **add()** ( $1 + 1$ ) có khớp với số được chỉ định là 2 hay không. Bạn sẽ tìm hiểu thêm về cách tạo các **assertion** sau trong bài thực hành này.

## 1.2 Chạy kiểm thử trong Android Studio

Trong phần này, bạn sẽ chạy các bài kiểm thử đơn vị trong thư mục kiểm thử và xem kết quả cho cả kiểm thử thành công và thất bại.

---

### Các bước thực hiện:

1. Trong ngăn **Project > Android**, nhấp chuột phải (hoặc **Control-click**) vào **CalculatorTest** và chọn **Run 'CalculatorTest'**.
  - Dự án sẽ được xây dựng (nếu cần thiết), và ngăn **CalculatorTest** sẽ xuất hiện ở dưới cùng của màn hình.
  - Ở đầu ngăn này, danh sách thả xuống cho các cấu hình thực thi khả dụng cũng thay đổi thành **CalculatorTest**.
2. Tất cả các bài kiểm thử trong lớp **CalculatorTest** sẽ được chạy.
  - Nếu các bài kiểm thử thành công, thanh tiến trình ở đầu khung nhìn sẽ chuyển sang màu **xanh lá cây**.
  - Một thông báo trạng thái ở cuối màn hình sẽ báo "**Tests Passed**".

3. Mở **CalculatorTest** nếu nó chưa được mở, và thay đổi biểu thức khẳng định trong phương thức **addTwoNumbers()** thành:

java

CopyEdit

```
assertThat(resultAdd, is(equalTo(3d)));
```

## 2. Chạy lại bài kiểm thử với cấu hình CalculatorTest

1. Trong danh sách thả xuống cấu hình chạy (run configurations) ở phía trên màn hình, chọn **CalculatorTest** (nếu nó chưa được chọn) và nhấn **Run**.
  - o Bài kiểm thử sẽ chạy lại như trước, nhưng lần này biểu thức khẳng định thất bại (**3 không bằng 1 + 1**).
  - o Thanh tiến trình trong khung chạy kiểm thử chuyển sang màu **đỏ**, và nhật ký kiểm thử chỉ ra vị trí bài kiểm thử (biểu thức khẳng định) thất bại cũng như lý do tại sao.
2. Thay đổi biểu thức khẳng định trong phương thức **addTwoNumbers()** quay lại bài kiểm thử đúng, và chạy lại kiểm thử để đảm bảo tất cả bài kiểm thử đều thành công.
3. Trong danh sách thả xuống cấu hình chạy, chọn **app** để chạy ứng dụng bình thường.

---

## Nhiệm vụ 2: Thêm các kiểm thử đơn vị khác vào CalculatorTest

Kiểm thử đơn vị là kiểm tra một phần nhỏ trong mã nguồn của bạn (chẳng hạn một phương thức hoặc một lớp), và tách phần đó khỏi phần còn lại của ứng dụng. Điều này giúp đảm bảo rằng phần mã nhỏ đó hoạt động đúng như mong đợi.

Thông thường, một kiểm thử đơn vị gọi một phương thức với nhiều đầu vào khác nhau, và kiểm tra rằng phương thức hoạt động như mong đợi và trả về kết quả đúng.

Trong nhiệm vụ này, bạn sẽ viết thêm các kiểm thử đơn vị cho các phương thức tiện ích của lớp **Calculator** trong ứng dụng **SimpleCalc**, và chạy các bài kiểm thử để đảm bảo rằng chúng tạo ra đầu ra như bạn mong đợi.

---

## 2.1 Thêm các bài kiểm thử khác cho phương thức add()

Mặc dù không thể kiểm thử mọi giá trị đầu vào mà phương thức **add()** có thể nhận, nhưng việc kiểm tra các trường hợp đầu vào đặc biệt là rất quan trọng. Ví dụ:

- Đầu vào có chứa **toán hạng âm**.
- Đầu vào có chứa **số thực (floating-point numbers)**.
- **Số rất lớn (exceptionally large numbers)**
- **Các loại toán hạng khác nhau (float và double)**
- **Toán hạng bằng 0 (zero)**
- **Toán hạng là vô cực (infinity)**

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

1. **Thêm phương thức mới vào CalculatorTest có tên là addTwoNumbersNegative().**

Sử dụng khung mã như sau:

```
java
```

```
CopyEdit
```

```
@Test
```

```
public void addTwoNumbersNegative() {
}
```

Phương thức kiểm thử này có cấu trúc tương tự như **addTwoNumbers()**:

- Là phương thức **public**, không có tham số, và trả về kiểu **void**.
- Được chú thích bằng **@Test**, cho biết đây là một bài kiểm thử đơn lẻ.

- 
2. **Tại sao không thêm nhiều lệnh kiểm tra (assertions) vào addTwoNumbers()?**

- Gom nhiều lệnh kiểm tra vào một phương thức có thể làm cho bài kiểm thử khó gỡ lỗi nếu chỉ một lệnh kiểm tra thất bại.
  - Ngoài ra, việc này cũng làm mờ các bài kiểm thử thành công khác.
  - Quy tắc chung cho các bài kiểm thử đơn vị là **tạo một phương thức kiểm thử cho từng lệnh kiểm tra riêng lẻ**.
- 

### 3. Chạy tất cả các bài kiểm thử trong CalculatorTest như trước.

- Trong cửa sổ kiểm thử, cả addTwoNumbers và addTwoNumbersNegative đều được liệt kê dưới dạng các bài kiểm thử khả dụng (và đang thành công) trong bảng điều khiển bên trái.
  - Bài kiểm thử addTwoNumbersNegative vẫn thành công ngay cả khi không chứa bất kỳ đoạn mã nào—một bài kiểm thử không làm gì vẫn được xem là thành công.
- 

### 4. Thêm một dòng mã vào addTwoNumbersNegative() để gọi phương thức add() trong lớp Calculator với một toán hạng âm.

java

CopyEdit

```
double resultAdd = mCalculator.add(1d, 2d);
```

**Ký hiệu d sau mỗi toán hạng cho biết đây là các số kiểu double.**

Do phương thức add() được định nghĩa với các tham số kiểu double, các kiểu dữ liệu khác như float hoặc int cũng có thể hoạt động. Việc chỉ rõ kiểu giúp bạn kiểm tra riêng biệt các kiểu khác nếu cần thiết.

---

### 4. Thêm một lệnh kiểm tra (assertion) với assertThat().

java

CopyEdit

```
assertThat(resultAdd, is(equalTo(1d)));
```

- Phương thức assertThat() là một lệnh kiểm tra trong JUnit4, khăng định biểu thức trong đối số đầu tiên bằng với biểu thức trong đối số thứ hai.
  - Các phiên bản cũ của JUnit sử dụng các phương thức kiểm tra cụ thể hơn như assertEquals(), assertNull(), hoặc assertTrue(), nhưng assertThat() linh hoạt hơn, dễ đọc và dễ gỡ lỗi hơn.
- 

### **assertThat() sử dụng matchers.**

- Matchers là các phương thức liên kết được gọi trong đối số thứ hai của lệnh kiểm tra này, chẳng hạn như is(equalTo()).
- Framework Hamcrest định nghĩa các matchers sẵn có để xây dựng các lệnh kiểm tra. (Tên "Hamcrest" là một phép đảo chữ từ "matchers.")
- Hamcrest cung cấp nhiều matchers cơ bản cho hầu hết các lệnh kiểm tra. Bạn cũng có thể tự định nghĩa matchers của riêng mình cho các lệnh kiểm tra phức tạp hơn.

#### **Ví dụ:**

Trong trường hợp này, lệnh kiểm tra xác nhận rằng kết quả của phép cộng add() (-1 + 2) bằng với 1.

---

### **5. Thêm bài kiểm tra đơn vị mới vào CalculatorTest cho số dấu phẩy động:**

java

CopyEdit

@Test

```
public void addTwoNumbersFloats() {
```

```
 double resultAdd = mCalculator.add(1.111f, 1.111d);
```

```
 assertThat(resultAdd, is(equalTo(2.222d)));
```

```
}
```

- Đây là một bài kiểm tra rất giống với phương thức kiểm tra trước, nhưng một đối số của phương thức add() được chỉ rõ là kiểu float thay vì double.

- Phương thức add() được định nghĩa với các tham số kiểu double, vì vậy bạn có thể gọi nó với kiểu float. Khi đó, số kiểu float sẽ được chuyển đổi thành kiểu double.
- 

## 6. Nhấn vào Run để chạy lại tất cả các bài kiểm tra.

**Lần này bài kiểm tra thất bại và thanh tiến trình chuyển sang màu đỏ. Đây là phần quan trọng trong thông báo lỗi:**

makefile

CopyEdit

java.lang.AssertionError:

Expected: is <2.222>

but: was <2.2219999418258665>

**Tính toán với các số dấu phẩy động không chính xác, và việc chuyển đổi đã gây ra hiệu ứng phụ về độ chính xác.**

Lệnh kiểm tra trong bài kiểm tra này 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 do việc chuyển đổi tham số kiểu float, đó có phải là vấn đề của mã nguồn hay bài kiểm tra?

- Trong trường hợp này, cả hai đối số đầu vào của phương thức add() từ ứng dụng SimpleCalc đều luôn thuộc kiểu double, do đó đây là một bài kiểm tra tùy ý và không thực tế.
  - Tuy nhiên, nếu ứng dụng của bạn được viết sao cho đầu vào của phương thức add() có thể là kiểu double hoặc float, và bạn chỉ quan tâm đến một mức độ chính xác nhất định, thì bạn cần thêm một biên độ "gần đúng" để bài kiểm tra thành công.
-

## 7. Thay đổi phương thức assertThat() để sử dụng matcher closeTo():

java

CopyEdit

```
assertThat(resultAdd, is(closeTo(2.222, 0.01)));
```

### Hướng dẫn:

- Bạn cần chọn matcher thích hợp. Nhấp vào closeTo hai lần (cho đến khi toàn bộ biểu thức được gạch chân) và nhấn **Alt+Enter (Option+Return)** trên Mac).
  - Chọn **isCloseTo.closeTo (org.hamcrest.number)**.
- 

## 8. Nhấn vào Run để chạy lại tất cả các bài kiểm tra.

Lần này bài kiểm tra đã thành công.

Với matcher closeTo(), thay vì kiểm tra sự bằng nhau tuyệt đối, bạn có thể kiểm tra sự bằng nhau trong một phạm vi delta nhất định.

- Trong trường hợp này, phương thức matcher closeTo() nhận hai tham số: giá trị mong đợi và giá trị delta.
- Trong ví dụ trên, delta là phạm vi gần đúng với độ chính xác hai chữ số thập phân.

## 2.2 Thêm các bài kiểm tra đơn vị cho các phương thức tính toán khác

Sử dụng những gì bạn đã học trong nhiệm vụ trước để hoàn thiện các bài kiểm tra đơn vị cho lớp Calculator.

1. Thêm một bài kiểm tra đơn vị có tên subTwoNumbers() để kiểm tra phương thức sub().
2. Thêm một bài kiểm tra đơn vị có tên subWorksWithNegativeResults() để kiểm tra phương thức sub() khi phép tính cho ra kết quả âm.
3. Thêm một bài kiểm tra đơn vị có tên mulTwoNumbers() để kiểm tra phương thức mul().

4. Thêm một bài kiểm tra đơn vị có tên mulTwoNumbersZero() để kiểm tra phương thức mul() khi ít nhất một tham số là số 0.
5. Thêm một bài kiểm tra đơn vị có tên divTwoNumbers() để kiểm tra phương thức div() với hai tham số khác 0.
6. Thêm một bài kiểm tra đơn vị có tên divTwoNumbersZero() để kiểm tra phương thức div() với một số chia kiểu double và số chia là 0.

Tất cả các bài kiểm tra trên đều phải thành công, ngoại trừ divTwoNumbersZero(), bài này sẽ gây ra ngoại lệ tham số không hợp lệ khi chia cho 0.

- Nếu bạn chạy ứng dụng, nhập 0 làm Số hạng 2 và nhấn Div để chia, kết quả sẽ là lỗi.

## 2.2 Vòng đời của Activity

Giải pháp mã cho nhiệm vụ 2:

**Dự án Android Studio:** SimpleCalcTest

Đoạn mã sau đây minh họa các bài kiểm tra cho nhiệm vụ này:

```
public void addTwoNumbers() {
 double resultAdd = mCalculator.add(1d, 1d);
 assertThat(resultAdd, is(equalTo(2d)));
}

@Test
public void addTwoNumbersNegative() {
 double resultAdd = mCalculator.add(-1d, 2d);
 assertThat(resultAdd, is(equalTo(1d)));
}

@Test
public void addTwoNumbersFloats() {
 double resultAdd = mCalculator.add(1.111f, 1.111d);
 assertThat(resultAdd, is(closeTo(2.222, 0.01)));
}

@Test
public void subTwoNumbers() {
 double resultSub = mCalculator.sub(1d, 1d);
 assertThat(resultSub, is(equalTo(0d)));
}

@Test
public void subWorksWithNegativeResult() {
 double resultSub = mCalculator.sub(1d, 17d);
 assertThat(resultSub, is(equalTo(-16d)));
}

@Test
public void multTwoNumbers() {
 double resultMul = mCalculator.mul(32d, 2d);
 assertThat(resultMul, is(equalTo(64d)));
}

@Test
public void divTwoNumbers() {
 double resultDiv = mCalculator.div(32d, 2d);
 assertThat(resultDiv, is(equalTo(16d)));
}

@Test
public void divTwoNumbersZero() {
 double resultDiv = mCalculator.div(32d, 0);
 assertThat(resultDiv, is(equalTo(Double.POSITIVE_INFINITY)));
}
```

```
@Test
public void subWorksWithNegativeResult() {
 double resultSub = mCalculator.sub(1d, 17d);
 assertThat(resultSub, is(equalTo(-16d)));
}

@Test
public void multTwoNumbers() {
 double resultMul = mCalculator.mul(32d, 2d);
 assertThat(resultMul, is(equalTo(64d)));
}

@Test
public void divTwoNumbers() {
 double resultDiv = mCalculator.div(32d, 2d);
 assertThat(resultDiv, is(equalTo(16d)));
}

@Test
public void divTwoNumbersZero() {
 double resultDiv = mCalculator.div(32d, 0);
 assertThat(resultDiv, is(equalTo(Double.POSITIVE_INFINITY)));
}
```

---

## Thách thức lập trình

**Lưu ý:** Tất cả các thách thức lập trình đều tùy chọn và không phải là yêu cầu bắt buộc cho các bài học sau.

---

### Thách thức 1:

Chia cho 0 luôn là một trường hợp đặc biệt trong toán học và đáng để kiểm tra. Làm thế nào bạn có thể thay đổi ứng dụng để xử lý chia cho 0 một cách dễ chịu hơn?

Để thực hiện thách thức này, hãy bắt đầu bằng một bài kiểm tra cho thấy hành vi đúng là gì.

- Xóa phương thức divTwoNumbersZero() khỏi lớp CalculatorTest và thêm một bài kiểm tra đơn vị mới có tên là divByZeroThrows() để kiểm tra phương thức div() với đối số thứ hai bằng 0, với kết quả mong đợi là IllegalArgumentException.class.

- Bài kiểm tra này sẽ thành công và chứng minh rằng bất kỳ phép chia nào với số 0 sẽ dẫn đến ngoại lệ này.

Sau khi bạn học cách viết mã cho một trình xử lý Exception, ứng dụng của bạn có thể xử lý ngoại lệ này một cách dễ chịu, ví dụ: hiển thị một thông báo Toast yêu cầu người dùng thay đổi Operand 2 từ 0 thành một số khác.

---

### Thách thức 2:

Đôi khi rất khó để cô lập một đơn vị mã khỏi tất cả các phụ thuộc bên ngoài của nó. Thay vì tổ chức mã của bạn theo những cách phức tạp chỉ để dễ dàng kiểm tra hơn, bạn có thể sử dụng một khung giả lập để tạo ra các đối tượng giả ("mock") giả vờ là các phụ thuộc.

- Nghiên cứu khung Mockito và tìm hiểu cách thiết lập nó trong Android Studio.
- Viết một lớp kiểm tra cho phương thức calcButton() trong ứng dụng SimpleCalc và sử dụng Mockito để mô phỏng ngữ cảnh Android mà các bài kiểm tra của bạn sẽ chạy.

### Tóm tắt

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

- **Bài kiểm tra đơn vị cục bộ** sử dụng JVM trên máy cục bộ của bạn. Chúng không sử dụng khung Android.
- Các bài kiểm tra đơn vị được viết bằng **JUnit**, một khung kiểm tra đơn vị phổ biến cho Java.
- Các bài kiểm tra JUnit được đặt trong thư mục (**test**) trong mục **Project > Android** của Android Studio.
- Các bài kiểm tra đơn vị cục bộ chỉ cần các gói sau: **org.junit**, **org.hamcrest**, và **android.test**.
- Chú thích **@RunWith(JUnit4.class)** báo cho trình chạy kiểm tra thực thi các bài kiểm tra trong lớp này.
- Các chú thích **@SmallTest**, **@MediumTest**, và **@LargeTest** là các quy ước giúp dễ dàng nhóm các bài kiểm tra tương tự lại với nhau.
- Chú thích **@SmallTest** cho biết tất cả các bài kiểm tra trong một lớp là các bài kiểm tra đơn vị không có phụ thuộc và chạy trong vài mili-giây.

- **Bài kiểm tra có công cụ hỗ trợ (Instrumented tests)** là các bài kiểm tra chạy trên một thiết bị hoặc trình giả lập chạy Android. Các bài kiểm tra này có quyền truy cập vào khung Android.
- **Trình chạy kiểm tra (Test runner)** là một thư viện hoặc tập hợp công cụ cho phép thực hiện các bài kiểm tra và in kết quả vào nhật ký (log).

## Khái niệm liên quan

Tài liệu khái niệm liên quan nằm trong **3.2: App testing**.

## Tìm hiểu thêm

### Tài liệu Android Studio:

- [Hướng dẫn sử dụng Android Studio \(Android Studio User Guide\)](#)
- [Ghi và xem nhật ký \(Write and View Logs\)](#)

### Tài liệu nhà phát triển Android:

- [Thực hành tốt nhất cho kiểm tra \(Best Practices for Testing\)](#)
- [Bắt đầu với kiểm tra \(Getting Started with Testing\)](#)
- [Xây dựng kiểm tra đơn vị cục bộ \(Building Local Unit Tests\)](#)

### Khác:

- [Trang chủ JUnit 4 \(JUnit 4 Home Page\)](#)
- [Tài liệu API JUnit 4 \(JUnit 4 API Reference\)](#)
- [java.lang.Math](#)
- [Java Hamcrest](#)
- [Trang chủ Mockito \(Mockito Home Page\)](#)
- [Video: Android Testing Support - Testing Patterns](#)
- [Hướng dẫn lập trình kiểm tra Android \(Android Testing Codelab\)](#)
- [Protip về chú thích kích thước kiểm tra Android \(Android Tools Protip: Test Size Annotations\)](#)
- [Lợi ích của việc sử dụng assertThat thay vì các phương thức Assert khác trong kiểm tra đơn vị \(The Benefits of Using assertThat over other Assert Methods in Unit Tests\)](#)

---

## Bài tập về nhà

## Xây dựng và chạy ứng dụng

Mở ứng dụng **SimpleCalc** từ bài thực hành về sử dụng trình gõ lỗi. Bạn sẽ thêm một nút **POW** vào giao diện. Nút này thực hiện phép tính số mũ: cơ số (operand đầu tiên) được nâng lên lũy thừa bởi số mũ (operand thứ hai). Ví dụ, với các toán hạng là 5 và 4, ứng dụng sẽ tính  $5^4 = 625$ .

### Trước khi thực hiện:

Hãy xem xét các trường hợp kiểm tra bạn muốn thực hiện cho phép tính này. Những giá trị bất thường nào có thể xảy ra?

---

## Các bước thực hiện

### 1. Cập nhật lớp Calculator:

- Thêm phương thức **pow()** vào lớp **Calculator** trong ứng dụng.
- Gợi ý: Tham khảo tài liệu về lớp **java.lang.Math**.

### 2. Cập nhật lớp MainActivity:

- Kết nối nút **POW** trong giao diện với phép tính trong lớp **Calculator**.
- 

## Viết và chạy các bài kiểm tra

- Kiểm tra với các 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.

Chạy toàn bộ bài kiểm tra của bạn mỗi khi viết xong một bài kiểm tra, và sửa phép tính trong ứng dụng nếu cần.

- Một bài kiểm tra với 0 làm toán hạng thứ hai.
- Một bài kiểm tra với 0 làm toán hạng thứ nhất và -1 làm toán hạng thứ hai.

(Gợi ý: tham khảo tài liệu về **Double.POSITIVE\_INFINITY**.)

- Một bài kiểm tra với -0 làm toán hạng thứ nhất và bất kỳ số âm nào làm toán hạng thứ hai.

---

## Trả lời các câu hỏi

### Câu hỏi 1

Phát biểu nào mô tả đúng nhất về kiểm tra đơn vị cục bộ? Chọn một:

- Các bài kiểm tra chạy trên một thiết bị hoặc trình giả lập Android và có quyền truy cập vào khung làm việc Android.
- Các bài kiểm tra cho phép bạn viết các phương thức kiểm tra giao diện người dùng tự động.
- Các bài kiểm tra được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với Máy ảo Java (JVM).

### Trả lời:

- *Các bài kiểm tra được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với Máy ảo Java (JVM).*

---

### Câu hỏi 2

Tập hợp nguồn (source set) là các tập hợp mã liên quan. Trong tập hợp nguồn nào bạn có thể tìm thấy các bài kiểm tra đơn vị? Chọn một:

- app/res
- com.example.android.SimpleCalcTest
- com.example.android.SimpleCalcTest (test)
- com.example.android.SimpleCalcTest (androidTest)

### Trả lời:

- *com.example.android.SimpleCalcTest (test)*

---

### Câu hỏi 3

Chú thích nào được sử dụng để đánh dấu một phương thức là một bài kiểm tra thực tế? Chọn một:

- *@RunWith(JUnit4.class)*

- • `@SmallTest`
- • `@Before`
- • `@Test`

### Trả lời:

- • `@Test`

Nộp ứng dụng của bạn để chấm điểm

### Hướng dẫn cho người chấm điểm

Kiểm tra rằng ứng dụng có các tính năng sau:

- • Ứng dụng hiển thị nút **POW** để thực hiện phép tính mũ ("power of").
- • Việc triển khai trong **MainActivity** bao gồm trình xử lý sự kiện nhấn nút cho nút **POW**.
- • Việc triển khai trong **Calculator** bao gồm phương thức **pow()** thực hiện phép tính lũy thừa.
- • Phương thức **CalculatorTest()** có các phương thức kiểm tra riêng biệt cho phương thức **pow()** trong lớp **Calculator** để kiểm tra các trường hợp toán hạng âm, toán hạng bằng 0, và trường hợp toán hạng là 0 và -1.

## Bài học 3.3: Thư viện hỗ trợ

### Giới thiệu

Android SDK bao gồm Android Support Library, là một tập hợp nhiều thư viện. Các thư viện này cung cấp các tính năng không được tích hợp sẵn trong Android framework, bao gồm:

- Các phiên bản tương thích ngược của các thành phần framework, cho phép các ứng dụng chạy trên các phiên bản Android cũ hơn hỗ trợ các tính năng có trong các phiên bản mới hơn của nền tảng.
- Các thành phần bổ sung cho bộ cục và giao diện người dùng.
- Hỗ trợ cho các thiết bị có hình thức khác nhau, chẳng hạn như thiết bị TV hoặc thiết bị đeo được.
- Các thành phần hỗ trợ các yếu tố của Material Design.
- Các tính năng khác, bao gồm hỗ trợ bảng màu (palette support), chú thích (annotations), kích thước bộ cục dựa trên phần trăm, và tùy chọn (preferences).

### Những điều bạn nên biết trước

Bạn cần có khả năng:

- Tạo một dự án trong Android Studio.
- Sử dụng trình chỉnh sửa bộ cục để làm việc với các thành phần **EditText** và **Button**.
- Xây dựng và chạy ứng dụng của bạn trong Android Studio, cả trên trình giả lập và trên thiết bị thực.
- Điều hướng trong **Project > Android** pane trong 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.

### Bạn sẽ học được gì

- Cách kiểm tra xem **Android Support Library** có sẵn trong cài đặt Android Studio của bạn hay không.
- Cách sử dụng các lớp của thư viện hỗ trợ trong ứng dụng của bạn.

- Cách phân biệt các giá trị của **compileSdkVersion**, **targetSdkVersion**, và **minSdkVersion**.
- Cách nhận biết các API đã lỗi thời hoặc không khả dụng trong mã của bạn.
- Hiểu thêm về các thư viện hỗ trợ Android.

## Bạn sẽ làm gì

- Tạo một ứng dụng mới với một **TextView** và một **Button**.
- Kiểm tra xem **Android Support Repository** (chứa Android Support Library) có sẵn trong cài đặt Android Studio của bạn hay không.
- Khám phá các tệp **build.gradle** của dự án ứng dụng của bạn.
- Quản lý các lớp hoặc phương thức không khả dụng cho phiên bản Android mà ứng dụng của bạn hỗ trợ.
- Sử dụng một lớp tương thích từ thư viện hỗ trợ để cung cấp khả năng tương thích ngược cho ứng dụng của bạn.

### 2.3 Intent ngầm định

Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ tạo một ứng dụng có tên **HelloCompat** với một **TextView** hiển thị dòng chữ "Hello World" trên màn hình và một **Button** thay đổi màu sắc của văn bản. Có 20 màu khác nhau được định nghĩa dưới dạng tài nguyên trong tệp **color.xml**, và mỗi lần nhấn nút sẽ ngẫu nhiên chọn một trong các màu đó.

Các phương thức để lấy giá trị màu từ tài nguyên của ứng dụng đã thay đổi theo các phiên bản khác nhau của Android framework. Ví dụ này sử dụng lớp **ContextCompat** trong Android Support Library, cho phép bạn sử dụng một phương thức hoạt động trên tất cả các phiên bản.

#### Nhiệm vụ 1: Thiết lập dự án của bạn để sử dụng các thư viện hỗ trợ

Trong nhiệm vụ này, bạn sẽ thiết lập một dự án mới cho ứng dụng **HelloCompat** và triển khai bộ cục cũng như hành vi cơ bản.

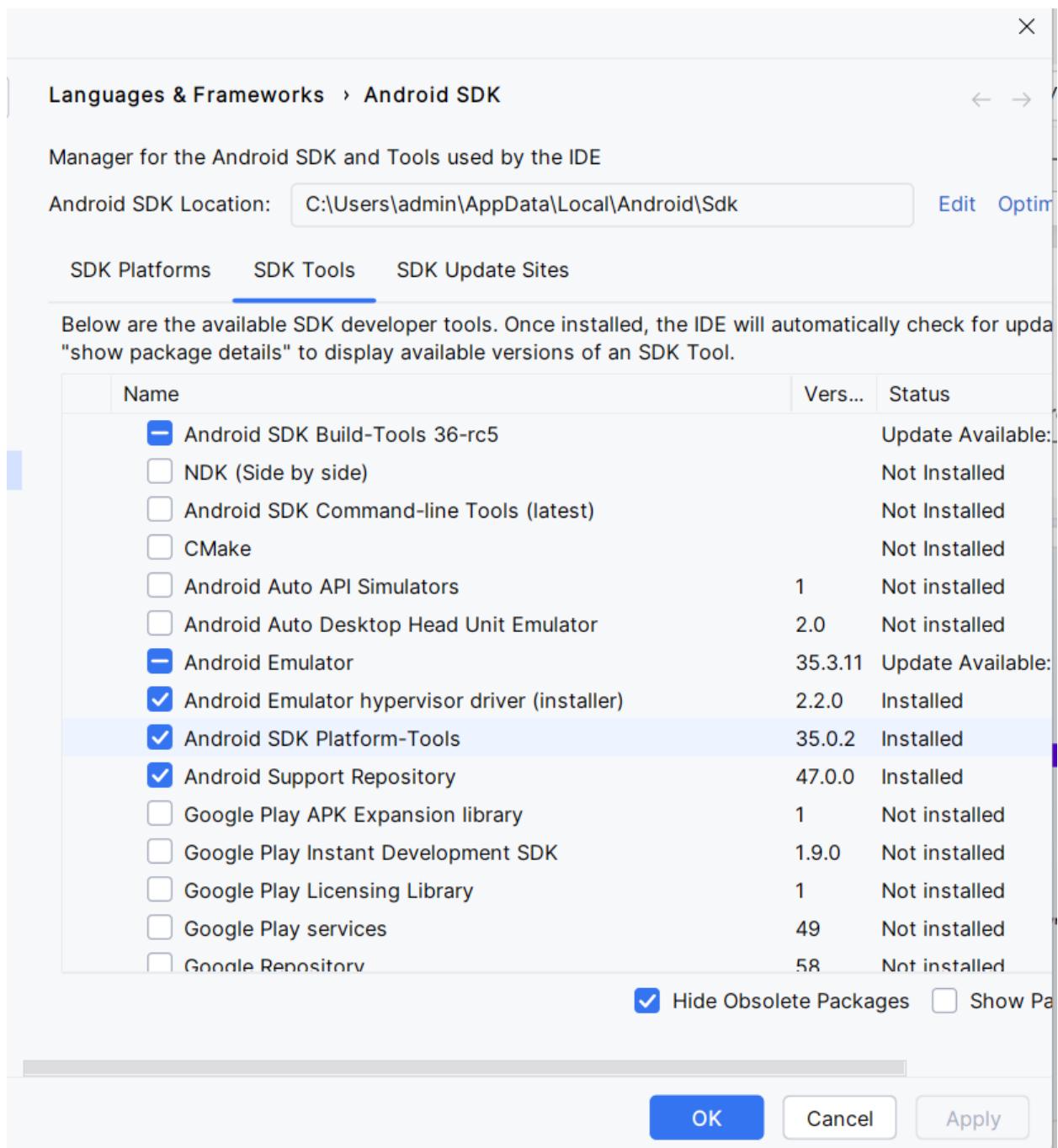
##### 1.1 Xác minh rằng Android Support Repository có sẵn

Android Support Library được tải xuống như một phần của Android SDK và có sẵn trong **Android SDK Manager**. Trong Android Studio, bạn sẽ sử dụng **Android Support Repository**—kho lưu trữ cục bộ cho các thư viện hỗ trợ—để truy cập các thư viện từ bên trong các tệp Gradle build của bạn. Trong nhiệm vụ này, bạn sẽ xác minh rằng **Android Support Repository** đã được tải xuống và sẵn sàng cho các dự án của bạn.

15.Trong Android Studio, chọn **Tools > Android > SDK Manager**, hoặc nhấp vào biểu tượng **SDK Manager**.

Bảng **Android SDK Default Preferences** sẽ xuất hiện.

16.Nhấp vào tab **SDK Tools** và mở rộng **Support Repository**, như hình minh họa bên dưới.



## 17. Tìm **Android Support Repository** trong danh sách.

- Nếu trạng thái **Installed** xuất hiện trong cột Status, bạn đã sẵn sàng. Nhập vào **Cancel**.
- Nếu trạng thái **Not installed** hoặc **Update Available** xuất hiện, hãy nhấp vào ô kiểm bên cạnh **Android Support Repository**. Một biểu tượng tải xuống sẽ xuất hiện bên cạnh ô kiểm. Nhập vào **OK**.

18.Nhấp vào **OK** một lần nữa, sau đó nhấp vào **Finish** khi lưu trữ hỗ trợ đã được cài đặt.

## 1.2 Thiết lập dự án và kiểm tra build.gradle

1. Tạo một dự án mới có tên **HelloCompat**.

Trên trang **Target Android Devices**, chọn **API 15: Android 4.0.3 (IceCreamSandwich)** làm phiên bản SDK tối thiểu. Như bạn đã học trong các bài học trước, đây là phiên bản Android cũ nhất mà ứng dụng của bạn sẽ hỗ trợ.

1. Nhấp vào **Next**, và chọn mẫu **Empty Activity**.
2. Nhấp vào **Next**, và đảm bảo rằng các tùy chọn **Generate Layout file** và **Backwards Compatibility (App Compat)** được chọn. Tùy chọn thứ hai đảm bảo rằng ứng dụng của bạn sẽ tương thích ngược với các phiên bản Android trước đó.
3. Nhấp vào **Finish**.

### Khám phá build.gradle (Module: app)

1. Trong Android Studio, đảm bảo rằng bảng **Project > Android** đang mở.
2. Mở rộng **Gradle Scripts** và mở tệp **build.gradle (Module: app)**.

Lưu ý rằng tệp **build.gradle** cho toàn bộ dự án (**build.gradle (Project: HelloCompat)**) là một tệp khác so với **build.gradle** cho mô-đun ứng dụng. Trong tệp **build.gradle (Module: app)**:

3. Tìm dòng **compileSdkVersion** gần đầu tệp. Ví dụ:

```
compileSdk = 35
```

**compileSdkVersion** là phiên bản Android framework mà ứng dụng của bạn được biên dịch trong Android Studio. Đối với các dự án mới, phiên bản compile thường là tập hợp API framework mới nhất mà bạn đã cài đặt. Giá trị này chỉ ảnh hưởng đến chính Android Studio và các cảnh báo hoặc lỗi mà bạn nhận được trong Android Studio nếu bạn sử dụng các API cũ hơn hoặc mới hơn.

4. Tìm dòng **minSdkVersion** trong phần **defaultConfig** cách vài dòng bên dưới.

minSdk = 24

**Phiên bản tối thiểu (minimum)** 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. Cửa hàng Google Play sử dụng số này để đảm bảo rằng 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.

5. Tìm dòng **targetSdkVersion** trong phần **defaultConfig**. Ví dụ:

targetSdkVersion 26

targetSdk = 35

versionCode = 1

**Phiên bản mục tiêu (target)** 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 chạy trên một thiết bị mới hơn), nền tảng có thể kích hoạt các hành vi tương thích để đảm bảo rằng ứng dụng của bạn tiếp tục hoạt động theo cách nó được thiết kế.

Ví dụ, Android 6.0 (API 23) cung cấp một mô hình cấp quyền khi chạy (runtime permissions model). Nếu ứng dụng của bạn nhắm mục tiêu đến một cấp API thấp hơn, nền tảng sẽ sử dụng lại mô hình cấp quyền khi cài đặt (install-time permissions model) cũ.

Mặc dù **target SDK** có thể giống với **compile SDK**, nhưng thường là một số thấp hơn để chỉ ra phiên bản API mới nhất mà bạn đã kiểm tra ứng dụng của mình.

6. Tìm phần **dependencies** trong tệp **build.gradle**, gần cuối tệp. Ví dụ:

```
dependencies {

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

Phần **dependencies** cho một dự án mới bao gồm nhiều thư viện phụ thuộc để hỗ trợ kiểm thử với Espresso và JUnit, cũng như thư viện hỗ trợ **appcompat v7**. Các số phiên bản của các thư viện này trong dự án của bạn có thể khác so với những số được hiển thị ở đây.

Thư viện hỗ trợ **appcompat v7** cung cấp khả năng tương thích ngược cho các phiên bản Android cũ hơn, từ API 9 trở đi. Nó cũng bao gồm cả thư viện hỗ trợ **compat v4**, vì vậy bạn không cần thêm cả hai thư viện này vào phụ thuộc.

## 7. Cập nhật số phiên bản, nếu cần.

- Nếu số phiên bản hiện tại của một thư viện thấp hơn so với phiên bản thư viện hiện có, Android Studio sẽ làm nổi bật dòng đó và cảnh báo rằng có phiên bản mới hơn ("A newer version of com.android.support:appcompat-v7 is available"). Hãy chỉnh sửa số phiên bản thành phiên bản mới nhất.
- Mẹo: Bạn cũng có thể nhập bất kỳ đâu trên dòng được đánh dấu và nhấn **Alt+Enter (Option+Return** trên Mac). Chọn **Change to xx.xx.x** từ menu, trong đó **xx.xx.x** là phiên bản mới nhất hiện có.

## 8. Cập nhật số **compileSdkVersion**, nếu cần.

- Số phiên bản chính của thư viện hỗ trợ (số đầu tiên) phải khớp với **compileSdkVersion** của bạn. Khi bạn cập nhật phiên bản thư viện hỗ trợ, bạn cũng có thể cần cập nhật **compileSdkVersion** để khớp.

## 9. Nhấp vào **Sync Now** để đồng bộ các tệp Gradle đã cập nhật với dự án, nếu được nhắc.

10. Cài đặt các tệp nền tảng SDK bị thiếu, nếu cần.

Nếu bạn cập nhật **compileSdkVersion**, bạn có thể cần cài đặt các thành phần nền tảng SDK để khớp. Nhập 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**.

### 2.1 Thay đổi bố cục và màu sắc

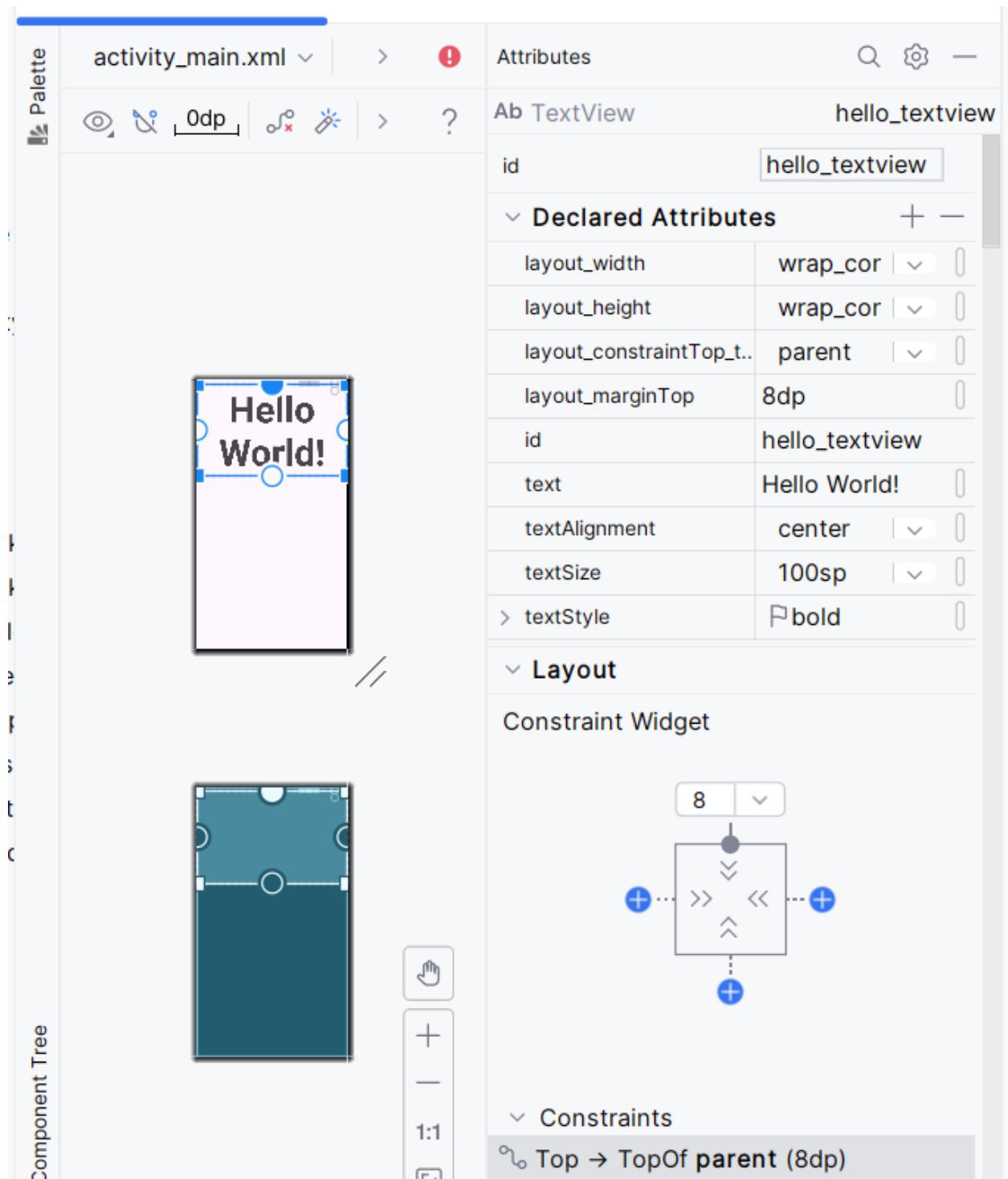
Trong nhiệm vụ này, bạn sẽ chỉnh sửa bố cục **activity\_main.xml** cho ứng dụng.

1. Mở tệp **activity\_main.xml** trong **Project > Android pane**.
2. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị trình chỉnh sửa bố cục.
3. Chọn **TextView "Hello World"** trong bố cục và mở bảng **Attributes**.
4. Thay đổi các thuộc tính của **TextView** như sau:

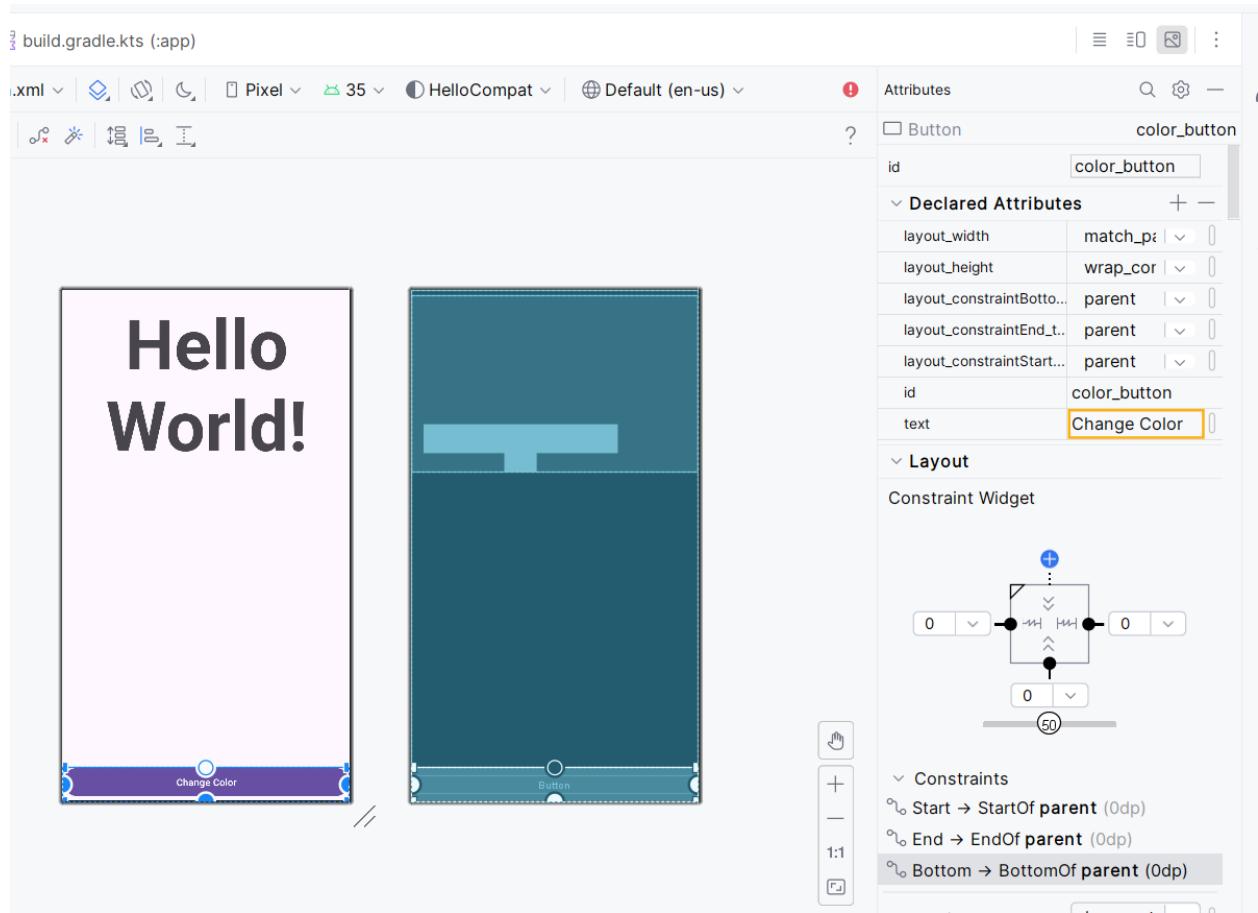
Attribute field	Enter the following:
ID	hello_textview
textStyle	B (bold)
textAlignment	Center the paragraph icon
textSize	100sp

Điều này thêm thuộc tính **android:id** vào **TextView** với **id** được đặt là **hello\_textview**, thay đổi căn chỉnh văn bản, làm cho văn bản in đậm, và đặt kích thước văn bản lớn hơn là **100sp**.

5. Xóa ràng buộc (constraint) kéo dài từ phần dưới của **hello\_textview** (**TextView**) đến phần dưới của bố cục, để **TextView** gắn lên phần trên của bố cục, và chọn **8** (8dp) cho lề trên (top margin) như hình dưới đây.



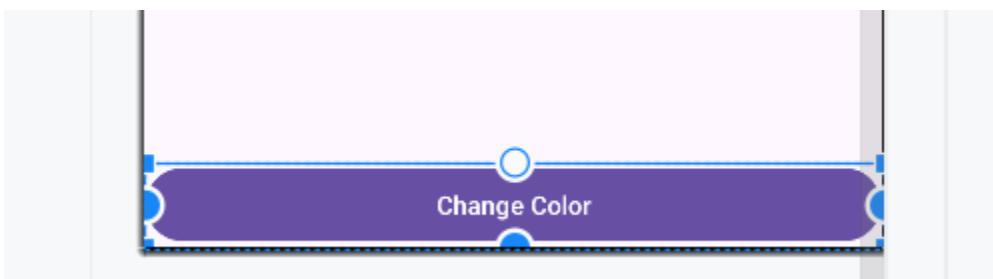
6. Kéo một **Button** vào phía dưới của bố cục, và thêm các ràng buộc (constraints) vào các cạnh trái, phải, và phía dưới của bố cục, như hiển thị trong hình dưới đây.



7. Thay đổi thuộc tính **layout\_width** trong bảng **Attributes** cho **Button** thành **match\_constraint**.
8. Thay đổi các thuộc tính khác trong bảng **Attributes** cho **Button** như sau:

<b>Attribute field</b>	<b>Enter the following:</b>
ID	color_button
text	"Change Color"

**Button** bây giờ sẽ hiển thị trong bố cục như minh họa bên dưới.



9. Trong một bài học trước, bạn đã học cách trích xuất một tài nguyên chuỗi từ một chuỗi văn bản cố định. Nhập vào tab **Text** để chuyển sang mã XML, trích xuất chuỗi "Hello Text!" và "Change Color" trong **TextView** và **Button**, và đặt tên tài nguyên chuỗi (string resource) cho chúng.

10. Thêm thuộc tính sau vào **Button**:

```
 android:onClick="changeColor"/>
```

11. Để thêm màu sắc, mở rộng **res** và **values** trong **Project > Android pane**, sau đó mở tệp **colors.xml**.

12. Thêm các tài nguyên màu sau vào tệp:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3 ■ <color name="black">#FF000000</color>
4 <color name="white">#FFFFFF</color>
5 ■ <color name="red">#FF0000</color>
6 ■ <color name="blue">#0000FF</color>
7 ■ <color name="green">#008000</color>
8 ■ <color name="yellow">#FFFF00</color>
9 ■ <color name="purple">#800080</color>
0 ■ <color name="pink">#FFC0CB</color>
1 ■ <color name="orange">#FFA500</color>
2 ■ <color name="gray">#808080</color>
3 ■ <color name="cyan">#00FFFF</color>
4 ■ <color name="magenta">#FF00FF</color>
5 ■ <color name="lime">#00FF00</color>
6 ■ <color name="maroon">#800000</color>
7 ■ <color name="navy">#000080</color>
8 ■ <color name="teal">#008080</color>
9 ■ <color name="olive">#808000</color>
0 ■ <color name="brown">#A52A2A</color>
1 ■ <color name="gold">#FFD700</color>
2 ■ <color name="beige">#F5F5DC</color>
3 ■ ! <color name="silver">#C0C0C0</color>
4
5
6 </resources>
```

Các giá trị và tên màu này đến từ bảng màu được khuyến nghị cho các ứng dụng Android được định nghĩa tại **Material Design - Style - Color**. Các mã này biểu thị giá trị RGB của màu theo hệ thập lục phân.

## 2.2 Thêm hành vi vào MainActivity

Trong nhiệm vụ này, bạn sẽ hoàn thành việc thiết lập dự án bằng cách thêm các biến **private** và triển khai các phương thức **onCreate()** và **onSaveInstanceState()**.

1. Mở **MainActivity**.
2. Thêm một biến **private** ở đầu lớp để giữ đối tượng **TextView**:

```
private TextView mHelloTextView;
```

3. Thêm mảng màu sau đây ngay sau biến **private**:

```
private String[] mColorArray = {
 "red",
 "pink",
 "deep_purple",
 "indigo",
 "blue",
 "light_blue",
 "cyan",
 "teal",
 "green",
 "light_green",
 "lime",
 "yellow",
 "amber",
 "orange",
 "deep_orange",
 "brown",
 "grey",
 "blue_grey",
 "black"
};|
```

Mỗi tên màu tương ứng với tên của một tài nguyên màu trong **color.xml**.

4. Trong phương thức **onCreate()**, sử dụng **findViewById()** để lấy tham chiếu đến đối tượng **TextView** và gán nó cho biến **private**:

```
mHelloTextView = findViewById(R.id.hello_textview);
```

1. Trong phương thức **onCreate()**, khôi phục trạng thái phiên lưu trữ, nếu có:

```
if (savedInstanceState != null) {
 mHelloTextView.setTextColor(savedInstanceState.getInt("color"));
}
```

2. Thêm phương thức **onSaveInstanceState()** vào lớp **MainActivity** để lưu màu của văn bản:

```
@Override
public void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 // lưu màu hiện tại của văn bản
 outState.putInt("color", mHelloTextView.getCurrentTextColor());
}
```

## Mã giải pháp cho Nhiệm vụ 2

Dưới đây là mã giải pháp cho bộ cục XML và một đoạn mã trong lớp **MainActivity** của ứng dụng **HelloCompat** cho đến thời điểm này.

Tệp bộ cục **activity\_main.xml** như sau. Trình xử lý **changeColor** cho thuộc tính **android:onClick** của nút **Button** đang được gạch đỏ vì nó chưa được định nghĩa. Bạn sẽ định nghĩa nó trong nhiệm vụ tiếp theo.

```
<?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/hello_textview"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/hello_world"
 android:textAlignment="center"
 android:textSize="100sp"
 android:textStyle="bold"
 android:layout_marginTop="8dp"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintLeft_toLeftOf="parent"
 app:layout_constraintRight_toRightOf="parent"/>

<Button
 android:id="@+id/color_button"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Change Color"
 android:layout_marginTop="8dp"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintLeft_toLeftOf="parent"
 app:layout_constraintRight_toRightOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 android:onClick="changeColor"/>

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

## Lớp MainActivity

Lớp **MainActivity** bao gồm các biến private sau ở đầu lớp:

```
private TextView mHelloTextView;
no usages
private String[] mColorArray = {
 "red",
 "pink",
 "deep_purple",
 "indigo",
 "blue",
 "light_blue",
 "cyan",
 "teal",
 "green",
 "light_green",
 "lime",
 "yellow",
 "amber",
 "orange",
 "deep_orange",
 "brown",
 "grey",
 "blue_grey",
 "black"
};
```

**Phương thức onCreate() và onSaveInstanceState() trong MainActivity**

```

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 EdgeToEdge.enable($this$enableEdgeToEdge: this);
 setContentView(R.layout.activity_main);
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
 return insets;
 });
 mHelloTextView = findViewById(R.id.hello_textview);
 if (savedInstanceState != null) {
 mHelloTextView.setTextColor(savedInstanceState.getInt(key: "color"));
 }
}

@Override
public void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 // lưu màu hiện tại của văn bản
 outState.putInt("color", mHelloTextView.getCurrentTextColor());
}

```

### Nhiệm vụ 3: Thực hiện hành vi cho Button

Nút **Change Color** trong ứng dụng **HelloCompat** chọn ngẫu nhiên một trong 20 màu từ tệp tài nguyên color.xml và đặt màu của văn bản thành màu đó. Trong nhiệm vụ này, bạn sẽ triển khai hành vi cho trình xử lý sự kiện khi nhấn Button.

#### 2.1 Thêm trình xử lý sự kiện changeButton()

1. Mở tệp activity\_main.xml nếu chưa mở. Chuyển sang tab **Text** để hiển thị mã XML.
2. Nhập vào "changeColor" trong thuộc tính android:onClick của phần tử Button.
3. Nhấn **Alt+Enter** (hoặc **Option+Enter** trên Mac) và chọn **Create onClick event handler**.
4. Chọn **MainActivity** và nhập **OK**.

Điều này sẽ tạo một phương thức mẫu cho changeColor() trong MainActivity:

1 usage

```
public void changeColor(View view) {
}
```

## 2.2 Triển khai hành động của Button

1. Chuyển sang tệp MainActivity.
2. Trong phương thức changeColor(), tạo một đối tượng số ngẫu nhiên bằng cách sử dụng lớp Random (một lớp trong Java):

```
Random random = new Random();
```

3. Sử dụng đối tượng random để chọn một màu ngẫu nhiên từ mảng mColorArray:

```
String colorName = mColorArray[random.nextInt(bound: 20)];
```

Phương thức nextInt() với tham số 20 sẽ trả về một số nguyên ngẫu nhiên trong khoảng từ 0 đến 19. Số nguyên này sẽ được dùng làm chỉ số để truy cập vào mảng và lấy tên màu.

4. Lấy mã định danh tài nguyên (một số nguyên) cho tên màu từ tài nguyên:

```
int colorResourceName = getResources().getIdentifier(colorName, "color",
getApplicationContext().getPackageName());
```

Khi ứng dụng của bạn được biên dịch, hệ thống Android chuyển đổi các định nghĩa trong các tệp XML của bạn thành các tài nguyên với các mã định danh nội bộ dạng số nguyên (ID). Có các ID riêng biệt cho cả tên và giá trị. Dòng mã này ánh xạ các chuỗi màu từ mảng colorName với các ID tên màu tương ứng trong tệp tài nguyên XML. Phương thức getResources() lấy tất cả các tài nguyên của ứng dụng, trong khi phương thức getIdentifier() tìm kiếm tên màu (chuỗi) trong tài nguyên màu ("color") của tên gói hiện tại.

5. **Lấy mã số nguyên cho màu thực tế từ các tài nguyên và gán nó cho biến colorRes, sau đó sử dụng phương thức getTheme() để lấy giao diện chủ đề cho ngữ cảnh ứng dụng hiện tại.**

```
int colorRes = getResources().getColor(colorResourceName, this.getTheme());
```

Phương thức getResources() lấy tập hợp các tài nguyên cho ứng dụng của bạn, và phương thức getColor() truy xuất một màu cụ thể từ các tài nguyên đó thông qua ID của tên màu. Tuy nhiên, getColor() bị gạch chân màu đỏ.

Nếu bạn trỏ chuột vào getColor(), Android Studio sẽ báo lỗi: "Call requires API 23 (current min is 15)". Vì minSdkVersion của bạn là 15, bạn sẽ nhận được thông báo này khi cố gắng sử dụng bất kỳ API nào được giới thiệu sau API 15. Bạn vẫn có thể biên dịch ứng dụng của mình, nhưng vì phiên bản getColor() này không khả dụng trên các thiết bị chạy API trước 23, ứng dụng của bạn sẽ bị lỗi khi người dùng nhấn nút **Change Color**.

Ở giai đoạn này, bạn có thể kiểm tra phiên bản nền tảng và sử dụng phiên bản phù hợp của getColor() tùy thuộc vào nơi ứng dụng đang chạy. Một cách tốt hơn để hỗ trợ cả các API Android cũ hơn và mới hơn mà không gặp cảnh báo là sử dụng một trong các lớp tương thích trong thư viện hỗ trợ.

## 6. Thay đổi dòng gán colorRes để sử dụng lớp ContextCompat:

```
int colorRes = ContextCompat.getColor(context: this, colorResourceName);
```

Lớp ContextCompat cung cấp nhiều phương thức hỗ trợ tương thích để xử lý sự khác biệt giữa các phiên bản API trong ngữ cảnh ứng dụng và tài nguyên của ứng dụng. Phương thức getColor() trong ContextCompat nhận hai đối số: ngữ cảnh hiện tại (trong trường hợp này là instance của Activity, tức this) và tên của màu sắc.

Việc triển khai phương thức này trong thư viện hỗ trợ ẩn đi sự khác biệt về cách thực hiện giữa các phiên bản API khác nhau. Bạn có thể gọi phương thức này mà không gặp bất kỳ cảnh báo, lỗi, hoặc sự cố nào, bất kể phiên bản SDK biên dịch hoặc phiên bản SDK tối thiểu của bạn là gì.

### 1. Đặt màu cho TextView bằng ID tài nguyên màu:

```
mHelloTextView.setTextColor(colorRes);
```

### 2. Chạy ứng dụng trên thiết bị hoặc trình giả lập, sau đó nhấn nút Change Color.

Nút **Change Color** bây giờ sẽ thay đổi màu văn bản trong ứng dụng, như minh họa bên dưới.



cessfully finished in 719 ms

Mã giải pháp

### Giải pháp cho MainActivity

Dưới đây là trình xử lý sự kiện **changeColor()** trong **MainActivity**:

```
public void changeColor(View view) {
 Random random = new Random();
 String colorName = mColorArray[random.nextInt(bound: 20)];
 int colorResourceName = getResources().getIdentifier(colorName, defType: "color", getApplicationContext().getPackageName());
 int colorRes = ContextCompat.getColor(context: this, colorResourceName);
 mHelloTextView.setTextColor(colorRes);
}
```

## Dự án Android Studio

### Dự án Android Studio: HelloCompat

#### Thử thách lập trình

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

**Thử thách:** Thay vì sử dụng **ContextCompat** để lấy tài nguyên màu, hãy sử dụng kiểm tra giá trị trong lớp **Build** để thực hiện một thao tác khác nếu ứng dụng đang chạy trên thiết bị hỗ trợ phiên bản Android cũ hơn API 23.

#### Tóm tắt

#### Cài đặt Android Support Library:

- Sử dụng **SDK Manager** để cài đặt **Android Support Repository**. Chọn **Tools > Android > SDK Manager**, nhấp vào tab **SDK Tools** và mở rộng **Support Repository**.
- Nếu cột **Status** hiển thị **Installed** cho **Android Support Repository**, nhấp vào **Cancel**; nếu hiển thị **Not installed** hoặc **Update Available**, đánh dấu vào ô kiểm. Một biểu tượng tải xuống sẽ xuất hiện bên cạnh ô kiểm. Nhấp vào **OK**.

**Android sử dụng ba chỉ thị để chỉ định cách ứng dụng của bạn hoạt động 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 được biên dịch.
- targetSdkVersion:** phiên bản API mà ứng dụng của bạn được thiết kế để chạy.

#### Quản lý dependencies trong dự án:

- Mở rộng **Gradle Scripts** trong mục **Project > Android**, và mở tệp **build.gradle (Module: app)**.

- Bạn có thể thêm dependencies trong phần **dependencies**.

**Lớp ContextCompat** cung cấp các phương thức hỗ trợ tương thích với ngữ cảnh và các phương thức liên quan đến tài nguyên cho cả API cũ và mới.

### **Khái niệm liên quan**

Tài liệu về khái niệm liên quan nằm trong **3.3: The Android Support Library**.

### **Tìm hiểu thêm**

Tài liệu Android Studio:

- **Android Studio User Guide**  
Tài liệu nhà phát triển Android:
- **Android Support Library (introduction)**
- **Support Library Setup**
- **Support Library Features**
- **Supporting Different Platform Versions**
- **Package Index** (tất cả các gói API bắt đầu bằng android.support).

**Khác:**

- **Picking your compileSdkVersion, minSdkVersion, and targetSdkVersion**
- **Understanding the Android Support Library**
- **All the Things Compat**

### **Bài tập về nhà**

#### **Chạy ứng dụng**

Mở ứng dụng **HelloCompat** mà bạn đã tạo trong bài thực hành về cách sử dụng các thư viện hỗ trợ.

1. Đặt một điểm dừng gỡ lỗi (debugger breakpoint) trên dòng mã trong phương thức changeColor() nơi thực sự thay đổi màu sắc:

java

CopyEdit

```
int colorRes = ContextCompat.getColor(this, colorResourceName);
```

2. Chạy ứng dụng ở chế độ debug trên một thiết bị hoặc trình giả lập (emulator) đang chạy phiên bản API 23 hoặc mới hơn. Nhấp vào **Step Into** để bước vào phương thức getColor() và theo dõi các lời gọi phương thức sâu hơn trong ngăn xếp.
  - Kiểm tra cách lớp ContextCompat xác định phương pháp lấy màu từ tài nguyên và các lớp framework khác mà nó sử dụng.
  - Một số lớp có thể hiển thị cảnh báo rằng "mã nguồn không khớp với bytecode." Nhấp vào **Step Out** để quay lại một tệp mã nguồn đã biết hoặc tiếp tục nhấp vào **Step Into** cho đến khi trình gõ lỗi tự quay trở lại.
3. Lặp lại bước trước đó trên một thiết bị hoặc trình giả lập chạy phiên bản API cũ hơn 23.
  - Ghi nhận các đường dẫn khác nhau mà framework thực hiện để lấy màu.

### Câu hỏi 1

Khi bạn bước vào lần đầu tiên bằng cách nhấp **Step Into** phương thức ContextCompat.getColor(), lớp nào sẽ xuất hiện? Chọn một:

- **MainActivity**
- **ContextCompat**
- **AppCompatActivity**
- **Context**

### Câu hỏi 2

Trong lớp xuất hiện, câu lệnh nào được thực thi nếu phiên bản API là 23 hoặc mới hơn? Chọn một:

- **return context.getColor(id);**
- **return context.getResources().getColor(id);**
- **throw new IllegalArgumentException("permission is null");**
- **return mResources == null ? super.getResources() : mResources;**

### Câu hỏi 3

Nếu bạn thay đổi phương thức ContextCompat.getColor() thành phương thức getColor(), điều gì sẽ xảy ra khi bạn chạy ứng dụng? Chọn một:

- Nếu minSdkVersion của bạn là 15, từ getColor sẽ bị gạch chân màu đỏ trong trình chỉnh sửa mã. Khi trỏ chuột vào, Android Studio sẽ báo: "Call requires API 23 (current min is 15)".
- Ứng dụng sẽ chạy mà không gặp lỗi trên các trình giả lập và thiết bị sử dụng API 23 hoặc mới hơn.
- Ứng dụng sẽ bị sập khi người dùng nhấn Change Color nếu trình giả lập hoặc thiết bị đang sử dụng API 17.
- Tất cả các điều trên.

Nộp ứng dụng của bạn để chấm điểm

#### Hướng dẫn cho người chấm điểm

Không có ứng dụng nào cần nộp cho bài tập về nhà này.

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

### 3.1 Trình gỡ lỗi

#### Giới thiệu

Kiểm thử mã của bạn có thể giúp bạn phát hiện lỗi sớm trong quá trình phát triển, khi chi phí xử lý lỗi là thấp nhất. Khi ứng dụng của bạn trở nên lớn hơn và phức tạp hơn, việc kiểm thử cải thiện độ tin cậy của mã.

Với các bài kiểm thử trong mã, bạn có thể kiểm tra từng phần nhỏ trong ứng dụng một cách độc lập, và bạn có thể kiểm thử theo cách tự động hóa và lặp lại được.

Android Studio và **Android Testing Support Library** hỗ trợ nhiều loại kiểm thử và khung kiểm thử khác nhau. Trong bài thực hành này, bạn sẽ khám phá chức năng kiểm thử tích hợp sẵn của Android Studio và học cách viết và chạy các bài kiểm thử đơn vị cục bộ.

**Kiểm thử đơn vị cục bộ** là các bài kiểm thử được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với **Java Virtual Machine (JVM)**. Bạn sử dụng kiểm thử đơn vị cục bộ để kiểm tra các phần của ứng dụng không cần truy cập vào **Android framework** hoặc thiết bị/emulator Android, chẳng hạn như logic nội bộ.

Bạn cũng có thể sử dụng kiểm thử đơn vị cục bộ để kiểm tra các phần của ứng dụng mà bạn có thể tạo ra các đối tượng giả ("mock" hoặc "stub") để bắt chước hành vi của các thành phần tương đương trong framework.

Kiểm thử đơn vị được viết bằng **JUnit**, một khung kiểm thử đơn vị phổ biến dành cho Java.

---

## Những gì bạn cần biết trước

Bạn cần phải:

- Tạo một dự án Android Studio.
- Xây dựng và chạy ứng dụng của bạn trong Android Studio, trên cả trình giả lập và thiết bị thực.
- Điều hướng trong **Project > Android pane** của Android Studio.

### • Tìm các thành phần chính của một dự án Android Studio, bao gồm:

- **AndroidManifest.xml**,
- Các tài nguyên (resources),
- Các tệp Java,
- Các tệp Gradle.

---

## Bạn sẽ học được gì?

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

---

## Bạn sẽ làm gì?

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

## Tổng quan về ứng dụng

Bài thực hành này sử dụng ứng dụng **SimpleCalc** từ bài codelab thực hành trước (**Android fundamentals 3.1: The debugger**). Bạn có thể chỉnh sửa ứng dụng này tại chỗ hoặc tạo một bản sao thư mục dự án trước khi tiếp tục.

---

### 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) trong **Android Studio**, cùng với mã 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 để sử dụng cho mục đích riêng của mình.

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

#### 1.1 Khám phá bộ mã nguồn (source sets) và CalculatorTest

**Source sets** là các tập hợp mã trong dự án của bạn dùng cho các mục tiêu build khác nhau hoặc các "phiên bản" khác nhau của ứng dụng. Khi Android Studio tạo dự án của bạn, nó sẽ tạo ra ba bộ mã nguồn:

- **Bộ mã nguồn chính (main source set)**: Chứa mã và tài nguyên chính của ứng dụng.
  - **Bộ mã nguồn (test)**: Chứa các bài kiểm thử đơn vị cục bộ của ứng dụng. Bộ mã nguồn này hiển thị (**test**) sau tên gói.
  - **Bộ mã nguồn (androidTest)**: Chứa các bài kiểm thử có công cụ (instrumented tests) dành cho Android. Bộ mã nguồn này hiển thị (**androidTest**) sau tên gói.
- 

**Trong nhiệm vụ này, bạn sẽ khám phá cách các bộ mã nguồn được hiển thị trong Android Studio, kiểm tra cấu hình Gradle dành cho việc kiểm thử, và chạy các bài kiểm thử đơn vị cho ứng dụng SimpleCalc.**

**Lưu ý:** Bộ mã nguồn (**androidTest**) đã được loại bỏ khỏi ví dụ này để đơn giản hóa. Nó sẽ được giải thích chi tiết hơn trong một bài học khác.

---

## Các bước thực hiện

3. Mở dự án **SimpleCalc** trong Android Studio nếu bạn chưa làm điều này.
4. Mở bảng điều hướng **Project > Android**, sau đó mở rộng các thư mục **app** và **java**.
  - o Thư mục **java** trong chế độ xem Android hiển thị tất cả các bộ mã nguồn trong ứng dụng theo tên gói.
  - o Trong trường hợp này (như minh họa bên dưới), mã của ứng dụng nằm trong bộ mã nguồn **com.android.example.SimpleCalc**.
  - o Mã kiểm thử nằm trong bộ mã nguồn có chữ **test** xuất hiện trong ngoặc đơn sau tên gói: **com.android.example.SimpleCalc (test)**.

## 2. Mở rộng thư mục com.android.example.SimpleCalc (test)

Thư mục này là nơi bạn đặt các bài kiểm thử đơn vị cục bộ của ứng dụng. Android Studio tạo sẵn một lớp kiểm thử mẫu trong thư mục này cho các dự án mới, nhưng với ứng dụng **SimpleCalc**, lớp kiểm thử được gọi là **CalculatorTest**.

---

### 1. Mở tệp CalculatorTest

Hãy kiểm tra mã nguồn và lưu ý những điểm sau:

- **Các thư viện được nhập:**  
Chỉ có các thư viện từ các gói **org.junit**, **org.hamcrest**, và **android.test** được nhập. Không có sự phụ thuộc nào vào các lớp của framework Android.
- **Chú thích @RunWith(JUnit4.class):**  
Chú thích này chỉ định **runner** sẽ được sử dụng để chạy các bài kiểm thử trong lớp này. **Test runner** là một thư viện hoặc bộ công cụ cho phép thực hiện kiểm thử và in kết quả ra log. Với các bài kiểm thử cần thiết lập hoặc hàn tầng phức tạp hơn (như Espresso), bạn sẽ sử dụng các test runner khác. Trong ví dụ này, chúng ta sử dụng **JUnit4 test runner** cơ bản.
- **Chú thích @SmallTest:**  
Chú thích này cho biết tất cả các bài kiểm thử trong lớp là kiểm thử đơn vị, không có phụ thuộc nào và chạy trong thời gian rất ngắn (tính bằng mili giây). Các chú thích **@SmallTest**, **@MediumTest**, và **@LargeTest** là các quy ước giúp dễ dàng nhóm các bài kiểm thử thành các bộ với chức năng tương tự nhau.

- **Phương thức setUp():**  
Phương thức này được dùng để thiết lập môi trường trước khi kiểm thử và có chú thích **@Before**. Trong trường hợp này, phương thức **setUp()** tạo một instance mới của lớp **Calculator** và gán nó vào biến thành viên **mCalculator**.
- **Phương thức addTwoNumbers():**  
Đây là một bài kiểm thử thực tế và được chú thích bằng **@Test**. Chỉ những phương thức trong lớp kiểm thử có chú thích **@Test** mới được trình kiểm thử (test runner) xem là bài kiểm thử. Theo quy ước, tên của phương thức kiểm thử không bao gồm từ "test."
- **Dòng lệnh đầu tiên của addTwoNumbers():**  
Dòng lệnh này gọi phương thức **add()** từ lớp **Calculator**. Bạn chỉ có thể kiểm thử các phương thức có phạm vi truy cập là **public** hoặc **package-protected**. Trong trường hợp này, lớp **Calculator** là lớp **public** với các phương thức **public**, nên không có vấn đề gì.
- **Dòng lệnh thứ hai:**  
Đây là một biểu thức khẳng định (assertion) cho bài kiểm thử. **Assertion** là các biểu thức phải đánh giá và trả về **true** để bài kiểm thử được thông qua. Trong trường hợp này, **assertion** kiểm tra xem kết quả nhận được từ phương thức **add()** ( $1 + 1$ ) có khớp với số được chỉ định là 2 hay không. Bạn sẽ tìm hiểu thêm về cách tạo các **assertion** sau trong bài thực hành này.

## 1.2 Chạy kiểm thử trong Android Studio

Trong phần này, bạn sẽ chạy các bài kiểm thử đơn vị trong thư mục kiểm thử và xem kết quả cho cả kiểm thử thành công và thất bại.

---

### Các bước thực hiện:

4. Trong ngăn **Project > Android**, nhấp chuột phải (hoặc Control-click) vào **CalculatorTest** và chọn **Run 'CalculatorTest'**.
  - Dự án sẽ được xây dựng (nếu cần thiết), và ngăn **CalculatorTest** sẽ xuất hiện ở dưới cùng của màn hình.
  - Ở đầu ngăn này, danh sách thả xuống cho các cấu hình thực thi khả dụng cũng thay đổi thành **CalculatorTest**.
5. Tất cả các bài kiểm thử trong lớp **CalculatorTest** sẽ được chạy.
  - Nếu các bài kiểm thử thành công, thanh tiến trình ở đầu khung nhìn sẽ chuyển sang màu **xanh lá cây**.

- Một thông báo trạng thái ở cuối màn hình sẽ báo "**Tests Passed**".
6. Mở **CalculatorTest** nếu nó chưa được mở, và thay đổi biểu thức khẳng định trong phương thức **addTwoNumbers()** thành:

java

CopyEdit

```
assertThat(resultAdd, is(equalTo(3d)));
```

## 2. Chạy lại bài kiểm thử với cấu hình CalculatorTest

4. Trong danh sách thả xuống cấu hình chạy (run configurations) ở phía trên màn hình, chọn **CalculatorTest** (nếu nó chưa được chọn) và nhấn **Run**.
  - Bài kiểm thử sẽ chạy lại như trước, nhưng lần này biểu thức khẳng định thất bại (**3 không bằng 1 + 1**).
  - Thanh tiến trình trong khung chạy kiểm thử chuyển sang màu **đỏ**, và nhật ký kiểm thử chỉ ra vị trí bài kiểm thử (biểu thức khẳng định) thất bại cũng như lý do tại sao.
5. Thay đổi biểu thức khẳng định trong phương thức **addTwoNumbers()** quay lại bài kiểm thử đúng, và chạy lại kiểm thử để đảm bảo tất cả bài kiểm thử đều thành công.
6. Trong danh sách thả xuống cấu hình chạy, chọn **app** để chạy ứng dụng bình thường.

---

### 3.2 Kiểm thử đơn vị

Kiểm thử đơn vị là kiểm tra một phần nhỏ trong mã nguồn của bạn (chẳng hạn một phương thức hoặc một lớp), và tách phần đó khỏi phần còn lại của ứng dụng. Điều này giúp đảm bảo rằng phần mã nhỏ đó hoạt động đúng như mong đợi.

Thông thường, một kiểm thử đơn vị gọi một phương thức với nhiều đầu vào khác nhau, và kiểm tra rằng phương thức hoạt động như mong đợi và trả về kết quả đúng.

Trong nhiệm vụ này, bạn sẽ viết thêm các kiểm thử đơn vị cho các phương thức tiện ích của lớp **Calculator** trong ứng dụng **SimpleCalc**, và chạy các bài kiểm thử để đảm bảo rằng chúng tạo ra đầu ra như bạn mong đợi.

---

## 2.1 Thêm các bài kiểm thử khác cho phương thức add()

Mặc dù không thể kiểm thử mọi giá trị đầu vào mà phương thức **add()** có thể nhận, nhưng việc kiểm tra các trường hợp đầu vào đặc biệt là rất quan trọng. Ví dụ:

- Đầu vào có chứa **toán hạng âm**.
- Đầu vào có chứa **số thực (floating-point numbers)**.
- **Số rất lớn (exceptionally large numbers)**
- **Các loại toán hạng khác nhau (float và double)**
- **Toán hạng bằng 0 (zero)**
- **Toán hạng là vô cực (infinity)**

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

### 2. Thêm phương thức mới vào CalculatorTest có tên là **addTwoNumbersNegative()**.

Sử dụng khung mã như sau:

java

CopyEdit

@Test

```
public void addTwoNumbersNegative() {
}
```

Phương thức kiểm thử này có cấu trúc tương tự như **addTwoNumbers()**:

- Là phương thức **public**, không có tham số, và trả về kiểu **void**.
  - Được chú thích bằng **@Test**, cho biết đây là một bài kiểm thử đơn lẻ.
-

### 3. Tại sao không thêm nhiều lệnh kiểm tra (assertions) vào addTwoNumbers()?

- Gom nhiều lệnh kiểm tra vào một phương thức có thể làm cho bài kiểm thử khó gỡ lỗi nếu chỉ một lệnh kiểm tra thất bại.
  - Ngoài ra, việc này cũng làm mờ các bài kiểm thử thành công khác.
  - Quy tắc chung cho các bài kiểm thử đơn vị là **tạo một phương thức kiểm thử cho từng lệnh kiểm tra riêng lẻ**.
- 

### 4. Chạy tất cả các bài kiểm thử trong CalculatorTest như trước.

- Trong cửa sổ kiểm thử, cả addTwoNumbers và addTwoNumbersNegative đều được liệt kê dưới dạng các bài kiểm thử khả dụng (và đang thành công) trong bảng điều khiển bên trái.
  - Bài kiểm thử addTwoNumbersNegative vẫn thành công ngay cả khi không chứa bất kỳ đoạn mã nào—một bài kiểm thử không làm gì vẫn được xem là thành công.
- 

### 5. Thêm một dòng mã vào addTwoNumbersNegative() để gọi phương thức add() trong lớp Calculator với một toán hạng âm.

java

CopyEdit

```
double resultAdd = mCalculator.add(1d, 2d);
```

**Ký hiệu d sau mỗi toán hạng cho biết đây là các số kiểu double.**

Do phương thức add() được định nghĩa với các tham số kiểu double, các kiểu dữ liệu khác như float hoặc int cũng có thể hoạt động. Việc chỉ rõ kiểu giúp bạn kiểm tra riêng biệt các kiểu khác nếu cần thiết.

---

### 4. Thêm một lệnh kiểm tra (assertion) với assertThat().

java

CopyEdit

```
assertThat(resultAdd, is(equalTo(1d)));
```

- Phương thức assertThat() là một lệnh kiểm tra trong JUnit4, khẳng định biểu thức trong đối số đầu tiên bằng với biểu thức trong đối số thứ hai.
  - Các phiên bản cũ của JUnit sử dụng các phương thức kiểm tra cụ thể hơn như assertEquals(), assertNull(), hoặc assertTrue(), nhưng assertThat() linh hoạt hơn, dễ đọc và dễ gỡ lỗi hơn.
- 

### **assertThat() sử dụng matchers.**

- Matchers là các phương thức liên kết được gọi trong đối số thứ hai của lệnh kiểm tra này, chẳng hạn như is(equalTo()).
- Framework Hamcrest định nghĩa các matchers sẵn có để xây dựng các lệnh kiểm tra. (Tên "Hamcrest" là một phép đảo chữ từ "matchers.")
- Hamcrest cung cấp nhiều matchers cơ bản cho hầu hết các lệnh kiểm tra. Bạn cũng có thể tự định nghĩa matchers của riêng mình cho các lệnh kiểm tra phức tạp hơn.

#### **Ví dụ:**

Trong trường hợp này, lệnh kiểm tra xác nhận rằng kết quả của phép cộng add() (-1 + 2) bằng với 1.

---

### **5. Thêm bài kiểm tra đơn vị mới vào CalculatorTest cho số dấu phẩy động:**

java

CopyEdit

@Test

```
public void addTwoNumbersFloats() {

 double resultAdd = mCalculator.add(1.111f, 1.111d);

 assertThat(resultAdd, is(equalTo(2.222d)));
}
```

}

- Đây là một bài kiểm tra rất giống với phương thức kiểm tra trước, nhưng một đối số của phương thức add() được chỉ rõ là kiểu float thay vì double.
  - Phương thức add() được định nghĩa với các tham số kiểu double, vì vậy bạn có thể gọi nó với kiểu float. Khi đó, số kiểu float sẽ được chuyển đổi thành kiểu double.
- 

## 6. Nhấn vào Run để chạy lại tất cả các bài kiểm tra.

Lần này bài kiểm tra thất bại và thanh tiến trình chuyển sang màu đỏ. Đây là phần quan trọng trong thông báo lỗi:

makefile

CopyEdit

java.lang.AssertionError:

Expected: is <2.222>

but: was <2.2219999418258665>

Tính toán với các số dấu phẩy động không chính xác, và việc chuyển đổi đã gây ra hiệu ứng phụ về độ chính xác.

Lệnh kiểm tra trong bài kiểm tra này 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 do việc chuyển đổi tham số kiểu float, đó có phải là vấn đề của mã nguồn hay bài kiểm tra?

- Trong trường hợp này, cả hai đối số đầu vào của phương thức add() từ ứng dụng SimpleCalc đều luôn thuộc kiểu double, do đó đây là một bài kiểm tra tùy ý và không thực tế.
- Tuy nhiên, nếu ứng dụng của bạn được viết sao cho đầu vào của phương thức add() có thể là kiểu double hoặc float, và bạn chỉ quan tâm đến một

mức độ chính xác nhất định, thì bạn cần thêm một biên độ "gần đúng" để bài kiểm tra thành công.

---

## 7. Thay đổi phương thức assertThat() để sử dụng matcher closeTo():

java

CopyEdit

```
assertThat(resultAdd, is(closeTo(2.222, 0.01)));
```

### Hướng dẫn:

- Bạn cần chọn matcher thích hợp. Nhập vào closeTo hai lần (cho đến khi toàn bộ biểu thức được gạch chân) và nhấn **Alt+Enter** (**Option+Return** trên Mac).
  - Chọn **isCloseTo.closeTo (org.hamcrest.number)**.
- 

## 8. Nhấn vào Run để chạy lại tất cả các bài kiểm tra.

Lần này bài kiểm tra đã thành công.

**Với matcher closeTo(), thay vì kiểm tra sự bằng nhau tuyệt đối, bạn có thể kiểm tra sự bằng nhau trong một phạm vi delta nhất định.**

- Trong trường hợp này, phương thức matcher closeTo() nhận hai tham số: giá trị mong đợi và giá trị delta.
- Trong ví dụ trên, delta là phạm vi gần đúng với độ chính xác hai chữ số thập phân.

## 2.2 Thêm các bài kiểm tra đơn vị cho các phương thức tính toán khác

Sử dụng những gì bạn đã học trong nhiệm vụ trước để hoàn thiện các bài kiểm tra đơn vị cho lớp Calculator.

7. Thêm một bài kiểm tra đơn vị có tên subTwoNumbers() để kiểm tra phương thức sub().

8. Thêm một bài kiểm tra đơn vị có tên subWorksWithNegativeResults() để kiểm tra phương thức sub() khi phép tính cho ra kết quả âm.
9. Thêm một bài kiểm tra đơn vị có tên multTwoNumbers() để kiểm tra phương thức mul().
10. Thêm một bài kiểm tra đơn vị có tên multTwoNumbersZero() để kiểm tra phương thức mul() khi ít nhất một tham số là số 0.
11. Thêm một bài kiểm tra đơn vị có tên divTwoNumbers() để kiểm tra phương thức div() với hai tham số khác 0.
12. Thêm một bài kiểm tra đơn vị có tên divTwoNumbersZero() để kiểm tra phương thức div() với một số chia kiểu double và số chia là 0.

Tất cả các bài kiểm tra trên đều phải thành công, ngoại trừ divTwoNumbersZero(), bài này sẽ gây ra ngoại lệ tham số không hợp lệ khi chia cho 0.

- Nếu bạn chạy ứng dụng, nhập 0 làm Số hạng 2 và nhấn Div để chia, kết quả sẽ là lỗi.

Giải pháp mã cho nhiệm vụ 2:

### **Dự án Android Studio: SimpleCalcTest**

Đoạn mã sau đây minh họa các bài kiểm tra cho nhiệm vụ này:

```
public void addTwoNumbers() {
 double resultAdd = mCalculator.add(1d, 1d);
 assertThat(resultAdd, is(equalTo(2d)));
}

@Test
public void addTwoNumbersNegative() {
 double resultAdd = mCalculator.add(-1d, 2d);
 assertThat(resultAdd, is(equalTo(1d)));
}

@Test
public void addTwoNumbersFloats() {
 double resultAdd = mCalculator.add(1.111f, 1.111d);
 assertThat(resultAdd, is(closeTo(2.222, 0.01)));
}

@Test
public void subTwoNumbers() {
 double resultSub = mCalculator.sub(1d, 1d);
 assertThat(resultSub, is(equalTo(0d)));
}

@Test
public void subWorksWithNegativeResult() {
 double resultSub = mCalculator.sub(1d, 17d);
 assertThat(resultSub, is(equalTo(-16d)));
}

@Test
public void multTwoNumbers() {
 double resultMul = mCalculator.mul(32d, 2d);
 assertThat(resultMul, is(equalTo(64d)));
}

@Test
public void divTwoNumbers() {
 double resultDiv = mCalculator.div(32d, 2d);
 assertThat(resultDiv, is(equalTo(16d)));
}

@Test
public void divTwoNumbersZero() {
 double resultDiv = mCalculator.div(32d, 0);
 assertThat(resultDiv, is(equalTo(Double.POSITIVE_INFINITY)));
}
```

```

@Test
public void subWorksWithNegativeResult() {
 double resultSub = mCalculator.sub(1d, 17d);
 assertThat(resultSub, is(equalTo(-16d)));
}

@Test
public void multTwoNumbers() {
 double resultMul = mCalculator.mul(32d, 2d);
 assertThat(resultMul, is(equalTo(64d)));
}

@Test
public void divTwoNumbers() {
 double resultDiv = mCalculator.div(32d, 2d);
 assertThat(resultDiv, is(equalTo(16d)));
}

@Test
public void divTwoNumbersZero() {
 double resultDiv = mCalculator.div(32d, 0);
 assertThat(resultDiv, is(equalTo(Double.POSITIVE_INFINITY)));
}

```

---

## Thách thức lập trình

**Lưu ý:** Tất cả các thách thức lập trình đều tùy chọn và không phải là yêu cầu bắt buộc cho các bài học sau.

---

### Thách thức 1:

Chia cho 0 luôn là một trường hợp đặc biệt trong toán học và đáng để kiểm tra. Làm thế nào bạn có thể thay đổi ứng dụng để xử lý chia cho 0 một cách dễ chịu hơn?

Để thực hiện thách thức này, hãy bắt đầu bằng một bài kiểm tra cho thấy hành vi đúng là gì.

- Xóa phương thức divTwoNumbersZero() khỏi lớp CalculatorTest và thêm một bài kiểm tra đơn vị mới có tên là divByZeroThrows() để kiểm tra phương thức div() với đối số thứ hai bằng 0, với kết quả mong đợi là IllegalArgumentException.class.

- Bài kiểm tra này sẽ thành công và chứng minh rằng bất kỳ phép chia nào với số 0 sẽ dẫn đến ngoại lệ này.

Sau khi bạn học cách viết mã cho một trình xử lý Exception, ứng dụng của bạn có thể xử lý ngoại lệ này một cách dễ chịu, ví dụ: hiển thị một thông báo Toast yêu cầu người dùng thay đổi Operand 2 từ 0 thành một số khác.

---

### Thách thức 2:

Đôi khi rất khó để cô lập một đơn vị mã khỏi tất cả các phụ thuộc bên ngoài của nó. Thay vì tổ chức mã của bạn theo những cách phức tạp chỉ để dễ dàng kiểm tra hơn, bạn có thể sử dụng một khung giả lập để tạo ra các đối tượng giả ("mock") giả vờ là các phụ thuộc.

- Nghiên cứu khung Mockito và tìm hiểu cách thiết lập nó trong Android Studio.
- Viết một lớp kiểm tra cho phương thức calcButton() trong ứng dụng SimpleCalc và sử dụng Mockito để mô phỏng ngữ cảnh Android mà các bài kiểm tra của bạn sẽ chạy.

### Tóm tắt

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

- **Bài kiểm tra đơn vị cục bộ** sử dụng JVM trên máy cục bộ của bạn. Chúng không sử dụng khung Android.
- Các bài kiểm tra đơn vị được viết bằng **JUnit**, một khung kiểm tra đơn vị phổ biến cho Java.
- Các bài kiểm tra JUnit được đặt trong thư mục (**test**) trong mục **Project > Android** của Android Studio.
- Các bài kiểm tra đơn vị cục bộ chỉ cần các gói sau: **org.junit**, **org.hamcrest**, và **android.test**.
- Chú thích **@RunWith(JUnit4.class)** báo cho trình chạy kiểm tra thực thi các bài kiểm tra trong lớp này.
- Các chú thích **@SmallTest**, **@MediumTest**, và **@LargeTest** là các quy ước giúp dễ dàng nhóm các bài kiểm tra tương tự lại với nhau.
- Chú thích **@SmallTest** cho biết tất cả các bài kiểm tra trong một lớp là các bài kiểm tra đơn vị không có phụ thuộc và chạy trong vài mili-giây.

- **Bài kiểm tra có công cụ hỗ trợ (Instrumented tests)** là các bài kiểm tra chạy trên một thiết bị hoặc trình giả lập chạy Android. Các bài kiểm tra này có quyền truy cập vào khung Android.
- **Trình chạy kiểm tra (Test runner)** là một thư viện hoặc tập hợp công cụ cho phép thực hiện các bài kiểm tra và in kết quả vào nhật ký (log).

## Khái niệm liên quan

Tài liệu khái niệm liên quan nằm trong **3.2: App testing**.

## Tìm hiểu thêm

### Tài liệu Android Studio:

- [Hướng dẫn sử dụng Android Studio \(Android Studio User Guide\)](#)
- [Ghi và xem nhật ký \(Write and View Logs\)](#)

### Tài liệu nhà phát triển Android:

- [Thực hành tốt nhất cho kiểm tra \(Best Practices for Testing\)](#)
- [Bắt đầu với kiểm tra \(Getting Started with Testing\)](#)
- [Xây dựng kiểm tra đơn vị cục bộ \(Building Local Unit Tests\)](#)

### Khác:

- [Trang chủ JUnit 4 \(JUnit 4 Home Page\)](#)
- [Tài liệu API JUnit 4 \(JUnit 4 API Reference\)](#)
- [java.lang.Math](#)
- [Java Hamcrest](#)
- [Trang chủ Mockito \(Mockito Home Page\)](#)
- [Video: Android Testing Support - Testing Patterns](#)
- [Hướng dẫn lập trình kiểm tra Android \(Android Testing Codelab\)](#)
- [Protip về chú thích kích thước kiểm tra Android \(Android Tools Protip: Test Size Annotations\)](#)
- [Lợi ích của việc sử dụng assertThat thay vì các phương thức Assert khác trong kiểm tra đơn vị \(The Benefits of Using assertThat over other Assert Methods in Unit Tests\)](#)

---

## Bài tập về nhà

## Xây dựng và chạy ứng dụng

Mở ứng dụng **SimpleCalc** từ bài thực hành về sử dụng trình gõ lỗi. Bạn sẽ thêm một nút **POW** vào giao diện. Nút này thực hiện phép tính số mũ: cơ số (operand đầu tiên) được nâng lên lũy thừa bởi số mũ (operand thứ hai). Ví dụ, với các toán hạng là 5 và 4, ứng dụng sẽ tính  $5^4 = 625$ .

### Trước khi thực hiện:

Hãy xem xét các trường hợp kiểm tra bạn muốn thực hiện cho phép tính này. Những giá trị bất thường nào có thể xảy ra?

---

## Các bước thực hiện

### 3. Cập nhật lớp Calculator:

- Thêm phương thức **pow()** vào lớp **Calculator** trong ứng dụng.
- Gợi ý: Tham khảo tài liệu về lớp **java.lang.Math**.

### 4. Cập nhật lớp MainActivity:

- Kết nối nút **POW** trong giao diện với phép tính trong lớp **Calculator**.
- 

## Viết và chạy các bài kiểm tra

- Kiểm tra với các 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.

Chạy toàn bộ bài kiểm tra của bạn mỗi khi viết xong một bài kiểm tra, và sửa phép tính trong ứng dụng nếu cần.

- Một bài kiểm tra với 0 làm toán hạng thứ hai.
- Một bài kiểm tra với 0 làm toán hạng thứ nhất và -1 làm toán hạng thứ hai.

(Gợi ý: tham khảo tài liệu về **Double.POSITIVE\_INFINITY**.)

- Một bài kiểm tra với -0 làm toán hạng thứ nhất và bất kỳ số âm nào làm toán hạng thứ hai.

---

## Trả lời các câu hỏi

### Câu hỏi 1

Phát biểu nào mô tả đúng nhất về kiểm tra đơn vị cục bộ? Chọn một:

- Các bài kiểm tra chạy trên một thiết bị hoặc trình giả lập Android và có quyền truy cập vào khung làm việc Android.
- Các bài kiểm tra cho phép bạn viết các phương thức kiểm tra giao diện người dùng tự động.
- Các bài kiểm tra được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với Máy ảo Java (JVM).

### Trả lời:

- *Các bài kiểm tra được biên dịch và chạy hoàn toàn trên máy cục bộ của bạn với Máy ảo Java (JVM).*

---

### Câu hỏi 2

Tập hợp nguồn (source set) là các tập hợp mã liên quan. Trong tập hợp nguồn nào bạn có thể tìm thấy các bài kiểm tra đơn vị? Chọn một:

- app/res
- com.example.android.SimpleCalcTest
- com.example.android.SimpleCalcTest (test)
- com.example.android.SimpleCalcTest (androidTest)

### Trả lời:

- *com.example.android.SimpleCalcTest (test)*

---

### Câu hỏi 3

Chú thích nào được sử dụng để đánh dấu một phương thức là một bài kiểm tra thực tế? Chọn một:

- *@RunWith(JUnit4.class)*

- • `@SmallTest`
- • `@Before`
- • `@Test`

### Trả lời:

- • `@Test`

Nộp ứng dụng của bạn để chấm điểm

### Hướng dẫn cho người chấm điểm

Kiểm tra rằng ứng dụng có các tính năng sau:

- • Ứng dụng hiển thị nút **POW** để thực hiện phép tính mũ ("power of").
- • Việc triển khai trong **MainActivity** bao gồm trình xử lý sự kiện nhấn nút cho nút **POW**.
- • Việc triển khai trong **Calculator** bao gồm phương thức **pow()** thực hiện phép tính lũy thừa.
- • Phương thức **CalculatorTest()** có các phương thức kiểm tra riêng biệt cho phương thức **pow()** trong lớp **Calculator** để kiểm tra các trường hợp toán hạng âm, toán hạng bằng 0, và trường hợp toán hạng là 0 và -1.

### 3.3 Thư viện hỗ trợ

#### Bài học 3.3: Thư viện hỗ trợ

##### Giới thiệu

Android SDK bao gồm Android Support Library, là một tập hợp nhiều thư viện. Các thư viện này cung cấp các tính năng không được tích hợp sẵn trong Android framework, bao gồm:

- Các phiên bản tương thích ngược của các thành phần framework, cho phép các ứng dụng chạy trên các phiên bản Android cũ hơn hỗ trợ các tính năng có trong các phiên bản mới hơn của nền tảng.
- Các thành phần bổ sung cho bộ cục và giao diện người dùng.
- Hỗ trợ cho các thiết bị có hình thức khác nhau, chẳng hạn như thiết bị TV hoặc thiết bị đeo được.
- Các thành phần hỗ trợ các yếu tố của Material Design.
- Các tính năng khác, bao gồm hỗ trợ bảng màu (palette support), chú thích (annotations), kích thước bộ cục dựa trên phần trăm, và tùy chọn (preferences).

## Những điều bạn nên biết trước

Bạn cần có khả năng:

- Tạo một dự án trong Android Studio.
- Sử dụng trình chỉnh sửa bô cục để làm việc với các thành phần **EditText** và **Button**.
- Xây dựng và chạy ứng dụng của bạn trong Android Studio, cả trên trình giả lập và trên thiết bị thực.
- Điều hướng trong **Project > Android** pane trong 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.

## Bạn sẽ học được gì

- Cách kiểm tra xem **Android Support Library** có sẵn trong cài đặt Android Studio của bạn hay không.
- Cách sử dụng các lớp của thư viện hỗ trợ trong ứng dụng của bạn.
- Cách phân biệt các giá trị của **compileSdkVersion**, **targetSdkVersion**, và **minSdkVersion**.
- Cách nhận biết các API đã lỗi thời hoặc không khả dụng trong mã của bạn.
- Hiểu thêm về các thư viện hỗ trợ Android.

## Bạn sẽ làm gì

- Tạo một ứng dụng mới với một **TextView** và một **Button**.
- Kiểm tra xem **Android Support Repository** (chứa Android Support Library) có sẵn trong cài đặt Android Studio của bạn hay không.
- Khám phá các tệp **build.gradle** của dự án ứng dụng của bạn.
- Quản lý các lớp hoặc phương thức không khả dụng cho phiên bản Android mà ứng dụng của bạn hỗ trợ.

- Sử dụng một lớp tương thích từ thư viện hỗ trợ để cung cấp khả năng tương thích ngược cho ứng dụng của bạn.

Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ tạo một ứng dụng có tên **HelloCompat** với một **TextView** hiển thị dòng chữ "Hello World" trên màn hình và một **Button** thay đổi màu sắc của văn bản. Có 20 màu khác nhau được định nghĩa dưới dạng tài nguyên trong tệp **color.xml**, và mỗi lần nhấn nút sẽ ngẫu nhiên chọn một trong các màu đó.

Các phương thức để lấy giá trị màu từ tài nguyên của ứng dụng đã thay đổi theo các phiên bản khác nhau của Android framework. Ví dụ này sử dụng lớp **ContextCompat** trong Android Support Library, cho phép bạn sử dụng một phương thức hoạt động trên tất cả các phiên bản.

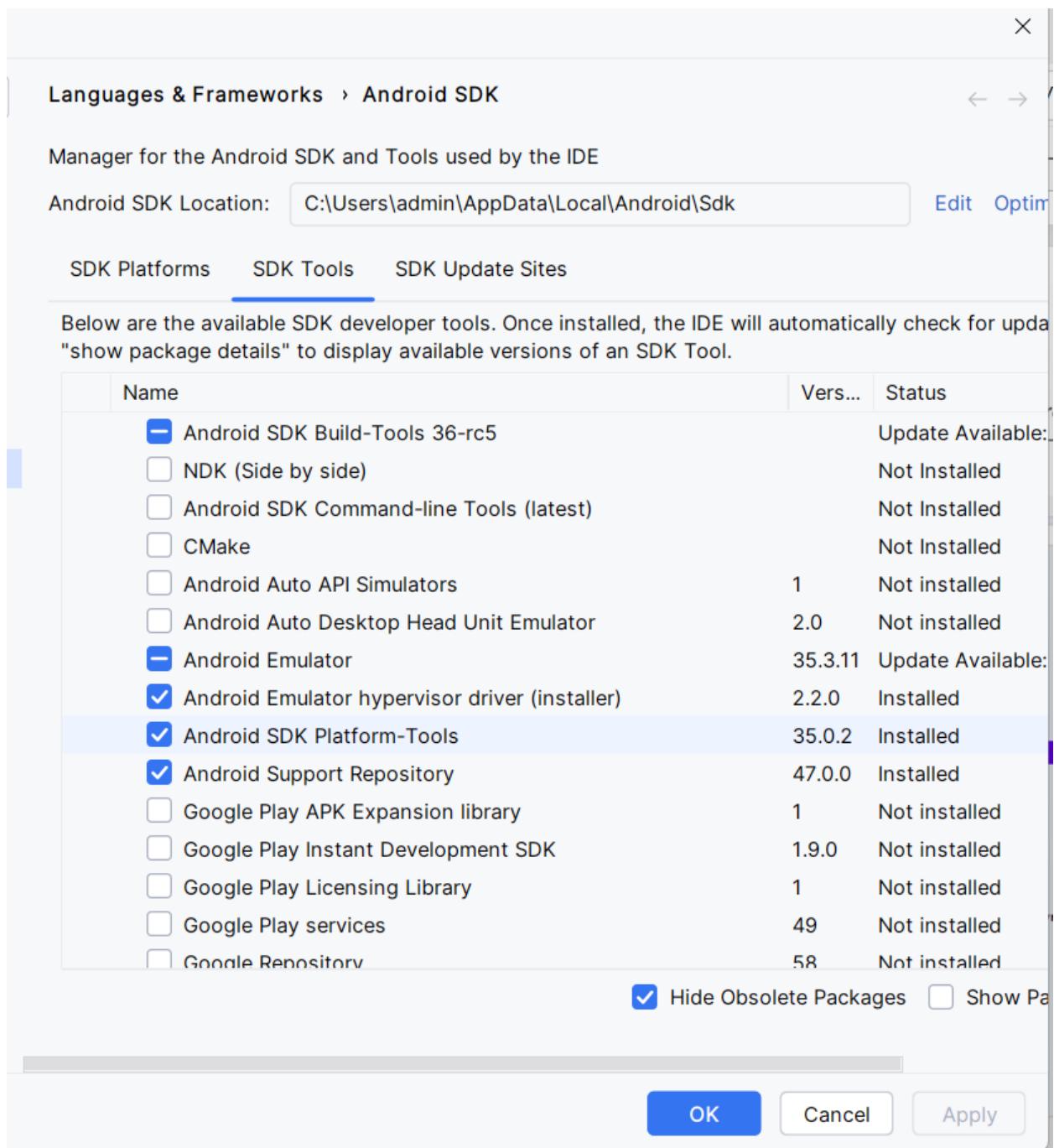
### Nhiệm vụ 1: Thiết lập dự án của bạn để sử dụng các thư viện hỗ trợ

Trong nhiệm vụ này, bạn sẽ thiết lập một dự án mới cho ứng dụng **HelloCompat** và triển khai bộ cục cũn như hành vi cơ bản.

#### 1.1 Xác minh rằng Android Support Repository có sẵn

Android Support Library được tải xuống như một phần của Android SDK và có sẵn trong **Android SDK Manager**. Trong Android Studio, bạn sẽ sử dụng **Android Support Repository**—kho lưu trữ cục bộ cho các thư viện hỗ trợ—để truy cập các thư viện từ bên trong các tệp Gradle build của bạn. Trong nhiệm vụ này, bạn sẽ xác minh rằng **Android Support Repository** đã được tải xuống và sẵn sàng cho các dự án của bạn.

- 17.Trong Android Studio, chọn **Tools > Android > SDK Manager**, hoặc nhấp vào biểu tượng **SDK Manager**.  
Bảng **Android SDK Default Preferences** sẽ xuất hiện.
- 18.Nhấp vào tab **SDK Tools** và mở rộng **Support Repository**, như hình minh họa bên dưới.



## 18. Tìm **Android Support Repository** trong danh sách.

- Nếu trạng thái **Installed** xuất hiện trong cột Status, bạn đã sẵn sàng. Nhập vào **Cancel**.
- Nếu trạng thái **Not installed** hoặc **Update Available** xuất hiện, hãy nhấp vào ô kiểm bên cạnh **Android Support Repository**. Một biểu tượng tải xuống sẽ xuất hiện bên cạnh ô kiểm. Nhập vào **OK**.

19. Nhấp vào **OK** một lần nữa, sau đó nhấp vào **Finish** khi lưu trữ hỗ trợ đã được cài đặt.

## 1.2 Thiết lập dự án và kiểm tra build.gradle

2. Tạo một dự án mới có tên **HelloCompat**.

Trên trang **Target Android Devices**, chọn **API 15: Android 4.0.3 (IceCreamSandwich)** làm phiên bản SDK tối thiểu. Như bạn đã học trong các bài học trước, đây là phiên bản Android cũ nhất mà ứng dụng của bạn sẽ hỗ trợ.

4. Nhấp vào **Next**, và chọn mẫu **Empty Activity**.

5. Nhấp vào **Next**, và đảm bảo rằng các tùy chọn **Generate Layout file** và **Backwards Compatibility (App Compat)** được chọn. Tùy chọn thứ hai đảm bảo rằng ứng dụng của bạn sẽ tương thích ngược với các phiên bản Android trước đó.

6. Nhấp vào **Finish**.

### Khám phá build.gradle (Module: app)

3. Trong Android Studio, đảm bảo rằng bảng **Project > Android** đang mở.

4. Mở rộng **Gradle Scripts** và mở tệp **build.gradle (Module: app)**.

Lưu ý rằng tệp **build.gradle** cho toàn bộ dự án (**build.gradle (Project: HelloCompat)**) là một tệp khác so với **build.gradle** cho mô-đun ứng dụng. Trong tệp **build.gradle (Module: app)**:

4. Tìm dòng **compileSdkVersion** gần đầu tệp. Ví dụ:

`compileSdk = 35`

**compileSdkVersion** là phiên bản Android framework mà ứng dụng của bạn được biên dịch trong Android Studio. Đối với các dự án mới, phiên bản compile thường là tập hợp API framework mới nhất mà bạn đã cài đặt. Giá trị này chỉ ảnh hưởng đến chính Android Studio và các cảnh báo hoặc lỗi mà bạn nhận được trong Android Studio nếu bạn sử dụng các API cũ hơn hoặc mới hơn.

5. Tìm dòng **minSdkVersion** trong phần **defaultConfig** cách vài dòng bên dưới.

minSdk = 24

**Phiên bản tối thiểu (minimum)** 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. Cửa hàng Google Play sử dụng số này để đảm bảo rằng 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.

6. Tìm dòng **targetSdkVersion** trong phần **defaultConfig**. Ví dụ:

targetSdkVersion 26

targetSdk = 35

versionCode = 1

**Phiên bản mục tiêu (target)** 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 chạy trên một thiết bị mới hơn), nền tảng có thể kích hoạt các hành vi tương thích để đảm bảo rằng ứng dụng của bạn tiếp tục hoạt động theo cách nó được thiết kế.

Ví dụ, Android 6.0 (API 23) cung cấp một mô hình cấp quyền khi chạy (runtime permissions model). Nếu ứng dụng của bạn nhắm mục tiêu đến một cấp API thấp hơn, nền tảng sẽ sử dụng lại mô hình cấp quyền khi cài đặt (install-time permissions model) cũ.

Mặc dù **target SDK** có thể giống với **compile SDK**, nhưng thường là một số thấp hơn để chỉ ra phiên bản API mới nhất mà bạn đã kiểm tra ứng dụng của mình.

7. Tìm phần **dependencies** trong tệp **build.gradle**, gần cuối tệp. Ví dụ:

```
dependencies {

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

Phần **dependencies** cho một dự án mới bao gồm nhiều thư viện phụ thuộc để hỗ trợ kiểm thử với Espresso và JUnit, cũng như thư viện hỗ trợ **appcompat v7**. Các số phiên bản của các thư viện này trong dự án của bạn có thể khác so với những số được hiển thị ở đây.

Thư viện hỗ trợ **appcompat v7** cung cấp khả năng tương thích ngược cho các phiên bản Android cũ hơn, từ API 9 trở đi. Nó cũng bao gồm cả thư viện hỗ trợ **compat v4**, vì vậy bạn không cần thêm cả hai thư viện này vào phụ thuộc.

#### 8. Cập nhật số phiên bản, nếu cần.

- Nếu số phiên bản hiện tại của một thư viện thấp hơn so với phiên bản thư viện hiện có, Android Studio sẽ làm nổi bật dòng đó và cảnh báo rằng có phiên bản mới hơn ("A newer version of com.android.support:appcompat-v7 is available"). Hãy chỉnh sửa số phiên bản thành phiên bản mới nhất.
- Mẹo: Bạn cũng có thể nhập bất kỳ đâu trên dòng được đánh dấu và nhấn **Alt+Enter (Option+Return** trên Mac). Chọn **Change to xx.xx.x** từ menu, trong đó **xx.xx.x** là phiên bản mới nhất hiện có.

#### 9. Cập nhật số **compileSdkVersion**, nếu cần.

- Số phiên bản chính của thư viện hỗ trợ (số đầu tiên) phải khớp với **compileSdkVersion** của bạn. Khi bạn cập nhật phiên bản thư viện hỗ trợ, bạn cũng có thể cần cập nhật **compileSdkVersion** để khớp.

#### 10. Nhấp vào **Sync Now** để đồng bộ các tệp Gradle đã cập nhật với dự án, nếu được nhắc.

11.Cài đặt các tệp nền tảng SDK bị thiếu, nếu cần.

Nếu bạn cập nhật **compileSdkVersion**, bạn có thể cần cài đặt các thành phần nền tảng SDK để khớp. 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**.

### 2.1 Thay đổi bố cục và màu sắc

Trong nhiệm vụ này, bạn sẽ chỉnh sửa bố cục **activity\_main.xml** cho ứng dụng.

9. Mở tệp **activity\_main.xml** trong **Project > Android pane**.

10.Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị trình chỉnh sửa bố cục.

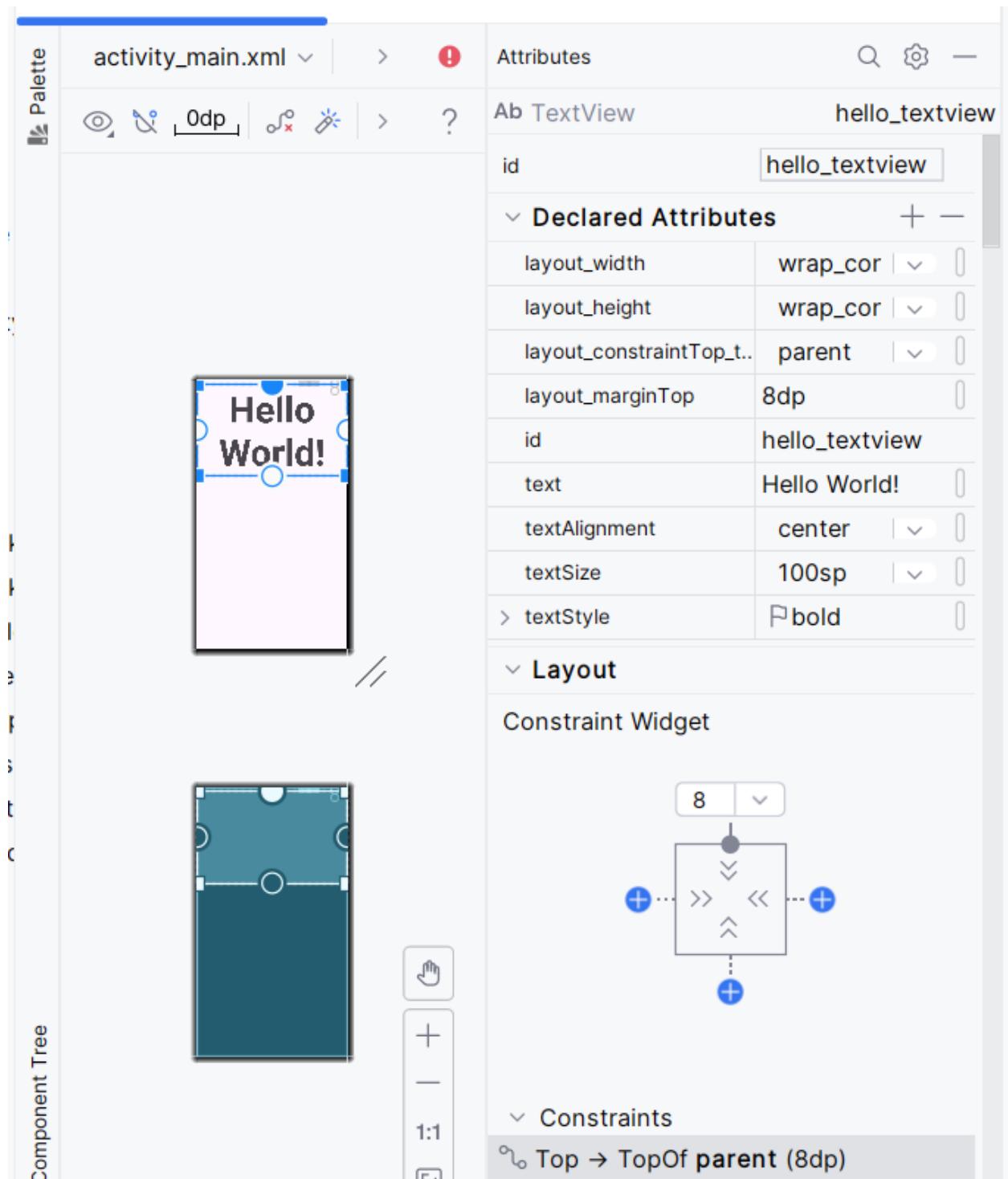
11.Chọn **TextView "Hello World"** trong bố cục và mở bảng **Attributes**.

12.Thay đổi các thuộc tính của **TextView** như sau:

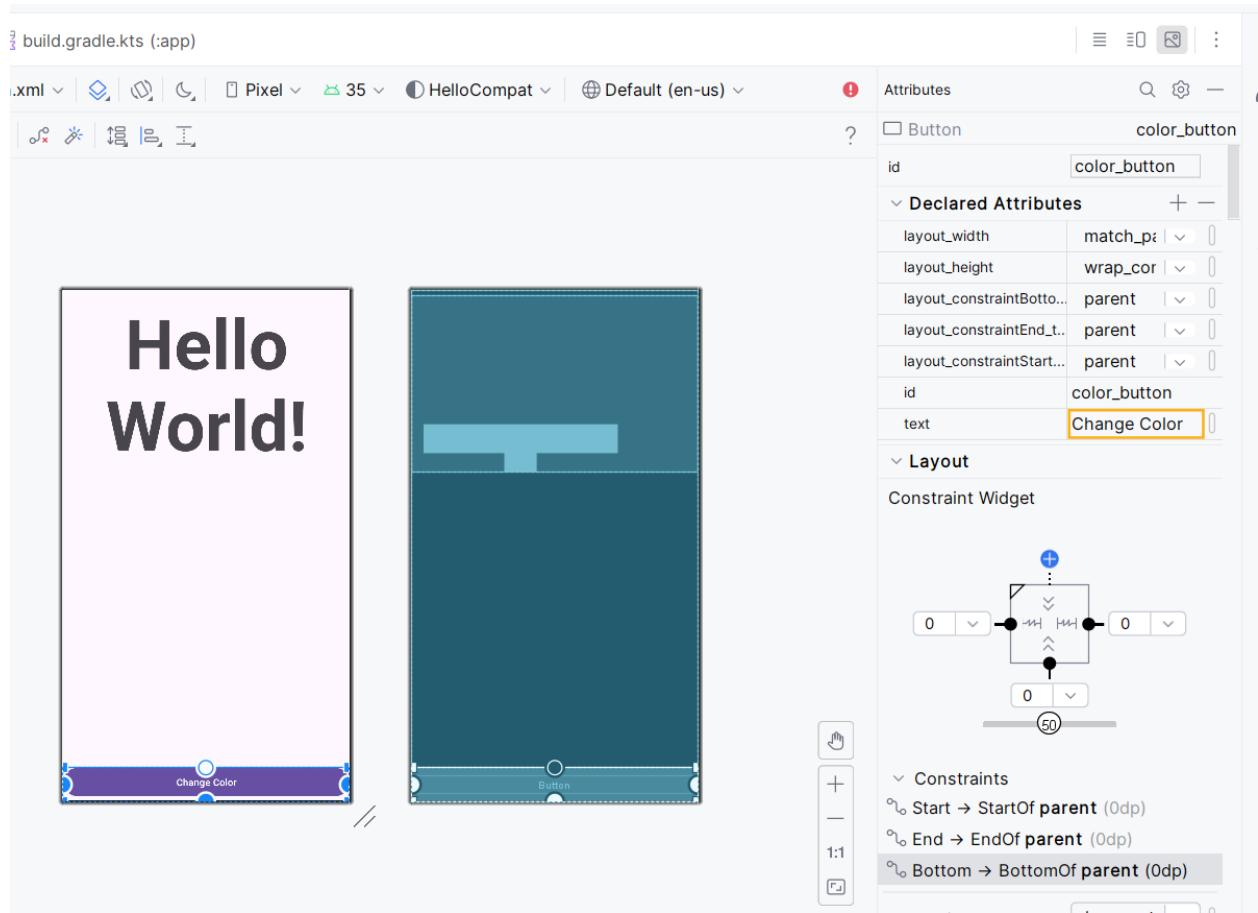
Attribute field	Enter the following:
ID	hello_textview
textStyle	B (bold)
textAlignment	Center the paragraph icon
textSize	100sp

Điều này thêm thuộc tính **android:id** vào **TextView** với **id** được đặt là **hello\_textview**, thay đổi căn chỉnh văn bản, làm cho văn bản in đậm, và đặt kích thước văn bản lớn hơn là **100sp**.

13.Xóa ràng buộc (constraint) kéo dài từ phần dưới của **hello\_textview** (**TextView**) đến phần dưới của bố cục, để **TextView** gắn lên phần trên của bố cục, và chọn **8** (8dp) cho lề trên (top margin) như hình dưới đây.



14. Kéo một **Button** vào phía dưới của bố cục, và thêm các ràng buộc (constraints) vào các cạnh trái, phải, và phía dưới của bố cục, như hiển thị trong hình dưới đây.

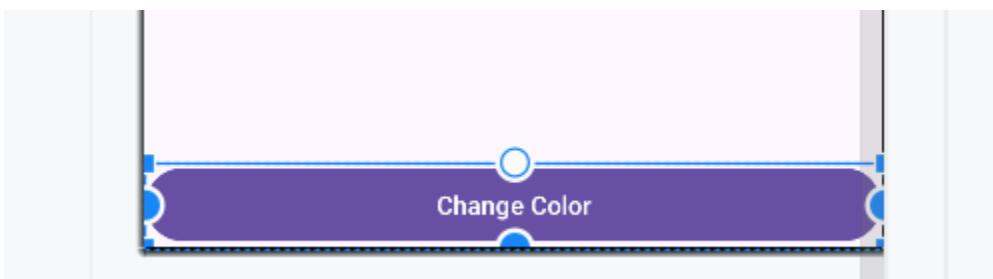


15.Thay đổi thuộc tính **layout\_width** trong bảng **Attributes** cho **Button** thành **match\_constraint**.

16.Thay đổi các thuộc tính khác trong bảng **Attributes** cho **Button** như sau:

<b>Attribute field</b>	<b>Enter the following:</b>
ID	color_button
text	"Change Color"

**Button** bây giờ sẽ hiển thị trong bố cục như minh họa bên dưới.



11.Trong một bài học trước, bạn đã học cách trích xuất một tài nguyên chuỗi từ một chuỗi văn bản cố định. Nhấp vào tab **Text** để chuyển sang mã XML, trích xuất chuỗi "Hello Text!" và "Change Color" trong **TextView** và **Button**, và đặt tên tài nguyên chuỗi (string resource) cho chúng.

12.Thêm thuộc tính sau vào **Button**:

```
 android:onClick="changeColor"/>
```

13.Để thêm màu sắc, mở rộng **res** và **values** trong **Project > Android pane**, sau đó mở tệp **colors.xml**.

14.Thêm các tài nguyên màu sau vào tệp:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3 ■ <color name="black">#FF000000</color>
4 <color name="white">#FFFFFF</color>
5 ■ <color name="red">#FF0000</color>
6 ■ <color name="blue">#0000FF</color>
7 ■ <color name="green">#008000</color>
8 ■ <color name="yellow">#FFFF00</color>
9 ■ <color name="purple">#800080</color>
0 ■ <color name="pink">#FFC0CB</color>
1 ■ <color name="orange">#FFA500</color>
2 ■ <color name="gray">#808080</color>
3 ■ <color name="cyan">#00FFFF</color>
4 ■ <color name="magenta">#FF00FF</color>
5 ■ <color name="lime">#00FF00</color>
6 ■ <color name="maroon">#800000</color>
7 ■ <color name="navy">#000080</color>
8 ■ <color name="teal">#008080</color>
9 ■ <color name="olive">#808000</color>
0 ■ <color name="brown">#A52A2A</color>
1 ■ <color name="gold">#FFD700</color>
2 ■ <color name="beige">#F5F5DC</color>
3 ■ ! <color name="silver">#C0C0C0</color>
4
5
6 </resources>
```

Các giá trị và tên màu này đến từ bảng màu được khuyến nghị cho các ứng dụng Android được định nghĩa tại **Material Design - Style - Color**. Các mã này biểu thị giá trị RGB của màu theo hệ thập lục phân.

## 2.2 Thêm hành vi vào MainActivity

Trong nhiệm vụ này, bạn sẽ hoàn thành việc thiết lập dự án bằng cách thêm các biến **private** và triển khai các phương thức **onCreate()** và **onSaveInstanceState()**.

3. Mở **MainActivity**.

4. Thêm một biến **private** ở đầu lớp để giữ đối tượng **TextView**:

```
private TextView mHelloTextView;
```

4. Thêm mảng màu sau đây ngay sau biến **private**:

```
private String[] mColorArray = {
 "red",
 "pink",
 "deep_purple",
 "indigo",
 "blue",
 "light_blue",
 "cyan",
 "teal",
 "green",
 "light_green",
 "lime",
 "yellow",
 "amber",
 "orange",
 "deep_orange",
 "brown",
 "grey",
 "blue_grey",
 "black"
};|
```

Mỗi tên màu tương ứng với tên của một tài nguyên màu trong **color.xml**.

5. Trong phương thức **onCreate()**, sử dụng **findViewById()** để lấy tham chiếu đến đối tượng **TextView** và gán nó cho biến **private**:

```
mHelloTextView = findViewById(R.id.hello_textview);
```

3. Trong phương thức **onCreate()**, khôi phục trạng thái phiên lưu trữ, nếu có:

```
if (savedInstanceState != null) {
 mHelloTextView.setTextColor(savedInstanceState.getInt("color"));
}
```

4. Thêm phương thức **onSaveInstanceState()** vào lớp **MainActivity** để lưu màu của văn bản:

```
@Override
public void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 // lưu màu hiện tại của văn bản
 outState.putInt("color", mHelloTextView.getCurrentTextColor());
}
```

## Mã giải pháp cho Nhiệm vụ 2

Dưới đây là mã giải pháp cho bộ cục XML và một đoạn mã trong lớp **MainActivity** của ứng dụng **HelloCompat** cho đến thời điểm này.

Tệp bộ cục **activity\_main.xml** như sau. Trình xử lý **changeColor** cho thuộc tính **android:onClick** của nút **Button** đang được gạch đỏ vì nó chưa được định nghĩa. Bạn sẽ định nghĩa nó trong nhiệm vụ tiếp theo.

```
<?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/hello_textview"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/hello_world"
 android:textAlignment="center"
 android:textSize="100sp"
 android:textStyle="bold"
 android:layout_marginTop="8dp"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintLeft_toLeftOf="parent"
 app:layout_constraintRight_toRightOf="parent"/>

<Button
 android:id="@+id/color_button"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Change Color"
 android:layout_marginTop="8dp"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintLeft_toLeftOf="parent"
 app:layout_constraintRight_toRightOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 android:onClick="changeColor"/>

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

## Lớp MainActivity

Lớp **MainActivity** bao gồm các biến private sau ở đầu lớp:

```
private TextView mHelloTextView;
no usages
private String[] mColorArray = {
 "red",
 "pink",
 "deep_purple",
 "indigo",
 "blue",
 "light_blue",
 "cyan",
 "teal",
 "green",
 "light_green",
 "lime",
 "yellow",
 "amber",
 "orange",
 "deep_orange",
 "brown",
 "grey",
 "blue_grey",
 "black"
};
```

**Phương thức onCreate() và onSaveInstanceState() trong MainActivity**

```

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 EdgeToEdge.enable($this$enableEdgeToEdge: this);
 setContentView(R.layout.activity_main);
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
 return insets;
 });
 mHelloTextView = findViewById(R.id.hello_textview);
 if (savedInstanceState != null) {
 mHelloTextView.setTextColor(savedInstanceState.getInt(key: "color"));
 }
}

@Override
public void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 // lưu màu hiện tại của văn bản
 outState.putInt("color", mHelloTextView.getCurrentTextColor());
}

```

### Nhiệm vụ 3: Thực hiện hành vi cho Button

Nút **Change Color** trong ứng dụng **HelloCompat** chọn ngẫu nhiên một trong 20 màu từ tệp tài nguyên color.xml và đặt màu của văn bản thành màu đó. Trong nhiệm vụ này, bạn sẽ triển khai hành vi cho trình xử lý sự kiện khi nhấn Button.

#### 2.1 Thêm trình xử lý sự kiện changeButton()

5. Mở tệp activity\_main.xml nếu chưa mở. Chuyển sang tab **Text** để hiển thị mã XML.
6. Nhập vào "changeColor" trong thuộc tính android:onClick của phần tử Button.
7. Nhấn **Alt+Enter** (hoặc **Option+Enter** trên Mac) và chọn **Create onClick event handler**.
8. Chọn **MainActivity** và nhập **OK**.

Điều này sẽ tạo một phương thức mẫu cho changeColor() trong MainActivity:

1 usage

```
public void changeColor(View view) {
}
```

## 2.2 Triển khai hành động của Button

3. Chuyển sang tệp MainActivity.
4. Trong phương thức changeColor(), tạo một đối tượng số ngẫu nhiên bằng cách sử dụng lớp Random (một lớp trong Java):

```
Random random = new Random();
```

7. Sử dụng đối tượng random để chọn một màu ngẫu nhiên từ mảng mColorArray:

```
String colorName = mColorArray[random.nextInt(bound: 20)];
```

Phương thức nextInt() với tham số 20 sẽ trả về một số nguyên ngẫu nhiên trong khoảng từ 0 đến 19. Số nguyên này sẽ được dùng làm chỉ số để truy cập vào mảng và lấy tên màu.

8. Lấy mã định danh tài nguyên (một số nguyên) cho tên màu từ tài nguyên:

```
int colorResourceName = getResources().getIdentifier(colorName, "color",
getApplicationContext().getPackageName());
```

Khi ứng dụng của bạn được biên dịch, hệ thống Android chuyển đổi các định nghĩa trong các tệp XML của bạn thành các tài nguyên với các mã định danh nội bộ dạng số nguyên (ID). Có các ID riêng biệt cho cả tên và giá trị. Dòng mã này ánh xạ các chuỗi màu từ mảng colorName với các ID tên màu tương ứng trong tệp tài nguyên XML. Phương thức getResources() lấy tất cả các tài nguyên của ứng dụng, trong khi phương thức getIdentifier() tìm kiếm tên màu (chuỗi) trong tài nguyên màu ("color") của tên gói hiện tại.

9. **Lấy mã số nguyên cho màu thực tế từ các tài nguyên và gán nó cho biến colorRes, sau đó sử dụng phương thức getTheme() để lấy giao diện chủ đề cho ngữ cảnh ứng dụng hiện tại.**

```
int colorRes = getResources().getColor(colorResourceName, this.getTheme());
```

Phương thức getResources() lấy tập hợp các tài nguyên cho ứng dụng của bạn, và phương thức getColor() truy xuất một màu cụ thể từ các tài nguyên đó thông qua ID của tên màu. Tuy nhiên, getColor() bị gạch chân màu đỏ.

Nếu bạn trỏ chuột vào getColor(), Android Studio sẽ báo lỗi: "Call requires API 23 (current min is 15)". Vì minSdkVersion của bạn là 15, bạn sẽ nhận được thông báo này khi cố gắng sử dụng bất kỳ API nào được giới thiệu sau API 15. Bạn vẫn có thể biên dịch ứng dụng của mình, nhưng vì phiên bản getColor() này không khả dụng trên các thiết bị chạy API trước 23, ứng dụng của bạn sẽ bị lỗi khi người dùng nhấn nút **Change Color**.

Ở giai đoạn này, bạn có thể kiểm tra phiên bản nền tảng và sử dụng phiên bản phù hợp của getColor() tùy thuộc vào nơi ứng dụng đang chạy. Một cách tốt hơn để hỗ trợ cả các API Android cũ hơn và mới hơn mà không gặp cảnh báo là sử dụng một trong các lớp tương thích trong thư viện hỗ trợ.

## 10. Thay đổi dòng gán colorRes để sử dụng lớp ContextCompat:

```
int colorRes = ContextCompat.getColor(context: this, colorResourceName);
```

Lớp ContextCompat cung cấp nhiều phương thức hỗ trợ tương thích để xử lý sự khác biệt giữa các phiên bản API trong ngữ cảnh ứng dụng và tài nguyên của ứng dụng. Phương thức getColor() trong ContextCompat nhận hai đối số: ngữ cảnh hiện tại (trong trường hợp này là instance của Activity, tức this) và tên của màu sắc.

Việc triển khai phương thức này trong thư viện hỗ trợ ẩn đi sự khác biệt về cách thực hiện giữa các phiên bản API khác nhau. Bạn có thể gọi phương thức này mà không gặp bất kỳ cảnh báo, lỗi, hoặc sự cố nào, bất kể phiên bản SDK biên dịch hoặc phiên bản SDK tối thiểu của bạn là gì.

### 1. Đặt màu cho TextView bằng ID tài nguyên màu:

```
mHelloTextView.setTextColor(colorRes);
```

### 2. Chạy ứng dụng trên thiết bị hoặc trình giả lập, sau đó nhấn nút Change Color.

Nút **Change Color** bây giờ sẽ thay đổi màu văn bản trong ứng dụng, như minh họa bên dưới.



cessfully finished in 719 ms

Mã giải pháp

### Giải pháp cho MainActivity

Dưới đây là trình xử lý sự kiện **changeColor()** trong **MainActivity**:

```
public void changeColor(View view) {
 Random random = new Random();
 String colorName = mColorArray[random.nextInt(bound: 20)];
 int colorResourceName = getResources().getIdentifier(colorName, defType: "color", getApplicationContext().getPackageName());
 int colorRes = ContextCompat.getColor(context: this,colorResourceName);
 mHelloTextView.setTextColor(colorRes);
}
```

## Dự án Android Studio

### Dự án Android Studio: HelloCompat

#### Thử thách lập trình

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

**Thử thách:** Thay vì sử dụng **ContextCompat** để lấy tài nguyên màu, hãy sử dụng kiểm tra giá trị trong lớp **Build** để thực hiện một thao tác khác nếu ứng dụng đang chạy trên thiết bị hỗ trợ phiên bản Android cũ hơn API 23.

#### Tóm tắt

#### Cài đặt Android Support Library:

- Sử dụng **SDK Manager** để cài đặt **Android Support Repository**. Chọn **Tools > Android > SDK Manager**, nhấp vào tab **SDK Tools** và mở rộng **Support Repository**.
- Nếu cột **Status** hiển thị **Installed** cho **Android Support Repository**, nhấp vào **Cancel**; nếu hiển thị **Not installed** hoặc **Update Available**, đánh dấu vào ô kiểm. Một biểu tượng tải xuống sẽ xuất hiện bên cạnh ô kiểm. Nhấp vào **OK**.

**Android sử dụng ba chỉ thị để chỉ định cách ứng dụng của bạn hoạt động 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 được biên dịch.
- targetSdkVersion:** phiên bản API mà ứng dụng của bạn được thiết kế để chạy.

#### Quản lý dependencies trong dự án:

- Mở rộng **Gradle Scripts** trong mục **Project > Android**, và mở tệp **build.gradle (Module: app)**.

- Bạn có thể thêm dependencies trong phần **dependencies**.

**Lớp ContextCompat** cung cấp các phương thức hỗ trợ tương thích với ngữ cảnh và các phương thức liên quan đến tài nguyên cho cả API cũ và mới.

### **Khái niệm liên quan**

Tài liệu về khái niệm liên quan nằm trong **3.3: The Android Support Library**.

### **Tìm hiểu thêm**

Tài liệu Android Studio:

- **Android Studio User Guide**  
Tài liệu nhà phát triển Android:
- **Android Support Library (introduction)**
- **Support Library Setup**
- **Support Library Features**
- **Supporting Different Platform Versions**
- **Package Index** (tất cả các gói API bắt đầu bằng android.support).

**Khác:**

- **Picking your compileSdkVersion, minSdkVersion, and targetSdkVersion**
- **Understanding the Android Support Library**
- **All the Things Compat**

### **Bài tập về nhà**

#### **Chạy ứng dụng**

Mở ứng dụng **HelloCompat** mà bạn đã tạo trong bài thực hành về cách sử dụng các thư viện hỗ trợ.

4. Đặt một điểm dừng gỡ lỗi (debugger breakpoint) trên dòng mã trong phương thức changeColor() nơi thực sự thay đổi màu sắc:

java

CopyEdit

```
int colorRes = ContextCompat.getColor(this, colorResourceName);
```

5. Chạy ứng dụng ở chế độ debug trên một thiết bị hoặc trình giả lập (emulator) đang chạy phiên bản API 23 hoặc mới hơn. Nhấp vào **Step Into** để bước vào phương thức getColor() và theo dõi các lời gọi phương thức sâu hơn trong ngăn xếp.
  - Kiểm tra cách lớp ContextCompat xác định phương pháp lấy màu từ tài nguyên và các lớp framework khác mà nó sử dụng.
  - Một số lớp có thể hiển thị cảnh báo rằng "mã nguồn không khớp với bytecode." Nhấp vào **Step Out** để quay lại một tệp mã nguồn đã biết hoặc tiếp tục nhấp vào **Step Into** cho đến khi trình gõ lỗi tự quay trở lại.
6. Lặp lại bước trước đó trên một thiết bị hoặc trình giả lập chạy phiên bản API cũ hơn 23.
  - Ghi nhận các đường dẫn khác nhau mà framework thực hiện để lấy màu.

## Câu hỏi 1

Khi bạn bước vào lần đầu tiên bằng cách nhấp **Step Into** phương thức ContextCompat.getColor(), lớp nào sẽ xuất hiện? Chọn một:

- **MainActivity**
- **ContextCompat**
- **AppCompatActivity**
- **Context**

## Câu hỏi 2

Trong lớp xuất hiện, câu lệnh nào được thực thi nếu phiên bản API là 23 hoặc mới hơn? Chọn một:

- **return context.getColor(id);**
- **return context.getResources().getColor(id);**
- **throw new IllegalArgumentException("permission is null");**
- **return mResources == null ? super.getResources() : mResources;**

### Câu hỏi 3

Nếu bạn thay đổi phương thức ContextCompat.getColor() thành phương thức getColor(), điều gì sẽ xảy ra khi bạn chạy ứng dụng? Chọn một:

- Nếu minSdkVersion của bạn là 15, từ getColor sẽ bị gạch chân màu đỏ trong trình chỉnh sửa mã. Khi trỏ chuột vào, Android Studio sẽ báo: "Call requires API 23 (current min is 15)".
- Ứng dụng sẽ chạy mà không gặp lỗi trên các trình giả lập và thiết bị sử dụng API 23 hoặc mới hơn.
- Ứng dụng sẽ bị sập khi người dùng nhấn Change Color nếu trình giả lập hoặc thiết bị đang sử dụng API 17.
- Tất cả các điều trên.

Nộp ứng dụng của bạn để chấm điểm

**Hướng dẫn cho người chấm điểm**

Không có ứng dụng nào cần nộp cho bài tập về nhà này.

## Unit 2: Trải nghiệm người dùng

PDF này chứa bản chụp nhanh một lần về các bài học trong **Unit 2: Trải nghiệm người dùng**.

Các bài học trong đơn vị này:

### Bài học 4: Tương tác người dùng

- **4.1:** Hình ảnh có thể nhấp (Clickable images)
- **4.2:** Các điều khiển đầu vào (Input controls)
- **4.3:** Menu và bộ chọn (Menus and pickers)
- **4.4:** Điều hướng người dùng (User navigation)
- **4.5:** RecyclerView

### Bài học 5: Trải nghiệm người dùng thú vị

- **5.1:** Các đối tượng đồ họa (Drawables), phong cách (Styles) và chủ đề (Themes)
- **5.2:** Thẻ (Cards) và màu sắc (Colors)
- **5.3:** Bố cục thích ứng (Adaptive layouts)

## Bài học 6: Kiểm thử giao diện người dùng

- **6.1:** Espresso cho kiểm thử giao diện người dùng (Espresso for UI testing)

### Bài học 4.1: Hình ảnh có thể nhấp (Clickable images)

#### Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị Android được tạo thành từ một hệ thống phân cấp các đối tượng được gọi là **views**. Mỗi phần tử trên màn hình là một **View**.

Lớp **View** đại diện cho khái niệm cơ bản cho tất cả các thành phần UI. **View** là lớp cơ sở cho các lớp cung cấp các thành phần UI tương tác, chẳng hạn như các phần tử **Button**. Một **Button** là một thành phần UI mà người dùng có thể nhấn hoặc nhấp để thực hiện một hành động.

Bạn có thể biến bất kỳ **View** nào, chẳng hạn như **ImageView**, thành một phần tử UI có thể nhấn hoặc nhấp. 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 làm phần tử mà người dùng có thể nhấn hoặc nhấp.

#### Những gì bạn cần biết trước

Bạn nên có khả năng:

- Tạo một dự án Android Studio từ một mẫu và tạo bố cục chính.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị được kết nối.
- Tạo và chỉnh sửa các phần tử UI bằng trình chỉnh sửa bố cục và mã XML.
- Truy cập các phần tử UI từ mã của bạn bằng **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 làm phần tử tương tác để thực hiện một hành động.

- Cách thiết lập các thuộc tính cho các phần tử **ImageView** trong trình chỉnh sửa bố cục.
- Cách thêm phương thức **onClick()** để hiển thị thông báo **Toast**.

## Những gì bạn sẽ làm

- Tạo một dự án Android Studio mới cho một ứng dụng giả lập đặt món tráng miệng, sử dụng hình ảnh làm các phần tử tương tác.
- Thiết lập các bộ xử lý **onClick()** cho các hình ảnh để hiển thị các thông báo **Toast** khác nhau.
- Thay đổi nút hành động nổi (floating action button) được cung cấp bởi mẫu, để nó hiển thị một biểu tượng khác và khởi chạy một **Activity** khác.

## Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ tạo và xây dựng một ứng dụng mới bắt đầu với mẫu **Basic Activity**, mô phỏng một ứng dụng đặt món tráng miệng. Người dùng có thể nhấn vào một hình ảnh để thực hiện một hành động—trong trường hợp này là hiển thị thông báo **Toast**—như được minh họa trong hình dưới đây. Người dùng cũng có thể nhấn vào nút giỏ hàng để chuyển đến **Activity** tiếp theo.



### Nhiệm vụ 1: Thêm hình ảnh vào bộ cục

Bạn có thể làm cho một view có thể nhấp, giống như một nút, bằng cách thêm thuộc tính **android:onClick** trong tệp bộ cục XML. Ví dụ, bạn có thể làm cho một hình ảnh hoạt động như một nút bằng cách thêm **android:onClick** vào **ImageView**.

Trong nhiệm vụ này, bạn sẽ tạo một nguyên mẫu ứng dụng để đặt món tráng miệng từ một quán cà phê. Sau khi bắt đầu một dự án mới dựa trên mẫu **Basic Activity**, bạn sẽ chỉnh sửa **TextView** "Hello World" với văn bản phù hợp và thêm các hình ảnh mà người dùng có thể nhấn.

#### 1.1 Bắt đầu dự án mới

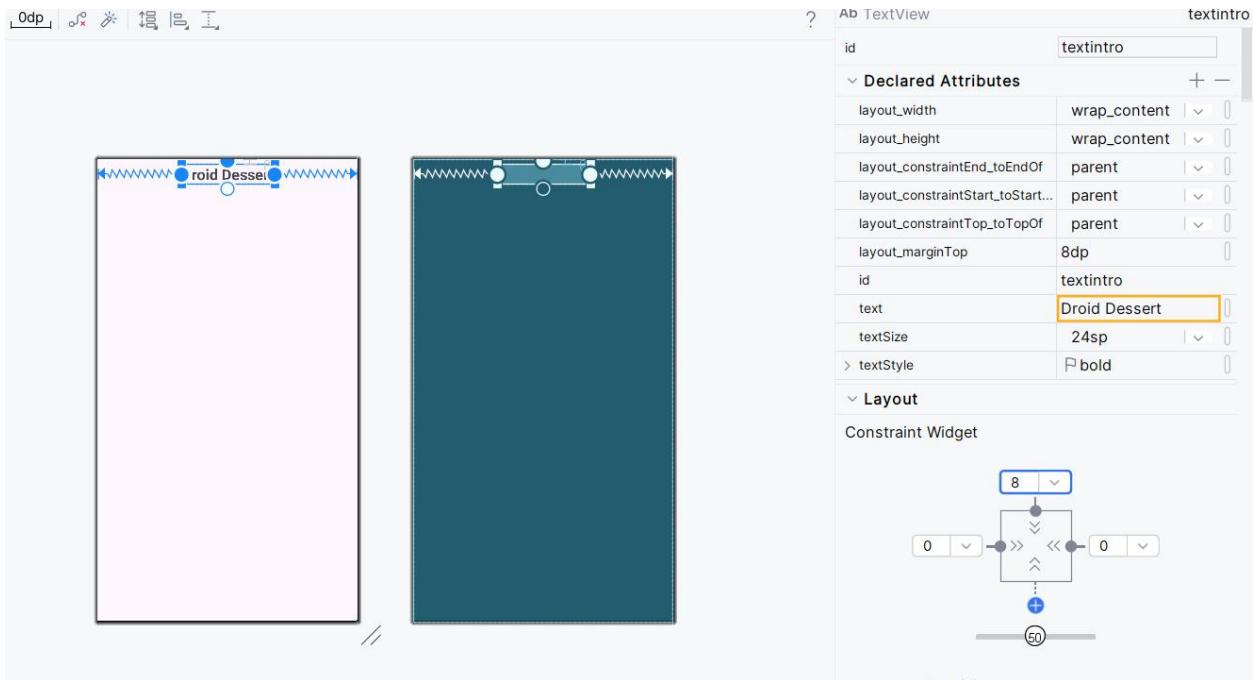
1. Bắt đầu một dự án mới trên Android Studio với tên ứng dụng **Droid Cafe**.

2. Chọn mẫu **Basic Activity** và chấp nhận tên **Activity** mặc định (**MainActivity**). Đảm bảo các tùy chọn **Generate Layout file** và **Backwards Compatibility (AppCompat)** được chọn.
3. Nhấp vào **Finish**.
  - o Dự án sẽ mở với hai tệp bố cục trong thư mục **res > layout**:
    - **activity\_main.xml** cho thanh ứng dụng (app bar) và nút hành động nổi (floating action button), mà bạn không thay đổi trong nhiệm vụ này.
    - **content\_main.xml** cho các thành phần còn lại trong bố cục.
4. Mở tệp **content\_main.xml** và nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị trình chỉnh sửa bố cục.
5. Chọn **TextView "Hello World"** trong bố cục và mở bảng **Attributes**.
6. Thay đổi các thuộc tính **textintro** như sau:

<b>Attribute field</b>	<b>Enter the following:</b>
ID	textintro
text	Change Hello World to Droid Dessert
textStyle	B (bold)
textSize	24sp

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

7. Xóa ràng buộc kéo dài từ đáy của TextView textintro đến đáy của bố cục, để TextView gắn vào đỉnh của bố cục, và chọn **8 (8dp)** cho khoảng cách ở trên như hình dưới



8. Trong bài học trước, bạn đã học cách trích xuất tài nguyên chuỗi từ một chuỗi văn bản tĩnh. Nhấp vào tab **Text** bản để chuyển sang mã XML và trích xuất chuỗi "Droid Desserts" trong TextView và nhập **intro\_text** làm tên tài nguyên chuỗi (string resource name).

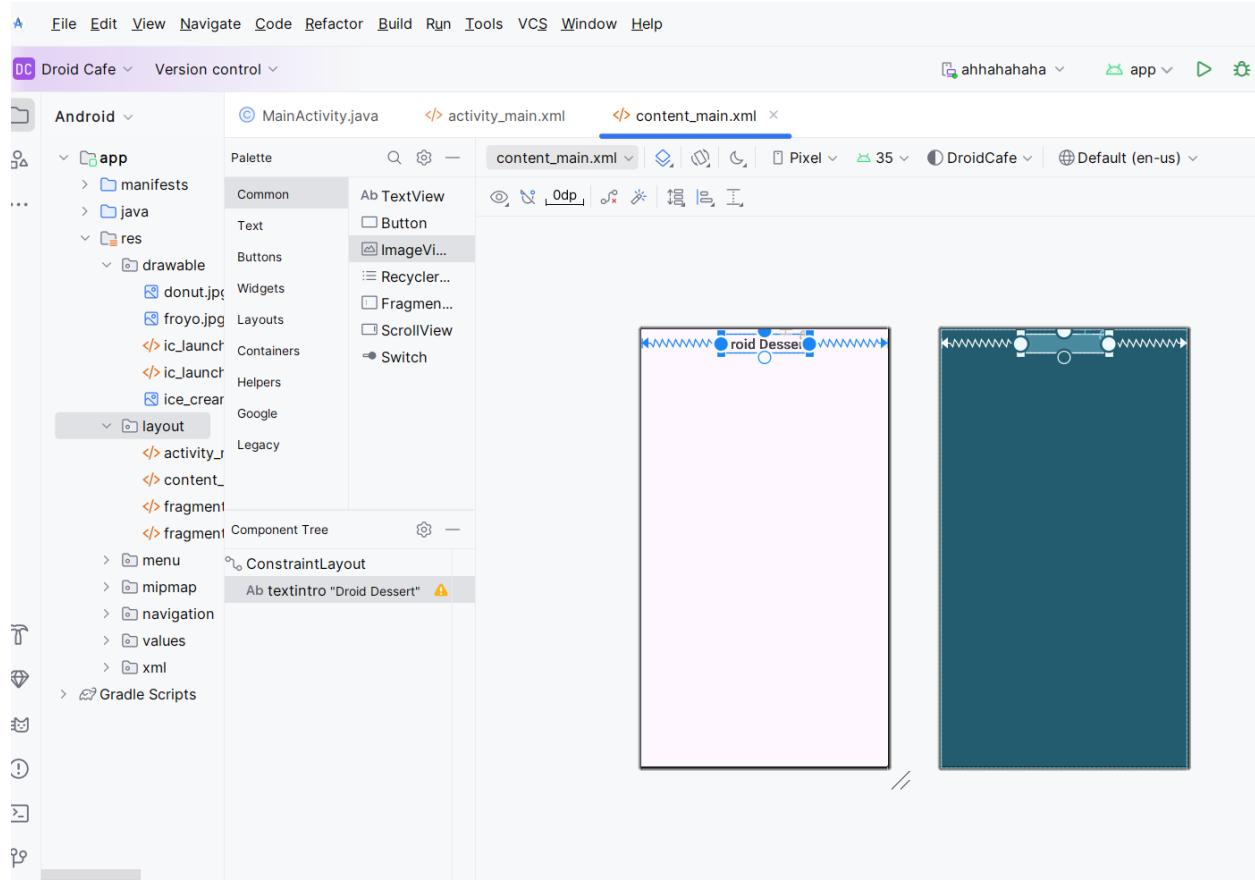
## 1.2 Thêm hình ảnh

Ba hình ảnh (donut\_circle.png, froyo\_circle.png và icecream\_circle.png) được cung cấp cho ví dụ này, bạn có thể [download](#). Thay vào đó, bạn có thể thay thế bằng các hình ảnh của riêng bạn dưới dạng tệp PNG, nhưng chúng phải có kích thước khoảng 113 x 113 pixel để sử dụng trong ví dụ này.

Bước này cũng giới thiệu một kỹ thuật mới trong trình chỉnh sửa bộ cục: sử dụng nút **Fix** trong các tin nhắn cảnh báo để trích xuất tài nguyên chuỗi.

1. Để sao chép hình ảnh vào dự án của bạn, trước tiên hãy đóng dự án.
2. Sao chép các tệp hình ảnh vào thư mục **drawable** của dự án của bạn. Tìm thư mục **drawable** trong một dự án bằng cách sử dụng đường dẫn này: **project\_name > app > src > main > res > drawable**.
3. Mở lại dự án của bạn.
4. Mở tệp **content\_main.xml** và nhấp vào tab **Design** (nếu nó chưa được chọn).

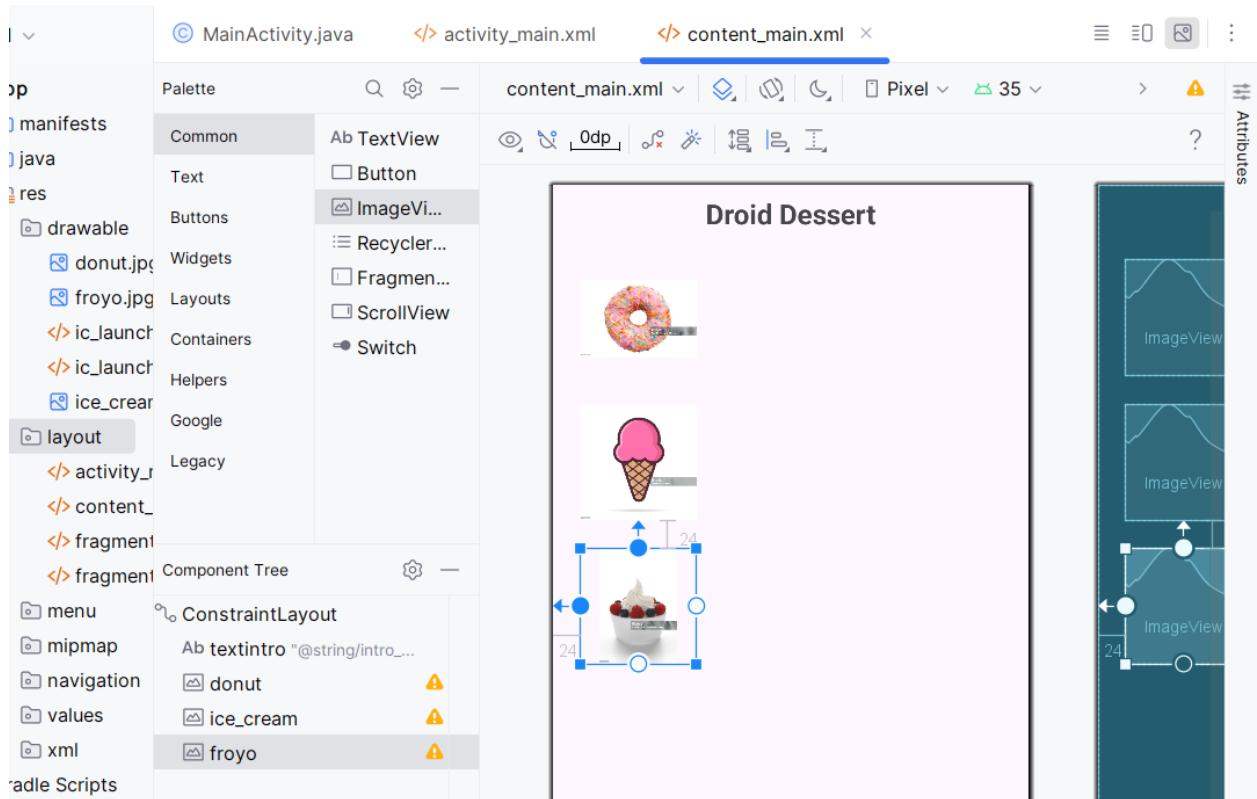
5. Kéo một ImageView vào bố cục, chọn hình ảnh **donut\_circle** cho nó và ràng buộc nó với TextView ở trên cùng và phía bên trái của bố cục với khoảng cách **24** (24dp) cho cả hai ràng buộc, như được thể hiện trong hình chuyển động bên dưới.



6. Trong bảng thuộc tính, nhập các giá trị sau cho các thuộc tính:

Attribute field	Enter the following:
ID	donut
contentDescription	Donuts are glazed and sprinkled with candy. (You can copy/paste the text into the field.)

7. Kéo một ImageView thứ hai vào bố cục, chọn hình ảnh **icecream\_circle** cho nó, và ràng buộc nó vào phía dưới của ImageView đầu tiên cũng như phía bên trái của bố cục với khoảng cách **24dp** cho cả hai ràng buộc.



8. Trong bảng **Attributes**, nhập các giá trị sau cho các thuộc tính:

<b>Attribute field</b>	<b>Enter the following:</b>
ID	ice_cream
contentDescription	Ice cream sandwiches have chocolate wafers and vanilla filling. (You can copy/paste the text into the field.)

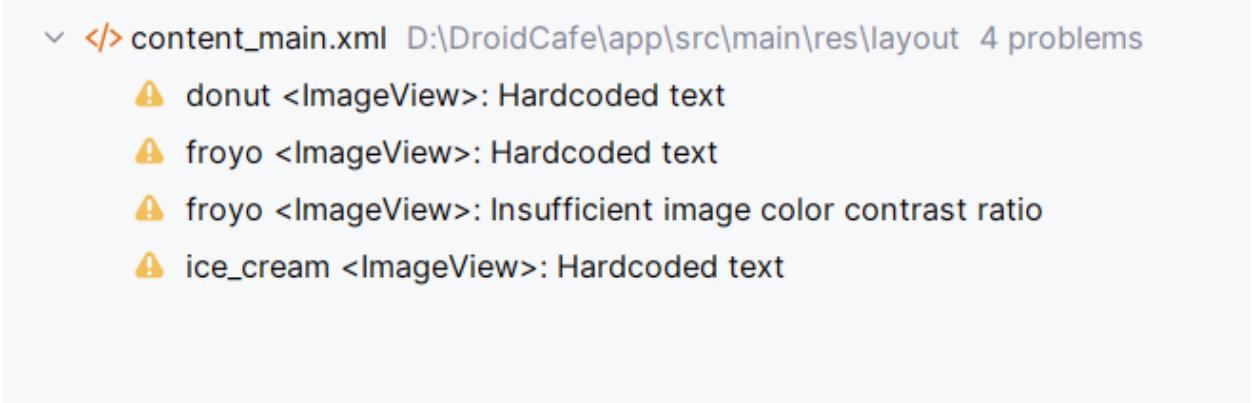
9. Kéo một **ImageView** thứ ba vào bố cục, chọn hình ảnh **froyo\_circle** cho nó, và ràng buộc nó vào phía dưới của **ImageView** thứ hai cũng như phía bên trái của bố cục với khoảng cách **24dp** cho cả hai ràng buộc.

10. Trong bảng **Attributes**, nhập các giá trị sau cho các thuộc tính:

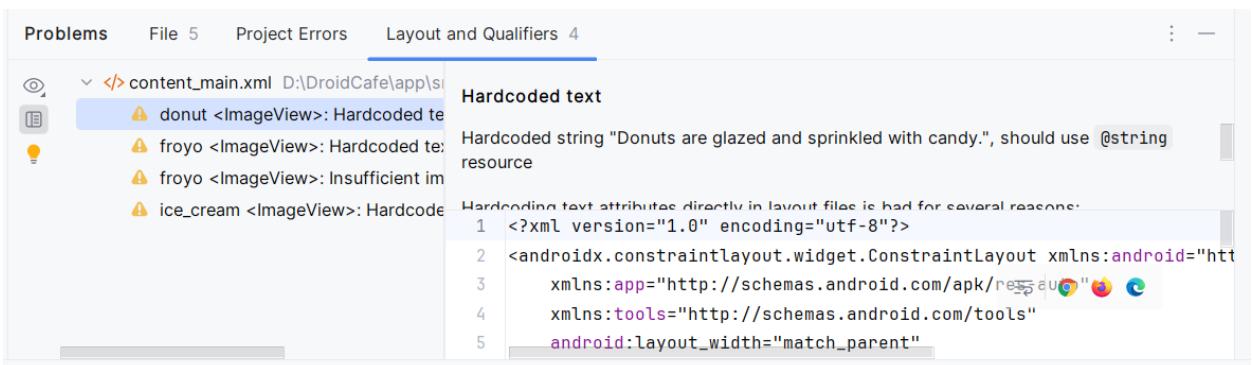
Trường thuộc tính	Nhập các thông tin sau
ID	froyo

contentDescription	FroYo is premium self-serve frozen yogurt. (You can copy/paste the text into the field.)
--------------------	------------------------------------------------------------------------------------------

11.Nhấp vào biểu tượng  ở góc trên bên trái của trình chỉnh sửa bố cục để mở bảng cảnh báo, bảng này sẽ hiển thị các cảnh báo về văn bản được mã hóa cứng:



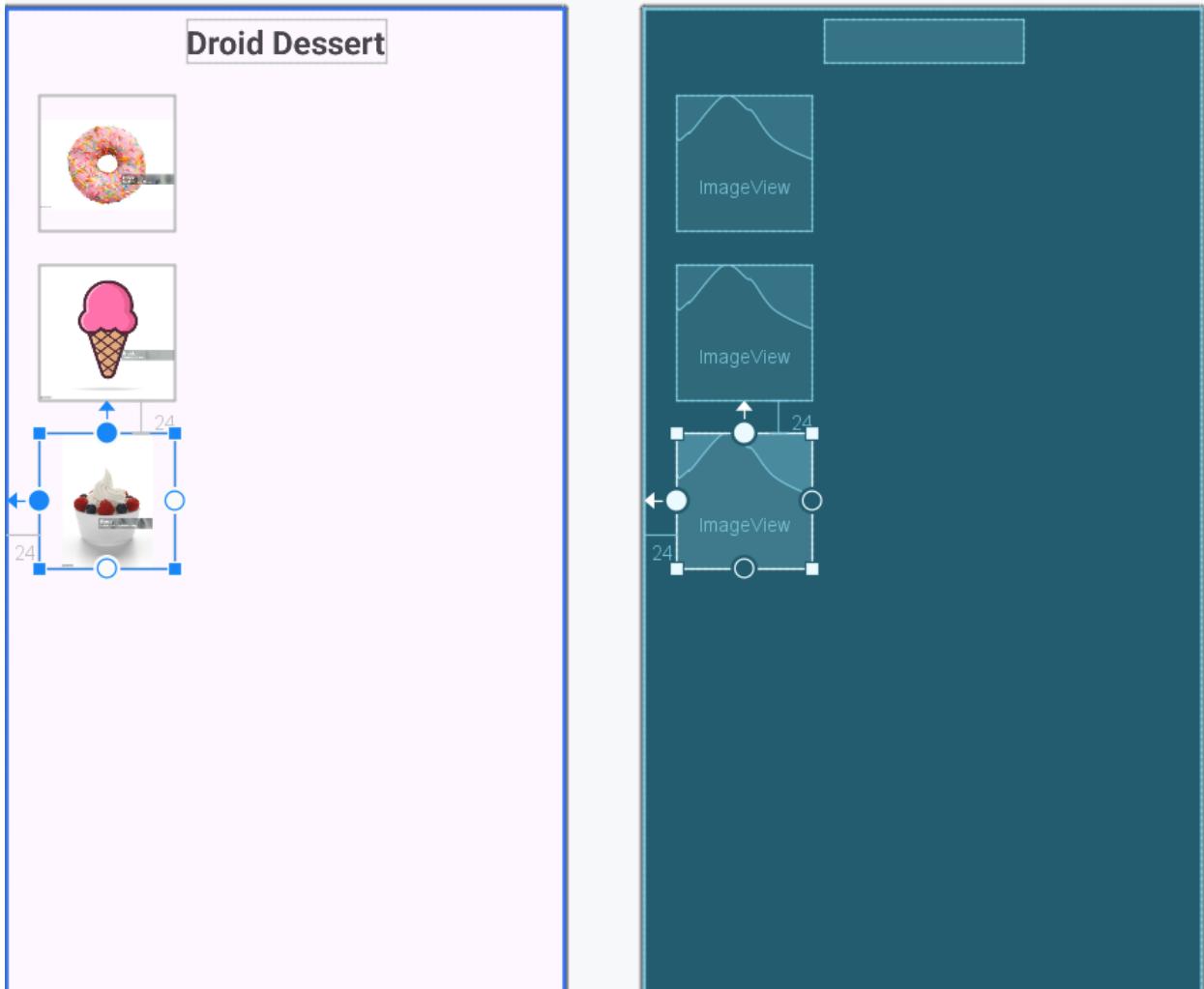
12.Mở rộng từng cảnh báo **Hardcoded text**, cuộn xuống cuối thông báo cảnh báo, và nhấp vào nút **Fix** như hình dưới đây:



Việc sửa từng cảnh báo văn bản được mã hóa cứng sẽ trích xuất tài nguyên chuỗi (string resource) cho chuỗi văn bản đó. Hộp thoại **Extract Resource** sẽ xuất hiện, và bạn có thể nhập tên cho tài nguyên chuỗi. Nhập các tên sau cho các tài nguyên chuỗi

String	Enter the following name:
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

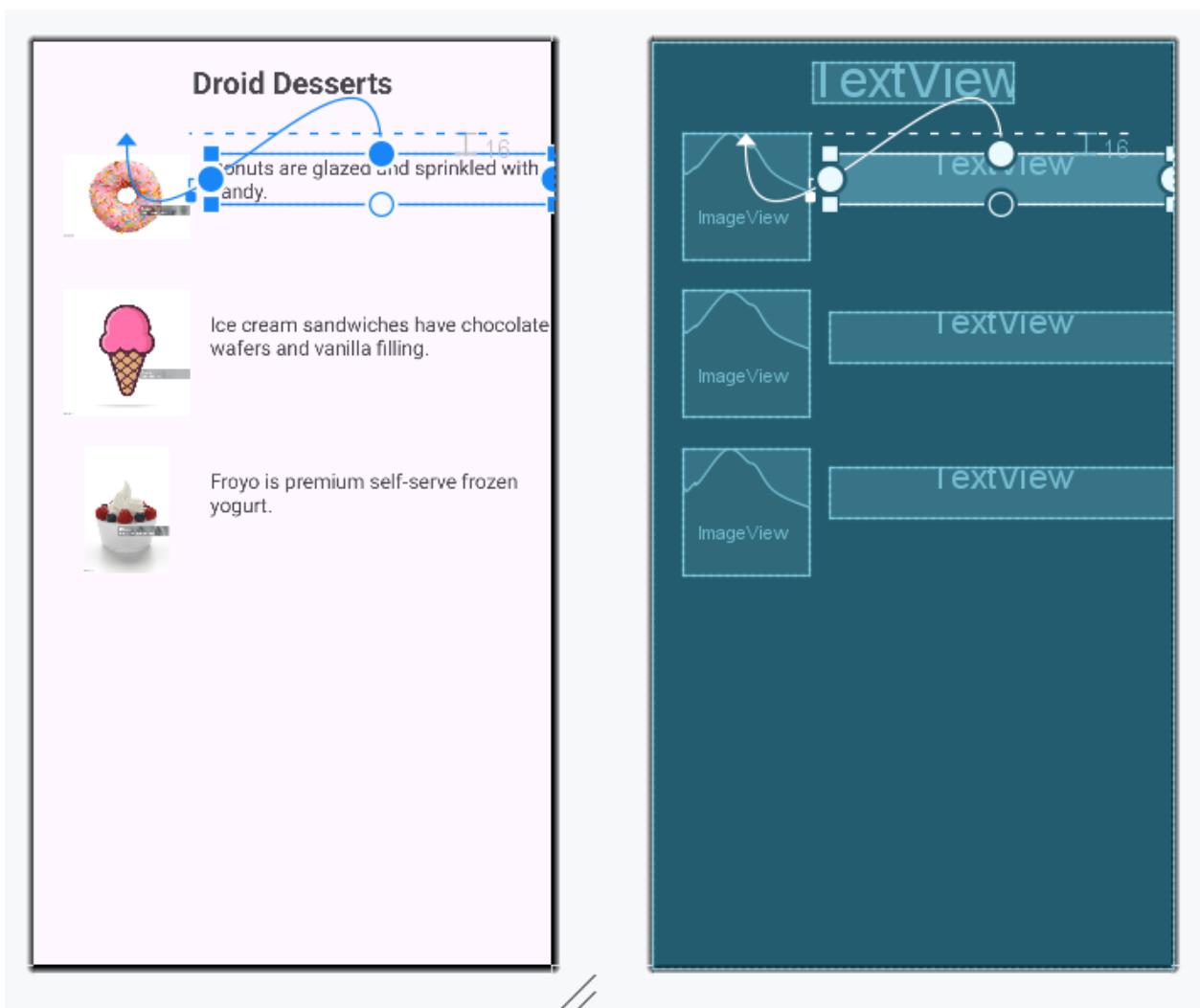
Bố cục sẽ giống hình dưới đây



### 1.3 Thêm miêu tả văn bản

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

1. Kéo một phần tử TextView vào bố cục.
2. Ràng buộc cạnh trái của phần tử với cạnh phải của ImageView donut và cạnh trên với cạnh trên của ImageView donut, cả hai đều có khoảng cách **24** (24dp).
3. Ràng buộc cạnh phải của phần tử với cạnh phải của bố cục và sử dụng khoảng cách giống như 24 (24dp). Nhập **donut\_description** vào trường ID trong bảng thuộc tính. TextView mới sẽ xuất hiện bên cạnh hình ảnh donut như hình dưới đây.



7. Trong thanh Thuộc tính, thay đổi độ rộng trong bảng kiểm tra thành **Match Constraints**:

Attributes

Ab TextView donut\_description

id donut\_description

Declared Attributes

layout_width	0dp
layout_height	wrap_c
layout_constraintStart...	@id/donu
layout_constraintEnd_t...	parent
layout_constraintTop_t...	@id/donu
layout_marginStart	@dimen/margi...
layout_marginTop	@dimen/margin_wi
layout_marginEnd	24dp
id	donut_descrip...
text	Donuts are gl...
textSize	16sp

Layout

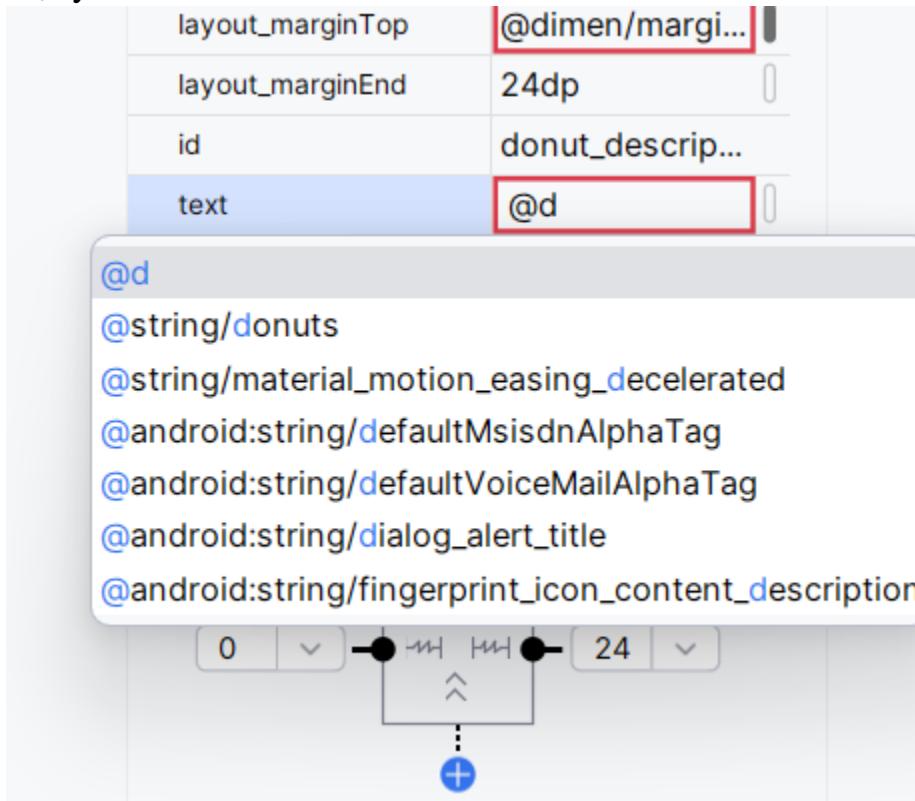
Constraint Widget

The diagram shows a central rectangular box with four black circular handles at its corners. Each corner has a horizontal or vertical dashed line extending towards the center of the box. The top-left handle is connected to a blue-bordered input field containing '0'. The top-right handle is connected to a white input field containing '24'. The bottom-left handle is connected to a white input field containing '24'. The bottom-right handle is connected to another white input field containing '24'. A blue '+' button is located below the bottom-left handle, and a grey slider with the value '50' is at the bottom.

> Constraints (3)

8. Trong bảng Thuộc tính, bắt đầu nhập tài nguyên chuỗi cho trường văn bản bằng cách đặt trước nó với ký hiệu @: @d. Nhập vào tên tài nguyên chuỗi (@string/donuts) mà xuất hiện như một gợi ý

Gợi ý



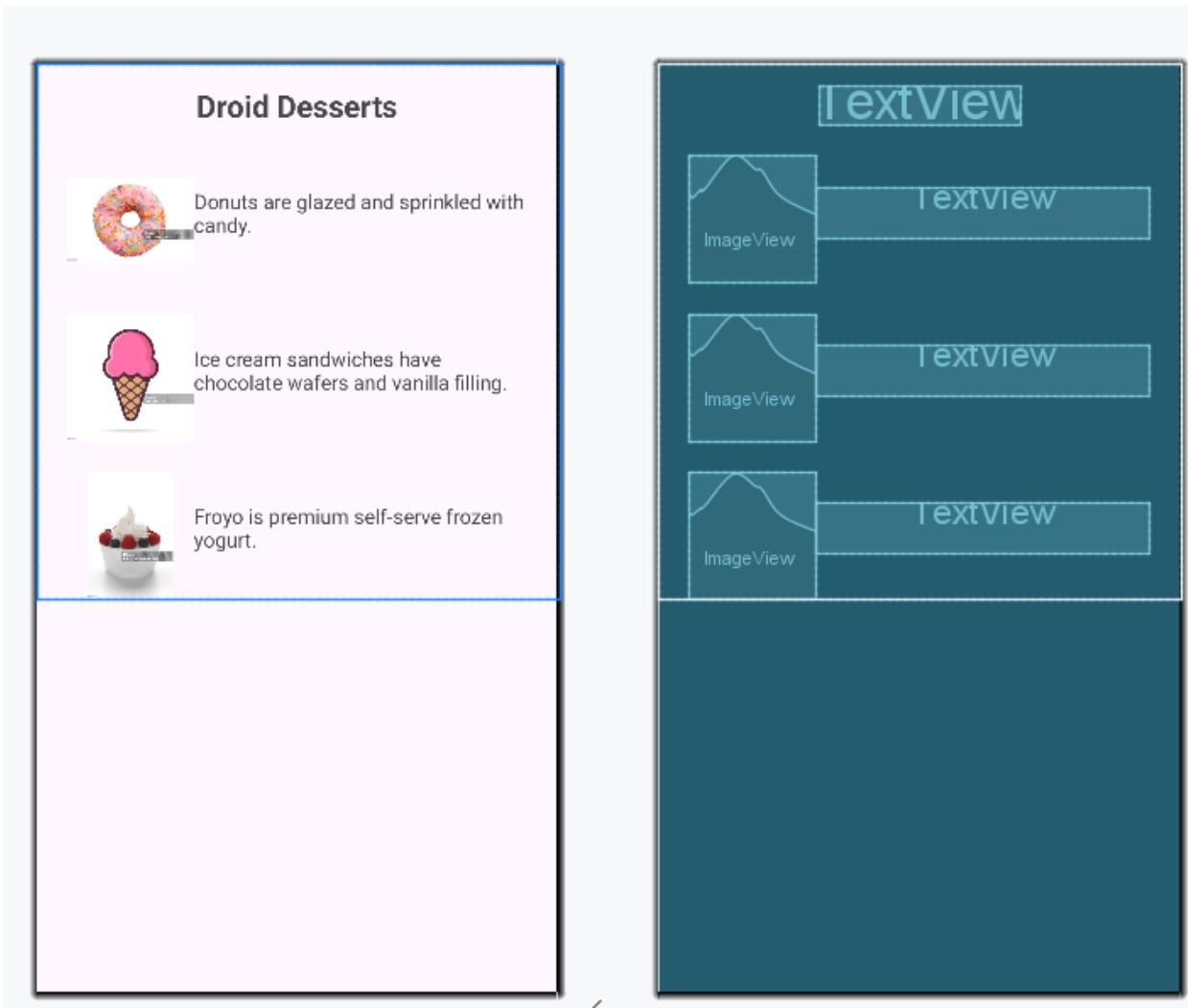
9. Lặp lại các bước trên để thêm một TextView thứ hai được ràng buộc ở bên phải và trên của ImageView ice\_cream, và cạnh phải của nó ràng buộc với cạnh phải của bố cục. Nhập thông tin sau vào bảng Thuộc tính:

Attribute field	Enter the following:
ID	ice_cream_description
Left, right, and top margins	24
layout_width	match_constraint
text	@string/ice_cream_sandwiches

10. Lặp lại các bước trên để thêm một TextView thứ ba bị ràng buộc ở bên phải và phía trên của froyo ImageView, và bên phải của nó với bên phải của layout. Nhập những thông tin sau vào bảng Attributes :

Attribute field	Enter the following:
ID	froyo_description
Left, right, and top margins	24
layout_width	match_constraint
text	@string/froyo

Bố cục sẽ như sau:



Task 1: mã giải pháp

Bố cục XML cho tệp content.xml được hiển thị dưới đây.

```
<?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="wrap_content"
 tools:context=".MainActivity">

 <!-- Tiêu đề -->
 <TextView
 android:id="@+id/title"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginTop="16dp"
 android:text="Droid Desserts"
 android:textSize="24sp"
 android:textStyle="bold"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent" />

 <!-- Hàng 1: Donut -->
 <ImageView
 android:id="@+id/donut"
 android:layout_width="100dp"
 android:layout_height="100dp"
 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/title"
 app:srcCompat="@drawable/donut" />

 <TextView
 android:id="@+id/donut_description"
 android:layout_width="0dp"
```

```
 android:layout_height="wrap_content"
 android:layout_marginStart="24dp"
 android:layout_marginTop="@dimen/margin_wide"
 android:layout_marginEnd="24dp"
 android:text="Donuts are glazed and sprinkled with candy."
 android:textSize="16sp"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toEndOf="@+id/donut"
 app:layout_constraintTop_toTopOf="@+id/donut" />

<!-- Hàng 2: Ice Cream -->
<ImageView
 android:id="@+id/ice_cream"
 android:layout_width="100dp"
 android:layout_height="100dp"
 android:layout_marginStart="@dimen/margin_wide"
 android:layout_marginTop="@dimen/margin_wide"
 android:layout_marginEnd="24dp"
 android:contentDescription="@string/ice_cream_sandwiches"
 app:layout_constraintEnd_toStartOf="@+id/ice_cream_description"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/donut"
 app:srcCompat="@drawable/ice_cream" />

<!-- Hàng 3: Froyo -->
<TextView
 android:id="@+id/ice_cream_description"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="24dp"
 android:layout_marginTop="24dp"
 android:layout_marginEnd="24dp"
 android:text="Ice cream sandwiches have chocolate wafers and vanilla
filling."
 android:textSize="16sp"
 app:layout_constraintEnd_toEndOf="parent"
```

```
 app:layout_constraintStart_toEndOf="@+id/ice_cream"
 app:layout_constraintTop_toTopOf="@+id/ice_cream" />

<ImageView
 android:id="@+id/froyo"
 android:layout_width="100dp"
 android:layout_height="100dp"
 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" />

<TextView
 android:id="@+id/froyo_description"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="24dp"
 android:layout_marginTop="@dimen/margin_wide"
 android:layout_marginEnd="24dp"
 android:text="Froyo is premium self-serve frozen yogurt."
 android:textSize="16sp"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toEndOf="@+id/froyo"
 app:layout_constraintTop_toTopOf="@+id/froyo" />

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

## Task 2: Thêm phương thức onClick cho hình ảnh

Để làm cho một View *clickable*, để người dùng có thể chạm (hoặc nhấn) vào nó, hãy thêm thuộc tính android:onClick trong bộ cục XML và chỉ định bộ xử lý nhấp. Ví dụ, bạn có thể làm cho một ImageView hoạt động như một nút đơn giản bằng cách thêm android:onClick vào ImageView. Trong nhiệm vụ này, bạn làm cho các hình ảnh trong bộ cục của bạn có thể nhấn được.

### 2.1 Tạo phương thức Toast

Trong nhiệm vụ này, bạn thêm từng phương thức cho thuộc tính android:onClick để gọi khi mỗi hình ảnh được nhấp. Trong nhiệm vụ này, các phương thức này đơn giản hiển thị một thông điệp Toast cho biết hình ảnh nào đã được chạm. (Trong một chương khác, bạn sửa đổi các phương thức này để khởi động một Activity khác.)

1. Để sử dụng tài nguyên chuỗi trong mã Java, bạn nên thêm chúng vào tệp strings.xml trước. Mở rộng **res > values** trong bảng **Project > Android**, và mở tệp **strings.xml**. Thêm các tài nguyên chuỗi sau cho các chuỗi sẽ được hiển thị trong thông điệp Toast:

```
<string name="donut_order_message">You ordered a donut.</string>
<string name="ice_cream_order_message">You ordered an ice cream sandwich.</string>
<string name="froyo_order_message">You ordered a FroYo.</string>
```

2. Mở **MainActivity** và thêm phương thức `displayToast()` sau cùng vào **MainActivity** (trước dấu ngoặc đóng):

```
public void displayToast(String message) {
 Toast.makeText(getApplicationContext(), message,
 Toast.LENGTH_SHORT).show();
}
```

Mặc dù bạn có thể đã thêm phương thức này ở bất kỳ vị trí nào trong **MainActivity**, nhưng thực tiễn tốt nhất là đặt các phương thức của bạn bên dưới các phương thức đã được cung cấp trong **MainActivity** bởi mẫu.

## 2.2 Tạo trình xử lý sự kiện khi nhấp chuột

Mỗi hình ảnh có thể nhấp cần một trình xử lý sự kiện khi nhấp chuột - một phương thức để thuộc tính android:onClick gọi. Trình xử lý sự kiện khi nhấp chuột, nếu được gọi từ thuộc tính android:onClick, phải là public, trả về void và định nghĩa một View là tham số duy nhất của nó. Làm theo các bước sau để thêm trình xử lý sự kiện khi nhấp chuột:

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

```
public void showDonutOrder(View view) {
 displayToast(getString(R.string.donut_order_message));
}
```

Ba dòng đầu tiên là một chú thích theo định dạng Javadoc, giúp cho mã dễ hiểu hơn và cũng giúp tạo tài liệu cho mã của bạn. Đây là một thực tiễn tốt nhất để thêm chú thích như vậy cho mỗi phương thức mới bạn tạo. Để biết thêm thông tin về cách viết chú thích, hãy xem Cách viết chú thích tài liệu cho công cụ Javadoc.

2. Thêm nhiều phương thức vào cuối **MainActivity** cho mỗi món tráng miệng:

```
1 usage
public void showIceCreamOrder(View view) {
 displayToast(getString(R.string.ice_cream_order_message));
}
```

```
1 usage
public void showFroyoOrder(View view) {
 displayToast(getString(R.string.froyo_order_message));
}
```

3. (Tùy chọn) Chọn **Code > Reformat Code** để định dạng lại mã mà bạn đã thêm vào MainActivity cho phù hợp với tiêu chuẩn và dễ đọc hơn.

### 2.3 Thêm thuộc tính onClick

Trong bước này, bạn thêm android:onClick vào từng phần tử ImageView trong tập tin content\_main.xml. Thuộc tính android:onClick gọi trình xử lý click cho từng phần tử.

1. Mở tập tin **content\_main.xml**, và nhấp vào tab **Text** trong trình chỉnh sửa bộ cục để hiển thị mã XML.

2. Thêm thuộc tính android:onClick vào ImageView donut. Khi bạn nhập, sẽ có các gợi ý hiển thị các trình xử lý click. Chọn trình xử lý click showDonutOrder. Mã hiện tại sẽ trông như sau:

```
<ImageView
 android:id="@+id/donut"
 android:layout_width="100dp"
 android:layout_height="100dp"
 android:onClick="showDonutOrder"
 android:layout_marginStart="@dimen/margin_wide"
 android:layout_marginTop="@dimen/margin_wide"
 android:contentDescription="@string/donuts"
 android:padding="10dp"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/title"
 app:srcCompat="@drawable/donut" />
```

Dòng cuối cùng (android:onClick="showDonutOrder") gán trình xử lý nhấp chuột (showDonutOrder) cho ImageView.

3. (Tùy chọn) Chọn **Code > Reformat Code** để định dạng lại mã XML bạn đã thêm vào content\_main.xml để tuân thủ các tiêu chuẩn và dễ đọc hơn. Android Studio tự động di chuyển thuộc tính android:onClick lên vài dòng để kết hợp chúng với các thuộc tính khác có android: làm tiền tố.

4. Thực hiện quy trình tương tự để thêm thuộc tính android:onClick vào các phần tử ImageView ice\_cream và froyo. Chọn các trình xử lý nhấp chuột showDonutOrder và showFroyoOrder. Bạn có thể tùy chọn chọn **Code > Reformat Code** để định dạng lại mã XML. Mã bây giờ sẽ trông như sau:

```
<ImageView
 android:id="@+id/froyo"
 android:layout_width="100dp"
 android:layout_height="100dp"
 android:layout_marginStart="@dimen/margin_wide"
 android:layout_marginTop="@dimen/margin_wide"
 android:contentDescription="@string/froyo"
 android:onClick="showFroyoOrder"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/ice_cream"
 app:srcCompat="@drawable/froyo" />
```

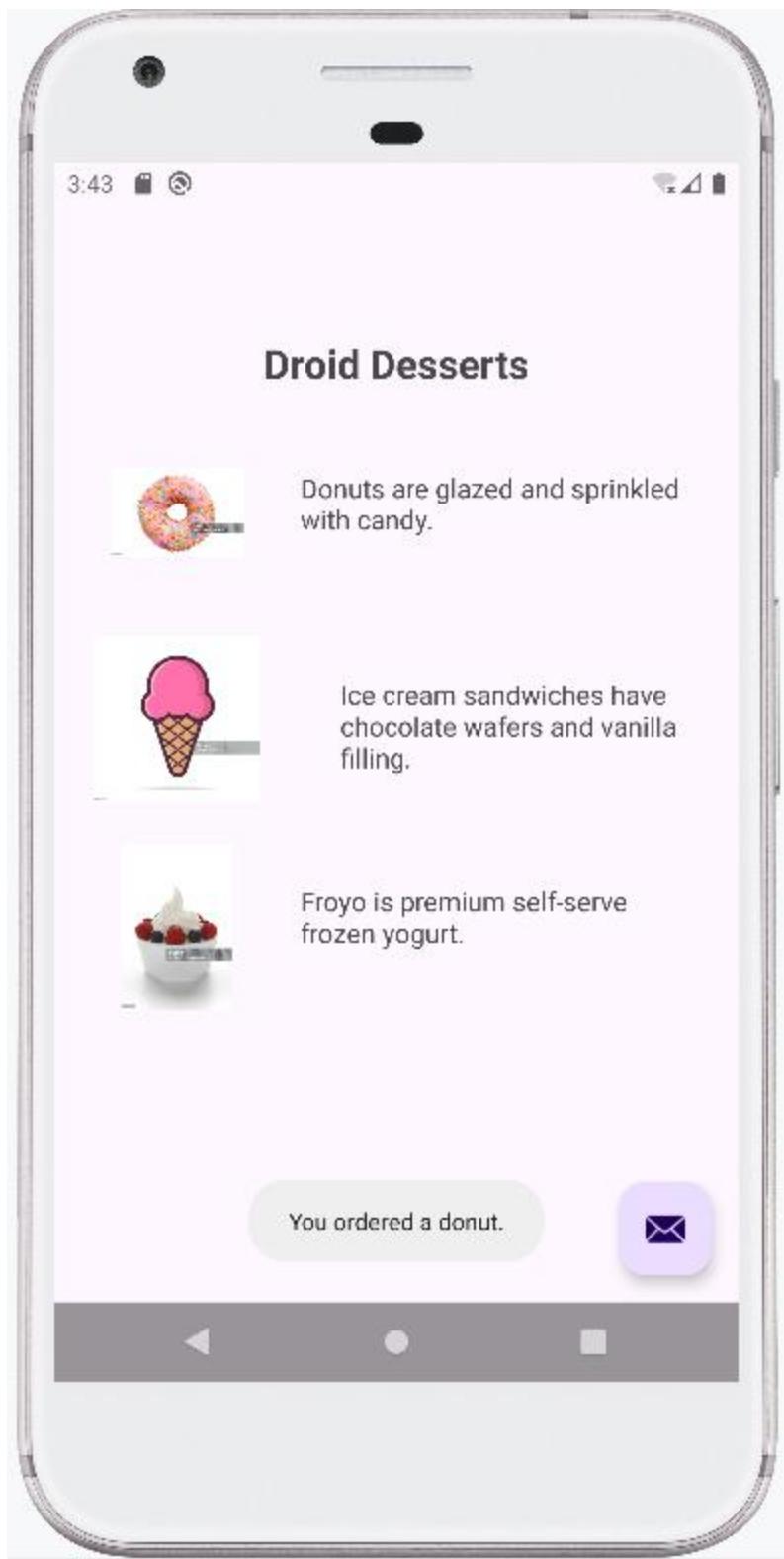
Lưu ý rằng thuộc tính android:layout\_marginStart trong mỗi ImageView được gạch chân bằng màu đỏ. Thuộc tính này xác định "lề" bắt đầu cho ImageView, mà bên trái đối với hầu hết các ngôn ngữ nhưng bên phải đối với các ngôn ngữ đọc từ phải sang trái (RTL).

5. Nhấn vào phần mở đầu android: của thuộc tính android:layout\_marginStart, và một biểu tượng đèn đỏ cảnh báo xuất hiện bên cạnh, như được hiển thị trong hình dưới đây.

6. Để làm cho ứng dụng của bạn tương thích với các phiên bản Android trước đó, hãy nhấp vào bóng đèn màu đỏ cho từng trường hợp của thuộc tính này và chọn **Set layout\_marginLeft...** để đặt layout\_marginLeft thành "@dimen/margin\_wide".

7. Chạy ứng dụng.

Nhấp vào hình ảnh bánh donut, bánh sandwich kem hoặc froyo sẽ hiển thị một thông báo Toast về đơn hàng, như hiển thị trong hình bên dưới.



Task 2 mã giải pháp

Mã giải pháp cho nhiệm vụ này được bao gồm trong mã và bộ cục của **MainActivity** trong dự án **DroidCafe** trên Android Studio.

### Task 3: Thay đổi nút cho hành động nổi

Khi bạn nhấn vào nút hành động nổi có biểu tượng email xuất hiện ở dưới cùng của màn hình, mã trong **MainActivity** sẽ hiển thị một tin nhắn ngắn trong một ngăn kéo mở ra từ dưới cùng của màn hình trên điện thoại thông minh hoặc từ góc dưới bên trái trên các thiết bị lớn hơn, sau đó tự động đóng sau vài giây. Đây được gọi là **Snackbar**. Nó được sử dụng để cung cấp phản hồi về một thao tác. Để biết thêm thông tin, hãy xem [Snackbar](#).

Hãy xem cách các ứng dụng khác triển khai **nút hành động nổi (Floating Action Button - FAB)**.

Ví dụ: Ứng dụng **Gmail** cung cấp một **nút hành động nổi** để tạo một email mới. Ứng dụng **Danh bạ (Contacts)** có một **nút hành động nổi** để tạo một liên hệ mới.

Để biết thêm thông tin về **nút hành động nổi**, hãy xem [FloatingActionButton](#).

Trong nhiệm vụ này, bạn sẽ thay đổi biểu tượng của **FloatingActionButton**



thành, và thay đổi hành động của **FloatingActionButton** để mở một **Activity** mới.

### 31. Thêm biểu tượng mới

Như bạn đã học trong bài học khác, bạn có thể chọn một biểu tượng từ bộ biểu tượng trong Android Studio.

Hãy làm theo các bước sau:

1. Mở rộng **res** trong bảng **Project > Android**, sau đó nhấp chuột phải (hoặc Control + nhấp) vào thư mục **drawable**.
2. Chọn **New > Image Asset**. Hộp thoại **Configure Image Asset** sẽ xuất hiện.
3. Trong menu thả xuống ở đầu hộp thoại, chọn **Action Bar and Tab Icons**. (Lưu ý rằng **action bar** chính là **app bar**.)

4. Thay đổi tên trong trường **Name** từ **ic\_action\_name** thành **ic\_shopping\_cart**.
5. Nhấp vào hình ảnh **clip art** (biểu tượng Android bên cạnh **Clipart:**) để chọn một hình ảnh clip art làm biểu tượng. Một trang chứa các biểu tượng sẽ xuất hiện. Nhấp vào biểu tượng bạn muốn sử dụng cho **Floating Action Button**, chẳng hạn như biểu tượng **giỏ hàng**.
6. Chọn **HOLO\_DARK** từ menu thả xuống **Theme**. Điều này giúp biểu tượng có màu trắng trên nền tối (hoặc đen). Nhấp vào **Next**.
7. Nhấp vào **Finish** trong hộp thoại **Confirm Icon Path**.

Mẹo: Để có mô tả đầy đủ về cách thêm biểu tượng, hãy xem **Tạo biểu tượng ứng dụng với Image Asset Studio**.

### 3.2 Thêm Activity

Như bạn đã học trong bài học trước, một **Activity** đại diện cho một màn hình trong ứng dụng, nơi người dùng có thể thực hiện một tác vụ cụ thể. Bạn đã có một activity là **MainActivity.java**. Nay, bạn sẽ thêm một activity mới có tên **OrderActivity.java**.

1. Nhấp chuột phải (hoặc Control + nhấp) vào thư mục **com.example.android.droidcafe** trong cột bên trái, sau đó chọn **New > Activity > Empty Activity**.
2. Chính sửa:
  - **Activity Name** thành **OrderActivity**.
  - **Layout Name** thành **activity\_order**.
3. Giữ nguyên các tùy chọn khác và nhấp vào **Finish**.

Sau khi hoàn thành, lớp **OrderActivity** sẽ xuất hiện cùng với **MainActivity** trong thư mục **java**, và tệp **activity\_order.xml** sẽ xuất hiện trong thư mục **layout**. **Empty Activity template** đã tự động tạo các tệp này.

### 3.3 Thay đổi hành động

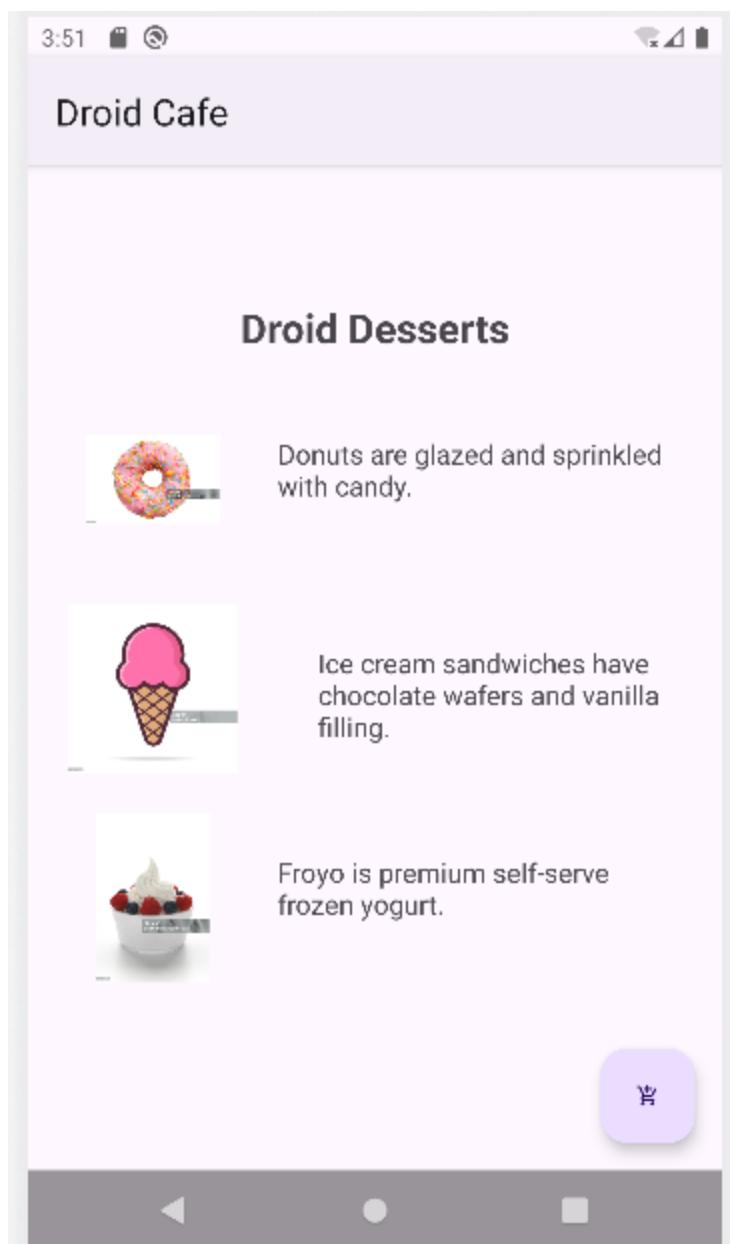
Trong bước này, bạn sẽ thay đổi hành động của FloatingActionButton để mở Activity mới.

1. Mở **MainActivity**.

- Thay đổi phương thức **onClick(View view)** để tạo một **explicit intent** nhằm khởi động **OrderActivity**.

```
public void onClick(View view) {}
 Intent intent = new Intent(packageContext: this, OrderActivity.class);
 startActivity(intent);
}
```

- Chạy ứng dụng. Nhấn vào **floating action button** hiện đang sử dụng biểu tượng giỏ hàng. Một **Activity** trống (**OrderActivity**) sẽ xuất hiện. Nhấn nút **Back** để quay lại **MainActivity**.



### Task 3 Mã giải pháp

Mã giải pháp cho nhiệm vụ này được bao gồm trong mã và bộ cục của dự án Android Studio DroidCafe.

### Thử thách lập trình

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

**Thử thách:** Ứng dụng DroidCafe có **MainActivity** khởi động một **Activity** thứ hai có tên **OrderActivity**.

Bạn đã học trong bài học khác cách gửi dữ liệu từ một Activity này sang Activity khác. Hãy thay đổi ứng dụng để gửi thông điệp đơn hàng cho món tráng miệng đã chọn trong MainActivity đến một TextView mới ở trên cùng của bố cục OrderActivity.

1. Thêm một TextView ở trên cùng của bố cục OrderActivity với id là `order_textview`.
2. Tạo một biến thành viên (`mOrderMessage`) trong MainActivity cho thông điệp đơn hàng sẽ hiển thị trong Toast.
3. Thay đổi các bộ xử lý nhấp chuột `showDonutOrder()`, `showIceCreamOrder()`, và `showFroyoOrder()` để gán chuỗi thông điệp `mOrderMessage` trước khi hiển thị Toast. Ví dụ, đoạn mã sau đây gán chuỗi `donut_order_message` cho `mOrderMessage` và hiển thị Toast:

```
mOrderMessage = getString(R.string.donut_order_message);
displayToast(mOrderMessage);
```
4. Thêm một public static final String có tên `EXTRA_MESSAGE` ở đầu MainActivity để định nghĩa khóa cho `intent.putExtra`:

```
public static final String EXTRA_ORDER_MESSAGE = "com.example.droidcafe.extra.ORDER_MESSAGE";
```

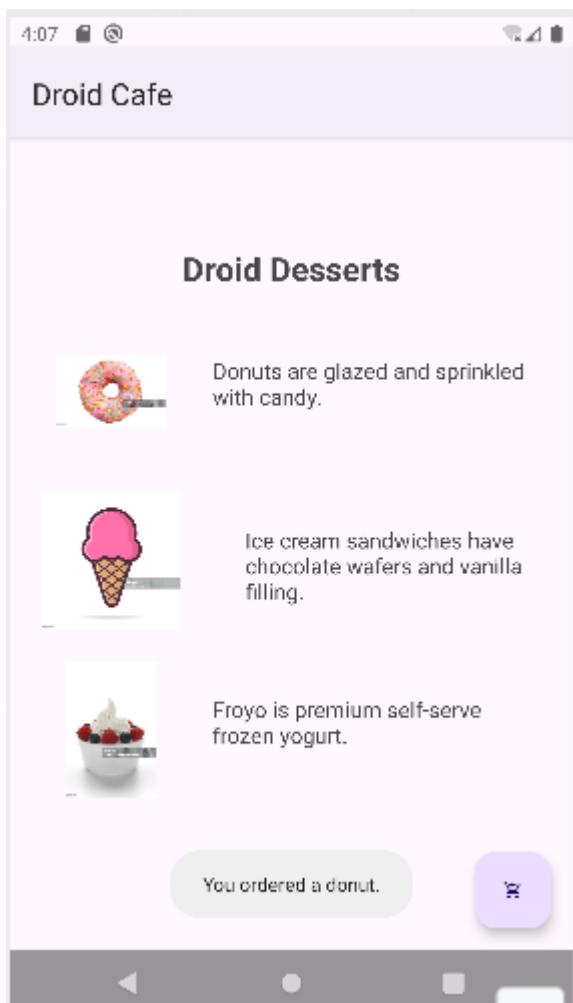
5. Thay đổi phương thức `onClick()` để bao gồm câu lệnh `intent.putExtra` trước khi khởi động OrderActivity:

```
intent.putExtra(EXTRA_ORDER_MESSAGE, mOrderMessage);
```

6. Trong OrderActivity, thêm mã sau vào phương thức `onCreate()` để lấy Intent khởi động Activity, trích xuất thông điệp chuỗi, và thay thế văn bản trong TextView với thông điệp:

```
Intent intent = getIntent();
String message = "Order: " +
 intent.getStringExtra(MainActivity.EXTRA_ORDER_MESSAGE);
TextView textView = findViewById(R.id.order_textview);
textView.setText(message);
```

7. Chạy ứng dụng. Sau khi chọn hình ảnh món tráng miệng, nhấn vào floating action button để khởi động OrderActivity, và OrderActivity sẽ hiển thị thông điệp đơn hàng như trong hình dưới đây.





## Giải pháp cho thử thách

Dự án Android Studio: [DroidCafeChallenge](#)

### Tóm tắt

- Để sử dụng một hình ảnh trong dự án, sao chép hình ảnh vào thư mục **drawable** của dự án ( **project\_name > app > src > main > res > drawable**).
- Định nghĩa một **ImageView** để sử dụng nó bằng cách kéo thả **ImageView** vào bố cục và chọn hình ảnh cho nó.
- Thêm thuộc tính **android:onClick** để làm cho **ImageView** có thể nhấp được như một nút. Chỉ định tên của bộ xử lý nhấp chuột.
- Tạo một bộ xử lý nhấp chuột trong **Activity** để thực hiện hành động.
- Chọn biểu tượng: Mở rộng **res** trong bảng **Project > Android**, nhấp chuột phải (hoặc Control + nhấp) vào thư mục **drawable**, và chọn **New > Image Asset**. Chọn **Action Bar and Tab Icons** trong menu thả xuồng, và nhấp vào hình ảnh **clip art** (biểu tượng Android bên cạnh **Clipart:**) để chọn một hình ảnh

clip art làm biểu tượng.

- Thêm một **Activity** khác: Trong bảng **Project > Android**, nhấp chuột phải (hoặc Control + nhấp) vào thư mục tên gói trong thư mục **java** và chọn **New > Activity** và một mẫu cho **Activity** (chẳng hạn như **Empty Activity**).
- Hiển thị một **Toast** message.

## **Khái niệm liên quan**

Tài liệu về khái niệm liên quan có trong **4.1: Nút và hình ảnh có thể nhấp**.

## **Tìm hiểu thêm**

Tài liệu Android Studio:

- **Hướng dẫn người dùng Android Studio**
- **Tạo biểu tượng ứng dụng với Image Asset Studio**

Tài liệu dành cho nhà phát triển Android:

- **Giao diện người dùng và điều hướng**
- **Xây dựng giao diện người dùng với Layout Editor**
- **Xây dựng giao diện người dùng đáp ứng với ConstraintLayout**
- **Layouts**
- **View**
- **Button**
- **ImageView**
- **TextView**
- **Buttons**
- **Styles and themes**

Khác:

- **Codelabs: Sử dụng ConstraintLayout để thiết kế các view Android của bạn**

## **Bài tập về nhà**

### **Thay đổi ứng dụng**

Ứng dụng **DroidCafe** hiển thị tốt khi thiết bị hoặc trình giả lập được xoay theo hướng dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang hướng ngang, các hình ảnh thứ hai và thứ ba không xuất hiện.

1. Mở (hoặc tải về) dự án ứng dụng **DroidCafe**.
2. Tạo một biến thẻ bô cục cho hướng ngang: **content\_main.xml (land)**.
3. Loại bỏ các ràng buộc từ ba hình ảnh và ba mô tả văn bản.

4. Chọn tất cả ba hình ảnh trong biến thể bố cục, và chọn **Expand Horizontally** trong nút **Pack** để phân phối đều các hình ảnh trên màn hình như hình dưới đây.
5. Ràng buộc các mô tả văn bản vào các cạnh và đáy của hình ảnh như hình dưới đây.

### Câu hỏi 1

Làm thế nào để bạn thêm hình ảnh vào một dự án Android Studio? Chọn một trong các đáp án sau:

- Kéo từng hình ảnh vào **layout editor**.
- Sao chép các tệp hình ảnh vào thư mục **drawable** của dự án.
- Kéo một **ImageButton** vào **layout editor**.
- Chọn **New > Image Asset** và sau đó chọn tệp hình ảnh.

### Câu hỏi 2

Làm thế nào để làm cho một **ImageView** có thể nhấp được như một nút đơn giản? Chọn một trong các đáp án sau:

- Thêm thuộc tính **android:contentDescription** vào **ImageView** trong bố cục và sử dụng nó để gọi bộ xử lý nhấp chuột trong **Activity**.
- Thêm thuộc tính **android:src** vào **ImageView** trong bố cục và sử dụng nó để gọi bộ xử lý nhấp chuột trong **Activity**.
- Thêm thuộc tính **android:onClick** vào **ImageView** trong bố cục và sử dụng nó để gọi bộ xử lý nhấp chuột trong **Activity**.
- Thêm thuộc tính **android:id** vào **ImageView** trong bố cục và sử dụng nó để gọi bộ xử lý nhấp chuột trong **Activity**.

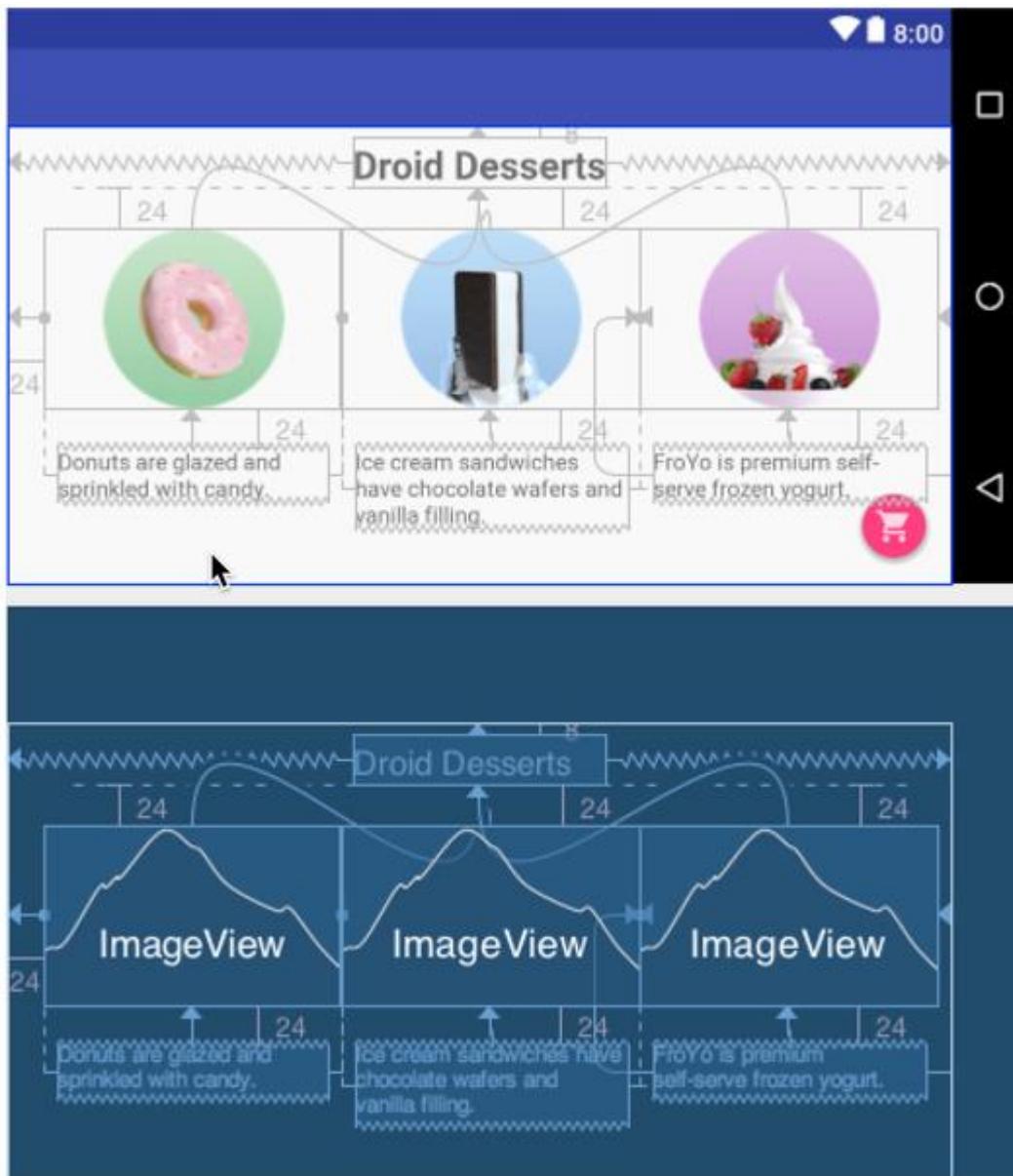
### Câu hỏi 3

Quy tắc nào áp dụng cho một bộ xử lý nhấp chuột được gọi từ thuộc tính trong bố cục? Chọn một trong các đáp án sau:

- Phương thức bộ xử lý nhấp chuột phải bao gồm **event listener View.OnClickListener**, đây là một giao diện trong lớp **View**.
- Phương thức bộ xử lý nhấp chuột phải là **public**, trả về **void**, và định nghĩa một **View** làm tham số duy nhất.
- Bộ xử lý nhấp chuột phải tùy chỉnh lớp **View.OnClickListener** và ghi đè bộ xử lý nhấp chuột của nó để thực hiện một hành động nào đó.
- Phương thức bộ xử lý nhấp chuột phải là **private** và trả về một **View**.

## Nộp ứng dụng để chấm điểm Hướng dẫn cho người chấm điểm

1. Chạy ứng dụng.
2. Chuyển sang chế độ ngang để xem biến thể bố cục mới. Nó sẽ trông giống như hình dưới đây.



## Bài 4.2 : Các điều khiển nhập liệu

Giới thiệu

Để cho phép người dùng nhập văn bản hoặc số, bạn sử dụng phần tử **EditText**. Một số điều khiển nhập liệu là các thuộc tính của **EditText** định nghĩa loại bàn phím sẽ hiển thị, giúp việc nhập dữ liệu trở nên dễ dàng hơn cho người dùng. Ví dụ, bạn có thể chọn phone cho thuộc tính android:inputType để hiển thị bàn phím số thay vì bàn phím chữ cái và số.

Các điều khiển nhập liệu khác giúp người dùng dễ dàng đưa ra lựa chọn. Ví dụ, phần tử **RadioButton** cho phép người dùng chọn một (và chỉ một) mục trong một tập hợp các mục.

Trong bài thực hành này, bạn sẽ sử dụng các thuộc tính để điều khiển giao diện bàn phím trên màn hình, và để đặt loại dữ liệu nhập vào cho **EditText**. Bạn cũng sẽ thêm các nút radio vào ứng dụng **DroidCafe** để người dùng có thể chọn một mục từ một tập hợp các mục.

### Những gì bạn đã biết

Bạn nên có khả năng:

- Tạo một dự án Android Studio từ một mẫu và tạo bố cục chính.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị kết nối.
- Tạo và chỉnh sửa các phần tử giao diện người dùng (UI) bằng trình chỉnh sửa bố cục và mã XML.
- Truy cập các phần tử giao diện người dùng từ mã của bạn bằng cách sử dụng **findViewById()**.
- Chuyển văn bản trong một **View** thành một chuỗi bằng cách sử dụng **getText()**.
- Tạo một bộ xử lý nhập chuột cho nút **Button**.
- Hiển thị một thông báo **Toast**.

### Những gì bạn sẽ học

- Cách thay đổi phương thức nhập liệu để bật gợi ý, tự động viết hoa và che mật khẩu.
- Cách thay đổi bàn phím trên màn hình chung thành bàn phím điện thoại hoặc các bàn phím chuyên dụng khác.
- Cách thêm các nút radio cho người dùng để chọn một mục trong một tập hợp các mục.
- Cách thêm một spinner để hiển thị một menu thả xuống với các giá trị, từ đó người dùng có thể chọn một giá trị.

### Những gì bạn sẽ làm

- Hiển thị bàn phím để nhập địa chỉ email.

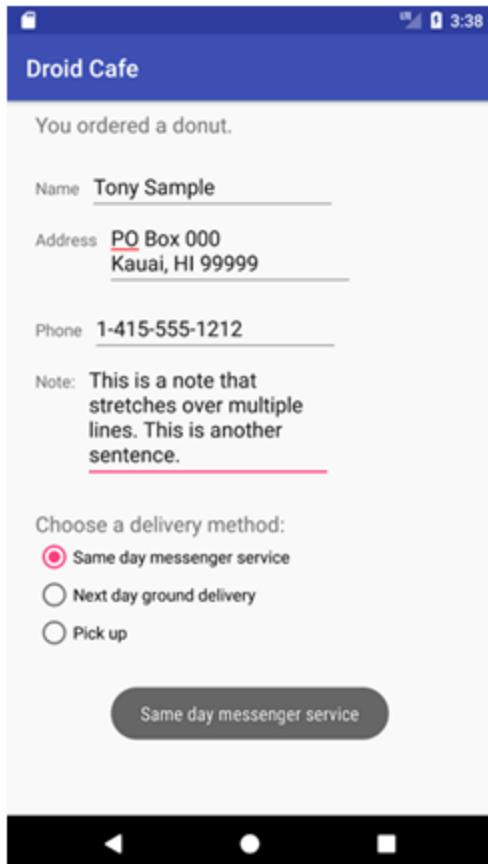
- Hiển thị bàn phím số để nhập số điện thoại.
- Cho phép nhập văn bản nhiều dòng với tự động viết hoa câu.
- Thêm các nút radio để chọn một tùy chọn.
- Đặt một bộ xử lý **onClick** cho các nút radio.
- Thêm một spinner cho trường số điện thoại để chọn một giá trị từ một tập hợp các giá trị.

## Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ thêm nhiều tính năng vào ứng dụng DroidCafe từ bài học về cách sử dụng hình ảnh có thể nháp.

Trong OrderActivity của ứng dụng, bạn sẽ thử nghiệm với thuộc tính `android:inputType` cho các phần tử `EditText`. Bạn thêm các phần tử `EditText` để nhập tên và địa chỉ của người dùng, và sử dụng các thuộc tính để định nghĩa các phần tử một dòng và nhiều dòng có tính năng gợi ý khi bạn nhập văn bản. Bạn cũng thêm một `EditText` hiển thị bàn phím số để nhập số điện thoại.

Các loại điều khiển nhập liệu khác bao gồm các phần tử tương tác giúp người dùng đưa ra lựa chọn. Bạn thêm các nút radio vào DroidCafe để chọn chỉ một tùy chọn giao hàng từ nhiều tùy chọn. Bạn cũng cung cấp một điều khiển nhập liệu dạng spinner để chọn nhãn ( Home , Work , Other , Custom ) cho số điện thoại.



## Nhiệm vụ 1: Thử nghiệm với các thuộc tính nhập liệu văn bản

Khi chạm vào trường văn bản có thể chỉnh sửa **EditText**, con trỏ sẽ xuất hiện trong trường văn bản và bàn phím trên màn hình sẽ tự động hiển thị để người dùng có thể nhập văn bản.

Một trường văn bản có thể chỉnh sửa kỳ vọng một loại văn bản nhập vào nhất định, chẳng hạn như văn bản thuần túy, địa chỉ e mail, số điện thoại hoặc mật khẩu. Việc chỉ định loại nhập liệu cho mỗi trường văn bản trong ứng dụng là rất quan trọng để hệ thống hiển thị phương pháp nhập liệu phù hợp, chẳng hạn như bàn phím trên màn hình cho văn bản thuần túy hoặc bàn phím số để nhập số điện thoại.

### 1.1 Thêm một **EditText** để nhập tên

Trong bước này, bạn sẽ thêm một **TextView** và một **EditText** vào bộ cục **OrderActivity** trong ứng dụng **DroidCafe** để người dùng có thể nhập tên của một người.

1. Sao chép ứng dụng **DroidCafe** từ bài học về sử dụng hình ảnh có thể nhấp, và đổi tên bản sao thành **DroidCafeInput**. Nếu bạn chưa hoàn thành thử

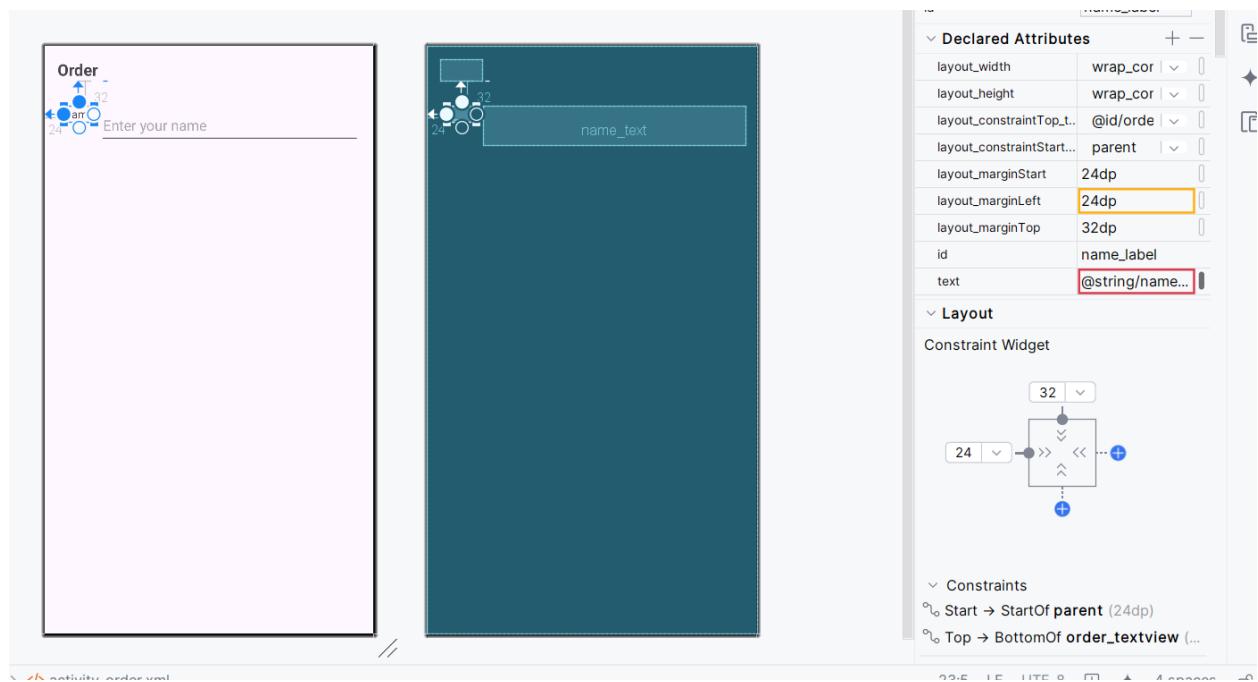
thách lập trình trong bài học đó, hãy tải dự án **DroidCafeChallenge** và đổi tên thành **DroidCafeInput**.

2. Mở tệp bố cục **activity\_order.xml**, tệp này sử dụng **ConstraintLayout**.
3. Thêm một **TextView** vào **ConstraintLayout** trong **activity\_order.xml** dưới phần tử **order\_textview** đã có trong bố cục. Sử dụng các thuộc tính sau cho **TextView** mới.

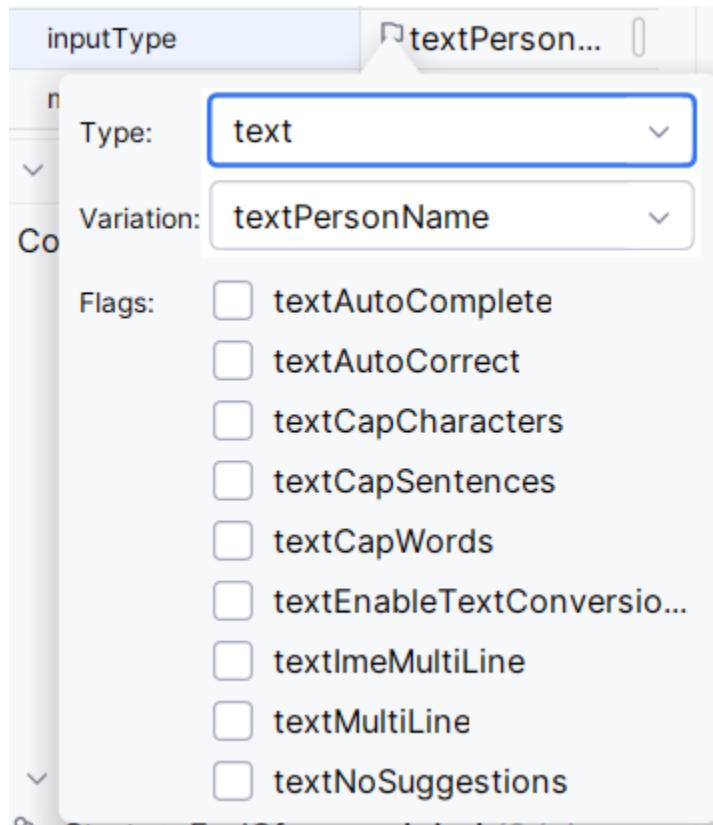
<b>TextView attribute</b>	<b>Value</b>
android:id	"@+id/name_label"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	"24dp"
android:layout_marginLeft	"24dp"

android:layout_marginTop	"32dp"
android:text	"Name"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/order_textview"

4. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính **android:text** để tạo một mục mới có tên **name\_label\_text** trong **strings.xml**.
5. Thêm một phần tử **EditText**. Để sử dụng trình chỉnh sửa bố cục trực quan, kéo một phần tử **Plain Text** từ bảng **Palette** vào vị trí bên cạnh **name\_label TextView**. Sau đó nhập **name\_text** cho trường **ID** và ràng buộc cạnh trái và đường chéo của phần tử với cạnh phải và đường chéo của phần tử **name\_label** như hình dưới đây.



6. Hình trên làm nổi bật trường **inputType** trong bảng **Attributes** để cho thấy Android Studio đã tự động chỉ định kiểu **textPersonName**. Nhập vào trường **inputType** để xem menu các loại nhập liệu.



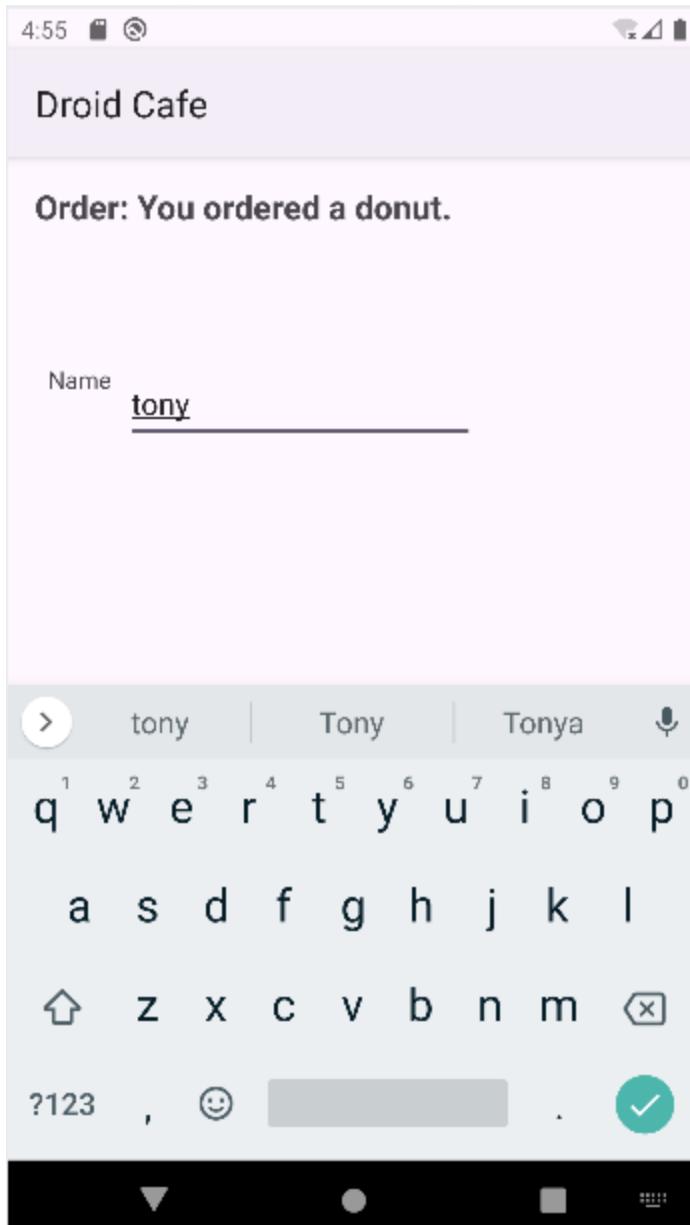
- Thêm một gợi ý cho việc nhập văn bản, chẳng hạn như **Enter your name**, vào trường **hint** trong bảng **Attributes**, và xóa mục **Name** trong trường **text**. Dưới dạng một gợi ý cho người dùng, văn bản "Enter your name" sẽ mờ trong **EditText**.
- Kiểm tra mã XML cho bố cục bằng cách nhấp vào tab **Text**. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính **android:hint** thành **enter\_name\_hint**. Các thuộc tính sau nên được thiết lập cho **EditText** mới (thêm thuộc tính **layout\_marginLeft** để tương thích với các phiên bản Android cũ hơn): Như bạn có thể thấy trong mã XML, thuộc tính **android:inputType** được đặt thành **textPersonName**.

<b>EditText attribute</b>	<b>Value</b>
<b>android:id</b>	"@+id/name_text"
<b>android:layout_width</b>	"wrap_content"
<b>android:layout_height</b>	"wrap_content"
<b>android:layout_marginStart</b>	8dp
<b>android:layout_marginLeft</b>	8dp
<b>android:ems</b>	"10"
<b>android:hint</b>	"@string/enter_name_hint"
<b>android:inputType</b>	"textPersonName"
<b>app:layout_constraintBaseline_toBaselineOf</b>	"@+id/name_label"
<b>app:layout_constraintStart_toEndOf</b>	"@+id/name_label"

- Chạy ứng dụng. Nhấn vào hình ảnh donut trên màn hình đầu tiên, sau đó nhán vào nút hành động nổi để xem **Activity** tiếp theo. Chạm vào trường nhập văn bản để hiển thị bàn phím và nhập văn bản, như hình dưới đây. Lưu ý rằng các gợi ý tự động xuất hiện cho các từ bạn nhập. Nhấn vào một gợi ý để sử dụng nó. Đây là một trong các tính năng của giá trị **textPersonName** cho thuộc tính **android:inputType**. Thuộc tính **inputType** điều khiển nhiều

tính năng, bao gồm cách bố trí bàn phím, việc viết hoa và việc nhập văn bản nhiều dòng.





10. Để đóng bàn phím, nhấn vào biểu tượng trong vòng tròn màu xanh lá cây, xuất hiện ở góc dưới bên phải của bàn phím. Đây được gọi là phím **Done**.

## 1.2 Thêm một EditText nhiều dòng

Trong bước này, bạn sẽ thêm một **EditText** khác vào bố cục **OrderActivity** trong ứng dụng **DroidCafe** để người dùng có thể nhập địa chỉ bằng nhiều dòng.

1. Mở tệp bố cục **activity\_order.xml** nếu nó chưa được mở.

2. Thêm một TextView bên dưới phần tử name\_label đã có trong bố cục. Sử dụng các thuộc tính sau cho TextView mới.

TextView attribute	Value
android:id	"@+id/address_label"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	"24dp"
android:layout_marginLeft	"24dp"
android:layout_marginTop	"24dp"
android:text	"Address"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/name_label"

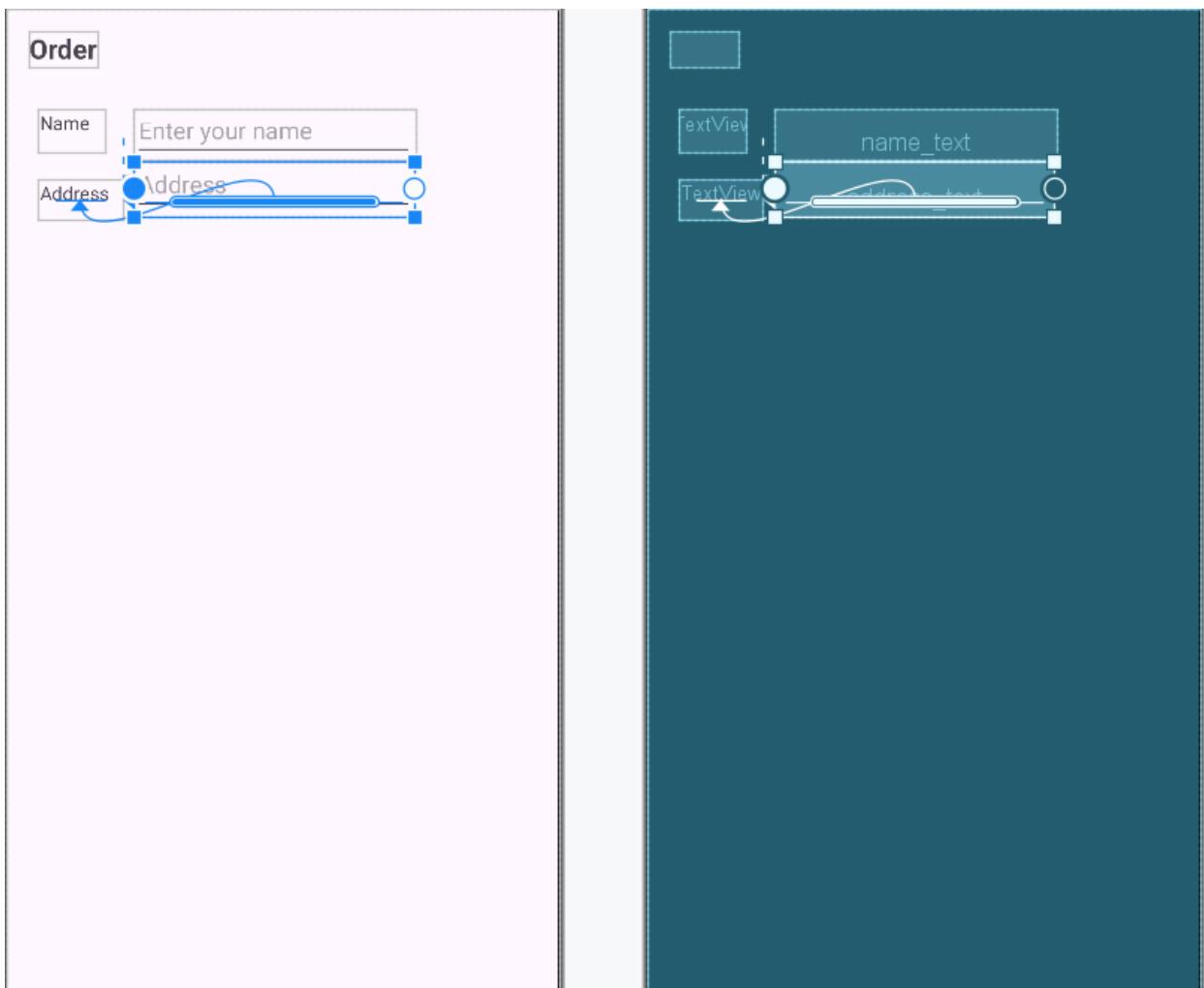
3. Trích xuất giá trị thuộc tính android:text thành một tài nguyên chuỗi mới trong strings.xml với tên address\_label\_text.

4. Thêm một phần tử EditText.

Trong trình chỉnh sửa bố cục trực quan, kéo một phần tử Multiline Text từ Palette đến vị trí bên cạnh TextView có ID address\_label.

Đặt address\_text làm giá trị cho ID của EditText.

Thiết lập ràng buộc bên trái và đường cơ sở của nó với address\_label như trong hình minh họa.



5. Thêm một gợi ý nhập văn bản, chẳng hạn như "Enter address", vào trường hint trong bảng thuộc tính Attributes.

Gợi ý này sẽ hiển thị dưới dạng văn bản mờ bên trong EditText.

6. Kiểm tra mã XML của bộ cục bằng cách nhấp vào tab Text.

Trích xuất giá trị thuộc tính android:hint thành một tài nguyên chuỗi mới có tên enter\_address\_hint.

Đảm bảo các thuộc tính sau được đặt cho EditText mới (thêm thuộc tính layout\_marginLeft để tương thích với các phiên bản Android cũ hơn).

<b>EditText attribute</b>	<b>Value</b>
android:id	"@+id/address_text"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	8dp
android:layout_marginLeft	8dp
android:ems	"10"
android:hint	"@string/enter_address_hint"
android:inputType	"textMultiLine"
app:layout_constraintBaseline_toBaselineOf	"@+id/address_label"
app:layout_constraintStart_toEndOf	"@+id/address_label"

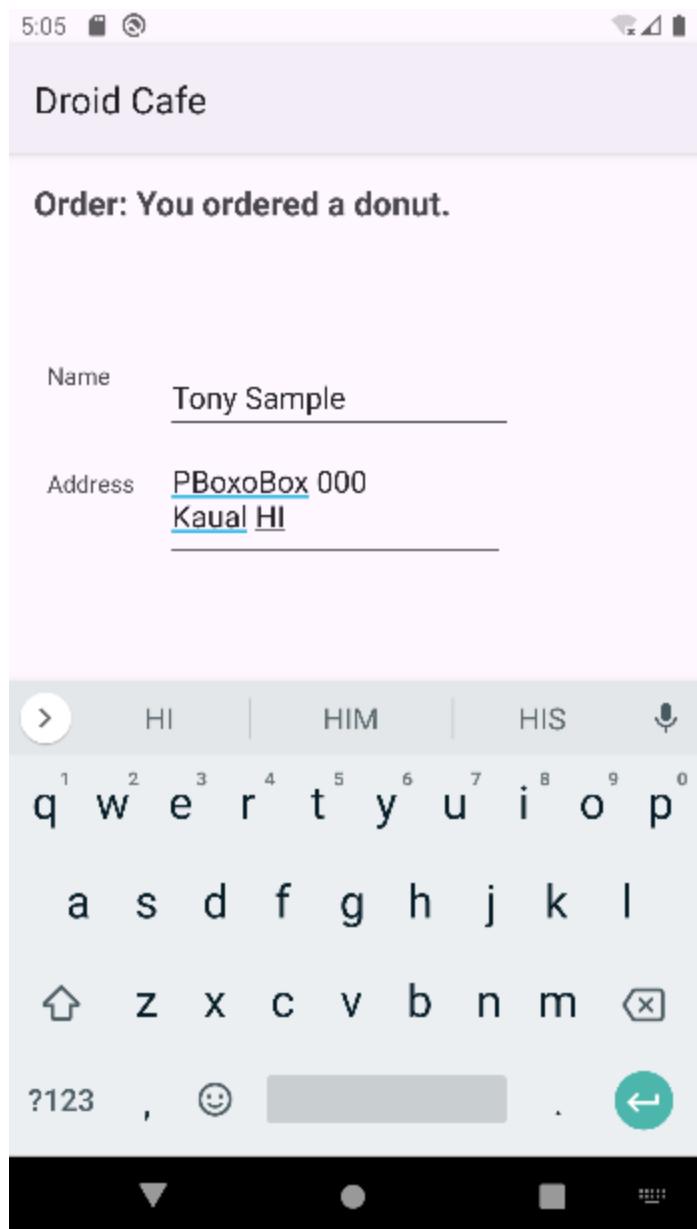
## 7. Chạy ứng dụng.

Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào Floating Action Button để chuyển đến OrderActivity.

## 8. Nhấn vào trường nhập "Address" để hiển thị bàn phím và nhập văn bản.

Sử dụng phím Return ở góc dưới bên phải của bàn phím (còn gọi là phím Enter hoặc New Line) để xuống dòng khi nhập văn bản.

Phím Return sẽ xuất hiện nếu bạn đặt thuộc tính android:inputType thành textMultiLine.



9. Để đóng bàn phím, nhấn vào nút mũi tên xuống xuất hiện thay vì nút Back trên hàng nút dưới cùng.

### 1.3 Sử dụng bàn phím số cho số điện thoại

Trong bước này, bạn sẽ thêm một EditText khác vào bố cục OrderActivity trong ứng dụng DroidCafe để người dùng có thể nhập số điện thoại bằng bàn phím số.

1.Mở tệp activity\_order.xml nếu nó chưa được mở.

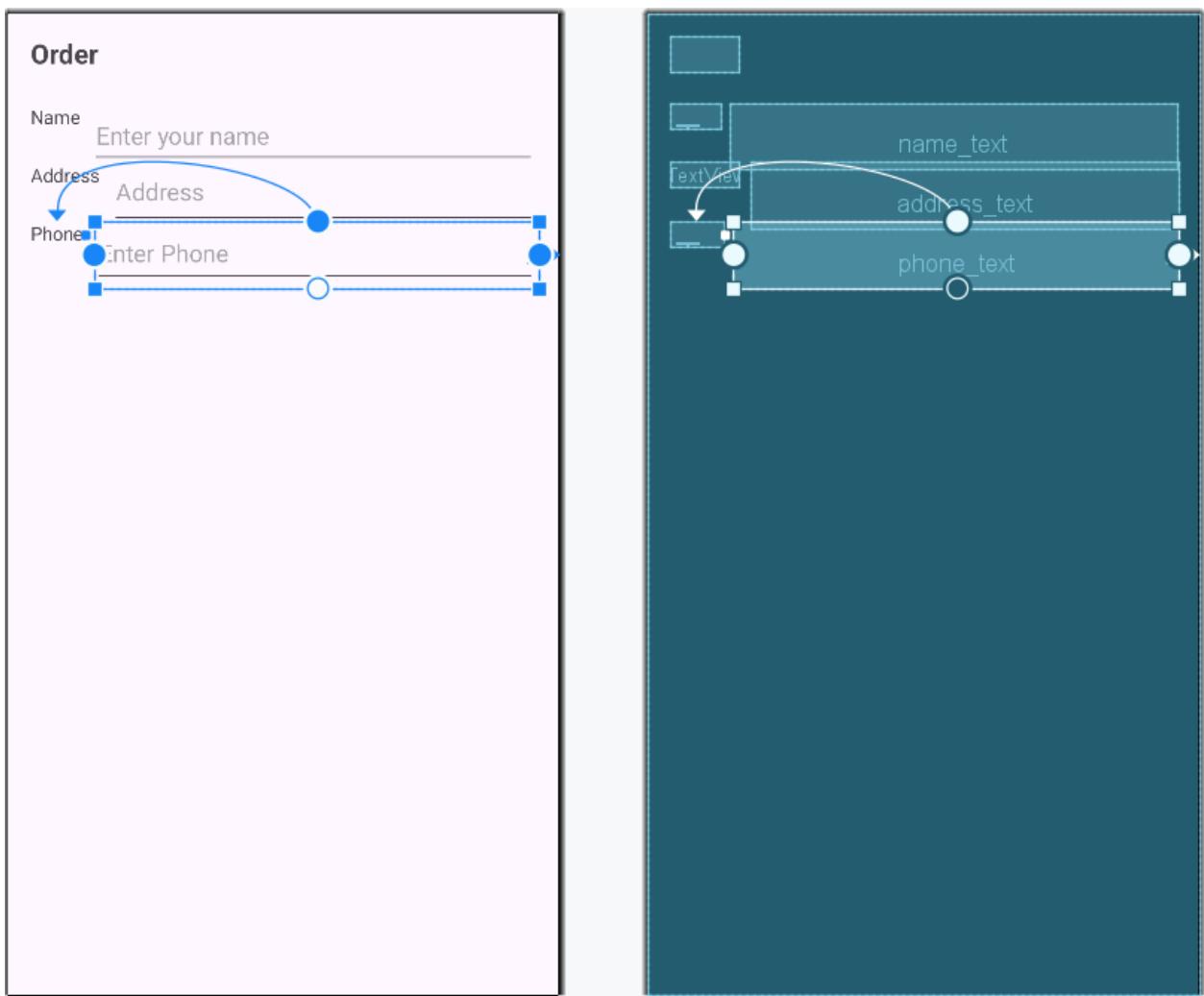
2.Thêm một TextView bên dưới phần tử address\_label đã có trong bố cục. Sử dụng các thuộc tính sau cho TextView mới:

TextView attribute	Value
android:id	"@+id/phone_label"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	"24dp"
android:layout_marginLeft	"24dp"
android:layout_marginTop	"24dp"
android:text	"Phone"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/address_text"

Lưu ý rằng TextView này được ràng buộc với đáy của EditText nhiều dòng (address\_text). Điều này là do address\_text có thể phát triển thành nhiều dòng, và TextView này sẽ xuất hiện bên dưới nó.

3.Trích xuất chuỗi tài nguyên cho giá trị thuộc tính android:text để tạo một mục nhập cho nó với tên phone\_label\_text trong tệp strings.xml.

4.Thêm một phần tử EditText. Để sử dụng trình chỉnh sửa bố cục trực quan, kéo một phần tử Phone từ bảng Palette vào vị trí bên cạnh TextView có ID phone\_label. Sau đó, đặt ID của nó là phone\_text, và ràng buộc cạnh trái và dòng cơ sở của phần tử này với cạnh phải và dòng cơ sở của phone\_label, như trong hình dưới đây:



5.Thêm một gợi ý nhập văn bản, chẳng hạn như Enter phone, trong trường hint trong bảng Attributes. Gợi ý "Enter phone" sẽ hiển thị dưới dạng văn bản mờ bên trong EditText.

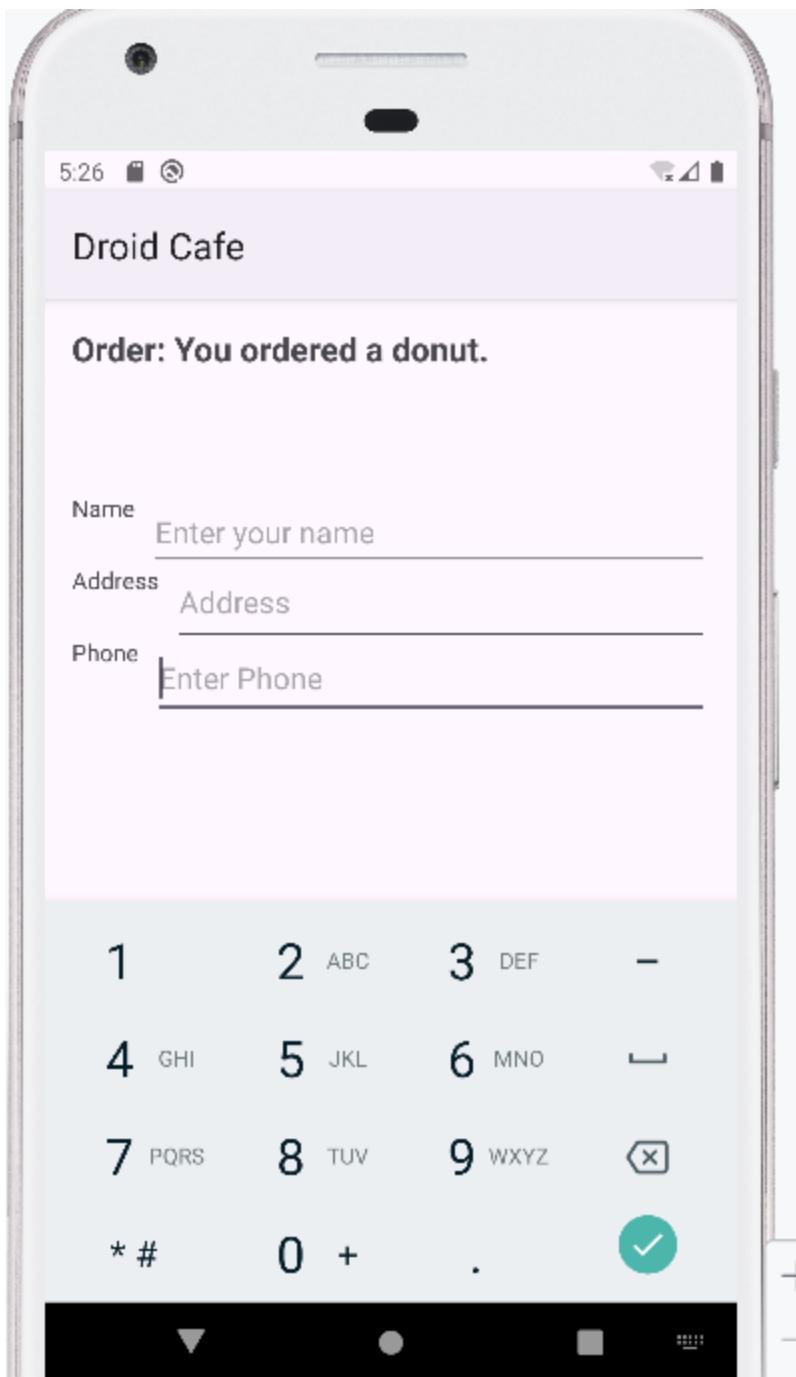
6.Kiểm tra mã XML cho bộ cục bằng cách nhấp vào tab Text. Trích xuất chuỗi tài nguyên cho giá trị thuộc tính android:hint thành enter\_phone\_hint. Các thuộc tính sau đây nên được đặt cho EditText mới (thêm thuộc tính layout\_marginLeft để tương thích với các phiên bản Android cũ hơn):

<b>EditText attribute</b>	<b>Value</b>
android:id	"@+id/phone_text"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	8dp
android:layout_marginLeft	8dp

android:ems	"10"
android:hint	"@string/enter_phone_hint"
android:inputType	"phone"
app:layout_constraintBaseline_toBaselineOf	"@+id/phone_label"
app:layout_constraintStart_toEndOf	"@+id/phone_label"

7.Chạy ứng dụng. Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem Activity tiếp theo.

8.Nhấn vào trường "Phone" để hiển thị bàn phím số. Bạn có thể nhập số điện thoại, như hình dưới đây.



9. Để đóng bàn phím, nhấn vào phím Done.

Thử nghiệm với thuộc tính android:inputType

Hãy thay đổi giá trị thuộc tính android:inputType của một phần tử EditText để xem kết quả:

- `textEmailAddress`: Khi nhấn vào trường, bàn phím nhập email sẽ xuất hiện với ký hiệu @ nằm gần phím cách.
- `textPassword`: Các ký tự mà người dùng nhập sẽ biến thành dấu chấm để ẩn mật khẩu đã nhập.

#### 1.4 Kết hợp nhiều kiểu nhập liệu trong một EditText

Bạn có thể kết hợp các giá trị thuộc tính `inputType` không xung đột với nhau. Ví dụ, bạn có thể kết hợp các giá trị `textMultiLine` và `textCapSentences` để tạo một ô nhập văn bản nhiều dòng, trong đó mỗi câu bắt đầu bằng một chữ cái viết hoa.

1. Mở tệp `activity_order.xml` nếu nó chưa được mở.
2. Thêm một `TextView` bên dưới phần tử `phone_label` đã có trong bố cục. Sử dụng các thuộc tính sau cho `TextView` mới.

<b>TextView attribute</b>	<b>Value</b>
<code>android:id</code>	"@+id/note_label"
<code>android:layout_width</code>	"wrap_content"
<code>android:layout_height</code>	"wrap_content"
<code>android:layout_marginStart</code>	"24dp"
<code>android:layout_marginLeft</code>	"24dp"
<code>android:layout_marginTop</code>	"24dp"
<code>android:text</code>	"Note"
<code>app:layout_constraintStart_toStartOf</code>	"parent"
<code>app:layout_constraintTop_toBottomOf</code>	"@+id/phone_label"

3. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính `android:text` để tạo một mục nhập trong tệp `strings.xml` với tên `note_label_text`.

4. Thêm một phần tử EditText. Để sử dụng trình chỉnh sửa bô cục trực quan, kéo một phần tử "Multiline Text" từ bảng Palette đến vị trí bên cạnh phần tử note\_label TextView. Sau đó, nhập note\_text vào trường ID, và ràng buộc cạnh trái và đường cơ sở của phần tử này với cạnh phải và đường cơ sở của phần tử note\_label như bạn đã làm trước đó với các phần tử EditText khác.
5. Thêm gợi ý nhập văn bản, chẳng hạn như "Enter note", trong trường hint trong bảng thuộc tính Attributes.
6. Nhấp vào bên trong trường inputType trong bảng thuộc tính Attributes. Giá trị textMultiLine đã được chọn sẵn. Ngoài ra, hãy chọn thêm textCapSentences để kết hợp hai thuộc tính này.
7. Kiểm tra mã XML của bô cục bằng cách nhấp vào tab Text. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:hint với tên enter\_note\_hint. Các thuộc tính sau đây nên được thiết lập cho phần tử EditText mới (thêm thuộc tính layout\_marginLeft để đảm bảo tương thích với các phiên bản Android cũ hơn):

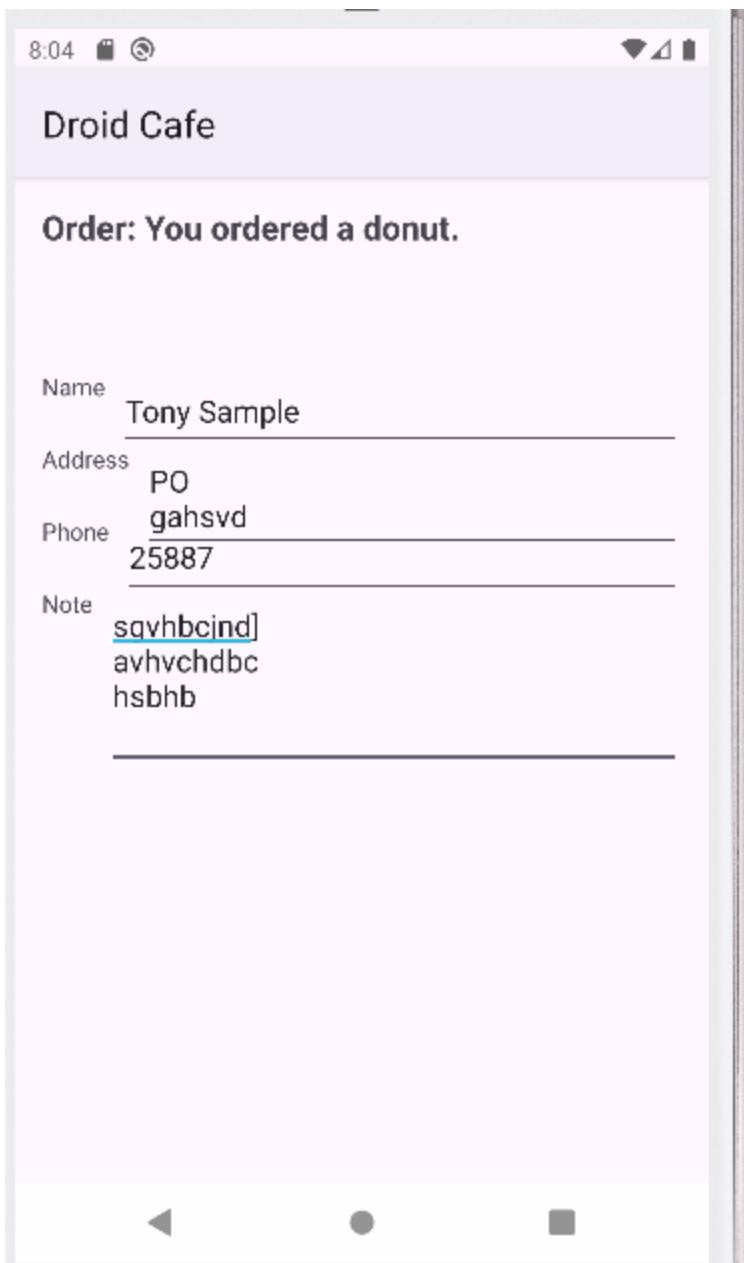
<b>EditText attribute</b>	<b>Value</b>
android:id	"@+id/note_text"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	8dp
android:layout_marginLeft	8dp

android:ems	"10"
android:hint	"@string/enter_note_hint"
android:inputType	"textCapSentences textMultiLine"
app:layout_constraintBaseline_toBaselineOf	"@+id/note_label"
app:layout_constraintStart_toEndOf	"@+id/note_label"

Để kết hợp nhiều giá trị cho thuộc tính android:inputType, hãy nối chúng bằng ký tự dấu gạch đứng |.

8. Chạy ứng dụng. Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem Activity tiếp theo.

9. Nhấn vào ô nhập "Note" để nhập câu hoàn chỉnh, như hình minh họa bên dưới. Sử dụng phím Return để tạo một dòng mới hoặc chỉ cần nhập để tự động xuống dòng trong nhiều dòng.



## Task 2: Sử dụng nút radio

Các điều khiển nhập liệu là các phần tử tương tác trong giao diện người dùng (UI) của ứng dụng, cho phép người dùng nhập dữ liệu. Nút radio là một loại điều khiển nhập liệu hữu ích khi bạn muốn người dùng chỉ chọn một tùy chọn từ một tập hợp các tùy chọn.

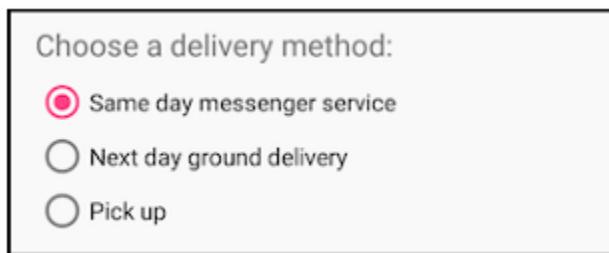
**Mẹo:** Bạn nên sử dụng nút radio nếu bạn muốn người dùng nhìn thấy tất cả các tùy chọn có sẵn bên cạnh nhau. Nếu không cần hiển thị tất cả các tùy chọn cùng

lúc, bạn có thể sử dụng **Spinner** thay thế, tính năng này được mô tả trong một chương khác.

Trong nhiệm vụ này, bạn sẽ thêm một nhóm nút radio vào ứng dụng **DroidCafeInput** để thiết lập tùy chọn giao hàng cho đơn hàng tráng miệng. Để có cái nhìn tổng quan và xem thêm mã mẫu về nút radio, hãy xem **Radio Buttons**.

## 2.1 Thêm một RadioGroup và các nút radio

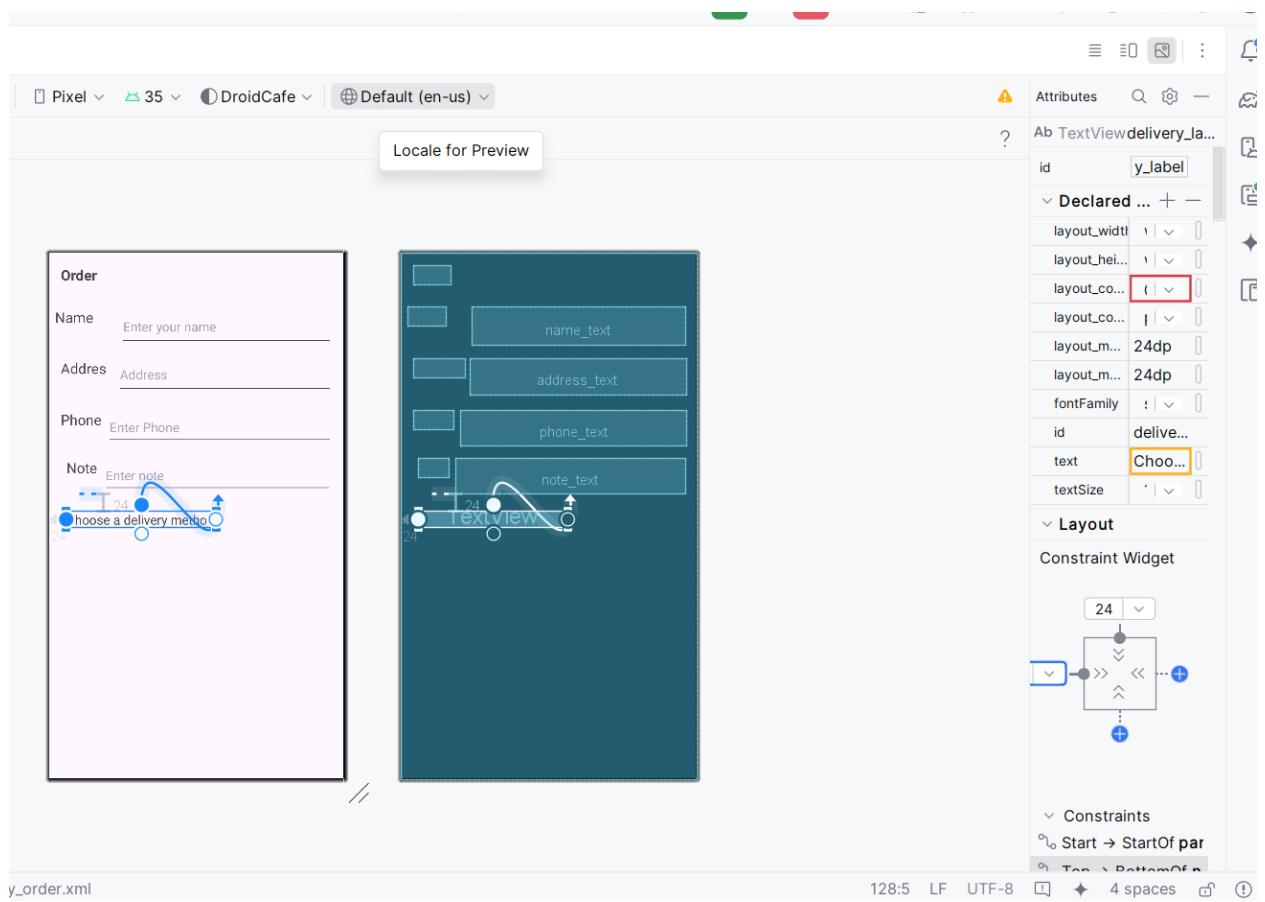
Để thêm các nút radio vào **OrderActivity** trong ứng dụng **DroidCafeInput**, bạn cần tạo các phần tử **RadioButton** trong tệp bố cục **activity\_order.xml**. Sau khi chỉnh sửa tệp bố cục, bố cục cho các nút radio trong **OrderActivity** sẽ trông giống như hình minh họa dưới đây.



Vì các lựa chọn nút radio là loại lựa chọn loại trừ lẫn nhau, bạn sẽ nhóm chúng lại với nhau trong một RadioGroup. Bằng cách nhóm chúng lại, hệ thống Android đảm bảo rằng chỉ một nút radio có thể được chọn cùng lúc.

Lưu ý: Thứ tự mà bạn liệt kê các phần tử RadioButton xác định thứ tự chúng xuất hiện trên màn hình.

1. Mở **activity\_order.xml** và thêm một phần tử **TextView** được ràng buộc ở dưới phần tử **note\_text** đã có trong layout, và ràng buộc với lề trái, như trong hình dưới đây.



2. Chuyển sang chỉnh sửa XML, và đảm bảo rằng bạn đã thiết lập các thuộc tính sau cho TextView mới:

<b>TextView attribute</b>	<b>Value</b>
android:id	"@+id/delivery_label"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	"24dp"
android:layout_marginLeft	"24dp"
android:layout_marginTop	"24dp"
android:text	"Choose a delivery method: "

3. Trích xuất tài nguyên chuỗi cho "Choose a delivery method:" thành choose\_delivery\_method.
4. Để thêm các nút radio, hãy bao chúng trong một RadioGroup. Thêm RadioGroup vào layout dưới TextView bạn vừa thêm, bao ba phần tử RadioButton như trong mã XML dưới đây:

```

<RadioButton
 android:id="@+id/sameday"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="onRadioButtonClicked"
 android:text="Same day messenger service" />

<RadioButton
 android:id="@+id/nextday"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="onRadioButtonClicked"
 android:text="Next day ground delivery" />

<RadioButton
 android:id="@+id/pickup"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:onClick="onRadioButtonClicked"
 android:text="Pick up" />

</RadioGroup>

```

Thuộc tính android:onClick với giá trị "onRadioButtonClicked" của mỗi RadioButton sẽ được gạch dưới bằng màu đỏ cho đến khi bạn thêm phương thức đó trong bước tiếp theo của tác vụ này.

5. Trích xuất ba tài nguyên chuỗi cho các thuộc tính android:text thành các tên sau để các chuỗi có thể dễ dàng được dịch: same\_day\_messenger\_service, next\_day\_ground\_delivery, và pick\_up.

## 2.2 Thêm phương thức xử lý sự kiện nhấn nút radio

Thuộc tính android:onClick cho mỗi phần tử nút radio chỉ định phương thức onRadioButtonClicked() để xử lý sự kiện nhấn nút. Vì vậy, bạn cần thêm một phương thức onRadioButtonClicked() mới trong lớp OrderActivity.

1. Mở tệp **activity\_order.xml** (nếu chưa mở) và tìm một giá trị onRadioButtonClicked cho thuộc tính android:onClick bị gạch dưới màu đỏ.
2. Nhấp vào giá trị onRadioButtonClicked, sau đó nhấp vào biểu tượng cảnh báo bóng đỏ ở lề trái.
3. Chọn **Create onRadioButtonClicked(View) in OrderActivity** trong menu bóng đỏ. Android Studio sẽ tạo phương thức onRadioButtonClicked(View view) trong OrderActivity.

```
public void onRadioButtonClicked(View view) {
}
```

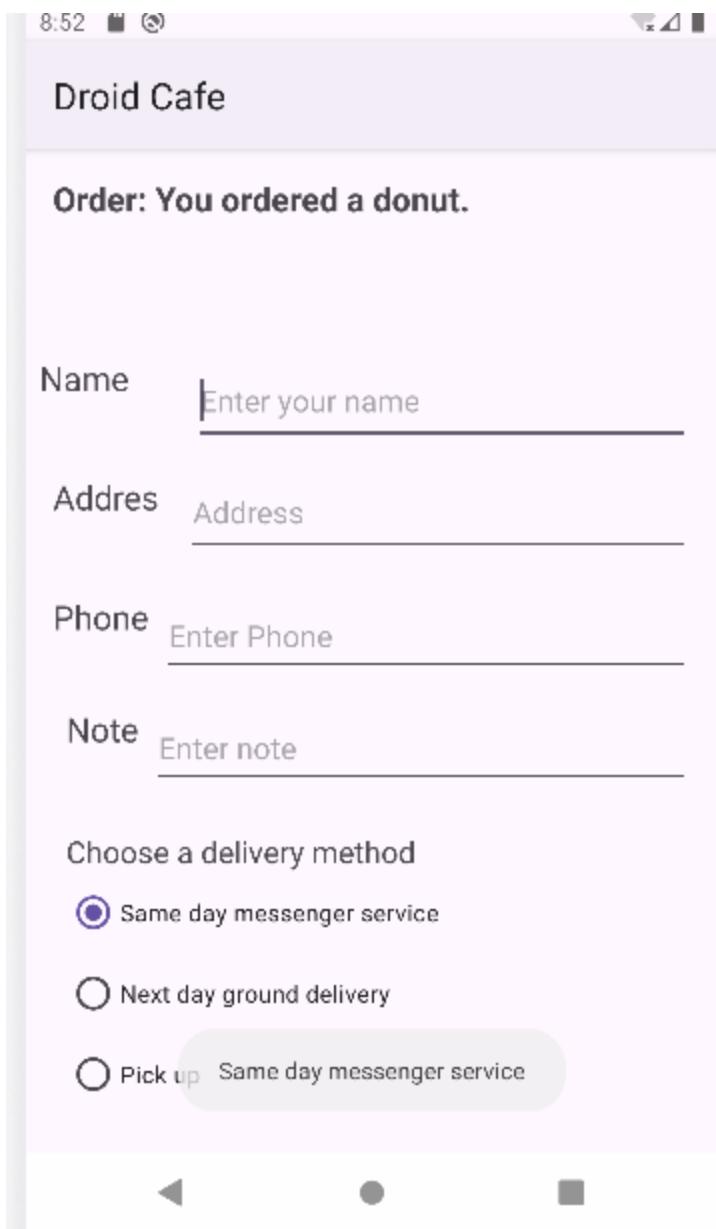
Ngoài ra, các giá trị onRadioButtonClicked cho các thuộc tính android:onClick khác trong activity\_order.xml sẽ được giải quyết và không còn bị gạch dưới nữa.

4. Để hiển thị nút radio nào đã được nhấn (tức là loại giao hàng mà người dùng chọn), hãy sử dụng một thông báo Toast. Mở **OrderActivity** và thêm phương thức showToast sau:

```
public void showToast(String message) {
 Toast.makeText(getApplicationContext(), message,
 Toast.LENGTH_SHORT).show();
}
```

5. Trong phương thức onRadioButtonClicked() mới, thêm một khối switch case để kiểm tra nút radio nào đã được chọn và gọi showToast() với thông điệp phù hợp. Mã sử dụng phương thức isChecked() của giao diện Checkable, trả về true nếu nút đã được chọn. Nó cũng sử dụng phương thức getId() của View để lấy định danh cho nút radio đã chọn.
6. Chạy ứng dụng. Nhấn vào một hình ảnh để xem hoạt động OrderActivity, hiển thị các lựa chọn giao hàng. Nhấn vào một lựa chọn giao hàng, và bạn sẽ

thấy một thông báo Toast ở dưới màn hình với lựa chọn đó, như trong hình dưới đây.



Task 2 mã giải pháp

Dự án Android Studio: [DroidCafeInput](#)

### Thử thách lập trình

**Lưu ý:** Tất cả các thử thách lập trình đều là tùy chọn và không phải là yêu cầu trước cho các bài học sau.

**Thử thách:** Các nút radio cho lựa chọn giao hàng trong ứng dụng DroidCafeInput ban đầu xuất hiện không được chọn, điều này ngụ ý rằng không có lựa chọn giao hàng mặc định. Thay đổi các nút radio sao cho một trong số chúng (chẳng hạn như nextday) được chọn mặc định khi các nút radio lần đầu tiên xuất hiện.

**Gợi ý:** Bạn có thể hoàn thành nhiệm vụ này hoàn toàn trong tệp layout. Một phương án thay thế là bạn có thể viết mã trong OrderActivity để chọn một trong các nút radio khi Activity xuất hiện lần đầu.

Mã giải pháp thử thách

Dự án Android Studio: [DroidCafeInput](#) (xem nút radio thứ hai trong tệp layout activity\_order.xml)

### **Task 3: Sử dụng Spinner cho các lựa chọn của người dùng**

Một Spinner cung cấp một cách nhanh chóng để chọn một giá trị từ một tập hợp các giá trị. Chạm vào Spinner sẽ hiển thị một danh sách thả xuống với tất cả các giá trị có sẵn, từ đó người dùng có thể chọn một giá trị. Nếu bạn chỉ cung cấp từ hai hoặc ba lựa chọn, bạn có thể muốn sử dụng nút radio cho các lựa chọn đó nếu có đủ chỗ trong bố cục; tuy nhiên, với hơn ba lựa chọn, Spinner hoạt động rất tốt, cuộn khi cần thiết để hiển thị các mục và chiếm ít không gian trong bố cục của bạn.

Mẹo: Để biết thêm thông tin về Spinner, hãy tham khảo Spinners.

Để cung cấp cách thức chọn nhãn cho số điện thoại (chẳng hạn như **Home**, **Work**, **Mobile**, **hoặc Other**), bạn có thể thêm một Spinner vào bố cục OrderActivity trong ứng dụng DroidCafe để nó xuất hiện ngay bên cạnh trường số điện thoại.

#### **3.1 Thêm một spinner vào bố cục**

Để thêm một Spinner vào bố cục OrderActivity trong ứng dụng DroidCafe, làm theo các bước dưới đây, được đánh số trong hình dưới:

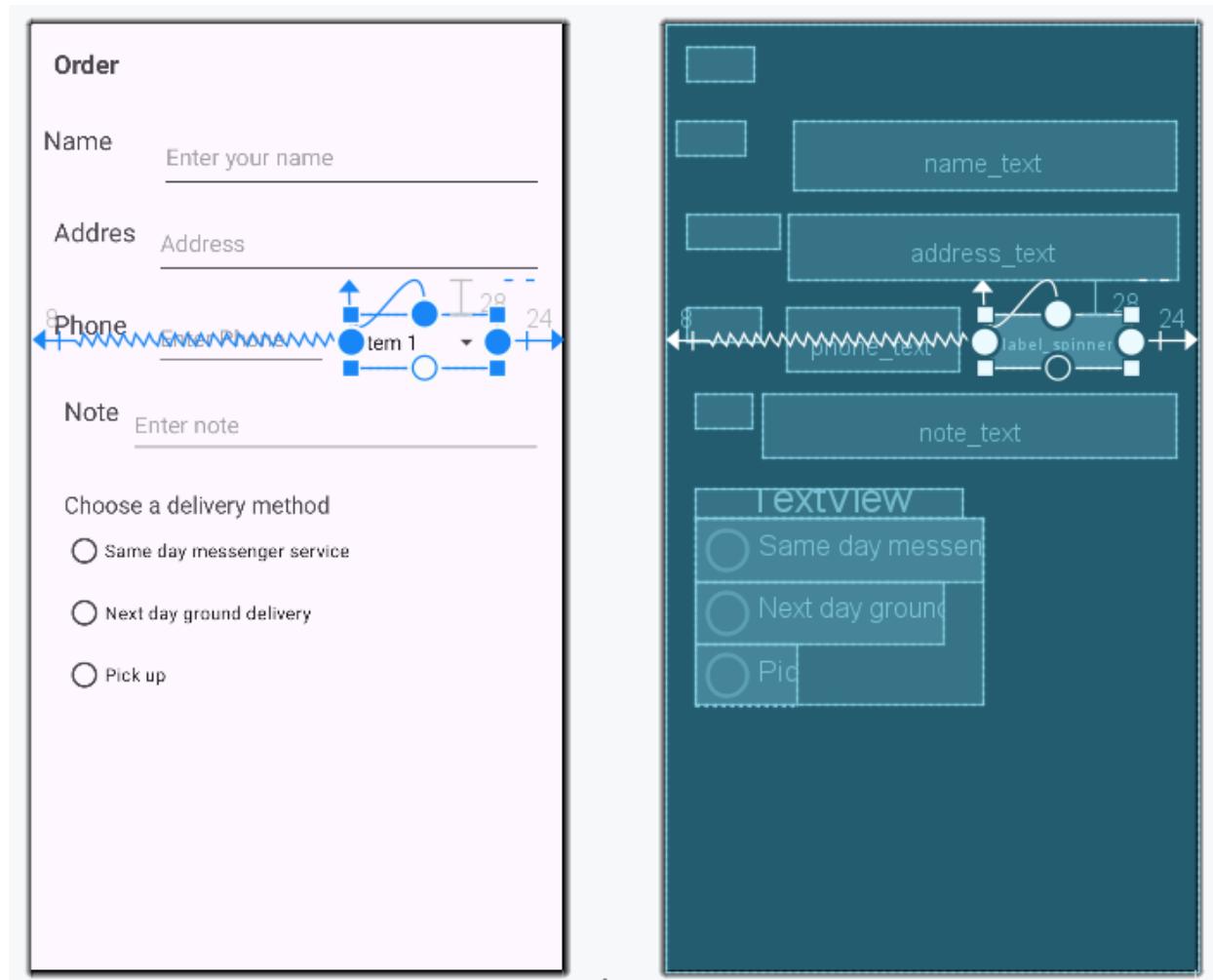
1. Mở **activity\_order.xml** và kéo **Spinner** từ bảng **Palette** vào bố cục.
2. Hạn chế phần trên của phần tử Spinner với dưới address\_text, phần bên phải với phần bên phải của bố cục, và phần bên trái với phone\_text.

Để căn chỉnh Spinner và phone\_text theo chiều ngang, sử dụng nút pack trong thanh công cụ, cung cấp các tùy chọn để đóng gói hoặc mở rộng các phần tử UI đã chọn.

Chọn cả Spinner và phone\_text trong **Component Tree**, nhấp vào nút pack, và chọn **Expand Horizontally**. Kết quả là, cả hai phần tử Spinner và phone\_text sẽ có chiều rộng cố định.

3. Trong bảng Attributes, thiết lập **ID** của Spinner là **label\_spinner**, và thiết lập các khoảng cách trên và bên phải là **24**, và khoảng cách bên trái là **8**. Chọn **match\_constraint** cho menu thả xuông **layout\_width**, và **wrap\_content** cho menu thả xuông **layout\_height**.

Bố cục sẽ trông giống như hình dưới đây. Menu thả xuông **layout\_width** của phần tử phone\_text trong bảng Attributes được thiết lập là 134dp. Bạn có thể thử nghiệm với các cài đặt chiều rộng khác nếu muốn.



Để xem mã XML của activity\_order.xml, nhấp vào tab **Text**.

```
<Spinner
 android:id="@+id/label_spinner"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginEnd="24dp" | ←
 android:layout_marginRight="24dp" | ←
 android:layout_marginStart="8dp" | ←
 android:layout_marginLeft="8dp" | ←
 android:layout_marginTop="24dp" | ←
 app:layout_constraintEnd_toEndOf="parent" | ←
 app:layout_constraintStart_toEndOf="@+id/phone_text" | ←
 app:layout_constraintTop_toBottomOf="@+id/address_text" />
```

Spinner phải có các thuộc tính sau: Hãy chắc chắn thêm các thuộc tính android:layout\_marginRight và android:layout\_marginLeft như trong đoạn mã trên để duy trì tính tương thích với các phiên bản Android cũ hơn.

Phần tử phone\_text hiện sẽ có các thuộc tính sau (sau khi sử dụng công cụ pack):

```
<EditText
 android:id="@+id/phone_text"
 android:layout_width="134dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="8dp"
 android:layout_marginEnd="16dp"
 android:hint="@string/enter_phone_hint"
 android:inputType="phone"
 android:minHeight="50dp"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="0.077"
 app:layout_constraintStart_toEndOf="@+id/phone_label"
 app:layout_constraintTop_toTopOf="@+id/phone_label" />
```

### 3.2 Thêm mã để kích hoạt Spinner và bộ lắng nghe của nó

Các lựa chọn cho Spinner là các chuỗi tĩnh được xác định rõ, chẳng hạn như "Home" và "Work", vì vậy bạn có thể sử dụng một mảng văn bản được định nghĩa trong **strings.xml** để lưu trữ các giá trị đó.

Để kích hoạt Spinner và trình nghe sự kiện của nó, hãy triển khai giao diện AdapterView.OnItemSelectedListener, giao diện này yêu cầu bạn cũng phải thêm các phương thức gọi lại `onItemSelected()` và `onNothingSelected()`.

1. Mở **strings.xml** và định nghĩa các giá trị có thể chọn (**Home**, **Work**, **Mobile** và **Other**) cho Spinner dưới dạng một mảng chuỗi `labels_array`.

```
<string-array name="labels_array">
 <item>Home</item>
 <item>Word</item>
 <item>Mobile</item>
 <item>Other</item>
</string-array>
```

- Để định nghĩa callback xử lý lựa chọn cho Spinner, hãy thay đổi lớp **OrderActivity** để triển khai giao diện **AdapterView.OnItemSelectedListener**, như được minh họa:

```
public class OrderActivity extends AppCompatActivity implements AdapterView.OnItemSelectedListener {
```

Khi bạn nhập **AdapterView**. trong câu lệnh trên, Android Studio sẽ tự động nhập tiện ích AdapterView. Lý do bạn cần AdapterView là vì bạn cần một adapter—cụ thể là ArrayAdapter—để gán mảng vào Spinner. Một adapter giúp kết nối dữ liệu của bạn—trong trường hợp này là mảng các mục của Spinner—with Spinner. Bạn sẽ tìm hiểu thêm về mô hình sử dụng adapter để kết nối dữ liệu trong một bài thực hành khác. Dòng này sẽ xuất hiện trong khôi import của bạn:

```
import android.widget.AdapterView;
```

Khi nhập **OnItemSelectedListener** trong câu lệnh trên, hãy đợi vài giây để một biểu tượng bóng đèn màu đỏ xuất hiện ở lề bên trái.

- Nhập vào biểu tượng bóng đèn và chọn **Implement methods**. Các phương thức `onItemSelected()` và `onNothingSelected()`, vốn là bắt buộc đối với **OnItemSelectedListener**, sẽ được làm nổi bật, và tùy chọn `Insert @Override` sẽ được chọn sẵn. Nhập vào **OK**.

Bước này sẽ tự động thêm các phương thức callback `onItemSelected()` và `onNothingSelected()` trống vào cuối lớp **OrderActivity**. Cả hai phương thức này đều sử dụng tham số `AdapterView<?>**`. Dấu `**<?>` là một ký hiệu đại diện (wildcard) của Java, giúp phương thức có thể linh hoạt chấp nhận bất kỳ loại AdapterView nào làm đối số.

- Khởi tạo một Spinner trong phương thức `onCreate()` bằng cách sử dụng phần tử `label_spinner` trong layout và đặt trình nghe sự kiện của nó bằng `spinner.setOnItemSelectedListener` trong `onCreate()`, như trong đoạn mã sau:

```
Spinner spinner = findViewById(R.id.label_spinner);
if (spinner != null) {
 spinner.setOnItemSelectedListener(this);
}
```

5. Tiếp tục chỉnh sửa phương thức onCreate(), thêm một câu lệnh để tạo ArrayAdapter với mảng chuỗi (labels\_array) bằng cách sử dụng layout Spinner được cung cấp sẵn bởi Android cho từng mục (layout.simple\_spinner\_item):

```
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(context: this,
 R.array.labels_array, android.R.layout.simple_spinner_item);
```

Layout simple\_spinner\_item được sử dụng trong bước này và layout simple\_spinner\_dropdown\_item được sử dụng trong bước tiếp theo là các layout mặc định do Android cung cấp trong lớp R.layout. Bạn nên sử dụng các layout này trừ khi bạn muốn tự định nghĩa layout riêng cho các mục trong Spinner và giao diện của nó.

6. Xác định layout cho các lựa chọn của Spinner là simple\_spinner\_dropdown\_item, sau đó áp dụng adapter vào Spinner.

```
adapter.setDropDownViewResource
 (android.R.layout.simple_spinner_dropdown_item);
// Apply the adapter to the spinner.
if (spinner != null) {
 spinner.setAdapter(adapter);
}
```

### 3.3 Thêm một mã để phản hồi khi chọn mục trong Spinner

Khi người dùng chọn một mục trong Spinner, Spinner sẽ nhận được sự kiện onItemSelected.

Để xử lý sự kiện này, bạn đã triển khai giao diện AdapterView.OnItemSelectedListener ở bước trước, đồng thời thêm các phương thức callback onItemSelected() và onNothingSelected() trống.

Trong bước này, bạn sẽ điền mã vào phương thức onItemSelected() để lấy mục đã chọn trong Spinner bằng cách sử dụng getItemAtPosition(), sau đó gán mục đó vào biến spinnerLabel.

1.Thêm mã vào phương thức callback onItemSelected() trống, như minh họa bên dưới, để lấy mục đã chọn của người dùng bằng getItemAtPosition() và gán

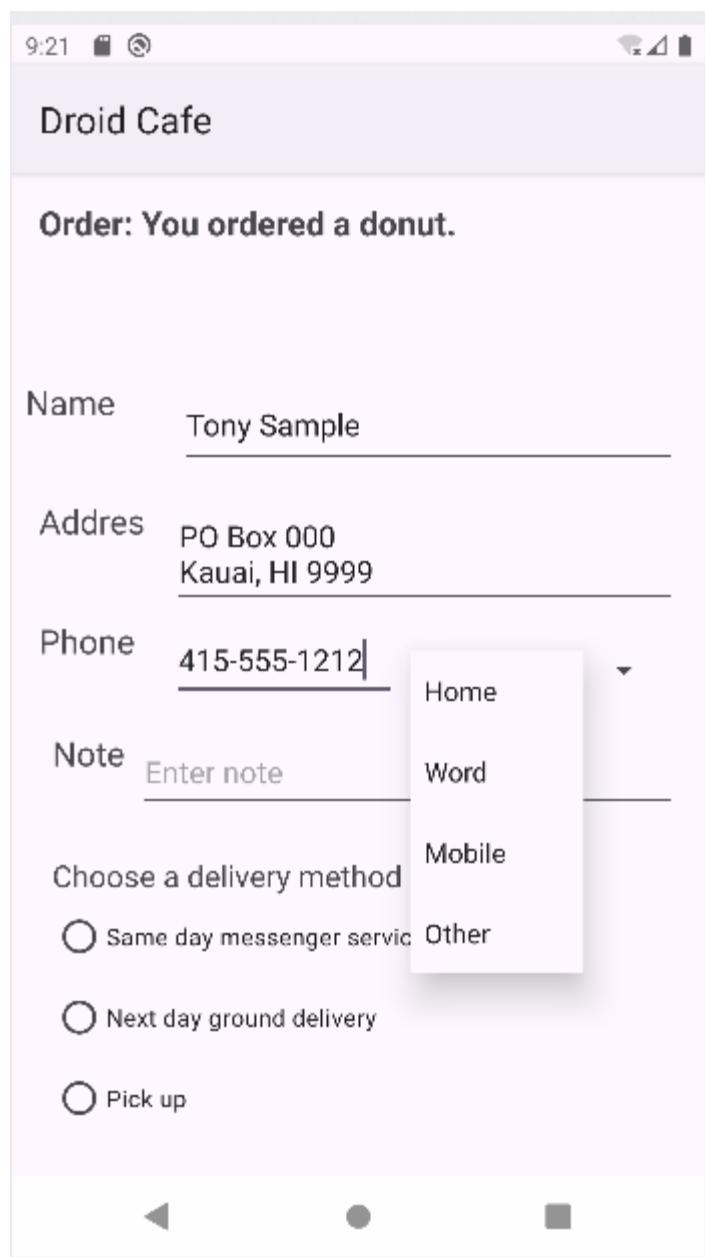
nó vào spinnerLabel. Bạn cũng có thể gọi phương thức displayToast() mà bạn đã thêm vào OrderActivity:

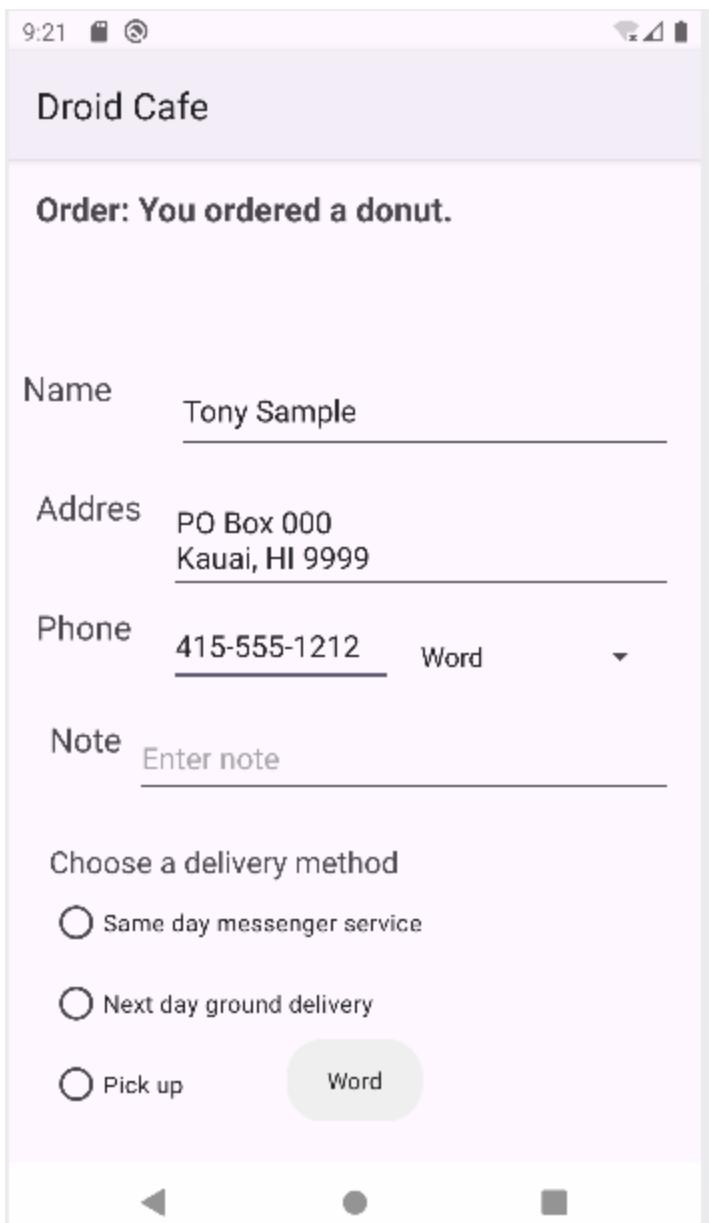
```
public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
 spinnerLabel = adapterView.getItemAtPosition(i).toString();
 displayToast(spinnerLabel);
}
```

Không cần thêm mã vào phương thức callback onNothingSelected() trong ví dụ này.

## 2.Chạy ứng dụng.

Spinner sẽ xuất hiện bên cạnh trường nhập số điện thoại và hiển thị lựa chọn đầu tiên (**Home**). Khi nhấn vào Spinner, tất cả các lựa chọn sẽ xuất hiện, như trong hình bên trái. Khi chọn một mục trong Spinner, một thông báo Toast sẽ hiển thị với lựa chọn đó, như trong hình bên phải.





### Task 3 Mã giải pháp

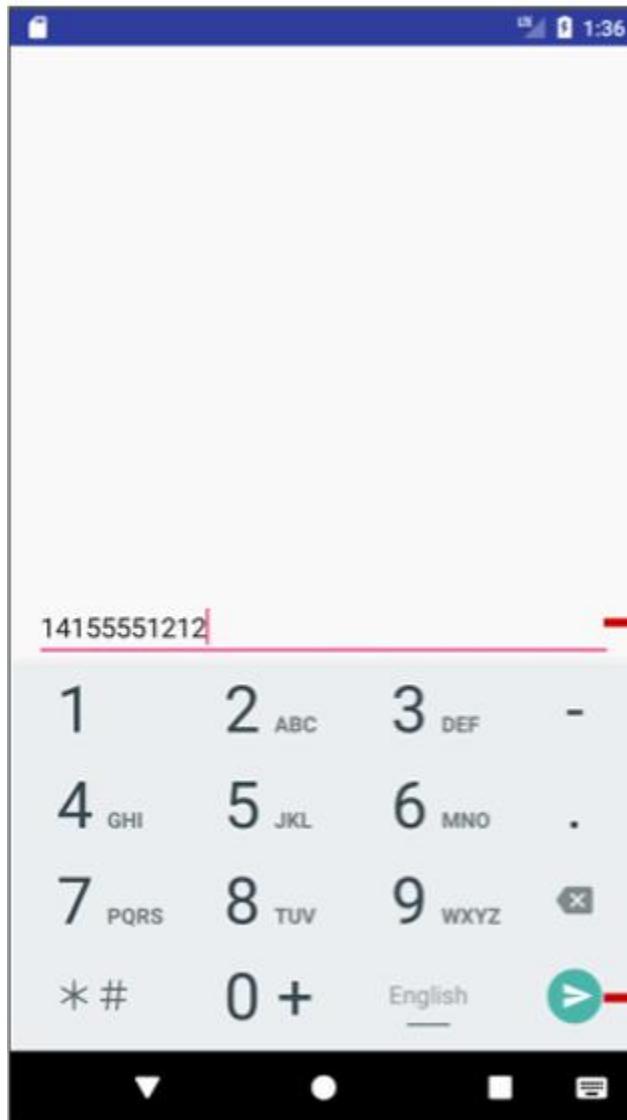
Dự án Android Studio: [DroidCafeInput](#)

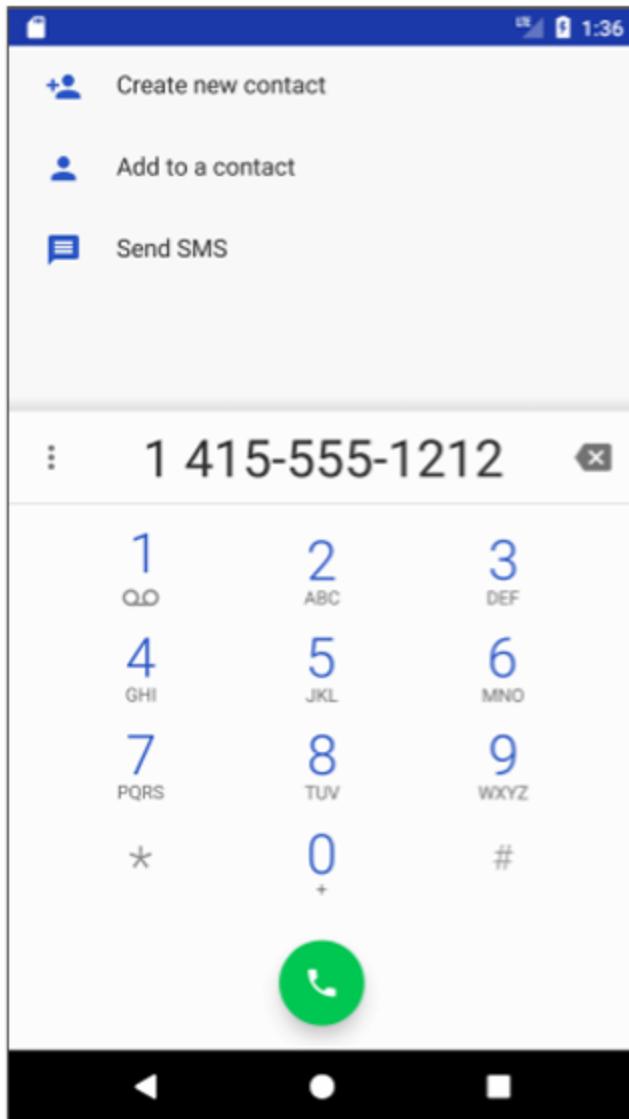
### Thử thách lập trình 2

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

**Thử thách:** Viết mã để thực hiện một hành động trực tiếp từ bàn phím bằng cách nhấn phím **Send**, chẳng hạn như để quay số điện thoại:

Trong hình trên:





1.Nhập số điện thoại vào trường EditText.

2.Nhấn phím **Send** để mở trình quay số điện thoại. Trình quay số sẽ xuất hiện ở phía bên phải của hình minh họa.

Hướng dẫn thực hiện thử thách, tạo một dự án ứng dụng mới, thêm một EditText và thiết lập thuộc tính android:inputType thành phone. Sử dụng thuộc tính android:imeOptions cho phần tử EditText với giá trị actionSend:

```
android:imeOptions="actionSend"
```

Người dùng có thể nhấn phím **Send** để quay số điện thoại, như trong hình minh họa.

Trong phương thức onCreate() của Activity, sử dụng setOnEditorActionListener() để thiết lập trình nghe sự kiện cho EditText nhằm phát hiện khi phím được nhấn:

```
EditText editText = findViewById(R.id.editText_main);
if (editText != null)
 editText.setOnEditorActionListener
 (new TextView.OnEditorActionListener() {
 // If view is found, set the listener for editText.
});
```

Để biết cách thiết lập trình nghe sự kiện, hãy xem phần Chỉ định loại phương thức nhập (Specify the input method type).

Bước tiếp theo là ghi đè phương thức onEditorAction() và sử dụng hằng số IME\_ACTION\_SEND trong lớp EditorInfo để phản hồi khi phím được nhấn. Trong ví dụ bên dưới, phím này được sử dụng để gọi phương thức dialNumber() nhằm quay số điện thoại:

```
@Override
public boolean onEditorAction(TextView textView, int actionId, KeyEvent keyEvent) {
 boolean handled = false;

 if (actionId == EditorInfo.IME_ACTION_SEND) {
 dialNumber();
 handled = true;
 }
 return handled;
}
```

Cuối cùng, tạo phương thức dialNumber(), sử dụng implicit intent với ACTION\_DIAL để chuyển số điện thoại sang một ứng dụng khác có thể quay số. Nó sẽ trông như sau:

---

```
private void dialNumber() {
 // Find the editText_main view.
 EditText editText = findViewById(R.id.editText_main);
 String phoneNum = null;
 // If the editText field is not null,
 // concatenate "tel: " with the phone number string.
 if (editText != null) phoneNum = "tel:" +
 editText.getText().toString();
 // Optional: Log the concatenated phone number for dialing.
 Log.d(TAG, "dialNumber: " + phoneNum);
 // Specify the intent.
 Intent intent = new Intent(Intent.ACTION_DIAL);
 // Set the data for the intent as the phone number.
 intent.setData(Uri.parse(phoneNum));
 // If the intent resolves to a package (app),
 // start the activity with the intent.
 if (intent.resolveActivity(getApplicationContext()) != null) {
 startActivity(intent);
 } else {
 Log.d("ImplicitIntents", "Can't handle this!");
 }
}
```

## Thử thách 2 mã giải pháp

Dự án Android Studio: [KeyboarDialPhone](#)

### Tóm tắt

Các giá trị thuộc tính android:inputType ảnh hưởng đến giao diện bàn phím trên màn hình:

- **textAutoCorrect:** Gợi ý sửa lỗi chính tả.
- **textCapSentences:** Viết hoa chữ cái đầu tiên của mỗi câu mới.
- **textPersonName:** Hiển thị một dòng văn bản với gợi ý khi nhập và nút **Done** để hoàn tất.
- **textMultiLine:** Cho phép nhập nhiều dòng văn bản và có phím Return để xuống dòng.

- **textPassword:** Ẩn mật khẩu khi nhập.
- **textEmailAddress:** Hiển thị bàn phím nhập email thay vì bàn phím thông thường.
- **phone:** Hiển thị bàn phím số thay vì bàn phím thông thường.

Bạn thiết lập giá trị cho thuộc tính android:inputType trong tệp layout XML cho phần tử EditText.

Để kết hợp nhiều giá trị, hãy sử dụng ký tự gạch đứng ( | ).

RadioButton là điều khiển đầu vào hữu ích để chọn một tùy chọn duy nhất trong một tập hợp tùy chọn:

- Nhóm các phần tử **RadioButton** bên trong **RadioGroup** để đảm bảo chỉ một **RadioButton** được chọn cùng một lúc.
- Thứ tự liệt kê các RadioButton trong nhóm xác định thứ tự hiển thị trên màn hình.
- Sử dụng thuộc tính android:onClick cho từng RadioButton để chỉ định trình xử lý sự kiện khi nhấn.
- Để kiểm tra xem một nút có được chọn không, sử dụng phương thức **isChecked()** của giao diện **Checkable**, trả về true nếu nút được chọn.

Một **Spinner** (Menu thả xuống)

- Thêm **Spinner** vào layout.
- Sử dụng **ArrayAdapter** để gán một mảng văn bản làm các mục trong menu Spinner.
- Triển khai giao diện **AdapterView.OnItemSelectedListener**, trong đó yêu cầu thêm các phương thức callback **onItemSelected()** và **onNothingSelected()** để kích hoạt **Spinner** và trình nghe sự kiện của nó.
- Sử dụng phương thức **onItemSelected()** để lấy mục được chọn trong Spinner bằng **getItemAtPosition()**.

## **Khái niệm liên quan**

Tài liệu về khái niệm liên quan có trong **4.2: Input controls**.

## **Tìm hiểu thêm**

Tài liệu hướng dẫn Android Studio:

- [Android Studio User Guide](#)

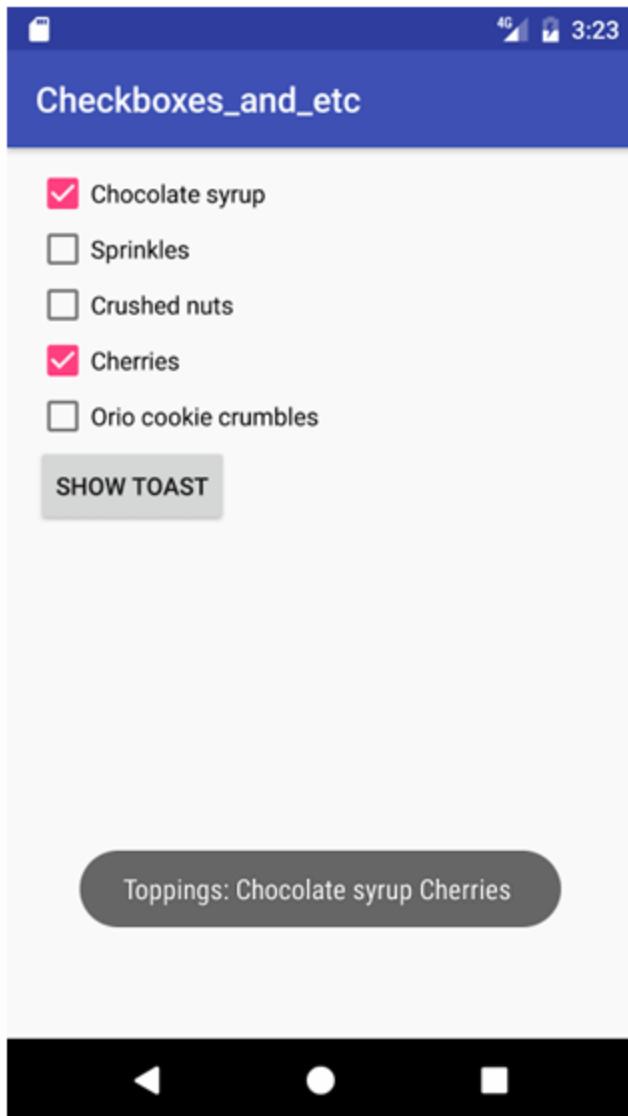
Tài liệu dành cho nhà phát triển Android:

- [Input events overview](#)
- [Specify the input method type](#)
- [Styles and themes](#)
- [Radio Buttons](#)
- [Spinners](#)
- [View](#)
- [Button](#)
- [Edittext](#)
- [Android:inputType](#)
- [TextView](#)
- [RadioGroup](#)
- [Checkbox](#)
- [SeekBar](#)
- [ToggleButton](#)
- [Spinner](#)

Bài tập về nhà

### Xây dựng và chạy một ứng dụng

1. Tạo một ứng dụng có năm Checkbox và một nút **Show Toast**, như hình minh họa.
2. Nếu người dùng chọn một Checkbox và nhấn **Show Toast**, hiển thị một thông báo **Toast** hiển thị nội dung của Checkbox đã chọn.
3. Nếu người dùng chọn nhiều hơn một Checkbox và nhấn **Show Toast**, hiển thị một Toast bao gồm nội dung của tất cả các Checkbox được chọn, như trong hình minh họa.



## Trả lời các câu hỏi

Câu hỏi 1

Sự khác biệt quan trọng nhất giữa checkbox và một nhóm radio button (RadioGroup)?

Chọn một:

- "Sự khác biệt chính là checkbox cho phép chọn nhiều mục, trong khi RadioGroup chỉ cho phép chọn một mục."

Câu hỏi 2

Nhóm layout nào cho phép căn chỉnh các phần tử CheckBox theo chiều dọc?

Chọn một:

- LinearLayout

### Câu hỏi 3

Phương thức nào của giao diện Checkable dùng để kiểm tra trạng thái của radio button (tức là kiểm tra xem nó đã được chọn hay chưa)?

Chọn một:

- isChecked()

### Hướng dẫn chấm điểm ứng dụng

#### Kiểm tra xem ứng dụng có các tính năng sau không:

- Bộ cục chứa năm CheckBox được căn chỉnh theo chiều dọc trên màn hình, cùng với một nút Show Toast.
- Phương thức onSubmit() xác định checkbox nào được chọn bằng cách sử dụng findViewById() kết hợp với isChecked().
- Các chuỗi mô tả topping được nối lại thành một thông báo Toast.

### Bài 4.3 : Menu và bộ chọn

#### Giới thiệu

Thanh ứng dụng (App Bar), còn được gọi là thanh hành động (Action Bar), là một khung gian chuyên dụng nằm ở phía trên cùng của mỗi màn hình **Activity**. Khi bạn tạo một Activity từ mẫu Basic Activity, Android Studio sẽ tự động bao gồm một thanh ứng dụng.

Menu tùy chọn (Options Menu) trong thanh ứng dụng thường cung cấp các lựa chọn điều hướng, chẳng hạn như chuyển đến một Activity khác trong ứng dụng. Menu cũng có thể cung cấp các tùy chọn ảnh hưởng đến cách sử dụng ứng dụng, ví dụ như thay đổi cài đặt hoặc thông tin hồ sơ, thường được thực hiện trong một Activity riêng biệt.

Trong bài thực hành này, bạn sẽ tìm hiểu về cách thiết lập App Bar và Options Menu trong ứng dụng của mình, như minh họa trong hình dưới đây.

Trong hình trên:



1. Thanh ứng dụng (App bar): Thanh ứng dụng bao gồm tiêu đề ứng dụng, menu tùy chọn và nút tràn (overflow button).
2. Biểu tượng hành động của menu tùy chọn(Options menu action icons): Hai mục menu tùy chọn đầu tiên xuất hiện dưới dạng biểu tượng trong thanh ứng dụng.
3. Nút tràn (Overflow button): Nút tràn (ba dấu chấm dọc) mở một menu hiển thị các mục menu tùy chọn bổ sung.
4. Menu tràn của các mục tùy chọn(Options overflow menu): Sau khi nhấp vào nút tràn, các mục menu tùy chọn bổ sung sẽ xuất hiện trong menu tràn.

Các mục menu tùy chọn xuất hiện trong menu tràn (xem hình trên). Tuy nhiên, bạn có thể đặt một số mục dưới dạng biểu tượng — càng nhiều càng tốt — trong thanh ứng dụng. Việc sử dụng thanh ứng dụng cho menu tùy chọn giúp ứng dụng của bạn đồng nhất với các ứng dụng Android khác, cho phép người dùng dễ dàng hiểu cách sử dụng ứng dụng và có trải nghiệm tuyệt vời.

**Mẹo:** Để cung cấp trải nghiệm người dùng quen thuộc và đồng nhất, hãy sử dụng Menu APIs để trình bày các hành động của người dùng và các tùy chọn khác trong các activity của bạn. Xem [Menus](#) để biết chi tiết.

Bạn cũng sẽ tạo một ứng dụng hiển thị một dialog yêu cầu người dùng chọn lựa, ví dụ như một thông báo yêu cầu người dùng nhấn **OK** hoặc **Cancel**. Một dialog là một cửa sổ xuất hiện trên màn hình hoặc chiếm toàn bộ màn hình, làm gián đoạn dòng chảy hoạt động. Android cung cấp các dialog sẵn có, gọi là pickers, để chọn thời gian hoặc ngày. Bạn có thể sử dụng chúng để đảm bảo người dùng chọn đúng thời gian hoặc ngày được định dạng chính xác và điều

chỉnh theo giờ và ngày địa phương của người dùng. Trong bài học này, bạn cũng sẽ tạo một ứng dụng với date picker.

## Những gì bạn nên biết trước

Bạn nên có khả năng:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các phần tử giao diện người dùng (UI) bằng trình chỉnh sửa layout.
- Chỉnh sửa mã layout XML và truy cập các phần tử từ mã Java của bạn.
- Thêm trình xử lý sự kiện click cho Button.

## Những gì bạn sẽ học

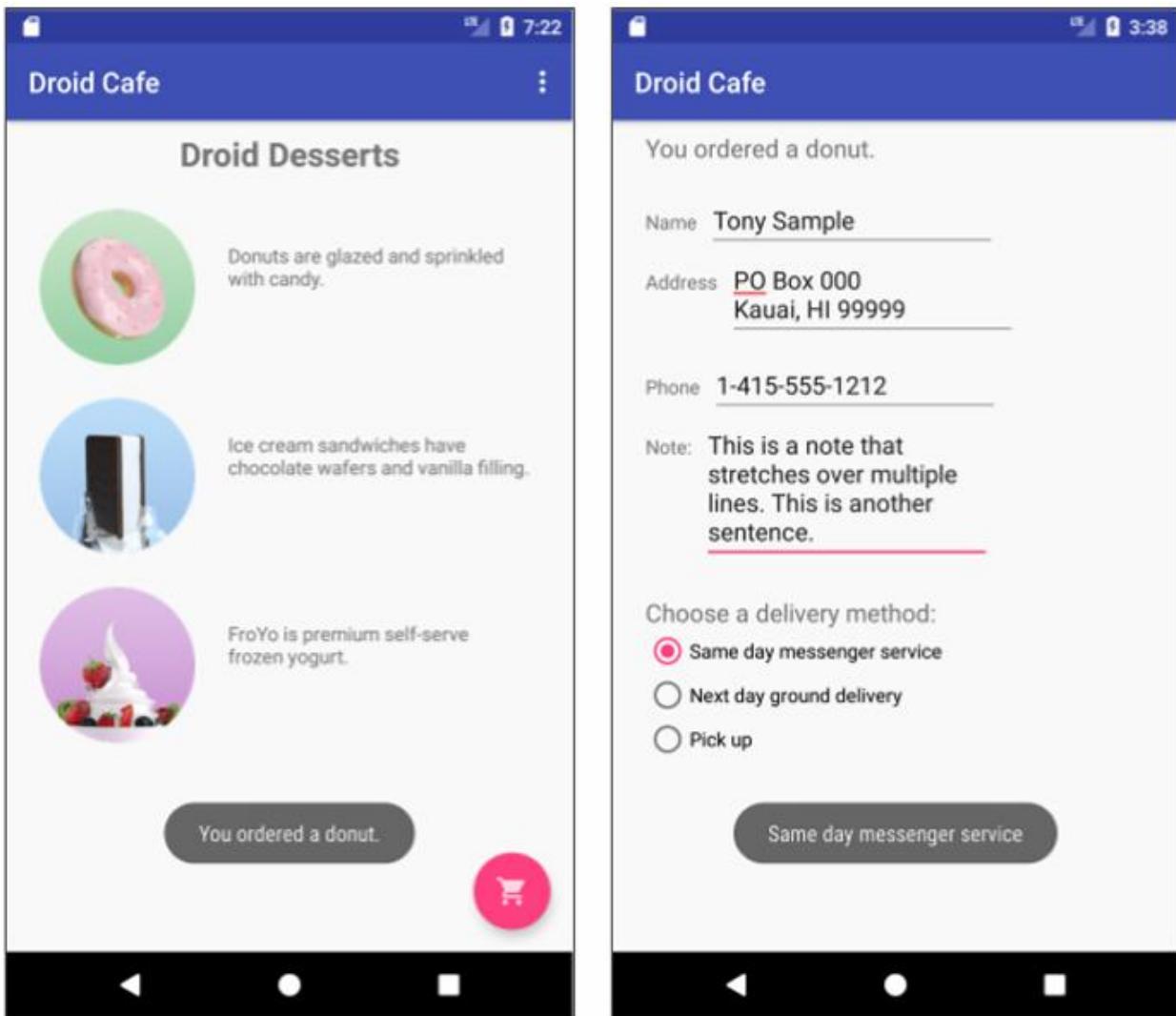
- Cách thêm các mục menu vào menu tùy chọn.
- Cách thêm biểu tượng cho các mục trong menu tùy chọn.
- Cách hiển thị các mục menu trong thanh ứng dụng.
- Cách thêm trình xử lý sự kiện cho các mục menu.
- Cách thêm một dialog để hiển thị thông báo.
- Cách thêm date picker.

## Những gì bạn sẽ làm

- Tiếp tục thêm các tính năng cho dự án Droid Cafe từ bài thực hành trước.
- Thêm các mục menu vào menu tùy chọn.
- Thêm biểu tượng cho các mục menu để xuất hiện trong thanh ứng dụng.
- Kết nối các sự kiện click vào các trình xử lý sự kiện để xử lý sự kiện nhấn.
- Sử dụng alert dialog để yêu cầu người dùng chọn lựa.
- Sử dụng date picker để nhập ngày.

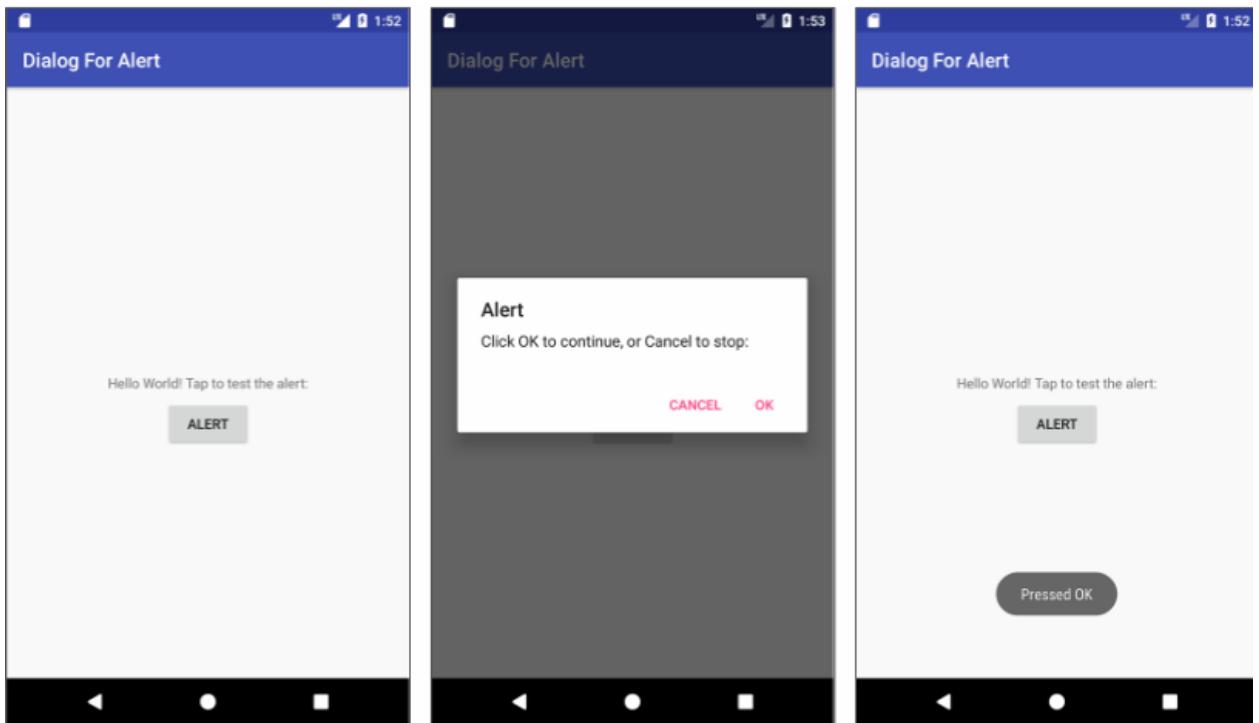
## Tổng quan về ứng dụng

Trong bài thực hành trước, bạn đã tạo một ứng dụng có tên Droid Cafe, như hình dưới đây, sử dụng mẫu Basic Activity. Mẫu này cũng cung cấp một menu tùy chọn cơ bản trong thanh ứng dụng ở phía trên màn hình.

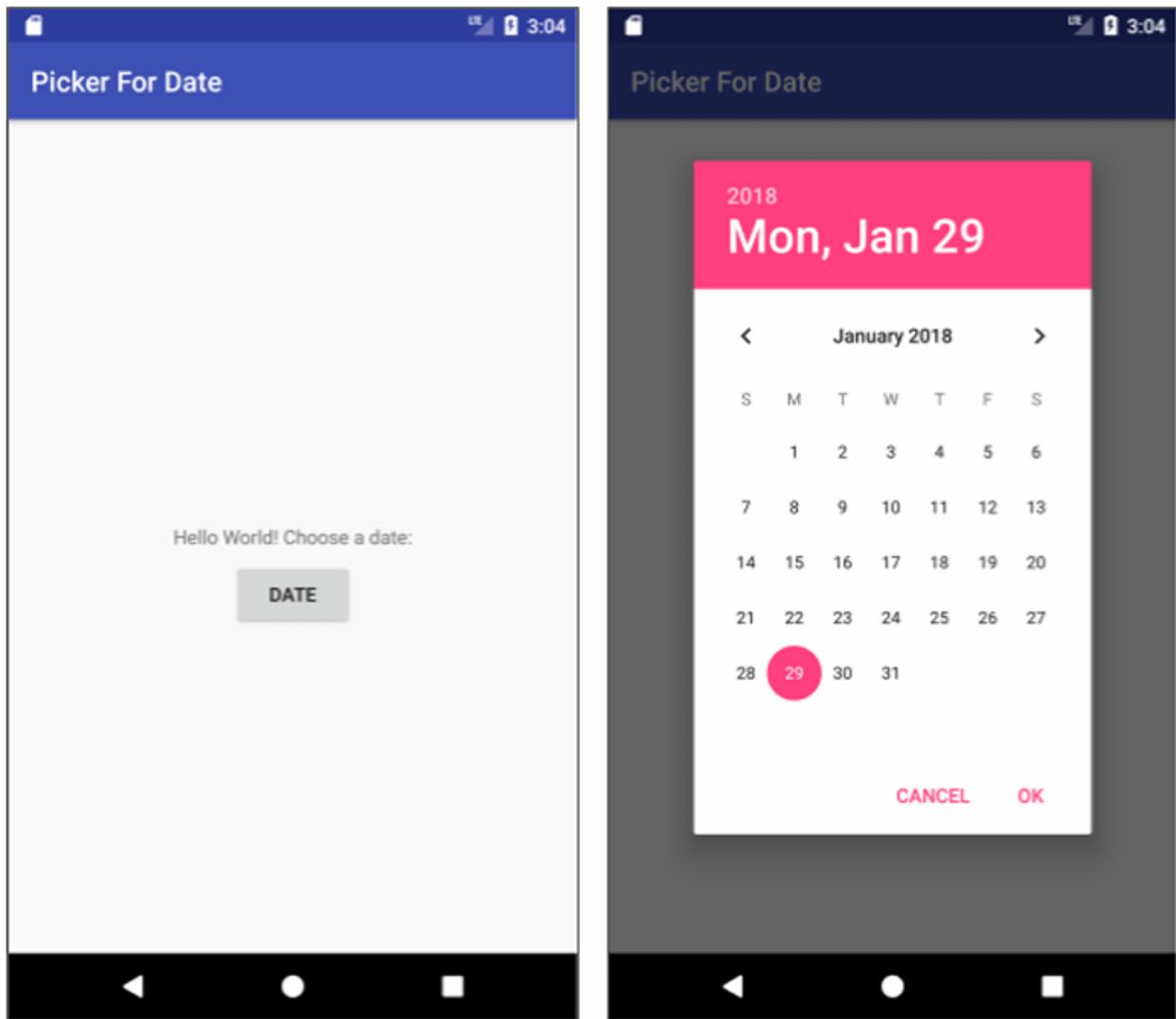


Trong bài tập này, bạn sẽ sử dụng thư viện hỗ trợ `v7 appcompat` với `Toolbar` như một thanh ứng dụng, hoạt động trên nhiều thiết bị và cũng cung cấp cho bạn không gian để tùy chỉnh thanh ứng dụng sau này khi ứng dụng của bạn phát triển. Để đọc thêm về các cân nhắc thiết kế khi sử dụng thanh ứng dụng, hãy xem [Responsive layout grid](#) trong thông số kỹ thuật Material Design.

Bạn sẽ tạo một ứng dụng mới hiển thị alert dialog. Dialog này gián đoạn quy trình làm việc của người dùng và yêu cầu người dùng đưa ra lựa chọn.



Bạn cũng sẽ tạo một ứng dụng cung cấp một Button để hiển thị date picker, và chuyển ngày đã chọn thành một chuỗi để hiển thị trong thông báo Toast.



### Task 1: Thêm các mục vào menu tùy chọn

Trong nhiệm vụ này, bạn sẽ mở dự án DroidCafeInput từ bài thực hành trước và thêm các mục vào menu tùy chọn trong thanh ứng dụng ở phía trên màn hình.

#### 1.1 Kiểm tra mã nguồn

Mở ứng dụng DroidCafeInput từ bài thực hành về việc sử dụng các điều khiển đầu vào và kiểm tra các tệp layout sau trong thư mục **res > layout**:

- **activity\_main.xml**: Layout chính cho MainActivity, màn hình đầu tiên mà người dùng thấy.
- **content\_main.xml**: Layout cho nội dung của màn hình MainActivity, được bao gồm trong **activity\_main.xml**.

- **activity\_order.xml**: Layout cho OrderActivity, được thêm vào trong bài thực hành về việc sử dụng các điều khiển đầu vào.

### Các bước thực hiện:

1. Mở **content\_main.xml** và nhấp vào tab **Text** để xem mã XML. Thuộc tính `app:layout_behavior` của ConstraintLayout được thiết lập là `@string/appbar_scrolling_view_behavior`, điều khiển cách màn hình cuộn liên quan đến thanh ứng dụng ở phía trên. (Chuỗi tài nguyên này được định nghĩa trong tệp values.xml đã được tạo, và bạn không nên chỉnh sửa nó.)

Để biết thêm về hành vi cuộn, xem [Android Design Support Library](#) trong blog của Android Developers. Để biết về các thực hành thiết kế có liên quan đến menu cuộn, xem [Scrolling in the Material Design specification](#).

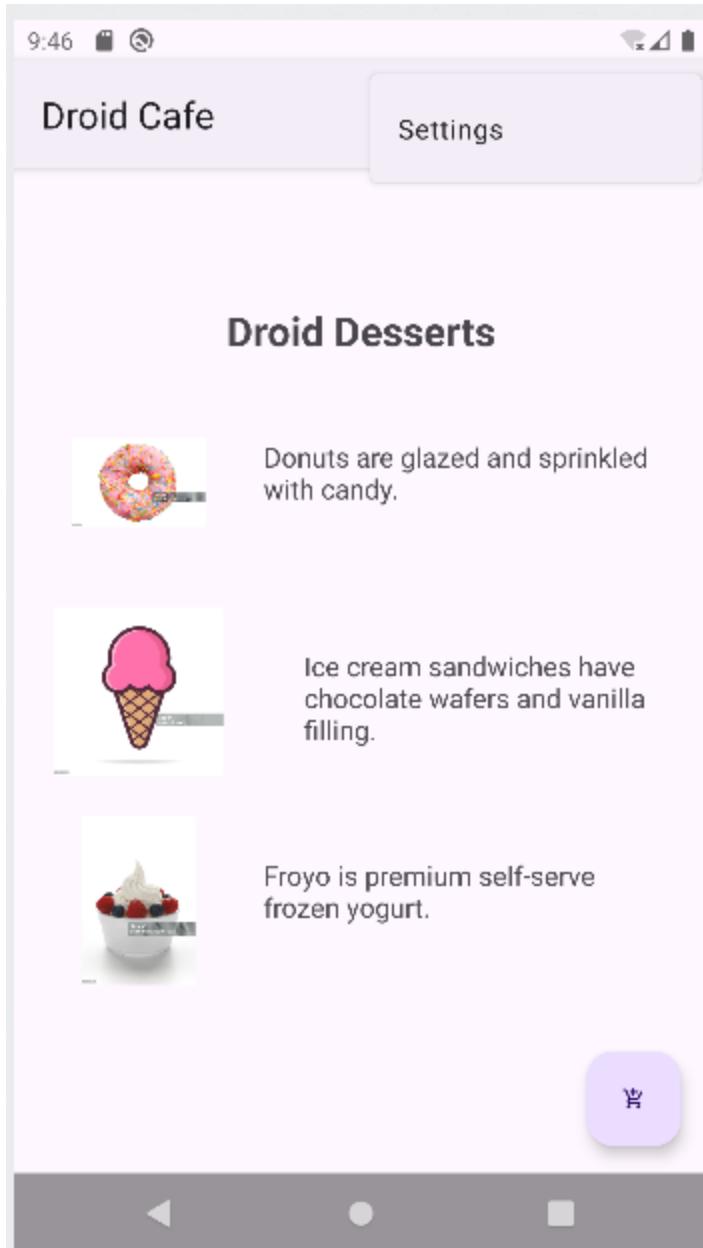
2. Mở **activity\_main.xml** và nhấp vào tab **Text** để xem mã XML cho layout chính, sử dụng layout **CoordinatorLayout** với layout **AppBarLayout** nhúng bên trong. Các thẻ CoordinatorLayout và AppBarLayout yêu cầu tên đầy đủ xác định android.support.design, là thư viện Hỗ trợ Thiết kế của Android. AppBarLayout giống như một LinearLayout theo chiều dọc. Nó sử dụng lớp Toolbar trong thư viện hỗ trợ, thay vì ActionBar gốc, để triển khai thanh ứng dụng. Toolbar trong layout này có id là toolbar, và cũng được chỉ định, giống như AppBarLayout, với tên đầy đủ (android.support.v7.widget).

Thanh ứng dụng là một phần của màn hình hiển thị, có thể hiển thị tiêu đề activity, điều hướng và các mục tương tác khác. ActionBar gốc hoạt động khác nhau tùy thuộc vào phiên bản Android chạy trên thiết bị. Vì lý do này, nếu bạn đang thêm menu tùy chọn, bạn nên sử dụng thư viện hỗ trợ [v7 appcompat](#) với [Toolbar](#) làm thanh ứng dụng. Việc sử dụng [Toolbar](#) giúp bạn dễ dàng thiết lập một thanh ứng dụng hoạt động trên nhiều thiết bị và cũng cung cấp không gian để tùy chỉnh thanh ứng dụng của bạn sau này khi ứng dụng phát triển. Toolbar bao gồm các tính năng mới nhất và hoạt động cho bất kỳ thiết bị nào có thể sử dụng thư viện hỗ trợ.

Layout **activity\_main.xml** cũng sử dụng câu lệnh layout include để bao gồm toàn bộ layout được định nghĩa trong **content\_main.xml**. Việc tách biệt các định nghĩa layout giúp bạn dễ dàng thay đổi nội dung của layout riêng biệt với định nghĩa thanh công cụ và layout điều phối.

3. Chạy ứng dụng. Chú ý đến thanh ở phía trên màn hình hiển thị tên ứng dụng (Droid Cafe). Nó cũng hiển thị nút tràn hành động (ba dấu chấm dọc) ở phía

bên phải. Nhấp vào nút tràn để xem menu tùy chọn, lúc này chỉ có một mục menu là **Settings**.



4. Kiểm tra tệp **AndroidManifest.xml**. Activity **MainActivity** được thiết lập sử dụng theme **NoActionBar**. Theme này được định nghĩa trong tệp **styles.xml** (mở **app > res > values > styles.xml** để xem). Các style được trình bày trong một bài học khác, nhưng bạn có thể thấy rằng theme **NoActionBar** thiết lập thuộc tính **windowActionBar** là **false** (không có thanh công cụ cửa sổ) và **windowNoTitle** là **true** (không có tiêu đề). Các giá trị này được thiết lập vì bạn đang định nghĩa thanh ứng dụng với

AppBarLayout, thay vì sử dụng ActionBar. Việc sử dụng một trong các theme NoActionBar ngăn không cho ứng dụng sử dụng lớp ActionBar gốc để cung cấp thanh ứng dụng.

5. Nhìn vào **MainActivity**, kế thừa từ AppCompatActivity và bắt đầu với phương thức onCreate(), trong đó thiết lập view nội dung là layout activity\_main.xml và thiết lập toolbar là Toolbar được định nghĩa trong layout. Sau đó, nó gọi setSupportActionBar() và truyền toolbar vào, thiết lập toolbar là thanh ứng dụng cho Activity.

Để biết thêm về các phương pháp hay khi thêm thanh ứng dụng vào ứng dụng của bạn, hãy tham khảo Add the app bar.

## 1.2 Thêm các mục menu khác vào menu Tùy chọn

Bạn sẽ thêm các mục menu sau vào menu Tùy chọn:

- Order: Điều hướng đến thứ tự để xem thứ tự trang miện.
- Status: Kiểm tra trạng thái của một đơn đặt hàng.
- Contact: Hiển thị món tráng miệng yêu thích.
- Liên hệ: Liên hệ với quán cà phê. Bởi vì bạn không cần mục Cài đặt hiện có, bạn sẽ thay đổi cài đặt để liên hệ.

Android cung cấp định dạng XML tiêu chuẩn để xác định các mục menu. Thay vì xây dựng một menu trong mã hoạt động của bạn, bạn có thể xác định một menu và tất cả các mục menu của nó trong tài nguyên menu XML. Sau đó, bạn có thể thổi phồng tài nguyên menu (tải nó làm đối tượng menu) trong hoạt động của bạn:

1. Mở rộng Res> menu trong ngăn dự án> Android và Mở menu\_main.xml. Duy nhất mục menu được cung cấp từ mẫu là action\_sinstall (lựa chọn cài đặt), đó là được định nghĩa là:

```
<item
 android:id="@+id/action_settings"
 android:orderInCategory="100"
 android:title="Settings"
 app:showAsAction="never" />
```

2. Thay đổi các thuộc tính sau của mục action\_sinstall để biến nó thành mục action\_contact (không thay đổi thuộc tính Android hiện tại: Orderinc Category):

<b>Attribute</b>	<b>Value</b>
android:id	"@+id/action_contact"

android:title	"Contact"
app:showAsAction	"never"

3. Trích xuất chuỗi "Liên hệ" được mã hóa cứng vào chuỗi tài nguyên action\_contact.

4. Thêm mục menu mới bằng cách sử dụng thẻ <item> trong khối <sences> và cung cấp cho mục các thuộc tính sau:

<b>Attribute</b>	<b>Value</b>
android:id	"@+id/action_order"
android:orderInCategory	"10"
android:title	"Order"
app:showAsAction	"never"

Android: Thuộc tính Orderinc Category Chỉ định thứ tự trong đó các mục menu xuất hiện trong menu, với số thấp nhất xuất hiện cao hơn trong menu. Mục liên hệ được đặt thành 100, là một con số lớn để chỉ định rằng nó hiển thị ở phía

dưới thay vì trên cùng. Bạn đặt mục đặt hàng thành 10, đưa nó lên trên liên hệ và để lại nhiều chỗ trong menu để có thêm các mục.

5. Trích xuất chuỗi "thứ tự" được mã hóa cứng vào chuỗi tài nguyên action\_order.

6. Thêm hai mục menu nữa theo cùng một cách với các thuộc tính sau:

Status item attribute	Value
android:id	"@+id/action_status"
android:orderInCategory	"20"

android:title	"Status"
app:showAsAction	"never"

Favorites item attribute	Value
android:id	"@+id/action_favorites"
android:orderInCategory	"30"
android:title	"Favorites"
app:showAsAction	"never"

7. Trích xuất "Status" vào tài nguyên action\_status và "Favourite" vào tài nguyên action\_favorites.

8. Bạn sẽ hiển thị một thông báo dạng Toast với một thông điệp hành động tùy thuộc vào mục menu mà người dùng chọn. Mở tệp strings.xml và thêm các tên và giá trị chuỗi sau cho các thông báo này:

```
<string name="action_order_message">You selected Order.</string>
<string name="action_status_message">You selected Status.</string>
<string name="action_favorites_message">You selected Favorites.</string>
<string name="action_contact_message">You selected Contact.</string>
```

9. Mở MainActivity và thay đổi câu lệnh IF trong phương thức OnOptionItemSelected()

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

Chạy ứng dụng và nhấn vào biểu tượng tràn hành động, được hiển thị ở phía bên trái của hình bên dưới, để xem menu Tùy chọn, được hiển thị ở phía bên phải của hình bên dưới. Bạn sẽ sớm thêm các cuộc gọi lại để trả lời các mục được chọn từ menu này.

## Droid Desserts



Donuts are glazed and sprinkled with candy.

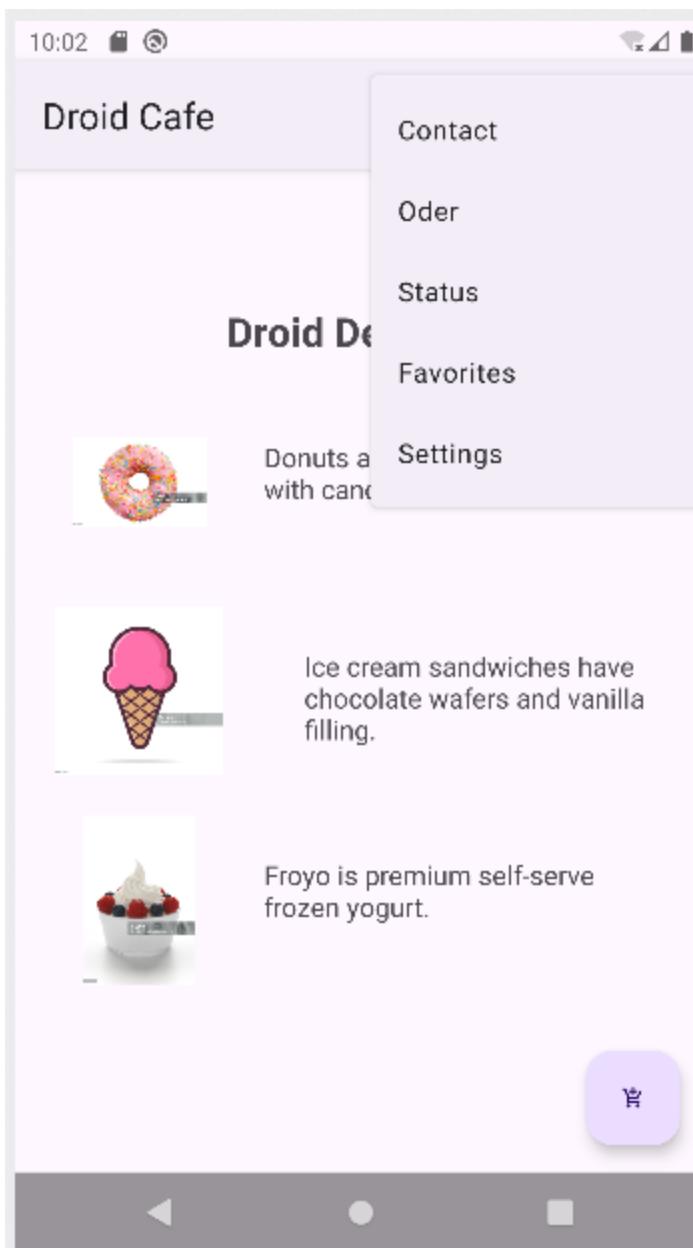


Ice cream sandwiches have chocolate wafers and vanilla filling.



Froyo is premium self-serve frozen yogurt.





Trong hình trên:

1. Nhấn vào biểu tượng tràn trong thanh ứng dụng để xem menu Tùy chọn.
2. Menu Tùy chọn giảm xuống từ thanh ứng dụng.

Lưu ý thứ tự các mục trong menu tùy chọn. Bạn đã sử dụng thuộc tính Android: OrderInCity để chỉ định mức độ ưu tiên của các mục menu trong menu: Mục đặt hàng là 10, tiếp theo là trạng thái (20) và mục yêu thích (30) và liên hệ là cuối cùng (100). Bảng sau đây cho thấy mức độ ưu tiên của các mục trong menu:

Menu item	orderInCategory attribute
Order	10
Status	20
Favorites	30
Contact	100

## Nhiệm vụ 2: Thêm biểu tượng cho các mục menu

Bất cứ khi nào có thể, bạn muốn hiển thị các hành động được sử dụng thường xuyên nhất bằng cách sử dụng các biểu tượng trong thanh ứng dụng, để người dùng có thể nhấp vào chúng mà không cần phải nhấp vào biểu tượng tràn (ba dấu chấm) trước. Trong nhiệm vụ này, bạn sẽ thêm biểu tượng cho một số mục menu và hiển thị một số mục menu trong thanh ứng dụng ở đầu màn hình dưới dạng biểu tượng.

Trong ví dụ này, giả sử rằng các hành động **Order** và **Status** là được sử dụng thường xuyên nhất. Hành động **Favorites** được sử dụng thỉnh thoảng, và **Contact** là ít được sử dụng nhất. Bạn có thể đặt biểu tượng cho các hành động này và chỉ định như sau:

- **Order** và **Status** nên luôn được hiển thị trong thanh ứng dụng.
- **Favorites** nên được hiển thị trong thanh ứng dụng nếu còn chỗ; nếu không, nó nên xuất hiện trong menu tràn.
- **Contact** không nên xuất hiện trong thanh ứng dụng; nó chỉ nên xuất hiện trong menu tràn.

### 2.1 Thêm biểu tượng cho các mục menu

Để chỉ định biểu tượng cho các hành động, trước tiên bạn cần thêm các biểu tượng dưới dạng tài nguyên hình ảnh vào thư mục **drawable**, sử dụng cùng quy trình bạn đã sử dụng trong thực hành với các hình ảnh có thẻ nhấp. Bạn cần sử dụng các biểu tượng sau (hoặc tương tự):

- **Order:** Sử dụng cùng biểu tượng mà bạn đã thêm cho nút hành động nổi trong thực hành sử dụng hình ảnh có thẻ nhấp (**ic\_shopping\_cart.png**).

- ⓘ **Status:**

- ❤ **Favorites:**

- Contact :Không cần biểu tượng vì nó sẽ chỉ xuất hiện trong menu tràn.

Đối với các biểu tượng trạng thái và yêu thích, hãy làm theo các bước sau:

1. Chọn **res** trong bảng **Project > Android** pane, và kích chuột phải) vào thư mục **drawable**.

2. Chọn **New > Image Asset**. Hộp thoại cấu hình tài sản hình ảnh xuất hiện.

3. Chọn Action Bar and Tab Itemstrong menu thả xuống.

4. Thay đổi **ic\_action\_name** sang tên khác (chẳng hạn như **IC\_STATUS\_INFO** cho biểu tượng trạng thái).

5. Nhập vào hình ảnh clip art (logo Android bên cạnh clipart :) để chọn hình ảnh clip art làm biểu tượng. Một trang của các biểu tượng xuất hiện. Nhập vào biểu tượng bạn muốn sử dụng.

6. Chọn **holo\_dark** từ menu thả xuống **Theme**. Điều này đặt biểu tượng là màu trắng trên nền màu tối (hoặc đen). Nhập vào **tiếp theo** và sau đó nhấp vào **Kết thúc**.

## 2.2 Hiển thị các mục menu dưới dạng biểu tượng trong thanh ứng dụng

Để hiển thị các mục menu dưới dạng biểu tượng trong thanh ứng dụng, hãy sử dụng thuộc tính ứng dụng: `showAsAction` trong menu\_main.xml.

Các giá trị sau cho thuộc tính chỉ định xem hành động có xuất hiện trong thanh ứng dụng dưới dạng biểu tượng hay không:

- "always": Luôn xuất hiện trong thanh ứng dụng. (Nếu không có đủ chỗ, nó có thể trùng với các biểu tượng menu khác.)
- "ifroom": xuất hiện trong thanh ứng dụng nếu có phòng.
- "Never": Không bao giờ xuất hiện trong thanh ứng dụng; Văn bản của nó xuất hiện trong menu tràn.

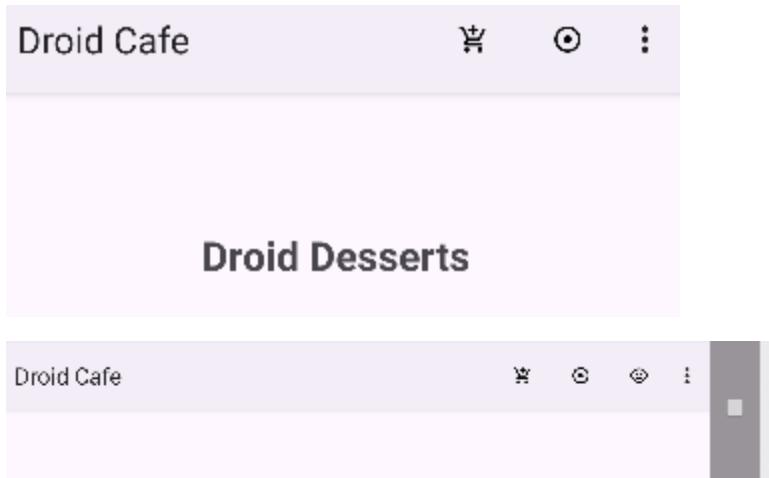
Thực hiện theo các bước này để hiển thị một số mục menu dưới dạng biểu tượng:

1. Mở menu\_main.xml một lần nữa và thêm các thuộc tính sau vào **Order**, **Status** và **Favourites** sao cho hai mục đầu tiên (thứ tự và trạng thái) luôn xuất hiện và mục yêu thích chỉ xuất hiện nếu có chỗ cho nó:

<b>Order item attribute</b>	<b>Old value</b>	<b>New value</b>
android:icon	"none"	"@drawable/ic_shopping_cart"
app:showAsAction	"never"	"always"

<b>Favorites item attribute</b>	<b>Old value</b>	<b>New value</b>
android:icon	"none"	"@drawable/ic_favorite"
app:showAsAction	"never"	"ifRoom"

- Chạy ứng dụng. Bây giờ bạn sẽ thấy ít nhất hai biểu tượng trong thanh ứng dụng: **Order** và **Status** như được hiển thị ở phía bên trái của hình bên dưới.
- Xoay thiết bị của bạn theo hướng ngang hoặc nếu bạn đang chạy trong trình giả lập, hãy nháp các biểu tượng **Rotate Left** or các biểu tượng t để xoay màn hình vào hướng ngang. Sau đó, bạn sẽ thấy tất cả ba biểu tượng trong thanh ứng dụng để **Order** , **Status** và **Favourite** như được hiển thị ở phía bên phải của hình bên dưới.



Có bao nhiêu nút hành động sẽ phù hợp với thanh ứng dụng? Nó phụ thuộc vào định hướng và kích thước của màn hình thiết bị. Ít nút hơn xuất hiện theo hướng thẳng đứng, như được hiển thị ở phía bên trái của hình trên, so với hướng ngang như thể hiện ở phía bên phải của hình trên. Các nút hành động có thể không chiếm hơn một nửa chiều rộng thanh ứng dụng chính.

### Nhiệm vụ 3: Xử lý mục menu đã chọn

Trong tác vụ này, bạn thêm một phương thức để hiển thị thông báo về mục menu nào được khai thác và sử dụng phương thức OnOptionsItemSelected () để xác định mục menu nào đã được khai thác.

#### 3.1 Tạo một phương thức để hiển thị lựa chọn menu

1. Mở MainActivity

2. Nếu bạn chưa thêm phương thức sau (trong một bài học khác) để hiển thị thông báo Toast, hãy thêm nó ngay bây giờ. Bạn sẽ sử dụng phương thức này làm hành động thực hiện cho mỗi lựa chọn trong menu. (Thông thường, bạn sẽ triển khai một hành động cụ thể cho mỗi mục menu, chẳng hạn như khởi động một Activity khác, như sẽ được trình bày sau trong bài học này.)

```
public void displayToast(String message) {
 Toast.makeText(getApplicationContext(), message,
 Toast.LENGTH_SHORT).show();
}
```

#### 3.2 Sử dụng trình xử lý sự kiện được chọn lọc

Phương thức **OnOptionsItemSelected ()** xử lý các lựa chọn từ menu Tùy chọn. Bạn sẽ thêm một khối trường hợp chuyển đổi để xác định mục menu nào đã được chọn và hành động nào sẽ thực hiện.

1. Tìm phương thức **onOptionsItemSelected()** được cung cấp bởi mẫu. Phương thức này xác định xem một mục menu cụ thể có được nhấp hay không, dựa trên **id** của mục menu. Trong ví dụ dưới đây, **id** là **action\_order**:

```
public boolean onOptionsItemSelected(MenuItem item) {
 // Handle action bar item clicks here. The action bar will
 // automatically handle clicks on the Home/Up button, so long
 // as you specify a parent activity in AndroidManifest.xml.
 int id = item.getItemId();

 //noinspection SimplifiableIfStatement
 if (id == R.id.action_order) {
 return true;
 }

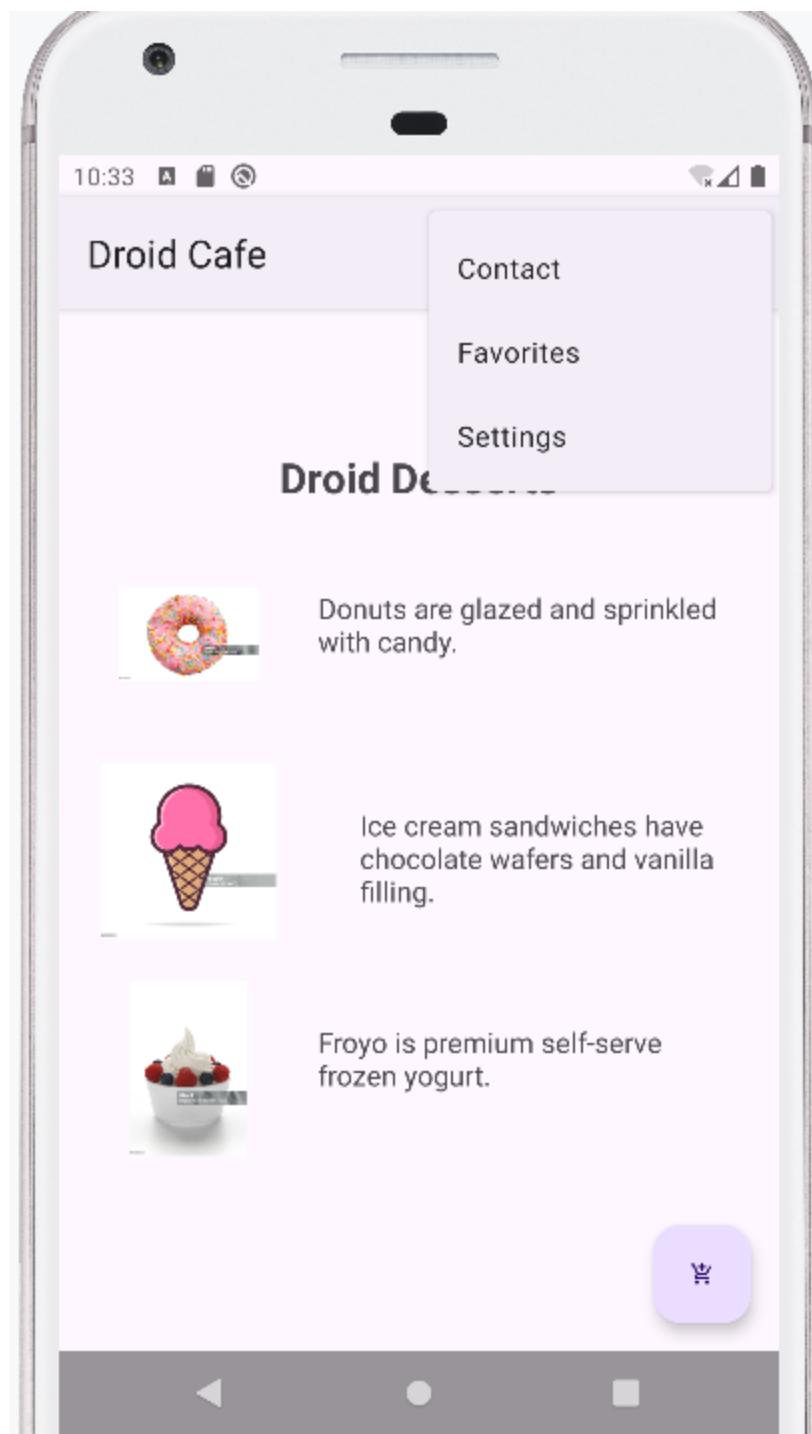
 return super.onOptionsItemSelected(item);
}
```

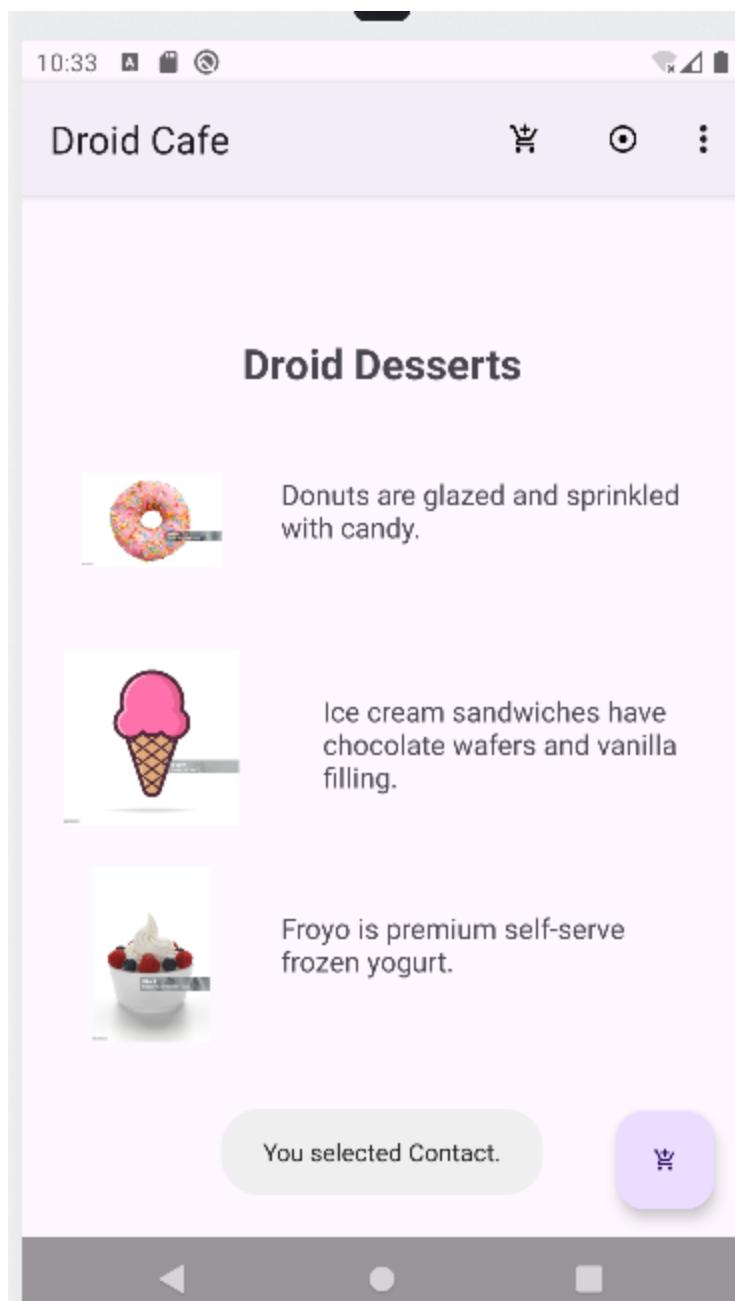
1. Thay thế câu lệnh gán **int id** và câu lệnh **if** bằng khối **switch-case** sau, khối này sẽ đặt thông báo (**message**) phù hợp dựa trên **id** của mục menu:

```
public boolean onOptionsItemSelected(MenuItem item) {
 switch (item.getItemId()) {
 case R.id.action_order:
 displayToast(getString(R.string.action_order_message));
 return true;
 case R.id.action_status:
 displayToast(getString(R.string.action_status_message));
 return true;
 case R.id.action_favorites:
 displayToast(getString(R.string.action_favorites_message));
 return true;
 }
}
```

```
case R.id.action_contact:
 displayToast(getString(R.string.action_contact_message));
 return true;
 default:
 // Do nothing
 }
 return super.onOptionsItemSelected(item);
}
```

2. Chạy ứng dụng. Bây giờ bạn sẽ thấy một thông báo **Toast** khác nhau trên màn hình, như được hiển thị ở phía bên phải của hình dưới đây, dựa trên mục menu mà bạn chọn.





Trong hình trên:

1. Chọn mục liên hệ trong menu Tùy chọn.
2. Thông điệp bánh mì nướng xuất hiện.

### 3.3 Bắt đầu một hoạt động từ một mục menu

Thông thường, bạn sẽ triển khai một hành động cho mỗi mục menu, chẳng hạn như bắt đầu một **Activity** khác. Tham khảo đoạn mã từ nhiệm vụ trước, hãy thay đổi mã cho trường hợp **action\_order** thành đoạn sau, đoạn mã này sẽ khởi

chạy **OrderActivity** (sử dụng cùng mã mà bạn đã sử dụng cho nút hành động nổi trong bài học về việc sử dụng hình ảnh có thể nhấp):

Chạy ứng dụng. Nhấp vào biểu tượng giỏ hàng trong thanh ứng dụng (mục **Order**) đưa bạn trực tiếp đến màn hình **OrderActivity**.

### Thử thách mã hóa

Thử thách: Menu ngữ cảnh cho phép người dùng thực hiện một hành động trên chế độ xem đã chọn. Mặc dù menu Tùy chọn trong thanh ứng dụng thường cung cấp các lựa chọn để điều hướng cho hoạt động khác, bạn sử dụng menu ngữ cảnh để cho phép người dùng sửa đổi chế độ xem trong hoạt động hiện tại.

Cả hai menu đều được mô tả trong XML, cả hai đều được khởi tạo bằng cách sử dụng menuinflater và cả hai đều sử dụng phương thức "trên vật phẩm được chọn" trong trường hợp này, OnContextItemSelected (). Vì vậy, các kỹ thuật xây dựng và sử dụng hai menu là tương tự nhau.

Menu ngữ cảnh xuất hiện dưới dạng danh sách nổi của các mục menu khi người dùng thực hiện cảm ứng & giữ aview, như được hiển thị ở phía bên trái của hình bên dưới. Đối với thử thách này, hãy thêm menu ngữ cảnh vào ứng dụng ScrollingText để hiển thị ba tùy chọn:**Edit**, **Share** và **Delete**, như trong hình bên dưới. Menu xuất hiện khi người dùng thực hiện cảm ứng và giữ TextView. Sau đó, ứng dụng hiển thị thông báo bánh mì nướng hiển thị tùy chọn menu được chọn, như được hiển thị ở phía bên phải của hình.

### Gợi ý

Menu ngữ cảnh tương tự như menu Tùy chọn, với hai sự khác biệt quan trọng:

- Menu ngữ cảnh phải được đăng ký để xem để menu khởi tạo khi chạm và giữ xảy ra trên chế độ xem.
- Mặc dù menu Tùy chọn được cung cấp bởi mẫu hoạt động cơ bản, đối với bối cảnh bạn phải tự thêm mã và tài nguyên menu.

Để giải quyết thách thức này, hãy làm theo các bước chung sau:

## 1. Tạo tệp tài nguyên menu XML cho các mục menu.

Nhấp chuột phải vào thư mục **New > Android Resource Directory**. Chọn Menu trong **menu** thả xuống Loại tài nguyên và bấm OK. Sau đó, nhấp chuột phải vào thư mục **menu** mới, chọn Tệp tài nguyên mới > Menu, nhập tên **menu\_context** và nhấp vào **OK**. Mở **menu\_context** và nhập các mục menu như bạn đã làm cho menu Tùy chọn.

## 2. Đăng ký Chế độ xem vào menu ngữ cảnh bằng phương thức RegisterForContextMenu () .

Trong phương thức OnCreate (), đăng ký TextView:

## 3. Thực hiện phương thức OnCreateContextMenu () trong hoạt động để khởi động menu.

4. Thực hiện phương thức OnContextItemSelected () trong hoạt động để xử lý các mục menu nhấp chuột. Trong trường hợp này, chỉ cần hiển thị bánh mì nướng với lựa chọn menu.

## 5. Chạy ứng dụng. Nếu bạn nhấn và kéo, văn bản cuộn như trước. Tuy nhiên, nếu bạn thực hiện một cú chạm dài, menu ngữ cảnh sẽ xuất hiện.

### **Nhiệm vụ 4: Sử dụng hộp thoại để yêu cầu sự lựa chọn của người dùng**

Bạn có thể cung cấp hộp thoại để yêu cầu sự lựa chọn của người dùng, chẳng hạn như cảnh báo yêu cầu người dùng nhấn OK hoặc hủy. Hộp thoại là một cửa sổ xuất hiện trên đỉnh của màn hình hoặc lấp đầy màn hình, làm gián đoạn luồng hoạt động.

Ví dụ: hộp thoại cảnh báo có thể yêu cầu người dùng nhấp vào Tiếp tục sau khi đọc nó hoặc cho người dùng lựa chọn đồng ý với một hành động bằng cách nhấp vào nút dương (chẳng hạn như OK hoặc Chấp nhận) hoặc không đồng ý bằng cách nhấp vào nút âm (như Hủy). Sử dụng lớp con alertDialog của lớp hộp thoại để hiển thị hộp thoại tiêu chuẩn để cảnh báo.

Trong thực tế này, bạn sử dụng một nút để kích hoạt hộp thoại cảnh báo tiêu chuẩn. Trong một ứng dụng trong thế giới thực, bạn có thể kích hoạt hộp thoại

cảnh báo dựa trên một số điều kiện hoặc dựa trên người dùng khai thác thứ gì đó.

#### 4.1 Tạo một ứng dụng mới để hiển thị hộp thoại cảnh báo

Trong bài tập này, bạn xây dựng một cảnh báo với các nút OK và hủy bỏ. Cảnh báo được kích hoạt bởi người dùng khai thác một nút.

1. Tạo một dự án mới có tên là **Hộp thoại để cảnh báo** dựa trên mẫu hoạt động trống.
2. Mở tệp Bố cục **Activity\_Main.xml** để hiển thị trình chỉnh sửa bố cục.
3. Chỉnh sửa phần tử **TextView** để nói Hello World! Nhấn để kiểm tra cảnh báo: Thay vì "**HelloWorld!**"
4. Thêm một nút dưới **TextView**. (Tùy chọn: Cắt nút ở dưới cùng của **TextView** và các cạnh của bố cục, với lề được đặt thành 8dp.)
5. Đặt văn bản của nút thành **Alert**.
6. Chuyển sang **Text tab** và trích xuất các chuỗi văn bản cho **TextView** và **Button** sang Chuỗi tài nguyên.
7. Thêm `Android:onClick` vào nút để gọi trình xử lý nhập vào `onClickShowAlert()`. Sau khi bạn nhập nó, trình xử lý nhập chuột được gạch chân màu đỏ vì nó chưa được tạo.

Bây giờ bạn có một bố cục tương tự như sau:

#### 4.2 Thêm hộp thoại cảnh báo vào hoạt động chính

Mẫu thiết kế xây dựng giúp dễ dàng tạo một đối tượng từ một lớp có nhiều thuộc tính cần thiết và tùy chọn và do đó sẽ yêu cầu rất nhiều tham số để xây dựng. Không có mô hình này, bạn sẽ phải tạo các hàm tạo để kết hợp và các thuộc tính tùy chọn; Với mẫu này, mã dễ đọc và bảo trì hơn. Để biết thêm thông tin về mẫu thiết kế xây dựng, hãy xem Builder pattern.

Lớp xây dựng thường là một lớp thành viên tĩnh của lớp mà nó xây dựng. Sử dụng `AlertDialog.Builder` để xây dựng hộp thoại cảnh báo tiêu chuẩn, với `setS`.

Để cảnh báo, bạn cần tạo một đối tượng của alertDialog.builder. Bạn sẽ thêm onclickShowalert () Nhập vào Trình xử lý cho nút cảnh báo, điều này làm cho đối tượng này làm

thứ tự đầu tiên của kinh doanh. Điều đó có nghĩa là hộp thoại sẽ chỉ được tạo khi người dùng nhấp vào nút cảnh báo. Mặc dù mẫu mã hóa này là hợp lý khi sử dụng nút để kiểm tra cảnh báo, nhưng đối với các ứng dụng khác, bạn có thể muốn tạo hộp thoại trong phương thức onCreate () để nó luôn sẵn sàng cho mã khác để kích hoạt nó.

1. Mở MainActivity và thêm phần đầu của phương thức onclickShowalert (

Nếu alertDialog.builder không được nhận ra khi bạn nhập nó, hãy nhấp vào biểu tượng bóng đèn màu đỏ và chọn phiên bản thư viện hỗ trợ (Android.support.v7.app.alertdialog) để nhập vào hoạt động của bạn.

2. Thêm mã để đặt tiêu đề và thông báo cho hộp thoại cảnh báo vào onclickshowalert () sau khi nhận xét:

3. Slide 98 tối về làm tiếp

## **Unit 3: Làm việc trong nền**

Đây là một tài liệu PDF chứa thông tin một lần về các bài học trong **Unit 3: Làm việc trong nền**.

**Các bài học trong đơn vị này:**

- **Bài học 7: Nhiệm vụ nền**
  - 7.1: AsyncTask
  - 7.2: AsyncTask và AsyncTaskLoader
  - 7.3: Broadcast receivers
- **Bài học 8: Kích hoạt, lên lịch và tối ưu hóa nhiệm vụ nền**
  - 8.1: Thông báo (Notifications)

- 8.2: Quản lý báo thức (Alarm Manager)
- 8.3: JobScheduler

## Bài học 7.1: AsyncTask

### Giới thiệu

Một **thread** là một đường dẫn độc lập của chương trình đang chạy. Khi một chương trình Android được khởi chạy, hệ thống tạo ra một **main thread**, còn gọi là **UI thread**. Đây là cách mà ứng dụng của bạn tương tác với các thành phần từ công cụ giao diện người dùng Android.

Đôi khi, ứng dụng cần thực hiện các nhiệm vụ đòi hỏi tài nguyên lớn như tải tệp, thực hiện truy vấn cơ sở dữ liệu, phát media, hoặc tính toán phân tích phức tạp. Các công việc này có thể làm chậm lại **UI thread** khiến ứng dụng không phản hồi với các thao tác của người dùng hoặc không vẽ được trên màn hình. Điều này có thể làm người dùng cảm thấy khó chịu và xóa ứng dụng.

Để giữ cho trải nghiệm người dùng (UX) luôn mượt mà, framework Android cung cấp một lớp trợ giúp gọi là **AsyncTask**, giúp xử lý công việc ngoài **UI thread**. Việc sử dụng **AsyncTask** để chuyển các tác vụ nặng vào một thread riêng giúp **UI thread** không bị chặn và vẫn phản hồi được.

Vì **thread** riêng biệt không đồng bộ với **thread** gọi, nên nó được gọi là **asynchronous thread**. **AsyncTask** cũng chứa một callback cho phép bạn hiển thị kết quả tính toán trở lại trên **UI thread**.

**Trong bài thực hành này, bạn sẽ học cách thêm một nhiệm vụ nền vào ứng dụng Android của bạn bằng cách sử dụng AsyncTask.**

### Bạn đã biết gì?

Bạn sẽ có khả năng:

- Tạo một **Activity**.
- Thêm một **TextView** vào layout của **Activity**.
- Lấy ID cho **TextView** và thiết lập nội dung của nó qua mã.
- Sử dụng các **Button** và chức năng **onClick** của chúng.

### Bạn sẽ học được gì?

- Cách thêm **AsyncTask** vào ứng dụng của bạn để chạy một tác vụ nền.

- Các nhược điểm khi sử dụng **AsyncTask** cho các nhiệm vụ nền.

### Những gì bạn sẽ làm:

- Tạo một ứng dụng đơn giản thực thi một tác vụ nền sử dụng **AsyncTask**.
- Chạy ứng dụng và xem điều gì xảy ra khi bạn xoay thiết bị.
- Cài đặt **activity instance state** để giữ trạng thái của một thông báo **TextView**.

### Tổng quan về ứng dụng:

Bạn sẽ xây dựng một ứng dụng có một **TextView** và một **Button**. Khi người dùng nhấn vào **Button**, ứng dụng sẽ "ngủ" trong một khoảng thời gian ngẫu nhiên, sau đó hiển thị một thông báo trong **TextView** khi nó thức dậy. Đây là hình ảnh của ứng dụng hoàn chỉnh:

### Nhiệm vụ 1: Thiết lập dự án SimpleAsyncTask

Giao diện người dùng của **SimpleAsyncTask** chứa một **Button** để khởi động **AsyncTask** và một **TextView** để hiển thị trạng thái của ứng dụng.

#### 1.1 Tạo dự án và giao diện

1. Tạo một dự án mới có tên **SimpleAsyncTask** sử dụng mẫu **Empty Activity**. Chấp nhận các tùy chọn mặc định cho tất cả các cài đặt khác.
2. Mở tệp giao diện **activity\_main.xml**. Nhập vào tab **Text**.
3. Thêm thuộc tính `layout_margin` vào **ConstraintLayout** cấp cao nhất:

```
xml
CopyEdit
android:layout_margin="16dp"
```

4. Thêm hoặc chỉnh sửa các thuộc tính sau của **TextView** chứa dòng chữ "Hello World!" để có các giá trị này. Trích xuất chuỗi này vào tài nguyên.

Attribute	Value
<code>android:id</code>	"@+id/textView1"
<code>android:text</code>	"I am ready to start work!"
<code>android:textSize</code>	"24sp"

5. Xóa các thuộc tính app:layout\_constraintRight\_toRightOf và app:layout\_constraintTop\_toTopOf.
6. Thêm một phần tử **Button** ngay dưới **TextView**, và gán cho nó các thuộc tính sau. Trích xuất văn bản của nút vào tài nguyên chuỗi.

Attribute	Value
android:id	"@+id/button"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:text	"Start Task"
android:layout_marginTop	"24dp"
android:onClick	"startTask"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/textView1"

7. Thuộc tính onClick của nút sẽ được làm nổi bật bằng màu vàng, vì phương thức startTask() chưa được triển khai trong **MainActivity**. Đặt con trỏ của bạn vào văn bản được làm nổi bật, nhấn Alt + Enter (hoặc Option + Enter trên Mac) và chọn **Create 'startTask(View)' in 'MainActivity'** để tạo phương thức khung trong **MainActivity**.

Mã giải pháp cho **activity\_main.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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"
 android:layout_margin="16dp"
 tools:context=".MainActivity">

 <TextView
 android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/ready_to_start"
 android:textSize="24sp"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"/>

 <Button
 android:id="@+id/button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginTop="24dp"
 android:onClick="startTask"
 android:text="@string/start_task"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/textView1"/>

</android.support.constraint.ConstraintLayout>
```

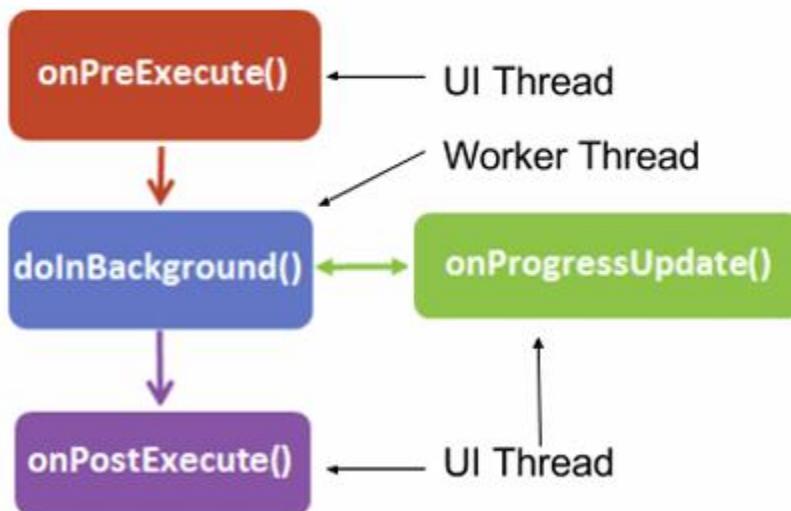
## Task 2: Tạo lớp con AsyncTask

AsyncTask là một lớp trừu tượng, điều này có nghĩa là bạn phải tạo một lớp con của nó để sử dụng. Trong ví dụ này, AsyncTask thực hiện một tác vụ nền rất đơn giản: nó ngủ trong một khoảng thời gian ngẫu nhiên. Trong một ứng dụng thực tế, tác vụ nền có thể thực hiện nhiều công việc khác nhau, từ truy vấn cơ sở dữ liệu, kết nối internet, cho đến tính toán nước đi tiếp theo trong trò chơi Go để đánh bại nhà vô địch Go hiện tại.

Một lớp con của AsyncTask có các phương thức sau để thực hiện công việc ngoài luồng chính:

- **onPreExecute()**: Phương thức này chạy trên luồng UI và được sử dụng để thiết lập tác vụ của bạn (chẳng hạn như hiển thị thanh tiến trình).

- **doInBackground()**: Đây là nơi bạn triển khai mã để thực hiện công việc sẽ được thực thi trên một luồng riêng biệt.
- **onProgressUpdate()**: Phương thức này được gọi trên luồng UI và dùng để cập nhật tiến trình trong giao diện người dùng (chẳng hạn như điền đầy thanh tiến trình).
- **onPostExecute()**: Lại trên luồng UI, phương thức này được sử dụng để cập nhật kết quả lên giao diện người dùng sau khi AsyncTask hoàn tất công việc.



Lưu ý: Một **background thread** (hay **worker thread**) là bất kỳ luồng nào không phải là luồng chính (UI thread).

Khi bạn tạo một lớp con của AsyncTask, bạn có thể cần cung cấp thông tin về công việc mà nó sẽ thực hiện, cách và việc có báo cáo tiến độ hay không, cũng như hình thức trả về kết quả. Khi bạn tạo một lớp con của AsyncTask, bạn có thể cấu hình nó bằng các tham số sau:

- **Params**: Kiểu dữ liệu của các tham số được gửi đến tác vụ khi thực thi phương thức ghi đè doInBackground().
- **Progress**: Kiểu dữ liệu của các đơn vị tiến độ được xuất bản thông qua phương thức ghi đè onProgressUpdated().
- **Result**: Kiểu dữ liệu của kết quả được cung cấp bởi phương thức ghi đè onPostExecute().

Ví dụ, một lớp con của AsyncTask có tên là MyAsyncTask với khai báo lớp như sau có thể sử dụng các tham số:

- Một String làm tham số trong doInBackground(), để sử dụng trong một truy vấn, chẳng hạn.
- Một Integer cho onProgressUpdate(), để đại diện cho phần trăm công việc đã hoàn thành.
- Một Bitmap làm kết quả trong onPostExecute(), biểu thị kết quả truy vấn.

java

CopyEdit

```
public class MyAsyncTask extends AsyncTask<String, Integer, Bitmap> { }
```

Trong tác vụ này, bạn sẽ sử dụng một lớp con của AsyncTask để xác định công việc sẽ chạy trong một luồng khác ngoài luồng giao diện người dùng (UI thread).

## 2.1 Tạo lớp con của AsyncTask

Trong ứng dụng này, lớp con AsyncTask mà bạn tạo không yêu cầu tham số truy vấn hoặc xuất bản tiến độ của nó. Bạn sẽ chỉ sử dụng các phương thức doInBackground() và onPostExecute().

1. Tạo một lớp Java mới có tên là SimpleAsyncTask, mở rộng AsyncTask và sử dụng ba tham số kiểu chung.

- Sử dụng Void cho tham số, vì AsyncTask này không yêu cầu bất kỳ đầu vào nào.
- Sử dụng Void cho kiểu tiến độ, vì tiến độ không được xuất bản.
- Sử dụng một String làm kiểu kết quả, vì bạn sẽ cập nhật TextView bằng một chuỗi khi AsyncTask hoàn thành việc thực thi.

```
public class SimpleAsyncTask extends AsyncTask<Void, Void, String> { }
```

**Lưu ý:** Khai báo lớp sẽ bị gạch đỏ, vì phương thức bắt buộc doInBackground() chưa được triển khai.

2. Ở đầu lớp, định nghĩa một biến thành viên mTextView với kiểu WeakReference<TextView>:

java

CopyEdit

```
private WeakReference<TextView> mTextView;
```

3. Triển khai một constructor cho AsyncTask nhận một TextView làm tham số và tạo một tham chiếu yếu mới cho TextView đó:

java

CopyEdit

```
SimpleAsyncTask(TextView tv) {
 mTextView = new WeakReference<>(tv);
}
```

AsyncTask cần cập nhật TextView trong Activity sau khi hoàn tất việc "ngủ" (trong phương thức onPostExecute()). Vì vậy, constructor của lớp sẽ cần một tham chiếu đến TextView để được cập nhật.

### **Tham chiếu yếu (lớp WeakReference) để làm gì?**

Nếu bạn truyền một TextView vào constructor của AsyncTask và sau đó lưu nó trong một biến thành viên, tham chiếu đó đến TextView sẽ khiến Activity không bao giờ bị thu gom rác (garbage collected) và do đó gây rò rỉ bộ nhớ, ngay cả khi Activity bị hủy và tạo lại, chẳng hạn khi thay đổi cấu hình thiết bị. Đây được gọi là **tạo ngứ cảnh rò rỉ (leaky context)**, và Android Studio sẽ cảnh báo bạn nếu bạn có làm điều này.

Tham chiếu yếu ngăn chặn rò rỉ bộ nhớ bằng cách cho phép đối tượng được giữ bởi tham chiếu đó bị thu gom rác nếu cần thiết.

## **2.2 Triển khai doInBackground()**

Phương thức doInBackground() là bắt buộc đối với lớp con của AsyncTask.

1. Đặt con trỏ vào khai báo lớp được đánh dấu, nhấn **Alt + Enter** (hoặc **Option + Enter** trên Mac) và chọn **Implement methods**. Chọn doInBackground() và nhấn **OK**. Mẫu phương thức sau sẽ được thêm vào lớp:

java

CopyEdit

@Override

```
protected String doInBackground(Void... voids) {
```

```
 return null;
}
```

- Thêm mã để tạo một số nguyên ngẫu nhiên từ 0 đến 10. Đây sẽ là số mili giây mà tác vụ sẽ tạm dừng. Số này không phải là thời gian dài để tạm dừng, vì vậy hãy nhân số đó với 200 để kéo dài thời gian:

java

CopyEdit

```
Random r = new Random();
int n = r.nextInt(11);
int s = n * 200;
```

- Thêm một khối try/catch và đưa luồng vào trạng thái ngủ:

java

CopyEdit

```
try {
 Thread.sleep(s);
} catch (InterruptedException e) {
 e.printStackTrace();
}
```

- Thay thế câu lệnh return hiện tại bằng câu lệnh trả về chuỗi "Awake at last after sleeping for xx milliseconds", trong đó xx là số mili giây mà ứng dụng đã tạm dừng:

java

CopyEdit

```
return "Awake at last after sleeping for " + s + " milliseconds!";
```

Phương thức doInBackground() hoàn chỉnh sẽ trông như sau:

java

CopyEdit

```

@Override
protected String doInBackground(Void... voids) {
 // Tạo một số ngẫu nhiên từ 0 đến 10
 Random r = new Random();
 int n = r.nextInt(11);
 // Đảm bảo tác vụ chạy đủ lâu để có thời gian xoay ngang điện thoại khi đang
 // chạy
 int s = n * 200;
 // Cho luồng tạm dừng trong khoảng thời gian ngẫu nhiên
 try {
 Thread.sleep(s);
 } catch (InterruptedException e) {
 e.printStackTrace();
 }
 // Trả về chuỗi kết quả
 return "Awake at last after sleeping for " + s + " milliseconds!";
}

```

### **2.3 Triển khai onPostExecute()**

Khi phương thức doInBackground() hoàn tất, giá trị trả về sẽ tự động được truyền tới hàm callback onPostExecute().

1. Triển khai onPostExecute() để nhận một đối số kiểu String và hiển thị chuỗi đó trong TextView:

java

CopyEdit

```

protected void onPostExecute(String result) {
 mTextView.get().setText(result);
}

```

```
}
```

Tham số String của phương thức này là tham số mà bạn đã định nghĩa ở tham số thứ ba trong khai báo lớp AsyncTask, và cũng là giá trị được trả về từ phương thức doInBackground().

Vì mTextView là một tham chiếu yếu, bạn cần sử dụng phương thức get() để trích xuất đối tượng TextView cơ bản, sau đó gọi setText() trên đối tượng đó.

### Lưu ý:

Bạn có thể cập nhật giao diện người dùng (UI) trong onPostExecute() vì phương thức này được chạy trên luồng chính (main thread). Tuy nhiên, bạn không thể cập nhật TextView bằng chuỗi mới trong phương thức doInBackground(), vì phương thức này được thực thi trên một luồng riêng biệt (background thread).

---

## Nhiệm vụ 3: Thực hiện các bước cuối cùng

### 3.1 Triển khai phương thức để bắt đầu AsyncTask

Ứng dụng của bạn hiện đã có một lớp AsyncTask thực hiện công việc trong nền (hoặc chỉ mô phỏng công việc bằng cách gọi sleep()). Nay, bạn có thể triển khai phương thức onClick cho nút "Start Task" để kích hoạt tác vụ chạy nền.

- Trong tệp MainActivity.java, thêm một biến thành viên để lưu trữ TextView:

```
java
```

```
CopyEdit
```

```
private TextView mTextView;
```

- Trong phương thức onCreate(), khởi tạo mTextView để liên kết với TextView trong giao diện:

```
java
```

```
CopyEdit
```

```
mTextView = findViewById(R.id.textView1);
```

- Trong phương thức startTask(), cập nhật TextView để hiển thị dòng chữ "Napping...". Trích xuất thông báo đó thành một tài nguyên chuỗi:

java

CopyEdit

```
mTextView.setText(R.string.napping);
```

4. Tạo một thê hiện của SimpleAsyncTask, truyền TextView (mTextView) vào constructor. Gọi execute() trên thê hiện SimpleAsyncTask:

java

CopyEdit

```
new SimpleAsyncTask(mTextView).execute();
```

**Lưu ý:**

Phương thức execute() là nơi bạn truyền các tham số, được phân tách bằng dấu phẩy, sau đó được hệ thống truyền vào doInBackground(). Vì AsyncTask này không có tham số, bạn để trống.

**Mã trong MainActivity:**

### 3.2 Triển khai onSaveInstanceState()

- Chạy ứng dụng và nhấp vào nút **Start Task**. Ứng dụng sẽ "ngủ" trong bao lâu?
- Nhấp lại vào nút **Start Task**, và khi ứng dụng đang "ngủ", xoay thiết bị. Nếu tác vụ nền hoàn thành trước khi bạn có thể xoay điện thoại, hãy thử lại.

**Lưu ý:** Bạn sẽ nhận thấy rằng khi thiết bị được xoay, TextView sẽ đặt lại về nội dung ban đầu, và AsyncTask dường như không thê cập nhật TextView.

Có một số điều đang xảy ra ở đây:

Chương 2: TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1: Tương tác người dùng

1.1 Hình ảnh có thê chọn

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của một thiết bị chạy Android bao gồm một hệ thống phân cấp các đối tượng 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 chạm hoặc nhấp vào để thực hiện một hành động.

Bạn có thể biến bất kỳ View nào, chẳng hạn như ImageView, thành một phần tử UI có thể được chạm hoặc nhấp. Bạn phải lưu hình ảnh cho ImageView trong thư mục drawables của dự án của bạn. Trong bài thực hành này, bạn sẽ học cách sử dụng hình ảnh như là các phần tử mà người dùng có thể chạm hoặc nhấp.

## Những thứ bạn nên biết

Bạn có thể làm gì:

- Tạo một dự án Android Studio ở mẫu và tạo giao diện chính
- Chạy ứng dụng trên trình giả lập hoặc thiết bị đã kết nối
- Tạo và chỉnh sửa các yếu tố giao diện sử dụng trình chỉnh sửa giao diện và mã XML.
- Truy cập các yếu tố giao diện người dùng và sử dụng từ mã bởi `findViewById()`.
- Xử lý sự kiện nhấn nút
- Hiển thị thông báo Toast.
- Thêm hình ảnh vào thư mục drawable của dự án.

## Những điều bạn học

- Cách sử dụng hình ảnh như một yếu tố tương tác để thực hiện một hành động.
- Cách thiết lập thuộc tính cho các yếu tố ImageView trong trình chỉnh sửa bối cảnh.
- Cách thêm phương thức `onClick()` để hiển thị một thông báo Toast.

## Những điều bạn làm

- Tạo một dự án Android Studio mới cho một ứng dụng đặt bánh giả sử dụng hình ảnh làm các yếu tố tương tác.
- Đặt 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 động một Activity khác.

## Tổng quan về ứng dụng

Tổng quan về ứng dụng trong thực hành này, bạn sẽ tạo và xây dựng một ứng dụng mới bắt đầu từ mẫu Hoạt động Cơ bản mô phỏng một ứng dụng đặt món tráng miệng. Người dùng có thể chạm vào một hình ảnh để thực hiện một hành động - trong trường hợp này là hiển thị một thông báo Toast - như được thể hiện trong hình bên dưới. Người dùng cũng có thể chạm vào một nút giở hàng để tiếp tục đến Hoạt động tiếp theo.

### Task 1: Thêm hình ảnh vào bộ cục

Bạn có thể làm cho một giao diện có thể nhấp vào, như một nút, bằng cách thêm thuộc tính android:onClick trong layout XML. Ví dụ, bạn có thể làm cho một hình ảnh hoạt động như một nút bằng cách thêm android:onClick vào ImageView.

Trong nhiệm vụ này, bạn tạo một nguyên mẫu ứng dụng đặt món tráng miệng từ một quán cà phê. Sau khi bắt đầu một dự án mới dựa trên mẫu Basic Activity, bạn chỉnh sửa TextView “Hello World” với văn bản phù hợp và thêm hình ảnh mà người dùng có thể chạm vào.

#### 1.1 Bắt đầu dự án mới

1. Bắt đầu một dự án Android Studio mới với tên ứng dụng là **Droid Cafe**.
2. Chọn mẫu Basic Activity và chấp nhận tên Activity mặc định (MainActivity). Đảm bảo tùy chọn Generate Layout file và Backwards Compatibility (AppCompat) đã được chọn.
3. Nhấn **Finish**.

Dự án sẽ mở với hai layout trong thư mục **res > layout**: activity\_main.xml cho thanh ứng dụng và nút hành động nổi (mà bạn không thay đổi trong nhiệm vụ này), và content\_main.xml cho mọi thứ khác trong layout.

4. Mở **content\_main.xml** và nhấn vào tab **Design** (nếu chưa được chọn) để hiển thị trình soạn thảo layout.

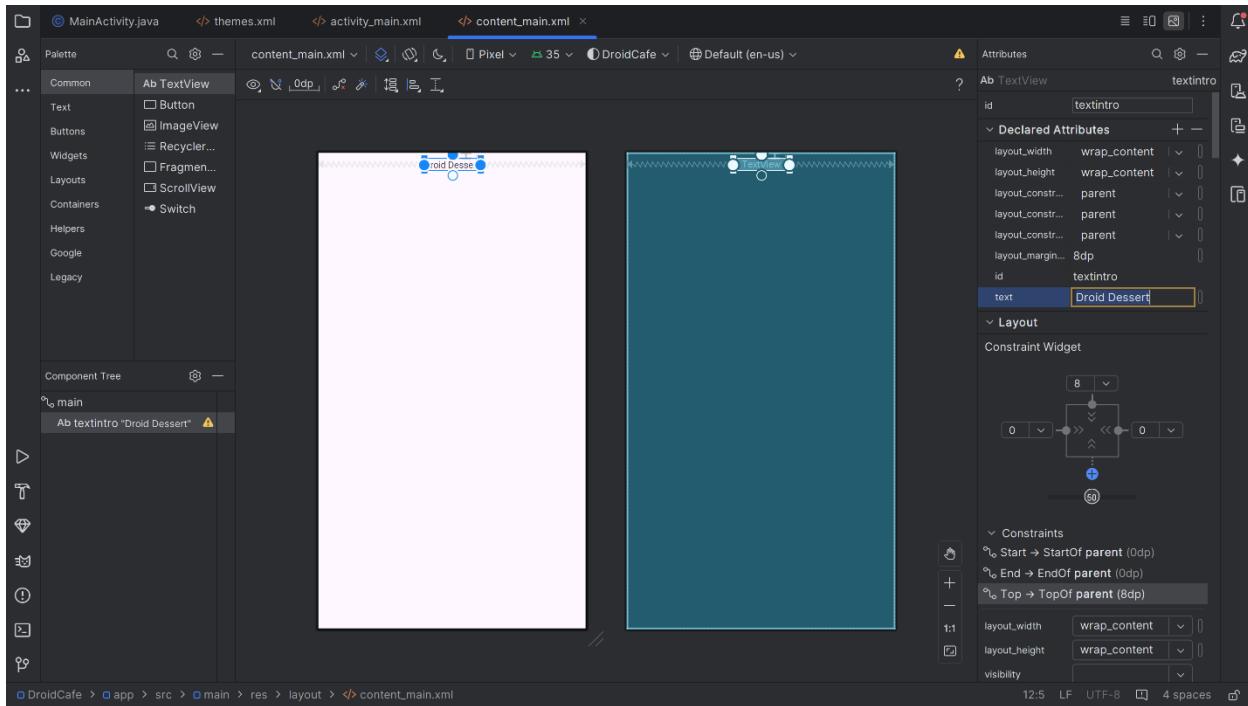
5. Chọn TextView "Hello World" trong layout và mở bảng **Attributes**.

6. Thay đổi các thuộc tính textintro như sau:

Attribute field	Enter the following:
ID	textintro
text	Change Hello World to Droid Dessert
textStyle	B (bold)
textSize	24sp

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

7. Xóa ràng buộc kéo dài từ đáy của TextView textintro đến đáy của bố cục, để TextView gắn vào đỉnh của bố cục, và chọn **8** (8dp) cho khoảng cách ở trên như hình dưới.



8. Trong bài học trước, bạn đã học cách trích xuất tài nguyên chuỗi từ một chuỗi văn bản tĩnh. Nhập vào tab **Text** bản để chuyển sang mã XML và trích xuất chuỗi "Droid Desserts" trong

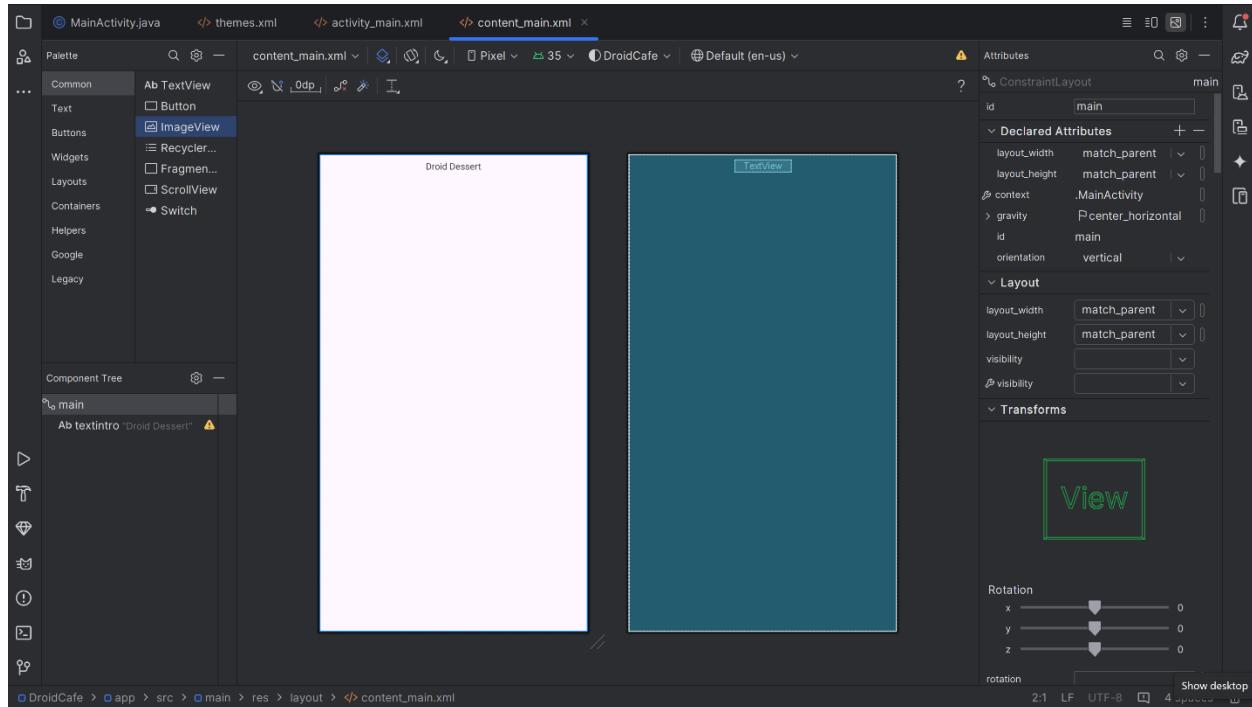
## 1.2 Thêm hình ảnh

Ba hình ảnh (donut\_circle.png, froyo\_circle.png và icecream\_circle.png) được cung cấp cho ví dụ này, bạn có thể [download](#). Thay vào đó, bạn có thể thay thế bằng các hình ảnh của riêng bạn dưới dạng tệp PNG, nhưng chúng phải có kích thước khoảng 113 x 113 pixel để sử dụng trong ví dụ này.

Bước này cũng giới thiệu một kỹ thuật mới trong trình chỉnh sửa bô cục: sử dụng nút **Fix** trong các tin nhắn cảnh báo để trích xuất tài nguyên chuỗi.

1. Để sao chép hình ảnh vào dự án của bạn, trước tiên hãy đóng dự án.
2. Sao chép các tệp hình ảnh vào thư mục **drawable** của dự án của bạn. Tìm thư mục **drawable** trong một dự án bằng cách sử dụng đường dẫn này: **project\_name > app > src > main > res > drawable**.
3. Mở lại dự án của bạn.
4. Mở tệp **content\_main.xml** và nhấp vào tab **Design** (nếu nó chưa được chọn).

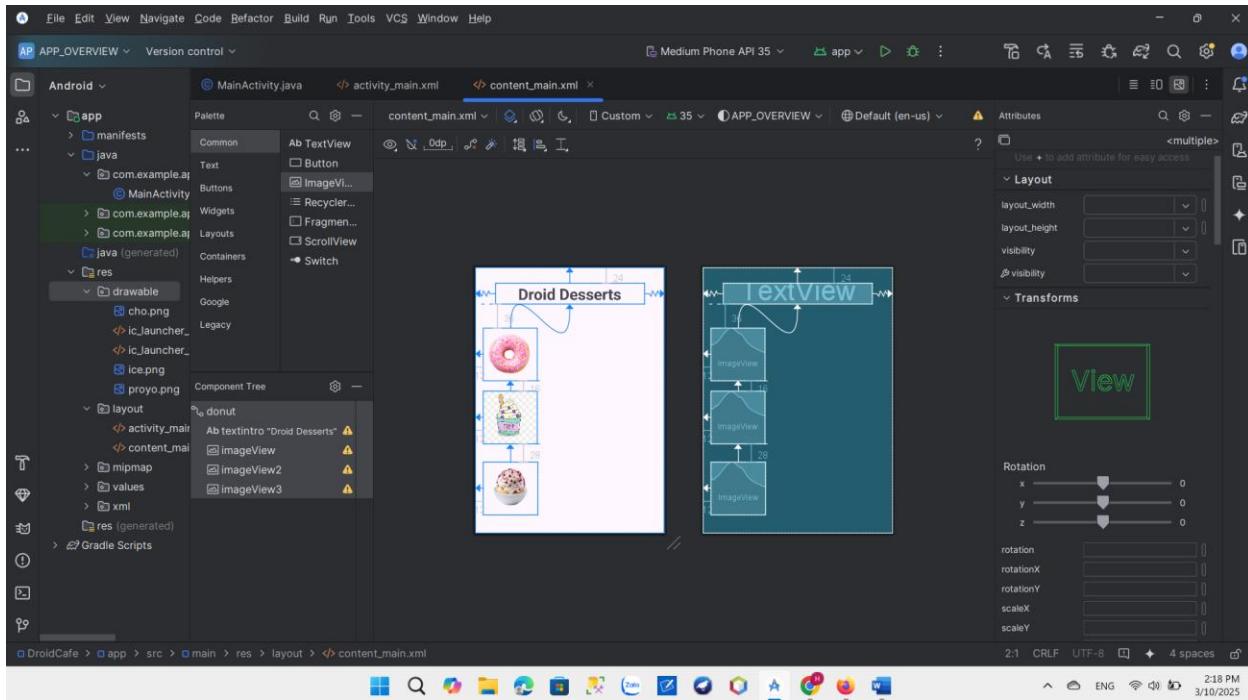
5. Kéo một ImageView vào bố cục, chọn hình ảnh **donut\_circle** cho nó và ràng buộc nó với TextView ở trên cùng và phía bên trái của bố cục với khoảng cách **24** (24dp) cho cả hai ràng buộc, như được thể hiện trong hình chuyển động bên dưới.



6. Trong bảng thuộc tính, nhập các giá trị sau cho các thuộc tính:

Attribute field	Enter the following:
ID	donut
contentDescription	Donuts are glazed and sprinkled with candy. (You can copy/paste the text into the field.)

7. Kéo một ImageView thứ hai vào bố cục, chọn hình ảnh **icecream\_circle** cho nó, và ràng buộc nó vào đáy của ImageView đầu tiên và vào bên trái của bố cục với một khoảng cách **24** (24dp) cho cả hai ràng buộc.



8. Trong bảng thuộc tính, nhập các giá trị sau cho các thuộc tính:

Attribute field	Nhập các 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.)

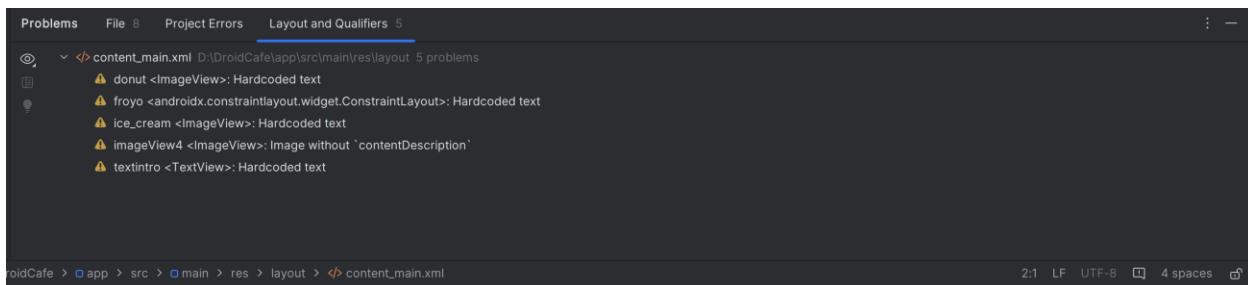
9. Kéo một ImageView thứ ba vào bố cục, chọn hình ảnh **froyo\_circle** cho nó, và ràng buộc nó vào đáy của ImageView thứ hai và cạnh trái của bố cục với một khoảng cách **24** (24dp) cho cả hai ràng buộc.

10. Trong bảng thuộc tính, nhập các giá trị sau cho các thuộc tính:

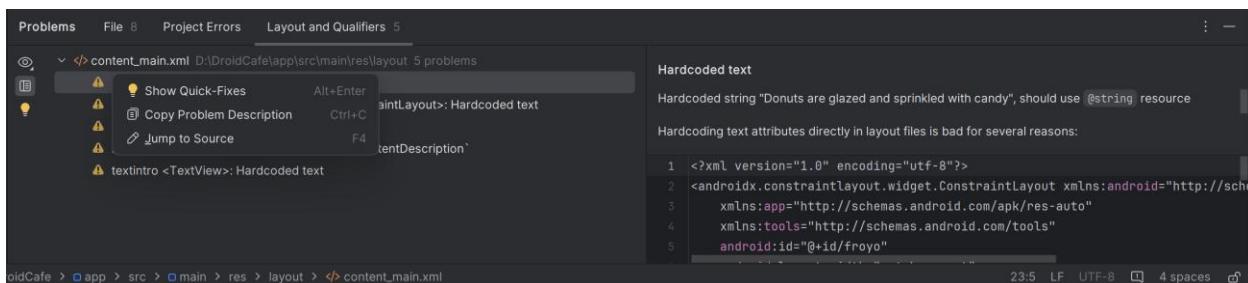
Trường thuộc tính	Nhập các thông tin sau
ID	froyo

contentDescription	FroYo is premium self-serve frozen yogurt. (You can copy/paste the text into the field.)
--------------------	------------------------------------------------------------------------------------------

11. Nhập vào biểu tượng  ở góc trên bên trái của trình chỉnh sửa bộ cục để mở bảng cảnh báo, bảng này sẽ hiển thị các cảnh báo về văn bản được mã hóa cứng:



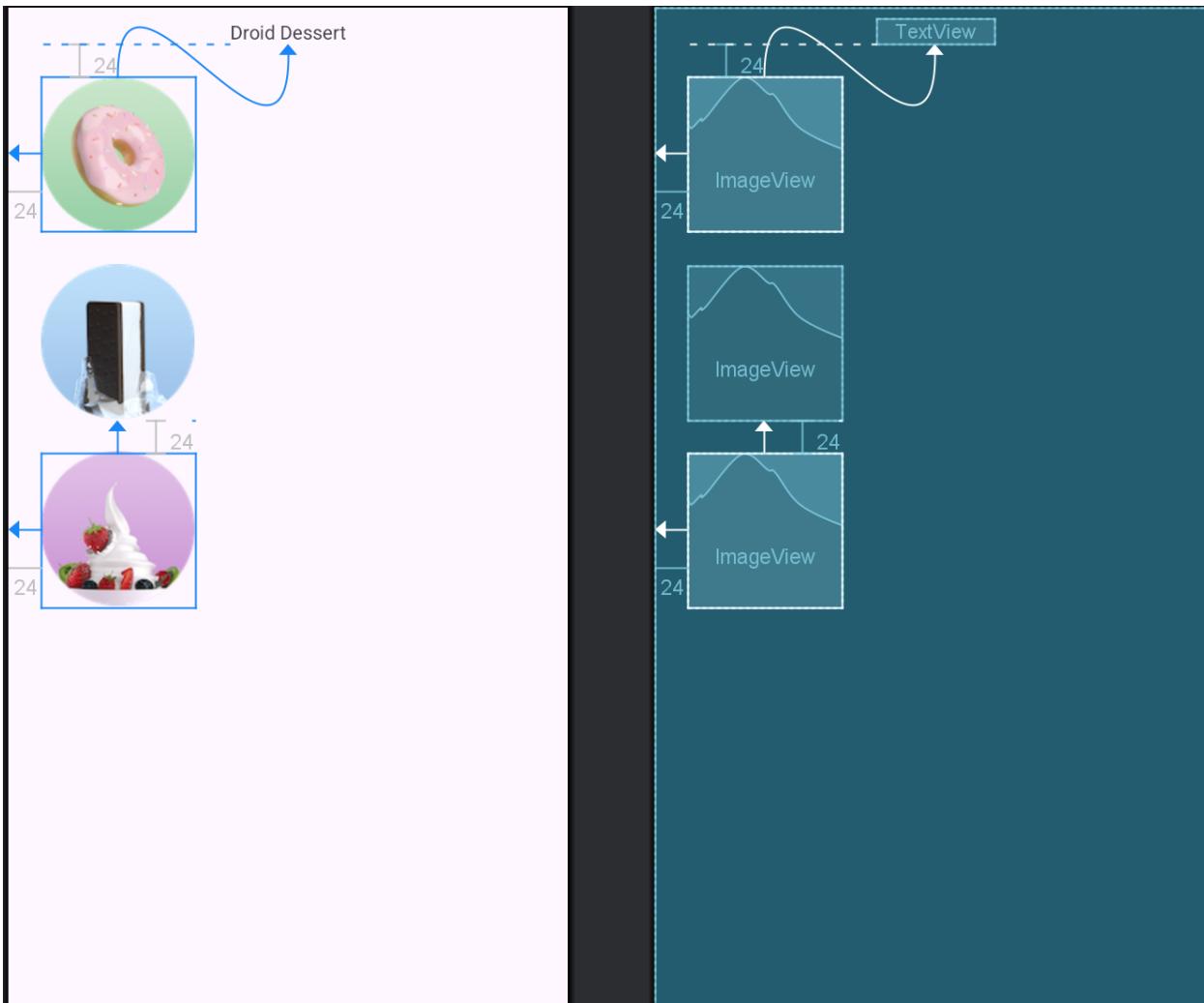
12. Mở rộng từng cảnh báo **Hardcoded text**, cuộn xuống dưới cùng của thông điệp cảnh báo, và nhấp vào nút **Fix** như hình bên dưới:



Sửa lỗi cho mỗi cảnh báo văn bản mã cứng trích xuất tài nguyên chuỗi cho chuỗi. Hộp thoại **Extract Resource** xuất hiện, và bạn có thể nhập tên cho tài nguyên chuỗi. Nhập các tên sau cho các tài nguyên chuỗi:

String	Enter the following name:
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

## Bố cục sẽ giống hình dưới đây

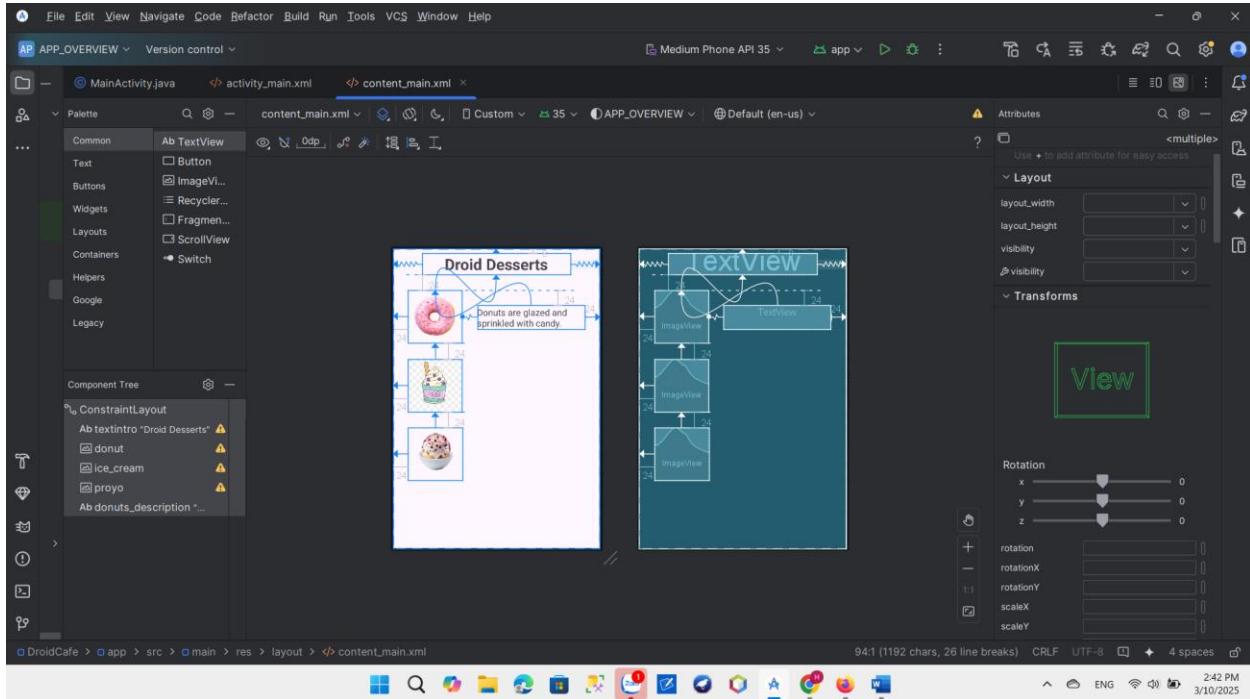


### 1.3 Thêm miêu tả văn bản

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

1. Kéo một phần tử TextView vào bố cục.
  2. Ràng buộc cạnh trái của phần tử với cạnh phải của ImageView donut và cạnh trên với cạnh trên của ImageView donut, cả hai đều có khoảng cách **24** (24dp).

3. Ràng buộc cạnh phải của phần tử với cạnh phải của bố cục và sử dụng khoảng cách giống như 24 (24dp). Nhập **donut\_description** vào trường ID trong bảng thuộc tính. **TextView** mới sẽ xuất hiện bên cạnh hình ảnh donut như hình dưới đây.



4. Trong thanh Thuộc tính, thay đổi độ rộng trong bảng kiểm tra thành **Match Constraints**:

id donut\_description

Declared Attributes

layout_width	0dp
layout_height	wrap_content
layout_constraintTop_toTopOf	@+id/donut
layout_constraintBottom_toBottomOf	@+id/donut
layout_marginTop	24dp
layout_marginBottom	24dp
id	donut_description
text	TextView

Layout

Constraint Widget

24

0

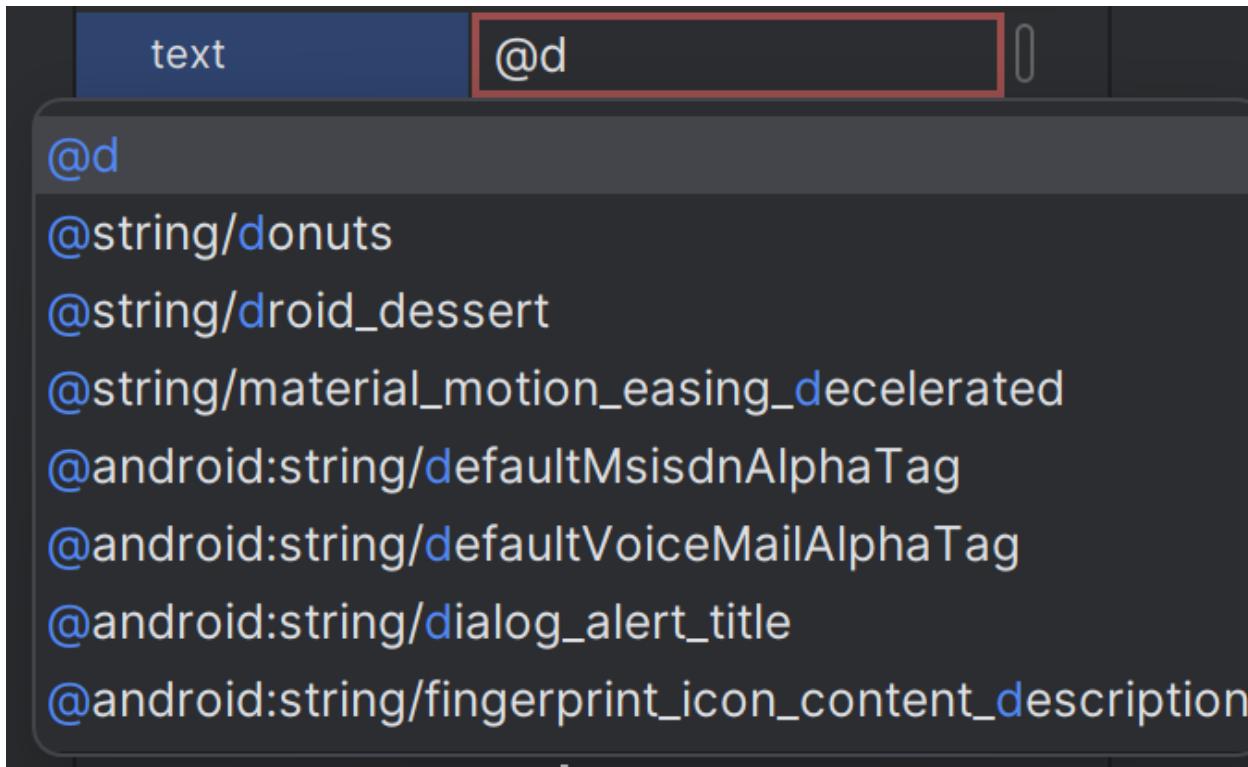
24

Match Constraints

50

5. Trong bảng Thuộc tính, bắt đầu nhập tài nguyên chuỗi cho trường văn bản bằng cách đặt trước nó với ký hiệu @: @d. Nhập vào tên tài nguyên chuỗi (@string/donuts) mà xuất hiện như một gợi ý

Gợi ý



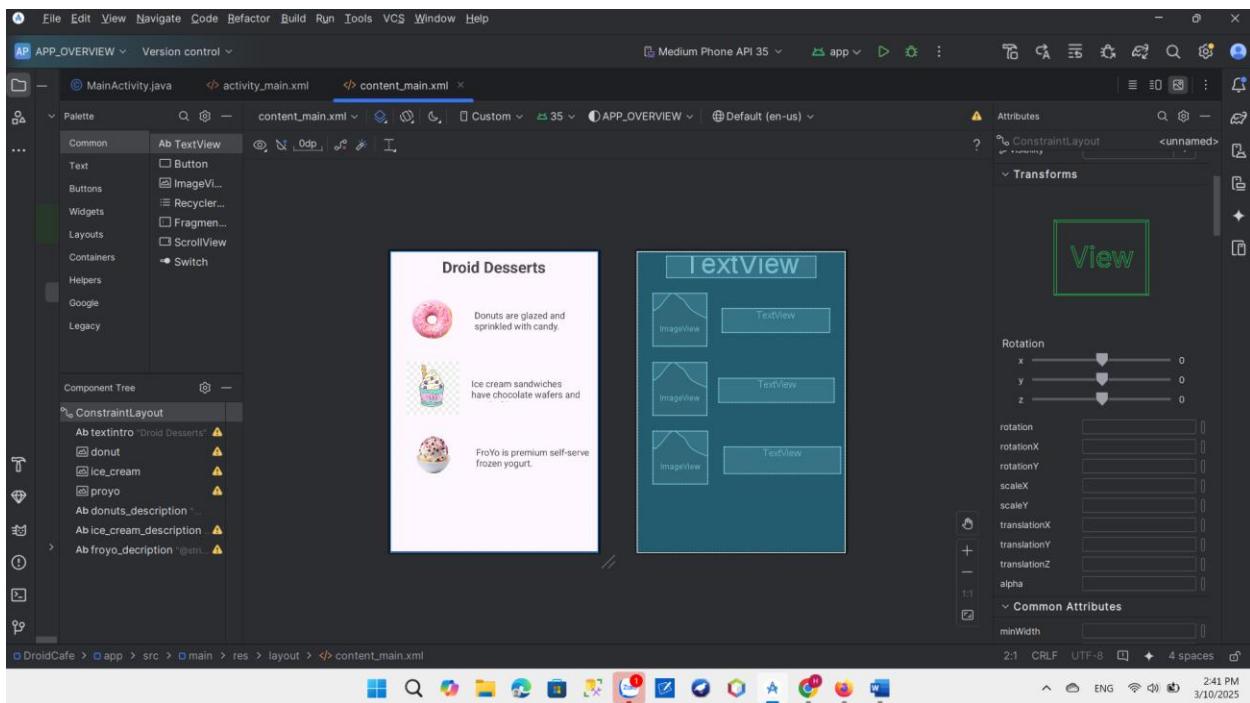
6. Lặp lại các bước trên để thêm một TextView thứ hai được ràng buộc ở bên phải và trên của ImageView ice\_cream, và cạnh phải của nó ràng buộc với cạnh phải của bố cục. Nhập thông tin sau vào bảng Thuộc tính:

Attribute field	Enter the following:
ID	ice_cream_description
Left, right, and top margins	24
layout_width	match_constraint
text	@string/ice_cream_sandwiches

7. Lặp lại các bước trên để thêm một TextView thứ ba bị ràng buộc ở bên phải và phía trên của froyo ImageView, và bên phải của nó với bên phải của layout. Nhập những thông tin sau vào bảng Attributes :

Attribute field	Enter the following:
ID	froyo_description
Left, right, and top margins	24
layout_width	match_constraint
text	@string/froyo

Bố cục sẽ như sau:



Task 1: mã giải pháp

Bố cục XML cho tệp content.xml được hiển thị dưới đây.

```
<?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"
 android:contentDescription="@string/froyo"
 android:gravity="center_horizontal"
 android:orientation="vertical"
 tools:context=".MainActivity">

<ImageView
 android:id="@+id/donut"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginStart="24dp"
 android:layout_marginTop="24dp"
 android:contentDescription="@string/donuts"
 android:src="@drawable/donut_circle"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/textintro"
 tools:ignore="ImageContrastCheck" />

<TextView
```

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

```
<ImageView
 android:id="@+id/ice_cream"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="24dp"
 android:layout_marginTop="24dp"
 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="wrap_content"
```

```
 android:layout_height="wrap_content"
 android:layout_marginStart="24dp"
 android:layout_marginTop="24dp"
 android:contentDescription="@string/froyo"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/ice_cream"
 app:srcCompat="@drawable/froyo_circle" />
```

```
<TextView
 android:id="@+id/donut_description"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="24dp"
 android:layout_marginTop="24dp"
 android:layout_marginEnd="24dp"
 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_marginStart="24dp"
```

```
 android:layout_marginTop="24dp"
 android:layout_marginEnd="24dp"
 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_marginStart="24dp"
 android:layout_marginTop="24dp"
 android:layout_marginEnd="24dp"
 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>
```

## Task 2: Thêm phương thức onClick cho hình ảnh

Để làm cho một View *clickable*, để người dùng có thể chạm (hoặc nhấn) vào nó, hãy thêm thuộc tính android:onClick trong bộ cục XML và chỉ định bộ xử lý nhấp.

Ví dụ, bạn có thể làm cho một ImageView hoạt động như một nút đơn giản bằng cách thêm android:onClick vào ImageView. Trong nhiệm vụ này, bạn làm cho các hình ảnh trong bộ cục của bạn có thể nhấn được.

## 2.1 Tạo phương thức Toast

Trong nhiệm vụ này, bạn thêm từng phương thức cho thuộc tính android:onClick để gọi khi mỗi hình ảnh được nhấp. Trong nhiệm vụ này, các phương thức này đơn giản hiển thị một thông điệp Toast cho biết hình ảnh nào đã được chạm. (Trong một chương khác, bạn sửa đổi các phương thức này để khởi động một Activity khác.)

1. Để sử dụng tài nguyên chuỗi trong mã Java, bạn nên thêm chúng vào tệp strings.xml trước. Mở rộng **res > values** trong bảng **Project > Android**, và mở tệp **strings.xml**. Thêm các tài nguyên chuỗi sau cho các chuỗi sẽ được hiển thị trong thông điệp Toast:

```
<string name="donut_order_message">You ordered a donut.</string>
<string name="ice_cream_order_message">You ordered an ice cream sandwich.</string>
<string name="froyo_order_message">You ordered a FroYo.</string>
```

2. Mở **MainActivity** và thêm phương thức `displayToast()` sau cùng vào **MainActivity** (trước dấu ngoặc đóng):

```
public void displayToast(String message) {
 Toast.makeText(getApplicationContext(), message,
 Toast.LENGTH_SHORT).show();
}
```

Mặc dù bạn có thể đã thêm phương thức này ở bất kỳ vị trí nào trong **MainActivity**, nhưng thực tiễn tốt nhất là đặt các phương thức của bạn bên dưới các phương thức đã được cung cấp trong **MainActivity** bởi mẫu.

## 2.2 Tạo trình xử lý sự kiện khi nhấp chuột

Mỗi hình ảnh có thể nhấp cần một trình xử lý sự kiện khi nhấp chuột - một phương thức để thuộc tính android:onClick gọi. Trình xử lý sự kiện khi nhấp chuột, nếu được gọi từ thuộc tính android:onClick, phải là public, trả về void và định nghĩa một View là tham số duy nhất của nó. Làm theo các bước sau để thêm trình xử lý sự kiện khi nhấp chuột:

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

```
/**
 * Shows a message that the donut image was clicked.
 */
no usages
public void showDonutOrder(View view) {
 displayToast(getString(R.string.donut_order_message));
}
```

Ba dòng đầu tiên là một chú thích theo định dạng Javadoc, giúp cho mã dễ hiểu hơn và cũng giúp tạo tài liệu cho mã của bạn. Đây là một thực tiễn tốt nhất để thêm chú thích như vậy cho mỗi phương thức mới bạn tạo. Để biết thêm thông tin về cách viết chú thích, hãy xem Cách viết chú thích tài liệu cho công cụ Javadoc.

2. Thêm nhiều phương thức vào cuối **MainActivity** cho mỗi món tráng miệng:

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

3. (Tùy chọn) Chọn **Code > Reformat Code** để định dạng lại mã mà bạn đã thêm vào MainActivity cho phù hợp với tiêu chuẩn và dễ đọc hơn.

## 2.3 Thêm thuộc tính onClick

Trong bước này, bạn thêm android:onClick vào từng phần tử ImageView trong tập tin content\_main.xml. Thuộc tính android:onClick gọi trình xử lý click cho từng phần tử.

1. Mở tập tin **content\_main.xml**, và nhấp vào tab **Text** trong trình chỉnh sửa bố cục để hiển thị mã XML.
2. Thêm thuộc tính android:onClick vào ImageView donut. Khi bạn nhập, sẽ có các gợi ý hiển thị các trình xử lý click. Chọn trình xử lý click showDonutOrder. Mã hiện tại sẽ trông như sau:

```
<ImageView
 android:id="@+id/donut"
 android:layout_width="96dp"
 android:layout_height="94dp"
 android:layout_marginStart="@dimen/margin_wide"
 android:layout_marginTop="@dimen/margin_wide"
 android:contentDescription="@string/donuts"
 android:onClick="showDonutOrder"
 android:padding="10dp"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/textintro"
 app:srcCompat="@drawable/cho" />
```

Dòng cuối cùng (android:onClick="showDonutOrder") gán trình xử lý nhấp chuột (showDonutOrder) cho ImageView.

3. (Tùy chọn) Chọn **Code > Reformat Code** để định dạng lại mã XML bạn đã thêm vào content\_main.xml để tuân thủ các tiêu chuẩn và dễ đọc hơn. Android Studio tự động di chuyển thuộc tính android:onClick lên vài dòng để kết hợp chúng với các thuộc tính khác có android: làm tiền tố.

4. Thực hiện quy trình tương tự để thêm thuộc tính android:onClick vào các phần tử ImageView ice\_cream và froyo. Chọn các trình xử lý nhấp chuột showDonutOrder và showFroyoOrder. Bạn có thể tùy chọn chọn **Code > Reformat Code** để định dạng lại mã XML. Mã bây giờ sẽ trông như sau:

```
<ImageView
 android:id="@+id/ice_cream"
 android:layout_width="96dp"
 android:layout_height="93dp"
 android:layout_marginStart="@dimen/margin_wide"
 android:layout_marginTop="@dimen/margin_wide"
 android:contentDescription="@string/ice_cream_sandwiches"
 android:onClick="showIceCream"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/donut"
 app:srcCompat="@drawable/proyo" />

<ImageView
 android:id="@+id/proyo"
 android:layout_width="95dp"
 android:layout_height="92dp"
 android:layout_marginStart="@dimen/margin_wide"
 android:layout_marginTop="@dimen/margin_wide"
 android:contentDescription="@string/froyo"
 android:onClick="showFroyoOrder"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/ice_cream"
 app:srcCompat="@drawable/ice" />
```

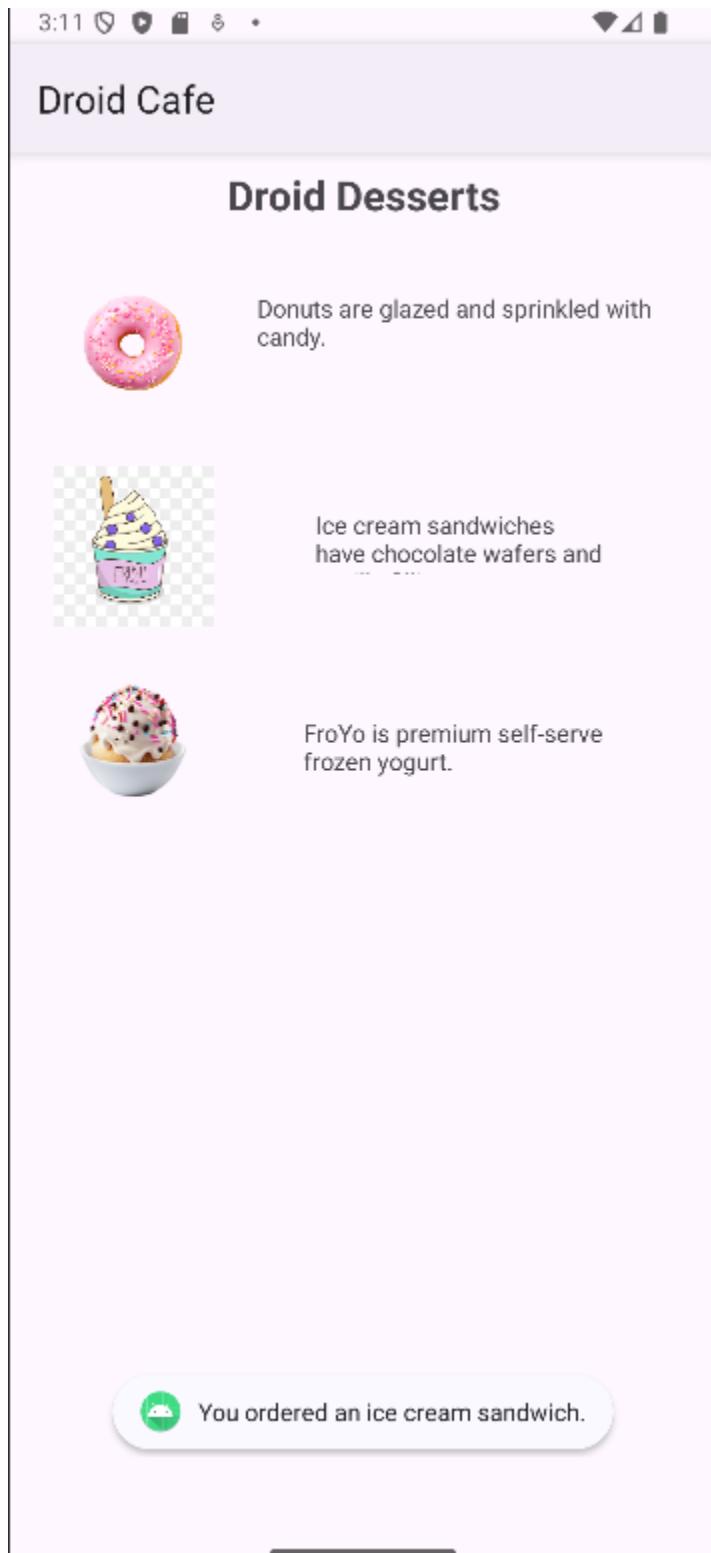
Lưu ý rằng thuộc tính android:layout\_marginStart trong mỗi ImageView được gạch chân bằng màu đỏ. Thuộc tính này xác định "lè" bắt đầu cho ImageView, mà bên trái đối với hầu hết các ngôn ngữ nhưng bên phải đối với các ngôn ngữ đọc từ phải sang trái (RTL).

5. Nhấn vào phần mở đầu android: của thuộc tính android:layout\_marginStart, và một biểu tượng đèn đỏ cảnh báo xuất hiện bên cạnh, như được hiển thị trong hình dưới đây.

6. Để làm cho ứng dụng của bạn tương thích với các phiên bản Android trước đó, hãy nhập vào bóng đèn màu đỏ cho từng trường hợp của thuộc tính này và chọn **Set layout\_marginLeft...** để đặt layout\_marginLeft thành "@dimen/margin\_wide".

7. Chạy ứng dụng.

Nhấp vào hình ảnh bánh donut, bánh sandwich kem hoặc froyo sẽ hiển thị một thông báo Toast về đơn hàng, như hiển thị trong hình bên dưới.



## Task 2 mã giải pháp

Mã giải pháp cho nhiệm vụ này được bao gồm trong mã và bộ cục của **MainActivity** trong dự án **DroidCafe** trên Android Studio.

### Task 3: Thay đổi nút cho hành động nổi

Khi bạn nhấn vào nút hành động nổi có biểu tượng email xuất hiện ở dưới cùng của màn hình, mã trong **MainActivity** sẽ hiển thị một tin nhắn ngắn trong một ngăn kéo mở ra từ dưới cùng của màn hình trên điện thoại thông minh hoặc từ góc dưới bên trái trên các thiết bị lớn hơn, sau đó tự động đóng sau vài giây. Đây được gọi là **Snackbar**. Nó được sử dụng để cung cấp phản hồi về một thao tác. Để biết thêm thông tin, hãy xem **Snackbar**.

Hãy xem cách các ứng dụng khác triển khai **nút hành động nổi (Floating Action Button - FAB)**.

Ví dụ: Ứng dụng **Gmail** cung cấp một **nút hành động nổi** để tạo một email mới. Ứng dụng **Danh bạ (Contacts)** có một **nút hành động nổi** để tạo một liên hệ mới.

Để biết thêm thông tin về **nút hành động nổi**, hãy xem **FloatingActionButton**.

Trong nhiệm vụ này, bạn sẽ thay đổi biểu tượng của **FloatingActionButton** thành,  và thay đổi hành động của **FloatingActionButton** để mở một **Activity** mới.

### 31. Thêm biểu tượng mới

Như bạn đã học trong bài học khác, bạn có thể chọn một biểu tượng từ bộ biểu tượng trong Android Studio.

Hãy làm theo các bước sau:

1. Mở rộng **res** trong bảng **Project > Android**, sau đó nhấp chuột phải (hoặc Control + nhấp) vào thư mục **drawable**.
2. Chọn **New > Image Asset**. Hộp thoại **Configure Image Asset** sẽ xuất hiện.
3. Trong menu thả xuống ở đầu hộp thoại, chọn **Action Bar and Tab Icons**. (Lưu ý rằng **action bar** chính là **app bar**.)
4. Thay đổi tên trong trường **Name** từ **ic\_action\_name** thành **ic\_shopping\_cart**.

5. Nhấp vào hình ảnh **clip art** (biểu tượng Android bên cạnh **Clipart:**) để chọn một hình ảnh clip art làm biểu tượng. Một trang chứa các biểu tượng sẽ xuất hiện. Nhấp vào biểu tượng bạn muốn sử dụng cho **Floating Action Button**, chẳng hạn như biểu tượng **giỏ hàng**.
6. Chọn **HOLO\_DARK** từ menu thả xuống **Theme**. Điều này giúp biểu tượng có màu trắng trên nền tối (hoặc đen). Nhấp vào **Next**.
7. Nhấp vào **Finish** trong hộp thoại **Confirm Icon Path**.

Mẹo: Để có mô tả đầy đủ về cách thêm biểu tượng, hãy xem **Tạo biểu tượng ứng dụng với Image Asset Studio**.

### 3.2 Thêm Activity

Như bạn đã học trong bài học trước, một **Activity** đại diện cho một màn hình trong ứng dụng, nơi người dùng có thể thực hiện một tác vụ cụ thể. Bạn đã có một activity là **MainActivity.java**. Nay giờ, bạn sẽ thêm một activity mới có tên **OrderActivity.java**.

1. Nhấp chuột phải (hoặc Control + nhấp) vào thư mục **com.example.android.droidcafe** trong cột bên trái, sau đó chọn **New > Activity > Empty Activity**.
2. Chính sửa:
  - **Activity Name** thành **OrderActivity**.
  - **Layout Name** thành **activity\_order**.
3. Giữ nguyên các tùy chọn khác và nhấp vào **Finish**.

Sau khi hoàn thành, lớp **OrderActivity** sẽ xuất hiện cùng với **MainActivity** trong thư mục **java**, và tệp **activity\_order.xml** sẽ xuất hiện trong thư mục **layout**. **Empty Activity template** đã tự động tạo các tệp này.

### 3.3 Thay đổi hành động

Trong bước này, bạn sẽ thay đổi hành động của **FloatingActionButton** để mở **Activity** mới.

1. Mở **MainActivity**.
2. Thay đổi phương thức **onClick(View view)** để tạo một **explicit intent** nhằm khởi động **OrderActivity**.

```
public void showFroyoOrder(View view) { displayToast("You ordered a FroYo."); }

1 usage
public void onClick(View v) {
 Intent intent = new Intent(packageContext: MainActivity.this, OrderActivity.class);
 intent.putExtra(EXTRA_MESSAGE, mOrderMessage);
 startActivity(intent);
}
```

3. Chạy ứng dụng. Nhấn vào **floating action button** hiện đang sử dụng biểu tượng giỏ hàng. Một **Activity** trống (**OrderActivity**) sẽ xuất hiện. Nhấn nút **Back** để quay lại **MainActivity**.

3:44 5 6 8 9



## Droid Cafe

### Droid Desserts



Donuts are glazed and sprinkled with candy.



Ice cream sandwiches have chocolate wafers and



FroYo is premium self-serve frozen yogurt.



### Task 3 Mã giải pháp

Mã giải pháp cho nhiệm vụ này được bao gồm trong mã và bộ cục của dự án Android Studio **DroidCafe**.

#### Thử thách lập trình

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

**Thử thách:** Ứng dụng DroidCafe có **MainActivity** khởi động một **Activity** thứ hai có tên **OrderActivity**.

Bạn đã học trong bài học khác cách gửi dữ liệu từ một **Activity** này sang **Activity** khác. Hãy thay đổi ứng dụng để gửi thông điệp đơn hàng cho món tráng miệng đã chọn trong **MainActivity** đến một **TextView** mới ở trên cùng của bộ cục **OrderActivity**.

1. Thêm một **TextView** ở trên cùng của bộ cục **OrderActivity** với **id** là **order\_textview**.
2. Tạo một biến thành viên (**mOrderMessage**) trong **MainActivity** cho thông điệp đơn hàng sẽ hiển thị trong **Toast**.
3. Thay đổi các bộ xử lý nhấp chuột **showDonutOrder()**, **showIceCreamOrder()**, và **showFroyoOrder()** để gán chuỗi thông điệp **mOrderMessage** trước khi hiển thị **Toast**. Ví dụ, đoạn mã sau đây gán chuỗi **donut\_order\_message** cho **mOrderMessage** và hiển thị **Toast**:

```
mOrderMessage = getString(R.string.donut_order_message);
displayToast(mOrderMessage);
```

4. Thêm một public static final String có tên **EXTRA\_MESSAGE** ở đầu **MainActivity** để định nghĩa khóa cho **intent.putExtra**:

```
2 usages
public static final String EXTRA_MESSAGE = "com.example.android.droidcafe.extra.MESSAGE";
```

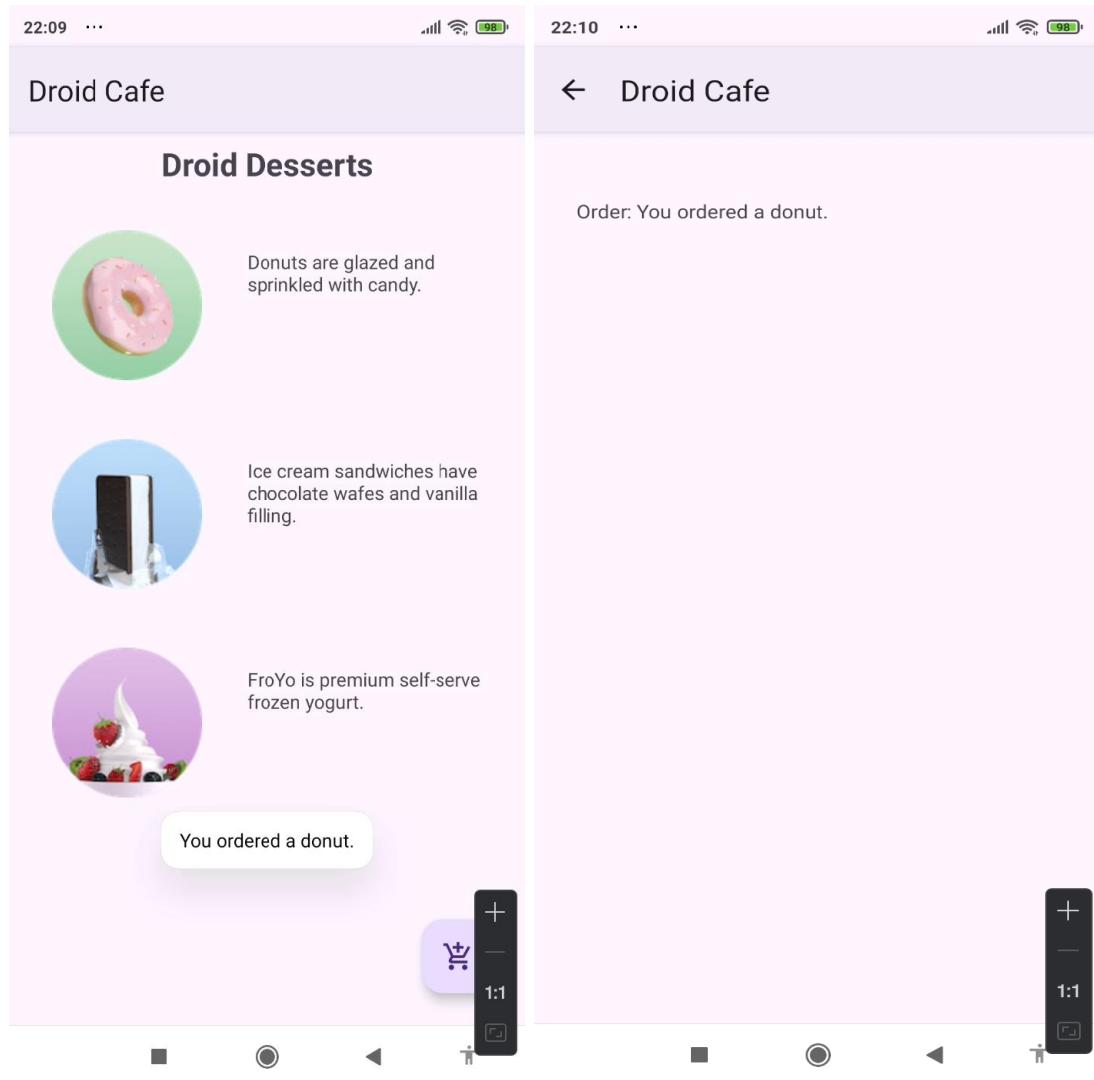
5. Thay đổi phương thức **onClick()** để bao gồm câu lệnh **intent.putExtra** trước khi khởi động **OrderActivity**:

```
public void onClick(View v) {
 Intent intent = new Intent(packageContext: MainActivity.this, OrderActivity.class);
 intent.putExtra(EXTRA_MESSAGE, mOrderMessage);
 startActivity(intent);
}
```

6. Trong OrderActivity, thêm mã sau vào phương thức onCreate() để lấy Intent khởi động Activity, trích xuất thông điệp chuỗi, và thay thế văn bản trong TextView với thông điệp:

```
Intent intent = getIntent();
String message = "Order: " + intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
TextView textView = findViewById(R.id.order_textview);
textView.setText(message);
```

7. Chạy ứng dụng. Sau khi chọn hình ảnh món tráng miệng, nhấn vào floating action button để khởi động OrderActivity, và OrderActivity sẽ hiển thị thông điệp đơn hàng như trong hình dưới đây.



## Giải pháp cho thử thách Dự án Android Studio: DroidCafeChallenge

### Tóm tắt

- Để sử dụng một hình ảnh trong dự án, sao chép hình ảnh vào thư mục **drawable** của dự án (`project_name > app > src > main > res > drawable`).
- Định nghĩa một **ImageView** để sử dụng nó bằng cách kéo thả **ImageView** vào bố cục và chọn hình ảnh cho nó.
- Thêm thuộc tính `android:onClick` để làm cho **ImageView** có thể nhấp được như một nút. Chỉ định tên của bộ xử lý nhấp chuột.
- Tạo một bộ xử lý nhấp chuột trong **Activity** để thực hiện hành động.
- Chọn biểu tượng: Mở rộng **res** trong bảng **Project > Android**, nhấp chuột phải (hoặc Control + nhấp) vào thư mục **drawable**, và chọn **New > Image Asset**. Chọn

**Action Bar and Tab Icons** trong menu thả xuống, và nhấp vào hình ảnh **clip art** (biểu tượng Android bên cạnh **Clipart:**) để chọn một hình ảnh clip art làm biểu tượng.

- Thêm một **Activity** khác: Trong bảng **Project > Android**, nhấp chuột phải (hoặc Control + nhấp) vào thư mục tên gói trong thư mục **java** và chọn **New > Activity** và một mẫu cho **Activity** (chẳng hạn như **Empty Activity**).
- Hiển thị một **Toast** message.

## **Khái niệm liên quan**

Tài liệu về khái niệm liên quan có trong **4.1: Nút và hình ảnh có thể nhấp**.

## **Tìm hiểu thêm**

Tài liệu Android Studio:

- **Hướng dẫn người dùng Android Studio**
- **Tạo biểu tượng ứng dụng với Image Asset Studio**

Tài liệu dành cho nhà phát triển Android:

- **Giao diện người dùng và điều hướng**
- **Xây dựng giao diện người dùng với Layout Editor**
- **Xây dựng giao diện người dùng đáp ứng với ConstraintLayout**
- **Layouts**
- **View**
- **Button**
- **ImageView**
- **TextView**
- **Buttons**
- **Styles and themes**

Khác:

- **Codelabs: Sử dụng ConstraintLayout để thiết kế các view Android của bạn**

## **Bài tập về nhà**

### **Thay đổi ứng dụng**

Ứng dụng **DroidCafe** hiển thị tốt khi thiết bị hoặc trình giả lập được xoay theo hướng dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang hướng ngang, các hình ảnh thứ hai và thứ ba không xuất hiện.

1. Mở (hoặc tải về) dự án ứng dụng **DroidCafe**.
2. Tạo một biến thẻ bố cục cho hướng ngang: **content\_main.xml (land)**.
3. Loại bỏ các ràng buộc từ ba hình ảnh và ba mô tả văn bản.

4. Chọn tất cả ba hình ảnh trong biến thể bố cục, và chọn **Expand Horizontally** trong nút **Pack** để phân phối đều các hình ảnh trên màn hình như hình dưới đây.
5. Ràng buộc các mô tả văn bản vào các cạnh và đáy của hình ảnh như hình dưới đây.

### Câu hỏi 1

Làm thế nào để bạn thêm hình ảnh vào một dự án Android Studio? Chọn một trong các đáp án sau:

- Kéo từng hình ảnh vào **layout editor**.
- Sao chép các tệp hình ảnh vào thư mục **drawable** của dự án.
- Kéo một **ImageButton** vào **layout editor**.
- Chọn **New > Image Asset** và sau đó chọn tệp hình ảnh.

### Câu hỏi 2

Làm thế nào để làm cho một **ImageView** có thể nhấp được như một nút đơn giản?

Chọn một trong các đáp án sau:

- Thêm thuộc tính **android:contentDescription** vào **ImageView** trong bố cục và sử dụng nó để gọi bộ xử lý nhấp chuột trong **Activity**.
- Thêm thuộc tính **android:src** vào **ImageView** trong bố cục và sử dụng nó để gọi bộ xử lý nhấp chuột trong **Activity**.
- Thêm thuộc tính **android:onClick** vào **ImageView** trong bố cục và sử dụng nó để gọi bộ xử lý nhấp chuột trong **Activity**.
- Thêm thuộc tính **android:id** vào **ImageView** trong bố cục và sử dụng nó để gọi bộ xử lý nhấp chuột trong **Activity**.

### Câu hỏi 3

Quy tắc nào áp dụng cho một bộ xử lý nhấp chuột được gọi từ thuộc tính trong bố cục? Chọn một trong các đáp án sau:

- Phương thức bộ xử lý nhấp chuột phải bao gồm **event listener View.OnClickListener**, đây là một giao diện trong lớp **View**.
- Phương thức bộ xử lý nhấp chuột phải là **public**, trả về **void**, và định nghĩa một **View** làm tham số duy nhất.
- Bộ xử lý nhấp chuột phải tùy chỉnh lớp **View.OnClickListener** và ghi đè bộ xử lý nhấp chuột của nó để thực hiện một hành động nào đó.
- Phương thức bộ xử lý nhấp chuột phải là **private** và trả về một **View**.

**Nộp ứng dụng để chấm điểm**  
**Hướng dẫn cho người chấm điểm**

1. Chạy ứng dụng.
2. Chuyển sang chế độ ngang để xem biến thể bố cục mới. Nó sẽ trông giống như hình dưới đây.

## 1.2 Các điều khiển nhập liệu

### Giới thiệu

Để cho phép người dùng nhập văn bản hoặc số, bạn sử dụng phần tử `EditText`. Một số điều khiển nhập liệu là các thuộc tính của `EditText` định nghĩa loại bàn phím sẽ hiển thị, giúp việc nhập dữ liệu trở nên dễ dàng hơn cho người dùng. Ví dụ, bạn có thể chọn phone cho thuộc tính `android:inputType` để hiển thị bàn phím số thay vì bàn phím chữ cái và số.

Các điều khiển nhập liệu khác giúp người dùng dễ dàng đưa ra lựa chọn. Ví dụ, phần tử **RadioButton** cho phép người dùng chọn một (và chỉ một) mục trong một tập hợp các mục.

Trong bài thực hành này, bạn sẽ sử dụng các thuộc tính để điều khiển giao diện bàn phím trên màn hình, và để đặt loại dữ liệu nhập vào cho `EditText`. Bạn cũng sẽ thêm các nút radio vào ứng dụng **DroidCafe** để người dùng có thể chọn một mục từ một tập hợp các mục.

### Những gì bạn đã biết

Bạn nên có khả năng:

- Tao một dự án Android Studio từ một mẫu và tạo bố cục chính.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị kết nối.
- Tạo và chỉnh sửa các phần tử giao diện người dùng (UI) bằng trình chỉnh sửa bố cục và mã XML.
- Truy cập các phần tử giao diện người dùng từ mã của bạn bằng cách sử dụng `findViewById()`.
- Chuyển văn bản trong một `View` thành một chuỗi bằng cách sử dụng `getText()`.
- Tạo một bộ xử lý nhập chuột cho nút **Button**.
- Hiển thị một thông báo **Toast**.

### Những gì bạn sẽ học

- Cách thay đổi phương thức nhập liệu để bật gợi ý, tự động viết hoa và che mật khẩu.
- Cách thay đổi bàn phím trên màn hình chung thành bàn phím điện thoại hoặc các

bàn phím chuyên dụng khác.

- Cách thêm các nút radio cho người dùng để chọn một mục trong một tập hợp các mục.
- Cách thêm một spinner để hiển thị một menu thả xuống với các giá trị, từ đó người dùng có thể chọn một giá trị.

### Những gì bạn sẽ làm

- Hiển thị bàn phím để nhập địa chỉ email.
- Hiển thị bàn phím số để nhập số điện thoại.
- Cho phép nhập văn bản nhiều dòng với tự động viết hoa câu.
- Thêm các nút radio để chọn một tùy chọn.
- Đặt một bộ xử lý **onClick** cho các nút radio.
- Thêm một spinner cho trường số điện thoại để chọn một giá trị từ một tập hợp các giá trị.

### Tổng quan về ứng dụng

Trong bài thực hành này, bạn sẽ thêm nhiều tính năng vào ứng dụng DroidCafe từ bài học về cách sử dụng hình ảnh có thể nhấp.

Trong **OrderActivity** của ứng dụng, bạn sẽ thử nghiệm với thuộc tính **android:inputType** cho các phần tử **EditText**. Bạn thêm các phần tử **EditText** để nhập tên và địa chỉ của người dùng, và sử dụng các thuộc tính để định nghĩa các phần tử một dòng và nhiều dòng có tính năng gợi ý khi bạn nhập văn bản. Bạn cũng thêm một **EditText** hiển thị bàn phím số để nhập số điện thoại.

Các loại điều khiển nhập liệu khác bao gồm các phần tử tương tác giúp người dùng đưa ra lựa chọn. Bạn thêm các nút radio vào DroidCafe để chọn chỉ một tùy chọn giao hàng từ nhiều tùy chọn. Bạn cũng cung cấp một điều khiển nhập liệu dạng spinner để chọn nhãn ( Home , Work , Other , Custom ) cho số điện thoại.

### Nhiệm vụ 1: Thủ nghiệm với các thuộc tính nhập liệu văn bản

Khi chạm vào trường văn bản có thể chỉnh sửa **EditText**, con trỏ sẽ xuất hiện trong trường văn bản và bàn phím trên màn hình sẽ tự động hiển thị để người dùng có thể nhập văn bản.

Một trường văn bản có thể chỉnh sửa kỳ vọng một loại văn bản nhập vào nhất định, chẳng hạn như văn bản thuần túy, địa chỉ e mail, số điện thoại hoặc mật khẩu. Việc chỉ định loại nhập liệu cho mỗi trường văn bản trong ứng dụng là rất quan trọng để hệ thống hiển thị phương pháp nhập liệu phù hợp, chẳng hạn như bàn phím trên màn hình cho văn bản thuần túy hoặc bàn phím số để nhập số điện thoại.

## 1.1 Thêm một **EditText** để nhập tên

Trong bước này, bạn sẽ thêm một **TextView** và một **EditText** vào bố cục **OrderActivity** trong ứng dụng **DroidCafe** để người dùng có thể nhập tên của một người.

1. Sao chép ứng dụng **DroidCafe** từ bài học về sử dụng hình ảnh có thể nhấp, và đổi tên bản sao thành **DroidCafeInput**. Nếu bạn chưa hoàn thành thử thách lập trình trong bài học đó, hãy tải dự án **DroidCafeChallenge** và đổi tên thành **DroidCafeInput**.
2. Mở tệp bố cục **activity\_order.xml**, tệp này sử dụng **ConstraintLayout**.
3. Thêm một **TextView** vào **ConstraintLayout** trong **activity\_order.xml** dưới phần tử **order\_textview** đã có trong bố cục. Sử dụng các thuộc tính sau cho **TextView** mới.
  - 4. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính **android:text** để tạo một mục mới có tên **name\_label\_text** trong **strings.xml**.
  - 5. Thêm một phần tử **EditText**. Để sử dụng trình chỉnh sửa bố cục trực quan, kéo một phần tử **Plain Text** từ bảng **Palette** vào vị trí bên cạnh **name\_label TextView**. Sau đó nhập **name\_text** cho trường **ID** và ràng buộc cạnh trái và đường chéo của phần tử với cạnh phải và đường chéo của phần tử **name\_label** như hình dưới đây.
  - 6. Hình trên làm nổi bật trường **inputType** trong bảng **Attributes** để cho thấy Android Studio đã tự động chỉ định kiểu **textPersonName**. Nhập vào trường **inputType** để xem menu các loại nhập liệu.
  - 7. Thêm một gợi ý cho việc nhập văn bản, chẳng hạn như **Enter your name**, vào trường **hint** trong bảng **Attributes**, và xóa mục **Name** trong trường **text**. Dưới dạng một gợi ý cho người dùng, văn bản "Enter your name" sẽ mờ trong **EditText**.
  - 8. Kiểm tra mã XML cho bố cục bằng cách nhấp vào tab **Text**. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính **android:hint** thành **enter\_name\_hint**. Các thuộc tính sau nên được thiết lập cho **EditText** mới (thêm thuộc tính **layout\_marginLeft** để tương thích với các phiên bản Android cũ hơn): Như bạn có thể thấy trong mã XML, thuộc tính **android:inputType** được đặt thành **textPersonName**.
  - 9. Chạy ứng dụng. Nhấn vào hình ảnh donut trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem **Activity** tiếp theo. Chạm vào trường

nhập văn bản để hiển thị bàn phím và nhập văn bản, như hình dưới đây. Lưu ý rằng các gợi ý tự động xuất hiện cho các từ bạn nhập. Nhấn vào một gợi ý để sử dụng nó. Đây là một trong các tính năng của giá trị **textPersonName** cho thuộc tính **android:inputType**. Thuộc tính **inputType** điều khiển nhiều tính năng, bao gồm cách bố trí bàn phím, việc viết hoa và việc nhập văn bản nhiều dòng.

10. Để đóng bàn phím, nhấn vào biểu tượng  trong vòng tròn màu xanh lá cây, xuất hiện ở góc dưới bên phải của bàn phím. Đây được gọi là phím **Done**.

## 1.2 Thêm một EditText nhiều dòng

Trong bước này, bạn sẽ thêm một **EditText** khác vào bố cục **OrderActivity** trong ứng dụng **DroidCafe** để người dùng có thể nhập địa chỉ bằng nhiều dòng.

1. Mở tệp bố cục `activity_order.xml` nếu nó chưa được mở.
2. Thêm một **TextView** bên dưới phần tử `name_label` đã có trong bố cục. Sử dụng các thuộc tính sau cho **TextView** mới.
3. Trích xuất giá trị thuộc tính `android:text` thành một tài nguyên chuỗi mới trong `strings.xml` với tên `address_label_text`.
4. Thêm một phần tử **EditText**.

Trong trình chỉnh sửa bố cục trực quan, kéo một phần tử **Multiline Text** từ Palette đến vị trí bên cạnh **TextView** có ID `address_label`.

Đặt `address_text` làm giá trị cho ID của **EditText**.

Thiết lập ràng buộc bên trái và đường cơ sở của nó với `address_label` như trong hình minh họa.

5. Thêm một gợi ý nhập văn bản, chẳng hạn như "Enter address", vào trường `hint` trong bảng thuộc tính `Attributes`.

Gợi ý này sẽ hiển thị dưới dạng văn bản mờ bên trong **EditText**.

6. Kiểm tra mã XML của bố cục bằng cách nhập vào tab `Text`.

Trích xuất giá trị thuộc tính android:hint thành một tài nguyên chuỗi mới có tên enter\_address\_hint.

Đảm bảo các thuộc tính sau được đặt cho EditText mới (thêm thuộc tính layout\_marginLeft để tương thích với các phiên bản Android cũ hơn).

### 7. Chạy ứng dụng.

Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào Floating Action Button để chuyển đến OrderActivity.

### 8. Nhấn vào trường nhập "Address" để hiển thị bàn phím và nhập văn bản.

Sử dụng phím Return  ở góc dưới bên phải của bàn phím (còn gọi là phím Enter hoặc New Line) để xuống dòng khi nhập văn bản.

Phím Return sẽ xuất hiện nếu bạn đặt thuộc tính android:inputType thành textMultiLine.

### 9. Để đóng bàn phím, nhấn vào nút mũi tên xuống xuất hiện thay vì nút Back trên hàng nút dưới cùng.

## 1.3 Sử dụng bàn phím số cho số điện thoại

Trong bước này, bạn sẽ thêm một EditText khác vào bộ cục OrderActivity trong ứng dụng DroidCafe để người dùng có thể nhập số điện thoại bằng bàn phím số.

1.Mở tệp activity\_order.xml nếu nó chưa được mở.

2.Thêm một TextView bên dưới phần tử address\_label đã có trong bộ cục. Sử dụng các thuộc tính sau cho TextView mới:

Lưu ý rằng TextView này được ràng buộc với đáy của EditText nhiều dòng (address\_text). Điều này là do address\_text có thể phát triển thành nhiều dòng, và TextView này sẽ xuất hiện bên dưới nó.

3.Trích xuất chuỗi tài nguyên cho giá trị thuộc tính android:text để tạo một mục nhập cho nó với tên phone\_label\_text trong tệp strings.xml.

4.Thêm một phần tử EditText. Để sử dụng trình chỉnh sửa bộ cục trực quan, kéo một phần tử Phone từ bảng Palette vào vị trí bên cạnh TextView có ID phone\_label. Sau đó, đặt ID của nó là phone\_text, và ràng buộc cạnh trái và dòng

cơ sở của phần tử này với cạnh phải và dòng cơ sở của phone\_label, như trong hình dưới đây:

5.Thêm một gợi ý nhập văn bản, chẳng hạn như Enter phone, trong trường hint trong bảng Attributes. Gợi ý "Enter phone" sẽ hiển thị dưới dạng văn bản mờ bên trong EditText.

6.Kiểm tra mã XML cho bộ cục bằng cách nhập vào tab Text. Trích xuất chuỗi tài nguyên cho giá trị thuộc tính android:hint thành enter\_phone\_hint. Các thuộc tính sau đây nên được đặt cho EditText mới (thêm thuộc tính layout\_marginLeft để tương thích với các phiên bản Android cũ hơn):

7.Chạy ứng dụng. Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem Activity tiếp theo.

8.Nhấn vào trường "Phone" để hiển thị bàn phím số. Bạn có thể nhập số điện thoại, như hình dưới đây.

9.Để đóng bàn phím, nhấn vào phím Done. 

Thử nghiệm với thuộc tính android:inputType

Hãy thay đổi giá trị thuộc tính android:inputType của một phần tử EditText để xem kết quả:

- textEmailAddress: Khi nhấn vào trường, bàn phím nhập email sẽ xuất hiện với ký hiệu @ nằm gần phím cách.
- textPassword: Các ký tự mà người dùng nhập sẽ biến thành dấu chấm để ẩn mật khẩu đã nhập.

#### 1.4 Kết hợp nhiều kiểu nhập liệu trong một EditText

Bạn có thể kết hợp các giá trị thuộc tính inputType không xung đột với nhau. Ví dụ, bạn có thể kết hợp các giá trị **textMultiLine** và **textCapSentences** để tạo một ô nhập văn bản nhiều dòng, trong đó mỗi câu bắt đầu bằng một chữ cái viết hoa.

1. Mở tệp activity\_order.xml nếu nó chưa được mở.
2. Thêm một TextView bên dưới phần tử phone\_label đã có trong bộ cục. Sử dụng các thuộc tính sau cho TextView mới.

3. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:text để tạo một mục nhập trong tệp strings.xml với tên note\_label\_text.
4. Thêm một phần tử EditText. Để sử dụng trình chỉnh sửa bô cục trực quan, kéo một phần tử "Multiline Text" từ bảng Palette đến vị trí bên cạnh phần tử note\_label TextView. Sau đó, nhập note\_text vào trường ID, và ràng buộc cạnh trái và đường cơ sở của phần tử này với cạnh phải và đường cơ sở của phần tử note\_label như bạn đã làm trước đó với các phần tử EditText khác.
5. Thêm gợi ý nhập văn bản, chẳng hạn như "Enter note", trong trường hint trong bảng thuộc tính Attributes.
6. Nhập vào bên trong trường inputType trong bảng thuộc tính Attributes. Giá trị textMultiLine đã được chọn sẵn. Ngoài ra, hãy chọn thêm textCapSentences để kết hợp hai thuộc tính này.
7. Kiểm tra mã XML của bô cục bằng cách nhấp vào tab Text. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:hint với tên enter\_note\_hint. Các thuộc tính sau đây nên được thiết lập cho phần tử EditText mới (thêm thuộc tính layout\_marginLeft để đảm bảo tương thích với các phiên bản Android cũ hơn):

<b>EditText attribute</b>	<b>Value</b>
android:id	"@+id/address_text"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	8dp

android:layout_marginLeft	8dp
android:ems	"10"
android:hint	"@string/enter_address_hint"
android:inputType	"textMultiLine"
app:layout_constraintBaseline_toBaselineOf	"@+id/address_label"
app:layout_constraintStart_toEndOf	"@+id/address_label"

Để kết hợp nhiều giá trị cho thuộc tính android:inputType, hãy nối chúng bằng ký tự dấu gạch đứng |.

8. Chạy ứng dụng. Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào nút hành động nối để xem Activity tiếp theo.

9. Nhấn vào ô nhập "Note" để nhập câu hoàn chỉnh, như hình minh họa bên dưới. Sử dụng phím Return để tạo một dòng mới hoặc chỉ cần nhập để tự động xuống dòng trong nhiều dòng.

## Task 2: Sử dụng nút radio

Các điều khiển nhập liệu là các phần tử tương tác trong giao diện người dùng (UI) của ứng dụng, cho phép người dùng nhập dữ liệu. Nút radio là một loại điều khiển nhập liệu hữu ích khi bạn muốn người dùng chỉ chọn một tùy chọn từ một tập hợp các tùy chọn.

**Mẹo:** Bạn nên sử dụng nút radio nếu bạn muốn người dùng nhìn thấy tất cả các tùy chọn có sẵn bên cạnh nhau. Nếu không cần hiển thị tất cả các tùy chọn cùng lúc, bạn có thể sử dụng **Spinner** thay thế, tính năng này được mô tả trong một chương khác.

Trong nhiệm vụ này, bạn sẽ thêm một nhóm nút radio vào ứng dụng **DroidCafeInput** để thiết lập tùy chọn giao hàng cho đơn hàng tráng miệng. Để có cái nhìn tổng quan và xem thêm mã mẫu về nút radio, hãy xem **Radio Buttons**.

### 2.1 Thêm một RadioGroup và các nút radio

Để thêm các nút radio vào **OrderActivity** trong ứng dụng **DroidCafeInput**, bạn cần tạo các phần tử **RadioButton** trong tệp bố cục **activity\_order.xml**. Sau khi chỉnh sửa tệp bố cục, bố cục cho các nút radio trong **OrderActivity** sẽ trông giống như hình minh họa dưới đây.

Vì các lựa chọn nút radio là loại lựa chọn loại trừ lẫn nhau, bạn sẽ nhóm chúng lại với nhau trong một RadioGroup. Bằng cách nhóm chúng lại, hệ thống Android đảm bảo rằng chỉ một nút radio có thể được chọn cùng lúc.

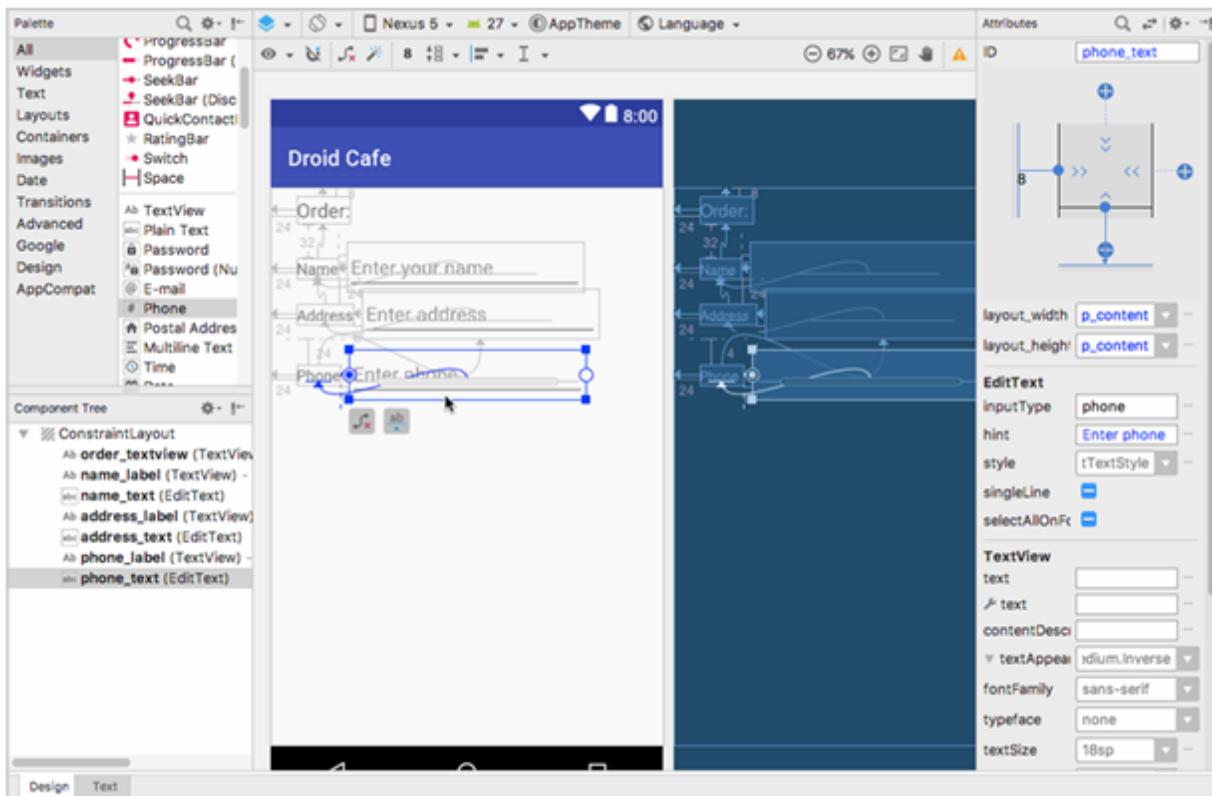
Lưu ý: Thứ tự mà bạn liệt kê các phần tử RadioButton xác định thứ tự chúng xuất hiện trên màn hình.

1. Mở **activity\_order.xml** và thêm một phần tử TextView được ràng buộc ở dưới phần tử note\_text đã có trong layout, và ràng buộc với lề trái, như trong hình dưới đây.
2. Chuyển sang chỉnh sửa XML, và đảm bảo rằng bạn đã thiết lập các thuộc tính sau cho TextView mới:

TextView attribute	Value
android:id	"@+id/phone_label"

android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	"24dp"
android:layout_marginLeft	"24dp"
android:layout_marginTop	"24dp"
android:text	"Phone"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/address_text"

- Trích xuất tài nguyên chuỗi cho "Choose a delivery method:" thành choose\_delivery\_method.
- Để thêm các nút radio, hãy bao chúng trong một RadioGroup. Thêm RadioGroup vào layout dưới TextView bạn vừa thêm, bao ba phần tử RadioButton như trong mã XML dưới đây:



Thuộc tính android:onClick với giá trị "onRadioButtonClicked" của mỗi RadioButton sẽ được gạch dưới bằng màu đỏ cho đến khi bạn thêm phương thức đó trong bước tiếp theo của tác vụ này.

- Trích xuất ba tài nguyên chuỗi cho các thuộc tính android:text thành các tên sau để các chuỗi có thể dễ dàng được dịch: same\_day\_messenger\_service, next\_day\_ground\_delivery, và pick\_up.
- Kiểm tra mã XML để biết bố cục bằng cách nhập vào tab Văn bản. Trích xuất tài nguyên chuỗi cho Android: Giá trị thuộc tính gợi ý với enter\_phone\_hint. Các thuộc tính sau phải được đặt đối với edittext mới (thêm thuộc tính bố cục marginleft để tương thích với cũ hơn phiên bản của Android):

<b>EditText attribute</b>	<b>Value</b>
android:id	"@+id/phone_text"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	8dp
android:layout_marginLeft	8dp

android:ems	"10"
android:hint	"@string/enter_phone_hint"
android:inputType	"phone"
app:layout_constraintBaseline_toBaselineOf	"@+id/phone_label"
app:layout_constraintStart_toEndOf	"@+id/phone_label"

7. Chạy ứng dụng. Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn nút hành động nổi để xem hoạt động tiếp theo.
8. Nhấn vào bên trong trường "Điện thoại" để hiển thị bàn phím số. Sau đó, bạn có thể nhập một số điện thoại, như trong hình bên dưới.

9. Để đóng bàn phím, nhấn phím đã thực hiện  .
- Để thử nghiệm với Android: Giá trị thuộc tính InputType, thay đổi phần tử EditText của phần tử

Android: InputType giá trị sau để xem kết quả:

- TextEmailAddress: Khai thác trường hiển thị bàn phím email với biểu tượng @ Biểu tượng nằm gần phím không gian.
- TextPassword: Thành ấy ký tự người dùng nhập biến thành dấu chấm để che giấu mật khẩu.

## 2.2 Thêm phương thức xử lý sự kiện nhấn nút radio

Thuộc tính **android:onClick** cho mỗi phần tử nút radio chỉ định phương thức **onRadioButtonClicked()** để xử lý sự kiện nhấn nút. Vì vậy, bạn cần thêm một phương thức **onRadioButtonClicked()** mới trong lớp **OrderActivity**.

1. Mở tệp **activity\_order.xml** (nếu chưa mở) và tìm một giá trị **onRadioButtonClicked** cho thuộc tính **android:onClick** bị gạch dưới màu đỏ.
2. Nhấp vào giá trị **onRadioButtonClicked**, sau đó nhấp vào biểu tượng cảnh báo bóng đỏ ở lề trái.
3. Chọn **Create onRadioButtonClicked(View) in OrderActivity** trong menu bóng đỏ. Android Studio sẽ tạo phương thức **onRadioButtonClicked(View view)** trong **OrderActivity**.

Ngoài ra, các giá trị **onRadioButtonClicked** cho các thuộc tính **android:onClick** khác trong **activity\_order.xml** sẽ được giải quyết và không còn bị gạch dưới nữa.

4. Để hiển thị nút radio nào đã được nhấn (tức là loại giao hàng mà người dùng chọn), hãy sử dụng một thông báo Toast. Mở **OrderActivity** và thêm phương thức **displayToast** sau:
5. Trong phương thức **onRadioButtonClicked()** mới, thêm một khối switch case để kiểm tra nút radio nào đã được chọn và gọi **displayToast()** với thông điệp phù hợp. Mã sử dụng phương thức **isChecked()** của giao diện **Checkable**, trả về **true** nếu nút đã được chọn. Nó cũng sử dụng phương thức **getId()** của View để định danh cho nút radio đã chọn.
6. Chạy ứng dụng. Nhấn vào một hình ảnh để xem hoạt động **OrderActivity**, hiển thị các lựa chọn giao hàng. Nhấn vào một lựa chọn giao hàng, và bạn sẽ thấy một thông báo Toast ở dưới màn hình với lựa chọn đó, như trong hình dưới đây.

Task 2 mã giải pháp

Dự án Android Studio: [DroidCafeInput](#)

### Thử thách lập trình

**Lưu ý:** Tất cả các thử thách lập trình đều là tùy chọn và không phải là yêu cầu trước cho các bài học sau.

**Thử thách:** Các nút radio cho lựa chọn giao hàng trong ứng dụng **DroidCafeInput** ban đầu xuất hiện không được chọn, điều này ngụ ý rằng không có lựa chọn giao hàng mặc định. Thay đổi các nút radio sao cho một trong số chúng (chẳng hạn như **nextday**) được chọn mặc định khi các nút radio lần đầu tiên xuất hiện.

**Gợi ý:** Bạn có thể hoàn thành nhiệm vụ này hoàn toàn trong tệp layout. Một phương án thay thế là bạn có thể viết mã trong OrderActivity để chọn một trong các nút radio khi Activity xuất hiện lần đầu.

Mã giải pháp thử thách

Dự án Android Studio: [DroidCafeInput](#) (xem nút radio thứ hai trong tệp layout activity\_order.xml)

### Task 3: Sử dụng Spinner cho các lựa chọn của người dùng

Một Spinner cung cấp một cách nhanh chóng để chọn một giá trị từ một tập hợp các giá trị. Chạm vào Spinner sẽ hiển thị một danh sách thả xuống với tất cả các giá trị có sẵn, từ đó người dùng có thể chọn một giá trị. Nếu bạn chỉ cung cấp từ hai hoặc ba lựa chọn, bạn có thể muốn sử dụng nút radio cho các lựa chọn đó nếu có đủ chỗ trong bố cục; tuy nhiên, với hơn ba lựa chọn, Spinner hoạt động rất tốt, cuộn khi cần thiết để hiển thị các mục và chiếm ít không gian trong bố cục của bạn.

Mẹo: Để biết thêm thông tin về Spinner, hãy tham khảo [Spinners](#).

Để cung cấp cách thức chọn nhãn cho số điện thoại (chẳng hạn như **Home**, **Work**, **Mobile**, hoặc **Other**), bạn có thể thêm một Spinner vào bố cục OrderActivity trong ứng dụng DroidCafe để nó xuất hiện ngay bên cạnh trường số điện thoại.

#### 3.1 Thêm một spinner vào bố cục

Để thêm một Spinner vào bố cục OrderActivity trong ứng dụng DroidCafe, làm theo các bước dưới đây, được đánh số trong hình dưới:

1. Mở **activity\_order.xml** và kéo **Spinner** từ bảng **Palette** vào bố cục.
2. Hạn chế phần trên của phần tử Spinner với dưới address\_text, phần bên phải với phần bên phải của bố cục, và phần bên trái với phone\_text.

Để căn chỉnh Spinner và phone\_text theo chiều ngang, sử dụng nút pack trong thanh công cụ, cung cấp các tùy chọn để đóng gói hoặc mở rộng các phần tử UI đã chọn.

Chọn cả Spinner và phone\_text trong **Component Tree**, nhấp vào nút pack, và chọn **Expand Horizontally**. Kết quả là, cả hai phần tử Spinner và phone\_text sẽ có chiều rộng cố định.

3. Trong bảng Attributes, thiết lập **ID** của Spinner là **label\_spinner**, và thiết lập các khoảng cách trên và bên phải là **24**, và khoảng cách bên trái là **8**. Chọn **match\_constraint** cho menu thả xuổng **layout\_width**, và **wrap\_content** cho menu thả xuổng **layout\_height**.

Bố cục sẽ trông giống như hình dưới đây. Menu thả xuổng **layout\_width** của phần tử **phone\_text** trong bảng Attributes được thiết lập là **134dp**. Bạn có thể thử nghiệm với các cài đặt chiều rộng khác nếu muốn.

Để xem mã XML của **activity\_order.xml**, nhấp vào tab **Text**.

Spinner phải có các thuộc tính sau: Hãy chắc chắn thêm các thuộc tính **android:layout\_marginRight** và **android:layout\_marginLeft** như trong đoạn mã trên để duy trì tính tương thích với các phiên bản Android cũ hơn.

Phần tử **phone\_text** hiện sẽ có các thuộc tính sau (sau khi sử dụng công cụ pack):

### 3.2 Thêm mã để kích hoạt Spinner và bộ lắng nghe của nó

Các lựa chọn cho **Spinner** là các chuỗi tĩnh được xác định rõ, chẳng hạn như "Home" và "Work", vì vậy bạn có thể sử dụng một mảng văn bản được định nghĩa trong **strings.xml** để lưu trữ các giá trị đó.

Để kích hoạt Spinner và trình nghe sự kiện của nó, hãy triển khai giao diện **AdapterView.OnItemSelectedListener**, giao diện này yêu cầu bạn cũng phải thêm các phương thức gọi lại **onItemSelected()** và **onNothingSelected()**.

1. Mở **strings.xml** và định nghĩa các giá trị có thể chọn (**Home**, **Work**, **Mobile** và **Other**) cho Spinner dưới dạng một mảng chuỗi **labels\_array**.
2. Để định nghĩa callback xử lý lựa chọn cho Spinner, hãy thay đổi lớp **OrderActivity** để triển khai giao diện **AdapterView.OnItemSelectedListener**, như được minh họa:

Khi bạn nhập **AdapterView** trong câu lệnh trên, Android Studio sẽ tự động nhập tiện ích **AdapterView**. Lý do bạn cần **AdapterView** là vì bạn cần một adapter—cụ thể là **ArrayAdapter**—để gán mảng vào Spinner. Một adapter giúp kết nối dữ liệu của bạn—trong trường hợp này là mảng các mục của Spinner—with Spinner. Bạn sẽ tìm hiểu thêm về mô hình sử dụng adapter để kết nối dữ liệu trong một bài thực hành khác. Dòng này sẽ xuất hiện trong khôi import của bạn:

Khi nhập **OnItemSelectedListener** trong câu lệnh trên, hãy đợi vài giây để một biểu tượng bóng đèn màu đỏ xuất hiện ở lề bên trái.

3. Nhấp vào biểu tượng bóng đèn và chọn **Implement methods**. Các phương thức **onItemSelected()** và **onNothingSelected()**, vốn là bắt buộc đối với **OnItemSelectedListener**, sẽ được làm nổi bật, và tùy chọn **Insert @Override** sẽ được chọn sẵn. Nhấp vào **OK**.

Bước này sẽ tự động thêm các phương thức callback onItemSelected() và onNothingSelected() trống vào cuối lớp OrderActivity. Cả hai phương thức này đều sử dụng tham số AdapterView<?>\*\*. Dấu \*\*<?> là một ký hiệu đại diện (wildcard) của Java, giúp phương thức có thể linh hoạt chấp nhận bất kỳ loại AdapterView nào làm đối số.

4. Khởi tạo một Spinner trong phương thức onCreate() bằng cách sử dụng phần tử label\_spinner trong layout và đặt trình nghe sự kiện của nó bằng spinner.setOnItemSelectedListener trong onCreate(), như trong đoạn mã sau:
5. Tiếp tục chỉnh sửa phương thức onCreate(), thêm một câu lệnh để tạo ArrayAdapter với mảng chuỗi (labels\_array) bằng cách sử dụng layout Spinner được cung cấp sẵn bởi Android cho từng mục (layout.simple\_spinner\_item):

Layout simple\_spinner\_item được sử dụng trong bước này và layout simple\_spinner\_dropdown\_item được sử dụng trong bước tiếp theo là các layout mặc định do Android cung cấp trong lớp R.layout. Bạn nên sử dụng các layout này trừ khi bạn muốn tự định nghĩa layout riêng cho các mục trong Spinner và giao diện của nó.

6. Xác định layout cho các lựa chọn của Spinner là simple\_spinner\_dropdown\_item, sau đó áp dụng adapter vào Spinner.

### 3.3 Thêm một mã để phản hồi khi chọn mục trong Spinner

Khi người dùng chọn một mục trong Spinner, Spinner sẽ nhận được sự kiện on-item-selected.

Để xử lý sự kiện này, bạn đã triển khai giao diện AdapterView.OnItemSelectedListener ở bước trước, đồng thời thêm các phương thức callback onItemSelected() và onNothingSelected() trống.

Trong bước này, bạn sẽ điền mã vào phương thức onItemSelected() để lấy mục đã chọn trong Spinner bằng cách sử dụng getItemAtPosition(), sau đó gán mục đó vào biến spinnerLabel.

1.Thêm mã vào phương thức callback onItemSelected() trống, như minh họa bên dưới, để lấy mục đã chọn của người dùng bằng getItemAtPosition() và gán nó vào spinnerLabel. Bạn cũng có thể gọi phương thức displayToast() mà bạn đã thêm vào OrderActivity:

Không cần thêm mã vào phương thức callback onNothingSelected() trong ví dụ này.

2.Chạy ứng dụng.

Spinner sẽ xuất hiện bên cạnh trường nhập số điện thoại và hiển thị lựa chọn đầu tiên (**Home**). Khi nhấn vào Spinner, tất cả các lựa chọn sẽ xuất hiện, như trong hình bên trái. Khi chọn một mục trong Spinner, một thông báo Toast sẽ hiển thị với lựa chọn đó, như trong hình bên phải.

## Task 3 Mã giải pháp

Dự án Android Studio: [DroidCafeInput](#)

## Thử thách lập trình 2

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

**Thử thách:** Viết mã để thực hiện một hành động trực tiếp từ bàn phím bằng cách nhấn phím **Send**, chẳng hạn như để quay số điện thoại:



Trong hình trên:

1.Nhập số điện thoại vào trường **EditText**.

2.Nhấn phím **Send** để mở trình quay số điện thoại. Trình quay số sẽ xuất hiện ở phía bên phải của hình minh họa.

Hướng dẫn thực hiện thử thách, tạo một dự án ứng dụng mới, thêm một **EditText** và thiết lập thuộc tính **android:inputType** thành **phone**. Sử dụng thuộc tính **android:imeOptions** cho phần tử **EditText** với giá trị **actionSend**:

Người dùng có thể nhấn phím **Send** để quay số điện thoại, như trong hình minh họa.

Trong phương thức **onCreate()** của **Activity**, sử dụng **setOnEditorActionListener()** để thiết lập trình nghe sự kiện cho **EditText** nhằm phát hiện khi phím được nhấn:

Để biết cách thiết lập trình nghe sự kiện, hãy xem phần **Chỉ định loại phương thức nhập (Specify the input method type)**.

Bước tiếp theo là ghi đè phương thức **onEditorAction()** và sử dụng hằng số **IME\_ACTION\_SEND** trong lớp **EditorInfo** để phản hồi khi phím được nhấn. Trong ví dụ bên dưới, phím này được sử dụng để gọi phương thức **dialNumber()** nhằm quay số điện thoại:

Cuối cùng, tạo phương thức **dialNumber()**, sử dụng **implicit intent** với **ACTION\_DIAL** để chuyển số điện thoại sang một ứng dụng khác có thể quay số. Nó sẽ trông như sau:

### Thử thách 2 mã giải pháp

Dự án Android Studio: [KeyboarDialPhone](#)

### Tóm tắt

Các giá trị thuộc tính **android:inputType** ảnh hưởng đến giao diện bàn phím trên màn hình:

- **textAutoCorrect:** Gợi ý sửa lỗi chính tả.

- **textCapSentences:** Viết hoa chữ cái đầu tiên của mỗi câu mới.
- **textPersonName:** Hiển thị một dòng văn bản với gợi ý khi nhập và nút **Done** để hoàn tất.
- **textMultiLine:** Cho phép nhập nhiều dòng văn bản và có phím **Return** để xuống dòng.
- **textPassword:** Ẩn mật khẩu khi nhập.
- **textEmailAddress:** Hiển thị bàn phím nhập email thay vì bàn phím thông thường.
- **phone:** Hiển thị bàn phím số thay vì bàn phím thông thường.

Bạn thiết lập giá trị cho thuộc tính **android:inputType** trong tệp layout XML cho phần tử **EditText**.

Để kết hợp nhiều giá trị, hãy sử dụng ký tự gạch đứng ( | ).

**RadioButton** là điều khiển đầu vào hữu ích để chọn một tùy chọn duy nhất trong một tập hợp tùy chọn:

- Nhóm các phần tử **RadioButton** bên trong **RadioGroup** để đảm bảo chỉ một **RadioButton** được chọn cùng một lúc.
- Thứ tự liệt kê các **RadioButton** trong nhóm xác định thứ tự hiển thị trên màn hình.
- Sử dụng thuộc tính **android:onClick** cho từng **RadioButton** để chỉ định trình xử lý sự kiện khi nhấn.
- Để kiểm tra xem một nút có được chọn không, sử dụng phương thức **isChecked()** của giao diện **Checkable**, trả về **true** nếu nút được chọn.

### Một **Spinner** (Menu thả xuống)

- Thêm **Spinner** vào layout.
- Sử dụng **ArrayAdapter** để gán một mảng văn bản làm các mục trong menu **Spinner**.
- Triển khai giao diện **AdapterView.OnItemSelectedListener**, trong đó yêu cầu thêm các phương thức callback **onItemSelected()** và **onNothingSelected()** để kích hoạt **Spinner** và trình nghe sự kiện của nó.
- Sử dụng phương thức **onItemSelected()** để lấy mục được chọn trong **Spinner** bằng **getItemAtPosition()**.

### Khái niệm liên quan

Tài liệu về khái niệm liên quan có trong [\*\*4.2: Input controls\*\*](#).

### Tìm hiểu thêm

Tài liệu hướng dẫn Android Studio:

- [\*\*Android Studio User Guide\*\*](#)

Tài liệu dành cho nhà phát triển Android:

- [Input events overview](#)
- [Specify the input method type](#)
- [Styles and themes](#)
- [Radio Buttons](#)
- [Spinners](#)

- View
- Button
- Edittext
- Android:inputType
- TextView
- RadioGroup
- Checkbox
- SeekBar
- ToggleButton
- Spinner

Bài tập về nhà

## Xây dựng và chạy một ứng dụng

1. Tạo một ứng dụng có năm Checkbox và một nút **Show Toast**, như hình minh họa.
2. Nếu người dùng chọn một Checkbox và nhấn **Show Toast**, hiển thị một thông báo Toast hiển thị nội dung của Checkbox đã chọn.
3. Nếu người dùng chọn nhiều hơn một Checkbox và nhấn **Show Toast**, hiển thị một Toast bao gồm nội dung của tất cả các Checkbox được chọn, như trong hình minh họa.

### Trả lời các câu hỏi

#### Câu hỏi 1

**Sự khác biệt quan trọng nhất giữa checkbox và một nhóm radio button (RadioGroup)?**

Chọn một:

- "Sự khác biệt chính là checkbox cho phép chọn nhiều mục, trong khi RadioGroup chỉ cho phép chọn một mục."

#### Câu hỏi 2

**Nhóm layout nào cho phép căn chỉnh các phần tử CheckBox theo chiều dọc?**

Chọn một:

- **LinearLayout**

#### Câu hỏi 3

**Phương thức nào của giao diện Checkable dùng để kiểm tra trạng thái của radio button (tức là kiểm tra xem nó đã được chọn hay chưa)?**

Chọn một:

- **isChecked()**

---

## Hướng dẫn chấm điểm ứng dụng

Kiểm tra xem ứng dụng có các tính năng sau không:

- **Bộ cục chứa năm CheckBox được căn chỉnh theo chiều dọc trên màn hình, cùng với một nút Show Toast.**
- **Phương thức onSubmit() xác định checkbox nào được chọn bằng cách sử dụng findViewById() kết hợp với isChecked().**
- **Các chuỗi mô tả topping được nối lại thành một thông báo Toast.**

## 1.3 Menu và bộ chọn

Giới thiệu

**Thanh ứng dụng (App Bar)**, còn được gọi là **thanh hành động (Action Bar)**, là một khung gian chuyên dụng nằm ở phía trên cùng của mỗi màn hình **Activity**. Khi bạn tạo một **Activity** từ mẫu **Basic Activity**, Android Studio sẽ tự động bao gồm một thanh ứng dụng.

**Menu tùy chọn (Options Menu)** trong thanh ứng dụng thường cung cấp các lựa chọn điều hướng, chẳng hạn như chuyển đến một **Activity** khác trong ứng dụng. Menu cũng có thể cung cấp các tùy chọn ảnh hưởng đến cách sử dụng ứng dụng, ví dụ như thay đổi cài đặt hoặc thông tin hồ sơ, thường được thực hiện trong một **Activity** riêng biệt.

Trong bài thực hành này, bạn sẽ tìm hiểu về cách thiết lập **App Bar** và **Options Menu** trong ứng dụng của mình, như minh họa trong hình dưới đây.

Trong hình trên:

1. **Thanh ứng dụng (App bar)**: Thanh ứng dụng bao gồm tiêu đề ứng dụng, menu tùy chọn và nút tràn (overflow button).
2. **Biểu tượng hành động của menu tùy chọn(Options menu action icons)**: Hai mục menu tùy chọn đầu tiên xuất hiện dưới dạng biểu tượng trong thanh ứng dụng.
3. **Nút tràn (Overflow button)**: Nút tràn (ba dấu chấm dọc) mở một menu hiển thị các mục menu tùy chọn bổ sung.
4. **Menu tràn của các mục tùy chọn(Options overflow menu)**: Sau khi nhấp vào nút tràn, các mục menu tùy chọn bổ sung sẽ xuất hiện trong menu tràn.

Các mục menu tùy chọn xuất hiện trong menu tràn (xem hình trên). Tuy nhiên, bạn có thể đặt một số mục dưới dạng biểu tượng — càng nhiều càng tốt — trong thanh ứng dụng. Việc sử dụng thanh ứng dụng cho menu tùy chọn giúp ứng dụng của bạn đồng nhất với các ứng dụng Android khác, cho phép người dùng dễ dàng hiểu cách sử dụng ứng dụng và có trải nghiệm tuyệt vời.

**Mẹo:** Để cung cấp trải nghiệm người dùng quen thuộc và đồng nhất, hãy sử dụng **Menu APIs** để trình bày các hành động của người dùng và các tùy chọn khác trong các activity của bạn. Xem [\*\*Menus\*\*](#) để biết chi tiết.

Bạn cũng sẽ tạo một ứng dụng hiển thị một **dialog** yêu cầu người dùng chọn lựa, ví dụ như một thông báo yêu cầu người dùng nhấn **OK** hoặc **Cancel**. Một **dialog** là một cửa sổ xuất hiện trên màn hình hoặc chiếm toàn bộ màn hình, làm gián đoạn dòng chảy hoạt động. Android cung cấp các **dialog** sẵn có, gọi là **pickers**, để chọn thời gian hoặc ngày. Bạn có thể sử dụng chúng để đảm bảo người dùng chọn đúng thời gian hoặc ngày được định dạng chính xác và điều chỉnh theo giờ và ngày địa phương của người dùng. Trong bài học này, bạn cũng sẽ tạo một ứng dụng với **date picker**.

## Những gì bạn nên biết trước

Bạn nên có khả năng:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các phần tử giao diện người dùng (UI) bằng trình chỉnh sửa layout.
- Chỉnh sửa mã layout XML và truy cập các phần tử từ mã Java của bạn.
- Thêm trình xử lý sự kiện click cho **Button**.

## Những gì bạn sẽ học

- Cách thêm các mục menu vào menu tùy chọn.
- Cách thêm biểu tượng cho các mục trong menu tùy chọn.
- Cách hiển thị các mục menu trong thanh ứng dụng.
- Cách thêm trình xử lý sự kiện cho các mục menu.
- Cách thêm một **dialog** để hiển thị thông báo.
- Cách thêm **date picker**.

## Những gì bạn sẽ làm

- Tiếp tục thêm các tính năng cho dự án Droid Cafe từ bài thực hành trước.
- Thêm các mục menu vào menu tùy chọn.
- Thêm biểu tượng cho các mục menu để xuất hiện trong thanh ứng dụng.
- Kết nối các sự kiện click vào các trình xử lý sự kiện để xử lý sự kiện nhấn.
- Sử dụng **alert dialog** để yêu cầu người dùng chọn lựa.
- Sử dụng **date picker** để nhập ngày.

## Tổng quan về ứng dụng

Trong bài thực hành trước, bạn đã tạo một ứng dụng có tên **Droid Cafe**, như hình dưới đây, sử dụng mẫu **Basic Activity**. Mẫu này cũng cung cấp một menu tùy chọn cơ bản trong thanh ứng dụng ở phía trên màn hình.

Trong bài tập này, bạn sẽ sử dụng thư viện hỗ trợ **v7 appcompat** với **Toolbar** như một thanh ứng dụng, hoạt động trên nhiều thiết bị và cũng cung cấp cho bạn không gian để tùy chỉnh thanh

ứng dụng sau này khi ứng dụng của bạn phát triển. Để đọc thêm về các cân nhắc thiết kế khi sử dụng thanh ứng dụng, hãy xem **Responsive layout grid** trong thông số kỹ thuật Material Design.

Bạn sẽ tạo một ứng dụng mới hiển thị **alert dialog**. **Dialog** này gián đoạn quy trình làm việc của người dùng và yêu cầu người dùng đưa ra lựa chọn.

Bạn cũng sẽ tạo một ứng dụng cung cấp một **Button** để hiển thị **date picker**, và chuyển ngày đã chọn thành một chuỗi để hiển thị trong thông báo **Toast**.

### Task 1: Thêm các mục vào menu tùy chọn

Trong nhiệm vụ này, bạn sẽ mở dự án **DroidCafeInput** từ bài thực hành trước và thêm các mục vào menu tùy chọn trong thanh ứng dụng ở phía trên màn hình.

#### 1.1 Kiểm tra mã nguồn

Mở ứng dụng **DroidCafeInput** từ bài thực hành về việc sử dụng các điều khiển đầu vào và kiểm tra các tệp layout sau trong thư mục **res > layout**:

- **activity\_main.xml**: Layout chính cho **MainActivity**, màn hình đầu tiên mà người dùng thấy.
- **content\_main.xml**: Layout cho nội dung của màn hình **MainActivity**, được bao gồm trong **activity\_main.xml**.
- **activity\_order.xml**: Layout cho **OrderActivity**, được thêm vào trong bài thực hành về việc sử dụng các điều khiển đầu vào.

#### Các bước thực hiện:

1. Mở **content\_main.xml** và nhấp vào tab **Text** để xem mã XML. Thuộc tính **app:layout\_behavior** của **ConstraintLayout** được thiết lập là **@string/appbar\_scrolling\_view\_behavior**, điều khiển cách màn hình cuộn liên quan đến thanh ứng dụng ở phía trên. (Chuỗi tài nguyên này được định nghĩa trong tệp **values.xml** đã được tạo, và bạn không nên chỉnh sửa nó.)

Để biết thêm về hành vi cuộn, xem **Android Design Support Library** trong blog của Android Developers. Để biết về các thực hành thiết kế có liên quan đến menu cuộn, xem **Scrolling in the Material Design specification**.

2. Mở **activity\_main.xml** và nhấp vào tab **Text** để xem mã XML cho layout chính, sử dụng layout **CoordinatorLayout** với layout **AppBarLayout** nhúng bên trong. Các thẻ **CoordinatorLayout** và **AppBarLayout** yêu cầu tên đầy đủ xác định **android.support.design**, là thư viện Hỗ trợ Thiết kế của Android. **AppBarLayout** giống như một **LinearLayout** theo chiều dọc. Nó sử dụng lớp **Toolbar** trong thư viện hỗ trợ, thay vì **ActionBar** gốc, để triển khai thanh ứng dụng. **Toolbar** trong layout này có id là **toolbar**, và cũng được chỉ định, giống như **AppBarLayout**, với tên đầy đủ (**android.support.v7.widget**).

Thanh ứng dụng là một phần của màn hình hiển thị, có thể hiển thị tiêu đề activity, điều hướng và các mục tương tác khác. **ActionBar** gốc hoạt động khác nhau tùy thuộc vào

phiên bản Android chạy trên thiết bị. Vì lý do này, nếu bạn đang thêm menu tùy chọn, bạn nên sử dụng thư viện hỗ trợ **v7 appcompat** với **Toolbar** làm thanh ứng dụng. Việc sử dụng **Toolbar** giúp bạn dễ dàng thiết lập một thanh ứng dụng hoạt động trên nhiều thiết bị và cũng cung cấp không gian để tùy chỉnh thanh ứng dụng của bạn sau này khi ứng dụng phát triển. **Toolbar** bao gồm các tính năng mới nhất và hoạt động cho bất kỳ thiết bị nào có thể sử dụng thư viện hỗ trợ.

Layout **activity\_main.xml** cũng sử dụng câu lệnh layout **include** để bao gồm toàn bộ layout được định nghĩa trong **content\_main.xml**. Việc tách biệt các định nghĩa layout giúp bạn dễ dàng thay đổi nội dung của layout riêng biệt với định nghĩa thanh công cụ và layout điều phối.

3. Chạy ứng dụng. Chú ý đến thanh ở phía trên màn hình hiển thị tên ứng dụng (Droid Cafe). Nó cũng hiển thị nút tràn hành động (ba dấu chấm dọc) ở phía bên phải. Nhấp vào nút tràn để xem menu tùy chọn, lúc này chỉ có một mục menu là **Settings**.
4. Kiểm tra tệp **AndroidManifest.xml**. **Activity MainActivity** được thiết lập sử dụng theme **NoActionBar**. Theme này được định nghĩa trong tệp **styles.xml** (mở **app > res > values > styles.xml** để xem). Các style được trình bày trong một bài học khác, nhưng bạn có thể thấy rằng theme **NoActionBar** thiết lập thuộc tính **windowActionBar** là **false** (không có thanh công cụ cửa sổ) và **windowNoTitle** là **true** (không có tiêu đề). Các giá trị này được thiết lập vì bạn đang định nghĩa thanh ứng dụng với **AppBarLayout**, thay vì sử dụng **ActionBar**. Việc sử dụng một trong các theme **NoActionBar** ngăn không cho ứng dụng sử dụng lớp **ActionBar** gốc để cung cấp thanh ứng dụng.
5. Nhìn vào **MainActivity**, kế thừa từ **AppCompatActivity** và bắt đầu với phương thức **onCreate()**, trong đó thiết lập view nội dung là layout **activity\_main.xml** và thiết lập **toolbar** là **Toolbar** được định nghĩa trong layout. Sau đó, nó gọi **setSupportActionBar()** và truyền **toolbar** vào, thiết lập **toolbar** là thanh ứng dụng cho Activity.

Để biết thêm về các phương pháp hay khi thêm thanh ứng dụng vào ứng dụng của bạn, hãy tham khảo [Add the app bar.](#)

## 1.4 Điều hướng người dùng

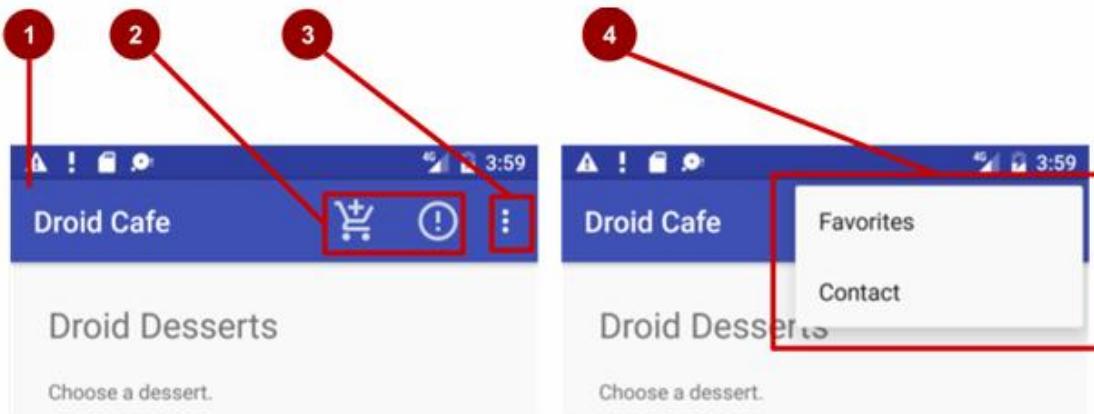
### Giới thiệu

Thanh ứng dụng (còn được gọi là Thanh hành động) là một không gian chuyên dụng ở đầu mỗi màn hình hoạt động. Khi bạn tạo một hoạt động từ mẫu hoạt động cơ bản, Android Studio bao gồm một thanh ứng dụng.

Menu Tùy chọn trong thanh ứng dụng thường cung cấp các lựa chọn để điều hướng, chẳng hạn như điều hướng đến một hoạt động khác trong ứng dụng. Menu cũng có thể cung cấp các lựa chọn ảnh hưởng đến

việc sử dụng chính ứng dụng, ví dụ như các cách để thay đổi cài đặt hoặc thông tin hồ sơ, thường xảy ra trong một hoạt động riêng biệt.

Trong thực tế này, bạn tìm hiểu về việc thiết lập menu Tùy chọn ứng dụng và tùy chọn trong ứng dụng của bạn, như trong hình bên dưới.



Trong hình trên:

1. Thanh ứng dụng. Thanh ứng dụng bao gồm tiêu đề ứng dụng, menu tùy chọn và nút tràn.
2. Tùy chọn biểu tượng hành động menu. Hai mục menu Tùy chọn đầu tiên xuất hiện dưới dạng biểu tượng trong ứng dụng
3. Nút tràn. Nút tràn (ba chấm dọc) mở ra một menu hiển thị nhiều hơn Tùy chọn các mục menu.
4. Tùy chọn menu tràn. Sau khi nhấp vào nút tràn, các mục menu tùy chọn khác xuất hiện trong menu tràn.

Các mục menu Tùy chọn xuất hiện trong menu Tùy chọn Overflow (xem hình trên). Tuy nhiên, bạn có thể đặt một số mục dưới dạng biểu tượng, với nhiều thứ như có thể phù hợp với thanh trong thanh ứng dụng. Sử dụng thanh ứng dụng cho menu Tùy chọn làm cho ứng dụng của bạn phù hợp với các ứng dụng Android khác, cho phép người dùng nhanh chóng hiểu cách vận hành ứng dụng của bạn và có trải nghiệm tuyệt vời.

Bạn cũng tạo một ứng dụng hiển thị hộp thoại để yêu cầu sự lựa chọn của người dùng, chẳng hạn như cảnh báo yêu cầu người dùng nhấn OK hoặc hủy. Hộp thoại là một cửa sổ xuất hiện trên đỉnh của màn hình hoặc lấp đầy màn hình, làm gián đoạn luồng hoạt động. Android cung cấp các hộp thoại sẵn sàng sử dụng, được gọi là người chọn, để chọn thời gian hoặc ngày. Bạn có thể sử dụng chúng để đảm bảo rằng người dùng của bạn chọn thời gian hoặc ngày hợp lệ được định dạng chính xác và điều chỉnh theo giờ và ngày của người dùng. Trong bài học này, bạn cũng sẽ tạo một ứng dụng với trình chọn ngày.

## Những gì bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy các ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các thành phần UI bằng trình soạn thảo bố cục.
- Chỉnh sửa mã bố cục XML và truy cập các yếu tố từ mã Java của bạn.
- Thêm một trình xử lý nhấp vào nút.

### Những gì bạn sẽ học

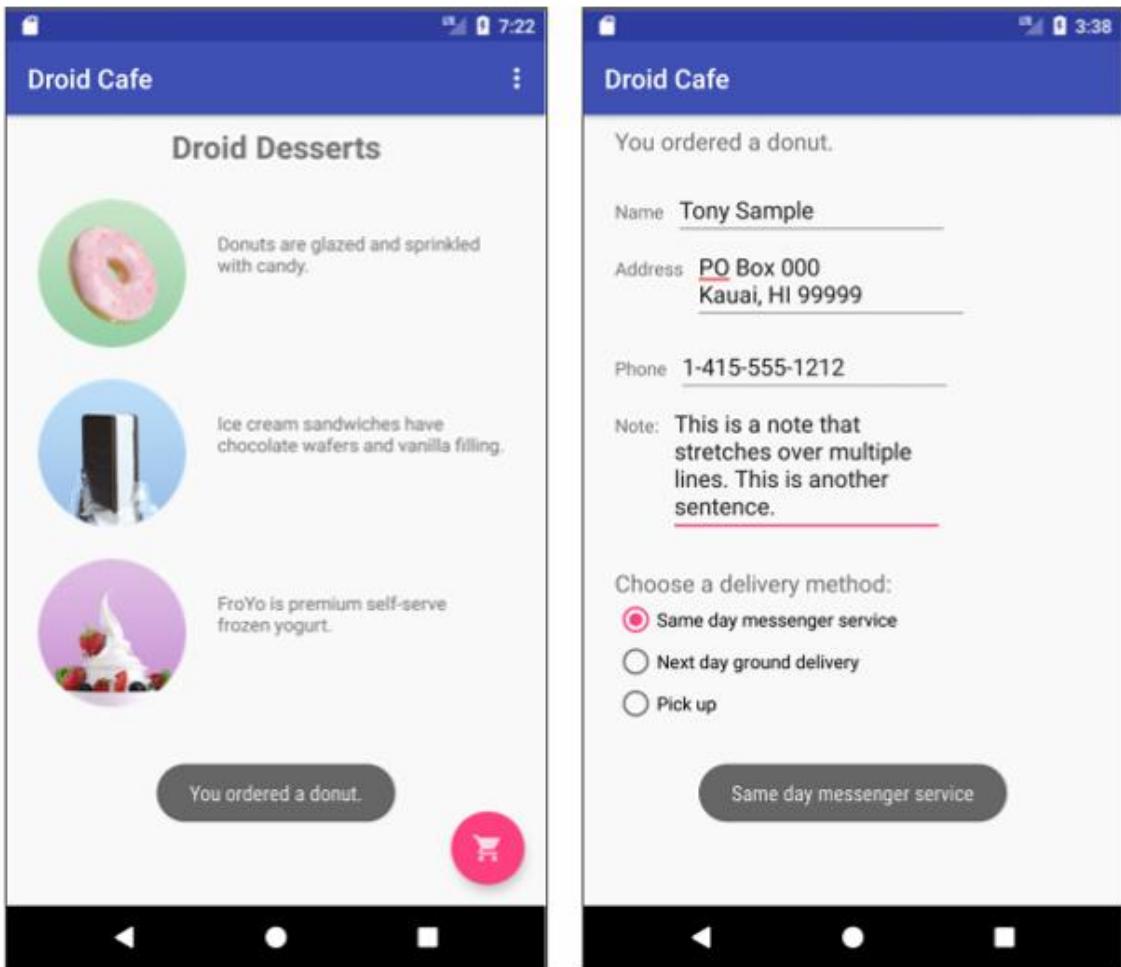
- Cách thêm các mục menu vào menu Tùy chọn.
- Cách thêm biểu tượng cho các mục trong menu Tùy chọn.
- Cách đặt các mục menu để hiển thị trong thanh ứng dụng.
- Cách thêm trình xử lý nhấp cho các mục menu.
- Cách thêm hộp thoại cho cảnh báo.
- Cách thêm trình chọn ngày.

### Những gì bạn sẽ làm

- Tiếp tục thêm các tính năng vào dự án Droid Cafe từ thực tế trước đó.
- Thêm các mục menu vào menu Tùy chọn.
- Thêm biểu tượng cho các mục menu xuất hiện trong thanh ứng dụng.
- Kết nối các mục Menu-item với trình xử lý sự kiện xử lý các sự kiện nhấp chuột.
- Sử dụng hộp thoại cảnh báo để yêu cầu sự lựa chọn của người dùng.
- Sử dụng trình chọn ngày cho đầu vào ngày.

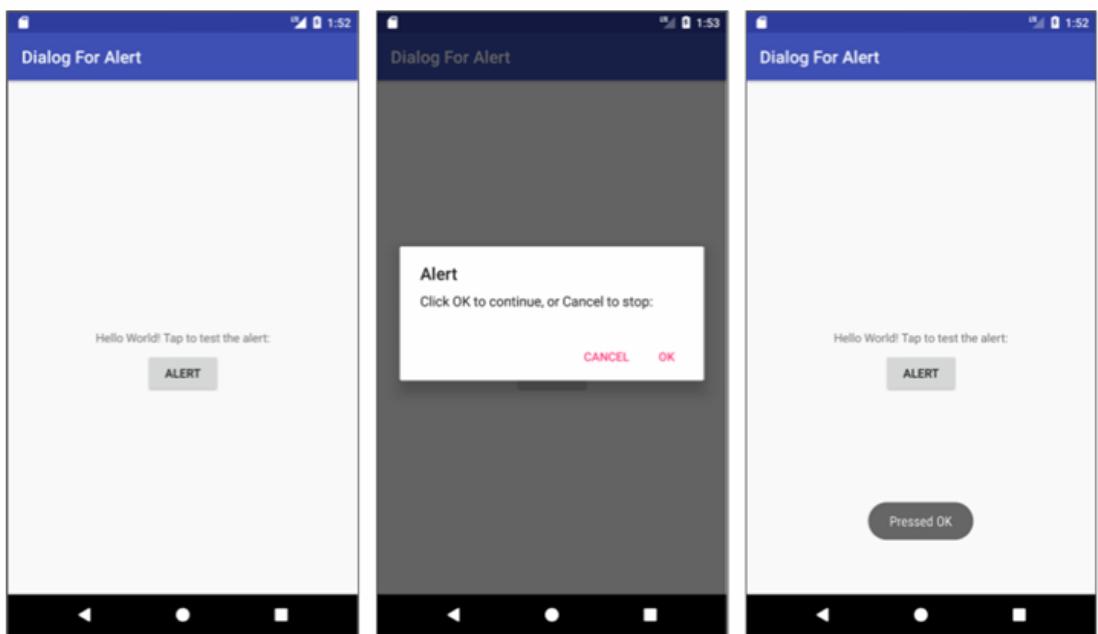
### Tổng quan về ứng dụng

Trong thực tế trước đây, bạn đã tạo một ứng dụng có tên Droid Cafe, được hiển thị trong hình bên dưới, sử dụng mẫu hoạt động cơ bản. Mẫu này cũng cung cấp menu Tùy chọn bộ xương trong thanh ứng dụng ở đầu màn hình.

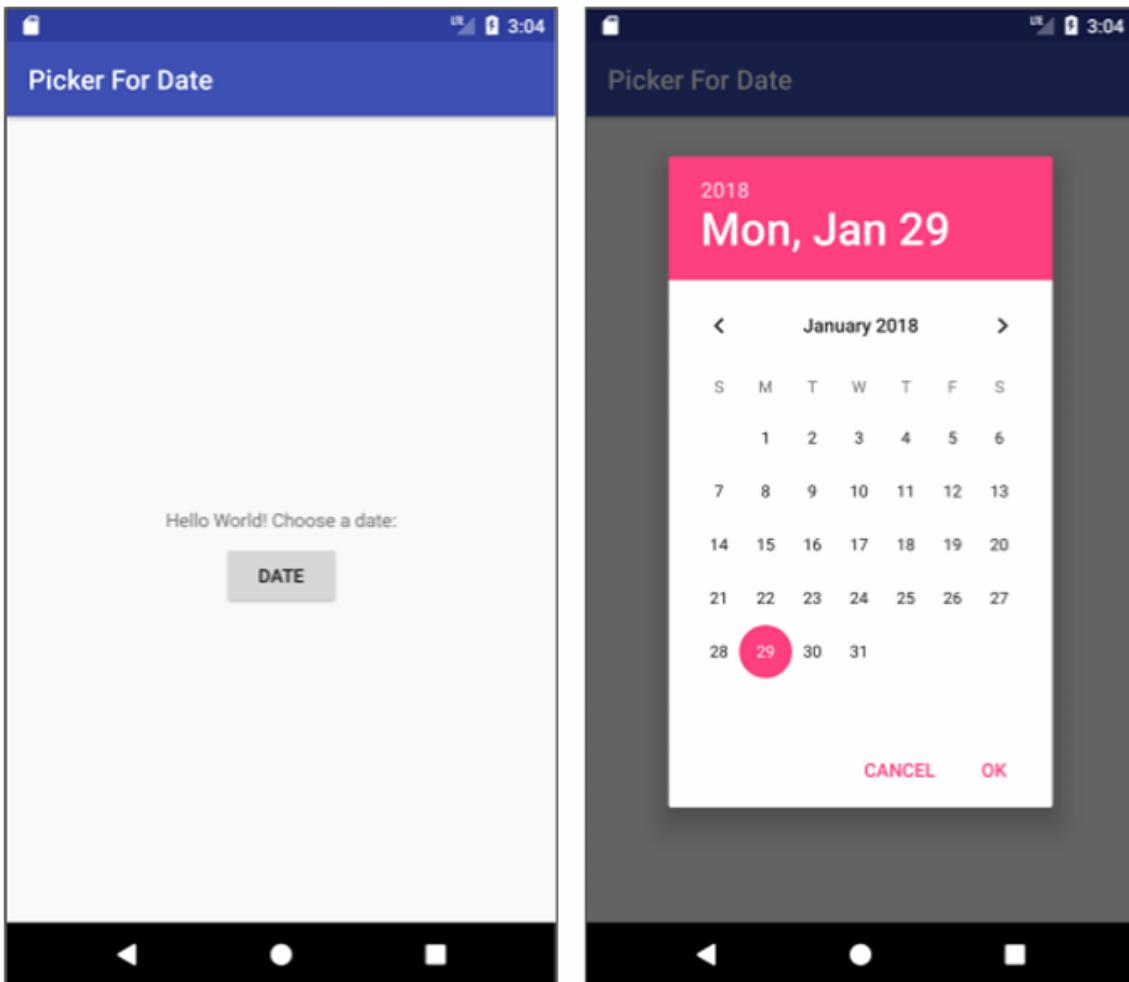


Đối với bài tập này, bạn đang sử dụng thanh công cụ Thư viện hỗ trợ V7 AppCompat làm thanh ứng dụng, hoạt động trên phạm vi rộng nhất của thiết bị và cũng cung cấp cho bạn chỗ để tùy chỉnh thanh ứng dụng của bạn sau này khi ứng dụng của bạn phát triển. Để đọc thêm về các cân nhắc thiết kế cho việc sử dụng thanh ứng dụng, hãy xem Lưới bố trí đáp ứng trong đặc tả thiết kế vật liệu.

Bạn tạo một ứng dụng mới hiển thị hộp thoại cảnh báo. Hộp thoại làm gián đoạn quy trình làm việc của người dùng và yêu cầu người dùng đưa ra lựa chọn.



Bạn cũng tạo một ứng dụng cung cấp một nút để hiển thị trình chọn ngày và chuyển đổi ngày đã chọn thành một chuỗi để hiển thị trong tin nhắn bánh mì nướng



## Nhiệm vụ 1: Thêm các mục vào menu Tùy chọn

Trong nhiệm vụ này, bạn mở dự án droidcafeInput từ các mục thực tế trước đó và thêm các mục menu vào menu Tùy chọn trong thanh ứng dụng ở đầu màn hình.

### 1.1 Kiểm tra mã

Mở ứng dụng droidcafeInput từ thực tế về việc sử dụng các điều khiển đầu vào và kiểm tra các điều khiển sau

Tệp bố cục trong thư mục res> Bố cục:

- Hoạt động\_main.xml: Bố cục chính cho MainActivity, màn hình đầu tiên mà người dùng nhìn thấy.
- Content\_main.xml: Bố cục cho nội dung của màn hình chính, mà (như bạn sẽ thấy trong thời gian ngắn) được bao gồm trong hoạt động\_main.xml.
- Activity\_order.xml: Bố cục cho thứ tự, mà bạn đã thêm vào thực tế trên

Sử dụng điều khiển đầu vào.

Thực hiện theo các bước sau:

1. Mở nội dung\_main.xml và nhấp vào tab Văn bản để xem mã XML. Các ứng dụng: Layout\_behavior cho ràng buộcLayout được đặt thành @String/appbar\_scrolling\_view\_behavior, điều khiển cách cuộn màn hình liên quan đến thanh ứng dụng ở trên cùng. (Tài nguyên chuỗi này được xác định trong một tệp được tạo có tên là value.xml, mà bạn không nên chỉnh sửa.)

Để biết thêm về hành vi cuộn, hãy xem Thư viện hỗ trợ thiết kế Android trong blog Android Developers. Đối với các thực hành thiết kế liên quan đến các menu cuộn, hãy xem cuộn trong tài liệu

Đặc điểm kỹ thuật thiết kế.

2. Mở hoạt động\_main.xml và nhấp vào tab văn bản để xem mã XML cho bố cục chính, trong đó sử dụng bố cục điều phối viên với bố cục appbarlayout nhúng. Điều phối viênLayout và thẻ appbarlayout yêu cầu tên đủ điều kiện chỉ định android.support.design, là thư viện hỗ trợ thiết kế Android.

AppBarLayout giống như một tuyến tính dọc. Nó sử dụng lớp thanh công cụ trong thư viện hỗ trợ, thay vì Actionbar gốc, để triển khai thanh ứng dụng. Thanh công cụ trong bố cục này có thanh công cụ ID và cũng được chỉ định, như AppBarLayout, với tên đủ điều kiện (Android.Support.v7.widget).

Thanh ứng dụng là một phần ở đầu màn hình có thể hiển thị tiêu đề hoạt động, điều hướng, và các mục tương tác khác. Thanh hành động gốc hoạt động khác nhau tùy thuộc vào phiên bản Android đang chạy trên thiết bị. Vì lý do này, nếu bạn đang thêm menu Tùy chọn, bạn nên sử dụng thanh công cụ Thư viện hỗ trợ V7 AppCompat làm thanh ứng dụng. Sử dụng

Thanh công cụ giúp bạn dễ dàng thiết lập một thanh ứng dụng hoạt động trên phạm vi rộng nhất của thiết bị và cung cấp cho bạn chỗ để tùy chỉnh thanh ứng dụng của bạn sau này khi ứng dụng của bạn phát triển. Thanh công cụ bao gồm các tính năng gần đây nhất và hoạt động cho bất kỳ thiết bị nào có thể sử dụng thư viện hỗ trợ.

Bố cục Activity\_main.xml cũng sử dụng câu lệnh Bố cục bao gồm để bao gồm toàn bộ bố cục được xác định trong Content\_Main.xml. Sự phân tách các định nghĩa bố cục này giúp dễ dàng hơn thay đổi nội dung bố cục khác với định nghĩa thanh công cụ và bố cục điều phối viên của bố cục. Đây là một thực tiễn tốt nhất để tách nội dung của bạn (có thể cần được dịch)

định dạng của bố cục của bạn.

3.Chạy ứng dụng. Lưu ý thanh ở đầu màn hình hiển thị tên của ứng dụng (droidcafe). Nó cũng hiển thị nút tràn hành động (ba chấm dọc) ở phía bên phải. Nhấn vào nút tràn để xem menu Tùy chọn, tại thời điểm này chỉ có một tùy chọn menu, Cài đặt.



#### 4. Kiểm tra tệp androidmanifest.xml. Hoạt động .MainActivity được đặt để sử dụng

Chủ đề NOActionBar. Chủ đề này được xác định trong tệp Styles.xml (Mở ứng dụng> Res> Value> Styles.xml để xem nó). Các kiểu được đề cập trong một bài học khác, nhưng bạn có thể thấy rằng chủ đề NoActionBar đặt thuộc tính WindowActionBar thành FALSE (không có thanh ứng dụng cửa sổ) và thuộc tính windownotitle thành true (không có tiêu đề). Các giá trị này được đặt bởi vì bạn đang xác định thanh ứng dụng với AppBarLayout, thay vì sử dụng thanh action. Sử dụng một trong các chủ đề NOActionBar ngăn ứng dụng sử dụng lớp ActionBar gốc để cung cấp thanh ứng dụng.

- Nhìn vào MainActivity, giúp mở rộng ứng dụng và bắt đầu bằng onCreate () Phương thức, đặt chế độ xem nội dung thành bố cục Activity\_main.xml và đặt thanh công cụ thành thanh công cụ được xác định trong bố cục. Sau đó, nó gọi SetSupportActionBar () và chuyển thanh công cụ cho nó, đặt thanh công cụ làm thanh ứng dụng cho hoạt động.

Để biết thực tiễn tốt nhất về việc thêm thanh ứng dụng vào ứng dụng của bạn, hãy xem thêm thanh ứng dụng.

## 1.2 Thêm các mục menu khác vào menu Tùy chọn

Bạn sẽ thêm các mục menu sau vào menu Tùy chọn:

- Đặt hàng: Điều hướng đến thứ tự để xem thứ tự trang miệng.
- Trạng thái: Kiểm tra trạng thái của một đơn đặt hàng.
- Yêu thích: Hiển thị món tráng miệng yêu thích.
- Liên hệ: Liên hệ với quán cà phê. Bởi vì bạn không cần mục Cài đặt hiện có, bạn sẽ thay đổi cài đặt để liên hệ.

Android cung cấp định dạng XML tiêu chuẩn để xác định các mục menu. Thay vì xây dựng một menu trong mã hoạt động của bạn, bạn có thể xác định một menu và tất cả các mục menu của nó trong tài nguyên menu XML. Sau đó, bạn có thể thổi phồng tài nguyên menu (tái nó làm đối tượng menu) trong hoạt động của bạn:

Mở rộng Res > menu trong ngăn dự án > Android và Mở menu\_main.xml. Duy nhất mục menu được cung cấp từ mẫu là action\_sinstall (lựa chọn cài đặt), đó là được định nghĩa là:

```
<item
 android:id="@+id/action_settings"
 android:orderInCategory="100"
 android:title="@string/action_settings"
 app:showAsAction="never" />
```

Thay đổi các thuộc tính sau của mục action\_sinstall để biến nó thành mục action\_contact (không thay đổi thuộc tính Android hiện tại: OrderInCategory):

Attribute	Value
android:id	"@+id/action_contact"

android:title	"Contact"
app:showAsAction	"never"

3. Trích xuất chuỗi "Liên hệ" được mã hóa cứng vào chuỗi tài nguyên action\_contact.

4. Thêm mục menu mới bằng cách sử dụng thẻ <item> trong khái <sences> và cung cấp cho mục các thuộc tính sau:

<b>Attribute</b>	<b>Value</b>
android:id	"@+id/action_order"
android:orderInCategory	"10"
android:title	"Order"
app:showAsAction	"never"

Android: Thuộc tính OrderInCategory Chỉ định thứ tự trong đó các mục menu xuất hiện trong menu, với số thấp nhất xuất hiện cao hơn trong menu. Mục liên hệ được đặt thành 100, là một con số lớn để chỉ định rằng nó hiển thị ở phía dưới thay vì trên cùng. Bạn đặt mục đặt hàng thành 10, đưa nó lên trên liên hệ và để lại nhiều chỗ trong menu để có thêm các mục.

5. Trích xuất chuỗi "thứ tự" được mã hóa cứng vào chuỗi tài nguyên action\_order.

6. Thêm hai mục menu nữa theo cùng một cách với các thuộc tính sau:

<b>Status item attribute</b>	<b>Value</b>
android:id	"@+id/action_status"
android:orderInCategory	"20"

android:title	"Status"
app:showAsAction	"never"

Favorites item attribute	Value
android:id	"@+id/action_favorites"
android:orderInCategory	"30"
android:title	"Favorites"
app:showAsAction	"never"

7. Trích xuất "Status" vào tài nguyên action\_status và "Favourite" vào tài nguyên action\_favorites.

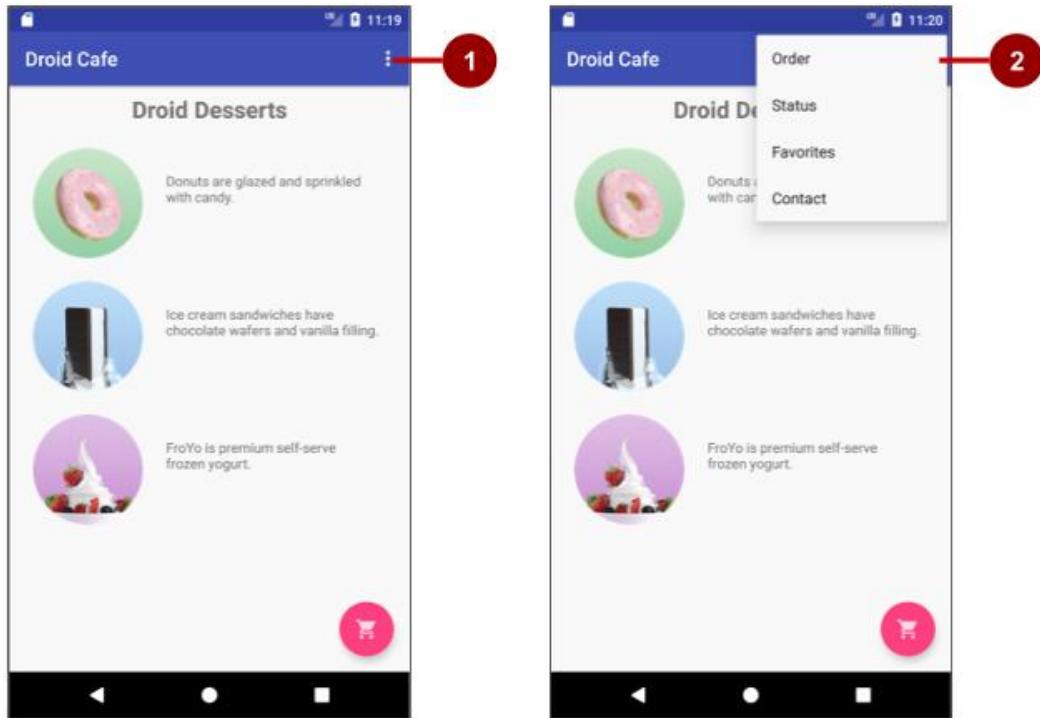
8. Bạn sẽ hiển thị thông báo bánh mì nướng với thông báo hành động tùy thuộc vào mục menu mà người dùng chọn. Mở chuỗi.xml và thêm các tên và giá trị chuỗi sau cho các thông báo sau:

```
<string name="action_order_message">You selected Order.</string>
<string name="action_status_message">You selected Status.</string>
<string name="action_favorites_message">You selected Favorites.</string>
<string name="action_contact_message">You selected Contact.</string>
```

9. Mở MainActivity và thay đổi câu lệnh IF trong phương thức  
OnOptionsItemSelected()

```
if (id == R.id.action_order)
```

Chạy ứng dụng và nhấn vào biểu tượng tràn hành động, được hiển thị ở phía bên trái của hình bên dưới, để xem menu Tùy chọn, được hiển thị ở phía bên phải của hình bên dưới. Bạn sẽ sớm thêm các cuộc gọi lại để trả lời các mục được chọn từ menu này.



Trong hình trên:

1. Nhấn vào biểu tượng tròn trong thanh ứng dụng để xem menu Tùy chọn.
2. Menu Tùy chọn giảm xuống từ thanh ứng dụng.

Lưu ý thứ tự các mục trong menu tùy chọn. Bạn đã sử dụng thuộc tính Android: `OrderInCategory` để chỉ định mức độ ưu tiên của các mục menu trong menu: Mục đặt hàng là 10, tiếp theo là trạng thái (20) và mục yêu thích (30) và liên hệ là cuối cùng (100). Bảng sau đây cho thấy mức độ ưu tiên của các mục trong menu:

Menu item	<code>orderInCategory</code> attribute
-----------	----------------------------------------

Order	10
Status	20
Favorites	30
Contact	100

## Nhiệm vụ 2: Thêm biểu tượng cho các mục menu

Bất cứ khi nào có thể, bạn muốn hiển thị các hành động được sử dụng thường xuyên nhất bằng cách sử dụng các biểu tượng trong thanh ứng dụng để người dùng có thể nhấp vào chúng mà không phải nhấp vào biểu tượng Overflow. Trong nhiệm vụ này, bạn thêm các biểu tượng cho một số mục menu và hiển thị một số mục menu trong thanh ứng dụng ở đầu màn hình dưới dạng biểu tượng.

Trong ví dụ này, giả sử rằng các hành động thứ tự và trạng thái được sử dụng thường xuyên nhất. Các Hành động yêu thích đôi khi được sử dụng, và liên hệ là ít được sử dụng nhất. Bạn có thể đặt các biểu tượng cho các hành động này và chỉ định những điều sau:

- Đặt hàng và trạng thái phải luôn được hiển thị trong thanh ứng dụng.
- Yêu thích nên được hiển thị trong thanh ứng dụng nếu nó phù hợp; Nếu không, nó sẽ xuất hiện trong menu tràn.
- Liên hệ không nên xuất hiện trong thanh ứng dụng; Nó chỉ nên xuất hiện trong menu tràn.

### 2.1 Thêm biểu tượng cho các mục menu

Để chỉ định các biểu tượng cho các hành động, trước tiên bạn cần thêm các biểu tượng làm tài sản hình ảnh vào thư mục có thể vẽ bằng cách sử dụng quy trình tương tự bạn đã sử dụng trong thực tế sử dụng hình ảnh có thể nhấp. Bạn muốn sử dụng các biểu tượng sau (hoặc các biểu tượng tương tự):

-  **Order**: Sử dụng cùng một biểu tượng bạn đã thêm cho nút hành động nổi trong thực tế bằng cách sử dụng các hình ảnh có thể nhấp (IC\_SHOPPING\_CART.PNG). Khóa học cơ bản của nhà phát triển Android (V2) - Đơn vị 2
-  **Status**:
-  **Favorites**:
- Contact :Không cần biểu tượng vì nó sẽ chỉ xuất hiện trong menu tràn.

Đối với các biểu tượng trạng thái và yêu thích, hãy làm theo các bước sau:

1. Chọn **res** trong bảng **Project > Android** pane, và kích chuột phải)\ vào thư mục **drawable** .
2. Chọn **New > Image Asset**. Hộp thoại cấu hình tài sản hình ảnh xuất hiện.
3. Chọn Action Bar and Tab Itemstrong menu thả xuống.
4. Thay đổi **ic\_action\_name** sang tên khác (chẳng hạn như **IC\_STATUS\_INFO** cho biểu tượng trạng thái).
5. Nhấp vào hình ảnh clip art (logo Android bên cạnh clipart :) để chọn hình ảnh clip art làm biểu tượng. Một trang của các biểu tượng xuất hiện. Nhấp vào biểu tượng bạn muốn sử dụng.
6. Chọn **holo\_dark** từ menu thả xuống **Theme**. Điều này đặt biểu tượng là màu trắng trên nền màu tối (hoặc đen). Nhấp vào **tiếp theo** và sau đó nhấp vào **Kết thúc**.

## 2.2 Hiển thị các mục menu dưới dạng biểu tượng trong thanh ứng dụng

Để hiển thị các mục menu dưới dạng biểu tượng trong thanh ứng dụng, hãy sử dụng thuộc tính ứng dụng: `showasaction` trong menu\_main.xml.

Các giá trị sau cho thuộc tính chỉ định xem hành động có xuất hiện trong thanh ứng dụng dưới dạng biểu tượng hay không:

- "always": Luôn xuất hiện trong thanh ứng dụng. (Nếu không có đủ chỗ, nó có thể trùng với các biểu tượng menu khác.)
- "ifroom": xuất hiện trong thanh ứng dụng nếu có phòng.
- "Never": Không bao giờ xuất hiện trong thanh ứng dụng; Văn bản của nó xuất hiện trong menu tràn.

Thực hiện theo các bước này để hiển thị một số mục menu dưới dạng biểu tượng:

1. Mở menu\_main.xml một lần nữa và thêm các thuộc tính sau vào **Order**, **Status** và **Favourites** sao cho hai mục đầu tiên (thứ tự và trạng thái) luôn xuất hiện và mục yêu thích chỉ xuất hiện nếu có chỗ cho nó:

Order item attribute	Old value	New value
android:icon	"none"	"@drawable/ic_shopping_cart"
app:showAsAction	"never"	"always"

Status item attribute	Old value	New value
android:icon	"none"	"@drawable/ic_status_info"
app:showAsAction	"never"	"always"

Favorites item attribute	Old value	New value
android:icon	"none"	"@drawable/ic_favorite"
app:showAsAction	"never"	"ifRoom"

Chạy ứng dụng. Bây giờ bạn sẽ thấy ít nhất hai biểu tượng trong thanh ứng dụng:

**Order** và **Status** như được hiển thị ở phía bên trái của hình bên dưới.

3. Xoay thiết bị của bạn theo hướng ngang hoặc nếu bạn đang chạy trong trình giả lập, hãy nhấp các biểu tượng **Rotate Left** or các biểu tượng t để xoay màn hình vào hướng ngang. Sau đó, bạn sẽ thấy tất cả ba biểu tượng trong thanh ứng dụng để **Order** , **Status** và **Favourite** như được hiển thị ở phía bên phải của hình bên dưới.



Có bao nhiêu nút hành động sẽ phù hợp với thanh ứng dụng? Nó phụ thuộc vào định hướng và kích thước của màn hình thiết bị. Ít nút hơn xuất hiện theo hướng thẳng đứng, như được hiển thị ở phía bên trái của

hình trên, so với hướng ngang như thể hiện ở phía bên phải của hình trên. Các nút hành động có thể không chiếm hơn một nửa chiều rộng thanh ứng dụng chính.

## Nhiệm vụ 3: Xử lý mục menu đã chọn

Trong tác vụ này, bạn thêm một phương thức để hiển thị thông báo về mục menu nào được khai thác và sử dụng phương thức OnOptionsItemSelected () để xác định mục menu nào đã được khai thác.

### 3.1 Tạo một phương thức để hiển thị lựa chọn menu

1. Mở **MainActivity**

2. Nếu bạn chưa thêm phương thức sau (trong một bài học khác) để hiển thị bánh mì nướng tin nhắn, thêm nó ngay bây giờ. Bạn sẽ sử dụng nó làm hành động để thực hiện cho mỗi lựa chọn menu. (Thông thường bạn sẽ thực hiện một hành động cho từng mục menu như bắt đầu một hoạt động khác, như được hiển thị sau trong bài học này.)

```
public void displayToast(String message) {
 Toast.makeText(getApplicationContext(), message,
 Toast.LENGTH_SHORT).show();
}
```

### 3.2 Sử dụng trình xử lý sự kiện được chọn lọc

Phương thức **OnOptionsItemSelected ()** xử lý các lựa chọn từ menu Tùy chọn. Bạn sẽ thêm một khối trường hợp chuyển đổi để xác định mục menu nào đã được chọn và hành động nào sẽ thực hiện.

1. Tìm phương thức **OnOptionsItemSelected ()** được cung cấp bởi mẫu. Phương pháp xác định xem một mục menu nhất định có được nhấp không, bằng cách sử dụng ID mục menu. Trong ví dụ dưới đây, ID là **action\_order**:

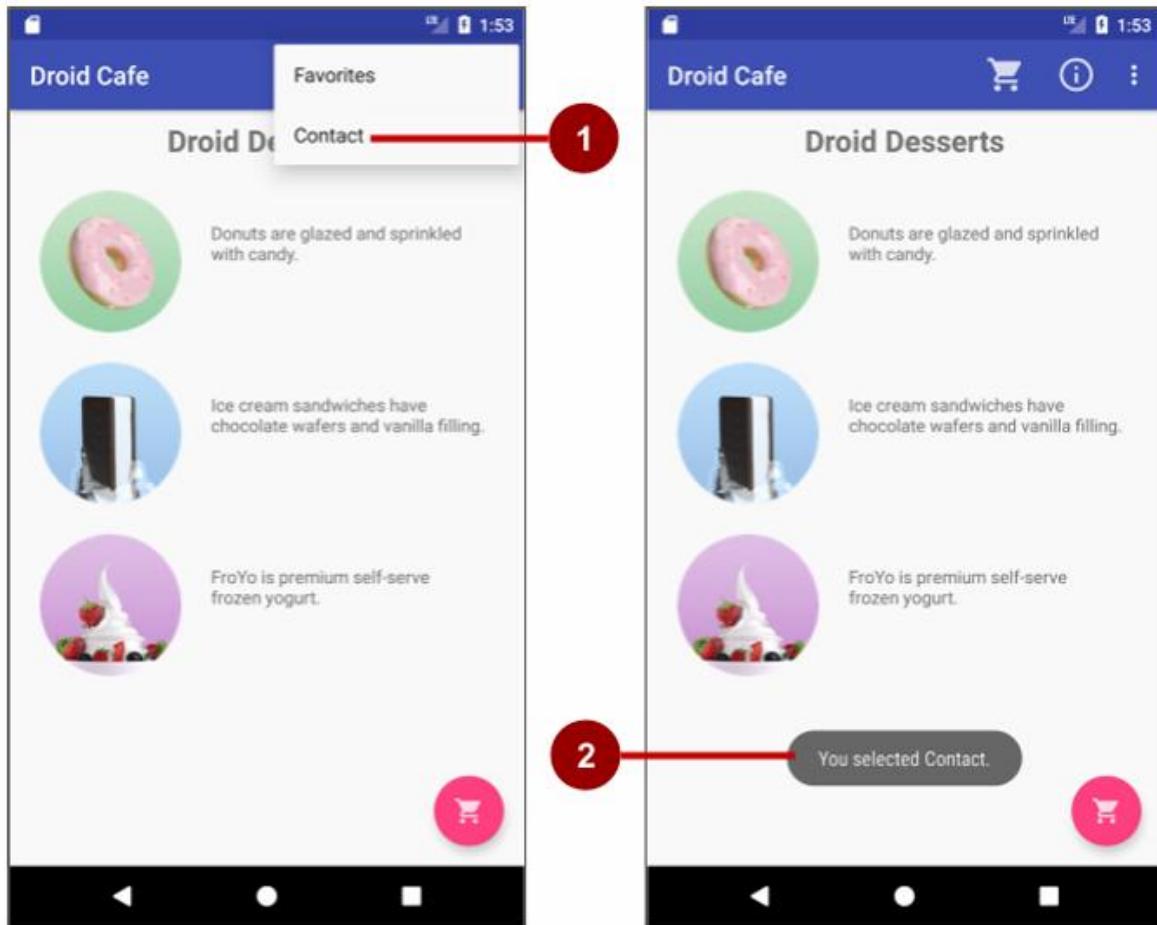
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
 int id = item.getItemId();
 if (id == R.id.action_order) {
 return true;
 }
 return super.onOptionsItemSelected(item);
}
```

1.Thay thế câu lệnh gán ID ID và câu lệnh IF bằng khối trường hợp chuyển đổi sau, đặt thông báo phù hợp dựa trên id id id id id:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
 switch (item.getItemId()) {
 case R.id.action_order:
 showToast(getString(R.string.action_order_message));
 return true;
 case R.id.action_status:
 showToast(getString(R.string.action_status_message));
 return true;
 case R.id.action_favorites:
 showToast(getString(R.string.action_favorites_message));
 return true;
 case R.id.action_contact:

 showToast(getString(R.string.action_contact_message));
 return true;
 default:
 // Do nothing
 }
 return super.onOptionsItemSelected(item);
}
```

2.Chạy ứng dụng. Bây giờ bạn sẽ thấy một thông báo bánh mì nướng khác trên màn hình, như được hiển thị ở phía bên phải của hình bên dưới, dựa trên mục menu bạn chọn.



Trong hình trên:

1. Chọn mục liên hệ trong menu Tùy chọn.
2. Thông điệp bánh mì nướng xuất hiện.

### 3.3 Bắt đầu một hoạt động từ một mục menu

Thông thường bạn sẽ thực hiện một hành động cho từng mục menu, chẳng hạn như bắt đầu một hoạt động khác Giới thiệu đoạn trích từ nhiệm vụ trước đó, thay đổi mã cho trường hợp Action\_order sang cách sau, bắt đầu thứ tự (sử dụng cùng một mã bạn đã sử dụng cho nút hành động nổi trong bài học về bảng cách sử dụng hình ảnh có thể nháp):

```
switch (item.getItemId()) {
 case R.id.action_order:
 Intent intent = new Intent(MainActivity.this, OrderActivity.class);
 intent.putExtra(EXTRA_MESSAGE, mOrderMessage);
 startActivity(intent);
 return true;
 // ... code for other cases
}
```

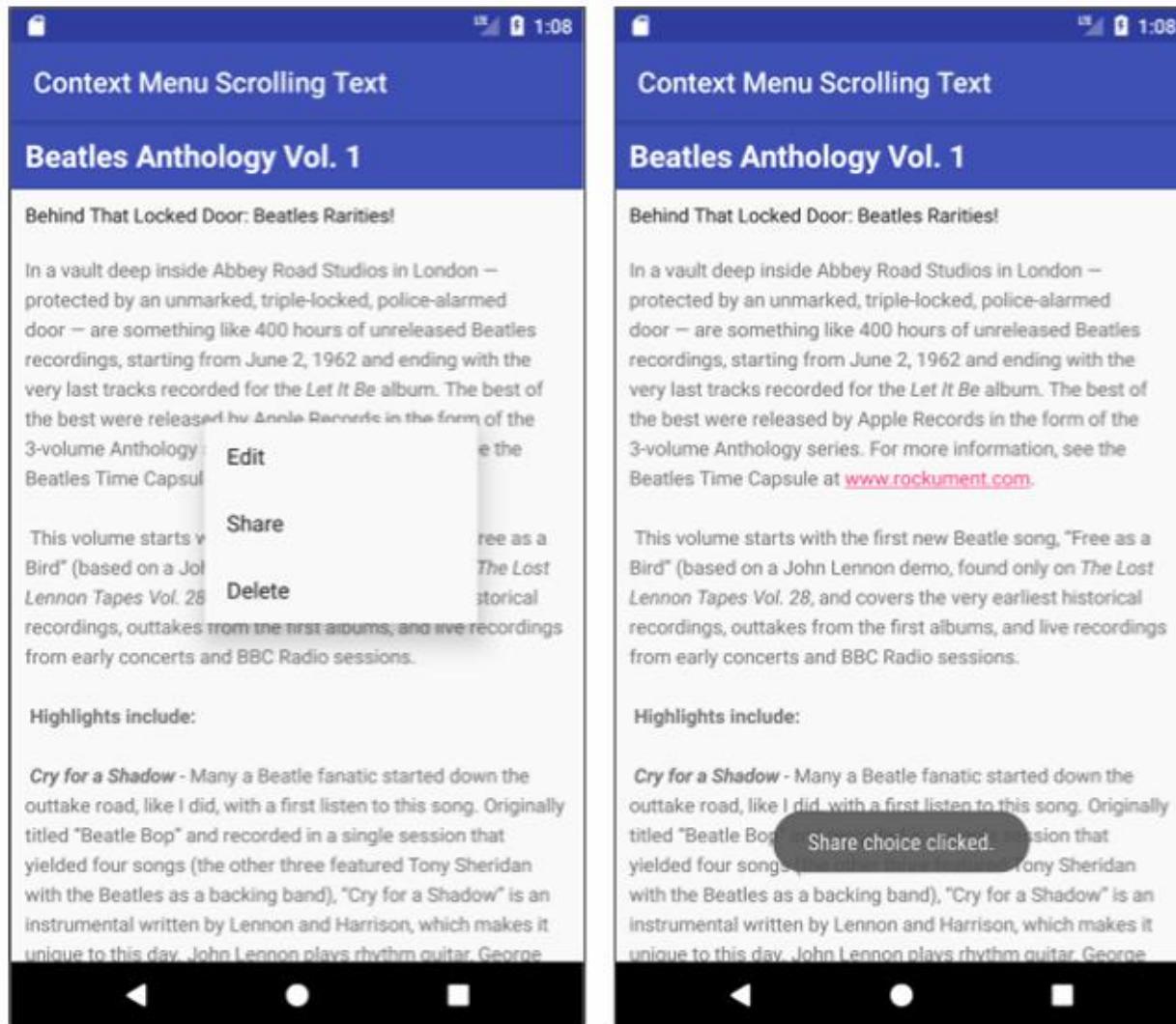
Chạy ứng dụng. Nhấp vào biểu tượng giỏ hàng trong thanh ứng dụng (mục **Order**) đưa bạn trực tiếp đến màn hình **OrderActivity**.

## Thử thách mã hóa

Thử thách: Menu ngữ cảnh cho phép người dùng thực hiện một hành động trên chế độ xem đã chọn. Mặc dù menu Tùy chọn trong thanh ứng dụng thường cung cấp các lựa chọn để điều hướng cho hoạt động khác, bạn sử dụng menu ngữ cảnh để cho phép người dùng sửa đổi chế độ xem trong hoạt động hiện tại.

Cả hai menu đều được mô tả trong XML, cả hai đều được khởi tạo bằng cách sử dụng menuinflater và cả hai đều sử dụng phương thức "trên vật phẩm được chọn" trong trường hợp này, OnContextItemSelected (). Vì vậy, các kỹ thuật xây dựng và sử dụng hai menu là tương tự nhau.

Menu ngữ cảnh xuất hiện dưới dạng danh sách nổi của các mục menu khi người dùng thực hiện cảm ứng & giữ aview, như được hiển thị ở phía bên trái của hình bên dưới. Đối với thử thách này, hãy thêm menu ngữ cảnh vào ứng dụng ScrollingText để hiển thị ba tùy chọn:**Edit**, **Share** và **Delete**, như trong hình bên dưới. Menu xuất hiện khi người dùng thực hiện cảm ứng và giữ TextView. Sau đó, ứng dụng hiển thị thông báo bánh mì nướng hiển thị tùy chọn menu được chọn, như được hiển thị ở phía bên phải của hình.



## Gợi ý

Menu ngữ cảnh tương tự như menu Tùy chọn, với hai sự khác biệt quan trọng:

- Menu ngữ cảnh phải được đăng ký để xem để menu thổi phồng khi chạm và giữ xảy ra trên chế độ xem.
- Mặc dù mã menu Tùy chọn được cung cấp bởi mẫu hoạt động cơ bản, đối với bối cảnh bạn phải tự thêm mã và tài nguyên menu.

Để giải quyết thách thức này, hãy làm theo các bước chung sau:

1. Tạo tệp tài nguyên menu XML cho các mục menu.

Nhấp chuột phải vào thư mục **New > Android Resource Directory**. Chọn **Menu** trong **menu** thả xuống Loại tài nguyên và bấm OK. Sau

đó, nhấp chuột phải vào thư mục **menu** mới, chọn Tệp tài nguyên mới > Menu, nhập tên menu\_context và nhấp vào **OK**. Mở **menu\_context** và nhập các mục menu như bạn đã làm cho menu Tùy chọn.

2. Đăng ký Chế độ xem vào menu ngữ cảnh bằng phương thức RegisterForContextMenu () .

Trong phương thức OnCreate (), đăng ký TextView:

```
TextView article_text = findViewById(R.id.article);
registerForContextMenu(article_text);
```

3. Thực hiện phương thức OnCreateContextMenu () trong hoạt động để khởi động menu.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
 ContextMenu.ContextMenuItemInfo menuInfo) {
 super.onCreateContextMenu(menu, v, menuInfo);
 MenuInflater inflater = getMenuInflater();
 inflater.inflate(R.menu.menu_context, menu);
}
```

4. Thực hiện phương thức OnContextItemSelected () trong hoạt động để xử lý các mục menu nhấp chuột. Trong trường hợp này, chỉ cần hiển thị bánh mì nướng với lựa chọn menu.

```
@Override
public boolean onContextItemSelected(MenuItem item) {
 switch (item.getItemId()) {
 case R.id.context_edit:
 displayToast("Edit choice clicked.");
 return true;
 case R.id.context_share:
 displayToast("Share choice clicked.");
 return true;
 case R.id.context_delete:
 displayToast("Delete choice clicked.");
 return true;
 default:
 return super.onContextItemSelected(item);
 }
}
```

Chạy ứng dụng. Nếu bạn nhấn và kéo, văn bản cuộn như trước.

Tuy nhiên, nếu bạn thực hiện một cú chạm dài, menu ngữ cảnh sẽ xuất hiện.

## Nhiệm vụ 4: Sử dụng hộp thoại để yêu cầu sự lựa chọn của người dùng

Bạn có thể cung cấp hộp thoại để yêu cầu sự lựa chọn của người dùng, chẳng hạn như cảnh báo yêu cầu người dùng nhấn OK hoặc hủy. Hộp thoại là một cửa sổ xuất hiện trên đỉnh của màn hình hoặc lấp đầy màn hình, làm gián đoạn luồng hoạt động.

Ví dụ: hộp thoại cảnh báo có thể yêu cầu người dùng nhấp vào Tiếp tục sau khi đọc nó hoặc cho người dùng lựa chọn đồng ý với một hành động bằng cách nhấp

vào nút dương (chẳng hạn như OK hoặc Chấp nhận) hoặc không đồng ý bằng cách nhấp vào nút âm (như Hủy). Sử dụng lớp con alertDialog của lớp hộp thoại để hiển thị hộp thoại tiêu chuẩn để cảnh báo.

Trong thực tế này, bạn sử dụng một nút để kích hoạt hộp thoại cảnh báo tiêu chuẩn. Trong một ứng dụng trong thế giới thực, bạn có thể kích hoạt hộp thoại cảnh báo dựa trên một số điều kiện hoặc dựa trên người dùng khai thác thứ gì đó.

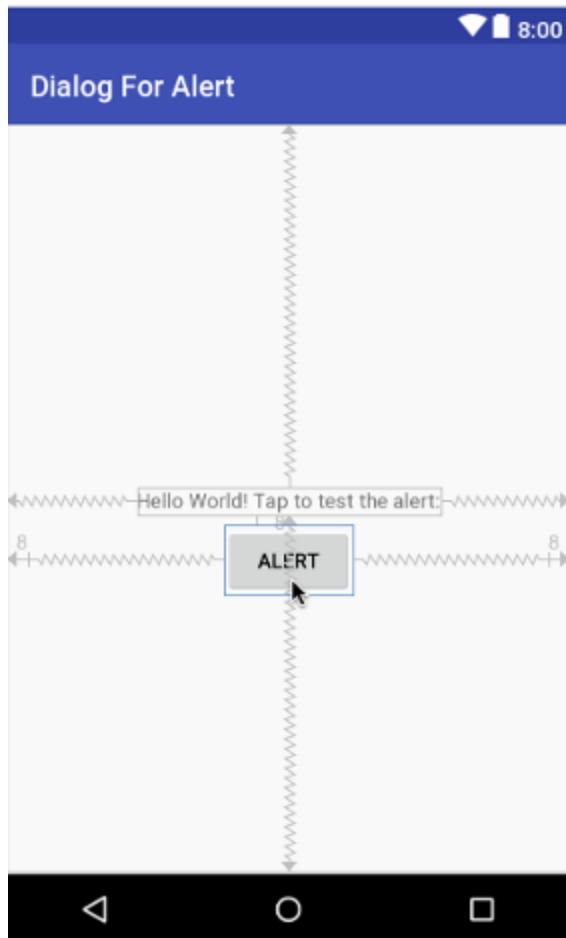
#### 4.1 Tạo một ứng dụng mới để hiển thị hộp thoại cảnh báo

Trong bài tập này, bạn xây dựng một cảnh báo với các nút OK và hủy bỏ. Cảnh báo được kích hoạt bởi người dùng khai thác một nút.

1. Tạo một dự án mới có tên là **Hộp thoại để cảnh báo** dựa trên mẫu hoạt động trống.
2. Mở tệp Bố cục **Activity\_Main.xml** để hiển thị trình chỉnh sửa bố cục.
3. Chính sửa phần tử **TextView** để nói Hello World! Nhấn để kiểm tra cảnh báo: Thay vì "HelloWorld!"
4. Thêm một nút dưới **TextView**. (Tùy chọn: Cắt nút ở dưới cùng của **TextView** và các cạnh của bố cục, với lề được đặt thành 8dp.)
5. Đặt văn bản của nút thành **Alert**.
6. Chuyển sang **Text tab** và trích xuất các chuỗi văn bản cho **TextView** và **Button** sang Chuỗi tài nguyên.
7. Thêm **Android:onClick** vào nút để gọi trình xử lý nhấp vào **onClickShowAlert()**. Sau khi bạn nhập nó, trình xử lý nhấp chuột được gạch chân màu đỏ vì nó chưa được tạo.

```
 android:onClick="onClickShowAlert"
```

Bây giờ bạn có một bố cục tương tự như sau:



## 4.2 Thêm hộp thoại cảnh báo vào hoạt động chính

Mẫu thiết kế xây dựng giúp dễ dàng tạo một đối tượng từ một lớp có nhiều thuộc tính cần thiết và tùy chọn và do đó sẽ yêu cầu rất nhiều tham số để xây dựng.

Không có mô hình này, bạn sẽ phải tạo các hàm tạo để kết hợp và các thuộc tính tùy chọn; Với mẫu này, mã dễ đọc và bảo trì hơn. Để biết thêm thông tin về mẫu thiết kế xây dựng, hãy xem Builder pattern.

Lớp xây dựng thường là một lớp thành viên tĩnh của lớp mà nó xây dựng. Sử dụng AlertDialog.Builder để xây dựng hộp thoại cảnh báo tiêu chuẩn, với setS.

Để cảnh báo, bạn cần tạo một đối tượng của alertDialog.builder. Bạn sẽ thêm onclickShowalert () Nhập vào Trình xử lý cho nút cảnh báo, điều này làm cho đối tượng này làm

thứ tự đầu tiên của kinh doanh. Điều đó có nghĩa là hộp thoại sẽ chỉ được tạo khi người dùng nhấp vào nút cảnh báo. Mặc dù mẫu mã hóa này là hợp lý khi sử dụng nút để kiểm tra cảnh báo, nhưng đối với các ứng dụng khác, bạn có thể muốn tạo

hộp thoại trong phương thức onCreate () để nó luôn sẵn sàng cho mã khác để kích hoạt nó.

1. Mở MainActivity và thêm phần đầu của phương thức onclickShowalert (

```
public void onClickShowAlert(View view) {
 AlertDialog.Builder myAlertDialog = new
 AlertDialog.Builder(MainActivity.this);
 // Set the dialog title and message.
}
```

Nếu alertDialog.builder không được nhận ra khi bạn nhập nó, hãy nhập vào biểu tượng bóng đèn màu đỏ và chọn phiên bản thư viện hỗ trợ (Android.support.v7.app.alertdialog) để nhập vào hoạt động của bạn.

2. Thêm mã để đặt tiêu đề và thông báo cho hộp thoại cảnh báo vào onclickshowalert () sau khi nhận xét:

```
// Set the dialog title and message.
myAlertDialog.setTitle("Alert");
myAlertDialog.setMessage("Click OK to continue, or Cancel to stop.");
// Add the dialog buttons.
```

3. Trích xuất các chuỗi trên thành các tài nguyên chuỗi như **alert\_title** và **alert\_message**.
- 4.Thêm các nút **OK** và **Cancel** vào cảnh báo bằng các phương thức **setPositiveButton()** và **setNegativeButton()**:

```
// Add the dialog buttons.

myAlertBuilder.setPositiveButton("OK", new
 DialogInterface.OnClickListener() {

 public void onClick(DialogInterface dialog, int which) {

 // User clicked OK button.

 Toast.makeText(getApplicationContext(), "Pressed OK",
 Toast.LENGTH_SHORT).show();

 }
 });

myAlertBuilder.setNegativeButton("Cancel", new
 DialogInterface.OnClickListener() {

 public void onClick(DialogInterface dialog, int which) {

 // User cancelled the dialog.

 Toast.makeText(getApplicationContext(), "Pressed Cancel",
 Toast.LENGTH_SHORT).show();
 }
 });

// Create and show the AlertDialog.
```

Sau

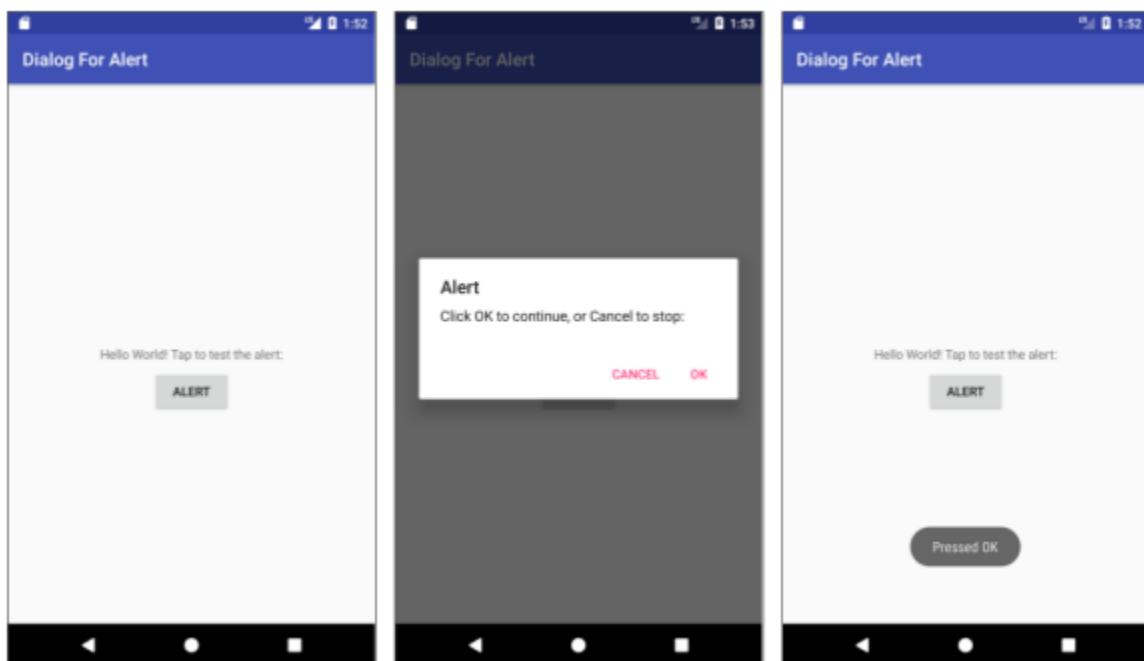
khi người dùng chạm vào nút **OK** hoặc **Cancel** trong cảnh báo, bạn có thể lấy lựa chọn của người dùng và sử dụng trong mã của mình. Trong ví dụ này, bạn hiển thị thông báo Toast.

5. Trích xuất chuỗi cho OK và Cancel thành chuỗi tài nguyên dưới dạng ok\_button và cancel\_button, và trích xuất chuỗi cho thông báo Toast.

6. Ở cuối phương thức **onClickShowAlert()**, hãy thêm **show()**, phương thức này sẽ tạo và sau đó hiển thị hộp thoại cảnh báo:

```
// Create and show the AlertDialog.
myAlertDialog.show();
```

- Chạy ứng dụng. Bạn sẽ có thể chạm vào nút Cảnh báo, được hiển thị ở phía bên trái của hình bên dưới, để xem hộp thoại cảnh báo, được hiển thị ở giữa hình bên dưới. Hộp thoại hiển thị các nút **OK** và **Cancel**, và một thông báo **Toast** xuất hiện cho biết bạn đã nhấn nút nào, như được hiển thị ở phía bên phải của hình bên dưới.



## Nhiệm vụ 5: Sử dụng bộ chọn để người dùng nhập dữ liệu

Android cung cấp các hộp thoại sẵn sàng sử dụng, được gọi là bộ chọn, để chọn thời gian hoặc ngày. Bạn có thể sử dụng chúng để đảm bảo rằng người dùng của bạn chọn thời gian hoặc ngày hợp lệ được định dạng chính xác và điều chỉnh theo thời gian và ngày cục bộ của người dùng. Mỗi bộ chọn cung cấp các điều khiển để chọn từng phần của thời gian (giờ, phút, AM/PM) hoặc ngày (tháng, ngày, năm). Bạn có thể đọc tất cả về cách thiết lập bộ chọn trong Bộ chọn.

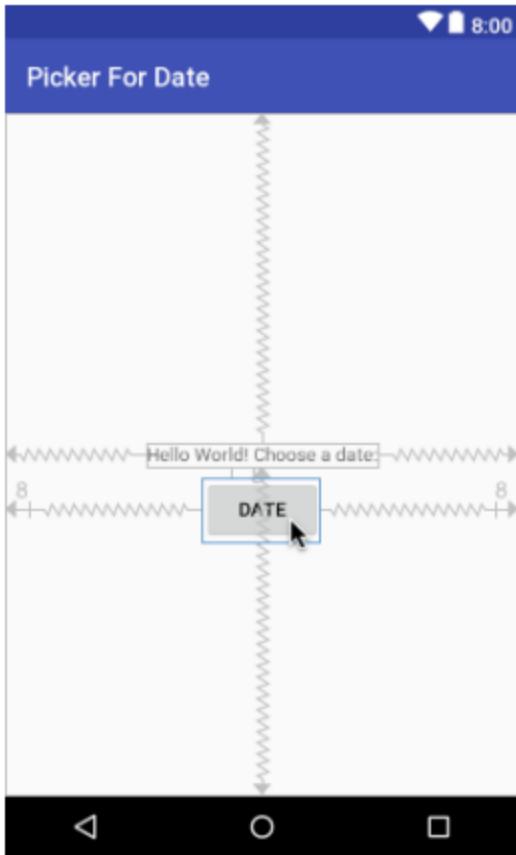
Trong nhiệm vụ này, bạn sẽ tạo một dự án mới và thêm bộ chọn ngày. Bạn cũng sẽ tìm hiểu cách sử dụng **Fragment**, đây là một hành vi hoặc một phần của UI trong một Hoạt động. Nó giống như một Hoạt động nhỏ trong hoạt động chính, có vòng đời riêng và được sử dụng để xây dựng bộ chọn. Tất cả công việc được thực hiện cho bạn. Để tìm hiểu về lớp **Fragment**, hãy xem **Fragment** trong Hướng dẫn API. Một lợi ích của việc sử dụng **Fragment** cho một trình chọn là bạn có thể cô lập các phần mã để quản lý ngày và giờ cho nhiều ngôn ngữ khác nhau hiển thị ngày và giờ theo nhiều cách khác nhau. Cách tốt nhất để hiển thị trình chọn là sử dụng một thẻ hiện của **DialogFragment**, là một lớp con của **Fragment**.

## 5.1 Tạo ứng dụng mới để hiển thị bộ chọn ngày

Để bắt đầu tác vụ này, hãy tạo ứng dụng cung cấp Nút để hiển thị bộ chọn ngày.

1. Tạo dự án mới có tên là Bộ chọn ngày dựa trên mẫu Hoạt động trống.
2. Mở tệp bố cục activity\_main.xml để hiển thị trình chỉnh sửa bố cục.
3. Chỉnh sửa văn bản "**Hello World!**" của phần tử **TextView** thành **Hello World!**. Chọn ngày:.
4. Thêm Nút bên dưới **TextView**. (Tùy chọn: Giới hạn Nút ở dưới cùng của **TextView** và các cạnh của bố cục, với lề được đặt thành 8dp.)
5. Đặt văn bản của Nút thành **Date**.
6. Chuyển sang tab Văn bản và trích xuất chuỗi cho **TextView** và Nút thành chuỗi resources.
7. Thêm thuộc tính **android:onClick** vào Nút để gọi trình xử lý nhập **showDatePicker()**. Sau khi nhập, trình xử lý nhập được gạch chân màu đỏ vì nó chưa được tạo

Bây giờ bạn sẽ có một bố cục tương tự như sau:



## 5.2 Tạo một đoạn mã mới cho bộ chọn ngày

Trong bước này, bạn thêm một Đoạn mã cho bộ chọn ngày.

1. Mở rộng ứng dụng > java > com.example.android.pickertodate và chọn MainActivity.
2. Chọn **File > New > Fragment > Fragment (Blank)** và đặt tên cho đoạn mã **DatePickerFragment**. Xóa cả ba hộp kiểm để bạn không tạo XML bô cục, bao gồm các phương thức nhà máy đoạn mã hoặc bao gồm các lệnh gọi lại giao diện. Bạn không cần tạo bô cục cho bộ chọn chuẩn. Nhấp vào Hoàn tất.
3. Mở **DatePickerFragment** và chỉnh sửa định nghĩa lớp DatePickerFragment để mở rộng DialogFragment và triển khai DatePickerDialog.OnDateSetListener để tạo bộ chọn ngày chuẩn với trình lắng nghe. Xem Bộ chọn để biết thêm thông tin về việc mở rộng DialogFragment cho bộ chọn ngày:

```
public class DatePickerFragment extends DialogFragment
 implements DatePickerDialog.OnDateSetListener {
```

Khi bạn nhập DialogFragment và DatePickerDialog.OnDateSetListener, Android Studio tự động thêm một số câu lệnh import vào khôi import ở trên cùng, bao gồm:

```
import android.app.DatePickerDialog;
import android.support.v4.app.DialogFragment;
```

Ngoài ra, biểu tượng bóng đèn màu đỏ sẽ xuất hiện ở lề trái sau vài giây.

- Nhập vào biểu tượng bóng đèn màu đỏ và chọn Implement methods từ menu bật lên. Một hộp thoại xuất hiện với onDateSet() đã được chọn và tùy chọn Insert @Override đã được chọn. Nhấp vào OK để tạo phương thức onDateSet() trống. Phương thức này sẽ được gọi khi người dùng đặt ngày. Sau khi thêm phương thức onDateSet() trống, Android Studio sẽ tự động thêm phần sau vào khối import ở trên cùng:

```
import android.widget.DatePicker;
```

Các tham số onDateSet() phải là int i, int i1 và int i2. Đổi tên của các tham số này thành tên dễ đọc hơn:

```
public void onDateSet(DatePicker datePicker,
 int year, int month, int day)
```

- Xóa hàm dựng public DatePickerFragment() trống.
- Thay thế toàn bộ phương thức onCreateView() bằng onCreateDialog() trả về Dialog, và chú thích onCreateDialog() bằng @NonNull để chỉ ra rằng giá trị trả về Dialog không thể là null. Android Studio hiển thị một bóng đèn màu đỏ bên cạnh phương thức vì nó chưa trả về bất kỳ thứ gì.

```
@NonNull
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
```

- Thêm mã sau vào onCreateDialog() để khởi tạo năm, tháng và ngày từ Calendar, và trả về hộp thoại và các giá trị này cho Activity. Khi bạn nhập Calendar.getInstance(), hãy chỉ định lệnh nhập là java.util.Calendar

```
// Use the current date as the default date in the picker.

final Calendar c = Calendar.getInstance();
int year = c.get(Calendar.YEAR);
int month = c.get(Calendar.MONTH);
int day = c.get(Calendar.DAY_OF_MONTH);

// Create a new instance of DatePickerDialog and return it.

return new DatePickerDialog(getActivity(), this, year, month, day);
```

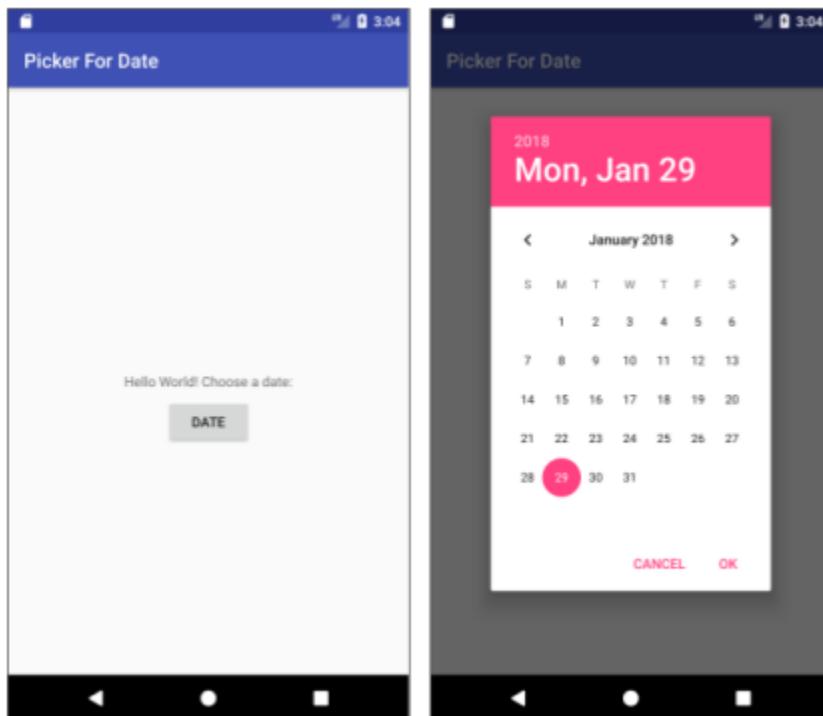
## 5.4 Sửa đổi hoạt động chính

Mặc dù phần lớn mã trong MainActivity.java vẫn giữ nguyên, bạn cần thêm một phương thức tạo một thẻ hiện của FragmentManager để quản lý Fragment và hiển thị trình chọn ngày.

1. Mở MainActivity.
2. Thêm trình xử lý showDatePickerDialog() cho Nút Ngày. Nó tạo một thẻ hiện của FragmentManager bằng cách sử dụng getSupportFragmentManager() để quản lý Fragment tự động và hiển thị trình chọn. Để biết thêm thông tin về lớp Fragment, hãy xem Fragments

```
public void showDatePicker(View view) {
 DialogFragment newFragment = new DatePickerFragment();
 newFragment.show(getSupportFragmentManager(),"datePicker");
}
```

3. Trích xuất chuỗi "datePicker" vào chuỗi tài nguyên datepicker.
4. Chạy ứng dụng. Bạn sẽ thấy trình chọn ngày sau khi chạm vào nút Ngày.



## 5.5 Sử dụng ngày đã chọn

Trong bước này, bạn truyền ngày trả lại MainActivity.java và chuyển đổi ngày thành chuỗi mà bạn có thể hiển thị trong thông báo Toast.

1. Mở MainActivity và thêm phương thức processDatePickerResult() rỗng lấy năm, tháng và ngày làm đối số:

```
public void processDatePickerResult(int year, int month, int day) { }
```

2. Thêm mã sau vào phương thức processDatePickerResult() để chuyển đổi tháng, ngày và năm thành các chuỗi riêng biệt và nối ba chuỗi bằng dấu gạch chéo cho định dạng ngày của Hoa Kỳ:

```
String month_string = Integer.toString(month+1);
String day_string = Integer.toString(day);
String year_string = Integer.toString(year);
String dateMessage = (month_string +
 "/" + day_string + "/" + year_string);
```

Số nguyên tháng được trả về bởi trình chọn ngày bắt đầu đếm từ 0 cho tháng 1, vì vậy bạn cần thêm 1 vào để hiển thị các tháng bắt đầu từ 1.

3. Thêm nội dung sau vào sau đoạn mã trên để hiển thị thông báo Toast:

```
Toast.makeText(this, "Date: " + dateMessage,
 Toast.LENGTH_SHORT).show();
```

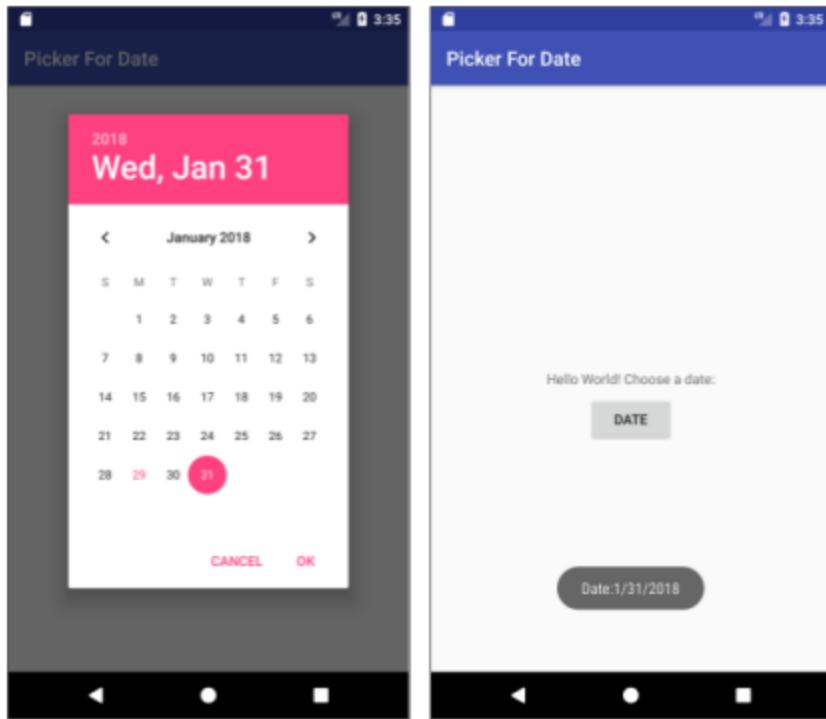
4. Trích xuất chuỗi được mã hóa cùng "Date: " thành một tài nguyên chuỗi có tên là date.
5. Mở DatePickerFragment và thêm nội dung sau vào phương thức onDateSet() để gọi processDatePickerResult() trong MainActivity và truyền cho nó năm, tháng và ngày:

```
@Override
public void onDateSet(DatePicker datePicker,
 int year, int month, int day) {
```

```
 MainActivity activity = (MainActivity) getActivity();
 activity.processDatePickerResult(year, month, day);
}
```

Bạn sử dụng getActivity(), khi được sử dụng trong một Fragment, sẽ trả về Activity mà Fragment hiện đang được liên kết. Bạn cần điều này vì bạn không thể gọi một phương thức trong MainActivity mà không có ngữ cảnh của MainActivity (bạn sẽ phải sử dụng một ý định thay vào đó, như bạn đã học trong một bài học khác). Activity kế thừa ngữ cảnh, do đó bạn có thể sử dụng nó làm ngữ cảnh để gọi phương thức (như trong activity.processDatePickerResult).

6. Chạy ứng dụng. Sau khi chọn ngày, ngày sẽ xuất hiện trong thông báo Toast như được hiển thị ở bên phải của hình sau.



## Thử thách mã hóa 2

Thử thách: Tạo một ứng dụng có tên là Picker For Time triển khai bộ chọn thời gian bằng cùng kỹ thuật bạn vừa học để thêm bộ chọn ngày.

Gợi ý:

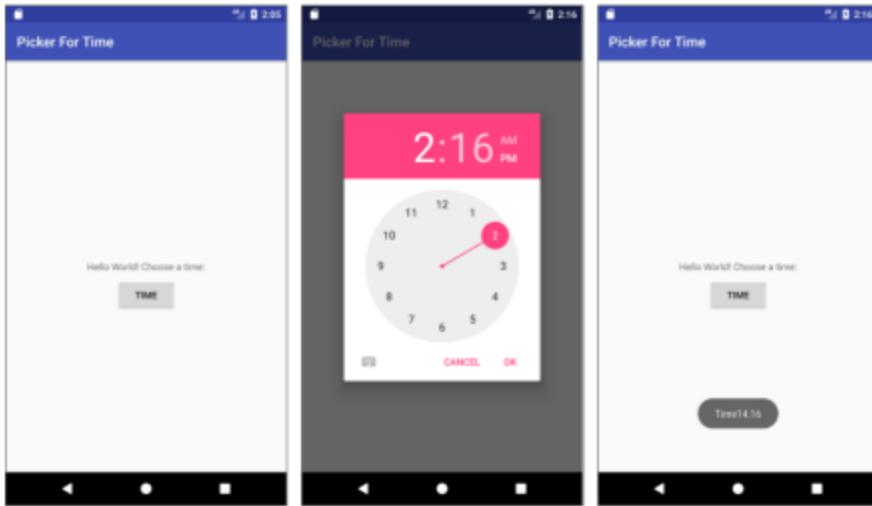
- Triển khai TimePickerDialog.OnTimeSetListener để tạo bộ chọn thời gian chuẩn với trình lắng nghe.
- Thay đổi các tham số của phương thức onTimeSet() từ int i thành int hourOfDay và int i1 thành int minute.
- Lấy giờ và phút hiện tại từ Calendar:

```
final Calendar c = Calendar.getInstance();
int hour = c.get(Calendar.HOUR_OF_DAY);
int minute = c.get(Calendar.MINUTE);
```

- Tao phương thức processTimePickerResult() tương tự như processDatePickerResult() trong tác vụ trước đó chuyển đổi các phần tử thời gian thành chuỗi và hiển thị kết quả trong thông báo Toast.

Chạy ứng dụng và nhấp vào nút Thời gian như được hiển thị ở phía bên trái của hình bên dưới. Bộ chọn thời gian

sẽ xuất hiện, như được hiển thị ở giữa hình. Chọn thời gian và nhấp vào OK. Thời gian sẽ xuất hiện trong thông báo Toast ở cuối màn hình, như được hiển thị ở phía bên phải của hình.



## Tóm tắt

Cung cấp menu tùy chọn và thanh ứng dụng:

- Khởi động ứng dụng hoặc Hoạt động của bạn bằng mẫu Hoạt động cơ bản để tự động thiết lập thanh ứng dụng, menu tùy chọn và nút hành động nội.
- Mẫu thiết lập bô cục CoordinatorLayout với bô cục AppBarLayout nhúng. AppBarLayout giống như LinearLayout theo chiều dọc. Nó sử dụng lớp Thanh công cụ trong thư viện hỗ trợ, thay vì ActionBar gốc, để triển khai thanh ứng dụng.
- Mẫu sửa đổi tệp AndroidManifest.xml để Hoạt động .MainActivity được thiết lập để sử dụng chủ đề NoActionBar. Chủ đề này được định nghĩa trong tệp styles.xml.
- Mẫu thiết lập MainActivity để mở rộng AppCompatActivity và bắt đầu bằng phương thức onCreate(), phương thức này thiết lập chế độ xem nội dung và Thanh công cụ. Sau đó, nó gọi setSupportActionBar() và truyền thanh công cụ cho phương thức này, thiết lập thanh công cụ làm thanh ứng dụng cho Hoạt động.
- Xác định các mục menu trong tệp menu\_main.xml. Thuộc tính android:orderInCategory chỉ định thứ tự các mục menu xuất hiện trong menu, với số thấp nhất xuất hiện cao hơn trong menu.
- Sử dụng phương thức onOptionsItemSelected() để xác định mục menu nào đã được chạm vào.

## Thêm biểu tượng cho mục menu tùy chọn:

- Mở rộng res trong ngăn Project > Android và nhấp chuột phải (hoặc giữ Control khi nhấp) vào thư mục drawable
  - . Chọn New > Image Asset.
  - Chọn Action Bar và Tab Items trong menu thả xuống và đổi tên tệp hình ảnh.
  - Nhấp vào hình ảnh clip art để chọn hình ảnh clip art làm biểu tượng. Chọn một biểu tượng.
  - Chọn HOLO\_DARK từ menu thả xuống Theme.

Hiển thị các mục menu dưới dạng biểu tượng trên thanh ứng dụng:

- Sử dụng thuộc tính app:showAsAction trong menu\_main.xml với các giá trị sau.
- "always": Luôn xuất hiện trên thanh ứng dụng. (Nếu không đủ chỗ, nó có thể chồng lên các biểu tượng menu khác.)
- "ifRoom": Xuất hiện trên thanh ứng dụng nếu có chỗ.
- "never": Không bao giờ xuất hiện trên thanh ứng dụng; văn bản của nó sẽ xuất hiện trong menu tràn.

Sử dụng hộp thoại cảnh báo:

- Sử dụng hộp thoại để yêu cầu người dùng lựa chọn, chẳng hạn như cảnh báo yêu cầu người dùng chạm vào OK hoặc

Hủy. Sử dụng hộp thoại một cách hạn chế vì chúng làm gián đoạn quy trình làm việc của người dùng.

#### **Hiển thị các mục menu dưới dạng biểu tượng trên thanh ứng dụng:**

- Sử dụng thuộc tính app:showAsAction trong menu\_main.xml với các giá trị sau.
  - "always": Luôn xuất hiện trên thanh ứng dụng. (Nếu không đủ chỗ, nó có thể chồng lên các biểu tượng menu khác.)
  - "ifRoom": Xuất hiện trên thanh ứng dụng nếu có chỗ.
  - "never": Không bao giờ xuất hiện trên thanh ứng dụng; văn bản của nó sẽ xuất hiện trong menu tràn.

#### **Sử dụng hộp thoại cảnh báo:**

- Sử dụng hộp thoại để yêu cầu người dùng lựa chọn, chẳng hạn như cảnh báo yêu cầu người dùng chạm vào OK hoặc

Hủy. Sử dụng hộp thoại một cách hạn chế vì chúng làm gián đoạn quy trình làm việc của người dùng.

- Sử dụng lớp AlertDialog của lớp Dialog để hiển thị hộp thoại chuẩn cho cảnh báo.
- Sử dụng AlertDialog.Builder để xây dựng hộp thoại cảnh báo chuẩn, với setTitle() để đặt tiêu đề, setMessage() để đặt thông báo và setPositiveButton() và setNegativeButton() để đặt các nút.

#### **Sử dụng bộ chọn cho đầu vào của người dùng:**

- Sử dụng DialogFragment, một lớp con của Fragment, để xây dựng bộ chọn như bộ chọn ngày hoặc bộ chọn giờ.
- Tạo DialogFragment và triển khai DatePickerDialog.OnDateSetListener để tạo bộ chọn ngày chuẩn với trình lắng nghe. Bao gồm onDateSet() trong Fragment này.
- Thay thế phương thức onCreateDialog() bằng onCreateDialog() trả về Dialog. Khởi tạo ngày cho bộ chọn ngày từ Calendar và trả về hộp thoại và các giá trị này cho Activity.
- Tạo một phiên bản của FragmentManager bằng cách sử dụng getSupportFragmentManager() để quản lý

#### **Khái niệm liên quan**

Tài liệu về khái niệm liên quan có trong 4.3: Menu và trình chọn.

#### **Tìm hiểu thêm**

Tài liệu Android Studio:

- Hướng dẫn sử dụng Android Studio
- Tạo biểu tượng ứng dụng bằng Image Asset Studio
- Thêm thanh ứng dụng
- Menu
- Thanh công cụ
- Thư viện hỗ trợ appcompat v7
- AppBarLayout
- onOptionsItemSelected()
- View
- MenuInflater
- registerForContextMenu()
- onCreateContextMenu()
- onContextItemSelected()
- Dialogs
- AlertDialog

- Pickers
- Fragments
- DialogFragment
- FragmentManager
- Calendar

### **Đặc tả thiết kế Material:**

- Lưới bố cục đáp ứng
- Dialogs

### **Khác:**

- Blog dành cho nhà phát triển Android: Thư viện hỗ trợ thiết kế Android
- Mẫu xây dựng trong Wikipedia

### **Bài tập về nhà**

Xây dựng và chạy ứng dụng

Mở ứng dụng DroidCafeOptions mà bạn đã tạo trong bài học này.

1. Thêm nút Ngày bên dưới tùy chọn phân phối để hiển thị trình chọn ngày.
2. Hiển thị ngày do người dùng chọn trong tin nhắn Toast.

Trả lời các câu hỏi sau

Câu hỏi 1

Tên tệp mà bạn tạo các mục menu tùy chọn là gì? Chọn một trong các câu sau:

- menu.java
- menu\_main.xml
- activity\_main.xml
- content\_main.xml

Câu hỏi 2

Phương thức nào được gọi khi nhấp vào mục menu tùy chọn? Chọn một trong các câu sau:

- onOptionsItemSelected(MenuItem item)
- onClick(View view)
- onContextItemSelected()
- onClickShowAlert()

Câu hỏi 3

Câu lệnh nào sau đây đặt tiêu đề cho hộp thoại cảnh báo? Chọn một:

- myAlertDialog.setMessage("Alert");
- myAlertDialog.setPositiveButton("Alert");
- myAlertDialog.setTitle("Alert");
- AlertDialog.Builder myAlertDialog = new AlertDialog.Builder("Alert");

Câu hỏi 4

Bạn tạo DialogFragment cho bộ chọn ngày ở đâu? Chọn một trong các mục sau:

- Trong phương thức onCreate() trong Hoạt động lưu trữ.
- Trong phương thức onCreateOptionsMenu() trong Fragment.
- Trong phương thức onCreateView() trong phần mở rộng của DialogFragment.
- Trong phương thức onCreateDialog() trong phần mở rộng của DialogFragment.

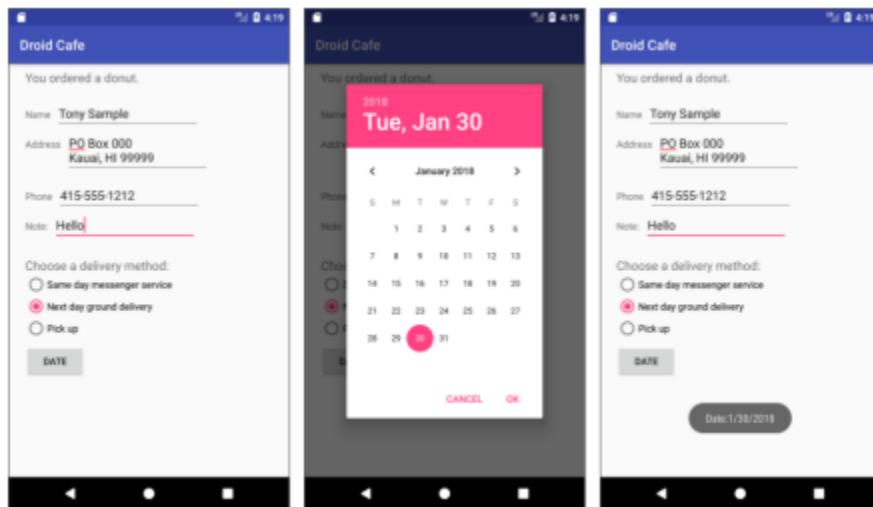
Nộp ứng dụng của bạn để chấm điểm

### **Hướng dẫn cho người chấm điểm**

Kiểm tra xem ứng dụng có các tính năng sau không:

- Bộ chọn ngày được thêm dưới dạng DialogFragment.
- Nhấp vào nút Ngày (tham khảo phía bên trái của hình bên dưới) trong OrderActivity sẽ hiển thị bộ chọn ngày (tham khảo phần giữa của hình).

- Nhập vào nút OK trong trình chọn ngày sẽ hiển thị thông báo Toast trong OrderActivity với ngày đã chọn (tham khảo phía bên phải của hình)



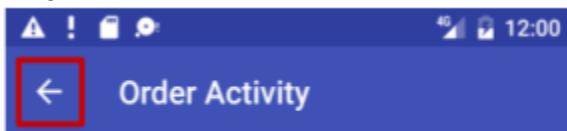
## Bài 4.4: Điều hướng người dùng

### Giới thiệu

Trong giai đoạn đầu phát triển ứng dụng, bạn nên xác định đường dẫn bạn muốn người dùng thực hiện thông qua ứng dụng của mình để thực hiện từng tác vụ. (Các tác vụ là những việc như đặt hàng hoặc duyệt nội dung.)

Mỗi đường dẫn cho phép người dùng điều hướng qua, vào và ra khỏi các tác vụ và phần nội dung trong ứng dụng.

Trong bài thực hành này, bạn sẽ học cách thêm nút Lên (mũi tên hướng sang trái) vào thanh ứng dụng, như được hiển thị bên dưới.



Nút Lên luôn được sử dụng để điều hướng đến màn hình cha trong hệ thống phân cấp. Nó khác với Nút Quay lại (hình tam giác ở cuối màn hình), cung cấp điều hướng đến bất kỳ màn hình nào mà người dùng đã xem lần cuối.

Bài thực hành này cũng giới thiệu điều hướng tab, trong đó các tab xuất hiện ở đầu màn hình, cung cấp điều hướng đến các màn hình khác. Điều hướng tab là một cách phổ biến để tạo điều hướng ngang từ một màn hình con đến một màn hình con anh chị em, như thể hiện trong hình bên dưới



Trong hình trên:

1. Điều hướng ngang từ màn hình danh mục này (Tin tức hàng đầu, Tin tức công nghệ và Nấu ăn) sang màn hình khác

2. Điều hướng ngang từ màn hình câu chuyện này (Câu chuyện) sang màn hình khác.

Với các tab, người dùng có thể điều hướng đến và đi từ màn hình anh chị em mà không cần điều hướng lên màn hình cha mẹ. Các tab cũng có thể cung cấp điều hướng đến và đi từ các câu chuyện, là các màn hình anh chị em trong màn hình cha mẹ Top Stories.

Các tab phù hợp nhất cho bốn hoặc ít hơn bốn màn hình anh chị em. Để xem một màn hình khác, người dùng có thể chạm vào một tab hoặc vuốt sang trái hoặc phải.

## Những điều bạn cần biết

### **Bạn cần có thể:**

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các thành phần UI bằng trình chỉnh sửa bố cục.
- Chính sửa mã bố cục XML và truy cập các thành phần từ mã Java của bạn.
- Thêm các mục menu và biểu tượng vào menu tùy chọn trong thanh ứng dụng.

### **Bạn sẽ học được gì**

- Cách thêm nút Lên vào thanh ứng dụng.
- Cách thiết lập ứng dụng với chế độ điều hướng tab và chế độ xem vuốt.

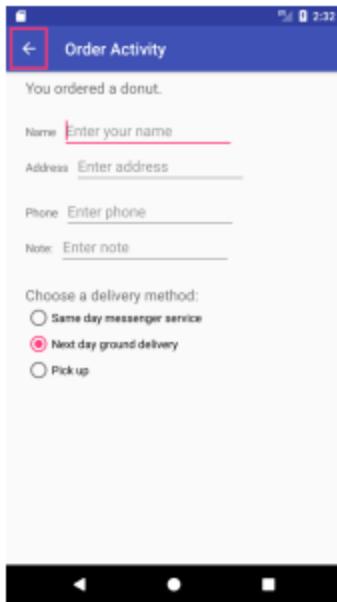
### **Bạn sẽ làm gì**

- Tiếp tục thêm các tính năng vào dự án Droid Cafe từ phần thực hành trước.
- Cung cấp nút Lên trong thanh ứng dụng để điều hướng lên Hoạt động chính.
- Tạo ứng dụng mới với các tab để điều hướng màn hình Hoạt động cũng có thể vuốt

### **Tổng quan về ứng dụng**

Trong bài thực hành trước về cách sử dụng menu tùy chọn, bạn đã làm việc trên một ứng dụng có tên là Droid Cafe được tạo bằng mẫu Hoạt động cơ bản. Mẫu này cung cấp một thanh ứng dụng ở đầu màn hình. Bạn sẽ học cách thêm nút Lên (mũi tên hướng sang trái) vào thanh ứng dụng để điều hướng lên từ Hoạt động thứ hai (OrderActivity) đến Hoạt động cha (MainActivity). Thao tác này sẽ hoàn tất ứng dụng Droid Cafe.

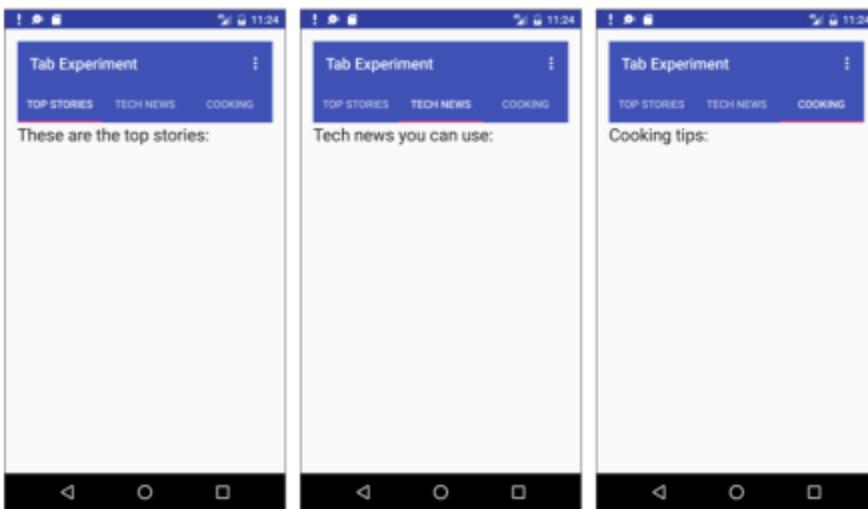
Để bắt đầu dự án từ nơi bạn đã dừng lại trong bài thực hành trước, hãy tải xuống dự án Android Studio DroidCafeOptions.



Bạn cũng sẽ tạo một ứng dụng để điều hướng tab hiển thị ba tab bên dưới thanh ứng dụng để điều hướng đến màn hình anh chị em. Khi người dùng chạm vào một tab, màn hình sẽ hiển thị màn hình nội dung, tùy thuộc vào

tab mà người dùng đã chạm vào. Người dùng cũng có thể vuốt sang trái và phải để truy cập màn hình nội dung.

Lớp ViewPager tự động xử lý thao tác vuốt của người dùng đến màn hình hoặc các thành phần View.

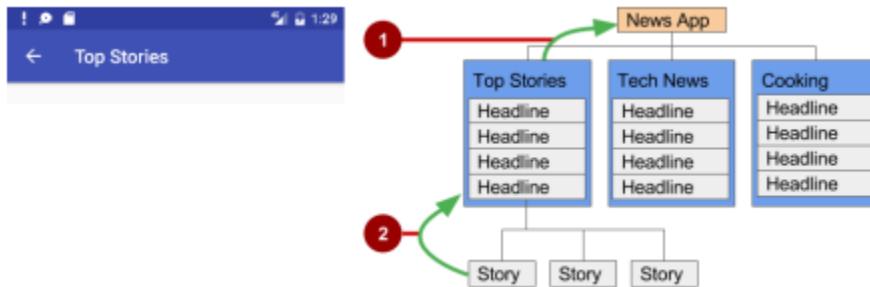


## Nhiệm vụ 1: Thêm nút Lên để điều hướng tố tiên

Ứng dụng của bạn phải giúp người dùng dễ dàng tìm đường quay lại màn hình chính của ứng dụng, thường là Hoạt động cha. Một cách để thực hiện điều này là cung cấp nút Lên trên thanh ứng dụng cho mỗi Hoạt động là con của Hoạt động cha.

Nút Lên cung cấp điều hướng "lên" tố tiên, cho phép người dùng đi lên từ trang con đến trang cha. Nút Lên là mũi tên hướng sang trái ở phía bên trái của thanh ứng dụng, như được hiển thị ở phía bên trái của hình bên dưới.

Khi người dùng chạm vào nút Lên, ứng dụng sẽ điều hướng đến Hoạt động cha. Sơ đồ ở phía bên phải của hình bên dưới cho thấy cách sử dụng nút Lên để điều hướng trong ứng dụng dựa trên các mối quan hệ phân cấp giữa các màn hình.



Trong hình trên:

1. Điều hướng từ các thành phần cấp một đến thành phần cấp hai.
  2. Điều hướng từ các thành phần cấp hai đến màn hình con cấp một đóng vai trò là màn hình cha
- Mẹo: Nút Quay lại (hình tam giác ở cuối thiết bị) và nút Lên trong Giao diện người dùng là hai thứ khác nhau:

Nút Quay lại cung cấp chức năng điều hướng đến màn hình được xem gần đây nhất. Nếu bạn có nhiều màn hình con mà người dùng có thể điều hướng bằng mẫu điều hướng ngang (như mô tả trong phần tiếp theo), nút Quay lại sẽ đưa người dùng trở lại màn hình con trước đó, không phải màn hình cha.

Để cung cấp chức năng điều hướng từ màn hình con trở lại màn hình cha, hãy sử dụng nút Lên. Để biết thêm về chức năng điều hướng Lên, hãy xem mục Cung cấp chức năng điều hướng Lên. Như bạn đã tìm hiểu trước đó, khi thêm hoạt động vào ứng dụng, bạn có thể thêm chức năng điều hướng Nút Lên vào một Hoạt động con như OrderActivity bằng cách khai báo thành phần cha của Hoạt động là MainActivity trong tệp AndroidManifest.xml. Bạn cũng có thể đặt thuộc tính android:label cho tiêu đề cho màn hình Activity, chẳng hạn như "Order Activity". Thực hiện theo các bước sau:

1. Nếu bạn chưa mở ứng dụng Droid Cafe từ bài thực hành trước, hãy tải xuống dự án Android Studio DroidCafeOptions và mở dự án.
2. Mở AndroidManifest.xml và thay đổi phần tử Activity cho OrderActivity thành sau

1.

```
<activity android:name="com.example.android.droidcafeinput.OrderActivity"
```

```

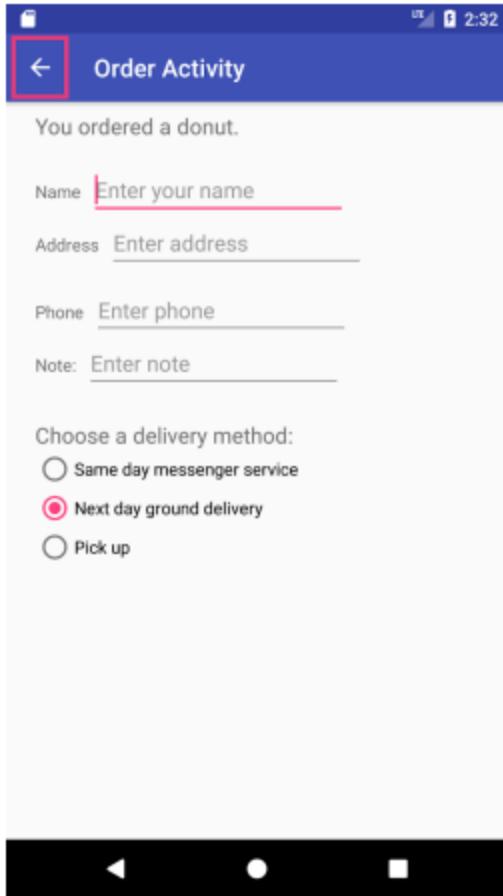
 android:label="Order Activity"
 android:parentActivityName=".MainActivity">
 <meta-data android:name="android.support.PARENT_ACTIVITY"
 android:value=".MainActivity"/>
 </activity>

```

2. Trích xuất giá trị android:label là "Order Activity" thành một tài nguyên chuỗi có tên title\_activity\_order.

3. Chạy ứng dụng

Màn hình Order Activity giờ đây bao gồm nút Up (được đánh dấu trong hình bên dưới) trên thanh ứng dụng để quay lại Activity cha.



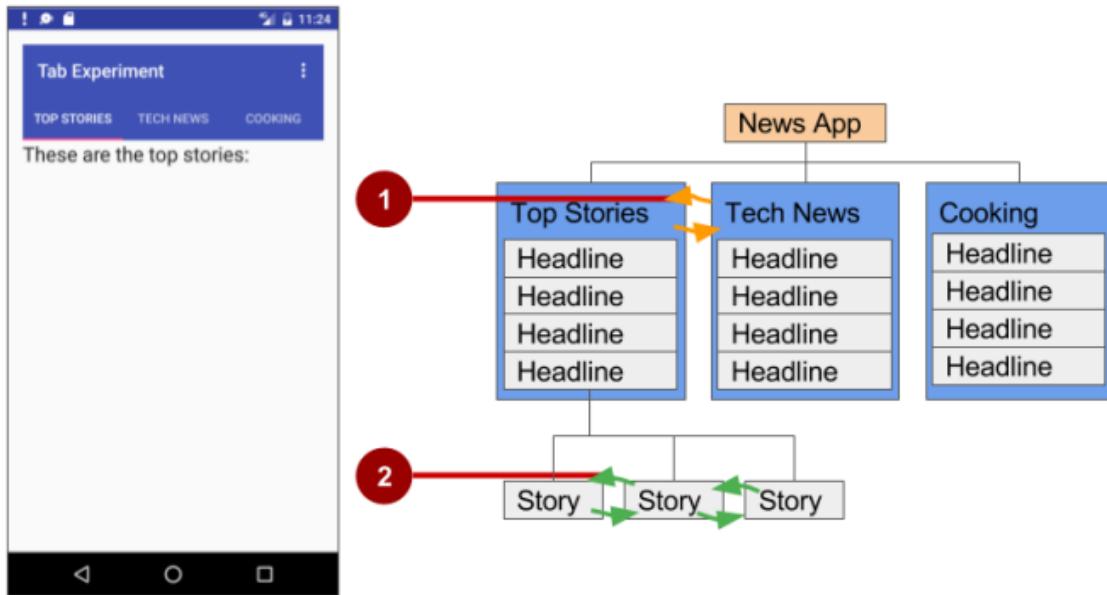
### Task 1: Mã nguồn giải pháp

Dự án Android Studio: [DroidCafeOptionsUp](#)

### Task 2: Sử dụng điều hướng bằng tab với chế độ vuốt

Với điều hướng ngang, bạn cho phép người dùng di chuyển từ một mục ngang cấp sang mục khác (ở cùng cấp trong một hệ thống phân cấp nhiều tầng).

Ví dụ, nếu ứng dụng của bạn cung cấp nhiều danh mục tin tức (chẳng hạn như **Tin tức hàng đầu**, **Công nghệ**, và **Nấu ăn**, như trong hình bên dưới), bạn sẽ muốn cung cấp cho người dùng khả năng di chuyển từ danh mục này sang danh mục khác mà không cần quay lại màn hình cha. Một ví dụ khác về điều hướng ngang là khả năng vuốt sang trái hoặc phải trong một cuộc trò chuyện của Gmail để xem email mới hơn hoặc cũ hơn trong cùng hộp thư đến.



Trong hình trên:

1. Điều hướng ngang từ màn hình danh mục này sang màn hình danh mục khác.
2. Điều hướng ngang từ màn hình bài viết này sang màn hình bài viết khác.

Bạn có thể triển khai điều hướng ngang bằng các tab đại diện cho từng màn hình. Các tab xuất hiện ở phía trên của màn hình, như minh họa ở bên trái của hình trên, để cung cấp khả năng điều hướng đến các màn hình khác.

Điều hướng bằng tab là một giải pháp rất phổ biến để điều hướng ngang từ một màn hình con sang một màn hình con khác ngang cấp—cùng vị trí trong hệ thống phân cấp và có chung đặc điểm.

Màn hình cha. Điều hướng bằng tab thường được kết hợp với khả năng vượt các màn hình con từ trái sang phải và từ phải sang trái.

Lớp chính được sử dụng để hiển thị tab là **TabLayout** trong **Android Design Support Library**. Lớp này cung cấp một bố cục ngang để hiển thị các tab. Bạn có thể hiển thị các tab bên dưới thanh ứng dụng và sử dụng lớp **PagerAdapter** để di chuyển nội dung cho các "trang" màn hình bên trong **ViewPager**.

**ViewPager** là một trình quản lý bối cảnh cho phép người dùng lật qua lại giữa các màn hình. Đây là một mô hình phổ biến để hiển thị nhiều màn hình nội dung trong một **Activity**—bạn sử dụng một **adapter** để di chuyển nội dung vào màn hình trong **Activity** và một **layout manager** để thay đổi nội dung màn hình tùy theo tab được chọn.

Bạn triển khai **PagerAdapter** để tạo các màn hình mà **ViewPager** hiển thị. **ViewPager** thường được sử dụng cùng với **Fragment**, giúp bạn quản lý vòng đời của một trang màn hình một cách tiện lợi.

Để sử dụng các lớp trong **Android Support Library**, hãy thêm dòng sau vào tệp **build.gradle (Module: app)** trong phần **dependencies**:

com.android.support:design:xx.xx.x (trong đó xx.xx.x là phiên bản mới nhất).

## 1.5 RecycleView

- **FragmentPagerAdapter:** Được thiết kế để điều hướng giữa các màn hình ngang cấp với số lượng màn hình cố định và nhỏ.
- **FragmentStatePagerAdapter:** Được thiết kế để phân trang qua một tập hợp màn hình có số lượng không xác định. Nó sẽ hủy từng **Fragment** khi người dùng chuyển sang màn hình khác để tối ưu hóa bộ nhớ. Ứng dụng trong bài tập này sử dụng **FragmentStatePagerAdapter**

### Bài 2:

#### 2.1 Tạo dự án và bô cục

1. Tạo một dự án mới bằng mẫu **Empty Activity**. Đặt tên ứng dụng là **Tab Experiment**.
2. Chỉnh sửa tệp **build.gradle (Module: app)** và thêm dòng sau vào phần **dependencies** để sử dụng **TabLayout** trong **Android Design Support Library**.

```
implementation 'com.android.support:design:26.1.0'
```

Nếu **Android Studio** đề xuất một phiên bản có số lớn hơn, hãy chỉnh sửa dòng trên để cập nhật phiên bản.

3. Để sử dụng **Toolbar** thay vì thanh ứng dụng (**app bar**) và tiêu đề ứng dụng (**app title**), hãy thêm các thuộc tính sau vào tệp **res > values > styles.xml** để ẩn thanh ứng dụng và tiêu đề.

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
 <!-- Other style attributes -->
 <item name="windowActionBar">false</item>
 <item name="windowNoTitle">true</item>
</style>
```

4. Mở tệp bô cục **activity\_main.xml**, sau đó nhấp vào tab **Text** để xem mã XML.
5. Thay đổi **ConstraintLayout** thành **RelativeLayout**, như đã thực hiện trong các bài tập trước.
6. Thêm thuộc tính **android:id** và **android:padding** với giá trị **16dp** vào **RelativeLayout**.
7. Xóa **TextView** do mẫu cung cấp và thêm **Toolbar**, **TabLayout**, và **ViewPager** bên trong **RelativeLayout** như trong đoạn mã dưới đây.

```
<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/activity_main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:padding="16dp"
 tools:context="com.example.android.tabexperiment.MainActivity">

 <android.support.v7.widget.Toolbar
 android:id="@+id/toolbar"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_alignParentTop="true"
 android:background="?attr/colorPrimary"
 android:minHeight="?attr/actionBarSize"
 android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>

 <android.support.design.widget.TabLayout
 android:id="@+id/tab_layout"

```

```
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_below="@id/toolbar"
 android:background="?attr/colorPrimary"
 android:minHeight="?attr/actionBarSize"
 android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>

 <android.support.v4.view.ViewPager
 android:id="@+id/pager"
 android:layout_width="match_parent"
 android:layout_height="fill_parent"
 android:layout_below="@id/tab_layout"/>

</RelativeLayout>

```

Khi bạn nhập thuộc tính **app:popupTheme** cho **Toolbar**, từ "app" sẽ hiển thị màu đỏ nếu bạn chưa thêm dòng khai báo sau vào **RelativeLayout**:

```
<RelativeLayout xmlns:app="http://schemas.android.com/apk/res-auto"
```

Bạn có thể nhấp vào **app** và nhấn **Option+Enter** (hoặc **Alt+Enter**) để **Android Studio** tự động thêm dòng khai báo cần thiết.

## Bài 2: Trải nghiệm người dùng thú vị

### 2.1 Hình vẽ, định kiểu, và chủ đề

Để thêm một **fragment** đại diện cho mỗi màn hình tab, hãy làm theo các bước sau:

1. Nhập vào **com.example.android.tabexperiment** trong **Android > Project pane**.
2. Chọn **File > New > Fragment > Fragment (Blank)**.
3. Đặt tên cho **fragment** là **TabFragment1**.
4. Chọn tùy chọn **Create layout XML?**.
5. Đổi tên **Fragment Layout Name** của tệp XML thành **tab\_fragment1**.
6. Bỏ chọn các tùy chọn **Include fragment factory methods?** và **Include interface callbacks?** vì bạn không cần các phương thức này.
7. Nhấp vào **Finish**.

Lặp lại các bước trên, thay **TabFragment1** bằng **TabFragment2** và **TabFragment3** ở **Bước 3**, đồng thời thay **tab\_fragment1** bằng **tab\_fragment2** và **tab\_fragment3** ở **Bước 4**.

Mỗi **fragment** được tạo ra với lớp mở rộng từ **Fragment**. Đồng thời, mỗi **Fragment** sẽ nạp (**inflate**) bộ cục tương ứng (**tab\_fragment1**, **tab\_fragment2**, và **tab\_fragment3**) bằng cách sử dụng mô hình **resource-inflate**, mà bạn đã học trong chương trước khi làm việc với **options menu**.

Ví dụ, **TabFragment1** sẽ trông như thế này:

```
public class TabFragment1 extends Fragment {

 public TabFragment1() {
 // Required empty public constructor
 }

 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
 Bundle savedInstanceState) {
 // Inflate the layout for this fragment.
 return inflater.inflate(R.layout.tab_fragment1, container, false);
 }
}
```

### 2.3 Chính sửa bộ cục của Fragment

Chỉnh sửa từng tệp bộ cục **Fragment XML** (**tab\_fragment1.xml**, **tab\_fragment2.xml**, và **tab\_fragment3.xml**):

1. Thay đổi **FrameLayout** thành **RelativeLayout**.
2. Thay đổi nội dung **TextView** thành "These are the top stories:", đồng thời đặt **layout\_width** và **layout\_height** thành **wrap\_content**.
3. Thiết lập kiểu chữ với thuộc tính

Lặp lại các bước trên cho từng tệp **fragment layout XML**, nhập nội dung khác nhau cho **TextView** trong **Bước 2**.

- **Nội dung của TextView trong tab\_fragment2.xml:** "Tech news you can use: "
- **Nội dung của TextView trong tab\_fragment3.xml:** "Cooking tips: "

Kiểm tra từng tệp **fragment layout XML**. Ví dụ, **tab\_fragment1.xml** sẽ trông như thế này:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context="com.example.android.tabexperiment.TabFragment1">

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="These are the top stories: "
 android:textAppearance="?android:attr/textAppearanceLarge"/>

</RelativeLayout>
```

1. Trong tệp **fragment layout XML** **tab\_fragment1.xml**, trích xuất chuỗi "These are the top stories:" thành một tài nguyên chuỗi có tên **tab\_1**. Thực hiện tương tự cho các chuỗi trong **tab\_fragment2.xml** và **tab\_fragment3.xml**.

## 2.2 Thẻ và màu sắc

Mô hình **adapter-layout manager** giúp bạn hiển thị nhiều màn hình nội dung trong một **Activity**:

- Sử dụng **adapter** để điền nội dung vào màn hình hiển thị trong **Activity**.
- Sử dụng **layout manager** để thay đổi màn hình nội dung dựa trên tab được chọn.

Làm theo các bước sau để thêm một lớp **PagerAdapter** mới vào ứng dụng, lớp này mở rộng từ **FragmentStatePagerAdapter** và xác định số lượng tab (**mNumberOfTabs**):

1. Nhập vào **com.example.android.tabexperiment** trong **Android > Project pane**.
2. Chọn **File > New > Java Class**.
3. Đặt tên lớp là **PagerAdapter**, sau đó nhập **FragmentStatePagerAdapter** vào trường **Superclass**. Mục nhập này sẽ tự động thay đổi thành **android.support.v4.app.FragmentStatePagerAdapter**.
4. Giữ nguyên các tùy chọn **Public** và **None**, sau đó nhấp vào **OK**.
5. Mở **PagerAdapter** trong **Project > Android** pane. Một biểu tượng **đèn đỏ** sẽ xuất hiện bên cạnh định nghĩa lớp. Nhấp vào đèn và chọn **Implement methods**, sau đó nhấn **OK** để triển khai các phương thức **getItem()** và **getCount()** đã được chọn sẵn.
6. Một biểu tượng **đèn đỏ** khác sẽ xuất hiện bên cạnh định nghĩa lớp. Nhấp vào đèn và chọn **Create constructor matching super**.

7. Thêm một biến thành viên kiểu **int** có tên **mNumOfTabs**, sau đó chỉnh sửa constructor để sử dụng nó. Mã nguồn lúc này sẽ trông như sau:

```
public class PagerAdapter extends FragmentStatePagerAdapter {
 int mNumOfTabs;

 public PagerAdapter(FragmentManager fm, int NumOfTabs) {
 super(fm);
 this.mNumOfTabs = NumOfTabs;
 }

 /**
 * Return the Fragment associated with a specified position.
 *
 * @param position
 */
 @Override
 public Fragment getItem(int position) {
 return null;
 }

 /**
 * Return the number of views available.
 */
 @Override
 public int getCount() {
 return 0;
 }
}
```

Khi nhập mã trên, **Android Studio** sẽ tự động nhập các thư viện sau:

```
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
```

Nếu **FragmentManager** trong mã hiển thị màu đỏ, một biểu tượng **đèn đỏ** sẽ xuất hiện khi bạn nhấp vào nó. Nhấp vào biểu tượng **đèn đỏ** và chọn **Import class**. Các tùy chọn nhập sẽ xuất hiện. Chọn **FragmentManager (android.support.v4)**.

8. Thay đổi phương thức **getItem()** vừa thêm thành đoạn mã sau, sử dụng **switch-case** để trả về **Fragment** tương ứng dựa trên tab được chọn:

```
@Override
public Fragment getItem(int position) {
 switch (position) {
 case 0: return new TabFragment1();
 case 1: return new TabFragment2();
 case 2: return new TabFragment3();
 default: return null;
 }
}
```

9. Thay đổi phương thức **getCount()** vừa thêm thành đoạn mã sau để trả về số lượng tab:

#### 2.4 Nạp (Inflate) Toolbar và TabLayout

Vì bạn đang sử dụng các **tab** nằm bên dưới **app bar**, bạn đã thiết lập **app bar** và **Toolbar** trong tệp **activity\_main.xml** ở bước đầu tiên. Nay, bạn cần **nạp Toolbar** (sử dụng phương pháp tương tự như trong chương trước về **options menu**) và tạo một **TabLayout** để định vị các tab.

1. Mở **MainActivity** và thêm đoạn mã sau vào phương thức **onCreate()** để nạp **Toolbar** bằng **setSupportActionBar()**:

```
protected void onCreate(Bundle savedInstanceState) {
 // ... Code inside onCreate() method
 android.support.v7.widget.Toolbar toolbar =
 findViewById(R.id.toolbar);
 setSupportActionBar(toolbar);
 // Create an instance of the tab layout from the view.
}
```

2. Mở tệp **strings.xml**, và tạo các tài nguyên chuỗi (**string resources**) sau:

```
<string name="tab_label1">Top Stories</string>
<string name="tab_label2">Tech News</string>
<string name="tab_label3">Cooking</string>
```

3. Ở cuối phương thức **onCreate()**, tạo một thẻ hiện của **TabLayout** từ phần tử **tab\_layout** trong bố cục, và đặt văn bản cho từng tab bằng **addTab()**.

```
// Create an instance of the tab layout from the view.
TabLayout tabLayout = findViewById(R.id.tab_layout);
// Set the text for each tab.
tabLayout.addTab(tabLayout.newTab().setText(R.string.tab_label1));
tabLayout.addTab(tabLayout.newTab().setText(R.string.tab_label2));
tabLayout.addTab(tabLayout.newTab().setText(R.string.tab_label3));
```

```
// Set the tabs to fill the entire layout.
tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);
// Use PagerAdapter to manage page views in fragments.
```

## 2.3 Bố cục và thích ứng

- Ở bên dưới đoạn mã bạn đã thêm vào phương thức **onCreate()** ở nhiệm vụ trước, thêm đoạn mã sau để sử dụng **PagerAdapter** nhằm quản lý các màn hình (**page views**) trong các **Fragment**:

```
// Use PagerAdapter to manage page views in fragments.
// Each page is represented by its own fragment.
final ViewPager viewPager = findViewById(R.id.pager);
final PagerAdapter adapter = new PagerAdapter
 (getSupportFragmentManager(), tabLayout.getTabCount());
viewPager.setAdapter(adapter);
// Setting a listener for clicks.
```

- Ở cuối phương thức **onCreate()**, thiết lập một **listener** (**TabLayoutOnPageChangeListener**) để phát hiện khi một **tab** được nhấp vào.

Sau đó, tạo phương thức **onTabSelected()** để đặt **ViewPager** hiển thị màn hình tab tương ứng.

Mã nguồn sẽ trông như sau:

```
// Setting a listener for clicks.
viewPager.addOnPageChangeListener(new
 TabLayout.TabLayoutOnPageChangeListener(tabLayout));
tabLayout.addOnTabSelectedListener(new
 TabLayout.OnTabSelectedListener() {
 @Override
 public void onTabSelected(TabLayout.Tab tab) {
 viewPager.setCurrentItem(tab.getPosition());
 }

 @Override
 public void onTabUnselected(TabLayout.Tab tab) {
 }
 });
```

```
@Override
public void onTabReselected(TabLayout.Tab tab) {
}
});
```

2. Chạy ứng dụng. Nhấn vào từng **tab** để xem từng "trang" (màn hình). Bạn cũng có thể **vuốt trái** và **vuốt phải** để chuyển giữa các trang khác nhau.

## Thử thách lập trình

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

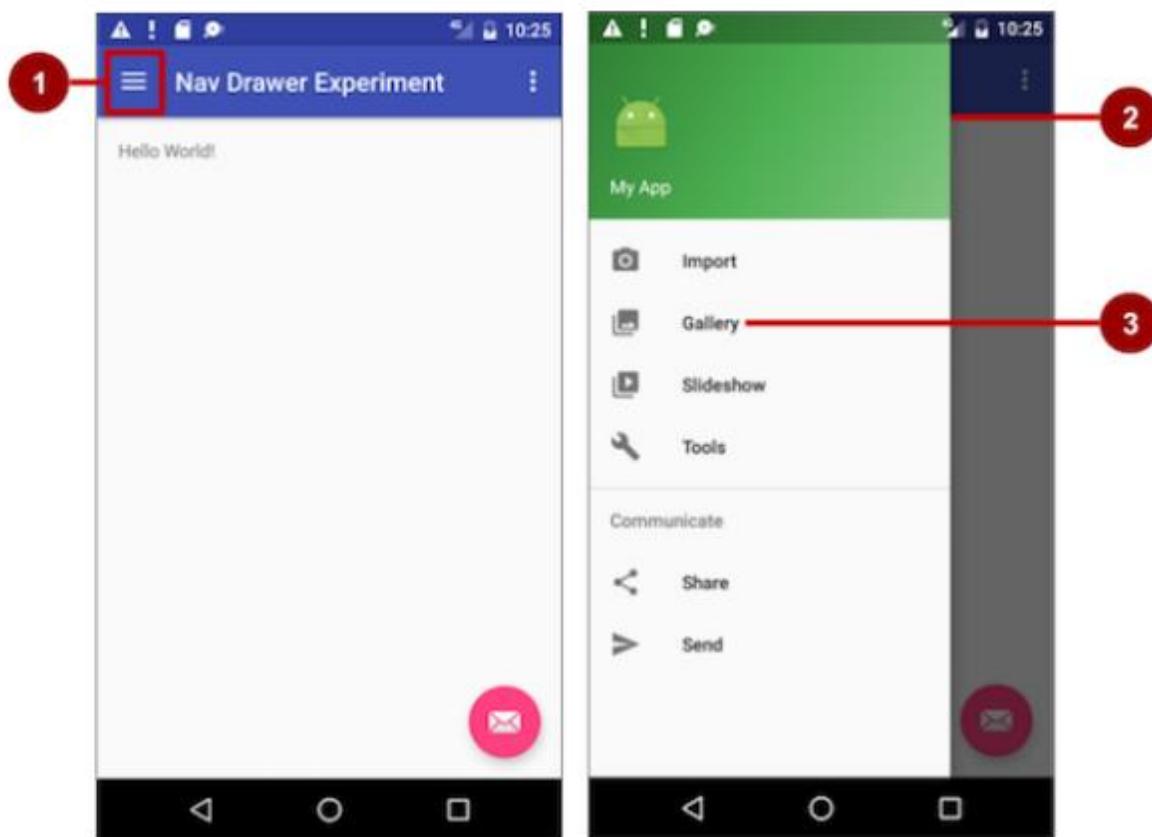
### Thử thách:

Tạo một ứng dụng mới với **navigation drawer**. Khi người dùng chọn một mục trong **navigation drawer**, đóng **drawer** và hiển thị **Toast message** với nội dung cho biết mục nào đã được chọn.

### Mô tả:

**Navigation drawer** là một **bảng điều hướng** hiển thị các tùy chọn ở cạnh trái của màn hình. Nó thường được ẩn và chỉ hiển thị khi:

- Người dùng **vuốt từ cạnh trái** của màn hình.
- Người dùng **nhấn vào biểu tượng điều hướng** trong app bar.



Trong hình minh họa trên:

1. **Biểu tượng điều hướng** trong app bar.

2. **Navigation drawer.**
3. **Mục menu** trong **navigation drawer**.

Các bước để tạo navigation drawer trong ứng dụng:

**Tạo các tệp bố cục (layout):**

- **Navigation drawer** làm **ViewGroup** gốc trong bố cục của Activity.
- **Navigation View** để hiển thị các mục trong **drawer**.
- **App bar layout** chứa **nút biểu tượng điều hướng**.
- **Content layout** hiển thị nội dung chính của Activity.
- **Header layout** cho **navigation drawer**.

**Cấu hình navigation drawer:**

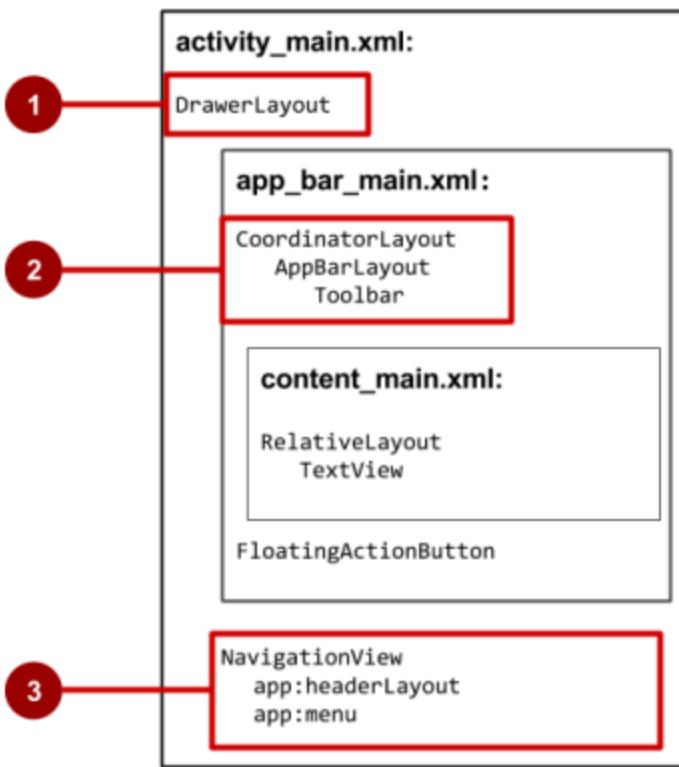
- **Thêm các mục menu** vào **navigation drawer**, bao gồm tiêu đề và biểu tượng.
- **Thiết lập navigation drawer** trong mã Activity.
- **Thêm listener** để xử lý sự kiện khi người dùng chọn một mục trong **drawer**.
- **Xử lý lựa chọn mục** trong **menu điều hướng**

Để tạo **navigation drawer layout**, sử dụng **DrawerLayout APIs** có sẵn trong **Android Support Library**. Khi thiết kế, hãy tuân theo **nguyên tắc thiết kế navigation drawer** trong **Navigation Drawer Design Guide**.

Các bước để thêm Navigation Drawer

1. **Sử dụng DrawerLayout làm ViewGroup gốc** trong bố cục Activity.
2. Bên trong DrawerLayout, thêm:
  - a. **Một View chứa nội dung chính** (hiển thị khi drawer ẩn).
  - b. **Một NavigationView** chứa menu của drawer.
3. **Sử dụng thẻ <include>** để tái sử dụng bộ cục XML, giúp mã dễ đọc hơn.
4. **Xử lý sự kiện khi chọn một mục** trong **navigation menu** để thực hiện hành động tương ứng.

Hình minh họa bên dưới sẽ giúp bạn hình dung cấu trúc của **activity\_main.xml** và cách nó bao gồm các bộ cục con.



In the figure above:

Trong hình minh họa trên:

1. **DrawerLayout** là View gốc của bố cục Activity.
2. **app\_bar\_main.xml** (được include) sử dụng **CoordinatorLayout** làm View gốc, định nghĩa **app bar layout** với **Toolbar**, trong đó có **nút điều hướng** để mở drawer.
3. **NavigationView** định nghĩa bố cục **navigation drawer**, bao gồm **header** và các mục menu.

Tóm tắt về điều hướng trong app bar:

- Thêm nút "Up" để điều hướng về **Activity cha** bằng cách khai báo trong **AndroidManifest.xml**.
- Khai báo **Activity cha** bên trong phần `<activity>...</activity>` của Activity con.

```

<activity>
 <meta-data android:name="android.support.PARENT_ACTIVITY"
 android:value=".MainActivity" />

```

Điều hướng bằng Tab:

- ✓ **Tab** là giải pháp tốt cho “điều hướng ngang” giữa các màn hình cùng cấp (**sibling views**).
- ✓ **TabLayout** là lớp chính dùng để tạo tab, có trong **Android Design Support Library**.
- ✓ **ViewPager** là trình quản lý bố cục, cho phép vuốt trái/phải để chuyển đổi giữa các trang dữ liệu.

Thường được dùng cùng với **Fragment**.

✓ **Sử dụng một trong hai adapter chuẩn** cho ViewPager:

- **FragmentPagerAdapter**: Dùng khi số lượng màn hình cố định, dữ liệu ít thay đổi.
- **FragmentStatePagerAdapter**: Dùng khi có số lượng màn hình **lớn hoặc không xác định**, tiết kiệm bộ nhớ bằng cách hủy Fragment khi không cần thiết.

Tìm hiểu thêm tài liệu dành cho nhà phát triển Android:

- **Giao diện người dùng & Điều hướng**
- **Thiết kế điều hướng hiệu quả**
- **Triển khai điều hướng hiệu quả**
- **Tạo chế độ xem vuốt với tab**
- **Tạo ngăn điều hướng**
- **Thiết kế điều hướng Quay lại và Lên trên**
- **Cung cấp điều hướng Lên trên**
- **Triển khai điều hướng con cháu**
- **TabLayout**
- **Ngăn điều hướng (Navigation Drawer)**
- **DrawerLayout**
- **Thư viện Hỗ trợ**

Thông số kỹ thuật Material Design:

- **Hiểu về điều hướng**
- **Lưới bố cục đáp ứng**

Blog của Android Developers:

- **Thư viện hỗ trợ thiết kế Android**

Khác:

- **AndroidHive: Thiết kế Material Design trên Android làm việc với Tab**
- **Truiton: Ví dụ về Tab trên Android – Với Fragment và ViewPager**

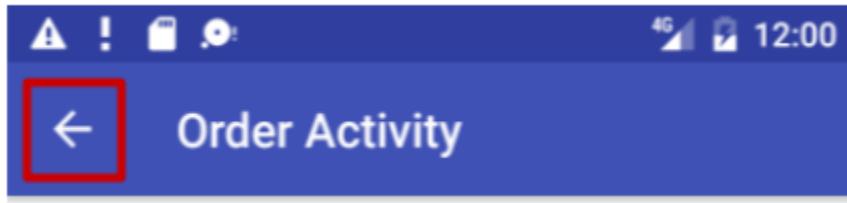
Bài tập về nhà

### Xây dựng và chạy ứng dụng

Tạo một ứng dụng với một **MainActivity** và ít nhất **ba Activity con** khác.

Mỗi Activity phải có một **menu tùy chọn** và sử dụng **Toolbar** từ **thư viện hỗ trợ v7 appcompat** làm thanh ứng dụng, như được hiển thị bên dưới.

- Trong **MainActivity**, xây dựng một bố cục dạng **lưới** (**GridLayout**) với các hình ảnh do bạn tự chọn. Ba hình ảnh mẫu (**donut\_circle.png**, **froyo\_circle.png**, và **icecream\_circle.png**) có sẵn và có thể tải xuống từ ứng dụng **DroidCafe**.
- Chỉnh kích thước hình ảnh** nếu cần thiết để ba hình ảnh có thể hiển thị vừa **trên cùng một hàng** trong bố cục lưới.
- Kích hoạt điều hướng** từ mỗi hình ảnh đến một **Activity con**. Khi người dùng **chạm vào hình ảnh**, ứng dụng sẽ mở một **Activity con tương ứng**. Từ mỗi Activity con, người dùng có thể **bấm nút Lên trên (Up button) trên thanh ứng dụng (app bar)** để quay lại **MainActivity**.



Câu hỏi 1:

Template nào cung cấp một Activity với menu tùy chọn (options menu) và thư viện hỗ trợ v7 appcompat Toolbar làm thanh ứng dụng (app bar)?

Đáp án đúng: Basic Activity template

Giải thích:

- Basic Activity template** tạo một Activity có **Toolbar** làm thanh ứng dụng (App Bar) và tích hợp sẵn menu tùy chọn.
- Nó cũng sử dụng thư viện **v7 appcompat** để đảm bảo khả năng tương thích với các phiên bản Android cũ hơn.

Câu hỏi 2:

Bạn cần thêm dependency nào để sử dụng TabLayout?

Đáp án đúng: com.android.support:design

Giải thích:

- TabLayout** là một thành phần trong **Android Design Support Library**.
- Để sử dụng **TabLayout**, bạn cần thêm thư viện **com.android.support:design**, thư viện này cũng bao gồm nhiều thành phần giao diện khác như **FloatingActionButton**, **NavigationView**, **BottomNavigationView**,...

Câu hỏi 3:

**Bạn cần khai báo mỗi Activity con và Activity cha ở đâu để cung cấp chức năng điều hướng Up?**

**Đáp án đúng:**

**"To provide the Up button for a child screen Activity, declare the parent Activity in the child Activity section of the AndroidManifest.xml file."**

**Giải thích:**

- Để bật tính năng điều hướng Up (nút mũi tên quay lại trên thanh công cụ), bạn cần khai báo **parentActivityName** trong **AndroidManifest.xml** cho mỗi Activity con.
- Điều này giúp hệ thống biết Activity cha của Activity con là gì và hiển thị nút Up tự động.

Hướng dẫn chấm điểm ứng dụng:

Ứng dụng cần có các tính năng sau để đạt yêu cầu:

- ✓ **GridLayout trong content\_main.xml** (bố cục dạng lưới).
- ✓ **Mỗi phần tử điều hướng trong lưới cần có Intent và startActivity() để mở Activity mới.**
- ✓ **Mỗi phần tử điều hướng cần có một Activity riêng biệt.**

Bài 4.5: RecyclerView

**Giới thiệu**

Cho phép người dùng hiển thị, cuộn và thao tác với một danh sách các mục dữ liệu tương tự là một tính năng phổ biến trong ứng dụng.

Các ví dụ về danh sách có thể cuộn bao gồm:

- Danh bạ
- Danh sách phát nhạc
- Trò chơi đã lưu
- Thư viện ảnh
- Từ điển
- Danh sách mua sắm
- Mục lục tài liệu

Trong thực hành về chế độ xem cuộn, bạn sử dụng **ScrollView** để cuộn một **View hoặc ViewGroup**. **ScrollView** dễ sử dụng, nhưng **không được khuyến nghị cho danh sách dài**.

RecyclerView là một lớp con của **ViewGroup** và là cách hiển thị danh sách có thể cuộn một cách tối ưu hơn.

Thay vì tạo một View riêng cho từng mục, kể cả những mục không hiển thị trên màn hình, **RecyclerView chỉ tạo một số lượng giới hạn mục danh sách và tái sử dụng chúng** cho nội dung hiển thị.

**Trong bài thực hành này, bạn sẽ:**

- ✓ Sử dụng **RecyclerView** để hiển thị một danh sách có thể cuộn.
- ✓ Thêm sự kiện nhấp vào từng mục trong danh sách.
- ✓ Thêm mục vào danh sách bằng **Floating Action Button (FAB)** (nút màu hồng trong ảnh chụp màn hình).
- ✓ **FAB** được dùng để thực hiện hành động chính mà bạn muốn người dùng thực hiện.

Những gì bạn cần biết trước:

Bạn cần có kỹ năng:

Tạo và chạy ứng dụng trong **Android Studio**.

Tạo và chỉnh sửa phần tử UI bằng **layout editor**, viết trực tiếp XML, và truy cập phần tử từ mã Java.

Tạo và sử dụng **string resources**.

Chuyển đổi nội dung trong một **View** thành chuỗi bằng `getText()`.

Thêm sự kiện `onClick()` vào một **View**.

Hiển thị **Toast message**.

Những gì bạn sẽ học:

Cách sử dụng **RecyclerView** để hiển thị danh sách có thể cuộn.

Cách thêm các mục vào **RecyclerView** khi chúng xuất hiện trong quá trình cuộn.

Cách xử lý sự kiện khi người dùng nhấn vào một mục cụ thể.

Cách hiển thị **FAB** và thực hiện hành động khi người dùng nhấn vào nó.

Những gì bạn sẽ làm:

- ✓ **Tạo một ứng dụng mới** sử dụng **RecyclerView** để hiển thị danh sách mục có thể cuộn và gán hành vi khi người dùng nhấp vào các mục trong danh sách.
- ✓ **Sử dụng FAB** để cho phép người dùng thêm mục mới vào **RecyclerView**.

Tổng quan về ứng dụng:

Ứng dụng **RecyclerView** sẽ minh họa cách sử dụng **RecyclerView** để hiển thị một danh sách dài các từ có thể cuộn.

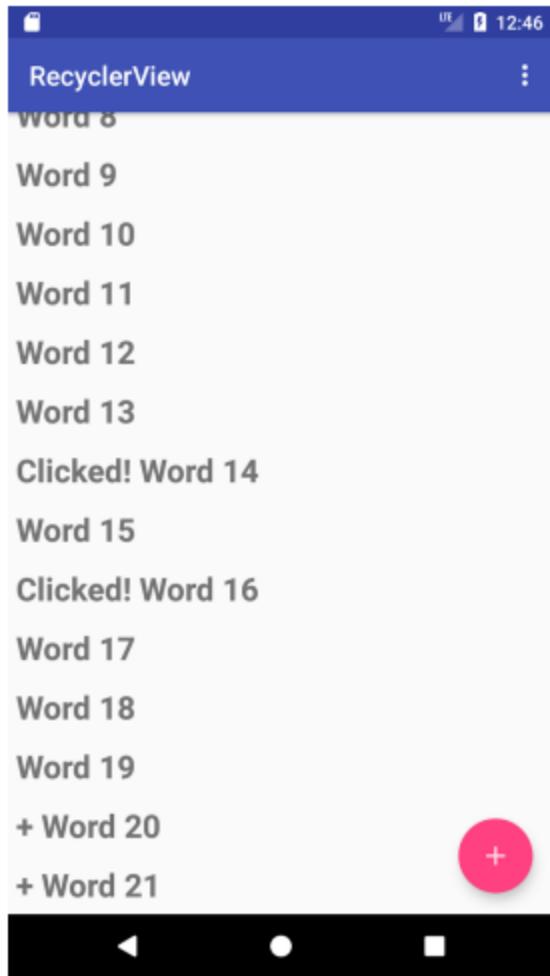
Bạn sẽ tạo:

**Tập dữ liệu** (các từ trong danh sách).

**RecyclerView** để hiển thị danh sách.

**Các hành động của người dùng**, bao gồm:

- Nhấn vào một từ để đánh dấu đã chọn.
- Nhấn vào **FAB** để thêm một từ mới vào danh sách.



### Nhiệm vụ 1: Tạo một dự án mới và tập dữ liệu

Trước khi hiển thị **RecyclerView**, bạn cần có dữ liệu để hiển thị. Trong nhiệm vụ này, bạn sẽ:

- ✓ **Tạo một dự án mới** cho ứng dụng.
- ✓ **Tạo tập dữ liệu** để hiển thị trong **RecyclerView**.

Giải thích:

Trong các ứng dụng phức tạp hơn, dữ liệu có thể đến từ:

- Bộ nhớ trong** (tệp, cơ sở dữ liệu SQLite, SharedPreferences).
- Ứng dụng khác** (Danh bạ,Ảnh).
- Internet** (lưu trữ đám mây, Google Sheets, API).

**Lưu trữ và truy xuất dữ liệu là một chủ đề riêng**, được đề cập trong chương **Lưu trữ dữ liệu**.

Trong bài thực hành này, bạn sẽ **mô phỏng dữ liệu** bằng cách tạo nó trực tiếp trong phương thức `onCreate()` của **MainActivity**.

Tạo dự án và bộ cục

## 1. Mở Android Studio.

2. Tạo một dự án mới với tên **RecyclerView**, chọn mẫu **Basic Activity** và tạo tệp bố cục.

- **Basic Activity** (được giới thiệu trong chương về sử dụng hình ảnh có thể nháp) cung cấp:

✓ Floating Action Button (FAB).

✓ App bar trong bố cục của Activity (activity\_main.xml).

✓ **Bố cục nội dung của Activity** (content\_main.xml)

## 3. Chạy ứng dụng.

- Bạn sẽ thấy **tiêu đề ứng dụng RecyclerView** và dòng chữ "**Hello World**" trên màn hình.
- Nếu gặp lỗi liên quan đến **Gradle**, hãy đồng bộ dự án theo hướng dẫn trong bài thực hành về **cài đặt Android Studio và chạy ứng dụng Hello World**.

Thêm mã để tạo dữ liệu

Trong bước này, bạn sẽ tạo một **LinkedList** gồm 20 chuỗi từ, mỗi từ kết thúc bằng một số tăng dần, ví dụ: ["Word 1", "Word 2", "Word 3", ...].

Các bước thực hiện:

1. **Mở tệp MainActivity.** Và **Thêm một biến thành viên riêng tư (private)** cho danh sách liên kết mWordList.

```
public class MainActivity extends AppCompatActivity {
 private final LinkedList<String> mWordList = new LinkedList<>();
 // ... Rest of MainActivity code ...
}
```

2. Thêm đoạn mã sau vào **phương thức onCreate()** để điền dữ liệu vào danh sách mWordList:

```
// Put initial data into the word list.
for (int i = 0; i < 20; i++) {
 mWordList.addLast("Word " + i);
}
```

Đoạn mã nối chuỗi "Word " với giá trị của i trong khi giá trị này tăng dần. Đây là tất cả những gì bạn cần làm để tạo tập dữ liệu cho bài thực hành này.

## Bài 3: Kiểm thử giao diện người dùng

## Espresso cho việc kiểm tra UI

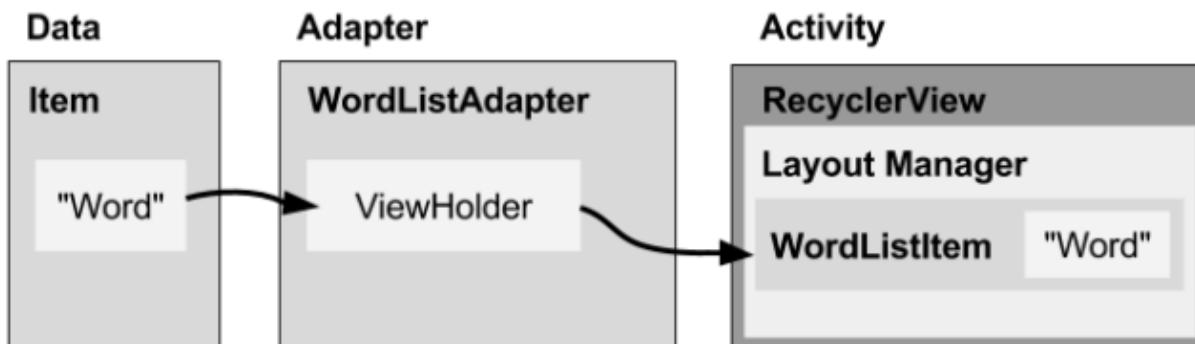
1. Mở Project > Android, mở rộng thư mục res, nhấp chuột phải vào thư mục **drawable**.
2. Chọn New > Image Asset. Hộp thoại **Configure Image Asset** sẽ xuất hiện.
3. Trong menu thả xuống ở đầu hộp thoại, chọn **Action Bar and Tab Items**.
4. Trong trường **Name**, đổi tên ic\_action\_name thành **ic\_add\_for\_fab**.
5. Nhấp vào hình **clip art** (biểu tượng Android bên cạnh **Clipart:**), chọn một biểu tượng để sử dụng cho FAB (ví dụ: dấu cộng (+)).
6. Chọn **HOLO\_DARK** từ menu **Theme** để biểu tượng có màu trắng trên nền tối hoặc đen. Nhấp **Next**.
7. Nhấp **Finish** trong hộp thoại **Confirm Icon Path** để hoàn tất.

## Task 2: Tạo một RecyclerView

Trong bài thực hành này, bạn sẽ hiển thị dữ liệu trong **RecyclerView**. Để làm được điều này, bạn cần:

- **Dữ liệu để hiển thị:** Sử dụng danh sách mWordList.
- **RecyclerView:** Danh sách cuộn chứa các mục dữ liệu.
- **Layout cho một mục dữ liệu:** Tất cả các mục trong danh sách có giao diện giống nhau.
- **Layout Manager:** RecyclerView.LayoutManager quản lý cấu trúc và bố cục của các phần tử View.
  - RecyclerView yêu cầu một **layout manager** rõ ràng để sắp xếp các mục trong danh sách.
  - Bố cục có thể là **dọc, ngang hoặc dạng lượn**.
  - Trong bài này, bạn sẽ sử dụng **LinearLayoutManager theo chiều dọc**.
- **Adapter:** RecyclerView.Adapter kết nối dữ liệu với RecyclerView.
  - Chuẩn bị dữ liệu trong RecyclerView.ViewHolder.
  - Bạn sẽ tạo một adapter để **thêm và cập nhật danh sách** từ trong RecyclerView.
- **ViewHolder:**
  - Được tạo bên trong adapter.
  - Chứa thông tin về **View** để hiển thị một mục dữ liệu từ layout của nó.

Sơ đồ bên dưới minh họa mối quan hệ giữa **dữ liệu, adapter, ViewHolder và layout manager** trong RecyclerView.



Các bước triển khai RecyclerView

1. **Thêm một phần tử RecyclerView vào tệp giao diện content\_main.xml** trong ứng dụng.
2. **Tạo một tệp XML (wordlist\_item.xml)** để xác định bố cục của từng mục trong danh sách (WordListItem).
3. **Tạo một adapter (WordListAdapter)** với một ViewHolder (WordViewHolder).
  - a. Triển khai phương thức **lấy dữ liệu**, đặt vào ViewHolder và thông báo cho layout manager hiển thị nó.
4. **Trong phương thức onCreate() của MainActivity**, tạo một RecyclerView và khởi tạo nó với:
  - a. **Adapter (WordListAdapter)** để hiển thị dữ liệu.
  - b. **Layout manager (LinearLayoutManager)** để sắp xếp danh sách theo chiều dọc.

Bạn sẽ thực hiện từng bước một để hoàn thành bài thực hành này.

### Chỉnh sửa bố cục trong content\_main.xml

1. Mở **content\_main.xml** trong ứng dụng **RecyclerView**. Tệp này chứa một **TextView** hiển thị "**Hello World**" ở trung tâm của ConstraintLayout.
2. Nhập vào **tab "Text"** để hiển thị mã XML.
3. Thay thế toàn bộ phần tử **TextView** bằng đoạn mã sau:

```
<android.support.v7.widget.RecyclerView
 android:id="@+id/recyclerview"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
</android.support.v7.widget.RecyclerView>
```

Bạn

cần chỉ định **đường dẫn đầy đủ**(android.support.v7.widget.RecyclerView), vì **RecyclerView** là một phần của **Support Library** )

Tạo bố cục cho một mục trong danh sách

**Adapter** cần một bố cục cho từng mục trong danh sách. Vì tất cả các mục đều có giao diện giống nhau, bạn phải tạo một **tệp tài nguyên bố cục riêng** để adapter sử dụng **tách biệt** khỏi RecyclerView.

### Các bước thực hiện

1. Nhấp chuột phải vào **app > res > layout**, chọn **New > Layout resource file**.
2. Đặt tên tệp là **wordlist\_item.xml**, sau đó nhấp **OK**.
3. Trong tệp bố cục mới, chuyển sang **tab "Text"** để hiển thị mã XML.
4. Thay thế **ConstraintLayout** được tạo mặc định bằng **LinearLayout** theo chiều dọc (vertical) với các thuộc tính sau (trích xuất tài nguyên khi cần thiết).

LinearLayout attribute	Value
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:orientation	"vertical"
android:padding	"6dp"

Thêm một TextView vào LinearLayout để hiển thị từ trong danh sách. Đặt ID cho TextView là word để adapter có thể truy cập và cập nhật nội dung sau này.

Attribute	Value
android:id	"@+id/word"
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:textSize	"24sp"
android:textStyle	"bold"

Tạo một style từ thuộc tính của TextView

Bạn có thể sử dụng **styles** để chia sẻ nhóm thuộc tính hiển thị giữa các phần tử. Một cách đơn giản để tạo style là **trích xuất** phong cách của một phần tử giao diện người dùng đã tạo.

#### Các bước trích xuất style cho TextView trong wordlist\_item.xml

1. Mở wordlist\_item.xml nếu chưa mở.
2. Nhấp chuột phải (**hoặc Control + Click**) vào **TextView** vừa tạo trong wordlist\_item.xml, chọn **Refactor > Extract > Style**.
  - a. Hộp thoại **Extract Android Style** sẽ xuất hiện.
3. Đặt tên cho style là **word\_title** và giữ nguyên tất cả các tùy chọn mặc định.
  - a. Chọn **Launch 'Use Style Where Possible'**.
  - b. Nhấp **OK**.
4. Khi được nhắc, áp dụng style cho **toàn bộ dự án (Whole Project)**.
5. Tìm và kiểm tra style **word\_title** trong **values > styles.xml**.
6. Mở lại **wordlist\_item.xml** nếu chưa mở.
  - a. TextView bây giờ sẽ sử dụng **style** thay vì các thuộc tính riêng lẻ.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical"
 android:padding="6dp">

 <TextView
 android:id="@+id/word"
 style="@style/word_title" />

</LinearLayout>
```

## Tạo một adapter

Android sử dụng **adapter** (từ lớp Adapter) để kết nối dữ liệu với các phần tử **View** trong danh sách. Có nhiều loại adapter khác nhau, và bạn cũng có thể viết adapter tùy chỉnh.

Trong nhiệm vụ này, bạn sẽ tạo một **adapter** liên kết danh sách từ (word list) với các phần tử View trong danh sách.

### Cách hoạt động của Adapter

Để kết nối dữ liệu với các phần tử View, adapter cần biết về các **View items**. Adapter sử dụng một **ViewHolder** để mô tả một mục **View** và vị trí của nó trong **RecyclerView**.

### Các bước thực hiện

1. Nhấp chuột phải vào **java/com.android.example.recyclerview**, chọn **New > Java Class**.
2. Đặt tên cho lớp là **WordListAdapter**.
3. Định nghĩa WordListAdapter với chữ ký sau:

```
public class WordListAdapter extends
 RecyclerView.Adapter<WordListAdapter.WordViewHolder> {}
```

#### WordListAdapter mở rộng một adapter chung cho RecyclerView

Lớp **WordListAdapter** mở rộng một **adapter chung** cho **RecyclerView** để sử dụng một **ViewHolder** được xác định bên trong **WordListAdapter** và phù hợp với ứng dụng của bạn. **WordViewHolder** hiện đang báo lỗi vì nó chưa được định nghĩa.

Nhấp vào **khai báo** lớp **WordListAdapter**. Nhấp vào **biểu tượng bóng đèn đỏ** ở bên trái của khung. Chọn **Implement methods**. Một hộp thoại xuất hiện yêu cầu bạn chọn các phương thức để triển khai. Chọn cả **ba phương thức** và nhấp **OK**. **Android Studio** sẽ tạo ra các phương thức trống. Lưu ý rằng: `onCreateViewHolder` và `onBindViewHolder` đều tham chiếu đến **WordViewHolder**, nhưng **WordViewHolder** chưa được triển khai.

## Tạo ViewHolder cho Adapter

Các bước thực hiện:

1. Bên trong lớp WordListAdapter, thêm một lớp nội bộ (inner class) mới với chữ ký sau:

```
class WordViewHolder extends RecyclerView.ViewHolder {}
```

Bạn sẽ thấy một lỗi về việc thiếu constructor mặc định. Bạn có thể xem chi tiết lỗi bằng cách di chuột qua đoạn mã có gạch đỏ hoặc qua bất kỳ đường gạch đỏ nào ở lề phải của trình chỉnh sửa.

- 2: Thêm các biến vào lớp **WordViewHolder** (lớp lòng trong) để lưu **TextView** và **adapter**.

```
public final TextView wordItemView;
final WordListAdapter mAdapter;
```

- 3: Trong lớp lòng **WordViewHolder**, thêm một constructor để khởi tạo **TextView** từ tệp giao diện XML và thiết lập **adapter**.

```
public WordViewHolder(View itemView, WordListAdapter adapter) {
 super(itemView);
 wordItemView = itemView.findViewById(R.id.word);
 this.mAdapter = adapter;
}
```

4. Chạy ứng dụng của bạn để đảm bảo không có lỗi. Bạn vẫn sẽ chỉ thấy một màn hình trống.

5. Nhấp vào tab **Logcat** để xem bảng **Logcat**, và lưu ý cảnh báo:

**E/RecyclerView: No adapter attached; skipping layout** Bạn sẽ gán adapter cho **RecyclerView** ở bước tiếp theo.

## 2.6 Lưu trữ dữ liệu trong adapter

Bạn cần lưu dữ liệu trong **adapter**, và **WordListAdapter** cần một constructor để khởi tạo danh sách từ dữ liệu. Thực hiện các bước sau:

1. Để lưu dữ liệu trong adapter, tạo một danh sách liên kết **private** chứa các chuỗi trong **WordListAdapter** và đặt tên nó là **mWordList**.

```
private final LinkedList<String> mWordList;
```

2. Nay giờ, bạn có thể điền vào phương thức **getCount()** để trả về kích thước của **mWordList**:

```
@Override
public int getCount() {
 return mWordList.size();
}
```

3. **WordListAdapter** cần một constructor để khởi tạo danh sách từ dữ liệu.

- o Để tạo View cho một mục trong danh sách, **WordListAdapter** cần inflate tệp XML của danh sách.
- o Bạn sử dụng **LayoutInflater** để đọc và chuyển đổi tệp XML thành các View tương ứng.
- o Bắt đầu bằng cách tạo một biến thành viên **mInflater** trong **WordListAdapter**:

```
private LayoutInflater mInflater;
```

#### 4: Cài đặt constructor cho **WordListAdapter**

- Constructor cần có một tham số **context** và một danh sách liên kết chứa các từ.
- Trong phương thức này, khởi tạo **LayoutInflater** cho **mInflater** và gán **mWordList** bằng dữ liệu được truyền vào.

```
public WordListAdapter(Context context,
 LinkedList<String> wordList) {
 mInflater = LayoutInflater.from(context);
 this.mWordList = wordList;
}
```

#### 5: Hoàn thành phương thức **onCreateViewHolder()**

- Phương thức này hoạt động giống **onCreate()**, dùng để inflate tệp giao diện của một mục trong danh sách.
- Nó trả về một **ViewHolder** chứa giao diện và adapter.

```
@Override
public WordViewHolder onCreateViewHolder(ViewGroup parent,
 int viewType) {
 View mItemView = mInflater.inflate(R.layout.wordlist_item,
 parent, false);
 return new WordViewHolder(mItemView, this);
}
```

#### 6: Hoàn thành phương thức **onBindViewHolder()**

- Phương thức này kết nối dữ liệu với **ViewHolder**, thiết lập nội dung hiển thị cho từng mục trong danh sách.

```
@Override
public void onBindViewHolder(WordViewHolder holder, int position) {
 String mCurrent = mWordList.get(position);
 holder.wordItemView.setText(mCurrent);
}
```

7: Chạy ứng dụng của bạn để đảm bảo không có lỗi.

## 2.7. Tạo RecyclerView trong Activity

Bây giờ bạn đã có một adapter với ViewHolder, cuối cùng bạn có thể tạo RecyclerView và kết nối tất cả các phần để hiển thị dữ liệu của bạn.

1. Mở MainActivity .

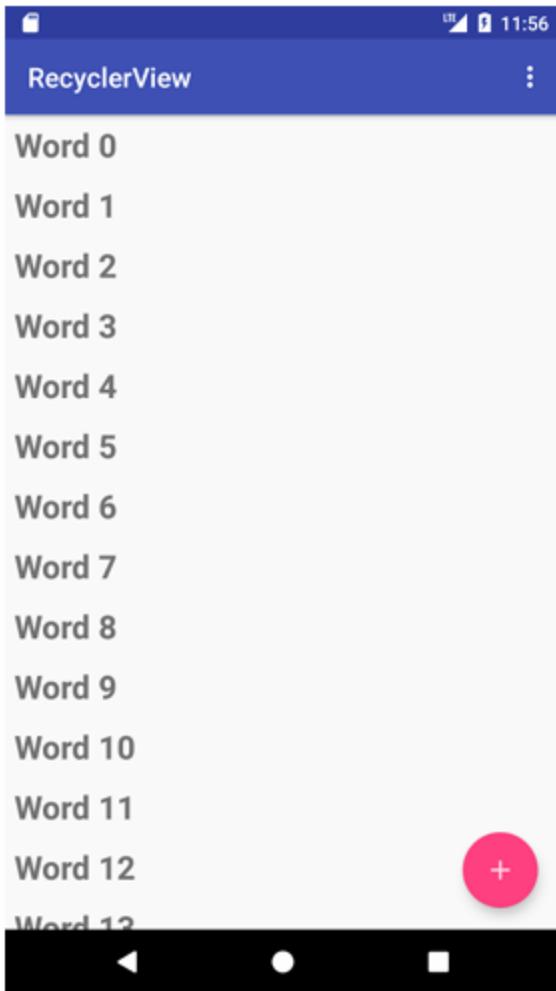
2. Thêm các biến thành viên cho RecyclerView và bộ điều hợp.

```
private RecyclerView mRecyclerView;
private WordListAdapter mAdapter;
```

3. Trong phương thức onCreate() của MainActivity , thêm mã sau để tạo RecyclerView và kết nối nó với bộ điều hợp và dữ liệu. Các nhận xét giải thích từng dòng. Bạn phải chèn mã này sau khi khởi tạo mWordList.

```
// Get a handle to the RecyclerView.
mRecyclerView = findViewById(R.id.recyclerview);
// Create an adapter and supply the data to be displayed.
mAdapter = new WordListAdapter(this, mWordList);
// Connect the adapter with the RecyclerView.
mRecyclerView.setAdapter(mAdapter);
// Give the RecyclerView a default layout manager.
mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
```

4. Chạy ứng dụng của bạn. Bạn sẽ thấy danh sách các từ của mình được hiển thị và bạn có thể cuộn danh sách.



### Nhiệm vụ 3: Làm cho danh sách tương tác

Nhìn vào danh sách các mục rất thú vị, nhưng sẽ thú vị và hữu ích hơn rất nhiều nếu người dùng của bạn có thể tương tác với chúng. Để xem RecyclerView có thể phản hồi như thế nào với đầu vào của người dùng, bạn sẽ đính kèm một trình xử lý nhấp chuột vào mỗi mục. Khi mục được nhấn, trình xử lý nhấp chuột sẽ được thực thi và văn bản của mục đó sẽ thay đổi.

Danh sách các mục mà RecyclerView hiển thị cũng có thể được sửa đổi động không nhất thiết phải là danh sách tĩnh. Có một số cách để thêm các hành vi bổ sung. Một là sử dụng nút hành động nổi (FAB). Ví dụ: trong Gmail, FAB được sử dụng để soạn một email mới. Đối với thực tế này, bạn sẽ tạo một từ mới để chèn vào danh sách. Để có một ứng dụng hữu ích hơn, bạn sẽ nhận được dữ liệu từ người dùng của mình.

#### 3.1. Làm cho các mục phản hồi các nhấp chuột

1. Mở WordListAdapter .
2. Thay đổi chữ ký lớp WordViewHolder để triển khai View.OnClickListener

```
class WordViewHolder extends RecyclerView.ViewHolder
 implements View.OnClickListener {
```

3. Nhập vào tiêu đề lớp và trên bóng đèn đỏ để thực hiện sơ khai cho các phương thức bắt buộc, trong trường hợp này chỉ là phương thức onClick().
4. Thêm mã sau vào nội dung của phương thức onClick().

```
// Get the position of the item that was clicked.
int mPosition = getLayoutPosition();
// Use that to access the affected item in mWordList.
String element = mWordList.get(mPosition);
// Change the word in the mWordList.
mWordList.set(mPosition, "Clicked! " + element);
// Notify the adapter, that the data has changed so it can
// update the RecyclerView to display the data.
mAdapter.notifyDataSetChanged();
```

5. Kết nối onClickListener với View. Thêm mã này vào hàm khởi tạo WordViewHolder (bên dưới dòng this.mAdapter = adapter):

```
itemView.setOnClickListener(this);
```

6. Chạy ứng dụng của bạn. Nhấp vào các mục để xem văn bản thay đổi.

### 3.2. Thêm hành vi vào FAB

Trong nhiệm vụ này, bạn sẽ thực hiện một hành động cho FAB để:

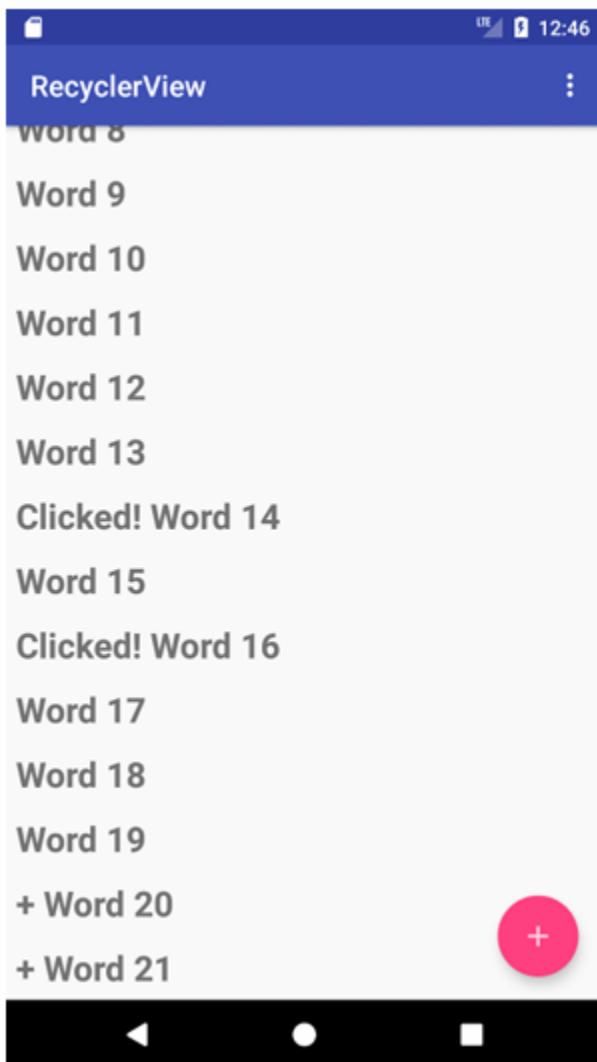
- Thêm một từ vào cuối danh sách từ.
- Thông báo cho bộ điều hợp rằng dữ liệu đã thay đổi.
- Cuộn đến mục đã chèn.

Làm theo các bước sau:

1. Mở MainActivity . Phương thức onCreate() đặt OnClickListener() thành FloatingActionButton với phương thức onClick() để thực hiện một hành động. Thay đổi phương thức onClick() thành như sau:

```
@Override
public void onClick(View view) {
 int wordListSize = mWordList.size();
 // Add a new word to the wordList.
 mWordList.addLast("+ Word " + wordListSize);
 // Notify the adapter, that the data has changed.
 mRecyclerView.getAdapter().notifyItemInserted(wordListSize);
 // Scroll to the bottom.
 mRecyclerView.smoothScrollToPosition(wordListSize);
}
```

2. Chạy ứng dụng.
3. Cuộn danh sách các từ và nhấp vào mục.
4. Thêm các mục bằng cách nhấp vào FAB.



Điều gì xảy ra nếu bạn xoay màn hình? Bạn sẽ học trong bài học sau cách giữ nguyên trạng thái của ứng dụng khi xoay màn hình.

Mã giải pháp

Dự án Android Studio: RecyclerView

Thách thức về mã hóa

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

Thử thách 1: Thay đổi menu tùy chọn để chỉ hiển thị một tùy chọn: Đặt lại. Tùy chọn này sẽ trả lại danh sách các từ về trạng thái ban đầu, không có gì được nhập vào và không có từ bổ sung.

Thử thách 2: Tạo trình nghe nhập chuột cho từng mục trong danh sách rất dễ dàng, nhưng nó có thể ảnh hưởng đến hiệu suất của ứng dụng nếu bạn có nhiều dữ liệu.

Nghiên cứu cách bạn có thể thực hiện điều này hiệu quả hơn. Đây là một thử thách nâng cao. Bắt đầu bằng cách suy nghĩ về nó theo khái niệm, và sau đó tìm kiếm một ví dụ triển khai.

Tóm tắt

- RecyclerView là một cách tiết kiệm tài nguyên để hiển thị danh sách các mặt hàng có thể cuộn được.
- Để tạo chế độ xem cho mỗi mục danh sách, bộ điều hợp thời phòng tài nguyên bô cục XML cho mục danh sách bằng cách sử dụng LayoutInflater.
- LinearLayoutManager là một trình quản lý bô cục RecyclerView hiển thị các mục trong danh sách cuộn dọc hoặc ngang.
- GridLayoutManager là một trình quản lý bô cục RecyclerView hiển thị các mục trong lưới
- StaggeredGridLayoutManager là trình quản lý bô cục RecyclerView hiển thị các mục trong lưới so le.
- Sử dụng RecyclerView.Adapter để kết nối dữ liệu của bạn với RecyclerView . Nó chuẩn bị dữ liệu trong RecyclerView.ViewHolder mô tả mục View và vị trí của nó trong RecyclerView .
- Triển khai View.OnClickListener để phát hiện các cú nhấp chuột trong RecyclerView.

Khái niệm liên quan

Tài liệu khái niệm liên quan là 4.5: RecyclerView .

Tìm hiểu thêm

Tài liệu Android Studio:

- Hướng dẫn sử dụng Android Studio

- Tạo biểu tượng ứng dụng với tài liệu dành cho nhà phát triển Image Asset Studio Android:

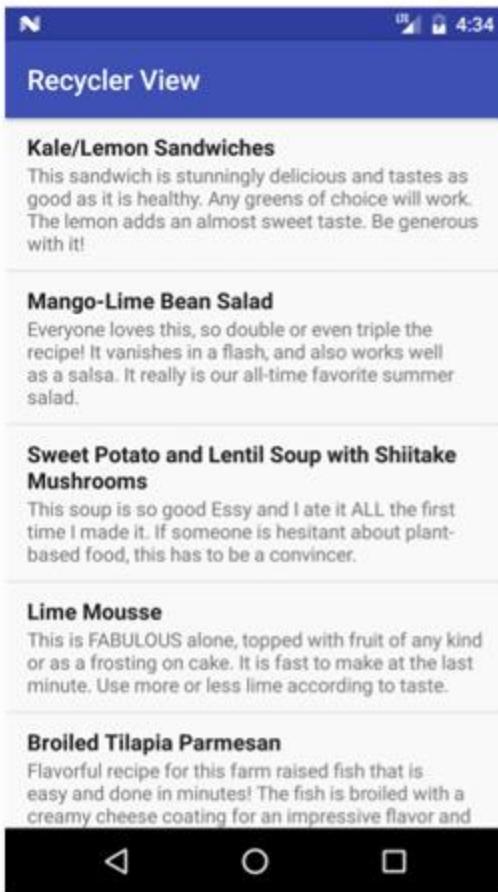
- Người tái chếView
- Bộ cục bom hơi
- RecyclerView.LayoutManager
- Trình quản lý bộ cục tuyến tính
- Trình quản lý bộ cục lưới
- Trình quản lý StaggeredGridLayoutManager
- Bộ cục điều phối viên
- Bộ cục ràng buộc
- RecyclerView.Adapter
- Người tái chếView.ViewHolder
- View.OnClickListener
- Tạo danh sách với RecyclerView

## Homework

### Xây dựng và chạy ứng dụng

Tạo một ứng dụng sử dụng RecyclerView để hiển thị danh sách các công thức nấu ăn. Mỗi mục trong danh sách phải hiển thị tên của công thức với một mô tả ngắn. Khi người dùng nhấp vào một công thức (một mục trong danh sách), hãy bắt đầu một Hoạt động hiển thị toàn bộ văn bản công thức.

- Sử dụng các phần tử TextView riêng biệt và tạo kiểu cho tên và mô tả công thức.
  - Bạn có thể sử dụng văn bản giữ chỗ cho các công thức nấu ăn đầy đủ.
  - Như một tùy chọn, hãy thêm hình ảnh cho món ăn đã hoàn thành vào mỗi công thức.
  - Nhấp vào nút Lên sẽ đưa người dùng trở lại danh sách công thức nấu ăn.
- Ảnh chụp màn hình bên dưới cho thấy một ví dụ để triển khai đơn giản. Ứng dụng của bạn có thể trông rất khác, miễn là nó có chức năng cần thiết.



Trả lời những câu hỏi này

Câu hỏi 1 Tuyên bố nào sau đây về RecyclerView là sai? Chọn một.

- RecyclerView là một cách tiết kiệm tài nguyên hơn để hiển thị danh sách có thể cuộn.
- Bạn chỉ cần cung cấp bộ cục cho một mục trong danh sách.
- Tất cả các mục danh sách trông giống nhau.
- Bạn không cần trình quản lý bộ cục với RecyclerView để xử lý hệ thống phân cấp và bộ cục của các phần tử View.

Câu hỏi 2: Thành phần nào sau đây là thành phần chính mà bạn cần cung cấp cho bộ điều hợp một mục View và vị trí của nó trong RecyclerView? Chọn một.

- Người tái chế View
- RecyclerView.Adapter
- Người tái chế View.ViewHolder
- Hoạt động AppCompatActivity

Câu hỏi 3: Bạn cần triển khai giao diện nào để nghe và phản hồi các nhấp chuột của người dùng trong RecyclerView? Chọn một.

- View.OnClickListener

- RecyclerView.Adapter
- Người tái chếView.ViewHolder
- View.OnKeyListener

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kiểm tra xem ứng dụng có các tính năng sau không:

- Triển khai RecyclerView hiển thị danh sách tiêu đề công thức và mô tả ngắn có thể cuộn được.
- Mã mở rộng hoặc triển khai RecyclerView , RecyclerView.Adapter , RecyclerView.ViewHolder và View.OnClickListener .
- Nhấp vào mục danh sách sẽ bắt đầu một Hoạt động hiển thị công thức đầy đủ.
- Tệp AndroidManifest.xml xác định mối quan hệ cha mẹ để nhấp vào nút Lên trong chế độ xem công thức sẽ quay lại danh sách công thức nấu ăn.
- ViewHolder chứa bộ cục với hai phần tử TextView; ví dụ: LinearLayout với hai phần tử TextView.

## Bài 5.1: Nội dung vẽ, kiểu và chủ đề

Giới thiệu

Trong chương này, bạn sẽ tìm hiểu cách áp dụng các kiểu phổ biến cho chế độ xem, sử dụng tài nguyên có thể vẽ và áp dụng chủ đề cho ứng dụng của mình. Những phương pháp này làm giảm mã của bạn và làm cho mã của bạn dễ đọc và bảo trì hơn. Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các yếu tố giao diện người dùng bằng trình chỉnh sửa bộ cục.
- Chỉnh sửa mã bộ cục XML và truy cập các phần tử từ mã Java của bạn.
- Trích xuất các chuỗi được mã hóa cứng vào tài nguyên chuỗi.
- Trích xuất các chiều được mã hóa cứng vào tài nguyên kích thước.
- Thêm các mục menu và biểu tượng vào menu tùy chọn trong thanh ứng dụng.
- Tạo trình xử lý nhấp chuột cho một nút bấm.
- Hiển thị tin nhắn Toast.
- Chuyển dữ liệu từ Hoạt động này sang Hoạt động khác bằng cách sử dụng Ý định.

Những gì bạn sẽ học

Bạn sẽ học cách:

- Xác định một tài nguyên phong cách.
- Áp dụng một kiểu cho Chế độ xem.
- Áp dụng chủ đề cho Hoạt động hoặc ứng dụng ở dạng XML và theo chương trình.

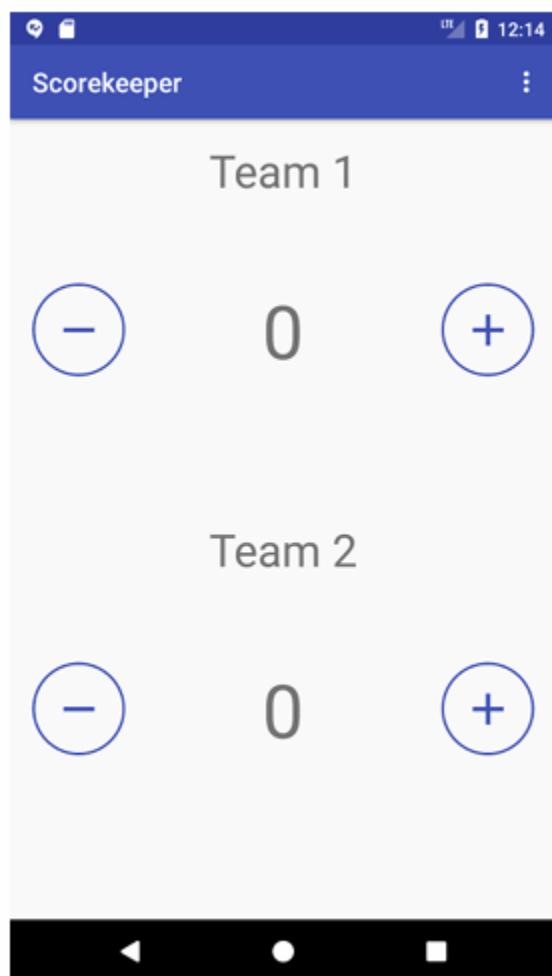
- Sử dụng tài nguyên có thể vẽ.

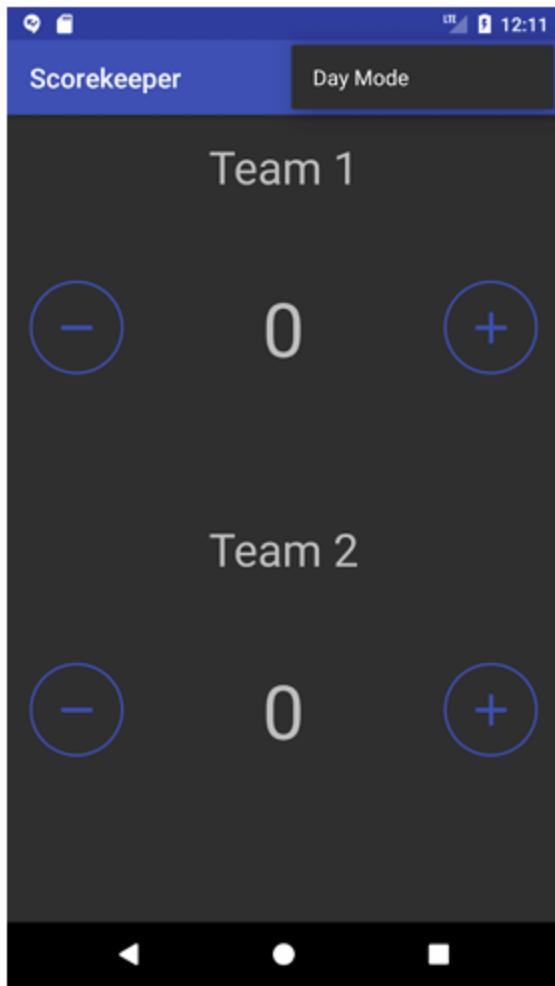
Bạn sẽ làm gì

- Tạo một ứng dụng mới và thêm các phần tử Button và TextView vào bố cục.
- Tạo tài nguyên Drawable trong XML và sử dụng chúng làm nền cho các phần tử Button của bạn.
- Áp dụng các kiểu cho các yếu tố giao diện người dùng.
- Thêm một mục menu thay đổi chủ đề của ứng dụng thành "chế độ ban đêm" có độ tương phản thấp.

Tổng quan về ứng dụng

Ứng dụng Scorekeeper bao gồm hai bộ phận từ Button và hai phần tử TextView, được sử dụng để theo dõi điểm số cho bất kỳ trò chơi dựa trên điểm nào có hai người chơi.





## Nhiệm vụ 1: Tạo ứng dụng Scorekeeper

Trong phần này, bạn sẽ tạo dự án Android Studio, sửa đổi bộ cục và thêm thuộc tính android:onClick vào các phần tử Button của nó.

### 1.1 Tạo dự án

1. Khởi động Android Studio và tạo một dự án Android Studio mới với tên Scorekeeper .
2. Chấp nhận SDK tối thiểu mặc định và chọn mẫu Hoạt động trống.
3. Chấp nhận tên Hoạt động mặc định (MainActivity) và đảm bảo các tùy chọn Tạo tệp bộ cục và Khả năng tương thích ngược (AppCompat) được chọn.
4. Nhập vào Kết thúc.

### 1.2 Tạo bộ cục cho MainActivity

Bước đầu tiên là thay đổi bộ cục từ ConstraintLayout thành LinearLayout :

1. Mở activity\_main.xml và nhấp vào Văn bản tab để xem mã XML. Ở trên cùng hoặc gốc của hệ thống phân cấp View là ConstraintLayout ViewGroup :

```
android.support.constraint.ConstraintLayout
```

2. Thay đổi ViewGroup này thành LinearLayout . Dòng mã thứ hai bây giờ trông giống như thế này:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Xóa dòng mã XML sau đây, có liên quan đến ConstraintLayout:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

Khối mã XML ở trên cùng bây giờ sẽ trông như thế này:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
```

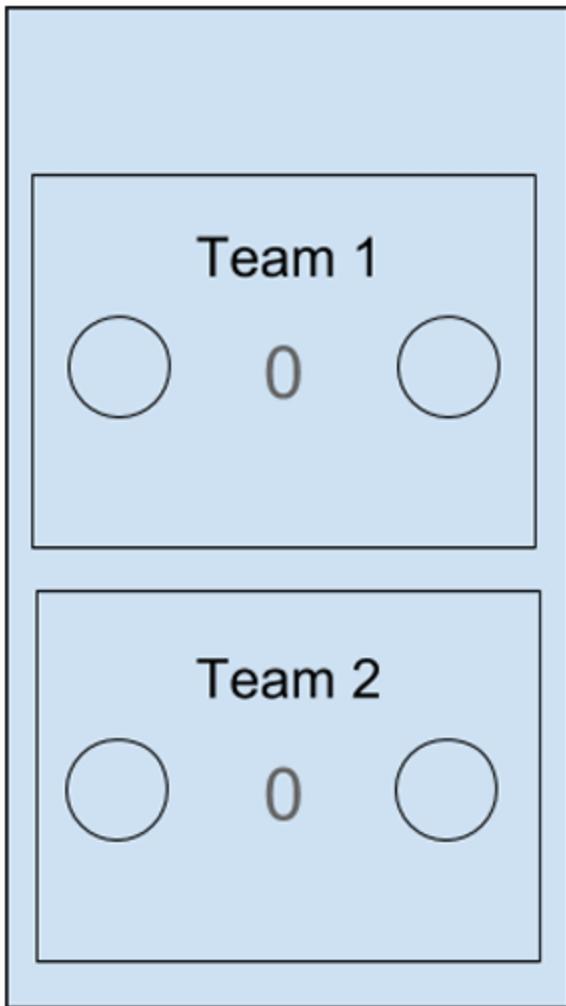
```
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context="com.example.android.scorekeeper.MainActivity">
```

4. Thêm các thuộc tính sau (mà không xóa các thuộc tính hiện có):

Attribute	Value
android:orientation	"vertical"
android:padding	"16dp"

### 1.3 Tạo vùng chứa điểm số

Bố cục cho ứng dụng này bao gồm các vùng chứa điểm được xác định bởi hai phần tử nhóm dạng xem RelativeLayout—một phần tử cho mỗi nhóm. Sơ đồ sau đây cho thấy bố cục ở dạng đơn giản nhất.



1. Bên trong LinearLayout , thêm hai phần tử RelativeLayout với các thuộc tính sau:

<b>RelativeLayout attribute</b>	<b>Value</b>
android:layout_width	"match_parent"
android:layout_height	"0dp"
android:layout_weight	"1"

Bạn có thể ngạc nhiên khi thấy rằng thuộc tính layout\_height được đặt thành 0dp. Điều này là do chúng ta đang sử dụng thuộc tính layout\_weight để xác định lượng không gian mà các phần tử RelativeLayout này chiếm trong LinearLayout mẹ.

2. Thêm hai phần tử ImageButton vào mỗi RelativeLayout : một phần tử để giảm điểm và một phần tử để tăng điểm. Sử dụng các thuộc tính sau:

<b>ImageButton attribute</b>	<b>Value</b>
android:id	"@+id/decreaseTeam1"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_alignParentLeft	"true"
android:layout_alignParentStart	"true"
android:layout_centerVertical	"true"

Sử dụng increaseTeam1 làm android:id cho ImageButton thứ hai để tăng điểm.

Sử dụng decreaseTeam2 và increaseTeam2 cho các phần tử ImageButton thứ ba và thứ tư. 3. Thêm TextView vào mỗi RelativeLayout giữa các phần tử ImageButton để hiển thị điểm số. Sử dụng các thuộc tính sau:

<b>TextView attribute</b>	<b>Value</b>
android:id	"@+id/score_1"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_centerHorizontal	"true"
android:layout_centerVertical	"true"
android:text	"0"

Sử dụng score\_2 làm android:id cho TextView thứ hai giữa các phần tử ImageButton.

4. Thêm một TextView khác vào mỗi RelativeLayout phía trên điểm số để đại diện cho Tên đội. Sử dụng các thuộc tính sau:

<b>TextView attribute</b>	<b>Value</b>
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_alignParentTop	"true"
android:layout_centerHorizontal	"true"
android:text	"Team 1"

#### 1.4 Thêm nội dung vector

Bạn sẽ sử dụng các biểu tượng vật liệu trong Vector Asset Studio cho các phần tử ImageButton ghi điểm.

1. Chọn File > New > Vector Asset để mở Vector Asset Studio.
  2. Nhấp vào biểu tượng để mở danh sách các tệp biểu tượng vật liệu. Chọn danh mục Nội dung.
  3. Chọn biểu tượng thêm và nhấp vào OK . 4. Đổi tên tệp tài nguyên ic\_plus và chọn Ghi đè hộp kiểm bên cạnh tùy chọn kích thước.
  5. Thay đổi kích thước của biểu tượng thành 40dp x 40dp.
  6. Nhấp vào Tiếp theo và sau đó Kết thúc .
  7. Lặp lại quy trình này để thêm biểu tượng xóa và đặt tên tệp ic\_minus.
- Bây giờ bạn có thể thêm các biểu tượng và mô tả nội dung cho các phần tử ImageButton. Mô tả nội dung cung cấp một nhãn hữu ích và mô tả giải thích ý nghĩa và mục đích của ImageButton cho những người dùng có thể yêu cầu các tính năng hỗ trợ tiếp cận của Android như trình đọc màn hình Google TalkBack.
8. Thêm các thuộc tính sau vào các phần tử ImageButton ở phía bên trái của bố cục

```
android:src="@drawable/ic_minus"
android:contentDescription="Minus Button"
```

9. Thêm các thuộc tính sau vào các phần tử ImageButton ở phía bên phải của bố cục:

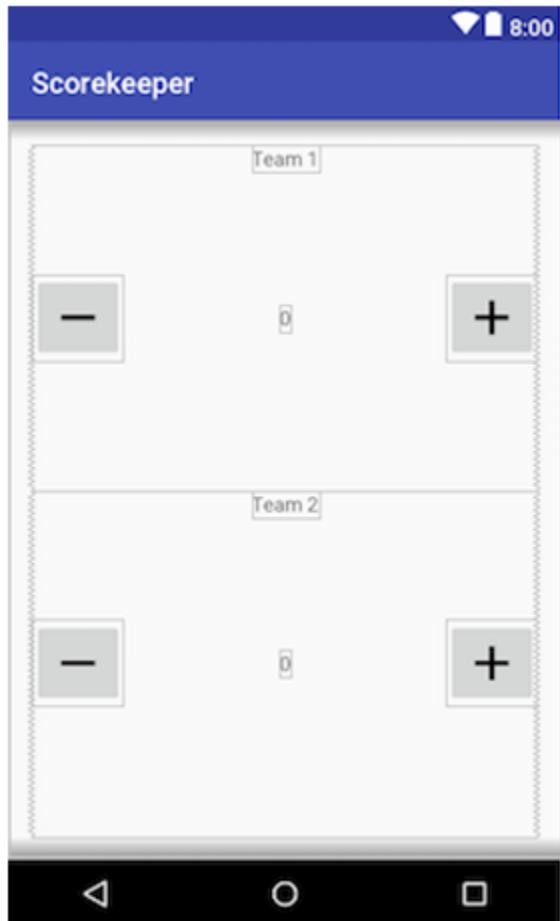
```
android:src="@drawable/ic_plus"
android:contentDescription="Plus Button"
```

10. Trích xuất tất cả các tài nguyên chuỗi của bạn. Quá trình này xóa tất cả các chuỗi của bạn khỏi mã Java và đưa chúng vào tệp strings.xml. Điều này cho phép ứng dụng của bạn dễ dàng được bản địa hóa sang các ngôn ngữ khác nhau.

Mã giải pháp cho bộ cục

Sau đây cho thấy bộ cục cho ứng dụng Scorekeeper và mã XML cho bộ cục.

Lưu ý: Mã của bạn có thể hơi khác một chút, vì có nhiều cách để đạt được cùng một bộ cục.



Mã XML:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:padding="16dp"
 tools:context="com.example.android.scorekeeper.MainActivity">

 <RelativeLayout
 android:layout_width="match_parent"
 android:layout_height="0dp"
 android:layout_weight="1">

 <TextView
```

```
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentTop="true"
 android:layout_centerHorizontal="true"
 android:text="@string/team_1" />

 <ImageButton
 android:id="@+id/decreaseTeam1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentLeft="true"
 android:layout_alignParentStart="true"
 android:layout_centerVertical="true"
 android:src="@drawable/ic_minus"
 android:contentDescription=
 "@string/minus_button_description" />

 <TextView
 android:id="@+id/score_1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
 android:layout_centerVertical="true"
 android:text="@string/initial_count" />

 <ImageButton
 android:id="@+id/increaseTeam1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentEnd="true"
 android:layout_alignParentRight="true"
 android:layout_centerVertical="true"
 android:src="@drawable/ic_plus"
 android:contentDescription=
 "@string/plus_button_description" />
</RelativeLayout>

<RelativeLayout
 android:layout_width="match_parent"
 android:layout_height="0dp"
 android:layout_weight="1">

 <TextView
```

```

 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentTop="true"
 android:layout_centerHorizontal="true"
 android:text="@string/team_2" />

 <ImageButton
 android:id="@+id/decreaseTeam2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentLeft="true"
 android:layout_alignParentStart="true"
 android:layout_centerVertical="true"
 android:src="@drawable/ic_minus"
 android:contentDescription=
 "@string/minus_button_description" />

 <TextView
 android:id="@+id/score_2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
 android:layout_centerVertical="true"
 android:text="@string/initial_count" />

 <ImageButton
 android:id="@+id/increaseTeam2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentEnd="true"
 android:layout_alignParentRight="true"
 android:layout_centerVertical="true"
 android:src="@drawable/ic_plus"
 android:contentDescription=
 "@string/plus_button_description" />
 </RelativeLayout>
</LinearLayout>

```

## 1.5 Khởi tạo các phần tử TextView và các biến đếm điểm

Để theo dõi điểm số, bạn sẽ cần hai điều:

- Biến số nguyên để theo dõi điểm số.
- Tham chiếu đến từng phần tử TextView điểm số trong MainActivity để bạn có thể cập nhật điểm số.

Làm theo các bước sau:

1. Tạo hai biến thành viên số nguyên đại diện cho điểm số của mỗi đội.

```
// Member variables for holding the score
private int mScore1;
private int mScore2;
```

2. Tạo hai biến thành viên TextView để giữ các tham chiếu đến các phần tử TextView.

```
// Member variables for holding the score
private int mScore1;
private int mScore2;
```

3. Trong phương thức onCreate() của MainActivity , tìm các phần tử TextView điểm số của bạn theo id và gán chúng cho các biến thành viên.

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

 //Find the TextViews by ID
 mScoreText1 = (TextView)findViewById(R.id.score_1);

 mScoreText2 = (TextView)findViewById(R.id.score_2);
}
```

## 1.6 Triển khai các trình xử lý nhấp chuột cho các phần tử ImageButton

Bạn sẽ thêm thuộc tính android:onClick vào mỗi ImageButton và tạo hai phương thức xử lý nhấp chuột trong MainActivity . Các phần tử ImageButton bên trái sẽ giảm điểm TextView , trong khi các phần tử bên phải sẽ tăng nó.

1. Mở activity\_main.xml nếu nó chưa được mở và thêm thuộc tính android:onClick sau vào ImageButton đầu tiên ở phía bên trái của bố cục:

```
android:onClick="decreaseScore"
```

2. Tên phương thức decreaseScore được gạch chân màu đỏ. Nhấp vào biểu tượng bóng đèn màu đỏ ở lề trái hoặc nhấp vào tên phương thức và nhấn Option-Return và chọn Tạo 'decreaseScore(view)' trong 'MainActivity' . Android Studio tạo sơ khai phương thức decreaseScore() trong MainActivity .

3. Thêm thuộc tính android:onClick ở trên vào ImageButton thứ hai ở phía bên trái của bố cục. Lần này tên phương thức hợp lệ, vì sơ khai đã được tạo.

4. Thêm thuộc tính android:onClick sau vào mỗi ImageButton ở phía bên phải của máy điện

```
 android:onClick="increaseScore"
```

5. Tên phương thức increaseScore được gạch chân bằng màu đỏ. Bấm vào biểu tượng bóng đèn màu đỏ ở lề trái, hoặc bấm vào tên phương thức và nhấn Option-Return và chọn Tạo 'increaseScore(view)' trong 'MainActivity'. Android Studio tạo sơ khai phương thức increaseScore() trong MainActivity.

6. Thêm mã vào các phương thức sơ khai để giảm và tăng điểm như hình dưới đây. Cả hai phương thức đều sử dụng view.getId() để lấy ID của ImageButton của nhóm đã được nhấp để mã của bạn có thể cập nhật nhóm thích hợp.

Mã giải pháp cho increaseScore() và decreaseScore()

```
/**
 * Method that handles the onClick of both the decrement buttons
 * @param view The button view that was clicked
 */
public void decreaseScore(View view) {
 // Get the ID of the button that was clicked
 int viewID = view.getId();
 switch (viewID){
 //If it was on Team 1
 case R.id.decreaseTeam1:
 //Decrement the score and update the TextView
 mScore1--;
 mScoreText1.setText(String.valueOf(mScore1));
 break;
 //If it was Team 2
 case R.id.decreaseTeam2:
 //Decrement the score and update the TextView
 mScore2--;
 mScoreText2.setText(String.valueOf(mScore2));
 }
}

/**
 * Method that handles the onClick of both the increment buttons
 * @param view The button view that was clicked
 */
public void increaseScore(View view) {
 //Get the ID of the button that was clicked
 int viewID = view.getId();
```

```

switch (viewID){
 //If it was on Team 1
 case R.id.increaseTeam1:
 //Increment the score and update the TextView
 mScore1++;
 mScoreText1.setText(String.valueOf(mScore1));
 break;
 //If it was Team 2
 case R.id.increaseTeam2:
 //Increment the score and update the TextView
 mScore2++;
 mScoreText2.setText(String.valueOf(mScore2));
}
}

```

## Nhiệm vụ 2: Tạo tài nguyên có thể vẽ

Bây giờ bạn đã có một ứng dụng Scorekeeper đang hoạt động! Tuy nhiên, bộ cục buồn tẻ và không truyền đạt chức năng của các phần tử ImageButton. Để làm rõ hơn, nền màu xám tiêu chuẩn của các nút có thể được thay đổi.

Trong Android, đồ họa thường được xử lý bởi một tài nguyên có tên là Drawable .

Trong bài tập sau, bạn sẽ tìm hiểu cách tạo một loại Drawable nhất định được gọi là ShapeDrawable và áp dụng nó cho các phần tử ImageButton của bạn làm nền.

Để biết thêm thông tin về Drawables , hãy xem Tài nguyên có thể vẽ .

### 2.1 Tạo một ShapeDrawable

ShapeDrawable là một hình dạng hình học nguyên thủy được xác định trong tệp XML bởi một số thuộc tính bao gồm màu sắc, hình dạng, khoảng đệm và hơn thế nữa. Nó xác định một đồ họa vector, có thể tăng và giảm tỷ lệ mà không làm mất bất kỳ định nghĩa nào.

1. Trong ngăn Project > Android, nhấp chuột phải (hoặc Control khi nhấp ) vào thư mục có thể vẽ trong thư mục res.
2. Chọn tệp tài nguyên mới > Drawable.
3. Đặt tên cho tệp button\_background và nhấp vào OK . (Không thay đổi bộ nguồn hoặc Tên thư mục và không thêm bộ hạn định). Android Studio tạo tệp button\_background.xml trong thư mục có thể vẽ.
4. Mở button\_background.xml , nhấp vào Văn bản tab để chỉnh sửa mã XML và xóa tất cả mã ngoại trừ:

```
<?xml version="1.0" encoding="utf-8"?>
```

5. Thêm mã sau tạo ra một hình bầu dục với một đường viền:

```
<shape
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:shape="oval">
 <stroke
 android:width="2dp"
 android:color="@color/colorPrimary"/>
</shape>
```

## 2.2 Áp dụng ShapeDrawable làm nền

1. Mở activity\_main.xml .
2. Thêm Drawable làm nền cho cả bốn phần tử ImageButton:

```
 android:background="@drawable/button_background"
```

3. Nhấp vào Xem trước tab trong trình chỉnh sửa bố cục để xem nền tự động chia tỷ lệ theo kích thước của ImageButton .
4. Để hiển thị đúng từng ImageButton trên tất cả các thiết bị, bạn sẽ thay đổi thuộc tính android:layout\_height và android:layout\_width cho mỗi ImageButton thành 70dp , đây là kích thước tốt trên hầu hết các thiết bị. Thay đổi thuộc tính đầu tiên cho ImageButton đầu tiên như sau:

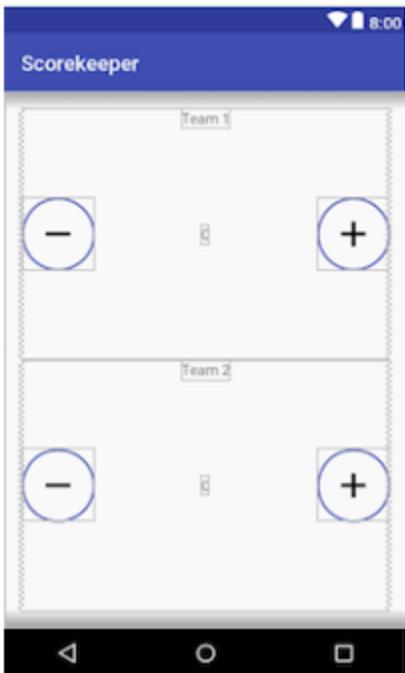
```
 android:layout_width="70dp"
```

- 5: Trích xuất tài nguyên kích thước từ giá trị "70dp".Nhấp chuột một lần vào "**70dp**" trong tệp XML giao diệnTrên **Windows**, nhấn Alt + Enter; trên **macOS**, nhấn Option + Enter. Trong menu bật lên, chọn "**Extract dimension resource**". Đặt tên cho tài nguyên (ví dụ: list\_item\_height) và nhấn **OK**.

6. Nhập button\_size cho Tên tài nguyên .
7. Nhấp vào OK . Thao tác này sẽ tạo ra một tài nguyên thứ nguyên trong tệp dimens.xml (trong thư mục giá trị) và thứ nguyên trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: @dimen/button\_size
8. Thay đổi thuộc tính android:layout\_height và android:layout\_width cho từng phần tử ImageButton bằng cách sử dụng tài nguyên thứ nguyên mới:

```
 android:layout_width="@dimen/button_size"
 android:layout_height="@dimen/button_size"
```

9. Nhấp vào tab Xem trước trong trình chỉnh sửa bố cục để xem bố cục. ShapeDrawable hiện được sử dụng cho các phần tử ImageButton.



## Nhiệm vụ 3: Tạo kiểu cho các thành phần View của bạn

Khi bạn tiếp tục thêm các phần tử và thuộc tính View vào bố cục, mã của bạn sẽ bắt đầu trở nên lớn và lặp đi lặp lại, đặc biệt là khi bạn áp dụng các thuộc tính giống nhau cho nhiều phần tử tương tự. Một kiểu có thể chỉ định các thuộc tính phổ biến như đậm, màu phông chữ, kích thước phông chữ và màu nền. Các thuộc tính hướng bố cục như chiều cao, chiều rộng và vị trí tương đối sẽ vẫn còn trong tệp tài nguyên bố cục. Trong bài tập sau, bạn sẽ tìm hiểu cách tạo kiểu và áp dụng chúng cho nhiều phần tử và bố cục View, cho phép các thuộc tính chung được cập nhật đồng thời từ một vị trí. Lưu ý: Kiểu dành cho các thuộc tính sửa đổi giao diện của Chế độ xem . Các thông số bố cục như chiều cao, trọng lượng và vị trí tương đối phải nằm trong tệp bố cục.

### 3.1 Tạo kiểu nút

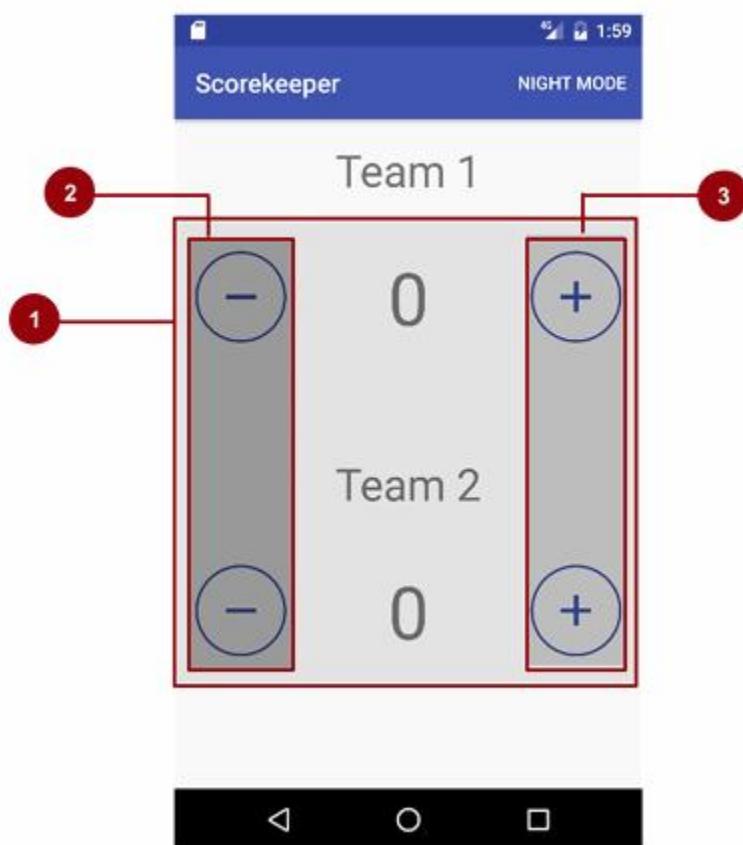
Trong Android, kiểu có thể kế thừa thuộc tính từ các kiểu khác. Bạn có thể khai báo cha cho kiểu của mình bằng cách sử dụng tham số cha tùy chọn và có các thuộc tính sau:

- Bất kỳ thuộc tính kiểu nào được xác định bởi kiểu mẹ sẽ tự động được đưa vào kiểu con.
- Thuộc tính kiểu được xác định trong cả kiểu mẹ và kiểu con sử dụng định nghĩa của kiểu con (thuộc tính con ghi đè kiểu mẹ).
- Kiểu con có thể bao gồm các thuộc tính bổ sung mà cha mẹ không xác định.

Ví dụ: cả bốn phần tử ImageButton trong ứng dụng Scorekeeper đều có chung nền Drawable nhưng có các biểu tượng khác nhau cho dấu cộng (tăng điểm) và trừ (giảm

điểm). Hơn nữa, hai phần tử ImageButton cộng chia sẻ cùng một biểu tượng, cũng như hai phần tử ImageButton trù. Do đó, bạn có thể tạo ba kiểu:

1. Một kiểu cho tất cả các phần tử ImageButton, bao gồm các thuộc tính mặc định của ImageButton và cả nền Drawable.
2. Một kiểu cho các phần tử ImageButton trù. Kiểu này kế thừa các thuộc tính của kiểu ImageButton và bao gồm biểu tượng trù.
3. Một kiểu cho các phần tử ImageButton cộng. Kiểu này kế thừa từ kiểu ImageButton và bao gồm biểu tượng dấu cộng. Các phong cách này được thể hiện trong hình dưới đây.



Thực hiện như sau:

1. Mở rộng res > values trong ngăn Project > Android và mở tệp styles.xml.

Đây là nơi chứa toàn bộ mã style của bạn. Style "AppTheme" luôn được tự động thêm vào và bạn có thể thấy rằng nó mở rộng từ Theme.AppCompat.Light.DarkActionBar

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

Lưu ý thuộc tính parent, đó là cách bạn chỉ định kiểu cha của mình bằng XML. Thuộc tính name, trong trường hợp này là AppTheme , xác định tên của kiểu. Thuộc tính mẹ, trong trường hợp này là Theme.AppCompat.Light.DarkActionBar, khai báo các thuộc tính kiểu cha mà AppTheme kế thừa. Trong trường hợp này, nó là chủ đề mặc định của Android, với nền sáng và thanh hành động tối.

Chủ đề là một kiểu được áp dụng cho toàn bộ Hoạt động hoặc ứng dụng thay vì một Chế độ xem duy nhất. Việc sử dụng chủ đề sẽ tạo kiểu nhất quán trong toàn bộ Hoạt động hoặc ứng dụng – ví dụ: giao diện nhất quán cho thanh ứng dụng trong mọi phần của ứng dụng.

2. Giữa các <resources> thẻ, thêm một kiểu mới với các thuộc tính sau để tạo kiểu chung cho tất cả các nút:

```
<style name="ScoreButtons" parent="Widget.AppCompat.Button">
 <item name="android:background">@drawable/button_background</item>
</style>
```

Đoạn mã trên đặt kiểu cha thành **Widget.AppCompat.Button** để giữ nguyên các thuộc tính mặc định của **Button**. Nó cũng thêm một thuộc tính để thay đổi nền của **Drawable** thành hình ảnh mà bạn đã tạo trong nhiệm vụ trước.

1. Tạo kiểu cho các nút dấu cộng bằng cách mở rộng kiểu **ScoreButtons**:

```
<style name="PlusButtons" parent="ScoreButtons">
 <item name="android:src">@drawable/ic_plus</item>
 <item name=
 "android:contentDescription">@string/plus_button_description</item>
</style>
```

Thuộc tính contentDescription dành cho người dùng khiếm thị. Nó hoạt động như một nhãn mà một số thiết bị hỗ trợ tiếp cận sử dụng để đọc to nhằm cung cấp một số ngữ cảnh về ý nghĩa của phần tử giao diện người dùng.

1. Tạo kiểu cho các nút trừ:

```

<style name="MinusButtons" parent="ScoreButtons">
 <item name="android:src">@drawable/ic_minus</item>
 <item name=
 "android:contentDescription">@string/minus_button_description</item>
</style>

```

1. Bây giờ bạn có thể sử dụng các kiểu này để thay thế các thuộc tính kiểu cụ thể của các phần tử ImageButton. Mở tệp bố cục activity\_main.xml cho MainActivity và thay thế các thuộc tính sau cho cả hai phần tử ImageButton trừ:

Delete these attributes	Add this attribute
android:src	<p>Lưu ý: Thuộc tính style không sử dụng không gian tên android: vì style là một phần của thuộc tính XML mặc định.</p>

2. Thay thế các thuộc tính sau cho cả hai phần tử ImageButton cộng:

Delete these attributes	Add this attribute
android:src	<h3>3.2 Tạo kiểu TextView</h3>

Các thành phần TextView hiển thị tên đội và điểm số cũng có thể được tạo kiểu vì chúng có màu sắc và phông chữ chung. Thực hiện như sau:

1. Thêm thuộc tính sau vào tất cả các phần tử TextView:

```
android:textAppearance="@style/TextAppearance.AppCompat.Headline"
```

2. Nhấp chuột phải (hoặc Control-nhấp chuột) ở bất kỳ đâu trong các thuộc tính TextView điểm đầu tiên và chọn Refactor > Extract > Style...

3. Đặt tên cho kiểu ScoreText và chọn hộp textAppearance (thuộc tính bạn vừa thêm) cũng như khởi chạy hộp kiểm tái cấu trúc 'Sử dụng kiểu nếu có thể' sau khi phong cách được trích xuất. Thao tác này sẽ quét tệp bố cục để tìm các chế độ xem có cùng thuộc tính và áp dụng kiểu cho bạn. Không trích xuất các thuộc tính có liên quan đến

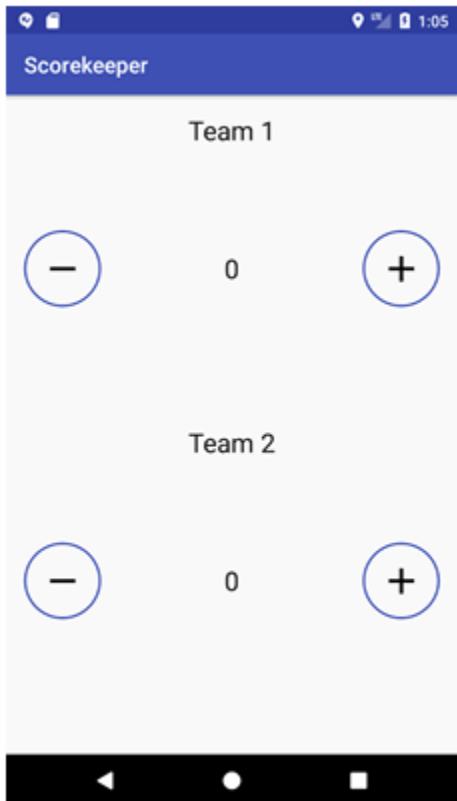
bố cục—bấm vào các hộp kiểm khác để tắt chúng.

4. Chọn OK.

5. Đảm bảo phạm vi được đặt thành tệp bố cục activity\_main.xml và nhấp vào OK.

6. Một ngăn ở cuối Android Studio sẽ mở ra nếu tìm thấy cùng một kiểu trong các chế độ xem khác. Chọn Thực hiện tái cấu trúc để áp dụng kiểu mới cho các dạng xem có cùng thuộc tính.

7. Chạy ứng dụng của bạn. Không có thay đổi nào ngoại trừ tất cả mã tạo kiểu của bạn bây giờ nằm trong tệp tài nguyên của bạn và tệp bố cục của bạn ngắn hơn.



Mã giải pháp bố cục và kiểu

Sau đây là đoạn mã cho bố cục và kiểu. styles.xml:

```
<resources>
 <!-- Base application theme. -->
 <style name="AppTheme"
 parent="Theme.AppCompat.Light.DarkActionBar">
 <!-- Customize your theme here. -->
 <item name="colorPrimary">@color/colorPrimary</item>
 <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
 <item name="colorAccent">@color/colorAccent</item>
 </style>

 <style name="ScoreButtons" parent="AppTheme">
 <item
 name="android:background">@drawable/button_background</item>
 </style>

 <style name="PlusButtons" parent="ScoreButtons">
 <item name="android:src">@drawable/ic_plus</item>
 <item name=
 "android:contentDescription">@string/plus_button_description</item>
 </style>

 <style name="MinusButtons" parent="ScoreButtons">
 <item name="android:src">@drawable/ic_minus</item>
 <item name=
 "android:contentDescription">@string/minus_button_description</item>
 </style>

 <style name="ScoreText">
 <item name=
 "android:textAppearance">@style/TextAppearance.AppCompat.Headline</item>
 </style>
</resources>
```

activity\_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:padding="16dp"
 tools:context="com.example.android.scorekeeper.MainActivity">

 <RelativeLayout
 android:layout_width="match_parent"
 android:layout_height="0dp"
 android:layout_weight="1">

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentTop="true"
 android:layout_centerHorizontal="true"
 android:text="@string/team_1"
```

```
 style="@style/ScoreText" />

 <ImageButton
 android:id="@+id/decreaseTeam1"
 android:layout_width="@dimen/button_size"
 android:layout_height="@dimen/button_size"
 android:layout_alignParentLeft="true"
 android:layout_alignParentStart="true"
 android:layout_centerVertical="true"
 style="@style/MinusButtons"
 android:onClick="decreaseScore"/>

 <TextView
 android:id="@+id/score_1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
 android:layout_centerVertical="true"
 android:text="@string/initial_count"
 style="@style/ScoreText" />

 <ImageButton
 android:id="@+id/increaseTeam1"
 android:layout_width="@dimen/button_size"
 android:layout_height="@dimen/button_size"
 android:layout_alignParentEnd="true"
 android:layout_alignParentRight="true"
 android:layout_centerVertical="true"
 style="@style/PlusButtons"
 android:onClick="increaseScore"/>

</RelativeLayout>

<RelativeLayout
 android:layout_width="match_parent"
 android:layout_height="0dp"
 android:layout_weight="1">

 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentTop="true"
 android:layout_centerHorizontal="true"
```

```

 android:text="@string/team_2"
 style="@style/ScoreText" />

 <ImageButton
 android:id="@+id/decreaseTeam2"
 android:layout_width="@dimen/button_size"
 android:layout_height="@dimen/button_size"
 android:layout_alignParentLeft="true"
 android:layout_alignParentStart="true"
 android:layout_centerVertical="true"
 style="@style/MinusButtons"
 android:onClick="decreaseScore"/>

 <TextView
 android:id="@+id/score_2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
 android:layout_centerVertical="true"
 android:text="@string/initial_count"
 style="@style/ScoreText" />

 <ImageButton
 android:id="@+id/increaseTeam2"
 android:layout_width="@dimen/button_size"
 android:layout_height="@dimen/button_size"
 android:layout_alignParentEnd="true"
 android:layout_alignParentRight="true"
 android:layout_centerVertical="true"
 style="@style/PlusButtons"
 android:onClick="increaseScore"/>
</RelativeLayout>
</LinearLayout>

```

### 3.3 Cập nhật các kiểu

Sức mạnh của việc sử dụng phong cách trở nên rõ ràng khi bạn muốn thực hiện các thay đổi đối với một số yếu tố của cùng một phong cách. Bạn có thể làm cho văn bản lớn hơn, đậm hơn và sáng hơn, đồng thời thay đổi các biểu tượng thành màu của nền nút.

1. Mở tệp styles.xml và thêm hoặc sửa đổi các thuộc tính sau cho các kiểu:

Style	Item
ScoreButtons	<item name="android:tint">@color/colorPrimary</item>
ScoreText	<item name="android:textAppearance">@style/TextAppearance.AppCompat.Display3 </item>

Giá trị colorPrimary được Android Studio tự động tạo khi bạn tạo dự án. Bạn có thể tìm thấy nó trong ngăn Project > Android trong tệp colors.xml bên trong thư mục values. Thuộc tính TextAppearance.AppCompat.Display3 là một kiểu văn bản được xác định trước do Android cung cấp.

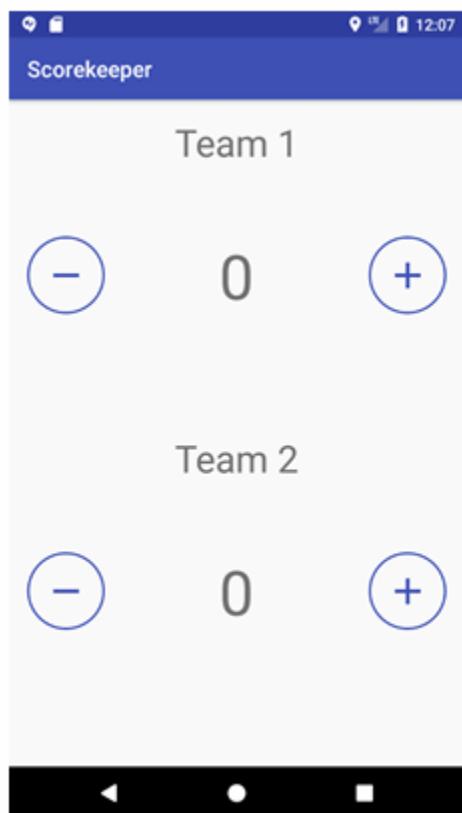
1. Tạo một kiểu mới có tên là TeamText với thuộc tính textAppearance được đặt như sau:

```
<style name="TeamText">
 <item name=
 "android:textAppearance">@style/TextAppearance.AppCompat.Display1
 </item>
</style>
```

2. Trong activity\_main.xml , thay đổi thuộc tính style của các phần tử tên nhóm TextView thành kiểu TeamText mới tạo.

```
style="@style/TeamText"
```

3. Chạy ứng dụng của bạn. Chỉ với những điều chỉnh này đối với tệp style.xml, tất cả các chế độ xem được cập nhật để phản ánh các thay đổi.



## Nhiệm vụ 4: Chủ đề và bước cuối cùng

Bạn đã thấy rằng các phần tử View có các đặc điểm tương tự có thể được tạo kiểu với nhau trong styles.xml file. Nhưng điều gì sẽ xảy ra nếu bạn muốn xác định kiểu cho toàn bộ Hoạt động hoặc toàn bộ ứng dụng? Có thể thực hiện điều này bằng cách sử dụng chủ đề. Để đặt chủ đề cho từng Hoạt động , bạn cần sửa đổi tệp AndroidManifest.xml.

Trong nhiệm vụ này, bạn sẽ thêm chủ đề "chế độ ban đêm" vào ứng dụng của mình, cho phép người dùng sử dụng phiên bản tương phản thấp của ứng dụng dễ nhìn hơn vào ban đêm, cũng như thực hiện một vài thao tác đánh bóng vào giao diện người dùng.

### 4.1 Khám phá chủ đề

1. Mở tệp AndroidManifest.xml, tìm <application> thẻ và thay đổi thuộc tính android:theme thành:

```
android:theme="@style/Theme.AppCompat.Light.NoActionBar"
```

Đây là chủ đề được xác định trước để xóa thanh hành động khỏi hoạt động của bạn.

2. Chạy ứng dụng của bạn. Thanh công cụ biến mất!
  3. Thay đổi chủ đề của ứng dụng trở lại AppTheme , là chủ đề con của chủ đề Theme.Appcompat.Light.DarkActionBar như có thể thấy trong styles.xml .
- Để áp dụng chủ đề cho một hoạt động thay vì toàn bộ ứng dụng, hãy đặt thuộc tính theme vào <Activity> thẻ thay vì <application> thẻ. Để biết thêm thông tin về chủ đề và kiểu, hãy xem Hướng dẫn về kiểu và chủ đề .

### 4.2 Thêm nút chủ đề vào menu

Một cách sử dụng để đặt chủ đề cho ứng dụng của bạn là cung cấp trải nghiệm trực quan thay thế để duyệt web vào ban đêm. Trong điều kiện như vậy, tốt hơn hết bạn nên có bố cục tối, độ tương phản thấp. Khung Android cung cấp một chủ đề được thiết kế chính xác cho việc này: Chủ đề DayNight

Chủ đề này có các tùy chọn tích hợp cho phép bạn kiểm soát màu sắc trong ứng dụng của mình theo chương trình: bằng cách đặt nó tự động thay đổi theo thời gian hoặc theo lệnh của người dùng.

Trong bài tập này, bạn sẽ thêm một mục menu tùy chọn sẽ chuyển đổi ứng dụng giữa chủ đề thông thường và chủ đề "chế độ ban đêm".

1. Mở strings.xml và tạo hai tài nguyên chuỗi cho mục menu tùy chọn này:

```
<string name="night_mode">Night Mode</string>
<string name="day_mode">Day Mode</string>
```

2. Nhấp chuột phải (hoặc Control khi nhấp) vào thư mục res trong ngăn Project > Android và chọn Mới > tệp tài nguyên Android .
3. Đặt tên cho tệp main\_menu , thay đổi Loại tài nguyên thành Menu và nhấp vào OK . Trình chỉnh sửa bộ cục xuất hiện cho tệp main\_menu.xml.
4. Nhấp vào tab Văn bản của trình chỉnh sửa bộ cục để hiển thị mã XML.
5. Thêm một mục menu với các thuộc tính sau:

```
<item
 android:id="@+id/night_mode"
 android:title="@string/night_mode"/>
```

6. Mở MainActivity , nhấn Ctrl-O để mở menu Phương pháp ghi đè và chọn phương thức onCreateOptionsMenu (menu: Menu): boolean nằm trong danh mục android.app.Activity .
7. Nhấp vào OK. Android Studio tạo sơ khai phương thức onCreateOptionsMenu() với return super.onCreateOptionsMenu(menu) làm câu lệnh duy nhất.
1. Trong onCreateOptionsMenu() ngay trước câu lệnh return.super, thêm mã để khởi động menu:

```
getMenuInflater().inflate(R.menu.main_menu, menu);
```

#### 4.3 Thay đổi chủ đề từ menu

Chủ đề **DayNight** sử dụng lớp **AppCompatDelegate** để đặt tùy chọn chế độ ban đêm trong **Activity**. Để tìm hiểu thêm về chủ đề này, hãy truy cập bài viết trên blog.

1. Trong tệp **styles.xml**, sửa đổi **parent** của **AppTheme** thành "Theme.AppCompat.DayNight.DarkActionBar".
2. Ghi đè phương thức **onOptionsItemSelected()** trong **MainActivity**, và thêm mã để kiểm tra mục menu nào đã được nhấp.

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
 //Check if the correct item was clicked
 if(item.getItemId()==R.id.night_mode) {}
 // TODO: Get the night mode state of the app.
 return true;
}

```

3. Thay thế nhận xét TODO: trong đoạn mã trên bằng mã kiểm tra xem chế độ ban đêm đã được bật chưa. Nếu được bật, mã sẽ thay đổi chế độ ban đêm sang trạng thái vô hiệu hóa; Nếu không, mã sẽ bật chế độ ban đêm:

```

if(item.getItemId()==R.id.night_mode) {
 // Get the night mode state of the app.
 int nightMode = AppCompatDelegate.getDefaultNightMode();
 //Set the theme mode for the restarted activity
 if (nightMode == AppCompatDelegate.MODE_NIGHT_YES) {
 AppCompatDelegate.setDefaultNightMode
 (AppCompatDelegate.MODE_NIGHT_NO);
 } else {
 AppCompatDelegate.setDefaultNightMode
 (AppCompatDelegate.MODE_NIGHT_YES);
 }
 // Recreate the activity for the theme change to take effect.
 recreate();
}

```

Để phản hồi khi nhấp vào mục menu, mã sẽ xác minh cài đặt chế độ ban đêm hiện tại bằng cách gọi `AppCompatDelegate.getDefaultNightMode()`.

Chủ đề chỉ có thể thay đổi trong khi hoạt động đang được tạo, vì vậy mã sẽ gọi `recreate()` để thay đổi chủ đề có hiệu lực.

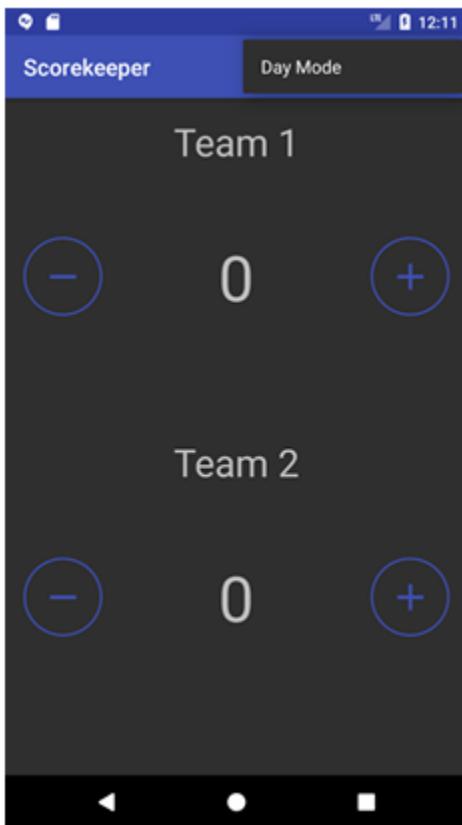
4. Chạy ứng dụng của bạn. Mục menu Chế độ ban đêm bây giờ sẽ chuyển đổi chủ đề của Hoạt động của bạn.

5. Bạn có thể nhận thấy rằng nhãn cho mục menu của bạn luôn hiển thị Chế độ ban đêm, điều này có thể gây nhầm lẫn cho người dùng của bạn nếu ứng dụng đã ở chế độ tối.

1. Thay thế câu lệnh `return super.onCreateOptionsMenu(menu)` trong phương thức `onCreateOptionsMenu()` bằng mã sau:

```
// Change the label of the menu based on the state of the app.
int nightMode = AppCompatDelegate.getDefaultNightMode();
if(nightMode == AppCompatDelegate.MODE_NIGHT_YES){
 menu.findItem(R.id.night_mode).setTitle(R.string.day_mode);
} else{
 menu.findItem(R.id.night_mode).setTitle(R.string.night_mode);
}
return true;
```

- Chạy ứng dụng của bạn. Nhấn mục menu Chế độ ban đêm , sau khi người dùng nhấn vào nó, bây giờ sẽ thay đổi thành Chế độ ban ngày (cùng với chủ đề).



#### 4.4 Lưu InstanceState

Bạn đã học được trong các bài học trước rằng bạn phải chuẩn bị cho Hoạt động của mình bị phá hủy và tạo lại vào những thời điểm bất ngờ, ví dụ như khi màn hình của bạn được xoay. Trong ứng dụng này, các phần tử TextView chứa điểm số được đặt lại về giá trị ban đầu là 0 khi thiết bị được xoay. Để khắc phục vấn đề này, hãy làm như sau:

- Mở MainActivity và thêm thẻ dưới các biến thành viên, sẽ được sử dụng làm khóa trong onSaveInstanceState() :

```
static final String STATE_SCORE_1 = "Team 1 Score";
static final String STATE_SCORE_2 = "Team 2 Score";
```

2. Ở cuối MainActivity , ghi đè phương thức onSaveInstanceState() để giữ nguyên giá trị của hai phần tử TextView điểm:

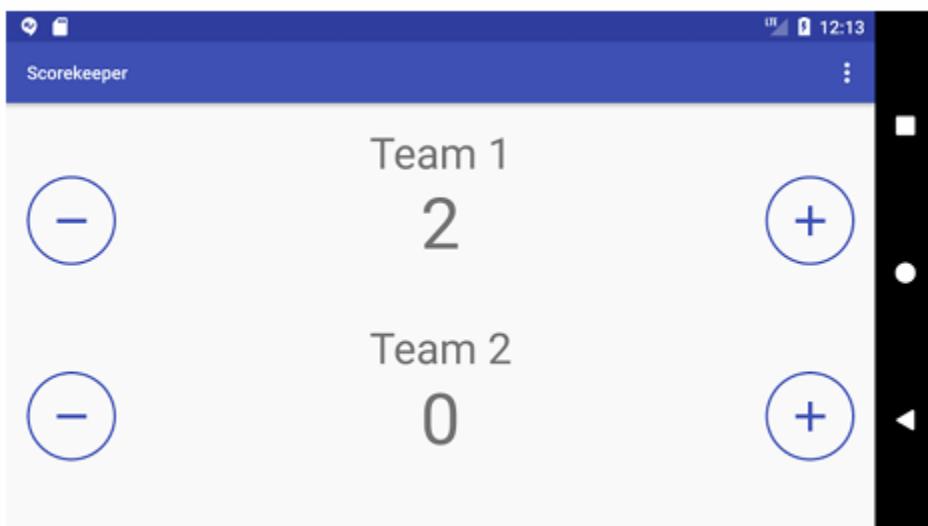
```
@Override
protected void onSaveInstanceState(Bundle outState) {
 // Save the scores.
 outState.putInt(STATE_SCORE_1, mScore1);
 outState.putInt(STATE_SCORE_2, mScore2);
 super.onSaveInstanceState(outState);
}
```

3. Ở cuối phương thức onCreate(), thêm mã để kiểm tra xem có savedInstanceState hay không. Nếu có, hãy khôi phục điểm số cho các phần tử TextView:

```
if (savedInstanceState != null) {
 mScore1 = savedInstanceState.getInt(STATE_SCORE_1);
 mScore2 = savedInstanceState.getInt(STATE_SCORE_2);

 //Set the score text views
 mScoreText1.setText(String.valueOf(mScore1));
 mScoreText2.setText(String.valueOf(mScore2));
}
```

4. Chạy ứng dụng và nhấn vào nút hình ảnh cộng để tăng điểm. Chuyển thiết bị hoặc trình mô phỏng sang hướng ngang để xem điểm số được giữ lại.



Vậy là xong! Xin chúc mừng, bây giờ bạn đã có một ứng dụng Scorekeeper được tạo kiểu tiếp tục hoạt động nếu người dùng thay đổi thiết bị theo hướng ngang hoặc dọc. Mã giải pháp

#### Dự án Android Studio: Thủ thách Scorekeeper Coding

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

Thử thách: Ngay bây giờ, các nút của bạn không hoạt động trực quan vì chúng không thay đổi giao diện khi chúng được nhấn. Android có một loại Drawable khác được gọi là StateListDrawable cho phép sử dụng một đồ họa khác nhau tùy thuộc vào trạng thái của đối tượng.

Đối với vấn đề thử thách này, hãy tạo một tài nguyên Drawable thay đổi nền của ImageButton thành cùng màu với đường viền khi trạng thái của ImageButton được "nhấn".

Bạn cũng nên đặt màu của văn bản bên trong các phần tử ImageButton thành một bộ chọn làm cho nó có màu trắng khi nút được "nhấn".

#### Tóm tắt

- Các yếu tố có thể vẽ nâng cao giao diện người dùng của ứng dụng.
- ShapeDrawable là một hình dạng hình học nguyên thủy được xác định trong tệp XML. Các thuộc tính xác định ShapeDrawable bao gồm màu sắc, hình dạng, khoảng đệm, v.v.
- Nền tảng Android cung cấp một bộ sưu tập lớn các phong cách và chủ đề.
- Sử dụng kiểu có thể giảm lượng mã cần thiết cho các thành phần giao diện người dùng của bạn.
- Một kiểu có thể chỉ định các thuộc tính phổ biến như chiều cao, đệm, màu phông chữ, kích thước phông chữ và màu nền.
- Một phong cách không được bao gồm thông tin liên quan đến bố cục.
- Một kiểu có thể được áp dụng cho Chế độ xem, Hoạt động hoặc toàn bộ ứng dụng. Kiểu được áp dụng cho Hoạt động hoặc toàn bộ ứng dụng phải được xác định trong tệp AndroidManifest.xml.
- Để kế thừa một kiểu, một kiểu mới xác định một thuộc tính cha trong XML.
- Khi bạn áp dụng một kiểu cho một bộ sưu tập các thành phần Chế độ xem trong một hoạt động hoặc trong toàn bộ ứng dụng của bạn, đó được gọi là chủ đề.
- Để áp dụng một chủ đề, bạn sử dụng thuộc tính android:theme. Các khái niệm liên quan Tài liệu khái niệm liên quan nằm trong 5.1: Drawables, styles, and themes .

#### Tìm hiểu thêm

##### Tài liệu Android Studio:

- Hướng dẫn sử dụng Android Studio
- Thêm đồ họa vector đa mật độ

- Tạo biểu tượng ứng dụng với nội dung hình ảnh

Tài liệu dành cho nhà phát triển Studio Android:

- Các phương pháp hay nhất cho giao diện người dùng
- Bộ cục tuyến tính
- Tài nguyên có thể vẽ
- Phong cách và chủ đề
- Các nút
- Bộ cục
- Thư viện hỗ trợ
- Tổng quan về khả năng tương thích màn hình
- Hỗ trợ các kích thước màn hình khác nhau
- Hoạt hình đồ họa có thể vẽ
- Tải bitmap lớn một cách hiệu quả
- Lớp kiểu và chủ đề R.style
- lớp support.v7.appcompat.R.style gồm các kiểu và chủ đề Thiết kế vật liệu:
- Hiểu điều hướng
- Blog dành cho nhà phát triển Android thanh ứng dụng: Thư viện hỗ trợ thiết kế Android Khác:

- Phương tiện: Hướng dẫn chủ đề DayNight
- Video: Udacity - Chủ đề và phong cách
- Android Asset Studio của Roman Nurik
- Tài liệu lướt

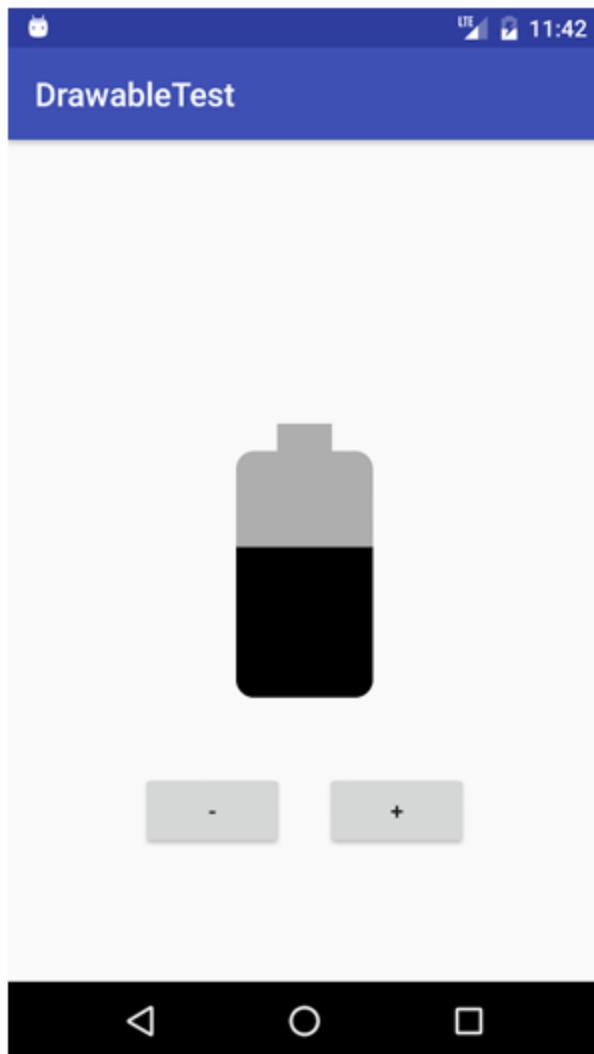
Homework

Xây dựng và chạy ứng dụng

Tạo một ứng dụng hiển thị ImageView và các nút cộng và trừ, như hình dưới đây. ImageView chứa một danh sách mức có thể vẽ là chỉ báo mức pin. Nhấn vào nút cộng hoặc trừ sẽ thay đổi mức của chỉ báo. Sử dụng các biểu tượng pin từ Vector Asset Studio để biểu thị 7 giá trị khác nhau cho mức pin.

Ứng dụng phải có các thuộc tính sau:

- Nút cộng tăng mức, làm cho đèn báo pin có vẻ đầy hơn.
- Nút trừ làm giảm mức, khiến chỉ báo trống một mức.



Trả lời những câu hỏi này

Câu hỏi 1: Bạn sử dụng loại Drawable nào để tạo Button với nền kéo dài phù hợp để phù hợp với văn bản hoặc hình ảnh bên trong Button để nó trông chính xác cho các kích thước và hướng màn hình khác nhau? Chọn một:

- LevelListDrawable
- TransitionDrawable
- StateListDrawable
- NinePatchCó thể vẽ được

Câu hỏi 2: Bạn sử dụng loại Drawable nào để tạo Button hiển thị một nền khi nhấn và một nền khác khi di chuột qua? Chọn một:

- LevelListDrawable
- TransitionDrawable
- StateListDrawable
- NinePatchCó thể vẽ được

Câu hỏi 3 Giả sử bạn muốn tạo một ứng dụng có nền trắng, văn bản tối và thanh hành động tối. Phong cách ứng dụng của bạn kế thừa từ kiểu cơ sở nào? Chọn một:

- Chu đề.AppCompat.Light
- Theme.AppCompat.Dark.NoActionBar
- Theme.AppCompat.Light.DarkActionBar
- Theme.AppCompat.NoActionBar
- Theme.NoActionBar

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm Kiểm tra xem ứng dụng có các tính năng sau không:

- Các nút tăng một biến đếm. Biến count đặt mức trên ImageView , sử dụng phương thức setImageLevel().
- Các cấp độ trong LevelListDrawable đi từ 0 đến 6.
- Trước khi tăng hoặc giảm mức hình ảnh, các phương thức onClick() kiểm tra xem biến count có nằm trong phạm vi của LevelListDrawable (0 đến 6) hay không. Bằng cách này, người dùng không thể đặt cấp độ không tồn tại.

## Bài 5.2: Thẻ và màu sắc Giới thiệu

Nguyên tắc Material Design của Google là một loạt các phương pháp hay nhất để tạo các ứng dụng trực quan và hấp dẫn trực quan. Trong thực tế này, bạn sẽ tìm hiểu cách thêm các tiện ích CardView và FloatingActionButton vào ứng dụng của mình, cách sử dụng hình ảnh một cách hiệu quả và cách sử dụng các phương pháp hay nhất của Material Design để làm cho trải nghiệm của người dùng trở nên thú vị.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các yếu tố giao diện người dùng bằng trình chỉnh sửa bố cục.
- Chính sửa mã bố cục XML và truy cập các phần tử từ mã Java của bạn.
- Tạo trình xử lý nhấp chuột cho một nút bấm.
- Trích xuất văn bản vào tài nguyên chuỗi và thứ nguyên cho tài nguyên thứ nguyên.
- Sử dụng các bản vẽ, kiểu và chủ đề.
- Sử dụng RecyclerView để hiển thị danh sách. Những gì bạn sẽ học
- Đề xuất sử dụng các tiện ích Material Design như

FloatingActionButton và CardView .

- Cách sử dụng hình ảnh hiệu quả trong ứng dụng của bạn.
- Các phương pháp hay nhất được đề xuất để thiết kế bố cục trực quan bằng cách sử dụng màu đậm.

Bạn sẽ làm gì

- Sửa đổi ứng dụng để tuân theo nguyên tắc Material Design.

- Thêm hình ảnh và kiểu dáng vào danh sách RecyclerView.
- Triển khai ItemTouchHelper để thêm chức năng kéo và thả vào ứng dụng của bạn.

Tổng quan về ứng dụng

Ứng dụng MaterialMe là một ứng dụng tin tức thể thao giả với việc triển khai thiết kế rất kém. Bạn sẽ sửa chữa nó để đáp ứng các nguyên tắc thiết kế để tạo ra trải nghiệm người dùng thú vị! Dưới đây là ảnh chụp màn hình của ứng dụng trước và sau khi cải tiến Material Design.



## Nhiệm vụ 1: Tải xuống mã khởi động

Dự án ứng dụng khởi động hoàn chỉnh cho thực tế này có sẵn tại MaterialMe-Starter . Trong tác vụ này, bạn sẽ tải dự án vào Android Studio và khám phá một số tính năng chính của ứng dụng.

### 1.1 Mở và chạy dự án MaterialMe

1. Tải xuống mã MaterialMe-Starter.
2. Mở ứng dụng trong Android Studio.
3. Chạy ứng dụng.

Ứng dụng hiển thị danh sách các tên thể thao với một số văn bản tin tức giữ chỗ cho từng môn thể thao. Bộ cục và phong cách hiện tại của ứng dụng khiến nó gần như không thể sử dụng được: mỗi hàng dữ liệu không được phân tách rõ ràng và không có hình ảnh hoặc màu sắc để thu hút người dùng.

## 1.2 Khám phá ứng dụng

Trước khi thực hiện sửa đổi ứng dụng, hãy khám phá cấu trúc hiện tại của nó. Nó chứa các yếu tố sau:

Sport.java

Lớp này đại diện cho mô hình dữ liệu cho mỗi hàng dữ liệu trong RecyclerView. Ngay bây giờ nó chưa một trường cho tiêu đề của môn thể thao và một trường cho một số thông tin về môn thể thao này.

SportsAdapter.java

Đây là bộ chuyển đổi cho RecyclerView. Nó sử dụng một ArrayList của các đối tượng Sport làm dữ liệu của nó và điền dữ liệu này vào mỗi hàng.

MainActivity.java

MainActivity khởi tạo RecyclerView và bộ điều hợp, đồng thời tạo dữ liệu từ các tệp tài nguyên. Strings.xml

Tệp tài nguyên này chứa tất cả dữ liệu cho ứng dụng, bao gồm tiêu đề và thông tin cho từng môn thể thao list\_item.xml

Tệp bố cục này xác định từng hàng của RecyclerView. Nó bao gồm ba phần tử TextView, một cho mỗi phần dữ liệu (tiêu đề và thông tin cho mỗi môn thể thao) và một phần tử được sử dụng làm nhãn.

## Nhiệm vụ 2: Thêm CardView và hình ảnh

Một trong những nguyên tắc cơ bản của Material Design là sử dụng hình ảnh đậm đà để nâng cao trải nghiệm người dùng. Thêm hình ảnh vào các mục danh sách

RecyclerView là một khởi đầu tốt để tạo ra trải nghiệm người dùng năng động và hấp dẫn.

Khi trình bày thông tin có phương tiện hỗn hợp (như hình ảnh và văn bản), nguyên tắc Material Design khuyên bạn nên sử dụng CardView, đây là FrameLayout với một số tính năng bổ sung (chẳng hạn như độ cao và góc tròn) mang lại giao diện nhất quán trên nhiều ứng dụng và nền tảng khác nhau. CardView là một thành phần giao diện người dùng được tìm thấy trong Thư viện hỗ trợ Android.

Trong phần này, bạn sẽ di chuyển từng mục danh sách vào CardView và thêm Hình ảnh để làm cho ứng dụng tuân thủ nguyên tắc về Material.

## 2.1 Thêm thẻView

CardView không được bao gồm trong SDK Android mặc định, vì vậy bạn phải thêm CardView dưới dạng phần phụ thuộc build.gradle. Thực hiện như sau:

1. Mở tệp build.gradle (Mô-đun: ứng dụng) và thêm dòng sau vào phần phụ thuộc::

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

Phiên bản của thư viện hỗ trợ có thể đã thay đổi kể từ khi viết thực tế này. Cập nhật phần trên lên phiên bản do Android Studio đề xuất và nhấp vào Đồng bộ hóa để đồng bộ hóa các tệp build.gradle của bạn.

2. Mở tệp list\_item.xml và bao quanh LinearLayout gốc bằng android.support.v7.widget.CardView . Di chuyển khai báo lược đồ

Attribute	Value
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:layout_margin	"8dp"

Khai báo lược đồ cần di chuyển sang CardView vì CardView hiện là dạng xem cấp cao nhất trong tệp bố cục của bạn.

3. Chọn Mã > Định dạng lại Mã để định dạng lại mã XML, bây giờ sẽ trông như thế này ở đầu và cuối tệp:

```
<android.support.v7.widget.CardView
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="8dp">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical">
 <!-- Rest of LinearLayout -->
 <!-- TextView elements -->
 </LinearLayout>
</android.support.v7.widget.CardView>
```

4. Chạy ứng dụng. Bây giờ mỗi mục hàng được chứa bên trong CardView, được nâng lên phía trên lớp dưới cùng và tạo ra một cái bóng.

```
<android.support.v7.widget.CardView
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="8dp">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical">
 <!-- Rest of LinearLayout -->
 <!-- TextView elements -->
 </LinearLayout>
</android.support.v7.widget.CardView>
```

## 2.2 Tải xuống hình ảnh

CardView không nhằm mục đích sử dụng riêng với văn bản thuần túy: nó là tốt nhất để hiển thị hỗn hợp nội dung. Bạn có cơ hội tốt để làm cho ứng dụng này thú vị hơn bằng cách thêm hình ảnh biểu ngữ vào mỗi hàng!

Việc sử dụng hình ảnh tồn tại nhiều tài nguyên đối với ứng dụng của bạn: khung Android phải tải toàn bộ hình ảnh vào bộ nhớ ở độ phân giải đầy đủ, ngay cả khi ứng dụng chỉ hiển thị một hình thu nhỏ của hình ảnh.

Trong phần này, bạn sẽ tìm hiểu cách sử dụng thư viện Glide để tải hình ảnh lớn một cách hiệu quả mà không làm cạn kiệt tài nguyên hoặc thậm chí làm hỏng ứng dụng của bạn do các trường hợp ngoại lệ 'Hết bộ nhớ'.

1. Tải xuống tệp zip hình ảnh biểu ngữ.
2. Mở ứng dụng MaterialMe >> src > thư mục > res chính trong trình khám phá tệp của hệ điều hành của bạn và tạo một thư mục có thể vẽ và sao chép các tệp đồ họa riêng lẻ vào thư mục có thể vẽ.
3. Bạn sẽ cần một mảng với đường dẫn đến mỗi hình ảnh để có thể đưa nó vào đối tượng Thẻ thao. Để thực hiện việc này, hãy xác định một mảng chứa tất cả các đường dẫn đến có thể vẽ dưới dạng các mục trong tệp string.xml của bạn. Đảm bảo rằng chúng theo cùng thứ tự với mảng sports\_titles cũng được xác định trong cùng một tệp:

```
<array name="sports_images">
 <item>@drawable/img_baseball</item>
 <item>@drawable/img_badminton</item>
 <item>@drawable/img_basketball</item>
 <item>@drawable/img_bowling</item>
 <item>@drawable/img_cycling</item>
 <item>@drawable/img_golf</item>
 <item>@drawable/img_running</item>
 <item>@drawable/img_soccer</item>
 <item>@drawable/img_swimming</item>
 <item>@drawable/img_tabletennis</item>
 <item>@drawable/img_tennis</item>
</array>
```

### 2.3 Sửa đổi đối tượng Thể thao

Đối tượng Thể thao sẽ cần bao gồm tài nguyên Có thể vẽ tương ứng với môn thể thao đó. Để đạt được điều đó:

1. Thêm một biến thành viên số nguyên vào đối tượng Sport sẽ chứa tài nguyên Drawable:

```
private final int imageResource;
```

2. Sửa đổi hàm khởi tạo để nó lấy một số nguyên làm tham số và gán nó cho biến thành viên:

```
public Sport(String title, String info, int imageResource) {
 this.title = title;
 this.info = info;
 this.imageResource = imageResource;
}
```

3. Tạo một getter cho số nguyên tài nguyên:

```
public int getImageResource() {
 return imageResource;
}
```

### 2.4 Sửa phương thức initializeData()

Trong MainActivity , phương thức initializeData() hiện đã bị hỏng, vì hàm khởi tạo cho đối tượng Sport yêu cầu tài nguyên hình ảnh làm tham số thứ ba.

Một cấu trúc dữ liệu thuận tiện để sử dụng sẽ là TypedArray . TypedArray cho phép bạn lưu trữ một mảng các tài nguyên XML khác. Bằng cách sử dụng TypedArray , bạn sẽ có thể lấy tài nguyên hình ảnh cũng như tiêu đề và thông tin thể thao bằng cách

sử dụng lặp chỉ mục trong cùng một vòng lặp.

1. Trong phương thức initializeData(), lấy TypedArray của các ID tài nguyên bằng cách gọi getResources().obtainTypedArray() , truyền tên của mảng tài nguyên Drawable mà bạn đã xác định trong tệp strings.xml của mình:

```
TypedArray sportsImageResources =
 getResources().obtainTypedArray(R.array.sports_images);
```

Bạn có thể truy cập một phần tử tại chỉ mục **i** trong **TypedArray** bằng cách sử dụng phương thức "get" phù hợp, tùy thuộc vào loại tài nguyên trong mảng. Trong trường hợp cụ thể này, mảng chứa **ID tài nguyên**, vì vậy bạn sử dụng phương thức **getResourceId()**.

2. Sửa mã trong vòng lặp tạo các đối tượng **Sport**, thêm **ID tài nguyên Drawable** phù hợp làm tham số thứ ba bằng cách gọi **getResourceId()** trên **TypedArray**.

```
for(int i=0;i<sportsList.length;i++){
 mSportsData.add(new Sport(sportsList[i],sportsInfo[i],
 sportsImageResources.getResourceId(i,0)));
}
```

3. Dọn dẹp dữ liệu trong mảng đã nhập sau khi bạn đã tạo Mảng dữ liệu thể thao Danh sách

```
sportsImageResources.recycle();
```

## 2.5 Thêm ImageView vào các mục danh sách

- Thay đổi LinearLayout bên trong tệp list\_item.xml thành RelativeLayout và xóa thuộc tính android:orientation.
- Thêm ImageView làm phần tử đầu tiên trong RelativeLayout với các thuộc tính sau:

Attribute	Value
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:id	"@+id/sportsImage"
android:adjustViewBounds	"true"

Thuộc tính adjustViewBounds làm cho ImageView điều chỉnh ranh giới của nó để giữ nguyên tỷ lệ khung hình của hình ảnh.

3. Thêm các thuộc tính sau vào phần tử TextView tiêu đề:

Attribute	Value
android:layout_alignBottom	"@+id/sportsImage"
android:theme	"@style/ThemeOverlay.AppCompat.Dark"

4. Thêm các thuộc tính sau vào phần tử newsTitle TextView:

Attribute	Value
android:layout_below	"@+id/sportsImage"
android:textColor	"?android:textColorSecondary"

android:layout_below	"@+id/sportsImage"
android:textColor	"?android:textColorSecondary"

5. Thêm các thuộc tính sau vào phần tử subTitle TextView:

Attribute	Value
android:layout_below	"@+id/newsTitle"

Dấu chấm hỏi trong thuộc tính **textColor** ở trên ("?android:textColorSecondary") có nghĩa là hệ thống sẽ áp dụng giá trị từ chủ đề (**theme**) hiện tại. Trong trường hợp này, thuộc tính này được kế thừa từ chủ đề "**Theme.AppCompat.Light.DarkActionBar**", trong đó định nghĩa màu là **xám nhạt**, thường được sử dụng cho tiêu đề phụ.

Sau khi tải xuống hình ảnh và thiết lập **ImageView**, bước tiếp theo là chỉnh sửa **SportsAdapter** để tải hình ảnh vào **ImageView** trong phương thức **onBindViewHolder()**. Nếu thực hiện theo cách này, bạn có thể thấy ứng dụng bị lỗi do "**Out of Memory**". Framework Android phải tải hình ảnh vào bộ nhớ mỗi lần với độ phân giải đầy đủ, bắt kể kích thước hiển thị của **ImageView**.

Có nhiều cách để giảm mức tiêu thụ bộ nhớ khi tải hình ảnh, nhưng một trong những cách dễ nhất là sử dụng **thư viện tải hình ảnh** như **Glide**, điều mà bạn sẽ thực hiện trong bước này. **Glide** sử dụng xử lý nền (**background processing**) cùng với một số quy trình phức tạp khác để giảm yêu cầu bộ nhớ khi tải hình ảnh. Nó cũng bao gồm một số tính năng hữu ích như hiển thị hình ảnh chờ (**placeholder images**) trong khi hình ảnh chính được tải.

**Lưu ý:** Để tìm hiểu thêm về cách giảm mức tiêu thụ bộ nhớ trong ứng dụng, hãy xem tài liệu **Loading Large Bitmaps Efficiently**.

1. Mở tệp build.gradle (Mô-đun: ứng dụng) và thêm phần phụ thuộc sau cho Glide trong phần phụ thuộc:

**dependencies section:**

```
implementation 'com.github.bumptech.glide:glide:3.7.0'
```

2. Mở SportsAdapter và thêm một biến trong lớp ViewHolder cho ImageView:

```
private ImageView mSportsImage;
```

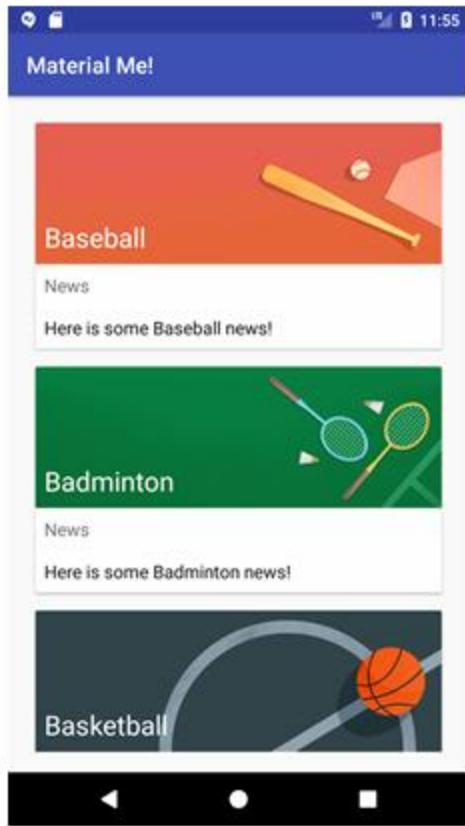
3. Khởi tạo biến trong hàm khởi tạo ViewHolder cho lớp ViewHolder:

```
mSportsImage = itemView.findViewById(R.id.sportsImage);
```

4. Thêm dòng mã sau vào phương thức bindTo() trong lớp ViewHolder để lấy tài nguyên hình ảnh từ đối tượng Sport và tải nó vào ImageView bằng Glide:

```
Glide.with(mContext).load(currentSport.getImageResource()).into(mSportsImage);
```

5. Chạy ứng dụng, các mục danh sách của bạn bây giờ sẽ có đồ họa đậm giúp trải nghiệm người dùng trở nên năng động và thú vị!



Đó là tất cả những gì cần thiết để tải một hình ảnh bằng Glide. Glide cũng có một số tính năng bổ sung cho phép bạn thay đổi kích thước, biến đổi và tải hình ảnh theo nhiều cách khác nhau. Truy cập trang github Glide để tìm hiểu thêm.

### Nhiệm vụ 3: Làm cho CardView của bạn có thể vuốt, di chuyển và có thể nhấp

Khi người dùng nhìn thấy thẻ trong một ứng dụng, họ có kỳ vọng về cách thẻ hoạt động. Nguyên tắc Material Design nói rằng:

- Thẻ có thể bị loại bỏ, thường bằng cách quét thẻ đi.
- Danh sách các thẻ có thể được sắp xếp lại bằng cách giữ và kéo các thẻ.
- Nhấn vào thẻ sẽ cung cấp thêm thông tin chi tiết.

Bây giờ bạn sẽ triển khai các hành vi này trong ứng dụng của mình.

#### 3.1 Thực hiện vuốt để loại bỏ

Android SDK bao gồm một lớp có tên ItemTouchHelper được sử dụng để xác định điều gì sẽ xảy ra với các mục trong danh sách RecyclerView khi người dùng thực hiện các hành động chạm khác nhau, chẳng hạn như vuốt hoặc kéo và thả. Một số trường hợp sử dụng phổ biến đã được triển khai trong một tập hợp các phương thức trong ItemTouchHelper.SimpleCallback .

ItemTouchHelper.SimpleCallback cho phép bạn xác định hướng nào được hỗ trợ để vuốt và di chuyển các mục trong danh sách, đồng thời triển khai hành vi vuốt và di chuyển.

Thực hiện như sau:

1. Mở MainActivity và tạo một đối tượng ItemTouchHelper mới trong phương thức onCreate() ở cuối, bên dưới phương thức initializeData(). Đối với đối số của nó, bạn sẽ tạo một thực thể mới của ItemTouchHelper.SimpleCallback . Khi bạn nhập ItemTouchHelper mới , các đè xuất sẽ xuất hiện. Chọn ItemTouchHelper.SimpleCallback{ ... } từ menu gợi ý. Android Studio điền vào các phương thức bắt buộc: onMove() và onSwiped() như hình bên dưới.

Nếu các phương pháp cần thiết không được thêm tự động, hãy nhấp vào bóng đèn màu đỏ ở lề trái và chọn Triển khai phương pháp.

Hàm khởi tạo SimpleCallback sẽ được gạch chân màu đỏ vì bạn chưa cung cấp các tham số bắt buộc: hướng mà bạn dự định hỗ trợ để di chuyển và vuốt các mục danh sách, tương ứng.

2. Vì chúng tôi chỉ triển khai vuốt để loại bỏ vào lúc này, bạn nên chuyển 0 cho các hướng di chuyển được hỗ trợ và ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT để biết các hướng vuốt được hỗ trợ:

```
ItemTouchHelper helper = new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
```

3. Bây giờ bạn phải thực hiện hành vi mong muốn trong onSwiped() . Trong trường hợp này, quét thẻ sang trái hoặc phải sẽ xóa thẻ khỏi danh sách. Gọi remove() trên tập dữ liệu, truyền vào chỉ mục thích hợp bằng cách lấy vị trí từ ViewHolder:

```
mSportsData.remove(viewHolder.getAdapterPosition());
```

4. Để cho phép RecyclerView tạo hiệu ứng cho việc xóa đúng cách, bạn cũng phải gọi notifyItemRemoved() , một lần nữa chuyển chỉ mục thích hợp bằng cách lấy vị trí từ ViewHolder:

```
mAdapter.notifyItemRemoved(viewHolder.getAdapterPosition());
```

5. Bên dưới đối tượng ItemTouchHelper mới trong phương thức onCreate() cho MainActivity , hãy gọi attachToRecyclerView() trên thực thể ItemTouchHelper để thêm nó vào RecyclerView của bạn:

```
helper.attachToRecyclerView(mRecyclerView);
```

6. Chạy ứng dụng của bạn, bây giờ bạn có thể vuốt các mục danh sách sang trái và phải để xóa chúng!

### 3.2 Thực hiện kéo và thả

Bạn cũng có thể triển khai chức năng kéo và thả bằng cách sử dụng cùng một SimpleCallback . Đối số đầu tiên của SimpleCallback xác định hướng mà ItemTouchHelper hỗ trợ để di chuyển các đối tượng xung quanh. Thực hiện như sau:

- Thay đổi đối số đầu tiên của SimpleCallback từ 0 để bao gồm mọi hướng, vì chúng ta muốn có thể kéo và thả ở bất cứ đâu:

```
ItemTouchHelper helper = new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT | ItemTouchHelper.DOWN | ItemTouchHelper.UP, ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
```

- Trong phương thức onMove(), lấy chỉ mục ban đầu và mục tiêu từ đối số thứ hai và thứ ba được truyền vào (tương ứng với người nắm giữ chế độ xem ban đầu và mục tiêu).

```
int from = viewHolder.getAdapterPosition();
int to = target.getAdapterPosition();
```

3. Hoán đổi các mục trong tập dữ liệu bằng cách gọi Collections.swap() và truyền vào tập dữ liệu, cũng như các chỉ mục ban đầu và cuối cùng:

```
Collections.swap(mSportsData, from, to);
```

4. Thông báo cho bộ điều hợp rằng mục đã được di chuyển, chuyển các chỉ mục cũ và mới và thay đổi câu lệnh return thành true

```
mAdapter.notifyItemMoved(from, to);
return true;
```

5. Chạy ứng dụng của bạn. Giờ đây, bạn có thể xóa các mục trong danh sách của mình bằng cách vuốt chúng sang trái hoặc phải hoặc sắp xếp lại chúng bằng cách nhấn và giữ để kích hoạt chế độ Kéo và Thả.

### 3.3 Triển khai bố cục DetailActivity

Theo hướng dẫn Material Design, thẻ được sử dụng để cung cấp điểm vào thông tin chi tiết hơn. Bạn có thể thấy mình chạm vào các thẻ để xem thêm thông tin về các môn thể thao, bởi vì đó là cách bạn mong đợi các thẻ hoạt động.

Trong phần này, bạn sẽ thêm một Hoạt động chi tiết sẽ được khởi chạy khi bất kỳ mục danh sách nào được nhấn. Đối với thực tế này, Hoạt động chi tiết sẽ chứa tên và hình ảnh của mục danh sách bạn đã nhấp vào, nhưng sẽ chỉ chứa văn bản chi tiết trình giữ chỗ chung chung, vì vậy bạn không phải tạo chi tiết tùy chỉnh cho từng mục danh sách.

1. Tạo một **Activity** mới bằng cách vào **File > New > Activity > Empty Activity**.
2. Đặt tên **DetailActivity** và chấp nhận tất cả các thiết lập mặc định.
3. Mở tệp **activity\_detail.xml** vừa tạo và thay đổi **ViewGroup** gốc thành **RelativeLayout**, như trong các bài tập trước.
4. Xóa dòng khai báo `xmlns:app="http://schemas.android.com/apk/res-auto"` khỏi **RelativeLayout**.

5. Sao chép tất cả các phần tử **TextView** và **ImageView** từ tệp **list\_item.xml** sang tệp **activity\_detail.xml**.
6. Thêm từ "**Detail**" vào tham chiếu trong thuộc tính **android:id** để phân biệt với ID trong **list\_item.xml**. Ví dụ: đổi ID của **ImageView** từ **sportsImage** thành **sportsImageDetail**.
7. Trong tất cả các phần tử **TextView** và **ImageView**, thay đổi tất cả các tham chiếu đến **ID** dùng để sắp xếp vị trí tương đối (**layout\_below**) để sử dụng **ID** có từ "**Detail**".
8. Đối với **TextView** có ID **subTitleDetail**, xóa chuỗi văn bản giữ chỗ và dán một đoạn văn bản chung để thay thế (ví dụ: một số đoạn **LOrem Ipsum**). Trích xuất văn bản này thành một **string resource**.
9. Thay đổi **padding** của các phần tử **TextView** thành **16dp**.
10. Bọc toàn bộ **RelativeLayout** trong một **ScrollView**. Thêm các thuộc tính **layout\_height** và **layout\_width** cần thiết, đồng thời thêm **xmlns:android="http://schemas.android.com/apk/res/android"** vào cuối khai báo **ScrollView**.
11. Thay đổi thuộc tính **layout\_height** của **RelativeLayout** thành "**wrap\_content**".

Hai phần tử đầu tiên của bố cục **activity\_detail.xml** bây giờ sẽ trông như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

 <RelativeLayout xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 tools:context="com.example.android.materialme.DetailActivity">
```

### **3.4 Triển khai chế độ xem chi tiết và trình nghe nhấp chuột**

Làm theo các bước sau để triển khai chế độ xem chi tiết và trình nghe nhấp chuột:

1. Mở **SportsAdapter** và thay đổi lớp bên trong **ViewHolder**, lớp này đã mở rộng **RecyclerView.ViewHolder**, để triển khai **View.OnClickListener** và triển khai phương thức bắt buộc (**onClick()**).

```
class ViewHolder extends RecyclerView.ViewHolder
 implements View.OnClickListener{
 // Rest of ViewHolder code.
 //
 @Override
 public void onClick(View view) {
 }
}
```

2. Đặt OnClickListener thành itemView trong hàm tạo ViewHolder. Toàn bộ constructor bây giờ sẽ trông như thế này:

```
ViewHolder(View itemView) {
 super(itemView);

 //Initialize the views
 mTitleText = itemView.findViewById(R.id.title);
 mInfoText = itemView.findViewById(R.id.subTitle);
 mSportsImage = itemView.findViewById(R.id.sportsImage);

 // Set the OnClickListener to the entire view.
 itemView.setOnClickListener(this);
}
```

3. Trong phương thức onClick(), lấy đối tượng Sport cho mục đã được nhấp bằng getAdapterPosition():

```
Sport currentSport = mSportsData.get(getAdapterPosition());
```

4. Trong cùng một phương thức, hãy thêm một Intent khởi chạy DetailActivity , đặt tiêu đề và image\_resource làm phần bổ sung trong Intent và gọi startActivity() trên biến mContext, truyền vào Intent mới .

```
Intent detailIntent = new Intent(mContext, DetailActivity.class);
detailIntent.putExtra("title", currentSport.getTitle());
detailIntent.putExtra("image_resource",
 currentSport.getImageResource());
mContext.startActivity(detailIntent);
```

5. Mở DetailActivity và khởi tạo ImageView và tiêu đề TextView trong onCreate():

```
TextView sportsTitle = findViewById(R.id.titleDetail);
ImageView sportsImage = findViewById(R.id.sportsImageDetail);
```

6. Lấy tiêu đề từ Ý định đến và đặt nó thành TextView:

```
sportsTitle.setText(getIntent().getStringExtra("title"));
```

## 7. Sử dụng Glide để tải hình ảnh vào ImageView

```
Glide.with(this).load(getIntent().getIntExtra("image_resource", 0))
.into(sportsImage);
```

## 8. Chạy ứng dụng. Nhấn vào mục danh sách bây giờ sẽ khởi chạy DetailActivity .

# Nhiệm vụ 4: Thêm FAB và chọn bảng màu Material Design

Một trong những nguyên tắc đầu tiên sau Material Design là sử dụng các yếu tố nhất quán trên các ứng dụng và nền tảng để người dùng nhận ra các mẫu và biết cách sử dụng chúng. Bạn đã sử dụng một yếu tố như vậy: Nút hành động nổi (FAB). FAB là một nút tròn nổi phía trên phần còn lại của giao diện người dùng. Nó được sử dụng để quảng bá một hành động cụ thể cho người dùng, một hành động rất có thể được sử dụng trong một hoạt động nhất định. Trong nhiệm vụ này, bạn sẽ tạo một FAB đặt lại tập dữ liệu về trạng thái ban đầu.

## 4.1 Thêm FAB

Nút hành động nổi là một phần của Thư viện hỗ trợ thiết kế.

1. Mở tệp build.gradle (Mô-đun: ứng dụng) và thêm dòng mã sau cho thư viện hỗ trợ thiết kế trong phần phụ thuộc:

```
implementation 'com.android.support:design:26.1.0'
```

2. Thêm biểu tượng cho FAB bằng cách nhấp chuột phải (hoặc Control khi nhấp vào) thư mục res trong ngăn Project > Android và chọn New > Vector Asset . FAB sẽ đặt lại nội dung của RecyclerView , vì vậy biểu tượng làm mới sẽ thực hiện: . Thay đổi tên thành ic\_reset , bấm Tiếp theo và bấm Kết thúc.

3. Mở activity\_main.xml và thêm FloatingActionButton với các thuộc tính sau:

Attribute	Value
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_alignParentBottom	"true"
android:layout_alignParentRight	"true"
android:layout_alignParentEnd	"true"
android:layout_margin	"16dp"

android:src	"@drawable/ic_reset"
android:tint	"@android:color/white"
android:onClick	resetSports

4. Mở MainActivity và thêm phương thức resetSports() với một câu lệnh để gọi initializeData() để đặt lại dữ liệu.

5. Chạy ứng dụng. Bây giờ bạn có thể đặt lại dữ liệu bằng cách nhấn vào FAB.

Vì Hoạt động bị hủy và tạo lại khi cấu hình thay đổi, nên việc xoay thiết bị sẽ đặt lại dữ liệu trong quá trình triển khai này. Để các thay đổi được duy trì (như trong trường hợp sắp xếp lại hoặc xóa dữ liệu), bạn sẽ phải triển khai onSaveInstanceState() hoặc ghi các thay đổi vào một nguồn liên tục (như cơ sở dữ liệu hoặc SharedPreferences, được mô tả trong các bài học khác).

## 4.2 Chọn bảng màu

Material Design khuyên bạn nên chọn ít nhất ba màu này cho ứng dụng của bạn:

- **Màu cơ bản.** Thanh này được sử dụng tự động để tô màu thanh ứng dụng của bạn (thanh chứa tiêu đề ứng dụng của bạn).
- **Màu tối cơ bản.** Một bóng tối hơn cùng màu. Điều này được sử dụng cho thanh trạng thái phía trên thanh ứng dụng, trong số những thứ khác.
- **Một màu nhán.** Một màu tương phản tốt với màu cơ bản. Điều này được sử dụng cho các điểm nổi bật khác nhau, nhưng nó cũng là màu mặc định của FAB.

Khi chạy ứng dụng của mình, bạn có thể nhận thấy rằng màu FAB và màu thanh ứng dụng đã được đặt. Trong nhiệm vụ này, bạn sẽ tìm hiểu nơi các màu này được đặt. Bạn có thể sử dụng Hướng dẫn màu vật liệu để chọn một số màu để thử nghiệm.

1. Trong ngăn Project > Android, điều hướng đến tệp styles.xml của bạn (nằm trong thư mục giá trị). Kiểu AppTheme xác định ba màu theo mặc định: colorPrimary , colorPrimaryDark và colorAccent . Các kiểu này được xác định bởi các giá trị từ tệp colors.xml.

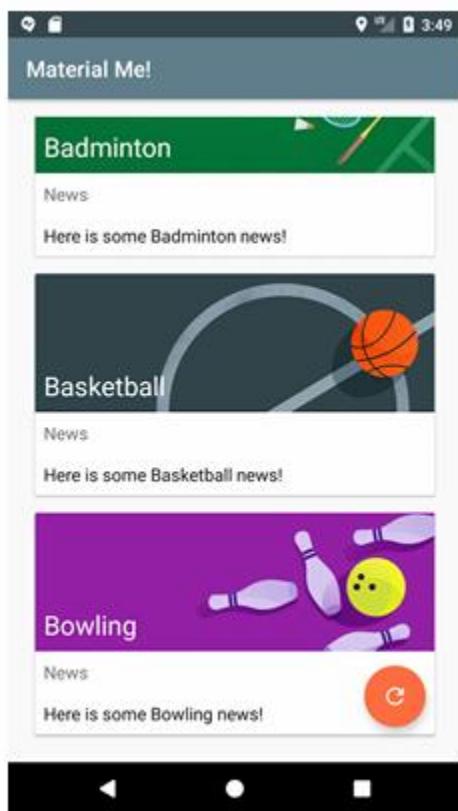
2. Chọn một màu từ Hướng dẫn màu vật liệu để sử dụng làm màu cơ bản của bạn, chẳng hạn như màu #607D8B (trong mẫu màu Xám Xanh). Nó phải nằm trong phạm vi 300-700 của mẫu màu để bạn vẫn có thể chọn điểm nhán và màu tối thích hợp.

3. Mở tệp colors.xml và sửa đổi giá trị hex colorPrimary để phù hợp với màu bạn đã chọn.

4. Chọn một bóng tối hơn cùng màu để sử dụng làm màu tối chính của bạn, chẳng hạn như #37474F . Một lần nữa, hãy sửa đổi giá trị hex colors.xml cho colorPrimaryDark để phù hợp.

5. Chọn một màu nhấn cho FAB của bạn từ các màu có giá trị bắt đầu bằng A và có màu tương phản tốt với màu cơ bản (như Deep Orange A200). Thay đổi giá trị colorAccent trong colors.xml để khớp.

6. Chạy ứng dụng. Thanh ứng dụng và FAB hiện đã thay đổi để phản ánh bảng màu mới!



Nếu bạn muốn thay đổi màu của FAB thành màu khác với màu chủ đề, hãy sử dụng thuộc tính app:backgroundTint. Xin lưu ý rằng thao tác này sử dụng không gian tên app: và Android Studio sẽ nhắc bạn thêm một câu lệnh để xác định không gian tên.

Mã giải pháp

dự án Android Studio:

Thử thách mã hóa MaterialMe

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

Thử thách mã hóa 1

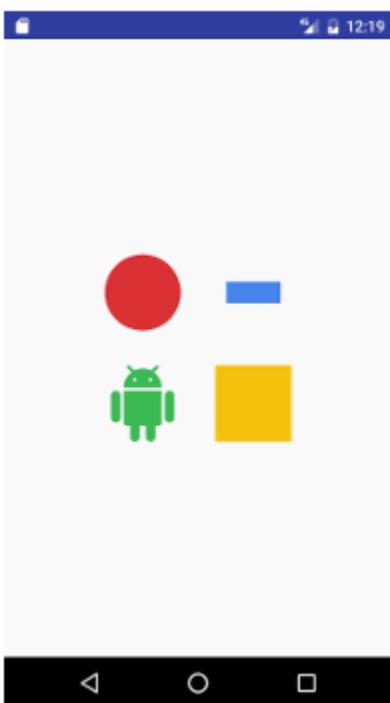
Thử thách này bao gồm hai cài đặt nhỏ đối với ứng dụng MaterialMe:

- Thêm chi tiết thực vào đối tượng Thể thao và chuyển các chi tiết đến DetailActivity
- Triển khai một cách để đảm bảo rằng trạng thái của ứng dụng liên tục sau khi thay đổi hướng.

Thử thách mã hóa 2:

Tạo một ứng dụng với bốn hình ảnh được sắp xếp trong một lưới ở giữa bố cục của bạn. Tạo ba hình nền màu đồng nhất đầu tiên với các hình dạng khác nhau (hình vuông, hình tròn, đường thẳng) và hình nền thứ tư là Biểu tượng thiết kế vật liệu Android. Làm cho mỗi hình ảnh này phản hồi các nhấp chuột như sau:

1. Một trong các khối màu khởi chạy lại Hoạt động bằng cách sử dụng hoạt ảnh Phát nổ cho cả chuyển tiếp vào và thoát.
2. Khởi chạy lại Hoạt động từ một khối màu khác, lần này sử dụng chuyển đổi Fade.
3. Chạm vào khối màu thứ ba sẽ bắt đầu hoạt ảnh tại chỗ của Chế độ xem (chẳng hạn như xoay).
4. Cuối cùng, chạm vào biểu tượng Android sẽ bắt đầu một Hoạt động phụ với Chuyển đổi phần tử được chia sẻ hoán đổi khối Android với một trong các khối khác.



Mã giải pháp Thủ thách 2  
dự án Android Studio:

Tóm tắt *TransitionsandAnimations*

- CardView là một bộ cục tốt để sử dụng để trình bày thông tin có phương tiện hỗn hợp (chẳng hạn như hình ảnh và văn bản).
- CardView là một thành phần giao diện người dùng được tìm thấy trong Thư viện hỗ trợ Android.
- CardView không được thiết kế chỉ dành cho các phần tử View con văn bản. • Tải hình ảnh trực tiếp vào ImageView tốn nhiều bộ nhớ, vì hình ảnh được tải ở độ phân giải đầy đủ. Để tải hình ảnh vào ứng dụng của bạn một cách hiệu quả, hãy sử dụng thư viện tải hình ảnh như Glide .

- SDK Android có một lớp gọi là ItemTouchHelper giúp ứng dụng của bạn có được thông tin về các sự kiện chạm, vuốt và kéo và thả trong giao diện người dùng của bạn.
- FloatingActionButton (FAB) tập trung người dùng vào một hành động cụ thể và "nổi" trong giao diện người dùng của bạn.
- Thiết kế vật liệu là một tập hợp các nguyên tắc hướng dẫn để tạo ra các ứng dụng nhất quán, trực quan và vui tươi.
  - Theo Material Design, bạn nên chọn ba màu cho ứng dụng của mình: màu cơ bản, màu tối cơ bản và màu nhấn.
  - Material Design thúc đẩy việc sử dụng hình ảnh và màu sắc đậm để nâng cao trải nghiệm người dùng. Nó cũng thúc đẩy các yếu tố nhất quán trên các nền tảng, ví dụ bằng cách sử dụng các tiện ích CardView và FAB.
  - Sử dụng Material Design để tạo chuyển động trực quan, có ý nghĩa cho các yếu tố giao diện người dùng như thẻ có thể loại bỏ hoặc sắp xếp lại.

#### Khái niệm liên quan

Tìm hiểu thêm tài liệu về Android Studio:

- Hướng dẫn sử dụng Android Studio
- Thêm đồ họa vector đa mật độ
- Tạo biểu tượng ứng dụng với Image Asset Studio Tài liệu dành cho nhà phát triển Android:

- Giao diện người dùng & Điều hướng
- Bộ cục tuyến tính
- FloatingActionButton
- Tài nguyên có thể vẽ
- Xem hoạt ảnh
- Thư viện hỗ trợ
- Tạo hoạt ảnh cho đồ họa có thể vẽ
- Tải bitmap lớn một cách hiệu quả Thiết kế vật liệu:
- Thiết kế vật liệu cho Android
- Trình tạo bảng màu Thiết kế vật liệu
- Tạo danh sách với RecyclerView
- Thanh ứng dụng
- Xác định hoạt ảnh tùy chỉnh Blog dành cho nhà phát triển Android: Thư viện hỗ trợ thiết kế Android Khác:

- Tài liệu lướt
- Trang github lướt

#### Bài tập về nhà

Xây dựng và chạy ứng dụng Mở ứng dụng MaterialMe.

1. Tạo quá trình chuyển đổi phần tử được chia sẻ giữa MainActivity và DetailActivity, với hình ảnh biểu ngữ cho môn thể thao làm phần tử được chia sẻ.

2. Nhấp vào một mục danh sách trong ứng dụng MaterialMe sẽ kích hoạt quá trình chuyển đổi. Hình ảnh biểu ngữ từ thẻ sẽ di chuyển lên đầu màn hình trong chế độ xem chi tiết.

Trả lời các câu hỏi sau:

Câu hỏi 1: Thuộc tính màu nào trong phong cách của bạn xác định màu của thanh trạng thái phía trên thanh ứng dụng? Chọn một:

- colorPrimary
- colorPrimaryDark
- colorAccent
- colorAccentDark Câu hỏi 2 FloatingActionButton thuộc thư viện hỗ trợ nào? Chọn một:

- Thư viện hỗ trợ v4
- Thư viện hỗ trợ v7
- Thư viện hỗ trợ thiết kế
- Thư viện hỗ trợ nút tùy chỉnh

Câu hỏi 3 Trong bảng màu Material Design, bạn nên sử dụng màu nào làm màu cơ bản cho thương hiệu trong ứng dụng của mình? Chọn một:

- Bất kỳ bóng màu nào bắt đầu bằng A.
- Bất kỳ màu nào được dán nhãn 200.
- Bất kỳ màu sắc nào được dán nhãn 500.
- Bất kỳ màu nào được dán nhãn 900. Gửi ứng dụng của bạn để chấm điểm Hướng dẫn cho người chấm điểm Kiểm tra xem ứng dụng có các tính năng sau không:
  - Chuyển đổi nội dung cửa sổ được bật trong chủ đề ứng dụng.
  - Chuyển đổi phần tử được chia sẻ được chỉ định trong kiểu ứng dụng.
  - Chuyển đổi được định nghĩa là một tài nguyên XML.
  - Tên chung được gán cho các phần tử được chia sẻ trong cả hai bộ cục với thuộc tính android:transitionName.
- Mã sử dụng phương thức ActivityOptions.makeSceneTransitionAnimation().

### Bài 5.3: Giới thiệu bộ cục thích ứng

Ứng dụng MaterialMe mà bạn đã tạo trong chương trước không xử lý đúng các thay đổi hướng thiết bị từ chế độ dọc (dọc) sang chế độ ngang (ngang). Trên máy tính bảng, kích thước phông chữ quá nhỏ và không gian không được sử dụng hiệu quả.

Khung Android có một cách để giải quyết cả hai vấn đề. Bộ hạn định tài nguyên cho phép thời gian chạy Android sử dụng các tệp tài nguyên XML thay thế tùy thuộc vào cấu hình thiết bị — hướng, ngôn ngữ và các bộ hạn định khác. Để biết danh sách đầy đủ các điều kiện có sẵn, hãy xem Cung cấp tài nguyên thay thế .

Trong thực tế này, bạn tối ưu hóa việc sử dụng không gian trong ứng dụng MaterialMe để ứng dụng hoạt động tốt ở chế độ ngang và trên máy tính bảng. Trong một thực tế khác về cách sử dụng trình chỉnh sửa bộ cục, bạn đã học cách tạo các biến thể bộ cục cho hướng ngang và máy tính bảng. Trong thực tế này, bạn sử dụng bộ cục thích ứng, là bộ cục hoạt động tốt cho các kích thước và hướng màn hình khác nhau, các thiết bị khác nhau, ngôn ngữ và ngôn ngữ khác nhau cũng như các phiên bản Android khác nhau.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các yếu tố giao diện người dùng bằng trình chỉnh sửa bộ cục.
- Chính sửa mã bộ cục XML và truy cập các phần tử từ mã Java của bạn.
- Tạo trình xử lý nhấp chuột cho một nút bấm.
- Sử dụng các bản vẽ, kiểu và chủ đề.
- Trích xuất văn bản vào tài nguyên chuỗi và thứ nguyên cho tài nguyên thứ nguyên.

Những gì bạn sẽ học

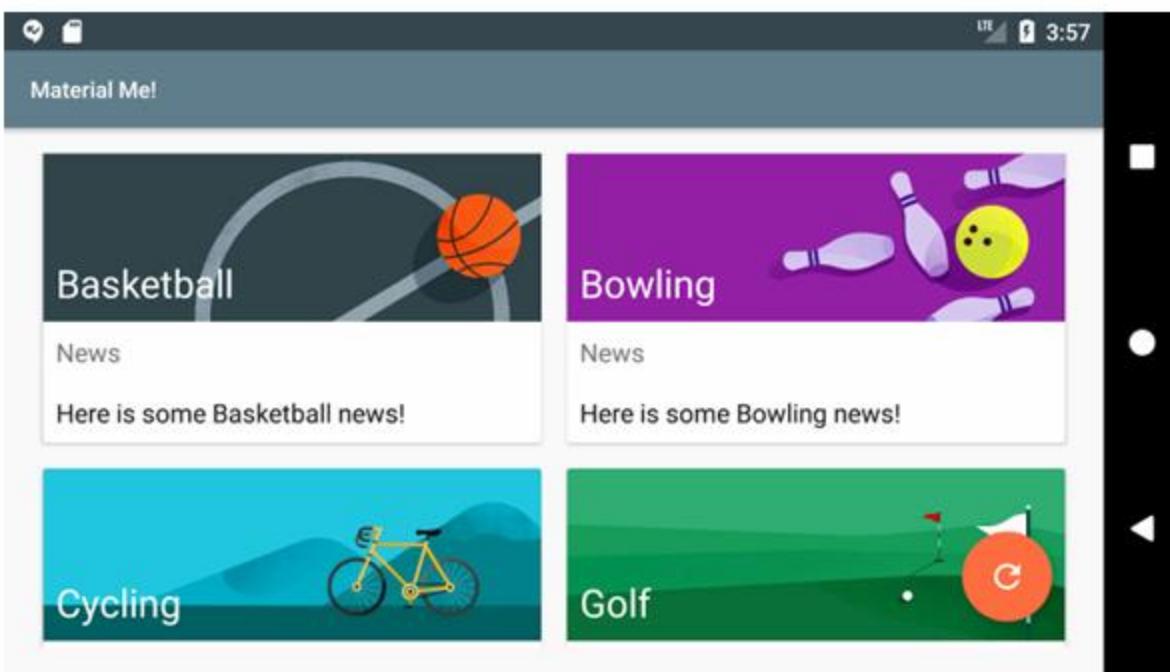
Bạn sẽ học cách:

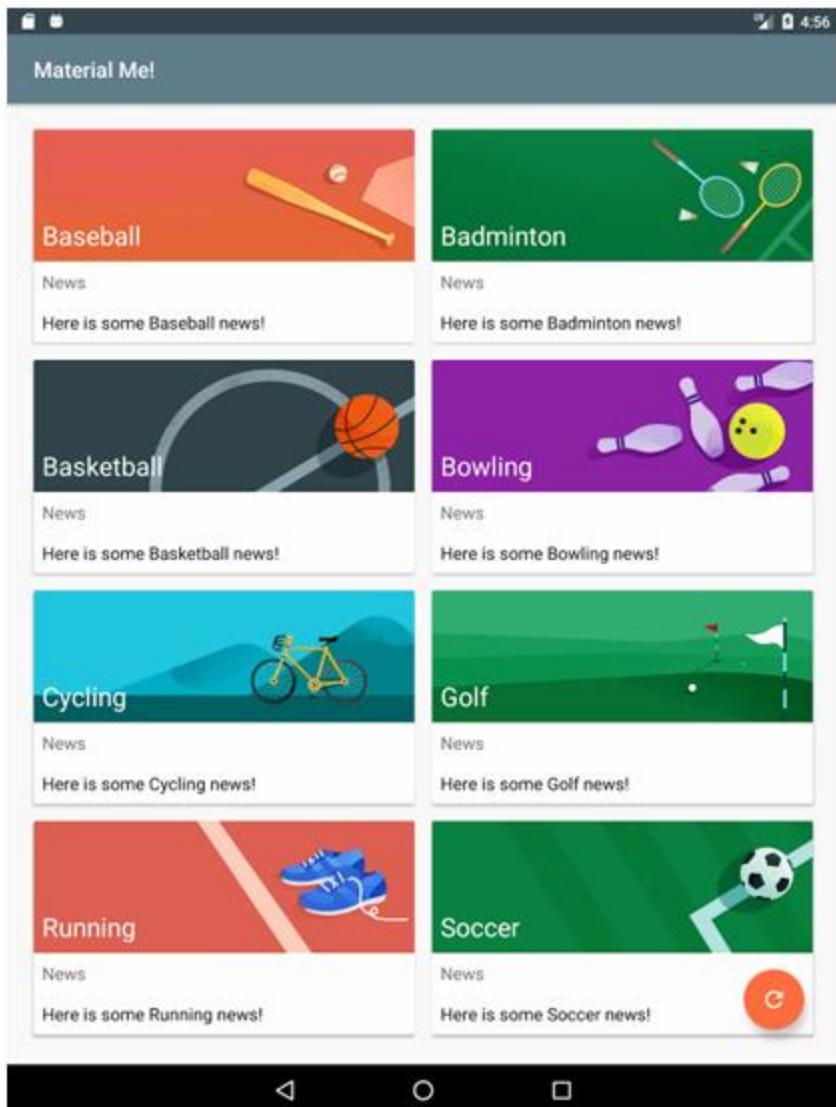
- Tạo tài nguyên thay thế cho các thiết bị ở chế độ ngang.
- Tạo tài nguyên thay thế cho máy tính bảng.
- Tạo tài nguyên thay thế cho các ngôn ngữ khác nhau. Những gì bạn sẽ làm • Cập nhật ứng dụng MaterialMe để sử dụng không gian tốt hơn ở chế độ ngang.
- Thêm bộ cục thay thế cho máy tính bảng.
- Bản địa hóa nội dung ứng dụng của bạn.

Tổng quan về ứng dụng

Ứng dụng MaterialMe được cập nhật sẽ bao gồm bộ cục được cải thiện cho chế độ ngang trên điện thoại. Nó cũng sẽ bao gồm bộ cục được cải thiện cho các chế độ dọc và ngang trên máy tính bảng, đồng thời nó sẽ cung cấp nội dung bản địa hóa cho người dùng bên ngoài Hoa Kỳ.

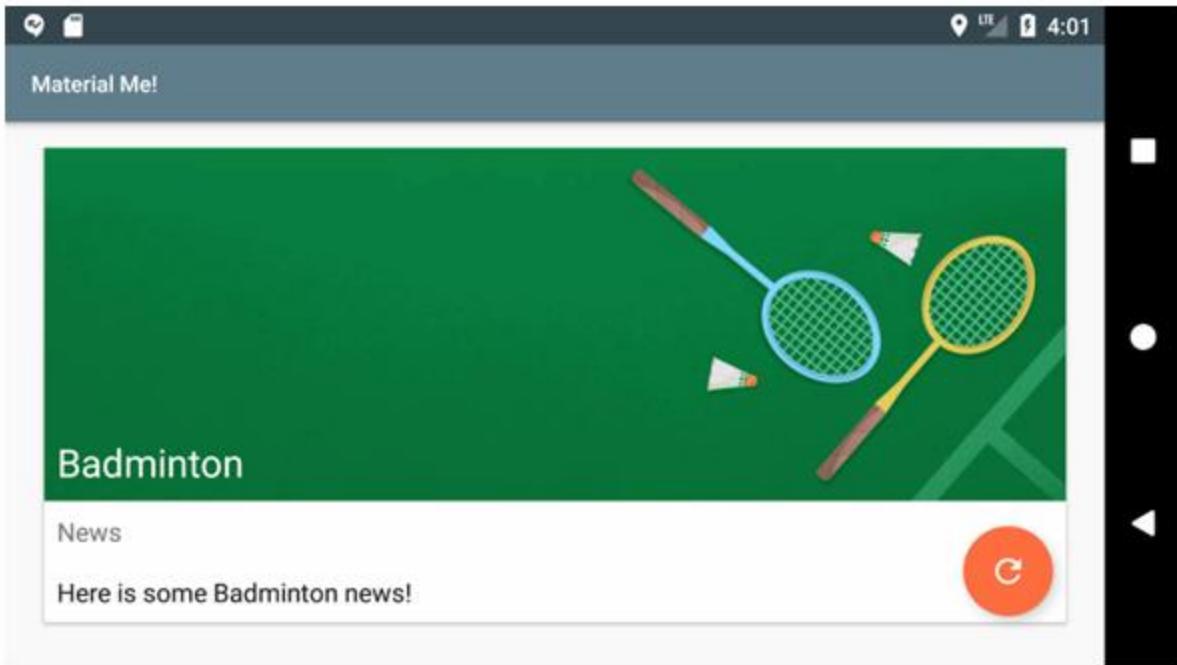
Ảnh chụp màn hình bên dưới cho thấy điện thoại chạy ứng dụng MaterialMe được cập nhật theo hướng ngang:





## Nhiệm vụ 1: Hỗ trợ định hướng ngang

Bạn có thể nhớ rằng khi người dùng thay đổi hướng của thiết bị, khung Android sẽ hủy và tạo lại hoạt động hiện tại. Định hướng mới thường có yêu cầu bố cục khác với hướng ban đầu. Ví dụ: ứng dụng MaterialMe trông đẹp ở chế độ dọc, nhưng không tận dụng tối ưu màn hình ở chế độ ngang. Với chiều rộng lớn hơn ở chế độ ngang, hình ảnh trong mỗi mục danh sách lấn át văn bản mang lại trải nghiệm người dùng kém.



Trong nhiệm vụ này, bạn sẽ tạo một tệp tài nguyên thay thế sẽ thay đổi giao diện của ứng dụng khi nó được sử dụng theo hướng ngang.

### 1.1 Thay đổi thành GridLayoutManager

Bố cục chứa các mục danh sách thường trông không cân bằng ở chế độ ngang khi các mục danh sách bao gồm hình ảnh có chiều rộng đầy đủ. Một giải pháp tốt là sử dụng lưới thay vì danh sách tuyến tính khi hiển thị các phần tử CardView ở chế độ ngang. Nhớ lại rằng các mục trong danh sách RecyclerView được đặt bằng cách sử dụng LinearLayoutManager ; cho đến nay, bạn đã sử dụng LinearLayoutManager bố trí từng mục trong danh sách cuộn dọc hoặc ngang. GridLayoutManager là một trình quản lý bố cục khác hiển thị các mục trong lưới, thay vì danh sách.

Khi bạn tạo GridLayoutManager , bạn cung cấp hai tham số: ngữ cảnh ứng dụng và một số nguyên đại diện cho số cột. Bạn có thể thay đổi số cột theo chương trình, điều này mang lại cho bạn sự linh hoạt trong việc thiết kế bố cục thích ứng. Trong trường hợp này, số lượng cột số nguyên phải là 1 theo hướng dọc (cột đơn) và 2 khi ở chế độ ngang. Lưu ý rằng khi số cột là 1, GridLayoutManager hoạt động tương tự như LinearLayoutManager

Thực tế này được xây dựng dựa trên ứng dụng MaterialMe từ thực tế trước đó.

1. Tiếp tục phát triển phiên bản ứng dụng MaterialMe của bạn hoặc tải xuống MaterialMe . Nếu bạn quyết định tạo một bản sao của dự án MaterialMe để giữ nguyên phiên bản từ thực tế trước đó, hãy đổi tên phiên bản đã sao chép MaterialMe-Resource .

2. Tạo một tệp tài nguyên mới có tên là integers.xml . Để thực hiện việc này, hãy mở thư mục res trong ngăn Project > Android, nhấp chuột phải (hoặc Control khi nhấp )

vào thư mục giá trị và chọn Tệp tài nguyên Giá trị > mới .

1. Đặt tên cho tệp integers.xml và nhấp vào OK .
2. Tạo một hằng số nguyên giữa các <resources> thẻ được gọi là grid\_column\_count và đặt nó bằng 1:

```
<integer name="grid_column_count">1</integer>
```

3. Tạo một tệp tài nguyên giá trị khác, một lần nữa được gọi là integers.xml; tuy nhiên, tên sẽ được sửa đổi khi bạn thêm bộ hạn định tài nguyên từ ngăn Bộ hạn định có sẵn. Bộ hạn định tài nguyên được sử dụng để gắn nhãn cấu hình tài nguyên cho các tình huống khác nhau.

4. Chọn Định hướng trong ngăn Trình hạn định có sẵn và nhấn biểu tượng >> ở giữa hộp thoại để gán bộ hạn định này.

5. Thay đổi menu Hướng màn hình thành Phong cảnh và để ý cách tên thư mục values-land xuất hiện. Đây là bản chất của bộ hạn định tài nguyên: tên thư mục cho Android biết khi nào nên sử dụng tệp bô cục cụ thể đó. Trong trường hợp này, đó là khi điện thoại được xoay sang chế độ ngang.

6. Nhấp chuột OK để tạo tệp bô cục mới.

7. Sao chép hằng số nguyên bạn đã tạo vào tệp tài nguyên mới này, nhưng thay đổi giá trị thành 2.

Bây giờ bạn sẽ có hai tệp integers.xml riêng lẻ được nhóm vào một thư mục integers.xml trong ngăn Project > Android. Tệp thứ hai được gắn nhãn với bộ hạn định bạn đã chọn, trong trường hợp này là land. Từ hạn định xuất hiện trong ngoặc đơn: integers.xml (đất) .

## 1.2 Sửa đổi MainActivity

1. Mở MainActivity và thêm mã vào onCreate() để lấy số nguyên từ tệp tài nguyên integers.xml:

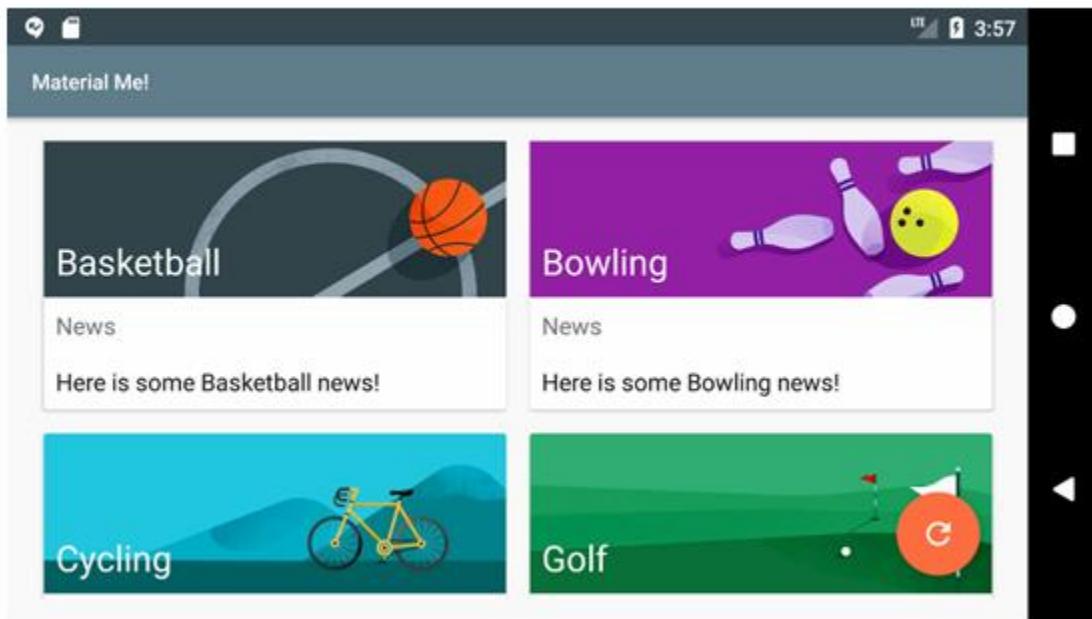
```
int gridColumnCount =
 getResources().getInteger(R.integer.grid_column_count);
```

Thời gian chạy Android sẽ đảm nhận việc quyết định sử dụng tệp integers.xml nào, tùy thuộc vào trạng thái của thiết bị.

2. Thay đổi LinearLayoutManager cho RecyclerView thành GridLayoutManager , truyền ngữ cảnh và số nguyên mới tạo:

```
mRecyclerView.setLayoutManager(new
 GridLayoutManager(this, gridColumnCount));
```

3. Chạy ứng dụng và xoay thiết bị. Số cột tự động thay đổi theo hướng của thiết bị.



Khi sử dụng ứng dụng ở chế độ ngang, bạn sẽ nhận thấy rằng chức năng vuốt để loại bỏ không còn trực quan nữa, vì các mục hiện nằm trong lối chú không phải là một cột duy nhất. Trong các bước tiếp theo, bạn sẽ tắt hành động vuốt nếu có nhiều cột.

4. Sử dụng biến gridColumnCount để tắt hành động vuốt (đặt swipeDirs thành không) khi có nhiều cột:

```
int swipeDirs;
if(gridColumnCount > 1) {
 swipeDirs = 0;
} else {
 swipeDirs = ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT;
}
```

5. Sử dụng swipeDirs thay cho các đối số hướng vuốt (ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT ) cho ItemTouchHelper.SimpleCallback()

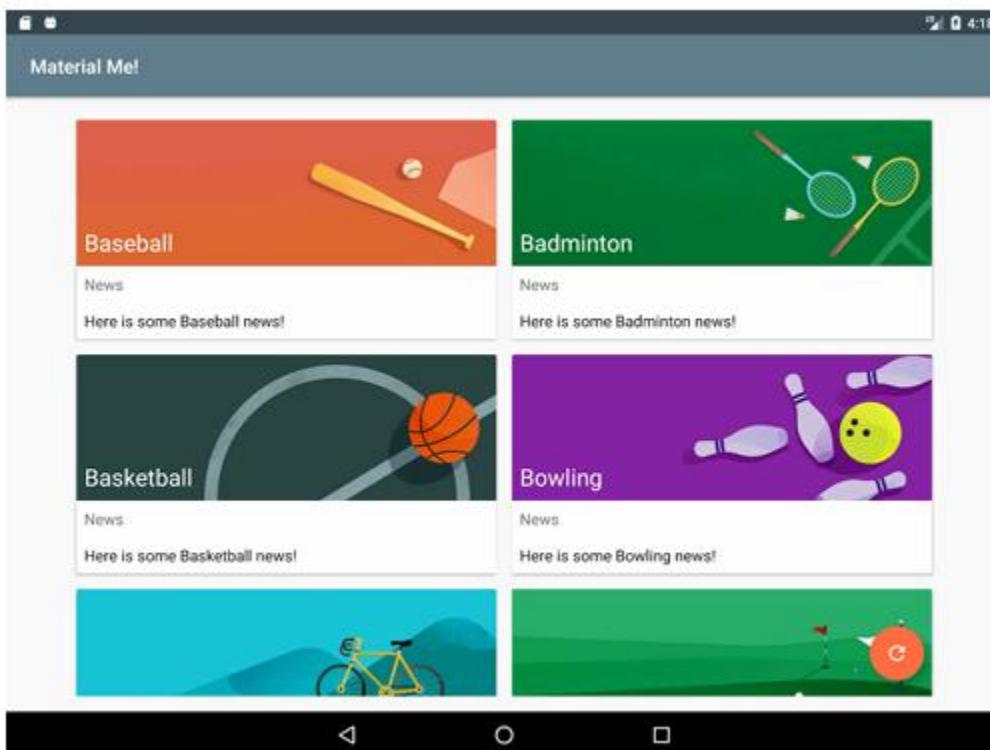
```
ItemTouchHelper helper = new ItemTouchHelper(new
 ItemTouchHelper.SimpleCallback(ItemTouchHelper.LEFT |
 ItemTouchHelper.RIGHT |
 ItemTouchHelper.DOWN | ItemTouchHelper.UP,
 swipeDirs) {
```

6. Chạy ứng dụng và xoay thiết bị. Ở hướng ngang (ngang), người dùng không thể vuốt để xóa thẻ nữa.

## Nhiệm vụ 2: Hỗ trợ máy tính bảng

Mặc dù bạn đã sửa đổi ứng dụng để trông đẹp hơn ở chế độ ngang, nhưng chạy ứng dụng trên máy tính bảng có kích thước vật lý lớn hơn sẽ dẫn đến tất cả văn bản xuất hiện quá nhỏ. Ngoài ra, khi thiết bị ở hướng ngang, màn hình không được sử dụng hiệu quả; ba cột sẽ thích hợp hơn cho màn hình có kích thước máy tính bảng ở chế độ ngang.

Trong nhiệm vụ này, bạn sẽ thêm các bộ hạn định tài nguyên bổ sung để thay đổi giao diện của ứng dụng khi được sử dụng trên máy tính bảng.



### 2.1 Điều chỉnh bố cục cho máy tính bảng

Trong bước này, bạn tạo các bộ hạn định tài nguyên khác nhau để tối đa hóa việc sử dụng màn hình cho các thiết bị có kích thước máy tính bảng, tăng số cột lên 2 cho hướng dọc (dọc) và 3 cho hướng ngang (ngang).

# Chương 3: Làm việc trong nền

## Bài 1. Các tác vụ nền

### 1.1 AsyncTask

Đây là một tài liệu PDF chứa thông tin một lần về các bài học trong **Unit 3: Làm việc trong nền.**

Các bài học trong đơn vị này:

- **Bài học 7: Nhiệm vụ nền**
  - 7.1: **AsyncTask**
  - 7.2: **AsyncTask và AsyncTaskLoader**
  - 7.3: **Broadcast receivers**
- **Bài học 8: Kích hoạt, lên lịch và tối ưu hóa nhiệm vụ nền**
  - 8.1: **Thông báo (Notifications)**
  - 8.2: **Quản lý báo thức (Alarm Manager)**
  - 8.3: **JobScheduler**

### Bài học 7.1: AsyncTask

#### Giới thiệu

Một **thread** là một đường dẫn độc lập của chương trình đang chạy. Khi một chương trình Android được khởi chạy, hệ thống tạo ra một **main thread**, còn gọi là **UI thread**. Đây là cách mà ứng dụng của bạn tương tác với các thành phần từ công cụ giao diện người dùng Android.

Đôi khi, ứng dụng cần thực hiện các nhiệm vụ đòi hỏi tài nguyên lớn như tải tệp, thực hiện truy vấn cơ sở dữ liệu, phát media, hoặc tính toán phân tích phức tạp. Các công việc này có thể làm chậm lại **UI thread** khiến ứng dụng không phản hồi với các thao tác của người dùng hoặc không vẽ được trên màn hình. Điều này có thể làm người dùng cảm thấy khó chịu và xóa ứng dụng.

Để giữ cho trải nghiệm người dùng (UX) luôn mượt mà, framework Android cung cấp một lớp trợ giúp gọi là **AsyncTask**, giúp xử lý công việc ngoài UI

**thread**. Việc sử dụng **AsyncTask** để chuyển các tác vụ nặng vào một thread riêng giúp **UI thread** không bị chặn và vẫn phản hồi được.

Vì **thread** riêng biệt không đồng bộ với **thread** gọi, nên nó được gọi là **asynchronous thread**. **AsyncTask** cũng chứa một callback cho phép bạn hiển thị kết quả tính toán trở lại trên **UI thread**.

Trong bài thực hành này, bạn sẽ học cách thêm một nhiệm vụ nền vào ứng dụng Android của bạn bằng cách sử dụng **AsyncTask**.

### Bạn đã biết gì?

Bạn sẽ có khả năng:

- Tạo một **Activity**.
- Thêm một **TextView** vào layout của **Activity**.
- Lấy ID cho **TextView** và thiết lập nội dung của nó qua mã.
- Sử dụng các **Button** và chức năng **onClick** của chúng.

### Bạn sẽ học được gì?

- Cách thêm **AsyncTask** vào ứng dụng của bạn để chạy một tác vụ nền.
- Các nhược điểm khi sử dụng **AsyncTask** cho các nhiệm vụ nền.

### Những gì bạn sẽ làm:

- Tạo một ứng dụng đơn giản thực thi một tác vụ nền sử dụng **AsyncTask**.
- Chạy ứng dụng và xem điều gì xảy ra khi bạn xoay thiết bị.
- Cài đặt **activity instance state** để giữ trạng thái của một thông báo **TextView**.

### Tổng quan về ứng dụng:

Bạn sẽ xây dựng một ứng dụng có một **TextView** và một **Button**. Khi người dùng nhấn vào **Button**, ứng dụng sẽ "ngủ" trong một khoảng thời gian ngẫu nhiên, sau đó hiển thị một thông báo trong **TextView** khi nó thức dậy. Đây là hình ảnh của ứng dụng hoàn chỉnh:

### Nhiệm vụ 1: Thiết lập dự án SimpleAsyncTask

Giao diện người dùng của **SimpleAsyncTask** chứa một **Button** để khởi động **AsyncTask** và một **TextView** để hiển thị trạng thái của ứng dụng.

## 1.1 Tạo dự án và giao diện

7. Tạo một dự án mới có tên **SimpleAsyncTask** sử dụng mẫu **Empty Activity**. Chấp nhận các tùy chọn mặc định cho tất cả các cài đặt khác.
8. Mở tệp giao diện **activity\_main.xml**. Nhập vào tab **Text**.
9. Thêm thuộc tính `layout_margin` vào **ConstraintLayout** cấp cao nhất:

```
xml
CopyEdit
android:layout_margin="16dp"
```

10. Thêm hoặc chỉnh sửa các thuộc tính sau của **TextView** chứa dòng chữ "Hello World!" để có các giá trị này. Trích xuất chuỗi này vào tài nguyên.

Attribute	Value
<code>android:id</code>	"@+id/textView1
<code>android:text</code>	"I am ready to start work!"
<code>android:textSize</code>	"24sp"

11. Xóa các thuộc tính `app:layout_constraintRight_toRightOf` và `app:layout_constraintTop_toTopOf`.
12. Thêm một phần tử **Button** ngay dưới **TextView**, và gán cho nó các thuộc tính sau. Trích xuất văn bản của nút vào tài nguyên chuỗi.

Attribute	Value
android:id	"@+id/button"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:text	"Start Task"
android:layout_marginTop	"24dp"
android:onClick	"startTask"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/textView1"

8. Thuộc tính onClick của nút sẽ được làm nổi bật bằng màu vàng, vì phương thức startTask() chưa được triển khai trong **MainActivity**. Đặt con trỏ của bạn vào văn bản được làm nổi bật, nhấn Alt + Enter (hoặc Option + Enter trên Mac) và chọn **Create 'startTask(View)' in 'MainActivity'** để tạo phương thức khung trong **MainActivity**.

Mã giải pháp cho **activity\_main.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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"
 android:layout_margin="16dp"
 tools:context=".MainActivity">

 <TextView
 android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/ready_to_start"
 android:textSize="24sp"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"/>

 <Button
 android:id="@+id/button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginTop="24dp"
 android:onClick="startTask"
 android:text="@string/start_task"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/textView1"/>

</android.support.constraint.ConstraintLayout>
```

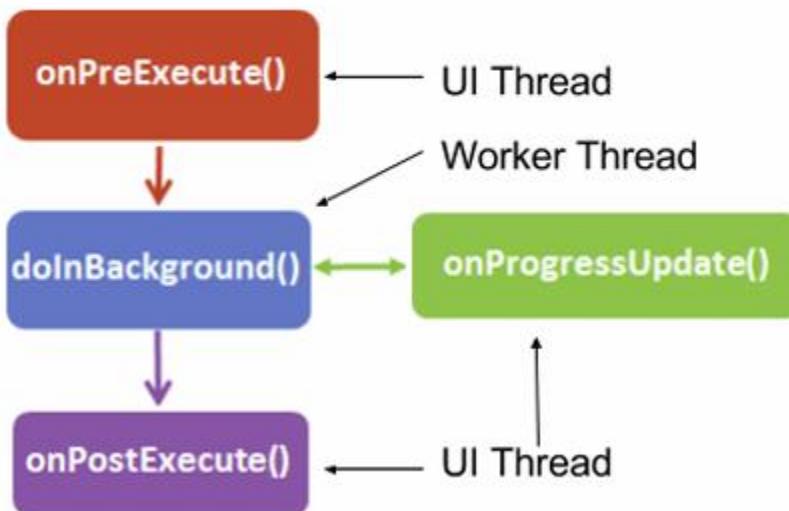
## Task 2: Tạo lớp con AsyncTask

AsyncTask là một lớp trừu tượng, điều này có nghĩa là bạn phải tạo một lớp con của nó để sử dụng. Trong ví dụ này, AsyncTask thực hiện một tác vụ nền rất đơn giản: nó ngủ trong một khoảng thời gian ngẫu nhiên. Trong một ứng dụng thực tế, tác vụ nền có thể thực hiện nhiều công việc khác nhau, từ truy vấn cơ sở dữ liệu, kết nối internet, cho đến tính toán nước đi tiếp theo trong trò chơi Go để đánh bại nhà vô địch Go hiện tại.

Một lớp con của AsyncTask có các phương thức sau để thực hiện công việc ngoài luồng chính:

- **onPreExecute()**: Phương thức này chạy trên luồng UI và được sử dụng để thiết lập tác vụ của bạn (chẳng hạn như hiển thị thanh tiến trình).

- **doInBackground()**: Đây là nơi bạn triển khai mã để thực hiện công việc sẽ được thực thi trên một luồng riêng biệt.
- **onProgressUpdate()**: Phương thức này được gọi trên luồng UI và dùng để cập nhật tiến trình trong giao diện người dùng (chẳng hạn như điền đầy thanh tiến trình).
- **onPostExecute()**: Lại trên luồng UI, phương thức này được sử dụng để cập nhật kết quả lên giao diện người dùng sau khi AsyncTask hoàn tất công việc.



**Lưu ý:** Một **background thread** (hay **worker thread**) là bất kỳ luồng nào không phải là luồng chính (UI thread).

Khi bạn tạo một lớp con của AsyncTask, bạn có thể cần cung cấp thông tin về công việc mà nó sẽ thực hiện, cách và việc có báo cáo tiến độ hay không, cũng như hình thức trả về kết quả. Khi bạn tạo một lớp con của AsyncTask, bạn có thể cấu hình nó bằng các tham số sau:

- **Params**: Kiểu dữ liệu của các tham số được gửi đến tác vụ khi thực thi phương thức ghi đè `doInBackground()`.
- **Progress**: Kiểu dữ liệu của các đơn vị tiến độ được xuất bản thông qua phương thức ghi đè `onProgressUpdate()`.
- **Result**: Kiểu dữ liệu của kết quả được cung cấp bởi phương thức ghi đè `onPostExecute()`.

Ví dụ, một lớp con của AsyncTask có tên là `MyAsyncTask` với khai báo lớp như sau có thể sử dụng các tham số:

- Một String làm tham số trong doInBackground(), để sử dụng trong một truy vấn, chẳng hạn.
- Một Integer cho onProgressUpdate(), để đại diện cho phần trăm công việc đã hoàn thành.
- Một Bitmap làm kết quả trong onPostExecute(), biểu thị kết quả truy vấn.

java

CopyEdit

```
public class MyAsyncTask extends AsyncTask<String, Integer, Bitmap> { }
```

Trong tác vụ này, bạn sẽ sử dụng một lớp con của AsyncTask để xác định công việc sẽ chạy trong một luồng khác ngoài luồng giao diện người dùng (UI thread).

## 2.1 Tạo lớp con của AsyncTask

Trong ứng dụng này, lớp con AsyncTask mà bạn tạo không yêu cầu tham số truy vấn hoặc xuất bản tiến độ của nó. Bạn sẽ chỉ sử dụng các phương thức doInBackground() và onPostExecute().

2. Tạo một lớp Java mới có tên là SimpleAsyncTask, mở rộng AsyncTask và sử dụng ba tham số kiểu chung.

- Sử dụng Void cho tham số, vì AsyncTask này không yêu cầu bất kỳ đầu vào nào.
- Sử dụng Void cho kiểu tiến độ, vì tiến độ không được xuất bản.
- Sử dụng một String làm kiểu kết quả, vì bạn sẽ cập nhật TextView bằng một chuỗi khi AsyncTask hoàn thành việc thực thi.

```
public class SimpleAsyncTask extends AsyncTask<Void, Void, String> { }
```

**Lưu ý:** Khai báo lớp sẽ bị gạch đỏ, vì phương thức bắt buộc doInBackground() chưa được triển khai.

3. Ở đầu lớp, định nghĩa một biến thành viên mTextView với kiểu WeakReference<TextView>:

java

CopyEdit

```
private WeakReference<TextView> mTextView;
```

4. Triển khai một constructor cho AsyncTask nhận một TextView làm tham số và tạo một tham chiếu yếu mới cho TextView đó:

java

CopyEdit

```
SimpleAsyncTask(TextView tv) {
 mTextView = new WeakReference<>(tv);
}
```

AsyncTask cần cập nhật TextView trong Activity sau khi hoàn tất việc "ngủ" (trong phương thức onPostExecute()). Vì vậy, constructor của lớp sẽ cần một tham chiếu đến TextView để được cập nhật.

### **Tham chiếu yếu (lớp WeakReference) để làm gì?**

Nếu bạn truyền một TextView vào constructor của AsyncTask và sau đó lưu nó trong một biến thành viên, tham chiếu đó đến TextView sẽ khiến Activity không bao giờ bị thu gom rác (garbage collected) và do đó gây rò rỉ bộ nhớ, ngay cả khi Activity bị hủy và tạo lại, chẳng hạn khi thay đổi cấu hình thiết bị. Đây được gọi là **tạo ngứ cảnh rò rỉ (leaky context)**, và Android Studio sẽ cảnh báo bạn nếu bạn có làm điều này.

Tham chiếu yếu ngăn chặn rò rỉ bộ nhớ bằng cách cho phép đối tượng được giữ bởi tham chiếu đó bị thu gom rác nếu cần thiết.

## **2.2 Triển khai doInBackground()**

Phương thức doInBackground() là bắt buộc đối với lớp con của AsyncTask.

2. Đặt con trỏ vào khai báo lớp được đánh dấu, nhấn **Alt + Enter** (hoặc **Option + Enter** trên Mac) và chọn **Implement methods**. Chọn doInBackground() và nhấn **OK**. Mẫu phương thức sau sẽ được thêm vào lớp:

java

CopyEdit

@Override

```
protected String doInBackground(Void... voids) {
```

```
 return null;
}
```

3. Thêm mã để tạo một số nguyên ngẫu nhiên từ 0 đến 10. Đây sẽ là số mili giây mà tác vụ sẽ tạm dừng. Số này không phải là thời gian dài để tạm dừng, vì vậy hãy nhân số đó với 200 để kéo dài thời gian:

java

CopyEdit

```
Random r = new Random();

int n = r.nextInt(11);

int s = n * 200;
```

4. Thêm một khối try/catch và đưa luồng vào trạng thái ngủ:

java

CopyEdit

```
try {
 Thread.sleep(s);
} catch (InterruptedException e) {
 e.printStackTrace();
}
```

5. Thay thế câu lệnh return hiện tại bằng câu lệnh trả về chuỗi "Awake at last after sleeping for xx milliseconds", trong đó xx là số mili giây mà ứng dụng đã tạm dừng:

java

CopyEdit

```
return "Awake at last after sleeping for " + s + " milliseconds!";
```

Phương thức doInBackground() hoàn chỉnh sẽ trông như sau:

java

CopyEdit

```

@Override
protected String doInBackground(Void... voids) {
 // Tạo một số ngẫu nhiên từ 0 đến 10
 Random r = new Random();
 int n = r.nextInt(11);
 // Đảm bảo tác vụ chạy đủ lâu để có thời gian xoay ngang điện thoại khi đang
 // chạy
 int s = n * 200;
 // Cho luồng tạm dừng trong khoảng thời gian ngẫu nhiên
 try {
 Thread.sleep(s);
 } catch (InterruptedException e) {
 e.printStackTrace();
 }
 // Trả về chuỗi kết quả
 return "Awake at last after sleeping for " + s + " milliseconds!";
}

```

### **2.3 Triển khai onPostExecute()**

Khi phương thức doInBackground() hoàn tất, giá trị trả về sẽ tự động được truyền tới hàm callback onPostExecute().

2. Triển khai onPostExecute() để nhận một đối số kiểu String và hiển thị chuỗi đó trong TextView:

java

CopyEdit

```

protected void onPostExecute(String result) {
 mTextView.get().setText(result);
}

```

```
}
```

Tham số String của phương thức này là tham số mà bạn đã định nghĩa ở tham số thứ ba trong khai báo lớp AsyncTask, và cũng là giá trị được trả về từ phương thức doInBackground().

Vì mTextView là một tham chiếu yếu, bạn cần sử dụng phương thức get() để trích xuất đối tượng TextView cơ bản, sau đó gọi setText() trên đối tượng đó.

### Lưu ý:

Bạn có thể cập nhật giao diện người dùng (UI) trong onPostExecute() vì phương thức này được chạy trên luồng chính (main thread). Tuy nhiên, bạn không thể cập nhật TextView bằng chuỗi mới trong phương thức doInBackground(), vì phương thức này được thực thi trên một luồng riêng biệt (background thread).

---

## Nhiệm vụ 3: Thực hiện các bước cuối cùng

### 3.1 Triển khai phương thức để bắt đầu AsyncTask

Ứng dụng của bạn hiện đã có một lớp AsyncTask thực hiện công việc trong nền (hoặc chỉ mô phỏng công việc bằng cách gọi sleep()). Nay, bạn có thể triển khai phương thức onClick cho nút "Start Task" để kích hoạt tác vụ chạy nền.

- Trong tệp MainActivity.java, thêm một biến thành viên để lưu trữ TextView:

```
java
```

```
CopyEdit
```

```
private TextView mTextView;
```

- Trong phương thức onCreate(), khởi tạo mTextView để liên kết với TextView trong giao diện:

```
java
```

```
CopyEdit
```

```
mTextView = findViewById(R.id.textView1);
```

- Trong phương thức startTask(), cập nhật TextView để hiển thị dòng chữ "Napping...". Trích xuất thông báo đó thành một tài nguyên chuỗi:

java

CopyEdit

```
mTextView.setText(R.string.napping);
```

5. Tạo một thê hiện của SimpleAsyncTask, truyền TextView (mTextView) vào constructor. Gọi execute() trên thê hiện SimpleAsyncTask:

java

CopyEdit

```
new SimpleAsyncTask(mTextView).execute();
```

#### Lưu ý:

Phương thức execute() là nơi bạn truyền các tham số, được phân tách bằng dấu phẩy, sau đó được hệ thống truyền vào doInBackground(). Vì AsyncTask này không có tham số, bạn để trống.

#### Mã trong MainActivity:

### 3.2 Triển khai onSaveInstanceState()

3. Chạy ứng dụng và nhấp vào nút **Start Task**. Ứng dụng sẽ "ngủ" trong bao lâu?
4. Nhấp lại vào nút **Start Task**, và khi ứng dụng đang "ngủ", xoay thiết bị. Nếu tác vụ nền hoàn thành trước khi bạn có thể xoay điện thoại, hãy thử lại.

**Lưu ý:** Bạn sẽ nhận thấy rằng khi thiết bị được xoay, TextView sẽ đặt lại về nội dung ban đầu, và AsyncTask dường như không thê cập nhật TextView.

Có một số điều đang xảy ra ở đây:

- Khi bạn muốn xoay thiết bị, hệ thống khởi động lại app, đang gọi onDestroy và sau đó onCreate(). AsyncTask sẽ tiếp tục chay thậm chí nếu activity bị hủy, nhưng nó cũng sẽ mất khả năng báo cáo lại cho giao diện người dùng activity. Nó sẽ không bao h cập nhập TextView, cái mà đã được chuyen vào, bởi vì TextView đó đã bị hủy
- Khi bị hủy, AsyncTask vẫn sẽ tiếp tục chạy tới khi hoàn thành trong nền, và tiêu tốn tài nguyên hệ thống

- Ngay cả khi không có AsyncTask, việc xoay thiết bị sẽ đặt lại toàn bộ thuộc tính UI về trạng thái mặc định. Đối với TextView là chuỗi mặc định, cái mà bạn thiết lập khi trong file layout

Vì lý do đó, 1 AsyncTask không phù hợp , có thể bị gián đoạn bởi sự biến mất của Activity. Trong trường hợp sd mà điều này là quan trọng bạn có thể sd 1 lớp nền gọi là AsyncTasLoader. Cái bạn sẽ học về ở lần thực hành sau. Để ngăn cản TextView từ thiết lập lại giá trị string ban đầu. Bạn cần lưu lại trạng thái của nó. Bạn sẽ thực thi onSaveInstanceState() để bảo vệ nội dung của TextView khi Activity bị hủy do thay đổi cấu hình, như xoay thiết bị

3.Ở đầu class thêm hằng số để làm khóa cho văn bản hiện tại trong Bundle trạng thái

4.Ghi đè onSaveInstanceState() method trong MainAcitivity để bảo tồn text trong TextView khi activity bị hủy:

```
@Override
protected void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 // Save the state of the TextView
```

5.trong onCreate, nhận giá trị của TextView từ trạng thái bundle khi activity đc khởi động lại

Nguồn code file MainActivity:

```
package android.example.com.simpleasynctask;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.TextView;

/**
 * The SimpleAsyncTask app contains a button that launches an AsyncTask
 * which sleeps in the asynchronous thread for a random amount of time.
 */
public class MainActivity extends AppCompatActivity {

 //Key for saving the state of the TextView
 private static final String TEXT_STATE = "currentText";

 // The TextView where we will show results
 private TextView mTextView = null;

 /**

```

```

 * Initializes the activity.
 * @param savedInstanceState The current state data
 */
@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 // Initialize mTextView
 mTextView = (TextView) findViewById(R.id.textView1);

 // Restore TextView if there is a savedInstanceState
 if(savedInstanceState!=null){
 mTextView.setText(savedInstanceState.getString(TEXT_STATE));
 }
}

/**
 * Handles the onClick for the "Start Task" button. Launches the
AsyncTask
 * which performs work off of the UI thread.
 *
 * @param view The view (Button) that was clicked.
 */
public void startTask (View view) {
 // Put a message in the text view
 mTextView.setText(R.string.mapping);

 // Start the AsyncTask.
 // The AsyncTask has a callback that will update the text view.
 new SimpleAsyncTask(mTextView).execute();
}

/**
 * Saves the contents of the TextView to restore on configuration
change.
 * @param outState The bundle in which the state of the activity is
saved
 * when it is spontaneously destroyed.
 */
@Override
protected void onSaveInstanceState(Bundle outState) {
 super.onSaveInstanceState(outState);
 // Save the state of the TextView
}

```

## Thử thách coding

Note: Tất cả các thử thách code là tùy chọn không bắt buộc cho các tiết học về sau

Thử thách: Lớp AsyncTask cung cấp các phương thức ghi đè hữu dụng; onProgressUpdate(), cái mà cho phép bạn cập nhật giao diện tổng khi AsyncTask đang chạy. Sử dụng phương thức này để úp dát giao diện của bạn với thời gian ngủ hiện tại. Xem tới tài liệu AsyncTask để biết cách OnProgressUpdate() được thực thi. Nhớ rằng trong định nghĩa lớp của

AsynTask, bạn cần xác minh kiểu dữ liệu để sử dụng trong phương thức onProgressUpdate

### Tóm tắt:

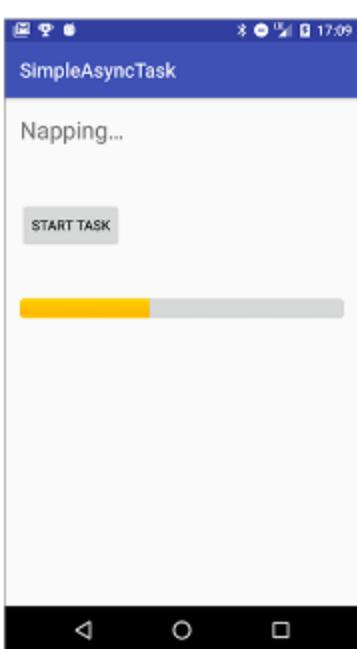
- AsyTask là 1 lớp trừu tượng giúp chuyển các tác vụ xử lý nặng sang một luồng riêng biệt (a separate thread.)
- AsynTAsk phải được kế thừa(subclass) từ lớp con để sử dụng
- Tránh làm các công việc tốn nhiều tài nguyên trên luồng giao diện, bởi ví nó làm giao diện chậm or không ổn định (erratic)
- Tất cả đoạn code không liên quan tới vẽ giao diện để được chuyển từ luồng UI sang luồng riêng biệt
- AsyncTask có 4 phương thức chính: onPreExecute(), doInBackground(), onPostExecute() and onProgressUpdate().
- Phương thức doInBackground() là phương thức duy nhất thường đc chạy trong luồng lm việc
- Các phương thức của lại của AsyncTask trong luồng UI ho phép tương tác với các thành phần giao diện.
- Khi xoay thiết bị, activity bị hủy và được tạo lại. Điều này ngăn kết nối UI tới luồng nền của AsyncTask. Cái mà sẽ tiếp tục chạy.

### HomeWork

#### Build và run 1 app

Mở ứng dụng **SimpleAsyncTask**. Thêm một **ProgressBar** hiển thị phần trăm thời gian ngủ đã hoàn thành. Thanh tiến trình sẽ đầy dần khi luồng AsyncTask ngủ, từ 0 đến 100 (phần trăm).

Gợi ý: chia thời gian ngủ thành nhiều phần



## Câu hỏi 1: ProgressBar

1. Phạm vi giá trị của **ProgressBar** được xác định bằng thuộc tính android:max trong XML hoặc bằng phương thức setMax(int value) trong Java/Kotlin.
2. Để thay đổi mức độ tiến trình được lấp đầy, sử dụng setProgress(int value).

---

## Câu hỏi 2: AsyncTask

Với AsyncTask được định nghĩa như sau:

Nộp app của bạn để chấm điểm

Hướng dẫn chán điểm

- Layout bao gồm Progress bar thiết lập các thuộc tính phù hợp, xác định phạm vi của giá trị
- AsyncTask chia tổng thời gian thành các phần và cập nhập thanh progressbar sau mỗi phần
- AsyncTask gọi phương thức phù hợp và thực thi call back phù hợp để cập nhập thanh trạng thái

- AsyncTask cần biết các view để cập nhật. Dựa vào AsyncTask để thực thi như 1 lớp nằm trong hoặc không, các view này truyền qua hàm khởi tạo của AsyncTask hoặc được định nghĩa như 1 biến thành viên trong Activity

## 1.2 AsyncTask and AsyncTaskLoader

Giới thiệu:

Trong bài thực hành này, bạn sẽ sử dụng **AsyncTask** để khởi chạy một tác vụ nền, lấy dữ liệu từ Internet thông qua một REST API đơn giản. Bạn sẽ:

- Sử dụng **Google APIs Explorer** để truy vấn **Books API**.
- Triển khai truy vấn này trong một luồng làm việc (**worker thread**) bằng **AsyncTask**.
- Hiển thị kết quả trên giao diện người dùng (**UI**).

Sau đó, bạn sẽ triển khai lại tác vụ nền tương tự bằng **AsyncTaskLoader**, một phương pháp hiệu quả hơn để cập nhật giao diện.

Cái bạn cần biết:

- Tạo activity
- Thêm textview
- Thực thi onclick đơn giản cho button
- Thực thi 1 AsyncTask và hiển thị kết quả
- Chuyển dữ liệu giữa các activity

Cái bạn sẽ học

Các sử dụng gg api explorer để tìm hiểu gg api và xem phản hồi json từ http requests

Cách sử dụng gg books api để nhặt dữ liệu trên internet và giữ UI mượt và phản hồi tốt. Bạn không cần học Books api. App của bạn sẽ chỉ sử dụng các hàm tìm kiếm đơn giản

Cách để phân tích kết quả json từ api query

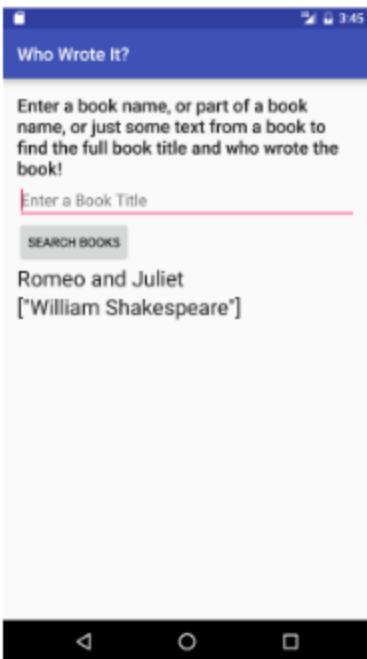
Cách thực thi AsyncTaskLoader cái mà bảo tồn dữ liệu của bạn khi cấu hình thay đổi

Cách update UI sử dụng loader Callback

## App overview

Bạn sẽ xây dựng 1 app gồm 1 edittext và 1 button

- Người dùng nhập tên sách và nhấn button
- Button thực thi như 1 AsyncTask, yêu cầu google book api tìm kiếm tác giả, tiêu đề của sách mà người dùng tìm kiếm
- Kết quả nhận được và hiển thị lên textView
- Khi app hoạt động, bạn thay đổi app để sử dụng AsynCTaskLoader thay vì AsyncTask



### Task1: khám phá google book api

Trong thực hành này bạn sử dụng api từ google book để tìm kiếm thông tin về sách ví dụ như tác giả và tiêu đề, api cung cấp chương trình truy cập tới dịch vụ gg book search sử dụng Rest apis. Cái này cũng giống với dịch vụ bên dưới màn hình khi bạn thực thi 1 tìm kiếm trên gg books. Bạn có thể sử dụng gg api Explore và gg book search trong trình duyệt để thay đổi

#### Gửi yêu cầu đến Books API

Truy cập Google APIs Explorer tại <https://developers.google.com/apis-explorer/>.

Nhập vào Services trong thanh điều hướng bên trái, sau đó chọn Books API.

Tìm books.volumes.list và nhập vào tên hàm đó. (Bạn có thể dùng Control+F trên Windows hoặc Command+F trên Mac để tìm kiếm trên trang).

Trang hiển thị danh sách các tham số của Books API để tìm kiếm sách.

Trong trường q, nhập tên sách hoặc một phần tên sách (ví dụ: "Romeo"). (q là tham số bắt buộc).

Trong trường maxResults, nhập 10 để giới hạn kết quả ở 10 cuốn sách hàng đầu.

Trong trường printType, nhập books để chỉ lấy sách có bản in.

Đảm bảo công tắc "Authorize requests using OAuth 2.0" ở đầu biểu mẫu đang tắt.

Nhập vào "Execute without OAuth" ở cuối biểu mẫu.

Cuộn xuống để xem yêu cầu HTTP và phản hồi HTTP.

Yêu cầu HTTP là một URI (Uniform Resource Identifier)—một chuỗi xác định tài nguyên. URL là một loại URI đặc biệt dùng để xác định tài nguyên trên web.

Đối với Books API, yêu cầu là một URL, trong đó các tham số tìm kiếm bạn đã nhập sẽ xuất hiện sau dấu ?.

Lưu ý: Trường API key thường xuất hiện ở cuối URL. Vì lý do bảo mật, khi truy cập một public API, bạn cần có API key và bao gồm nó trong yêu cầu. Tuy nhiên, Books API không yêu cầu API key, vì vậy bạn có thể bỏ qua phần này trong URI khi sử dụng trong ứng dụng của mình.

## Phân tích phản hồi của book api

Phản hồi từ truy vấn nằm ở cuối trang. Phản hồi sử dụng định dạng json- định dạng thông thường cho phản hồi truy vấn api. Trong trang apis Explorer, code json hiển thị dễ đọc. Trong app, phản hồi json sẽ được trả về dưới dạng 1 chuỗi duy nhất, bạn sẽ cần phân tích chuỗi để trích xuất thông tin cần.

Phản hồi bao gồm cặp name/values cái đc chia cắt bởi dấu phẩy. Ví dụ: kind#dd

Tìm giá trị của "title":

Giá trị của "title" cho một cuốn sách là một chuỗi duy nhất. Ví dụ:

Khi tìm kiếm sách, API trả về danh sách tất cả các cuốn sách có chứa chuỗi tìm kiếm, với mỗi cuốn sách được biểu diễn bằng một đối tượng JSON riêng. Trong bài thực hành này, bạn chỉ cần trích xuất tiêu đề (title) và tác giả (authors) của mục đầu tiên trong phản hồi JSON.

Task 2: Tạo app Who wrote it

2.1 tạo dự án và giao diện người dùng

1. Tạo 1 dự án mới, sử dụng Empty Activity template, chấp nhận cấp lựa chọn mặc định
2. mở activity\_main.xml. click vào Text tab
3. thêm layout\_margin attribute to top-level constraintLayout
4. xóa textview tồn tại
5. thêm các thành phần và thuộc tính của giao diện. Chú ý nguồn chuỗi xuất hiện màu đỏ. Định nghĩa chúng trong bước tiếp theo

Will appear in red, you define those in the next step.

View	Attributes	Values
TextView	android:layout_width android:layout_height android:id android:text android:textAppearance  app:layout_constraintStart_toStartOf app:layout_constraintTop_toTopOf	"match_parent" "wrap_content" "@+id/instructions" "@string/instructions" "@style/TextAppearance. AppCompat.Title" "parent" "parent"

EditText	android:layout_width android:layout_height android:id android:layout_marginTop android:inputType android:hint app:layout_constraintEnd_toEndOf app:layout_constraintStart_toStartOf app:layout_constraintTop_toBottomOf	"match_parent" "wrap_content" "@+id/bookInput" "8dp" "text" "@string/input_hint" "parent" "parent" "@+id/instructions"
Button	android:layout_width android:layout_height android:id android:layout_marginTop android:text android:onClick app:layout_constraintStart_toStartOf app:layout_constraintTop_toBottomOf	"wrap_content" "wrap_content" "@+id/searchButton" "8dp" "@string/button_text" "searchBooks" "parent" "@+id/bookInput"
TextView	android:layout_width android:layout_height android:id android:layout_marginTop android:textAppearance  app:layout_constraintStart_toStartOf app:layout_constraintTop_toBottomOf	"wrap_content" "wrap_content" "@+id/titleText" "16dp" "@style/TextAppearance. AppCompat.Headline" "parent" "@+id/searchButton"
TextView	android:layout_width android:layout_height android:id android:layout_marginTop android:textAppearance  app:layout_constraintStart_toStartOf app:layout_constraintTop_toBottomOf	"wrap_content" "wrap_content" "@+id/authorText" "8dp" "@style/TextAppearance. AppCompat.Headline" "parent" "@+id/titleText"

## 2.2 lấy input từ người dùng

Để truy vấn book api, bạn cần lấy đầu vào từ edittext mà ng dùng nhập

1.Trong MainActivity.java tạo 1 biến thành viên ánh xạ tới Edittext

2. trong searBook(), lấy text từ Edittext. Đổi text sang string và gán vào 1 biến

### 2.3 Tạo 1 lớp AsyncTask chống

Bây giờ bạn đã sẵn sàng kết nối Internet và sử dụng **Books API**. Trong nhiệm vụ này, bạn sẽ tạo một lớp con **AsyncTask** mới có tên **FetchBook** để xử lý kết nối mạng.

Kết nối mạng có thể chậm, khiến ứng dụng hoạt động không ổn định hoặc chậm. Vì vậy, không nên thực hiện kết nối mạng trên **UI thread**. Nếu bạn cố gắng kết nối mạng trên **UI thread**, Android runtime có thể phát sinh ngoại lệ **NetworkOnMainThreadException** để cảnh báo rằng đây là một cách làm không đúng.

Thay vào đó, hãy sử dụng một lớp con của **AsyncTask** để thực hiện kết nối mạng. Một **AsyncTask** yêu cầu ba tham số kiểu dữ liệu:

1.tạo lớp java trong app gọi là Fetchbook, kế thừaAsyncTask. Kiểu đối số <T> cho lớp này sẽ là <String, void , String>

```
public class FetchBook extends AsyncTask<String, Void, String>{
}
```

2, thực thi phương thức được yêu cầu, doInBackground(). Để làm điều này, đặt con chỏ tới đoạn text gạch đỏ, nhấn alt+enter và click o

Đảm bảo đối số và kiểu trả về là chính xác

3, chọn code> override methods, hoặc nhấn ctrl+0. Chọn onPostExecute() để chèn các định nghĩa phương thức vào trong lớp. onPostExecute() lấy 1 chuỗi như 1 đối số và trả về void

4, Để hiển thị kết quả trong các TextView trong MainActivity, bạn cần có quyền truy cập vào các TextView đó bên trong AsyncTask. Hãy tạo biến thành viên WeakReference để tham chiếu đến hai TextView dùng để hiển thị kết quả.

5, tạo khôi tạo cho FetchBook

```
FetchBook(TextView titleText, TextView authorText) {
 this.mTitleText = new WeakReference<>(titleText);
 this.mAuthorText = new WeakReference<>(authorText)
}
```

Code của FetchBook:

```
public class FetchBook extends AsyncTask<String,Void,String> {
 private WeakReference<TextView> mTitleText;
 private WeakReference<TextView> mAuthorText;
 public FetchBook(TextView mTitleText, TextView mAuthorText) {
```

```

this.mTitleText = new WeakReference<>(titleText);
this.mAuthorText = new WeakReference<>(authorText);
}

@Override
protected String doInBackground(String... strings) {
 return null;
}

@Override
protected void onPostExecute(String s) {
 super.onPostExecute(s);
}
}

```

#### 2.4 tạo lớp NetworkUtils và xay dung Uri

Bạn cần mở kêt snoois internet và truy vấn books api. Bởi vì bạn ssesx có thể dùng cám hàm này lại, bạn có thể muôn tạo lớp tiện chứ chúc năng và phát triển 1 lớp con hữu ích để dễ dàng tái sử dụng

rong nhiệm vụ này, bạn sẽ viết mã để kết nối Internet trong một lớp trợ giúp (helper class) có tên NetworkUtils.

Các bước thực hiện:

1. Tạo một lớp Java mới trong ứng dụng của bạn có tên NetworkUtils.
  - o Lớp NetworkUtils không kế thừa từ bất kỳ lớp nào khác.
2. Tạo một biến LOG\_TAG để ghi log, đặt tên theo tên của lớp:

```

private static final String LOG_TAG =
 NetworkUtils.class.getSimpleName()

```

3. Tạo một hàm tĩnh tên getBookInfo(). getBookInfo() lấy các chuỗi và trả về jsonString từ api, bạn sẽ được kiểm tra sớm
4. Tạo các biến local dưới đây trong getBookinfor():
 

```

HttpURLConnection urlConnection = null;
BufferedReader reader = null;
String bookJSONString = null;

```
5. Cuối cùng, trả về giá trị bookJsonString
 

```

Return bookJsonString;

```
6. Thêm cấu trúc try catch finnally trong getBookInfo, sau biến và tược return:

Trogn khói này, bạn sẽ build 1 UI và giải quyết các truy vấn. Trong blok, bạn sẽ giải quyết các vấn đề với request. Trong block cuối, bạn sẽ đóng kết nối sau khi kết thúc nhân json data:

7. Tạo các hằng số dưới đây ở đầu class NetWork Utils:

use URL for Books API.

```
private static final String BOOK_BASE_URL =
 "https://www.googleapis.com/books/v1/volumes?";
 // Parameter for the search string.
 private static final String QUERY_PARAM = "q";
 // Parameter that limits search results.
 private static final String MAX_RESULTS = "maxResults";
 // Parameter to filter by print type.
 private static final String PRINT_TYPE = "printType"
```

Khi bạn thấy yêu cầu từ trang books api, tất cả các request bđ giống với URI. Để xác minh kiểu của nguồn, thêm truy vấn đối số tới các URI base. Nó thường thực hành để chia tất cả cách truy vấn đối số thành hằng và tích hợp chúng sử dụng Uri.Builder vì vậy bạn có thể sử dụng từ uri khác nhau. Lớp uri là 1 phương thức tiện dụng. Uri.buildUpon(), trả về 1 uri.builder bạn có thể sử dụng

Trong app này, bạn giới hạn số và iêu dữ liệu trả về để tăng tốc độ truy vấn. Để giảm hạn truy vấn. Bạn sẽ chỉ xem sách cái được in

- 8 . Trong getBookInfor(mothed), xay dựng yêu cầu uri trong try{ }

## 2.5 tạo request

Api request này sử dụng HttpURLConnection lớp tích hợp với InputStream BufferedReader và StringBuffer để có phản hồi json từ web. Nếu ở mỗi mỗi quá trình bị fail và inputStream hoặc StringBuffer bị trống, request sẽ trả về null, chú ý rằng truy vấn đã thất bại

1, trong khối try{} của getbookinfo, mở url kết nối và làm request:

```
urlConnection = (HttpURLConnection) requestURL.openConnection();
```

```
urlConnection.setRequestMethod("GET");
```

```
urlConnection.connect();
```

1, trong try{} .thiết lập trả về từ kết nối sử dụng 1 InputStream, 1 BufferedReader và StringBuilder

Get the InputStream.

```
InputStream inputStream = urlConnection.getInputStream();
```

```
// Create a buffered reader from that input stream.
```

```
reader = new BufferedReader(new InputStreamReader(inputStream));
```

```
// Use a StringBuilder to hold the incoming response.
```

```
StringBuilder builder = new StringBuilder()
```

```

1, đọc input từng dòng vào string trong khi nó vẫn là input
String line;
while ((line = reader.readLine()) != null) {
 builder.append(line);
 // Since it's JSON, adding a newline isn't necessary (it won't
 // affect parsing) but it does make debugging a *lot* easier
 // if you print out the completed buffer for debugging.
 builder.append("\n");
}

```

**Note:** trong khi vòng lặp thêm line tiếp để tạo string trong 2 bước: 1 bước cho dòng của dữ liệu phản hồi, 1 bước thêm kí hiệu xuống dòng\n

**2,** ở đầu vào cuối, kiểm tra chuỗi để xem liệu có đang tồn tại phản hồi nội dung. Trả về null nếu phản hồi là empty:

3, đổi StringBuilder thành kiểu string và lưu trong bookJsonString

4, trong khối finally{}, cả kết nối và BufferedReader:

```

finally {
 if (urlConnection != null) {
 urlConnection.disconnect();
 }
 if (reader != null) {
 try {
 reader.close();
 } catch (IOException e) {
 e.printStackTrace();
 }
 }
}

```

**Note :** Mỗi khi kết nối thất bại vì bất kỳ lý do nào, đoạn mã này sẽ trả về null. Điều này có nghĩa là phương thức onPostExecute() trong lớp FetchBook cần kiểm tra tham số đầu vào của nó để xem có phải là chuỗi null không và thông báo cho người dùng về lỗi.

## 2.6: Thêm quyền truy cập internet

1. trong AndroidManifest.xml

2. Thêm như sau:

```
<uses permission android:name="android.permission.INTERNET" />
```

```
<uses permission
```

```
android:name="android.permission.ACCESS_NETWORK_STATE" /
```

## 2.7: Phân tích json

Bây giờ bạn đã có phản hồi JSON từ truy vấn của mình, bạn phải phân tích cú pháp kết quả để trích xuất thông tin cần hiển thị trên giao diện người dùng của ứng dụng. Java có các lớp trong API cốt lõi giúp bạn phân tích cú pháp và xử lý dữ liệu kiểu JSON.

Quá trình này, cùng với việc cập nhật giao diện người dùng, diễn ra trong phương thức onPostExecute() của lớp FetchBook.

Có khả năng phương thức doInBackground() sẽ không trả về chuỗi JSON như mong đợi. Ví dụ: khối try/catch có thể thất bại và ném ra một ngoại lệ, mạng có thể bị ngắt kết nối do hết thời gian chờ, hoặc có thể xảy ra các lỗi khác chưa được xử lý.

Trong những trường hợp đó, quá trình phân tích cú pháp JSON sẽ thất bại và ném ra một ngoại lệ. Để xử lý trường hợp này, hãy thực hiện phân tích cú pháp JSON trong một khối try/catch và xử lý tình huống khi dữ liệu không đúng hoặc không đầy đủ được trả về.

1, trong lớp fetchbook, trong onPostExecute, them try/catch

```
try {
 //...
} catch (JSONException e) {
 e.printStackTrace();}
```

2, trong try block, sử dụng các lớp JSONObject và JSONArray để có được mảng json các phần tử từ chuỗi kết quả

```
JSONObject jsonObject = new JSONObject(s);
JSONArray itemsArray = jsonObject.getJSONArray("items")
```

3, Khởi tạo biến sử dụng trong phân tích lặp

```
int i = 0;
String title = null;
String authors = null;
```

4, Lặp qua mảng itemsArray, kiểm tra từng cuốn sách để tìm thông tin tiêu đề (title) và tác giả (author). Trong mỗi vòng lặp, kiểm tra xem cả tiêu đề và tác giả có tồn tại hay không. Nếu tìm thấy cả hai, thoát khỏi vòng lặp ngay lập tức. Cách tiếp cận này đảm bảo chỉ những mục có đầy đủ tiêu đề và tác giả mới được hiển thị.

```
while (i < itemsArray.length() &&
(authors == null && title == null)) {
 // Get the current item information.

 JSONObject book = itemsArray.getJSONObject(i);

 JSONObject volumeInfo = book.getJSONObject("volumeInfo");

 // Try to get the author and title from the current item,
```

```

// catch if either field is empty and move on.

try {
 title = volumeInfo.getString("title");
 authors = volumeInfo.getString("authors");
} catch (Exception e) {
 e.printStackTrace();
}

// Move to the next item.

i++;
}

}

```

**Note:** vòng lặp kết thúc khi tìm thấy kết quả phù hợp đầu tiên của phản hồi. Nhiều phản hồi có lẽ có sẵn nhưng app này chỉ hiển thị cái đầu tiên

5, Nếu kết quả phản hồi được tìm thấy, cập nhập Ui vs phản hồi đó. Bởi vì tượng trưng cho đối tượng textview là weakReference, bạn phải gán lại chúng sử dụng get()

```

if (title != null && authors != null) {

 mTitleText.get().setText(title);
 mAuthorText.get().setText(authors);

}

6, nếu vòng lặp dùng và kết quả không có gì ta set title của textview là nōeults
else {

 mTitleText.get().setText(R.string.no_results);
 mAuthorText.get().setText("");
}

```

7, trong khói catch{}, in lỗi. Set tittle textbiew là no result

8, them no result vào resource trong strings.xml

File nguồn code

```
@Override
protected void onPostExecute(String s) {
 super.onPostExecute(s);
 try {
 // Convert the response into a JSON object.
 JSONObject jsonObject = new JSONObject(s);
 // Get the JSONArray of book items.
 JSONArray itemsArray = jsonObject.getJSONArray("items");
 // Initialize iterator and results fields.
 int i = 0;
 String title = null;
 String authors = null;
 // Look for results in the items array, exiting
 // when both the title and author
 // are found or when all items have been checked.
 while (i < itemsArray.length() &&
 (authors == null && title == null)) {
 // Get the current item information.
 JSONObject book = itemsArray.getJSONObject(i);
 JSONObject volumeInfo = book.getJSONObject("volumeInfo");
 // Try to get the author and title from the current item,
 // catch if either field is empty and move on.
 try {
 title = volumeInfo.getString("title");
 authors = volumeInfo.getString("authors");
 } catch (Exception e) {
 e.printStackTrace();
 }
 // Move to the next item.
```

```

 i++;
}

// If both are found, display the result.

if (title != null && authors != null) {

 mTitleText.get().setText(title);

 mAuthorText.get().setText(authors);

} else {

 // If none are found, update the UI to

 // show failed results.

 mTitleText.get().setText(R.string.no_results);

 mAuthorText.get().setText("");

}

} catch (Exception e) {

 // If onPostExecute does not receive a proper JSON string,
 // update the UI to show failed results.

 mTitleText.get().setText(R.string.no_results);

 mAuthorText.get().setText("");

}
}

```

### **Task 3: thực hành thực thi UI tốt nhất**

**Bây h, bạn có 1 app chức năng, sử dụng book api để thi hành tìm kiếm sách, tuy nhiên 1 vài điều khoongf như mon đợi:**

- Khi người dùng nhấn Search Books, bàn phím không biến mất, khiến người dùng không biết rằng truy vấn đang được thực hiện.
- Nếu không có kết nối mạng hoặc ô tìm kiếm trống, ứng dụng vẫn có găng truy vấn API và thất bại mà không cập nhật đúng cách trên giao diện người dùng (UI).
- Nếu xoay màn hình trong khi truy vấn, AsyncTask sẽ bị ngắt kết nối khỏi Activity, khiến nó không thể cập nhật kết quả lên UI.

Bạn sử 2 bấn để đầu tiên trong bài này, và vẫn để cuối trong task 4

### 3.1 ấn bàn phím và cập nhập textview

Trải nghiệm tìm kiếm người dùng chưa trực quan (intuitive). Khi người dùng ấn button, bàn phím vẫn hiển thị, người dùng ko có cách nào để cho thấy truy vấn đang thực hiện.

Giải pháp là để ấn phím và cập nhập 1 kết quả textview là đọc “Loading”. Trong khi truy vấn được thi hành

1, Trong MainActivity, them code searchbook(), sau đinhj nghĩa queryString. Code ấn keyboard khi người dùng ấn button

```
InputMethodManager inputManager = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);

if (inputManager != null) {
 inputManager.hideSoftInputFromWindow(view.getWindowToken(),
 InputMethodManager.HIDE_NOT_ALWAYS);}
```

1, Chỉ ngay dưới lời gọi để thực thi FetchBook, them code để thay đổi tiêu đề TextView để tải tin thoại và clear textview tác giả

2, thêm cập nhập nguồn tới String.xml

```
<string name="loading">Loading...</string>
```

### 3.2 quản lý trạng thái mạng và trường hợp trường tìm kiếm trả về rỗng

Bất kì khi nào app sử dụng mạng, nó cần xử lý kết nối mạng không khả dụng. Trước khi cố gắng kết nối mạng, ứng dụng của bạn nên kiểm tra trạng thái kết nối mạng. Ngoài ra, ứng dụng không nên cố gắng truy vấn Books API nếu người dùng chưa nhập chuỗi tìm kiếm.

1, Trong phương thức searchBooks(), sử dụng các lớp ConnectivityManager và NetworkInfo để kiểm tra kết nối mạng.

```
ConnectivityManager connMgr = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo networkInfo = null;

if (connMgr != null) {
 networkInfo = connMgr.getActiveNetworkInfo();

}
```

2, Thêm hêm một điều kiện kiểm tra xung quanh lời gọi thực thi tác vụ FetchBook và cập nhật TextView để đảm bảo rằng: kết nối tồn tại, mạng đang được kết nối, người dùng đã nhập chuỗi tìm kiếm

3, thêm else để kiểm tra. Trong khôi else, cù nhập ui với 1 lỗi no\_search\_term nếu không có khái niệm nào để tìm kiếm và lỗi no\_network

```
} else {
 if (queryString.length() == 0) {
 mAuthorText.setText("");
 mTitleText.setText(R.string.no_search_term);
 } else {
 mAuthorText.setText("");
 mTitleText.setText(R.string.no_network);
 }
}
```

4, them no\_search\_team vaf no\_network vào file strings.xml

```
<string name="no_search_term">Please enter a search term</string>
<string name="no_network">Please check your network connection and try
again.</string>
```

#### Task 4: Chuyển sang AsyncTaskLoader

Khi bạn sử dụng AsyncTask để thực hiện các thao tác trong nền, luồng nền không thể cập nhật giao diện người dùng (UI) nếu có sự thay đổi cấu hình xảy ra trong khi tác vụ nền đang chạy. Để giải quyết vấn đề này, sử dụng lớp AsyncTaskLoader thay vì AsyncTask.

AsyncTaskLoader tải dữ liệu trong nền và tự động liên kết lại các tác vụ nền với Activity, ngay cả khi xảy ra thay đổi cấu hình. Khi sử dụng AsyncTaskLoader, nếu xoay thiết bị trong khi tác vụ đang chạy, kết quả vẫn sẽ được hiển thị chính xác trong Activity.

Vậy tại sao vẫn dùng AsyncTask nếu AsyncTaskLoader hữu ích hơn?

Câu trả lời là tùy vào tình huống : Nếu tác vụ nền có khả năng hoàn thành trước khi có thay đổi cấu hình. Nếu không bắt buộc phải cập nhật UI sau khi hoàn thành

thì AsyncTask có thể đủ dùng.Thực tế, AsyncTaskLoader cũng sử dụng AsyncTask bên trong để hoạt động.

4.1, Tạo 1 lớp AsyncTaskLoader:

Để bảo toàn kết quả của thực hành trước, hãy sao chép dự án WhoWroteIt.  
Đổi tên dự án đã sao chép thành WhoWroteItLoader.

1, Tạo một lớp mới có tên BookLoader.

2, Kế thừa từ AsyncTaskLoader, với kiểu tham số hóa là <String>

```
import android.support.v4.content.AsyncTaskLoader;

public class BookLoader extends AsyncTaskLoader<String> {
}
```

Đảm bảo import lớp AsyncTaskLoad từ v4 Support Libarry

3, Triển khai phương thức bắt buộc loadInBackground. Lưu ý rằng phương thức này **tương tự với** phương thức doInBackground() ban đầu của AsyncTask.

```
@Nullable
```

```
@Override
```

```
public String loadInBackground() {
 return null;
}
```

4, tạo hàm khởi tạo từ lớp BookLoader.

```
public BookLoader(@NonNull Context context) {
 super(context);}
```

4.2, triển khai phương thức bắt buộc

1, Nhấn ctrl + O để mở menu phương thức ghi đè và chọn onStartLoading. Hệ thống gọi phương thức này khi bạn bắt đầu loader

```
Override
```

```
protected void onStartLoading() {
 super.onStartLoading();}
```

2, Bên trong phương thức onStartLoading(), hãy gọi forceLoad() để khởi chạy phương thức loadInBackground(). Bộ nạp dữ liệu sẽ không bắt đầu tải dữ liệu cho đến khi bạn gọi phương thức forceLoad().

```
@Override
```

```
protected void onStartLoading() {
 super.onStartLoading();}
```

```
}
```

3, Tạo một biến thành viên có tên mQueryString để lưu trữ chuỗi truy vấn cho Books API.

Chỉnh sửa hàm khởi tạo để nhận một chuỗi (String) làm đối số và gán nó cho biến mQueryString

Android Developer Fundamentals Course (V2) – Unit 3

3. Tạo một biến thành viên có tên mQueryString để lưu trữ chuỗi truy vấn cho Books API.

Chỉnh sửa hàm khởi tạo để nhận một chuỗi (String) làm đối số và gán nó cho biến mQueryString.

```
private String mQueryString;
```

```
BookLoader(Context context, String queryString) {
```

```
 super(context);
```

```
 mQueryString = queryString;
```

```
}
```

4.3, Chính sửa MainActivity

Kết nối giữa AsyncTaskLoader và Activity gọi nó được triển khai bằng giao diện LoaderManager.LoaderCallbacks. Các callback của loader là một tập hợp các phương thức trong Activity, được LoaderManager gọi khi: Loader được tạo thành công, dữ liệu đã tải xong và, loader được đặt lại. Các callback này lấy kết quả của tác vụ và truyền chúng trở lại giao diện người dùng của Activity.

Trong nhiệm vụ này, bạn sẽ triển khai giao diện LoaderManager.LoaderCallbacks trong MainActivity để xử lý kết quả của phương thức loadInBackground() từ AsyncTaskLoader.

1, Trong mainActivity, them LoaderManager. LoaderCallback implement ở khai báo lớp, với đối số là kiểu String:

```
public class MainActivity extends AppCompatActivity
implements LoaderManager.LoaderCallbacks<String> {
```

2, implement tất cả các phương thức callback bắt buộc từ interface. Bôm gồm: onCreateLoader(), onLoadFinished(), and onLoaderReset()

```
@NonNull
```

```
@Override
```

```
public Loader<String> onCreateLoader(int id, @Nullable Bundle args) {
```

```
 return null;
```

```
}
```

```
@Override
```

```
public void onLoadFinished(@NonNull Loader<String> loader, String data) {
```

```

}

@Override

public void onLoaderReset(@NonNull Loader<String> loader) { }

```

Về các phương thức bắt buộc:

- onCreateLoader() là được gọi khi bạn khởi tạo(intantiate) loader
- onLoadFinished() được gọi khi công việc loader kết thúc. Cái này là nơi bạn thêm code để cập nhật giao diện UI vs kết quả trả về
- onLoaderReset() xóa hết các nguồn còn tồn tại

Phương thức searchBooks() là phương thức onClick của nút bấm. Trong searchBooks(), hãy thay thế lệnh gọi execute của tác vụ FetchBook bằng lệnh gọi restartLoader(). Truyền chuỗi truy vấn lấy từ EditText vào đối tượng Bundle của loader

```

Bundle queryBundle = new Bundle();

queryBundle.putString("queryString", queryString);

getSupportLoaderManager().restartLoader(0, queryBundle, this);

```

Phương thức restartLoader() được định nghĩa bởi LoaderManager, quản lý tất cả các loader được sử dụng trong một activity hoặc fragment. Mỗi activity có chính xác một thẻ hiện LoaderManager, chịu trách nhiệm cho vòng đời của các Loader mà activity quản lý.

Phương thức restartLoader() nhận ba tham số:

- **ID của loader**, hữu ích khi bạn triển khai nhiều loader trong activity.
- **Đối tượng Bundle** chứa dữ liệu mà loader cần.
- **Thẻ hiện của LoaderCallbacks** mà bạn đã triển khai trong activity. Nếu muốn loader chuyển kết quả về MainActivity, hãy truyền this làm đối số thứ ba.

#### 4.4, implement loader callbacks

Trong nhiệm vụ này, bạn triển khai các phương thức callback onCreateLoader() và onLoadFinished() để xử lý tác vụ chạy nền.

1. Trong onCreateLoader(), thay thế câu lệnh return bằng một câu lệnh trả về một thẻ hiện của lớp BookLoader. Truyền vào **context**(this), Truyền vào **queryString** được lấy từ Bundle đã truyền vào.

@NonNull

@Override

```

public Loader onCreateLoader(int id, @Nullable Bundle args) {

```

```

String queryString = "";
if (args != null) {
 queryString = args.getString("queryString");
}
return new BookLoader(this, queryString);

```

1. Sao chép mã từ onPostExecute() trong lớp FetchBook sang onLoadFinished() trong MainActivity. Xóa lời gọi super.onPostExecute(). Mã này sẽ phân tích cú pháp kết quả JSON để tìm kết quả khớp với chuỗi truy vấn.

2. Xóa tất cả các lời gọi get() cho từng đối tượng TextView. Vì bây giờ việc cập nhật UI diễn ra trong chính Activity, bạn không cần tham chiếu yếu (WeakReference) đến các View ban đầu nữa.

3. hay thế đối số của JSONObject: Thay thế biến s trong hàm khởi tạo JSONObject bằng tham số data.

```
JSONObject jsonObject = new JSONObject(data);
```

**4, Chạy ứng dụng của bạn.** Bạn sẽ có cùng chức năng như trước, nhưng bây giờ sử dụng một loader! Tuy nhiên, khi xoay thiết bị, dữ liệu trên giao diện bị mất. Điều này xảy ra vì khi Activity được tạo lại, nó **không biết** rằng một loader đang chạy. **Giải pháp:** Để kết nối lại với loader, bạn cần gọi **initLoader()** trong phương thức **onCreate()** của MainActivity.

5, Thêm code vào onCreate() để kết nối lại với loader, nếu loader đã tồn tại

```
f(getSupportLoaderManager().getLoader(0)!=null){
 getSupportLoaderManager().initLoader(0,null,this); }
```

Nếu loader đã tồn tại, khởi tạo nó. Và bạn sẽ chỉ muốn

## CODING CHALLENGE

**Thử thách 1:** Khám phá Books API chi tiết hơn và tìm một tham số tìm kiếm giúp giới hạn kết quả chỉ hiển thị các sách có thể tải xuống ở định dạng EPUB. Thêm tham số này vào yêu cầu của bạn và xem kết quả.

## Tóm tắt

- Các tác vụ kết nối mạng không nên được thực thi trên luồng UI. Android runtime thường sẽ đưa ra ngoại lệ nếu bạn cố gắng kết nối mạng hoặc truy cập tệp trên luồng UI.
- Sử dụng Books Search API để truy cập Google Books một cách lập trình. Một yêu cầu API đến Google Books có dạng một URL, và phản hồi là một chuỗi JSON.
- Sử dụng Google APIs Explorer để khám phá Google APIs một cách tương tác.
- Dùng getText() để lấy văn bản từ một EditText view. Để chuyển đổi văn bản thành một chuỗi đơn giản, sử dụng `toString()`.

- Phương thức Uri.buildUpon() trả về một URI.Builder, giúp bạn xây dựng các chuỗi URI.
- Để kết nối internet, bạn phải cấu hình quyền mạng trong tệp Android manifest.

### **Lớp AsyncTask cho phép bạn chạy các tác vụ trong nền thay vì trên luồng UI:**

- Để sử dụng AsyncTask, bạn cần tạo một lớp con kế thừa từ nó. Lớp con phải ghi đè phương thức doInBackground(Params...). Thông thường, lớp con cũng ghi đè onPostExecute(Result).
- Để bắt đầu một AsyncTask, sử dụng execute().
- AsyncTask không thể cập nhật UI nếu activity mà nó điều khiển dừng lại, ví dụ như khi có thay đổi cấu hình thiết bị.

### **Khi một AsyncTask thực thi, nó trải qua bốn bước:**

1. **onPreExecute()** chạy trên luồng UI trước khi tác vụ bắt đầu. Bước này thường được sử dụng để thiết lập tác vụ, chẳng hạn như hiển thị thanh tiến trình trên UI.
2. **doInBackground(Params...)** chạy trên luồng nền ngay sau khi onPreExecute() kết thúc. Bước này thực hiện các tính toán nền có thể mất nhiều thời gian.
3. **onProgressUpdate(Progress...)** chạy trên luồng UI sau khi publishProgress(Progress...) được gọi.
4. **onPostExecute(Result)** chạy trên luồng UI sau khi quá trình tính toán nền kết thúc. Kết quả của quá trình này được truyền vào onPostExecute().

### **AsyncTaskLoader là phiên bản loader của AsyncTask:**

- AsyncTaskLoader cung cấp phương thức loadInBackground(), chạy trên một luồng riêng.
- Kết quả của loadInBackground() được chuyển đến luồng UI thông qua callback onLoadFinished() của LoaderManager.
- Để tạo và phân tích chuỗi JSON, sử dụng các lớp JSON có sẵn trong Java như JSONObject và JSONArray.
- AsyncTaskLoader sử dụng một lớp trung gian giúp AsyncTask để thực hiện công việc trong nền, ngoài luồng chính.
- Các instance của AsyncTaskLoader được quản lý bởi LoaderManager.
- LoaderManager cho phép liên kết một Activity mới tạo với một loader bằng cách sử dụng getSupportFragmentManager().initLoader().

### **Khái niệm liên quan**

#### **Bài tập về nhà**

Bạn cần xây dựng một ứng dụng Android hiển thị nội dung của một trang web từ URL do người dùng nhập. Ứng dụng sẽ có các thành phần sau:

- Trường nhập URL: Cho phép người dùng nhập địa chỉ trang web.
- Lựa chọn giao thức (HTTP hoặc HTTPS): Sử dụng Spinner hoặc RadioGroup để chọn giao thức.

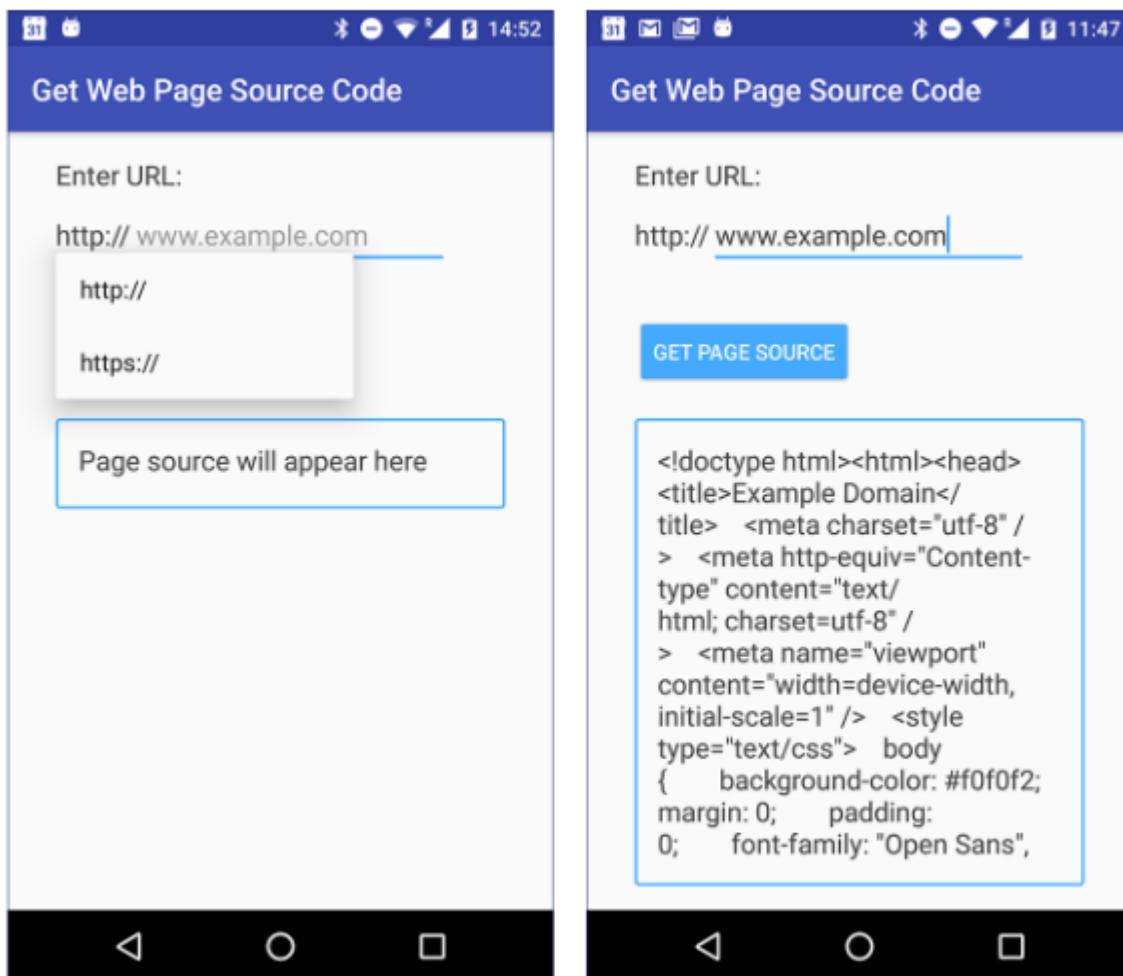
- Nút tải trang: Khi nhấn, ứng dụng sẽ gửi yêu cầu HTTP và hiển thị nội dung của trang.
- Hiển thị kết quả: Một TextView hoặc WebView để hiển thị nội dung của trang web.

Một nút thực hiện tác vụ khi người dùng nhấn vào. Một màn hình cuộn hiển thị mã nguồn của trang web từ URL.

Sử dụng **AsyncTaskLoader** để lấy mã nguồn của trang web từ URL. Bạn cần triển khai một lớp con của **AsyncTaskLoader**. Nếu kết nối internet không khả dụng khi người dùng nhấn nút, ứng dụng phải hiển thị thông báo phù hợp, ví dụ:

*"Kiểm tra kết nối internet và thử lại."*

- Giao diện hiển thị phải chứa một **TextView** trong một **ScrollView** để hiển thị mã nguồn, nhưng bố cục cụ thể tùy thuộc vào bạn. Bạn có thể sử dụng menu bật lên, **Spinner**, hoặc hộp kiểm để cho phép người dùng chọn giao thức HTTP hoặc HTTPS.
- Hình ảnh bên trái hiển thị màn hình bắt đầu với menu bật lên để chọn giao thức. Hình ảnh bên phải hiển thị kết quả sau khi lấy mã nguồn của trang web với URL đã nhập.



Câu hỏi 1

Ứng dụng của bạn cần quyền gì để kết nối internet?

- android.permission.CONNECTIVITY
- android.permission.INTERNET
- Ứng dụng không cần bất kỳ quyền đặc biệt nào, vì tất cả ứng dụng Android đều được phép kết nối internet.

#### Câu hỏi 2

Làm cách nào để ứng dụng của bạn kiểm tra xem kết nối internet có sẵn không?

Trong tệp manifest:

- Yêu cầu quyền ACCESS\_NETWORK\_STATE
- Yêu cầu quyền ALL\_NETWORK\_STATE
- Yêu cầu quyền NETWORK\_CONNECT

Trong mã:

- Bọc mã kết nối internet trong khối try/catch và bắt lỗi NO\_NETWORK.
  - Sử dụng ConnectivityManager để kiểm tra mạng đang hoạt động trước khi kết nối.
  - Hiển thị hộp thoại nhắc người dùng đảm bảo kết nối internet trước khi cố gắng kết nối.
- 

#### Câu hỏi 3

Bạn triển khai phương thức callback của loader ở đâu khi nó hoàn thành tác vụ?

- Trong lớp con của AsyncTaskLoader. AsyncTaskLoader phải triển khai LoaderManager.LoaderCallbacks.
- Trong Activity hiển thị kết quả của tác vụ. Activity phải triển khai LoaderManager.LoaderCallbacks.
- Trong một lớp tiện ích (Utility class) mở rộng Object và triển khai LoaderManager.LoaderCallbacks.

#### Câu hỏi 4

Khi người dùng xoay thiết bị, AsyncTask và AsyncTaskLoader hoạt động khác nhau như thế nào nếu chúng đang chạy một tác vụ trong nền?

- Một AsyncTask đang chạy sẽ bị ngắt kết nối khỏi Activity nhưng vẫn tiếp tục chạy.
- Một AsyncTaskLoader đang chạy sẽ bị ngắt kết nối khỏi Activity và dừng lại, giúp tiết kiệm tài nguyên hệ thống.
- Khi người dùng xoay thiết bị, AsyncTask và AsyncTaskLoader hoạt động khác nhau như thế nào nếu chúng đang chạy một tác vụ trong nền?
- Một AsyncTask đang chạy sẽ bị ngắt kết nối khỏi Activity và dừng lại, giúp tiết kiệm tài nguyên hệ thống. Một AsyncTaskLoader đang chạy sẽ tự động khởi động lại tác vụ từ đầu. Activity sẽ hiển thị kết quả.
- Một AsyncTask đang chạy sẽ bị ngắt kết nối khỏi Activity, nhưng vẫn tiếp tục chạy. Một AsyncTaskLoader đang chạy sẽ tự động kết nối lại với Activity sau khi thiết bị xoay. Activity sẽ hiển thị kết quả.

## 1.3 Broadcast receivers

Broadcasts (phát sóng) là các tin nhắn mà hệ thống Android và các ứng dụng Android gửi khi có các sự kiện xảy ra có thể ảnh hưởng đến chức năng của các ứng dụng hoặc thành phần ứng dụng khác. Ví dụ, hệ thống Android gửi một broadcast hệ thống khi thiết bị khởi động hoặc khi tai nghe được kết nối hoặc ngắt kết nối. Nếu tai nghe có dây bị rút ra, ứng dụng phát nhạc của bạn có thể muốn tạm dừng nhạc.

Ứng dụng Android của bạn cũng có thể gửi broadcast về các sự kiện, chẳng hạn khi dữ liệu mới được tải xuống và có thể hữu ích cho một ứng dụng khác. Những sự kiện mà ứng dụng của bạn gửi đi được gọi là custom broadcasts (broadcast tùy chỉnh).

Nhìn chung, bạn có thể sử dụng broadcast như một hệ thống nhắn tin giữa các ứng dụng và bên ngoài luồng hoạt động thông thường của người dùng.

Broadcast được nhận bởi bất kỳ ứng dụng hoặc thành phần ứng dụng nào có broadcast receiver đã được đăng ký để nhận hành động đó. BroadcastReceiver là lớp cơ sở cho mã nhận broadcast intents. Để tìm hiểu thêm về broadcast receivers, hãy xem tổng quan về Broadcasts và tài liệu tham khảo về Intent.

Lưu ý: Mặc dù lớp Intent được sử dụng để gửi và nhận broadcast, nhưng cơ chế broadcast Intent hoàn toàn tách biệt với các intent được sử dụng để khởi chạy activities.

Trong bài thực hành này, bạn sẽ tạo một ứng dụng phản hồi khi trạng thái sạc của thiết bị thay đổi. Để làm điều này, ứng dụng của bạn sẽ nhận và phản hồi một **broadcast hệ thống**, đồng thời cũng gửi và nhận một **broadcast tùy chỉnh**.

Bạn nên đã biết:

Bạn có thể

- Định danh khóa của AndroidManifest.xml
- Tạo intent ẩn

Bạn sẽ học:

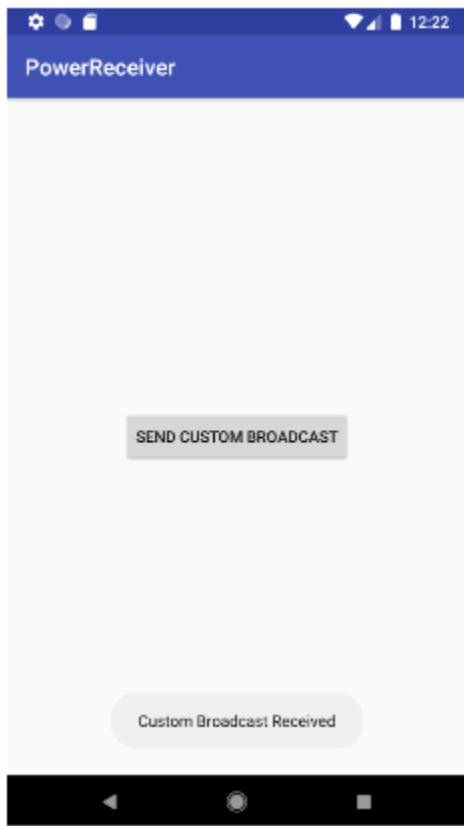
- Cách để kế thừa BroadcastReceiver và implement nó
- Đăng ký cho intent broadcast hệ thống
- Tạo và gửi các intent được chỉnh sửa

Bạn sẽ làm:

- Kế thừa BroadcastReceiver để hiển thị 1 toast khi broadcast được nhận
- Đăng ký trình nhận để lắng nghe broadcasts từ hệ thống
- Gửi và nhận intent broadcast được chỉnh sửa

App overview:

Ứng dụng PowerReceiver sẽ đăng ký một BroadcastReceiver để hiển thị thông báo Toast khi thiết bị được kết nối hoặc ngắt kết nối với nguồn điện. Ứng dụng cũng sẽ gửi và nhận một broadcast tùy chỉnh để hiển thị một thông báo Toast khác.



## Task 1: Thiết lập PowerReceiver Project

### 1.1, tạo 1 dự án

1, Trong Android Studio, tạo một dự án Java mới có tên PowerReceiver. Chấp nhận các tùy chọn mặc định và sử dụng mẫu Empty Activity.

2, Để tạo một broadcast receiver mới, chọn tên package trong Android Project View, sau đó điều hướng đến File > New > Other > Broadcast Receiver.

3, Đặt tên lớp là CustomReceiver. Đảm bảo rằng Java được chọn làm ngôn ngữ nguồn, và đánh dấu chọn Exported và Enabled:

- Exported cho phép broadcast receiver nhận broadcasts từ bên ngoài ứng dụng.
- Enabled cho phép hệ thống khởi tạo receiver.

### 1.2 Đăng ký receiver để nhận system broadcasts

Một system broadcast là thông báo mà hệ thống Android gửi khi một sự kiện hệ thống xảy ra. Mỗi system broadcast được bao bọc trong một đối tượng Intent:

- Trường action của Intent chứa thông tin chi tiết về sự kiện, chẳng hạn như android.intent.action.HEADSET\_PLUG, được gửi khi một tai nghe có dây được kết nối hoặc ngắt kết nối.
- Intent có thể chứa extra data bổ sung về sự kiện trong extra field, ví dụ như một giá trị boolean để chỉ báo xem tai nghe có đang kết nối hay không.

Ứng dụng có thể đăng ký để nhận broadcasts cụ thể. Khi hệ thống gửi một broadcast, nó sẽ chuyển tiếp broadcast đó đến các ứng dụng đã đăng ký nhận loại broadcast cụ thể đó.

Có hai cách đăng ký một BroadcastReceiver:

- Static Receiver: Đăng ký thông qua thẻ <receiver> trong AndroidManifest.xml. Những receiver này còn được gọi là manifest-declared receivers.
- Dynamic Receiver: Đăng ký bằng app context hoặc activity context. Receiver này chỉ nhận broadcasts khi context đăng ký vẫn còn hiệu lực (tức là khi ứng dụng hoặc activity đang chạy). Những receiver này còn được gọi là context-registered receivers.

Trong ứng dụng này, bạn quan tâm đến hai system broadcasts:

- ACTION\_POWER\_CONNECTED: Được gửi khi thiết bị được kết nối với nguồn điện.
- ACTION\_POWER\_DISCONNECTED: Được gửi khi thiết bị bị ngắt kết nối với nguồn điện.



Lưu ý:

Từ Android 8.0 (API level 26) trở lên, bạn không thể sử dụng static receivers để nhận hầu hết các system broadcasts, ngoại trừ một số trường hợp đặc biệt. Vì vậy, trong nhiệm vụ này, bạn sẽ sử dụng dynamic receivers.

## 1. (Tùy chọn) Chỉnh sửa AndroidManifest.xml

- Mở tệp AndroidManifest.xml.
- Android Studio đã tự động tạo một thẻ <receiver>, nhưng bạn không cần nó, vì bạn không thể sử dụng static receiver để lắng nghe system broadcasts liên quan đến kết nối nguồn điện.
- Xóa toàn bộ thẻ <receiver> khỏi tệp AndroidManifest.xml.

## 2. Tạo CustomReceiver trong MainActivity.java

- Mở MainActivity.java.
- Tạo một biến thành viên (member variable) cho CustomReceiver.
- Khởi tạo đối tượng CustomReceiver trong onCreate().

```
private CustomReceiver mReceiver = new CustomReceiver()
```

## Tạo Intent Filter với Intent Actions

**Intent filters** được sử dụng để xác định loại Intent mà một thành phần có thể nhận. Chúng giúp lọc Intent dựa trên các giá trị như **action** và **category**.

1, trong mainActivity.java ở cuối onCreate(), tạo Intentfilter object

```
IntentFilter filter = new IntentFilter();
```

Khi hệ thống nhận một Intent dưới dạng broadcast, nó sẽ tìm kiếm các broadcast receiver dựa trên giá trị action được chỉ định trong đối tượng IntentFilter.

2, Trong MainActivity.java, ở cuối phương thức onCreate(), thêm các hành động ACTION\_POWER\_CONNECTED và ACTION\_POWER\_DISCONNECTED vào IntentFilter.

Register the receiver using the activity context.

```
this.registerReceiver(mReceiver, filter);
```

### Đăng ký và hủy đăng ký receiver

1, Trong MainActivity.java, ở cuối phương thức onCreate(), đăng ký receiver của bạn bằng cách sử dụng context của MainActivity. Receiver của bạn sẽ hoạt động và có thể nhận broadcast miễn là MainActivity đang chạy.

```
this.registerReceiver(mReceiver, filter);
```

2, Trong MainActivity.java, ghi đè phương thức onDestroy() và hủy đăng ký receiver của bạn. Để tiết kiệm tài nguyên hệ thống và tránh rò rỉ bộ nhớ, các dynamic receiver phải được hủy đăng ký khi không còn cần thiết hoặc trước khi activity hoặc ứng dụng tương ứng bị hủy, tùy thuộc vào context đã sử dụng.

### Task 2. Gửi và nhận một broadcast tùy chỉnh

Ngoài việc phản hồi các system broadcasts, ứng dụng của bạn có thể gửi và nhận các broadcast tùy chỉnh. Sử dụng broadcast tùy chỉnh khi bạn muốn ứng dụng thực hiện một hành động mà không cần khởi chạy một activity, ví dụ như khi bạn muốn thông báo cho các ứng dụng khác biết rằng dữ liệu đã được tải xuống thiết bị.

Android cung cấp ba cách để ứng dụng gửi broadcast tùy chỉnh:

- Normal broadcasts (Broadcast thông thường): Hoạt động bất đồng bộ (asynchronous). Các receiver của loại broadcast này chạy theo thứ tự không xác định, thường là cùng lúc. Để gửi một normal broadcast, hãy tạo một Intent broadcast và truyền nó vào phương thức sendBroadcast(Intent).
- Local broadcasts (Broadcast cục bộ): Chỉ được gửi đến các receiver nằm trong cùng một ứng dụng với sender. Để gửi một local broadcast, hãy tạo một Intent broadcast và truyền nó vào phương thức LocalBroadcastManager.sendBroadcast.
- Ordered broadcasts (Broadcast có thứ tự): Được gửi đến từng receiver một. Mỗi receiver có thể chuyển tiếp kết quả đến receiver tiếp theo hoặc có thể hủy broadcast, khiến broadcast không tiếp tục được gửi đi.

Thông điệp broadcast được gói trong một đối tượng Intent. Chuỗi action của Intent phải tuân theo cú pháp tên gói Java của ứng dụng và phải duy nhất để xác định sự kiện broadcast.

Đối với broadcast tùy chỉnh, bạn cần tự định nghĩa action của Intent (một chuỗi duy nhất). Bạn có thể tạo các đối tượng Intent với action tùy chỉnh và tự phát broadcast từ ứng dụng của mình bằng một trong các phương thức đã đề cập ở trên. Các ứng dụng có BroadcastReceiver được đăng ký cho action đó sẽ nhận được broadcast.

Trong bài này, bạn sẽ:

- Thêm một nút vào activity để gửi local broadcast Intent.
- Receiver sẽ đăng ký Intent broadcast đó và hiển thị kết quả trong một thông báo toast.

```
private static final String ACTION_CUSTOM_BROADCAST =
BuildConfig.APPLICATION_ID + ".ACTION_CUSTOM_BROADCAST";
```

## 2.1 Xác định chuỗi action tùy chỉnh cho broadcast

Cả bên gửi và bên nhận của một custom broadcast cần phải thống nhất về chuỗi action duy nhất cho Intent được broadcast.

Một cách phổ biến để tạo action duy nhất là thêm tên gói ứng dụng của bạn vào trước action name.

Cách lấy tên gói ứng dụng:

Bạn có thể sử dụng BuildConfig.APPLICATION\_ID, giá trị này được lấy từ thuộc tính applicationId trong tệp build.gradle cấp mô-đun.

Thực hiện:

1, Tạo một biến hằng số (constant member variable) trong cả MainActivity và CustomReceiver.

2, Dùng biến này làm action cho Intent broadcast.

**Quan trọng:** Mặc dù Intent được sử dụng cho cả việc gửi broadcast và khởi chạy activity bằng startActivity(Intent), nhưng hai hành động này **hoàn toàn không liên quan** đến nhau.

- **Broadcast receivers không thể** nhìn thấy hoặc chặn một Intent được dùng để **khởi chạy activity**.
- Tương tự, khi bạn **broadcast một Intent**, bạn **không thể** sử dụng Intent đó để **tìm hoặc khởi chạy một activity**.

## 2.2 Thêm nút "Gửi Broadcast Tùy Chỉnh"

1, Trong tệp **activity\_main.xml**, thay thế **TextView "Hello World"** bằng một **Button** có các thuộc tính sau:

2, Trích xuất **chuỗi tài nguyên**.

Phương thức sendCustomBroadcast() sẽ là **trình xử lý sự kiện khi nhấn nút**. Để tạo một **stub** cho sendCustomBroadcast() trong **Android Studio**:

1, Nhập vào **tên phương thức** sendCustomBroadcast, phần này sẽ **được đánh dấu màu vàng**. Một **biểu tượng bóng đèn đỏ** sẽ xuất hiện bên trái.

2, Nhập vào **bóng đèn đỏ**, sau đó chọn **Create 'sendCustomBroadcast(View)' in 'MainActivity'**.

```
<Button
 android:id = "@+id/sendBroadcast"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Send Custom Broadcast"
 android:onClick="sendCustomBroadcast"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintLeft_toLeftOf="parent"
 app:layout_constraintRight_toRightOf="parent"
 app:layout_constraintTop_toTopOf="parent" />
```

### 2.3 Triển khai phương thức sendCustomBroadcast()

Vì broadcast này chỉ dành riêng cho ứng dụng của bạn, hãy sử dụng **LocalBroadcastManager** để quản lý broadcast. LocalBroadcastManager là một lớp cho phép bạn đăng ký và gửi broadcast **chỉ trong phạm vi ứng dụng**.

Tạo một Intent mới, truyền vào **chuỗi hành động tùy chỉnh** của bạn làm đối số.

Gửi broadcast nội bộ giúp đảm bảo rằng **dữ liệu ứng dụng của bạn không bị chia sẻ** với các ứng dụng Android khác. Điều này **tăng cường bảo mật thông tin** và **cải thiện hiệu suất hệ thống**.

Trong **MainActivity.java**, bên trong phương thức sendCustomBroadcast(), hãy thực hiện các bước sau:

1, Tạo 1 Intent mới, với chuỗi hành động tùy chỉnh làm đối số

```
Intent customBroadcastIntent = new Intent(ACTION_CUSTOM_BROADCAST);
```

2, Sau khi khai báo Intent tùy chỉnh, gửi broadcast bằng cách sử dụng lớp LocalBroadcastManager.  
LocalBroadcastManager.getInstance(this).sendBroadcast(customBroadcastIntent);

### 2.4 Đăng ký và hủy đăng ký broadcast tùy chỉnh của bạn

Việc đăng ký một local broadcast tương tự như đăng ký một system broadcast, nghĩa là bạn sẽ sử dụng một dynamic receiver. Đối với các broadcast được gửi bằng LocalBroadcastManager, bạn không thể đăng ký tĩnh trong tệp manifest.

Nếu bạn đăng ký một broadcast receiver động, bạn cần phải hủy đăng ký nó khi không còn cần thiết. Trong ứng dụng của bạn, receiver chỉ cần phản hồi với custom broadcast khi ứng dụng đang chạy, vì vậy bạn có thể đăng ký action trong onCreate() và hủy đăng ký trong onDestroy().

```
LocalBroadcastManager.getInstance(this).registerReceiver(mReceiver, new IntentFilter(ACTION_CUSTOM_BROADCAST));
```

2.5 Trong phương thức onReceive() của lớp CustomReceiver, bạn cần thêm một câu lệnh case để xử lý hành động Intent tùy chỉnh. Để làm điều này, hãy thực hiện các bước sau:

1. Mở tệp CustomReceiver.java.
2. Trong phương thức onReceive(), thêm một câu lệnh case cho hành động Intent tùy chỉnh của bạn.  
Ví dụ:

**Code Challenge:** Nếu bạn đang phát triển một ứng dụng trình phát nhạc, ứng dụng của bạn có thể cần phát hoặc tạm dừng nhạc khi người dùng kết nối hoặc ngắt kết nối tai nghe có dây. Để triển khai chức năng này, bạn cần một broadcast receiver phản hồi các sự kiện tai nghe có dây.

## Tóm tắt

- *Broadcast receivers* là thành phần quan trọng của ứng dụng Android.
- *Broadcast receivers* có thể nhận *broadcast* do hệ thống hoặc ứng dụng khác gửi.
- *Intent* được sử dụng trong cơ chế *broadcast* hoàn toàn khác với *intent* dùng để khởi chạy *activities*.
- Để xử lý *Intent* nhận được từ *broadcast*, bạn cần kế thừa lớp BroadcastReceiver và triển khai phương thức onReceive().
- Bạn có thể đăng ký một *broadcast receiver* trong tệp Android manifest hoặc bằng mã lệnh (*programmatically*).
- *Local broadcasts* chỉ hoạt động trong phạm vi ứng dụng của bạn. Để đăng ký và gửi *local broadcasts*, sử dụng LocalBroadcastManager. *Local broadcasts* không liên quan đến giao tiếp giữa các tiến trình (IPC), giúp tăng hiệu suất và bảo mật cho ứng dụng.
- Để tạo tên hành động (*action name*) duy nhất cho *broadcasts*, một cách phổ biến là thêm tiền tố tên gói (*package name*) vào tên hành động.
- Nếu ứng dụng của bạn nhắm đến API cấp 26 trở lên, bạn không thể khai báo *receiver* cho hầu hết

## Bài 2. Kích hoạt lập lịch và thông báo nhiệm vụ

### 2.1 Thông báo

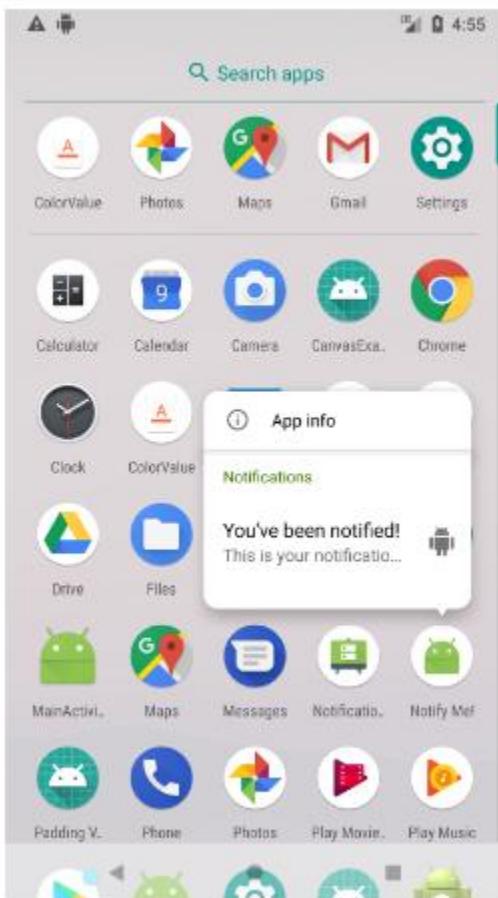
#### Giới thiệu

Đôi khi, bạn muốn ứng dụng của mình hiển thị thông tin cho người dùng ngay cả khi ứng dụng không chạy ở chế độ nền. Ví dụ, bạn có thể muốn thông báo cho người dùng rằng có nội dung mới hoặc đội thể thao yêu thích của họ vừa ghi bàn trong một trận đấu.

Hệ thống thông báo (notification) của Android cung cấp cách để ứng dụng gửi thông báo đến người dùng ngay cả khi ứng dụng không ở chế độ nền.

Một thông báo là một tin nhắn mà ứng dụng hiển thị bên ngoài giao diện thông thường. Thông báo xuất hiện dưới dạng biểu tượng trong khu vực thông báo của thiết bị (trên thanh trạng thái). Người dùng có thể xem chi tiết thông báo bằng cách mở ngăn thông báo (vuốt xuống từ thanh trạng thái). Khu vực thông báo và ngăn thông báo là các phần do hệ thống kiểm soát, người dùng có thể truy cập bất cứ lúc nào.

Trên các thiết bị chạy Android 8.0 trở lên, khi ứng dụng có thông báo mới, biểu tượng ứng dụng sẽ hiển thị dấu chấm thông báo (notification badge/dot). Khi người dùng nhấn giữ biểu tượng ứng dụng, thông báo sẽ xuất hiện phía trên biểu tượng.



## App Overview

Notify Me! là một ứng dụng cho phép người dùng kích hoạt, cập nhật và hủy một thông báo bằng ba nút hiển thị trong ảnh chụp màn hình bên dưới. Trong quá trình tạo ứng dụng, bạn sẽ thử nghiệm với các kiểu thông báo, hành động và mức độ ưu tiên.

### Task1: Tạo thông báo đơn giản

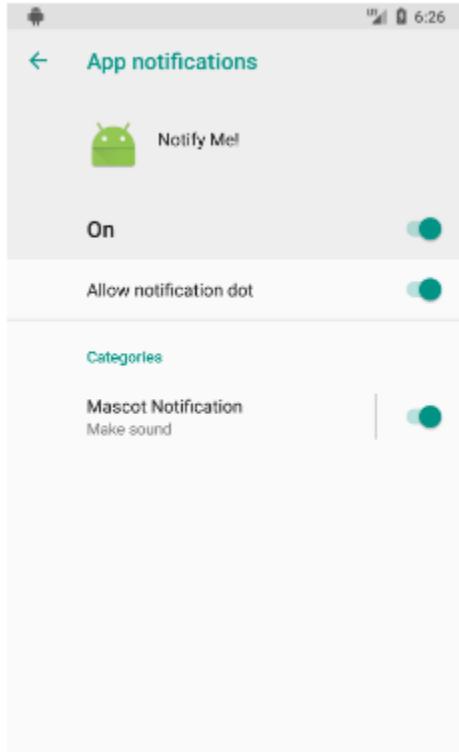
#### 1.1, Tạo project

1, In Android Studio, create a new project called "Notify Me!" Accept the default options and use the Empty Activity template.

2, In your activity\_main.xml layout file, replace the default TextView with a button that has the following attributes:

```
<Button
 android:id="@+id/notify"
```

#### 1.2 Tạo kênh thông báo:



Trong ứng dụng *Cài đặt* trên thiết bị chạy Android, người dùng có thể điều chỉnh các thông báo họ nhận được.

Bắt đầu từ Android 8.0 (API cấp 26), mã của bạn có thể gán từng thông báo của ứng dụng vào một kênh thông báo có thể tùy chỉnh bởi người dùng:

- Mỗi kênh thông báo đại diện cho một loại thông báo.
- Trong mã của bạn, có thể nhóm nhiều thông báo vào cùng một kênh thông báo.
- Ứng dụng của bạn có thể thiết lập hành vi cho từng kênh thông báo, và hành vi này sẽ áp dụng cho tất cả thông báo trong kênh đó. Ví dụ, ứng dụng có thể đặt thông báo trong kênh phát âm thanh, nháy đèn hoặc rung.
- Dù ứng dụng đặt hành vi thế nào cho kênh thông báo, người dùng vẫn có thể thay đổi hoặc tắt toàn bộ thông báo của ứng dụng.

Để hiển thị thông báo trong một ứng dụng Android nhắm đến Android 8.0 (API cấp 26) trở lên, bạn phải tạo ít nhất một kênh thông báo. Dưới đây là các bước chính để thực hiện:

1, Trong MainActivity.java, hãy tạo một biến thành viên để lưu trữ đối tượng NotificationManager

```
private static final String PRIMARY_CHANNEL_ID = "primary_notification_channel";
```

2, Trong MainActivity, tạo một hằng số cho ID kênh thông báo. Mỗi kênh thông báo phải được liên kết với một ID duy nhất trong gói ứng dụng của bạn. Bạn sẽ sử dụng ID này sau đó để gửi thông báo.

```
private NotificationManager mNotifyManager;
```

3, Trong MainActivity.java, tạo một phương thức createNotificationChannel() và khởi tạo NotificationManager bên trong phương thức này

```
public void createNotificationChannel()
{mNotifyManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);}
```

### 1.3 Xây dựng thông báo đầu tiên của bạn

Thông báo được tạo bằng lớp NotificationCompat.Builder, cho phép bạn thiết lập nội dung và hành vi của thông báo. Một thông báo có thể chứa các thành phần sau:

- Biểu tượng (bắt buộc): Được thiết lập bằng phương thức setSmallIcon().
- Tiêu đề (tùy chọn): Được thiết lập bằng phương thức setContentTitle().
- Văn bản chi tiết (tùy chọn): Được thiết lập bằng phương thức setContentText().

Tạo biểu tượng thông báo bắt buộc:

1, Trong Android Studio, vào File > New > Image Asset.

2, Trong danh sách thả xuống Icon Type, chọn Notification Icons.

3, Nhập vào biểu tượng bên cạnh mục Clip Art để chọn một biểu tượng theo tiêu chuẩn Material Design cho thông báo.

Đổi với ứng dụng này, hãy sử dụng biểu tượng Android

Đổi tên tài nguyên ic\_android, sau đó nhấn Next và Finish. Thao tác này sẽ tạo các tệp drawable với độ phân giải khác nhau cho các cấp API khác nhau.

Để tạo và hiển thị thông báo:

1, Bạn cần liên kết thông báo với một ID thông báo để có thể cập nhật hoặc hủy thông báo trong tương lai.

Trong MainActivity.java, tạo một hằng số cho ID thông báo:

#### 1.4 Thêm intent nội dung và đóng thông báo

Các intent nội dung cho thông báo tương tự như các intent mà bạn đã sử dụng trong suốt khóa học này. Intent nội dung có thể là: Intent tường minh để khởi chạy một Activity. Intent ẩn để thực hiện một hành động. Intent phát sóng để thông báo cho hệ thống về một sự kiện hệ thống hoặc sự kiện tùy chỉnh. Sự khác biệt chính với một Intent được sử dụng cho thông báo là Intent phải được bao bọc trong một PendingIntent. PendingIntent cho phép hệ thống thông báo của Android thực hiện hành động được chỉ định thay mặt cho mã của bạn.

Trong bước này, bạn sẽ cập nhật ứng dụng của mình như sau: Khi người dùng nhấn vào thông báo, ứng dụng sẽ gửi một intent nội dung để khởi chạy MainActivity.(Nếu ứng dụng đang mở và đang hoạt động, thao tác nhấn vào thông báo sẽ không có hiệu lực.)

Thực hiện trong MainActivity.java

1, Trong MainActivity.java, ở đầu phương thức getNotificationBuilder(), tạo một intent tường minh để khởi chạy MainActivity

```
Intent notificationIntent = new Intent(this, MainActivity.class);
```

```
PendingIntent notificationPendingIntent =
```

```
PendingIntent.getActivity(this,
```

```
NOTIFICATION_ID, notificationIntent,
```

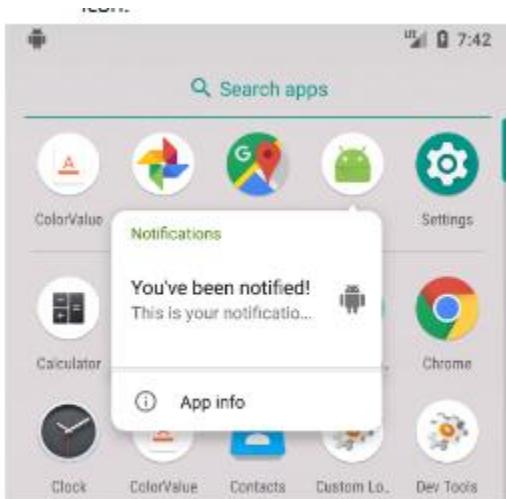
```
PendingIntent.FLAG_UPDATE_CURRENT);
```



Trong ảnh chụp màn hình trên:

- 1, Thông báo trong thanh trạng thái.
- 2, Dấu chấm thông báo trên biểu tượng ứng dụng (chỉ có trên API 26 trở lên).

Khi người dùng chạm và giữ biểu tượng ứng dụng, một cửa sổ bật lên sẽ hiển thị các thông báo cùng với biểu tượng



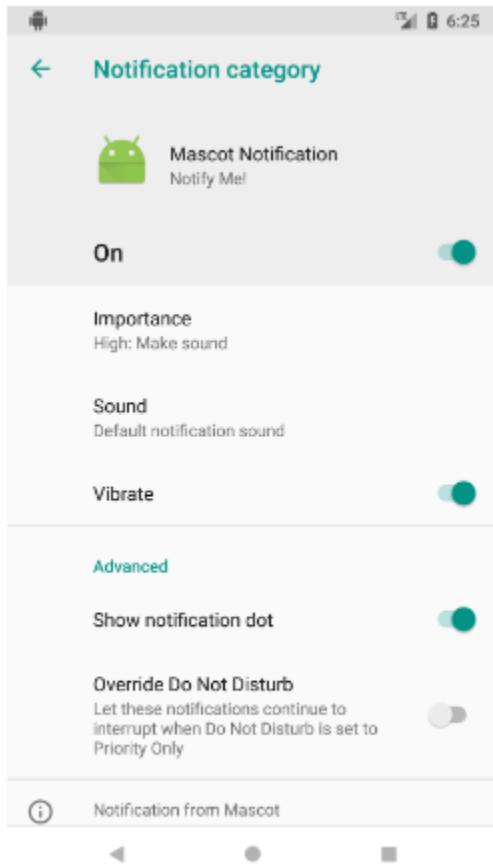
Nếu bạn đang chạy trên thiết bị hoặc trình giả lập với API 26 trở lên, đây là cách xem kênh thông báo mà bạn đã tạo:

Mở ứng dụng Cài đặt trên thiết bị.

- 1, Trong thanh tìm kiếm, nhập tên ứng dụng của bạn, "Notify Me!"

2., Mở Notify Me! > Thông báo ứng dụng > Mascot Notifications. Sử dụng cài đặt này để tùy chỉnh kênh thông báo.

Mô tả của kênh thông báo sẽ hiển thị ở cuối màn hình.



### 1.5 Thêm mức độ ưu tiên và tùy chọn mặc định cho thông báo để hỗ trợ các thiết bị cũ

Lưu ý: Nhiệm vụ này cần thiết cho các thiết bị chạy Android 7.1 trở xuống, đây là phần lớn thiết bị Android hiện nay. Đối với các thiết bị chạy Android 8.0 trở lên, bạn sử dụng kênh thông báo để thiết lập mức độ ưu tiên và tùy chọn mặc định.

Tuy nhiên, để đảm bảo tương thích ngược, bạn vẫn nên hỗ trợ các thiết bị cũ hơn.

Mô tả. Khi người dùng nhấn vào nút Notify Me! trong ứng dụng, thông báo sẽ được gửi đi. Tuy nhiên, người dùng chỉ thấy biểu tượng trong thanh thông báo.

Để thu hút sự chú ý của người dùng, bạn cần đặt các tùy chọn mặc định cho thông báo.

Mức độ ưu tiên của thông báo

- Giá trị mức độ ưu tiên dao động từ PRIORITY\_MIN (-2) đến PRIORITY\_MAX (2).
- Thông báo có mức độ ưu tiên cao hơn sẽ hiển thị trước trong khay thông báo.
- Thông báo có mức độ ưu tiên HIGH hoặc MAX sẽ hiển thị dưới dạng "heads-up", tức là rơi xuống phía trên màn hình đang hoạt động của người dùng.

Lưu ý: Không nên đặt tất cả thông báo ở mức MAX, vì điều này có thể gây khó chịu cho người dùng.

## Thực hiện

1, Trong phương thức getNotificationBuilder(), thiết lập mức độ ưu tiên của thông báo thành HIGH bằng cách thêm dòng sau vào đối tượng notification builder:

```
.setPriority(NotificationCompat.PRIORITY_HIGH)
```

Task 2:

Button

```
android:id="@+id/notify"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Notify Me!"

app:layout_constraintBottom_toTopOf="@+id/update"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

<Button
 android:id="@+id/update"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Update Me!"

 app:layout_constraintBottom_toTopOf="@+id/cancel"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/notify" />

<Button
```

```
 android:id="@+id/cancel"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Cancel Me!"

 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@+id/update" />
```

1, Trích thông tin từ chuỗi của strings.xml

2, thêm 1 biến chổ mỗi button

```
private Button button_cancel;

private Button button_update
```

## 2.2 Triển khai phương thức hủy và cập nhật thông báo

Để hủy thông báo, bạn gọi phương thức cancel() của NotificationManager, truyền vào ID của thông báo.

Thực hiện

1, Trong tệp MainActivity.java, bên trong phương thức cancelNotification(), thêm dòng sau:

2, Chạy ứng dụng trên thiết bị hoặc trình giả lập.

3, Nhấn nút Notify Me! để gửi thông báo.

4, Biểu tượng thông báo sẽ xuất hiện trên thanh trạng thái.

5, Nhấn nút Cancel Me!.

Thông báo sẽ bị hủy.

---

## Cập nhật thông báo với BigPictureStyle

Việc cập nhật thông báo phức tạp hơn việc hủy vì Android hỗ trợ các kiểu thông báo có thể làm gọn nội dung.

Ví dụ:

Ứng dụng Gmail sử dụng InboxStyle nếu có nhiều email chưa đọc, giúp gộp thông tin thành một thông báo duy nhất.

Trong ví dụ này, bạn sẽ cập nhật thông báo để sử dụng BigPictureStyle, cho phép thêm hình ảnh vào thông báo.

Thực hiện:

- Tải xuống hình ảnh để sử dụng trong thông báo và đổi tên thành mascot\_1.
- Nếu sử dụng hình ảnh của riêng bạn, hãy đảm bảo:
  - Tỷ lệ khung hình là 2:1

Chiều rộng không quá 450 dp.

Đặt hình ảnh mascot\_1 vào thư mục res/drawable.

mNotifyManager.cancel(NOTIFICATION\_ID)

### 2.3 Chuyển đổi trạng thái của nút

Trong ứng dụng này, **người dùng có thể bị nhầm lẫn** vì trạng thái của thông báo không được theo dõi trong Activity.

Ví dụ:

- Người dùng có thể nhấn Cancel Me! khi không có thông báo nào đang hiển thị.
- Bạn có thể khắc phục vấn đề này bằng cách bật hoặc tắt các nút tùy thuộc vào trạng thái của thông báo:
- Khi ứng dụng mới chạy, chỉ nút Notify Me! được bật, vì chưa có thông báo để cập nhật hoặc hủy.
- Sau khi thông báo được gửi đi, hai nút Cancel Me! và Update Me! được bật, còn Notify Me! bị tắt.

Khi thông báo được cập nhật, cả hai nút Notify Me! và Update Me! sẽ bị tắt, chỉ còn nút Cancel Me! được bật. Nếu thông báo bị hủy, tất cả các nút sẽ trở lại trạng thái ban đầu, chỉ có nút Notify Me! được bật.

1, Trong tệp MainActivity.java, thêm một phương thức tiện ích có tên setNotificationButtonState() để chuyển đổi trạng thái của các nút.

```
void setNotificationButtonState(Boolean isNotifyEnabled,
Boolean isUpdateEnabled,
Boolean isCancelEnabled) {

 button_notify.setEnabled(isNotifyEnabled);

 button_update.setEnabled(isUpdateEnabled);

 button_cancel.setEnabled(isCancelEnable)
```

## 2.2 Trình quản lý cảnh báo

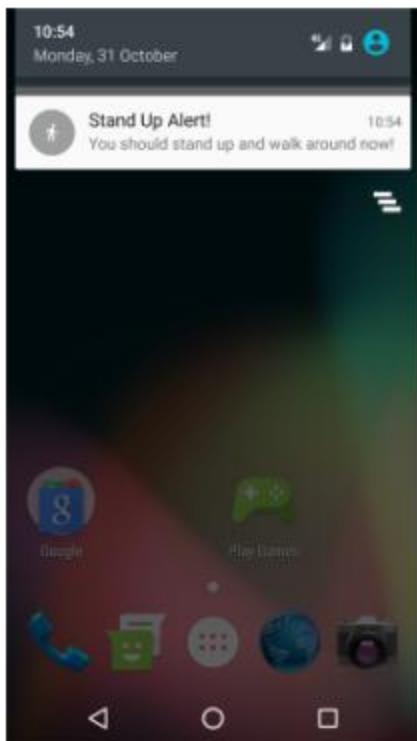
### Giới thiệu

Trong các bài học trước, bạn đã học cách làm cho ứng dụng phản hồi khi người dùng nhấn vào một nút hoặc một thông báo. Bạn cũng đã học cách làm cho ứng dụng phản hồi với các sự kiện hệ thống bằng broadcast receivers. Nhưng nếu ứng dụng của bạn cần thực hiện một hành động vào một thời điểm cụ thể, chẳng hạn như thông báo lịch thì sao?

Trong trường hợp này, bạn sẽ sử dụng AlarmManager. Lớp AlarmManager cho phép bạn khởi chạy và lặp lại một PendingIntent vào một thời điểm xác định hoặc sau một khoảng thời gian nhất định.

Trong bài thực hành này, bạn sẽ tạo một bộ đếm thời gian nhắc nhở người dùng đứng lên sau mỗi 15 phút.

## App Overview



### Task 1: Thiết lập dự án Stand Up! và giao diện

#### 1.1 Tạo bối cảnh cho dự án Stand Up!

1, Trong Android Studio, tạo một dự án mới có tên "Stand Up!". Chấp nhận các tùy chọn mặc định và sử dụng mẫu Empty Activity.

2, Mở tệp activity\_main.xml. Thay thế TextView "Hello World" bằng ToggleButton sau:

#### 1.2 Thiết lập phương thức setOnCheckedChangeListener()

Ứng dụng Stand Up! bao gồm một ToggleButton dùng để bật và tắt báo thức, đồng thời hiển thị trạng thái của báo thức. Để thiết lập báo thức khi nút bật, ứng dụng sử dụng phương thức onCheckedChangeListener().

1, Gọi setOnCheckedChangeListener() trên thẻ hiện ToggleButton, sau đó bắt đầu nhập "new OnCheckedChangeListener". Android Studio sẽ tự động hoàn thành phương thức cho bạn, bao gồm cả phương thức onCheckedChanged() được ghi đè.

```
String toastMessage;
if(isChecked){
 //Set the toast message for the "on" case.
 toastMessage = "Stand Up Alarm On!";
} else {
 //Set the toast message for the "off" case.
 toastMessage = "Stand Up Alarm Off!";
}

//Show a toast to say the alarm is turned on or off.
Toast.makeText(MainActivity.this, toastMessage,Toast.LENGTH_SHORT
```

### Task2: Set up the notification

Bước tiếp theo là tạo thông báo nhắc nhở người dùng đứng dậy sau mỗi 15 phút. Hiện tại, thông báo sẽ được gửi ngay lập tức khi bật nút chuyển đổi.

#### 2.1 Tạo thông báo

Trong bước này, bạn sẽ tạo phương thức deliverNotification() để gửi lời nhắc đứng dậy và đi lại.

Thực hiện các bước sau trong MainActivity.java:

1, Tạo một biến thành viên có tên mNotificationManager thuộc kiểu NotificationManager.

```
private NotificationManager mNotificationManager;
```

2, Trong phương thức onCreate(), khởi tạo mNotificationManager bằng cách sử dụng getSystemService().

```
NotificationManager = (NotificationManager)
```

```
getSystemService(NOTIFICATION_SERVICE)
```

3, Tạo các hằng số thành viên cho **ID thông báo** và **ID kênh thông báo**. Bạn sẽ sử dụng chúng để hiển thị thông báo. Để tìm hiểu thêm về thông báo, hãy xem phần **Tổng quan về thông báo**.

```
private static final int NOTIFICATION_ID = 0;
```

```
private static final String PRIMARY_CHANNEL_ID =
```

```
"primary_notification_channel"
```

#### Tạo kênh thông báo

Đối với Android 8.0 (API cấp 27) trở lên, để hiển thị thông báo cho người dùng, bạn cần tạo một kênh thông báo.

Các bước thực hiện:

1, Tạo một phương thức có tên createNotificationChannel().

2, Gọi createNotificationChannel() ở cuối phương thức onCreate().

```

/**
 * Creates a Notification channel, for OREO and higher.
 */
public void createNotificationChannel() {
 // Create a notification manager object.
 mNotificationManager =
 (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
 // Notification channels are only available in OREO and higher.
 // So, add a check on SDK version.
 if (android.os.Build.VERSION.SDK_INT >=
 android.os.Build.VERSION_CODES.O) {
 // Create the NotificationChannel with all the parameters.
 NotificationChannel notificationChannel = new NotificationChannel
 (PRIMARY_CHANNEL_ID,
 "Stand up notification",
 NotificationManager.IMPORTANCE_HIGH);
 notificationChannel.enableLights(true);
 notificationChannel.setLightColor(Color.RED);
 notificationChannel.enableVibration(true);
 notificationChannel.setDescription
 ("Notifies every 15 minutes to stand up and walk");
 mNotificationManager.createNotificationChannel(notificationChannel);
 }
}

```

## **hêm biểu tượng thông báo và xây dựng thông báo**

### **1. Thêm biểu tượng thông báo**

1. Sử dụng **Image Asset Studio** để thêm một tài nguyên hình ảnh làm biểu tượng thông báo.
2. Chọn một biểu tượng phù hợp cho báo thức và đặt tên là **ic\_stand\_up**.

- Ví dụ: Bạn có thể sử dụng biểu tượng "**đi bộ**" trong danh mục **hướng dẫn đường đi**.

## 2. Xây dựng thông báo

- Trong phương thức **deliverNotification()**, sử dụng **NotificationCompat.Builder** để tạo thông báo.
- Thêm biểu tượng thông báo và **content intent**.
- Đặt mức độ ưu tiên của thông báo và các tùy chọn khác.

NotificationCompat.Builder builder = new NotificationCompat.Builder(context,

PRIMARY\_CHANNEL\_ID)

```
.setSmallIcon(R.drawable.ic_stand_up)
.setContentTitle("Stand Up Alert")
.setContentText("You should stand up and walk around now!")
.setContentIntent(contentPendingIntent)
.setPriority(NotificationCompat.PRIORITY_HIGH)
.setAutoCancel(true)
.setDefaults(NotificationCompat.DEFAULT_ALL)
```



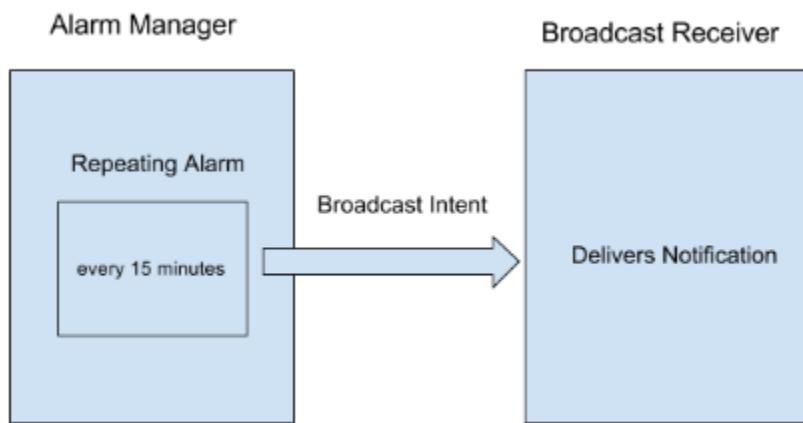
### Task 3: Tạo báo thức lặp lại

Bây giờ ứng dụng của bạn có thể gửi thông báo, đã đến lúc triển khai phần chính của ứng dụng: AlarmManager. Lớp này sẽ gửi lời nhắc đúng dậy theo chu kỳ.

AlarmManager có nhiều loại báo thức được tích hợp sẵn, bao gồm báo thức một lần, báo thức lặp lại, báo thức chính xác và không chính xác. Để tìm hiểu thêm về các loại báo thức khác nhau, hãy xem phần Lên lịch báo thức lặp lại.

Giống như thông báo, AlarmManager sử dụng một PendingIntent để gửi thông điệp với các tùy chọn được chỉ định. Vì vậy, AlarmManager có thể gửi Intent ngay cả khi ứng dụng không còn chạy.

Một broadcast receiver sẽ nhận Intent từ hệ thống và hiển thị thông báo.



Báo thức sẽ không kích hoạt khi thiết bị ở chế độ Doze (chế độ nhàn rỗi). Thay vào đó, báo thức sẽ bị hoãn lại cho đến khi thiết bị thoát khỏi chế độ Doze.

Để đảm bảo báo thức được thực thi, bạn có thể sử dụng setAndAllowWhileIdle() hoặc setExactAndAllowWhileIdle(). Bạn cũng có thể sử dụng API WorkManager, được thiết kế để thực hiện các công việc nền một lần hoặc theo chu kỳ. Để biết thêm chi tiết, hãy xem phần Lên lịch tác vụ với WorkManager.

AlarmManager có thể kích hoạt các sự kiện một lần hoặc lặp lại ngay cả khi ứng dụng không chạy. Đối với báo thức đồng hồ thời gian thực (RTC), hãy lên lịch sự kiện bằng System.currentTimeMillis().

### 3.1 Tạo broadcast receiver

Tạo một broadcast receiver để nhận broadcast intents từ AlarmManager và phản hồi phù hợp:

Các bước thực hiện:

1, Trong Android Studio, chọn File > New > Other > Broadcast Receiver.

2, Đặt tên lớp là AlarmReceiver.

3, Đảm bảo rằng hộp kiểm Exported bị bỏ chọn để các ứng dụng khác không thể gọi broadcast receiver này.

```
private NotificationManager mNotificationManager;
```

```
private static final int NOTIFICATION_ID = 0;
// Notification channel ID.

private static final String PRIMARY_CHANNEL_ID =
"primary_notification_channel";
```

4, Khởi tạo biến mNotificationManager ở đầu phương thức onReceive().

Bạn cần gọi getSystemService() từ context được truyền vào.

@Override

```
public void onReceive(Context context, Intent intent) {
mNotificationManager = (NotificationManager)
```

### 3.2 Thiết lập broadcast PendingIntent

AlarmManager chịu trách nhiệm gửi PendingIntent theo khoảng thời gian đã định. PendingIntent này sẽ gửi một Intent để thông báo cho ứng dụng biết đã đến lúc cập nhật thời gian còn lại trong thông báo.

Thực hiện các bước sau trong MainActivity.java, bên trong onCreate():

1 Tạo một Intent có tên notifyIntent.

- Truyền vào context và lớp AlarmReceiver.

2, Tạo notify PendingIntent bằng cách sử dụng context, biến NOTIFICATION\_ID, notifyIntent mới và cờ FLAG\_UPDATE\_CURRENT.

### 3.3 thiết lập lại báo thức

Bây giờ, bạn sẽ sử dụng AlarmManager để gửi broadcast sau mỗi 15 phút.

Loại báo thức phù hợp trong trường hợp này là báo thức lặp lại không chính xác, sử dụng thời gian đã trôi qua và đánh thức thiết bị nếu nó đang ở chế độ ngủ. Đồng hồ thời gian thực (RTC) không cần thiết vì bạn chỉ cần gửi thông báo mỗi 15 phút, không phải vào một thời điểm cụ thể trong ngày.

1, Khởi tạo AlarmManager trong onCreate() bằng cách gọi getSystemService().

```
AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
```

#### Thực hiện các bước sau:

1. Xóa lời gọi **deliverNotification()** khỏi phương thức **onCheckedChanged()**.
2. Trong phương thức **onCheckedChanged()**, **gọi setInexactRepeating()** trên **instance của AlarmManager** bên trong **câu lệnh if** (khi báo thức được bật).

#### Giải thích về setInexactRepeating()

- Lý do sử dụng **setInexactRepeating()**:

- Giúp tối ưu tài nguyên hơn so với báo thức chính xác.

- Cho phép hệ thống nhóm các báo thức từ nhiều ứng dụng khác nhau lại với nhau để tiết kiệm pin.
- Việc lệch một chút so với đúng khoảng thời gian 15 phút là chấp nhận được.

```
long repeatInterval = AlarmManager.INTERVAL_FIFTEEN_MINUTES;
```

```
long triggerTime = SystemClock.elapsedRealtime()
```

```
+ repeatInterval;
```

```
//If the Toggle is turned on, set the repeating alarm with a 15 minute
interval
```

```
if (alarmManager != null) {
```

```
alarmManager.setInexactRepeating
(AlarmManager.ELAPSED_REALTIME_WAKEUP,
triggerTime, repeatInterval, notifyPendingIntent);
```

Giữ lại lời gọi cancelAll() trên NotificationManager

- Khi người dùng tắt báo thức, mọi thông báo hiện có cần được xóa.
- Vì vậy, hãy giữ nguyên lời gọi mNotificationManager.cancelAll() khi tắt báo thức.

Cách kiểm tra chức năng báo thức mà không cần đợi 15 phút:

1. Chạy ứng dụng.
2. Nếu không muốn chờ 15 phút, có thể:
  - Thay đổi thời gian kích hoạt thành SystemClock.elapsedRealtime() để thông báo xuất hiện ngay lập tức.
  - Giảm khoảng thời gian lặp lại xuống một giá trị nhỏ hơn để kiểm tra xem báo thức có lặp lại đúng không.

### 3.4 Kiểm tra trạng thái báo thức

Để theo dõi trạng thái của báo thức, bạn cần một biến boolean có giá trị true nếu báo thức tồn tại và false nếu không. Để thiết lập giá trị cho biến boolean này, bạn có thể gọi PendingIntent.getBroadcast() với cờ FLAG\_NO\_CREATE. Nếu một PendingIntent tồn tại, phương thức sẽ trả về PendingIntent đó; nếu không, phương thức sẽ trả về null.

Hãy triển khai các bước sau trong MainActivity.java:

1, Tạo một biến boolean có giá trị true nếu PendingIntent không phải null, và false nếu ngược lại. Sử dụng biến boolean này để thiết lập trạng thái của ToggleButton khi ứng dụng khởi động. Đoạn mã này phải được đặt trước khi PendingIntent được tạo (nếu không, nó sẽ luôn trả về true).

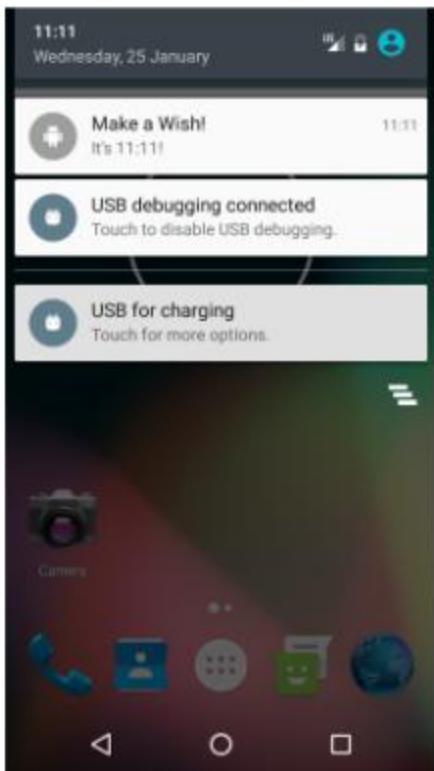
**Code challenge:** Lớp AlarmManager cũng xử lý loại báo thức thông thường, loại báo thức đánh thức bạn vào buổi sáng. Trên các thiết bị chạy API 21 trở lên, bạn có thể lấy thông tin về báo thức tiếp theo bằng cách gọi getNextAlarmClock() trên AlarmManager.

## Tóm tắt

- AlarmManager cho phép bạn lén lịch các tác vụ dựa trên đồng hồ thời gian thực hoặc thời gian đã trôi qua kể từ khi thiết bị khởi động.
- AlarmManager cung cấp nhiều loại báo thức khác nhau, bao gồm báo thức một lần và báo thức định kỳ.
- Báo thức sẽ không kích hoạt khi thiết bị ở chế độ Doze (nhàn rỗi). Các báo thức đã lén lịch sẽ bị hoãn lại cho đến khi thiết bị thoát khỏi chế độ Doze.
- Nếu bạn cần thực hiện tác vụ ngay cả khi thiết bị ở chế độ nhàn rỗi, bạn có thể sử dụng setAndAllowWhileIdle() hoặc setExactAndAllowWhileIdle(). Bạn cũng có thể sử dụng API WorkManager, được thiết kế để thực hiện công việc nền một lần hoặc định kỳ. Để biết thêm thông tin, hãy xem **Lên lịch tác vụ với WorkManager**.
- Bất cứ khi nào có thể, hãy sử dụng phiên bản AlarmManager có thời gian không chính xác. Việc này giúp giảm tải khi nhiều thiết bị hoặc nhiều ứng dụng thực hiện tác vụ cùng lúc.
- AlarmManager sử dụng PendingIntent để thực hiện các thao tác của nó. Bạn có thể lén lịch cho broadcast, service và activity bằng cách sử dụng PendingIntent phù hợp.

## Homework

Xây dựng và chạy app:



## 2.3:JobScheduler

### Giới thiệu

Bạn đã thấy rằng bạn có thể sử dụng lớp AlarmManager để kích hoạt sự kiện dựa trên đồng hồ thời gian thực hoặc dựa trên thời gian đã trôi qua kể từ khi thiết bị khởi động. Tuy nhiên, hầu hết các tác vụ không yêu cầu thời gian chính xác mà nên được lên lịch dựa trên sự kết hợp giữa yêu cầu của hệ thống và người dùng

Để bảo vệ dữ liệu của người dùng và tài nguyên hệ thống, một ứng dụng tin tức có thể đợi cho đến khi thiết bị đang sạc và kết nối Wi-Fi mới cập nhật tin tức.

Lớp JobScheduler cho phép bạn thiết lập các điều kiện hoặc tham số để quyết định khi nào nên chạy tác vụ. Dựa trên những điều kiện này, JobScheduler sẽ tính toán thời điểm tốt nhất để thực thi công việc. Ví dụ, các tham số của công việc có thể bao gồm:

- Duy trì công việc ngay cả sau khi thiết bị khởi động lại
- Kiểm tra xem thiết bị có đang cắm sạc không
- Kiểm tra xem thiết bị có đang ở trạng thái nhàn rỗi không

Tác vụ cần thực hiện được triển khai dưới dạng một lớp con của JobService và sẽ chạy theo các tham số đã thiết lập.

Lưu ý rằng JobScheduler chỉ khả dụng trên các thiết bị chạy API 21 trở lên và hiện không có trong thư viện hỗ trợ. Để đảm bảo khả năng tương thích ngược, bạn có thể sử dụng WorkManager. API WorkManager cho phép bạn lặp lịch các tác vụ nền cần được hoàn thành, ngay cả khi tiến trình của ứng dụng không còn hoạt động. Đối với các thiết bị chạy API 14 trở lên, bao gồm cả các thiết bị không có Google Play Services, WorkManager cung cấp các khả năng tương tự như JobScheduler.

Trong bài thực hành này, bạn sẽ tạo một ứng dụng lặp lịch thông báo. Thông báo sẽ được gửi khi các tham số do người dùng đặt ra được đáp ứng và yêu cầu của hệ thống được thỏa mãn.

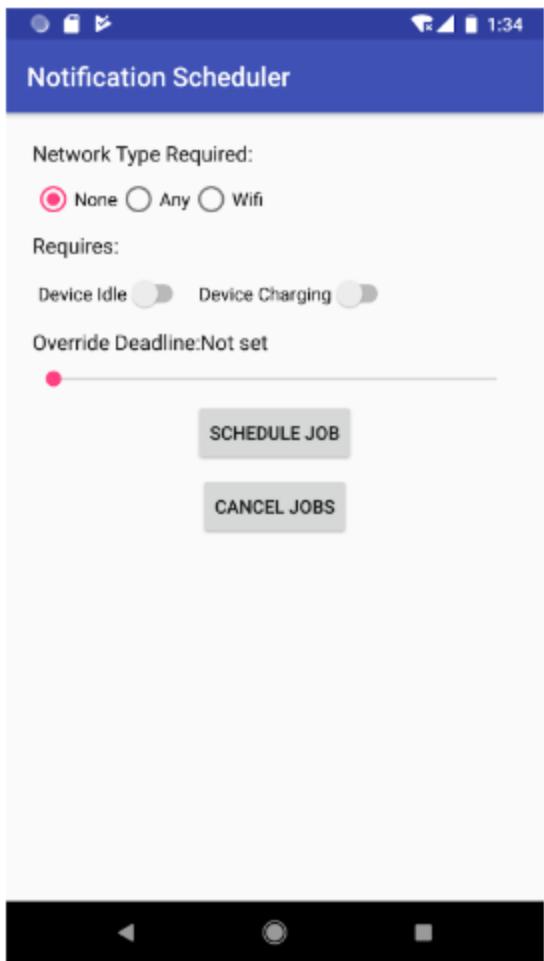
## Những gì bạn cần biết trước

Bạn nên có khả năng:

- Tạo một ứng dụng gửi thông báo.
- Lấy giá trị số nguyên từ một Spinner view.
- Sử dụng Switch view để nhận dữ liệu từ người dùng.
- Tạo PendingIntent.

## App overview

Bạn sẽ tạo một ứng dụng có tên Notification Scheduler. Ứng dụng của bạn sẽ minh họa cách sử dụng JobScheduler bằng cách cho phép người dùng chọn các điều kiện ràng buộc và lặp lịch một công việc. Khi công việc đó được thực thi, ứng dụng sẽ gửi một thông báo. (Trong ứng dụng này, thông báo chính là "công việc" cần thực hiện.)



### Task 1: Triển khai JobService

Trước tiên, hãy tạo một Service để chạy vào thời điểm được xác định dựa trên các điều kiện đã thiết lập. Hệ thống sẽ tự động thực thi JobService, và bạn chỉ cần triển khai hai phương thức:

Về phương thức onStartJob()

- Được gọi khi hệ thống xác định rằng công việc của bạn cần được thực hiện. Trong phương thức này, bạn triển khai công việc cần làm.
- Trả về một giá trị boolean:
  - Nếu true, công việc sẽ tiếp tục trên một luồng khác, và ứng dụng của bạn phải gọi jobFinished() trong luồng đó để thông báo rằng công việc đã hoàn tất.
  - Nếu false, hệ thống hiểu rằng công việc đã hoàn tất ngay sau khi onStartJob() kết thúc, và sẽ tự động gọi jobFinished().

Lưu ý:

Phương thức onStartJob() được thực thi trên main thread, vì vậy bất kỳ tác vụ nào kéo dài đều phải

chuyển sang một luồng khác. Tuy nhiên, trong ứng dụng này, bạn chỉ cần gửi một thông báo (notification), nên có thể thực hiện trực tiếp trên main thread một cách an toàn.

Về phương thức onStopJob()

- Nếu các điều kiện được thiết lập trong JobInfo không còn được đáp ứng, công việc phải dừng lại và hệ thống sẽ gọi onStopJob().
- Phương thức onStopJob() trả về một giá trị boolean để xác định hành động tiếp theo nếu công việc chưa hoàn thành:
  - Nếu true, công việc sẽ được lặp lại.
  - Nếu false, công việc sẽ bị hủy.

### 1.1 Tạo dự án và lớp NotificationJobService

Đảm bảo rằng minimum SDK của dự án là API 21 (Lollipop). Trước API 21, JobScheduler không hoạt động do thiếu các API cần thiết.

```
<service
```

```
 android:name=".NotificationJobService"
 android:permission="android.permission.BIND_JOB_SERVICE"/>
```

Creates a Notification channel, for OREO and higher.

```
*/
```

```
public void createNotificationChannel() {
 // Define notification manager object.

 mNotifyManager =
 (NotificationManager)
 getSystemService(NOTIFICATION_SERVICE);

 // Notification channels are only available in OREO and higher.

 // So, add a check on SDK version.

 if (android.os.Build.VERSION.SDK_INT >=
 android.os.Build.VERSION_CODES.O) {

 // Create the NotificationChannel with all the parameters.

 NotificationChannel notificationChannel = new
 NotificationChannel
 (PRIMARY_CHANNEL_ID,
 "Job Service notification",
```

```
NotificationManager.IMPORTANCE_HIGH);
notificationChannel.enableLights(true);
notificationChannel.setLightColor(Color.RED);
notificationChannel.enableVibration(true);
notificationChannel.setDescription
("Notifications from Job Service");
mNotifyManager.createNotificationChannel(notificationChannel);
```

### Triển khai điều kiện chạy công việc (JobInfo)

Sau khi đã tạo JobService, bước tiếp là xác định các tiêu chí để chạy công việc. Để làm điều này, bạn sử dụng JobInfo. Bạn sẽ tạo một nhóm điều kiện dựa trên loại kết nối mạng và trạng thái thiết bị.

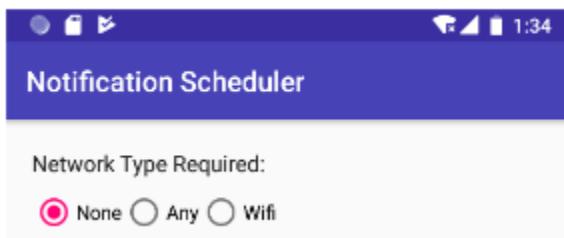
---

#### 2.1 Triển khai điều kiện mạng (network constraint)

Một trong những điều kiện có thể đặt cho JobService là trạng thái kết nối mạng của thiết bị. Bạn có thể giới hạn JobService để chỉ thực thi khi đáp ứng một số điều kiện mạng nhất định. Các tùy chọn gồm:

1. NETWORK\_TYPE\_NONE
  - o Công việc sẽ chạy dù có hoặc không có kết nối mạng.
  - o Đây là giá trị mặc định nếu không có điều kiện mạng nào được đặt.
2. NETWORK\_TYPE\_ANY
  - o Công việc sẽ chạy khi có bất kỳ mạng nào khả dụng (Wi-Fi hoặc mạng di động).
3. NETWORK\_TYPE\_UNMETERED
  - o Công việc sẽ chỉ chạy khi thiết bị được kết nối với Wi-Fi không phải điểm phát sóng (HotSpot).

### Tạo layout cho app



## 2.2 Kiểm tra điều kiện

JobScheduler yêu cầu ít nhất một điều kiện được đặt. Nếu không có điều kiện nào, công việc sẽ không thể chạy. Vì vậy, bạn cần tạo một biến boolean để kiểm tra xem có điều kiện nào được thiết lập hay chưa.

Bạn sẽ thực hiện các bước sau trong MainActivity.java, bên trong scheduleJob():

1 Tạo biến constraintSet để kiểm tra điều kiện

- Biến này sẽ true nếu người dùng chọn một loại kết nối mạng khác với mặc định (NETWORK\_TYPE\_NONE).

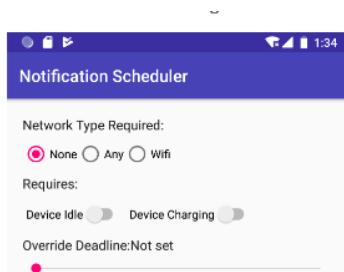
2, Cập nhật biến constraintSet khi thêm các tùy chọn khác

- Khi bạn thêm các điều kiện khác (ví dụ: thiết bị đang sạc, nhàn rỗi...), cần cập nhật constraintSet để đảm bảo ít nhất một điều kiện được đặt.

```
boolean constraintSet = selectedNetworkOption != JobInfo.NETWORK_TYPE_NONE;
```

## 2.4 Triển khai điều kiện giới hạn thời gian (Override-Deadline)

Cho đến thời điểm này, không có cách nào để biết chính xác khi nào hệ thống sẽ thực thi công việc (JobService). Hệ thống quản lý tài nguyên hiệu quả, điều này có thể làm trì hoãn công việc tùy theo trạng thái của thiết bị.



Cho đến thời điểm này, không có cách nào để biết chính xác khi nào hệ thống sẽ thực thi công việc (JobService). Hệ thống quản lý tài nguyên hiệu quả, điều này có thể làm trì hoãn công việc tùy theo trạng thái của thiết bị.

JobScheduler cung cấp một tùy chọn để đặt thời gian giới hạn tuyệt đối (override deadline). Khi đến hạn, hệ thống sẽ thực thi công việc bất kể các điều kiện khác.

## Chương 4: LUU DỮ LIỆU NGƯỜI DÙNG

### Bài 1: Tùy chọn và cài đặt

#### 1.1 Shared preferences

##### Bài 9.1: Tùy chọn được chia sẻ Giới thiệu

Tùy chọn được chia sẻ cho phép bạn lưu trữ một lượng nhỏ dữ liệu nguyên thủy dưới dạng cặp khóa/giá trị trong một tệp trên thiết bị. Để có được một tay cầm cho một tệp tùy chọn, và để đọc, ghi và quản lý dữ liệu tùy chọn, hãy sử dụng lớp SharedPreferences. Khung Android tự quản lý tệp tùy chọn được chia sẻ. Tất cả các thành phần của ứng dụng của bạn có thể truy cập tệp nhưng các ứng dụng khác không thể truy cập được. Dữ liệu bạn lưu vào các tùy chọn được chia sẻ khác với dữ liệu ở trạng thái hoạt động đã lưu, mà bạn đã tìm hiểu trong chương trước:

- Dữ liệu ở trạng thái phiên bản hoạt động đã lưu được giữ lại trên các phiên bản hoạt động trong cùng một phiên người dùng.
- Tùy chọn được chia sẻ vẫn tồn tại trên các phiên người dùng. Tùy chọn được chia sẻ vẫn tồn tại ngay cả khi ứng dụng của bạn dừng và khởi động lại hoặc nếu thiết bị khởi động lại. Chỉ sử dụng tùy chọn được chia sẻ khi bạn cần lưu một lượng nhỏ dữ liệu dưới dạng các cặp khóa/giá trị đơn giản. Để quản lý lượng dữ liệu ứng dụng liên tục lớn hơn, hãy sử dụng phương pháp lưu trữ như thư viện Room hoặc cơ sở dữ liệu SQL. Những điều bạn nên biết Bạn nên quen thuộc:
  - Tạo, xây dựng và chạy ứng dụng trong Android Studio.
  - Thiết kế bố cục với các nút và chế độ xem văn bản.
  - Sử dụng phong cách và chủ đề.
  - Lưu và khôi phục trạng thái phiên bản hoạt động.

Những gì bạn sẽ học Bạn sẽ học cách:

- Xác định những sở thích được chia sẻ là gì.
- Tạo tệp tùy chọn được chia sẻ cho ứng dụng của bạn.
- Lưu dữ liệu vào các tùy chọn được chia sẻ và đọc lại các tùy chọn đó.
- Xóa dữ liệu trong tùy chọn được chia sẻ.

Bạn sẽ làm gì

- Cập nhật ứng dụng để ứng dụng có thể lưu, truy xuất và đặt lại tùy chọn được chia sẻ.

Tổng quan về ứng dụng

Ứng dụng HelloSharedPrefs là một biến thể khác của ứng dụng HelloToast mà bạn đã tạo trong Bài 1. Nó bao gồm các nút để tăng số, thay đổi màu nền và đặt lại cả số và màu về mặc định của chúng. Ứng dụng cũng sử dụng các chủ đề và kiểu để xác định các nút.

Bạn bắt đầu với ứng dụng khởi động và thêm tùy chọn được chia sẻ vào mã hoạt động chính. Bạn cũng thêm nút đặt lại để đặt cả số lượng và màu nền thành mặc định, đồng thời xóa tệp tùy chọn. Nhiệm vụ 1: Khám phá HelloSharedPrefs Dự án ứng dụng khởi động hoàn chỉnh cho thực tế này có sẵn tại

HelloSharedPrefs-Starter . Trong tác vụ này, bạn tải dự án vào Android Studio và khám phá một số tính năng chính của ứng dụng.

### 1.1 Mở và chạy dự án HelloSharedPrefs

1. Tải xuống ứng dụng HelloSharedPrefs-Starter và giải nén tệp.
2. Mở dự án trong Android Studio, xây dựng và chạy ứng dụng. Hãy thử những điều sau:
  - Nhấp vào nút Đếm để tăng số trong chế độ xem văn bản chính.
  - Nhấp vào bất kỳ nút màu nào để thay đổi màu nền của chế độ xem văn bản chính. ○ Xoay thiết bị và lưu ý rằng cả màu nền và số lượng đều được giữ nguyên.
  - Nhấp vào nút Đặt lại để đặt màu và đếm ngược về mặc định.
3. Buộc thoát ứng dụng bằng một trong các phương pháp sau:
  - Trong Android Studio, chọn Run > Stop 'app' hoặc nhấp vào Biểu tượng Stop trên thanh công cụ.
  - Trên thiết bị, nhấn nút Gần đây (nút vuông ở góc dưới bên phải). Vuốt thẻ ứng dụng HelloSharedPrefs để thoát ứng dụng hoặc bấm vào X ở góc phải của thẻ. Nếu bạn thoát ứng dụng theo cách này, hãy đợi vài giây trước khi khởi động lại để hệ thống có thể dọn dẹp.
4. Chạy lại ứng dụng. Ứng dụng khởi động lại với giao diện mặc định—số lượng là 0 và màu nền là màu xám.

### 1.2 Khám phá mã hoạt động

1. Mở MainActivity .
2. Kiểm tra mã và lưu ý những điều sau:
  - Số đếm (mCount) được định nghĩa là một số nguyên. Phương thức countUp() onClick tăng giá trị này và cập nhật TextView chính.
  - Màu (mColor) cũng là một số nguyên ban đầu được định nghĩa là màu xám trong tệp tài nguyên colors.xml là default\_background .
  - Phương thức changeBackground() onClick lấy màu nền của nút đã được nhấp, sau đó đặt chế độ xem văn bản chính thành màu đó.
  - Cả số nguyên mCount và mColor đều được lưu vào gói trạng thái phiên bản trong onSaveInstanceState() và được khôi phục trong onCreate() . Các khóa gói cho số lượng và màu sắc được xác định bởi các biến riêng ( COUNT\_KEY ) và ( COLOR\_KEY ).

Nhiệm vụ 2: Lưu và khôi phục dữ liệu vào tệp tùy chọn được chia sẻ

Trong tác vụ này, bạn lưu trạng thái của ứng dụng vào tệp tùy chọn được chia sẻ và đọc lại dữ liệu đó khi ứng dụng được khởi động lại. Vì dữ liệu trạng thái mà bạn lưu vào các tệp tùy chọn được chia sẻ (số lượng và màu hiện tại) là cùng dữ liệu mà bạn giữ lại trong trạng thái phiên bản, bạn không phải thực hiện hai lần. Bạn có thể thay thế hoàn toàn trạng thái phiên bản bằng trạng thái tùy chọn được chia sẻ.

#### 1.1 Khởi tạo các tùy chọn

Thêm các biến thành viên vào lớp MainActivity để giữ tên của tệp tùy chọn được chia sẻ và tham chiếu đến đối tượng SharedPreferences.

In the onCreate() method, initialize the shared preferences. Insert this code before the if statement:

Lưu ý: Các phiên bản Android cũ hơn có các chế độ khác cho phép bạn tạo tệp tùy chọn được chia sẻ có thể đọc được trên thế giới hoặc có thể ghi trên thế giới. Các chế độ này không được dùng nữa trong API 17 và hiện không được khuyến khích vì lý do bảo mật. Nếu bạn cần chia sẻ dữ liệu với các ứng dụng khác, hãy cân nhắc sử dụng URI nội dung do FileProvider cung cấp

### 1.2 Lưu tùy chọn trong onPause()

Lưu tùy chọn giống như lưu trạng thái phiên bản -- cả hai thao tác đều đặt dữ liệu sang một đối tượng Bundle dưới dạng cặp khóa/giá trị. Tuy nhiên, đối với các tùy chọn được chia sẻ, bạn lưu dữ liệu đó trong lệnh gọi lại vòng đời onPause() và bạn cần một đối tượng trình soạn thảo được chia sẻ (SharedPreferences.Editor) để ghi vào đối tượng tùy chọn được chia sẻ.

1. Thêm phương thức vòng đời onPause() vào MainActivity .
2. Trong onPause(), lấy một trình chỉnh sửa cho đối tượng SharedPreferences:  
Cần có trình chỉnh sửa tùy chọn được chia sẻ để ghi vào đối tượng tùy chọn được chia sẻ. Thêm dòng này vào onPause() sau cuộc gọi đến super.onPause().
3. Sử dụng phương thức putInt() để đặt cả số nguyên mCount và mColor vào các tùy chọn được chia sẻ với các khóa thích hợp:

Lớp SharedPreferences.Editor bao gồm nhiều phương thức "put" cho các kiểu dữ liệu khác nhau, bao gồm.putInt() và putString() .

4. Gọi apply() để lưu các tùy chọn:

Phương thức apply() lưu các tùy chọn không đồng bộ, ngoài luồng giao diện người dùng. Trình soạn thảo tùy chọn được chia sẻ cũng có một phương thức commit() để lưu đồng bộ các tùy chọn. Phương thức commit() không được khuyến khích vì nó có thể chặn các hoạt động khác.

1. Xóa toàn bộ phương thức onSaveInstanceState(). Vì trạng thái phiên bản hoạt động chứa dữ liệu giống như tùy chọn được chia sẻ, bạn có thể thay thế hoàn toàn trạng thái phiên bản.

### 1.3 Khôi phục tùy chọn trong onCreate()

Như với trạng thái phiên bản, ứng dụng của bạn đọc mọi tùy chọn được chia sẻ đã lưu trong phương thức onCreate(). Một lần nữa, vì các tùy chọn được chia sẻ chứa cùng dữ liệu với trạng thái phiên bản, chúng ta cũng có thể thay thế trạng thái bằng các tùy chọn ở đây. Mỗi khi onCreate() được gọi -- khi ứng dụng khởi động, khi thay đổi cấu hình -- các tùy chọn được chia sẻ sẽ được sử dụng để khôi phục trạng thái của chế độ xem.

1. Xác định vị trí một phần của phương thức onCreate() kiểm tra xem đối số savedInstanceState có rỗng hay không và khôi phục trạng thái phiên bản:
2. Xóa toàn bộ khối đó.
3. Trong phương thức onCreate(), ở cùng vị trí có mã trạng thái phiên bản, lấy số lượng từ các tùy chọn bằng phím COUNT\_KEY và gán nó cho biến mCount.  
Khi đọc dữ liệu từ các tùy chọn, bạn không cần phải có trình chỉnh sửa tùy chọn được chia sẻ. Sử dụng bất kỳ phương thức "get" nào trên đối tượng tùy chọn được chia sẻ (chẳng hạn như getInt() hoặc getString()) để truy xuất dữ liệu tùy chọn. Lưu ý rằng phương thức getInt()

- nhận hai đối số: một cho khóa và một cho giá trị mặc định nếu không thể tìm thấy khóa. Trong trường hợp này, giá trị mặc định là 0, giống với giá trị ban đầu của mCount .
4. Cập nhật giá trị của TextView chính với số lượng mới.
  5. Lấy màu từ các tùy chọn bằng phím COLOR\_KEY và gán nó cho biến mColor. Như trước đây, đối số thứ hai cho getInt() là giá trị mặc định để sử dụng trong trường hợp khóa không tồn tại trong các tùy chọn được chia sẻ. Trong trường hợp này, bạn chỉ có thể sử dụng lại giá trị của mColor, vừa được khởi tạo thành nền mặc định ở phía trên trong phương thức.
  6. Cập nhật màu nền của chế độ xem văn bản chính.
  7. Chạy ứng dụng. Nhấp vào nút Đếm và thay đổi màu nền để cập nhật trạng thái phiên bản và tùy chọn.
  8. Xoay thiết bị hoặc trình giả lập để xác minh rằng số lượng và màu sắc đã được lưu sau các thay đổi cấu hình.
  9. Buộc thoát ứng dụng bằng một trong các phương pháp sau:
    - Trong Android Studio, chọn Chạy > Dừng 'ứng dụng'.
    - Trên thiết bị, nhấn nút Gần đây (nút vuông ở góc dưới bên phải). Vuốt thẻ ứng dụng HelloSharedPreferences để thoát ứng dụng hoặc bấm vào X ở góc phải của thẻ.
  10. Chạy lại ứng dụng. Ứng dụng khởi động lại và tải các tùy chọn, duy trì trạng thái.
- Mã giải pháp cho phương thức MainActivity onCreate():

#### 1.4 Đặt lại tùy chọn trong trình xử lý nhấp chuột reset()

Nút đặt lại trong ứng dụng khởi động đặt lại cả số lượng và màu sắc cho hoạt động về giá trị mặc định. Bởi vì các tùy chọn giữ trạng thái của hoạt động, điều quan trọng là phải xóa các tùy chọn cùng một lúc.

1. Trong phương thức reset() onClick, sau khi màu sắc và số lượng được đặt lại, hãy lấy một trình chỉnh sửa cho đối tượng SharedPreferences:
2. Xóa tất cả các tùy chọn được chia sẻ:
  1. Áp dụng các thay đổi:

Thử thách mã hóa Lưu ý: Tất cả các thử thách mã hóa là tùy chọn và không phải là điều kiện tiên quyết cho các bài học sau. Thử thách: Sửa đổi ứng dụng HelloSharedPreferences để thay vì tự động lưu trạng thái vào tệp tùy chọn, hãy thêm một hoạt động thứ hai để thay đổi, đặt lại và lưu các tùy chọn đó. Thêm một nút vào ứng dụng có tên Cài đặt để khởi chạy hoạt động đó. Bao gồm các nút chuyển đổi và con quay để sửa đổi tùy chọn và các nút Lưu và Đặt lại để lưu và xóa các tùy chọn.

Tóm tắt

- Lớp SharedPreferences cho phép ứng dụng lưu trữ một lượng nhỏ dữ liệu nguyên thủy dưới dạng cặp khóa-giá trị.
- Tùy chọn được chia sẻ tồn tại trên các phiên người dùng khác nhau của cùng một ứng dụng.
- Để ghi vào các tùy chọn được chia sẻ, hãy lấy đối tượng SharedPreferences.Editor.
- Sử dụng các phương thức "put" khác nhau trong đối tượng SharedPreferences.Editor, chẳng hạn như putInt() hoặc putString(), để đưa dữ liệu vào các tùy chọn được chia sẻ bằng khóa và giá trị.
- Sử dụng các phương thức "get" khác nhau trong đối tượng SharedPreferences, chẳng hạn như getInt() hoặc getString() , để lấy dữ liệu từ các tùy chọn được chia sẻ bằng khóa.

- Sử dụng phương thức clear() trong đối tượng SharedPreferences.Editor để xóa tất cả dữ liệu được lưu trữ trong tùy chọn.
- Sử dụng phương thức apply() trong đối tượng SharedPreferences.Editor để lưu các thay đổi vào tệp tùy chọn. Khái niệm liên quan Tài liệu khái niệm liên quan có trong 9.0: Lưu trữ dữ liệu và 9.1: Tùy chọn được chia sẻ .

Tìm hiểu thêm tài liệu dành cho nhà phát triển Android:

- Tổng quan về lưu trữ dữ liệu và tệp
- Lưu trữ dữ liệu khóa-giá trị
- SharedPreferences
- SharedPreferences.Editor

Stack Overflow:

- Cách sử dụng SharedPreferences trong Android để lưu trữ, tìm nạp và chỉnh sửa giá trị
- onSavedInstanceState so với SharedPreferences

Bài tập về nhà

Xây dựng và chạy ứng dụng Mở ứng dụng ScoreKeeper mà bạn đã tạo trong các nguyên tắc cơ bản về Android 5.1: Bài học có thể vẽ, kiểu và chủ đề.

1. Thay thế trạng thái phiên bản đã lưu bằng các tùy chọn được chia sẻ cho từng điểm.
2. Để kiểm tra ứng dụng, hãy xoay thiết bị để đảm bảo rằng các thay đổi cấu hình đọc các tùy chọn đã lưu và cập nhật giao diện người dùng.
3. Dùng ứng dụng và khởi động lại ứng dụng để đảm bảo rằng các tùy chọn đã được lưu.
4. Thêm nút Đặt lại để đặt lại giá trị điểm số về 0 và xóa các tùy chọn được chia sẻ. Trả lời các câu hỏi này

Câu hỏi 1: Bạn lưu trạng thái ứng dụng vào các tùy chọn được chia sẻ trong phương pháp vòng đời nào?

Câu hỏi 2: Bạn khôi phục trạng thái ứng dụng trong phương pháp lifecycle nào? Câu hỏi 3: Bạn có thể nghĩ ra một trường hợp mà việc có cả sở thích chung và trạng thái thực thể là hợp lý không?

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn dành cho người chấm điểm Kiểm tra xem ứng dụng có các tính năng sau không:

- Ứng dụng giữ lại điểm số khi xoay thiết bị.
- Ứng dụng giữ lại điểm số hiện tại sau khi ứng dụng dừng và khởi động lại.
- Ứng dụng lưu điểm số hiện tại vào các tùy chọn được chia sẻ trong phương thức onPause () .
- Ứng dụng khôi phục các tùy chọn được chia sẻ trong phương thức onCreate(). • Ứng dụng hiển thị nút Đặt lại để đặt lại điểm số về 0. Đảm bảo rằng việc triển khai phương thức xử lý khi nhấp chuột cho nút Đặt lại thực hiện những việc sau:

- Đặt lại cả hai biến điểm thành 0.
- Cập nhật cả hai chế độ xem văn bản.
- Xóa các tùy chọn được chia sẻ.

## 1.2 Cài đặt ứng dụng

### Bài 9.2: Cài đặt ứng dụng

Giới thiệu Các ứng dụng thường bao gồm các cài đặt cho phép người dùng sửa đổi các tính năng và hành vi của ứng dụng. Ví dụ: một số ứng dụng cho phép người dùng đặt vị trí nhà, đơn vị mặc định cho các phép đo và các cài đặt khác áp dụng cho toàn bộ ứng dụng. Người dùng không truy cập cài đặt thường xuyên, bởi vì một khi người dùng thay đổi cài đặt, chẳng hạn như vị trí nhà, họ hiếm khi cần phải quay lại và thay đổi lại. Người dùng mong muốn điều hướng đến cài đặt ứng dụng bằng cách nhấp vào Cài đặt trong điều hướng bên, chẳng hạn như ngăn điều hướng như được hiển thị ở phía bên trái của hình bên dưới hoặc trong menu tùy chọn trong thanh ứng dụng, được hiển thị ở phía bên phải của hình bên dưới.

Trong hình trên:

1. Cài đặt trong điều hướng bên (ngăn điều hướng)
2. Cài đặt trong menu tùy chọn của thanh ứng dụng

Những gì bạn nên biết

Bạn sẽ có thể:

- Tạo một dự án Android Studio từ một mẫu và tạo bố cục chính.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị được kết nối.
- Tạo và chỉnh sửa các yếu tố giao diện người dùng bằng trình chỉnh sửa bố cục và mã XML.
- Trích xuất tài nguyên chuỗi và chỉnh sửa các giá trị chuỗi.
- Truy cập các phần tử giao diện người dùng từ mã của bạn bằng cách sử dụng findViewById().
- Xử lý một nút bấm.
- Hiển thị tin nhắn Toast.
- Thêm Hoạt động vào ứng dụng.
- Tạo menu tùy chọn trong thanh ứng dụng.
- Thêm và chỉnh sửa các mục menu trong menu tùy chọn.

- Sử dụng các phong cách và chủ đề trong một dự án.
- Sử dụng SharedPreferences .

Những gì bạn sẽ học Bạn sẽ học cách:

- Thêm một Fragment để quản lý cài đặt.
- Tạo tệp tài nguyên XML của cài đặt với các thuộc tính của chúng.
- Tạo điều hướng đến cài đặt Hoạt động.
- Đặt các giá trị mặc định của cài đặt.
- Đọc các giá trị cài đặt do người dùng thay đổi.
- Tùy chỉnh mẫu Hoạt động Cài đặt. Những gì bạn sẽ làm
- Tạo một ứng dụng bao gồm Cài đặt trong menu tùy chọn.
- Thêm công tắc bật tắt tùy chọn Cài đặt.
- Thêm mã để đặt giá trị mặc định cho cài đặt và truy cập giá trị cài đặt sau khi nó đã thay đổi.
- Sử dụng và tùy chỉnh mẫu Hoạt động cài đặt Android Studio. Tổng quan về ứng dụng: Android Studio cung cấp một phím tắt để thiết lập menu tùy chọn với Cài đặt .

Nếu bạn bắt đầu dự án Android Studio cho điện thoại hoặc máy tính bảng bằng mẫu Basic Activity, thì ứng dụng mới sẽ bao gồm Settings như minh họa bên dưới:

Bạn sẽ bắt đầu bằng cách tạo một ứng dụng có tên AppWithSettings bằng cách sử dụng mẫu Hoạt động cơ bản và bạn sẽ thêm một cài đặt Hoạt động cài đặt cung cấp một cài đặt công tắc chuyển đổi mà người dùng có thể bật hoặc tắt:

Bạn sẽ thêm mã để đọc cài đặt và thực hiện một hành động dựa trên giá trị của nó. Để đơn giản, hành động sẽ là hiển thị thông báo Toast với giá trị của cài đặt.

Trong nhiệm vụ thứ hai, bạn sẽ thêm mẫu Hoạt động cài đặt tiêu chuẩn do Android Studio cung cấp vào ứng dụng DroidCafeOptionsUp mà bạn đã tạo trong bài học trước. Mẫu Hoạt động Cài đặt được điều chỉnh với các cài đặt bạn có thể tùy chỉnh cho ứng dụng và cung cấp bộ lọc khác cho điện thoại và máy tính bảng:

- Điện thoại: Màn hình Cài đặt chính với liên kết tiêu đề cho từng nhóm cài đặt, chẳng hạn như Cài đặt chung cho cài đặt chung, như hình dưới đây.
- Máy tính bảng: Bộ lọc màn hình chính / chi tiết với liên kết tiêu đề cho từng nhóm ở phía bên trái (chính) và nhóm cài đặt ở phía bên phải (chi tiết), như thể hiện trong hình bên dưới.

Để tùy chỉnh mẫu, bạn sẽ thay đổi tiêu đề, tiêu đề cài đặt, mô tả cài đặt và giá trị cho cài đặt. Ứng dụng DroidCafeOptionsUp đã được tạo trong bài học trước từ mẫu Hoạt động cơ bản, cung cấp menu tùy chọn trong thanh ứng dụng để đặt tùy chọn Cài đặt. Bạn sẽ tùy chỉnh mẫu Hoạt động Cài đặt được cung cấp bằng cách thay đổi tiêu đề, mô tả, giá trị và giá trị mặc định của một cài đặt. Bạn sẽ thêm mã để đọc giá trị của cài đặt sau khi người dùng thay đổi và hiển thị giá trị đó.

Nhiệm vụ 1: Thêm cài đặt công tắc vào ứng dụng Trong tác vụ này, bạn thực hiện như sau:

- Tạo dự án mới dựa trên mẫu Hoạt động cơ bản, cung cấp menu tùy chọn.
  - Thêm công tắc bật tắt (SwitchPreference) với các thuộc tính trong tệp XML tùy chọn.
  - Thêm một hoạt động cho cài đặt và một mảnh cho một cài đặt cụ thể. Để duy trì khả năng tương thích với AppCompatActivity , bạn sử dụng PreferenceFragmentCompat thay vì PreferenceFragment . Bạn cũng thêm thư viện android.support.v7.preference .
  - Kết nối mục Cài đặt trong menu tùy chọn với hoạt động cài đặt.
- 1.1 Tạo dự án và thêm thư mục xml và tệp tài nguyên

1. Trong Android Studio, tạo một dự án mới với các thông số sau:

Attribute	Value
Application Name	AppWithSettings
Company Name	android.example.com (or your own domain)
Project location	Path to your directory of projects
Phone and Tablet Minimum SDK	API15: Android 4.0.3 IceCreamSandwich
Template	Basic Activity
Activity Name	MainActivity

Layout Name	activity_main
Title	MainActivity

- Chạy ứng dụng và nhấn vào biểu tượng tròn trong thanh ứng dụng để xem menu tùy chọn, như trong hình bên dưới. Mục duy nhất trong menu tùy chọn là Cài đặt .
- Bạn cần tạo một thư mục tài nguyên mới để chứa tệp XML chứa các cài đặt. Chọn thư mục res trong ngăn Project > Android và chọn File > New > Android Resource Directory . Hộp thoại Thư mục Tài nguyên Mới xuất hiện.
- Trong menu thả xuống Loại tài nguyên, chọn xml . Tên thư mục tự động thay đổi thành xml . Nhập vào OK .
- Thư mục xml xuất hiện trong ngăn Project > Android bên trong thư mục res. Chọn xml và chọn File > New > XML resource file (hoặc nhấp chuột phải vào xml và chọn New > XML resource file ).
- Nhập tên của tệp XML, tùy chọn , vào trường Tên tệp và nhập vào OK . Tệp preferences.xml xuất hiện bên trong thư mục xml và trình chỉnh sửa bối cảnh xuất hiện, như trong hình bên dưới.

Trong hình trên:

- Tệp preferences.xml bên trong thư mục xml.
  - Trình chỉnh sửa bối cảnh hiển thị nội dung preferences.xml.
- 1.2 Thêm tùy chọn và thuộc tính XML cho cài đặt
- Kéo SwitchPreference từ ngăn Bảng màu ở phía bên trái lên đầu bối cảnh, như trong hình bên dưới.

2. Thay đổi các giá trị trong Thuộc tính ngắn ở phía bên phải của trình chỉnh sửa bộ cục như sau và như thể hiện trong hình bên dưới:

- defaultValue: true
- khóa: example\_switch
- title: Tùy chọn cài đặt
- tóm tắt: Bật hoặc tắt tùy chọn này

3. Nhập vào tab Văn bản ở cuối trình chỉnh sửa bộ cục để xem mã XML:

4. Trích xuất tài nguyên chuỗi cho các giá trị thuộc tính android:title và android:summary thành @string/switch\_title và @string/switch\_summary .

Các thuộc tính XML cho tùy chọn là:

- android:defaultValue : Giá trị mặc định của cài đặt khi ứng dụng khởi động lần đầu tiên.
- android:title: Tiêu đề của cài đặt. Đối với SwitchPreference , tiêu đề xuất hiện ở bên trái của công tắc bật tắt.
- android: key: Khóa sử dụng để lưu trữ giá trị cài đặt. Mỗi chế độ cài đặt có một cặp khóa-giá trị tương ứng mà hệ thống sử dụng để lưu cài đặt trong tệp SharedPreferences mặc định cho các tùy chọn cài đặt của ứng dụng.
- android: tóm tắt: Tóm tắt văn bản xuất hiện bên dưới cài đặt.

1.3 Sử dụng SwitchPreferenceCompat Để sử dụng phiên bản PreferenceFragmentCompat của PreferenceFragment , bạn cũng phải sử dụng phiên bản Android.support.v7 của SwitchPreference ( SwitchPreferenceCompat ). 1. Trong ngăn Project > Android, hãy mở tệp build.gradle (Module: app) trong thư mục Gradle Scripts và thêm thông tin sau vào phần phụ thuộc:

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

Dòng SwitchPreferenceCompat ở trên có thể hiển thị biểu tượng bóng đèn màu vàng với cảnh báo, nhưng bạn có thể bỏ qua nó ngay bây giờ.

3. Mở tệp styles.xml trong thư mục giá trị và thêm mục preferenceTheme sau vào khai báo AppTheme:

Để sử dụng PreferenceFragmentCompat , bạn cũng phải khai báo preferenceTheme với kiểu PreferenceThemeOverlay cho giao diện ứng dụng.

1.4 Thêm Hoạt động cho cài đặt Để tạo Hoạt động cài đặt cung cấp giao diện người dùng cho cài đặt, hãy thêm Hoạt động trống vào ứng dụng. Làm theo các bước sau:

1. Chọn ứng dụng ở đầu ngăn Dự án > Android và chọn Hoạt động > mới > Hoạt động trống .
2. Đặt tên cho Activity SettingsActivity . Bỏ chọn tùy chọn Tạo tệp bộ cục (bạn không cần) và bỏ chọn tùy chọn Hoạt động trình khởi chạy.

3. Để tùy chọn Tương thích ngược (AppCompat) được chọn. Tên gói phải được đặt thành com.example.android. tên dự án .

4. Nhập vào Kết thúc.

#### 1.5 Thêm Fragment cho một cài đặt cụ thể

Fragment giống như một phần mô-đun của Activity —nó có vòng đời riêng và nhận các sự kiện đầu vào của riêng nó và bạn có thể thêm hoặc xóa Fragment trong khi Activity đang chạy. Bạn sử dụng một lớp con Fragment chuyên dụng để hiển thị danh sách các cài đặt. Cách tốt nhất là sử dụng Hoạt động thông thường lưu trữ PreferenceFragment hiển thị cài đặt ứng dụng. PreferenceFragment cung cấp kiến trúc linh hoạt hơn cho ứng dụng của bạn so với việc sử dụng Hoạt động cho các tùy chọn. Bạn sẽ sử dụng PreferenceFragmentCompat thay vì PreferenceFragment để duy trì khả năng tương thích với AppCompatActivity . Trong bước này, bạn sẽ thêm một Fragment trống cho một nhóm các cài đặt tương tự (không có bô cục, phương thức xuất xưởng hoặc callback giao diện) vào ứng dụng và mở rộng PreferenceFragmentCompat.

Làm theo các bước sau:

1. Chọn lại ứng dụng và chọn New > Fragment > Fragment (Trống) .
2. Đặt tên cho mảnh là SettingsFragment . Bỏ chọn Tạo bô cục XML? (bạn không cần).
3. Bỏ chọn các tùy chọn để bao gồm các phương thức nhà máy phân đoạn và lệnh gọi lại giao diện.
4. Bộ nguồn mục tiêu phải được đặt thành chính.
5. Nhập vào Kết thúc. Kết quả là định nghĩa lớp sau trong SettingsFragment
6. Chính sửa định nghĩa lớp của SettingsFragment để mở rộng PreferenceFragmentCompat

Khi bạn thay đổi định nghĩa lớp để nó khớp với định nghĩa hiển thị ở trên, một bóng đèn màu đỏ sẽ xuất hiện ở lề trái. Nhập vào bóng đèn màu đỏ và chọn Triển khai phương pháp , sau đó chọn onCreatePreferences . Android Studio tạo sơ khai onCreatePreferences() sau:

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

7. Xóa toàn bộ phương thức onCreateView() trong mảnh. Lý do tại sao về cơ bản bạn thay thế onCreateView() bằng onCreatePreferences() là vì bạn sẽ thêm SettingsFragment này vào SettingsActivity hiện có để hiển thị tùy chọn, thay vì hiển thị một màn hình Fragment riêng biệt. Việc thêm Fragment vào Activity hiện có giúp bạn dễ dàng thêm hoặc xóa Fragment trong khi Activity đang chạy. Mảnh tùy chọn được root tại PreferenceScreen bằng cách sử dụng rootKey . Bạn cũng có thể xóa hàm khởi tạo trống khỏi SettingsFragment một cách an toàn, vì Fragment không tự hiển thị:
8. Bạn cần liên kết với Fragment này tài nguyên cài đặt preferences.xml mà bạn đã tạo ở bước trước. Thêm vào sơ khai onCreatePreferences() mới tạo một lệnh gọi đến setPreferencesFromResource() truyền id của tệp XML (R.xml.preferences) và rootKey để xác định gốc tùy chọn trong PreferenceScreen

Phương thức onCreatePreferences() bây giờ sẽ trông như sau:

#### 1.6 Hiển thị Fragment trong SettingsActivity

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

1. Mở SettingsActivity .
2. Thêm mã sau vào cuối phương thức onCreate() để Fragment được hiển thị dưới dạng nội dung chính:  
Mã trên sử dụng mẫu điển hình để thêm một mảnh vào một hoạt động để phân đoạn xuất hiện dưới dạng nội dung chính của hoạt động:
  - Sử dụng getSupportFragmentManager() nếu lớp mở rộng Activity và Fragment mở rộng PreferenceFragment .
  - Sử dụng getSupportFragmentManager() nếu lớp mở rộng AppCompatActivity và Fragment mở rộng PreferenceFragmentCompat. Toàn bộ phương thức onCreate() trong SettingsActivity bây giờ sẽ trông như sau:

- 1.7 Kết nối mục menu Cài đặt với SettingsActivity Sử dụng  
Ý định để khởi chạy SettingsActivity từ MainActivity khi người dùng chọn Cài đặt từ menu tùy chọn.
  1. Mở MainActivity và tìm khôi if trong phương thức onOptionsItemSelected(), phương thức này xử lý việc nhấn vào Cài đặt trong menu tùy chọn:

2. Thêm Ý định vào khôi if để khởi chạy SettingsActivity :
3. Để thêm nút điều hướng lên trên thanh ứng dụng vào SettingsActivity , bạn cần chỉnh sửa khai báo của nó trong tệp AndroidManifest.xml để xác định công cụ chính SettingsActivity là MainActivity . Mở AndroidManifest.xml và tìm khai báo SettingsActivity:  
Thay đổi tờ khai thành như sau:
  4. Chạy ứng dụng. Nhấn vào biểu tượng tròn cho menu tùy chọn, như được hiển thị ở phía bên trái của thiết bị bên dưới. Nhấn vào Cài đặt để xem hoạt động cài đặt, như được hiển thị ở giữa hình bên dưới. Nhấn vào nút Lên trên thanh ứng dụng của hoạt động cài đặt, được hiển thị ở phía bên phải của hình bên dưới, để quay lại hoạt động chính.

#### 1.8 Lưu các giá trị mặc định trong tùy chọn được chia sẻ

Mặc dù giá trị mặc định cho cài đặt công tắc bật/tắt đã được đặt trong thuộc tính android:defaultValue (trong Bước 1.2 của tác vụ này), ứng dụng phải lưu giá trị mặc định trong tệp SharedPreferences cho từng cài đặt khi người dùng mở ứng dụng lần đầu tiên.

Làm theo các bước sau để đặt giá trị mặc định cho công tắc bật/tắt:

1. Mở MainActivity .
2. Thêm thông tin sau vào cuối phương thức onCreate() sau mã FloatingActionButton:

Mã trên đảm bảo rằng các cài đặt được khởi tạo đúng cách với các giá trị mặc định của chúng. Phương thức PreferenceManager.setDefaultValues() có ba đối số:

- Ngữ cảnh ứng dụng, chẳng hạn như .

• ID tài nguyên (tùy chọn) cho tệp tài nguyên XML với một hoặc nhiều cài đặt. • Một boolean cho biết liệu các giá trị mặc định có nên được đặt nhiều lần hay không. Khi false, hệ thống chỉ đặt giá trị mặc định nếu phương thức này chưa bao giờ được gọi. Miễn là bạn đặt đối số thứ ba này thành false , bạn có thể gọi phương thức này một cách an toàn mỗi khi MainActivity bắt đầu mà không ghi đè các giá trị cài đặt đã lưu của người dùng. Tuy nhiên, nếu bạn đặt nó thành true , phương thức sẽ ghi đè bất kỳ giá trị nào trước đó bằng mặc định.

1.9 Đọc giá trị cài đặt đã thay đổi từ các tùy chọn được chia sẻ Khi ứng dụng khởi động, phương thức MainActivity onCreate() có thể đọc các giá trị cài đặt đã thay đổi và sử dụng các giá trị đã thay đổi thay vì giá trị mặc định. Mỗi cài đặt được xác định bằng cách sử dụng cặp khóa-giá trị. Hệ thống Android sử dụng cặp khóa-giá trị này khi lưu hoặc truy xuất các tùy chọn cài đặt từ tệp SharedPreferences cho ứng dụng của bạn. Khi người dùng thay đổi cài đặt, hệ thống sẽ cập nhật giá trị tương ứng trong tệp SharedPreferences. Để sử dụng giá trị của cài đặt, ứng dụng có thể sử dụng khóa để lấy cài đặt từ tệp SharedPreferences bằng cách sử dụng PreferenceManager.getDefaultSharedPreferences() .

Làm theo các bước sau để thêm mã đó:

1. Mở SettingsActivity và tạo một biến String tĩnh để giữ khóa cho giá trị:
2. Mở MainActivity và thêm thông tin sau vào cuối phương thức onCreate():
3. Chạy ứng dụng và Nhấn vào Cài đặt để xem Hoạt động cài đặt.
4. Nhấn vào cài đặt để thay đổi chuyển đổi từ bật sang tắt, như được hiển thị ở phía bên trái của hình bên dưới.
5. Nhấn vào nút Lên trong Hoạt động cài đặt để quay lại MainActivity . Thông báo Toast sẽ xuất hiện trong MainActivity với giá trị của cài đặt, như được hiển thị ở phía bên phải của hình bên dưới.
6. Lặp lại các bước này để xem thông báo Toast thay đổi khi bạn thay đổi cài đặt.

Đoạn mã hiển thị ở trên sử dụng như sau:

• android.support.v7.preference.PreferenceManager.getDefaultSharedPreferences(th) để lấy cài đặt dưới dạng đối tượng SharedPreferences (sharedPref). • getBoolean () để lấy giá trị Boolean của cài đặt sử dụng khóa (KEY\_PREF\_EXAMPLE\_SWITCH) được xác định trong SettingsActivity ) và gán nó cho switchPref . Nếu không có giá trị cho khóa, phương thức getBoolean() đặt giá trị cài đặt ( switchPref ) thành false . Đối với các giá trị khác như chuỗi, số nguyên hoặc số dấu phẩy động, bạn có thể sử dụng các phương thức getString(), getInt() hoặc getFloat() tương ứng. • Toast.makeText() và show() để hiển thị giá trị của cài đặt switchPref. Bắt đầu khi nào MainActivity khởi động hoặc khởi động lại, phương thức onCreate() sẽ đọc các giá trị cài đặt để sử dụng chúng trong ứng dụng. Phương thức Toast.makeText() sẽ được thay thế bằng một phương thức khởi tạo cài đặt. Bây giờ bạn đã có Hoạt động cài đặt đang hoạt động trong ứng dụng của mình.

Mã giải pháp tác vụ 1 Dự án Android Studio: AppWithSettings

Nhiệm vụ 2: Sử dụng mẫu Hoạt động cài đặt

Nếu bạn cần tạo một số màn hình phụ của cài đặt và muốn tận dụng màn hình có kích thước máy tính bảng cũng như duy trì khả năng tương thích với các phiên bản Android cũ hơn dành cho máy

tính bảng, Android Studio cung cấp một phím tắt: mẫu Hoạt động cài đặt. Trong nhiệm vụ trước, bạn đã tìm hiểu cách sử dụng Hoạt động cài đặt trống và Fragment trống để thêm cài đặt vào ứng dụng. Nhiệm vụ 2 bây giờ sẽ chỉ cho bạn cách sử dụng mẫu Hoạt động Cài đặt được cung cấp cùng với Android Studio để:

- Chia nhiều cài đặt thành các nhóm.
- Tùy chỉnh cài đặt và giá trị của chúng.
- Hiển thị màn hình Cài đặt chính với liên kết tiêu đề cho từng nhóm cài đặt, chẳng hạn như Cài đặt chung cho cài đặt chung, như trong hình bên dưới.
- Hiển thị bố cục màn hình chính / chi tiết với liên kết tiêu đề cho từng nhóm ở phía bên trái (chính) và nhóm cài đặt ở phía bên phải (chi tiết), như thể hiện trong hình bên dưới.
- Hiển thị bố cục màn hình chính / chi tiết với liên kết tiêu đề cho từng nhóm ở phía bên trái (chính) và nhóm cài đặt ở phía bên phải (chi tiết), như thể hiện trong hình bên dưới.

Trong một thực tế trước đây, bạn đã tạo một ứng dụng có tên là DroidCafeOptionsUp bằng cách sử dụng mẫu Hoạt động cơ bản, cung cấp menu tùy chọn trong thanh ứng dụng như hình dưới đây.

Trong một thực tế trước đây, bạn đã tạo một ứng dụng có tên là DroidCafeOptionsUp bằng cách sử dụng mẫu Hoạt động cơ bản, cung cấp menu tùy chọn trong thanh ứng dụng như hình dưới đây.

Chú thích cho hình trên:

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

2.1 Khám phá mẫu Hoạt động Cài đặt Để đưa mẫu Hoạt động Cài đặt vào dự án ứng dụng trong Android Studio, hãy làm theo các bước sau:

1. Sao chép thư mục dự án DroidCafeOptionsUp và đổi tên thành DroidCafeWithSettings . Chạy ứng dụng để đảm bảo ứng dụng chạy bình thường.
2. Chọn ứng dụng ở đầu ngăn Dự án > Android và chọn Hoạt động > mới > Hoạt động cài đặt.

3. Trong hộp thoại xuất hiện, chấp nhận Tên hoạt động ( SettingsActivity là tên được đề xuất) và Tiêu đề ( Cài đặt ).

4. Nhấp vào ba dấu chấm ở cuối menu Hierarchical Parent và nhấp vào tab Project trong hộp thoại Select Activity xuất hiện (tham khảo hình bên dưới).

5. Mở rộng ứng dụng DroidCafeWithSettings > > src > chính > java > com.example.android.droidcafeinput và chọn MainActivity làm hoạt động mẹ, như trong hình bên dưới. Nhấp vào OK .

Bạn chọn MainActivity làm cha mẹ để nút Thanh ứng dụng Lên trong Hoạt động Cài đặt đưa người dùng trở lại MainActivity . Chọn Hoạt động chính sẽ tự động cập nhật tệp AndroidManifest.xml để hỗ trợ điều hướng nút Lên.

6. Nhấp vào Kết thúc .

7. Trong ngăn Project > Android, mở rộng ứng dụng > thư mục res > xml để xem các tệp XML được tạo bởi mẫu Hoạt động Cài đặt.

Bạn có thể mở và sau đó thêm vào hoặc tùy chỉnh các tệp XML cho các cài đặt bạn muốn:

- pref\_data\_sync.xml: Bố cục PreferenceScreen cho cài đặt "Dữ liệu & đồng bộ hóa".
- pref\_general.xml: Bố cục PreferenceScreen cho cài đặt "Chung".
- pref\_headers.xml: Bố cục tiêu đề cho màn hình chính Cài đặt.
- pref\_notification.xml: Bố cục tùy chọn Màn hình cho cài đặt "Thông báo". Các bố cục XML ở trên sử dụng các lớp con khác nhau của lớp Preference thay vì View và các lớp con trực tiếp cung cấp các vùng chứa cho các bố cục liên quan đến nhiều cài đặt. Ví dụ: PreferenceScreen đại diện cho Preference cấp cao nhất là gốc của hệ thống phân cấp Preference. Các tệp trên sử dụng PreferenceScreen ở đầu mỗi màn hình cài đặt. Các lớp con Tùy chọn khác cho cài đặt cung cấp giao diện người dùng thích hợp để người dùng thay đổi cài đặt. Chẳng hạn:

- CheckBoxPreference: Hộp kiểm cho cài đặt được bật hoặc tắt.
- ListPreference: Một hộp thoại với danh sách các nút radio.
- SwitchPreference: Một tùy chọn hai trạng thái có thể được chuyển đổi (chẳng hạn như bật / tắt hoặc đúng / sai).
- EditTextPreference: Một hộp thoại với EditText.

- RingtonePreference : Một hộp thoại với nhạc chuông trên thiết bị.

Mẹo: Bạn có thể chỉnh sửa các tệp XML để thay đổi cài đặt mặc định và tùy chỉnh văn bản cài đặt. Tất cả các chuỗi được sử dụng trong hoạt động cài đặt, chẳng hạn như tiêu đề cài đặt, mảng chuỗi cho danh sách và mô tả cài đặt, được định nghĩa là tài nguyên chuỗi trong tệp strings.xml. Chúng được đánh dấu bằng các nhận xét như sau: <!-- Chuỗi liên quan đến Cài đặt --> <!-- Ví dụ Cài đặt chung -->

Mẫu Hoạt động cài đặt cũng tạo:

- SettingsActivity trong java/com.example.android. projectname, bạn có thể sử dụng nguyên trạng. Đây là hoạt động hiển thị cài đặt. SettingsActivity mở rộng AppCompatActivityActivity để duy trì khả năng tương thích với các phiên bản Android cũ hơn.
- AppCompatActivityActivity trong java / com.example.android. projectname, mà bạn sử dụng nguyên trạng. Hoạt động này là một lớp trợ giúp mà SettingsActivity sử dụng để duy trì khả năng tương thích ngược với các phiên bản Android trước đó.

## 2.2 Thêm mục menu Cài đặt và kết nối nó với hoạt động

Như bạn đã học trong một thực tế khác, bạn có thể chỉnh sửa tệp menu\_main.xml cho menu tùy chọn để thêm hoặc xóa các mục menu.

1. Mở rộng thư mục res trong ngăn Project > Android và mở tệp menu\_main.xml. Nhấp vào tab Văn bản để hiển thị mã XML.
2. Thêm một mục menu khác có tên là Cài đặt với id tài nguyên mới action\_settings

Bạn chỉ định "không bao giờ" cho thuộc tính app:showAsAction để Cài đặt chỉ xuất hiện trong menu tùy chọn tràn chứ không phải trong chính thanh ứng dụng, vì nó không nên được sử dụng thường xuyên. Bạn chỉ định "50" cho thuộc tính android:orderInCategory để Cài đặt xuất hiện bên dưới Yêu thích (đặt thành "30") nhưng phía trên Liên hệ (đặt thành "100").

3. Trích xuất tài nguyên chuỗi cho "Cài đặt" trong thuộc tính android:title thành cài đặt tên tài nguyên.
4. Mở MainActivity và tìm khói trường hợp chuyển đổi trong phương thức onOptionsItemSelected() xử lý việc nhấn vào các mục trong menu tùy chọn.

Dưới đây là đoạn trích của phương pháp đó hiển thị trường hợp đầu tiên (đối với action\_order ):

5. Lưu ý trong mã trên rằng trường hợp đầu tiên sử dụng Intent để khởi chạy OrderActivity. Thêm trường hợp mới để action\_settings vào khôi hộp chuyển đổi
6. Chạy ứng dụng bằng điện thoại hoặc trình giả lập để bạn có thể xem cách mẫu Hoạt động cài đặt xử lý kích thước màn hình điện thoại.
7. Nhấn vào biểu tượng tràn cho menu tùy chọn và nhấn vào Cài đặt để xem Hoạt động cài đặt, như được hiển thị ở phía bên trái của hình bên dưới.
8. Nhấn vào từng tiêu đề cài đặt (Chung, Thông báo và Dữ liệu & đồng bộ hóa ), như được hiển thị ở giữa hình bên dưới, để xem nhóm cài đặt trên mỗi màn hình con của màn hình Cài đặt, được hiển thị ở phía bên phải của hình bên dưới.
9. Nhấn vào nút Lên trong Hoạt động Cài đặt để quay lại MainActivity .

Bạn sử dụng mã mẫu Hoạt động Cài đặt nguyên trạng. Nó không chỉ cung cấp bối cảnh cho màn hình có kích thước điện thoại và máy tính bảng mà còn cung cấp chức năng lắng nghe thay đổi cài đặt và thay đổi tóm tắt để phản ánh thay đổi cài đặt. Ví dụ: nếu bạn thay đổi cài đặt "Thêm bạn bè vào tin nhắn" (các lựa chọn là Luôn luôn, Khi có thể hoặc Không bao giờ ), lựa chọn bạn thực hiện sẽ xuất hiện trong bản tóm tắt bên dưới cài đặt:

Nói chung, bạn không cần thay đổi mã mẫu Hoạt động Cài đặt để tùy chỉnh Hoạt động cho các cài đặt bạn muốn trong ứng dụng của mình. Bạn có thể tùy chỉnh tiêu đề cài đặt, tóm tắt, giá trị có thể và giá trị mặc định mà không cần thay đổi mã mẫu và thậm chí thêm cài đặt vào các nhóm được cung cấp.

### 2.3 Tùy chỉnh cài đặt do mẫu cung cấp

Để tùy chỉnh cài đặt được cung cấp bởi mẫu Hoạt động Cài đặt, hãy chỉnh sửa tài nguyên chuỗi và mảng chuỗi trong tệp strings.xml và thuộc tính bối cảnh cho từng cài đặt trong tệp trong thư mục xml. Trong bước này, bạn sẽ thay đổi cài đặt "Dữ liệu và đồng bộ hóa".

1. Mở rộng thư mục res > values và mở tệp strings.xml. Cuộn nội dung đến <!-- Ví dụ cài đặt cho Data & Sync --> bình luận:

2. Chính sửa tài nguyên chuỗi pref\_header\_data\_sync, được đặt thành Data & sync (& là mã HTML cho dấu hiệu). Thay đổi giá trị thành Tài khoản (không có dấu ngoặc kép).
3. Bây giờ bạn nên tái cấu trúc tên tài nguyên (ứng dụng sẽ vẫn hoạt động mà không cần tái cấu trúc tên, nhưng tái cấu trúc giúp mã dễ hiểu hơn). Nhấp chuột phải (hoặc giữ Control khi bấm) tên tài nguyên pref\_header\_data\_sync chọn Refactor > Đổi tên. Thay đổi tên để pref\_header\_account, nhấp vào tùy chọn để tìm kiếm trong nhận xét và chuỗi, sau đó nhấp vào Tái cấu trúc.
4. Bạn cũng nên tái cấu trúc tên tệp XML (ứng dụng sẽ vẫn hoạt động mà không cần tái cấu trúc tên, nhưng tái cấu trúc giúp mã dễ hiểu hơn). Nhấp chuột phải (hoặc giữ Control khi nhấp) tên tài nguyên pref\_data\_sync trong ngăn Project > Android và chọn Refactor > Đổi tên. Thay đổi tên thành pref\_account, nhấp vào tùy chọn để tìm kiếm trong nhận xét và chuỗi, sau đó nhấp vào Tái cấu trúc.
5. Chính sửa tài nguyên chuỗi pref\_title\_sync\_frequency (được đặt thành Tần suất đồng bộ hóa) thành Thị trường.
6. Tái cấu trúc > Đổi tên tài nguyên pref\_title\_sync\_frequency thành pref\_title\_account như bạn đã làm trước đó.
7. Tái cấu trúc > Đổi tên tài nguyên mảng chuỗi pref\_sync\_frequency\_titles thành pref\_market\_titles.
8. Thay đổi từng giá trị trong mảng chuỗi pref\_market\_titles (15 phút, 30 phút, 1 giờ, v.v.) thành tiêu đề của thị trường, chẳng hạn như Hoa Kỳ, Canada, v.v., thay vì tần số:
9. Tái cấu trúc > Đổi tên tài nguyên mảng chuỗi pref\_sync\_frequency\_values thành pref\_market\_values.
10. Thay đổi từng giá trị trong mảng chuỗi pref\_market\_values (15, 30, 60, v.v.) thành giá trị cho thị trường — chữ viết tắt tương ứng với các quốc gia ở trên, chẳng hạn như Hoa Kỳ, CA, v.v.:
11. Cuộn xuống tài nguyên chuỗi pref\_title\_system\_sync\_settings và chỉnh sửa tài nguyên (được đặt thành Cài đặt đồng bộ hệ thống) thành Cài đặt tài khoản.
12. Tái cấu trúc > Đổi tên tài nguyên mảng chuỗi pref\_title\_system\_sync\_settings thành pref\_title\_account\_settings.

13. Mở tệp pref\_account.xml. ListPreference trong bộ cục này xác định cài đặt bạn vừa thay đổi. Lưu ý rằng các tài nguyên chuỗi cho các thuộc tính android:entries , android:entryValues và android:title hiện được thay đổi thành các giá trị bạn đã cung cấp trong các bước trước:

14. Thay đổi thuộc tính android:defaultValue thành "US":

Vì khóa cho tùy chọn cài đặt này ( "sync\_frequency" ) được mã hóa cứng ở nơi khác trong mã Java, nên không thay đổi thuộc tính android:key. Thay vào đó, hãy tiếp tục sử dụng "sync\_frequency" làm chìa khóa cho cài đặt này trong ví dụ này. Nếu bạn đang tùy chỉnh kỹ lưỡng các tùy chọn cài đặt cho một ứng dụng trong thế giới thực, bạn sẽ dành thời gian để thay đổi các khóa được mã hóa cứng trong toàn bộ mã.

Lưu ý: Tại sao không sử dụng tài nguyên chuỗi cho khóa? Vì tài nguyên chuỗi có thể được bản địa hóa cho các ngôn ngữ khác nhau bằng cách sử dụng tệp XML nhiều ngôn ngữ và chuỗi khóa có thể vô tình được dịch cùng với các chuỗi khác, điều này sẽ khiến ứng dụng gặp sự cố.

## 2.4 Thêm mã để đặt giá trị mặc định cho cài đặt

Để thêm mã để đặt giá trị mặc định cho cài đặt, hãy làm theo các bước sau:

1. Mở MainActivity và tìm phương thức onCreate().

2. Thêm các câu lệnh PreferenceManager.setDefaultValues sau đây ở cuối phương thức onCreate():

Các giá trị mặc định đã được chỉ định trong tệp XML với thuộc tính android:defaultValue, nhưng các câu lệnh trên đảm bảo rằng tệp SharedPreferences được khởi tạo đúng cách với các giá trị mặc định. Phương thức setDefaultValues() có ba đối số:

- Ngữ cảnh ứng dụng, chẳng hạn như .
- ID tài nguyên cho tệp XML bộ cục cài đặt bao gồm các giá trị mặc định được đặt bởi thuộc tính android:defaultValue.
- Một boolean cho biết liệu các giá trị mặc định có nên được đặt nhiều lần hay không. Khi false, hệ thống chỉ đặt các giá trị mặc định khi phương thức này được gọi lần đầu tiên. Miễn là bạn đặt đối số thứ ba này thành false , bạn có thể

gọi phương thức này một cách an toàn mỗi khi Hoạt động bắt đầu mà không ghi đè các giá trị cài đặt đã lưu của người dùng bằng cách đặt lại chúng về giá trị mặc định. Tuy nhiên, nếu bạn đặt nó thành true , phương thức sẽ ghi đè bất kỳ giá trị nào trước đó bằng mặc định.

## 2.5 Thêm mã để đọc giá trị cho cài đặt

1. Thêm mã sau ở cuối phương thức MainActivity onCreate(). Bạn có thể thêm nó ngay sau mã bạn đã thêm ở bước trước để đặt mặc định cho cài đặt:

Như bạn đã học trong nhiệm vụ trước, bạn sử dụng

PreferenceManager.getDefaultSharedPreferences(this) để lấy cài đặt dưới dạng đối tượng SharedPreferences ( marketPref ). Sau đó, bạn sử dụng getString() để lấy giá trị chuỗi của cài đặt sử dụng khóa ( sync\_frequency ) và gán nó cho marketPref . Nếu không có giá trị cho khóa, phương thức getString() gán giá trị cài đặt của marketPref thành -1, là giá trị của Other trong mảng pref\_market\_values.

2. Chạy ứng dụng. Khi màn hình chính của ứng dụng xuất hiện lần đầu tiên, bạn sẽ thấy thông báo Toast ở cuối màn hình. Lần đầu tiên bạn chạy ứng dụng, bạn sẽ thấy "-1" hiển thị trong Toast vì bạn chưa thay đổi cài đặt.

3. Nhấn vào Cài đặt trong menu tùy chọn và nhấn vào Tài khoản trong màn hình Cài đặt. Nhấn vào Thị trường và chọn Canada như hình dưới đây:

4. Nhấn vào nút Lên trên thanh ứng dụng để quay lại màn hình Cài đặt và nhấn lại để quay lại màn hình chính.

5. Chạy lại ứng dụng từ Android Studio. Bạn sẽ thấy thông báo Toast có "CA" (đối với Canada) và cài đặt Thị trường hiện được đặt thành Canada.

6. Jetzt füge den folgenden Code in die Methode `onCreate()` hinzu:  
Bây giờ hãy chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng. Vì máy tính bảng có màn hình lớn hơn nên thời gian chạy Android tận dụng dung lượng bổ sung. Trên máy tính bảng, cài đặt và chi tiết được hiển thị trên cùng một màn hình giúp người dùng quản lý cài đặt của họ dễ dàng hơn.

Mã giải pháp nhiệm vụ 2 Dự án Android Studio:

Thử thách mã hóa DroidCafeWithSettings

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

Thử thách: Ứng dụng DroidCafeWithSettings hiển thị cài đặt trên màn hình có kích thước bằng máy tính bảng một cách chính xác, nhưng nút Lên trên thanh

Ứng dụng không đưa người dùng trở lại MainActivity như trên màn hình có kích thước điện thoại. Điều này là do ba phương thức onOptionsItemSelected() — một phương thức cho mỗi Fragment — trong SettingsActivity . Nó sử dụng các tùy chọn sau để khởi động lại SettingsActivity khi người dùng nhấn vào nút Lên:

Trên đây là hành động thích hợp trên màn hình điện thoại trong đó tiêu đề Cài đặt ( Chung , Thông báo và Tài khoản ) xuất hiện trong một màn hình riêng biệt ( SettingsActivity ). Sau khi thay đổi cài đặt, bạn muốn người dùng nhấn vào nút Lên để đưa người dùng trở lại tiêu đề Cài đặt trong SettingsActivity . Nhấn thêm vào Lên sẽ đưa người dùng trở lại MainActivity .

Tuy nhiên, trên máy tính bảng, tiêu đề luôn hiển thị trong ngăn bên trái (trong khi cài đặt ở ngăn bên phải). Do đó, nhấn vào nút Lên không đưa người dùng đến MainActivity . Tìm cách làm cho nút Lên hoạt động bình thường trong Cài đặt Hoạt động trên màn hình có kích thước bằng máy tính bảng.

Gợi ý : Mặc dù có một số cách để khắc phục sự cố này, nhưng hãy xem xét các bước sau:

1. Thêm một tệp dimens.xml khác để phù hợp cụ thể với kích thước màn hình lớn hơn 600dp. Khi ứng dụng chạy trên một thiết bị cụ thể, tệp dimens.xml thích hợp sẽ được chọn dựa trên các bộ hạn định cho các tệp dimens.xml. Bạn có thể thêm một tệp dimens.xml khác với bộ hạn định Chiều rộng màn hình nhỏ nhất được đặt thành 600 dp ( sw600dp ), như bạn đã học trong thực tế về hỗ trợ kích thước màn hình và ngang để chỉ định bất kỳ thiết bị nào có màn hình lớn, chẳng hạn như máy tính bảng.
2. Thêm tài nguyên bool sau giữa thẻ <resources> và </resources> trong tệp dimens.xml (sw600dp), được chọn tự động cho máy tính bảng:
3. Thêm tài nguyên bool sau vào tệp dimens.xml chuẩn, được chọn khi ứng dụng chạy trên bất kỳ thiết bị nào không lớn:
3. In SettingsActivity , you can add to all three onOptionsItemSelected() methods (one for each Fragment ) an if else block that checks to see if isTablet is true. If it is, your code can redirect the Up button action to MainActivity .

## Mã giải pháp thử thách dự án Android Studio: DroidCafeWithSettings

### Tóm tắt thử thách

Người dùng mong muốn điều hướng đến cài đặt ứng dụng bằng cách nhấn vào Cài đặt trong điều hướng bên, chẳng hạn như ngăn điều hướng hoặc trong menu tùy chọn trên thanh ứng dụng. Để cung cấp cài đặt người dùng cho ứng dụng của bạn, hãy cung cấp Hoạt động cho cài đặt:

- Sử dụng Hoạt động lưu trữ PreferenceFragment để hiển thị cài đặt ứng dụng. • Để duy trì khả năng tương thích với AppCompatActivity và thư viện android.support.v7.preference, hãy sử dụng PreferenceFragmentCompat thay vì PreferenceFragment .

Hiển thị từng phân đoạn trong hoạt động cài đặt:

- Nếu lớp hoạt động mở rộng Hoạt động và lớp phân đoạn mở rộng PreferenceFragment , hãy sử dụng getSupportFragmentManager() .
- Nếu lớp hoạt động mở rộng AppCompatActivity và lớp phân đoạn mở rộng PreferenceFragmentCompat , hãy sử dụng getSupportFragmentManager() .
- Để liên kết tài nguyên cài đặt preferences.xml với mảnh, hãy sử dụng setPreferencesFromResource() .
- Để đặt các giá trị mặc định cho cài đặt, hãy sử dụng PreferenceManager.setDefaultValues().
- Để kết nối mục menu Cài đặt với hoạt động cài đặt, hãy sử dụng Ý định. Thêm các tệp tài nguyên XML cho cài đặt:
  1. Tạo một thư mục tài nguyên mới ( Tệp > Mới > Thư mục tài nguyên Android ).
  2. Trong menu thả xuống Loại tài nguyên, chọn xml . Thư mục xml xuất hiện bên trong thư mục res.
  3. Nhập vào xml và chọn File > New > XML resource file.
  4. Nhập tùy chọn làm tên của tệp XML. Tệp preferences.xml xuất hiện bên trong thư mục xml. Thêm các điều khiển giao diện người dùng như công tắc bật tắt, với các thuộc tính trong tệp XML tùy chọn:

- Để duy trì khả năng tương thích với AppCompatActivity , hãy sử dụng phiên bản thư viện android.support.v7.preference. Ví dụ: sử dụng SwitchPreferenceCompat cho các công tắc bật tắt. Sử dụng các thuộc tính với từng phần tử giao diện người dùng cho cài đặt:

- android:defaultValue là giá trị của cài đặt khi ứng dụng khởi động lần đầu tiên.
- Android: Title là tiêu đề cài đặt mà người dùng hiển thị.
- Android: Key là khóa được sử dụng để lưu trữ giá trị cài đặt.
- android:summary là văn bản mà người dùng hiển thị xuất hiện trong cài đặt. Lưu và đọc các giá trị cài đặt:
- Khi ứng dụng khởi động, phương thức MainActivity onCreate() có thể đọc các giá trị cài đặt đã thay đổi và sử dụng các giá trị đã thay đổi thay vì giá trị mặc định.
- Mỗi cài đặt được xác định bằng cách sử dụng một cặp khóa-giá trị. Hệ thống Android sử dụng cặp khóa-giá trị này khi lưu hoặc truy xuất các tùy chọn cài đặt từ tệp SharedPreferences cho ứng dụng của bạn. Khi người dùng thay đổi cài đặt, hệ thống sẽ cập nhật giá trị tương ứng trong tệp SharedPreferences.
- Để sử dụng giá trị của cài đặt, ứng dụng của bạn có thể sử dụng phím để lấy cài đặt từ tệp SharedPreferences.
- Ứng dụng của bạn đọc các giá trị cài đặt từ SharedPreferences bằng cách sử dụng PreferenceManager.getDefaultSharedPreferences() và lấy từng giá trị cài đặt bằng .getString, .getBoolean, v.v. Các khái niệm liên quan Tài liệu về khái niệm liên quan có trong 9.2: Cài đặt ứng dụng . Tìm hiểu thêm tài liệu về Android

Studio: Hướng dẫn sử dụng Android Studio: Tài liệu dành cho nhà phát triển Android:

- Cài đặt (tổng quan)
- Tùy chọn
- PreferenceFragment
- PreferenceFragmentCompat
- Fragment
- SharedPreferences
- Lưu dữ liệu khóa-giá trị
- Hỗ trợ các kích thước màn hình khác nhau Đặc điểm kỹ thuật Material Design: Cài đặt Android Stack Overflow:
  - Làm thế nào để người ta dimens.xml vào Android Studio?
  - Xác định xem thiết bị là điện thoại thông minh hay máy tính bảng? Bài tập về nhà Xây dựng và chạy một ứng dụng Mở dự án ứng dụng DroidCafeWithSettings.

1. Thêm ListPreference (hộp thoại có các nút radio) vào cài đặt chung. Đặt hộp thoại vào màn hình Cài đặt chung, bên dưới "Thêm bạn bè để đặt hàng tin nhắn" ListPreference .
2. Chính sửa các mảng chuỗi được sử dụng cho ListPreference để bao gồm tiêu đề "Chọn phương thức phân phối". Sử dụng các lựa chọn giao hàng tương tự được sử dụng trong các nút radio trong OrderActivity .
3. Làm cho phương thức phân phối mà người dùng đã chọn xuất hiện trong cùng một thông báo Toast với cài đặt Thị trường đã chọn.
4. Tín dụng bổ sung: Hiển thị phương thức phân phối đã chọn dưới dạng văn bản tóm tắt cài đặt xuất hiện bên dưới tiêu đề ListPreference. Cho phép văn bản này thay đổi theo mỗi bản cập nhật.

Trả lời các câu hỏi này

Câu hỏi 1: Trong tệp nào của dự án DroidCafeWithSettings, bạn xác định mảng các mục nhập và mảng giá trị cho ListPreference? Chọn một:

- pref\_general.xml
- strings.xml
- menu\_main.xml
- activity\_main.xml
- content\_main.xml

Câu hỏi 2 Trong tệp nào của dự án DroidCafeWithSettings, bạn sử dụng mảng mục nhập và mảng giá trị để thiết lập ListPreference, đồng thời đặt khóa ListPreference và giá trị mặc định? Chọn một:

- pref\_general.xml
- strings.xml
- menu\_main.xml
- content\_main.xml
- SettingsActivity.java

Câu hỏi 3 Làm cách nào để bạn thiết lập cài đặt Hoạt động và Fragment với SwitchPreference cho giao diện người dùng mà vẫn tương thích với thư viện appcompat v7 để tương thích ngược với các phiên bản Android cũ hơn?

- Sử dụng hoạt động cài đặt mở rộng Hoạt động , một mảnh mở rộng PreferenceFragment và SwitchPreference cho giao diện người dùng.
- Thay đổi MainActivity để mở rộng Hoạt động.
- Sử dụng hoạt động cài đặt mở rộng AppCompatActivity , một phân đoạn mở rộng PreferenceFragmentCompat và SwitchPreferenceCompat cho giao diện người dùng.
- Bạn không thể sử dụng một mảnh với SwitchPreference và vẫn tương thích với thư viện appcompat v7 . Gửi ứng dụng của bạn để chấm điểm Hướng dẫn dành cho người chấm điểm Kiểm tra xem ứng dụng có các tính năng sau không: • Phương thức onCreate() đọc cài đặt deliveryPref bằng sharedPref.getString() .
- Tệp pref\_general.xml bao gồm ListPreference sử dụng cho các mục nhập của nó một mảng các lựa chọn phân phối.
- Tín dụng bổ sung: Câu lệnh bindPreferenceSummaryToValue(findPreference("delivery")) đã được thêm vào phương thức onCreate() của lớp GeneralPreferenceFragment trong SettingsActivity để hiển thị lựa chọn phân phối trong tóm tắt tùy chọn.

## Bài 2: Lưu trữ dữ liệu với Room

### 2.1 Room, LiveData và ViewModel

Giới thiệu Hệ điều hành Android cung cấp nền tảng vững chắc để xây dựng các ứng dụng chạy tốt trên nhiều loại thiết bị và yêu tố hình thức. Tuy nhiên, các vấn đề như vòng đời phức tạp và thiếu kiến trúc ứng dụng được đề xuất khiến việc viết các ứng dụng mạnh mẽ trở nên khó khăn. Các thành phần kiến trúc Android cung cấp các thư viện cho các tác vụ phổ biến như quản lý vòng đời và tính năng duy trì dữ liệu để giúp triển khai kiến trúc được đề xuất dễ dàng hơn. Thành phần kiến trúc giúp bạn cấu trúc ứng dụng theo cách mạnh mẽ, có thể kiểm tra và có thể bảo trì với ít mã soạn sẵn hơn.

Các thành phần kiến trúc được đề xuất là gì?

Khi nói đến kiến trúc, bạn nên nhìn thấy bức tranh toàn cảnh trước. Để giới thiệu thuật ngữ, đây là một cái nhìn tổng quan ngắn gọn về các Thành phần Kiến trúc và cách chúng hoạt động cùng nhau. Mỗi thành phần được giải thích thêm khi bạn sử dụng nó trong thực tế này.

Sơ đồ dưới đây cho thấy hình thức cơ bản của kiến trúc được đề xuất cho các ứng dụng sử dụng Thành phần kiến trúc. Kiến trúc bao gồm bộ điều khiển giao diện người dùng, ViewModel phục vụ LiveData, Kho lưu trữ và cơ sở dữ liệu Room. Cơ sở dữ

liệu Room được hỗ trợ bởi cơ sở dữ liệu SQLite và có thể truy cập thông qua đối tượng truy cập dữ liệu (DAO). Mỗi thành phần được mô tả ngắn gọn bên dưới và chi tiết trong chương khái niệm Thành phần kiến trúc, 10.1: Lưu trữ dữ liệu với Room. Bạn thực hiện các thành phần trong thực tế này.

Bởi vì tất cả các thành phần tương tác, bạn sẽ gặp phải các tham chiếu đến các thành phần này trong suốt thực tế này, vì vậy đây là một giải thích ngắn gọn về từng thành phần.

**Thực thể:** Trong ngữ cảnh của Thành phần kiến trúc, thực thể là một lớp có chú thích mô tả một bảng cơ sở dữ liệu.

**Cơ sở dữ liệu SQLite:** Trên thiết bị, dữ liệu được lưu trữ trong cơ sở dữ liệu SQLite. Thư viện tính bền vững của Room tạo và duy trì cơ sở dữ liệu này cho bạn.

**DAO:** Viết tắt của đối tượng truy cập dữ liệu. Ánh xạ các truy vấn SQL đến các hàm. Bạn đã từng phải xác định các truy vấn này trong một lớp trợ giúp. Khi bạn sử dụng DAO, mã của bạn sẽ gọi các hàm và các thành phần sẽ xử lý phần còn lại.

**Cơ sở dữ liệu phòng:** Lớp cơ sở dữ liệu trên cơ sở dữ liệu SQLite xử lý các tác vụ thông thường mà bạn đã từng xử lý với lớp trợ giúp. Cơ sở dữ liệu Room sử dụng DAO để đưa ra truy vấn đến cơ sở dữ liệu SQLite dựa trên các hàm được gọi.

**Kho lưu trữ:** Một lớp mà bạn tạo để quản lý nhiều nguồn dữ liệu. Ngoài cơ sở dữ liệu Room, Kho lưu trữ có thể quản lý các nguồn dữ liệu từ xa như máy chủ web.

**ViewModel :** Cung cấp dữ liệu cho giao diện người dùng và hoạt động như một trung tâm giao tiếp giữa Kho lưu trữ và giao diện người dùng. Ân phần phụ trợ khỏi giao diện người dùng. Các phiên bản ViewModel vẫn tồn tại các thay đổi về cấu hình thiết bị.

**LiveData :** Một lớp giữ dữ liệu tuân theo mẫu người quan sát , có nghĩa là nó có thể được quan sát. Luôn lưu trữ / lưu trữ phiên bản dữ liệu mới nhất. Thông báo cho người quan sát khi dữ liệu thay đổi. Nói chung, các thành phần giao diện người dùng quan sát dữ liệu có liên quan. LiveData nhận biết vòng đời, vì vậy nó sẽ tự động quản lý việc dừng và tiếp tục quan sát dựa trên trạng thái của hoạt động quan sát hoặc phân đoạn của nó.

Những điều bạn nên biết Bạn sẽ có thể tạo và chạy ứng dụng trong Android Studio 3.0 trở lên.

Đặc biệt, hãy làm quen với:

- RecyclerView và bộ điều hợp

- Cơ sở dữ liệu SQLite và ngôn ngữ truy vấn SQLite
- Threading nói chung và AsyncTask nói riêng

Nó giúp làm quen với:

- Các mẫu kiến trúc phần mềm tách dữ liệu khỏi giao diện người dùng.
- Mô hình quan sát . Tóm lại, mẫu observer xác định sự phụ thuộc một-nhiều giữa các đối tượng. Bất cứ khi nào một đối tượng thay đổi trạng thái của nó, tất cả các phụ thuộc của đối tượng sẽ được thông báo và cập nhật tự động. Đối tượng chính được gọi là "chủ thẻ" và những người phụ thuộc của nó được gọi là "người quan sát". Thông thường, đối tượng thông báo cho người quan sát bằng cách gọi một trong các phương thức của người quan sát. Đối tượng biết phương thức nào để gọi, bởi vì các quan sát viên được "đăng ký" với chủ thẻ và chỉ định các phương thức để gọi.

Lưu ý quan trọng: Thực tế này triển khai kiến trúc được xác định trong Hướng dẫn về Kiến trúc ứng dụng và được giải thích trong chương Khái niệm Thành phần kiến trúc, 10.1: Lưu trữ dữ liệu bằng Room. Bạn nên đọc chương khái niệm.

Những gì bạn sẽ học

- Cách thiết kế và xây dựng một ứng dụng bằng cách sử dụng một số thành phần kiến trúc Android. Bạn sẽ sử dụng Room, ViewModel và LiveData . Những gì bạn sẽ làm
- Tạo một ứng dụng với Hoạt động hiển thị các từ trong RecyclerView .
- Tạo một Thực thể đại diện cho các đối tượng từ.
- Xác định ánh xạ các truy vấn SQL với các phương thức Java trong DAO (đối tượng truy cập dữ liệu).
- Sử dụng LiveData để thực hiện các thay đổi đối với dữ liệu hiển thị trên giao diện người dùng, thông qua người quan sát.
- Thêm cơ sở dữ liệu Room vào ứng dụng để lưu trữ dữ liệu cục bộ và khởi tạo cơ sở dữ liệu.
- Trừu tượng hóa phần phụ trợ dữ liệu dưới dạng lớp Kho lưu trữ với một API không liên quan đến cách dữ liệu được lưu trữ hoặc thu thập.
- Sử dụng ViewModel để tách tất cả các thao tác dữ liệu khỏi giao diện người dùng.
- Thêm Hoạt động thứ hai cho phép người dùng thêm từ mới.

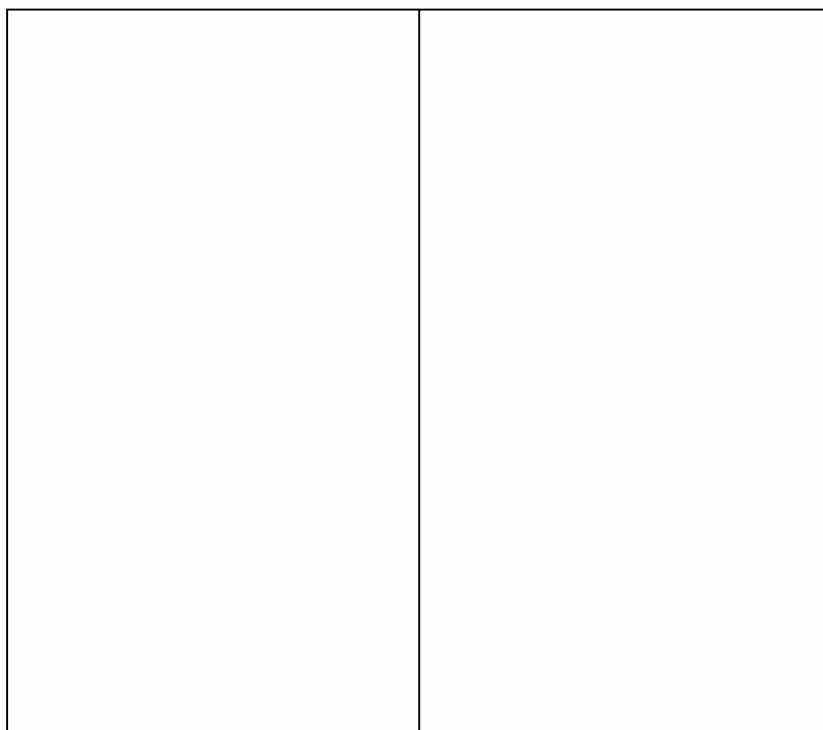
Tổng quan về ứng dụng

Trong thực tế này, bạn xây dựng một ứng dụng sử dụng Android Architecture Components . Ứng dụng này có tên là RoomWordsSample lưu trữ danh sách các từ trong cơ sở dữ liệu Room và hiển thị danh sách trong RecyclerView . Ứng dụng RoomWordsSample là cơ bản, nhưng đủ đầy đủ để bạn có thể sử dụng nó như một mẫu để xây dựng. Ứng dụng RoomWordsSample thực hiện như sau:

- Hoạt động với cơ sở dữ liệu để lấy và lưu từ, đồng thời điền trước cơ sở dữ liệu với một số từ.
- Hiển thị tất cả các từ trong RecyclerView trong MainActivity.
- Mở Hoạt động thứ hai khi người dùng nhấn vào nút + FAB. Khi người dùng nhập một từ, ứng dụng sẽ thêm từ đó vào cơ sở dữ liệu và sau đó danh sách sẽ tự động cập nhật.

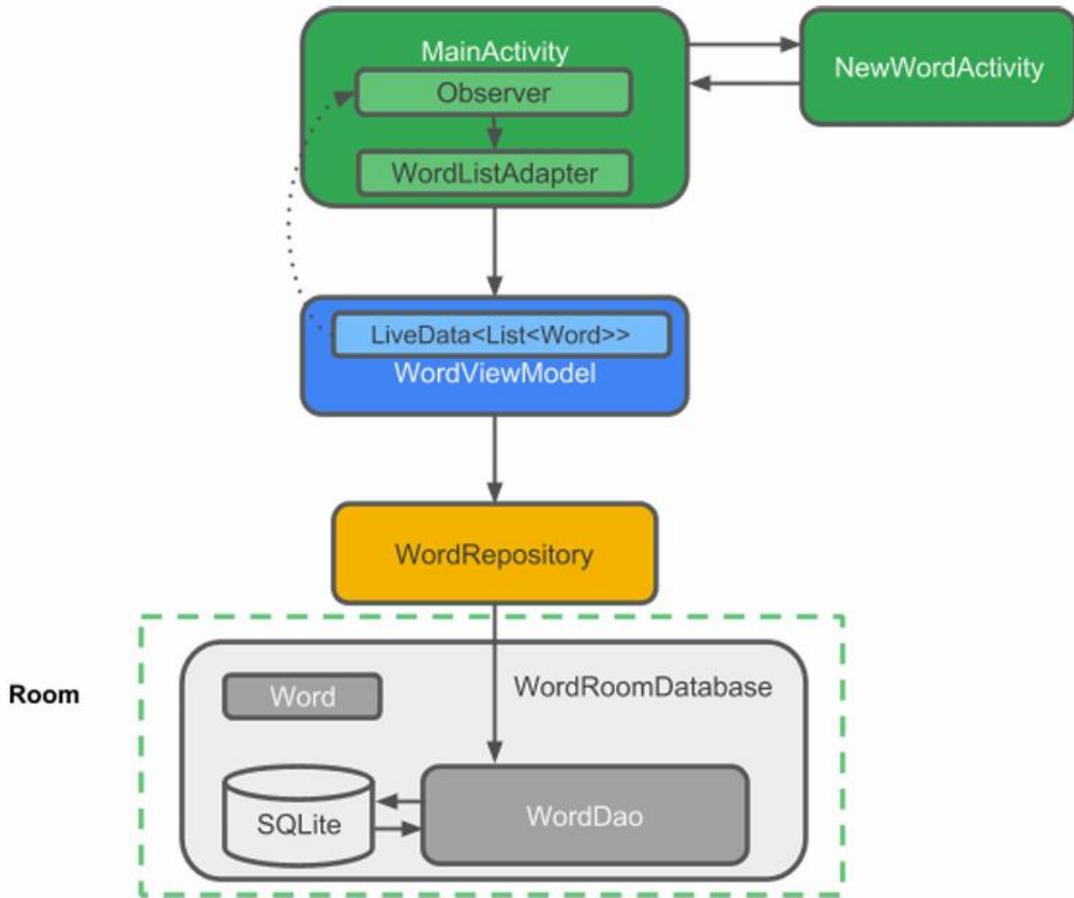
Ảnh chụp màn hình bên dưới cho thấy những điều sau:

- Ứng dụng RoomWordsSample khi nó bắt đầu, với danh sách các từ ban đầu
- Hoạt động thêm từ



Tổng quan về kiến trúc RoomWordsSample Sơ đồ sau phản ánh sơ đồ tổng quan từ phần giới thiệu và hiển thị tất cả các phần của ứng dụng RoomWordsSample. Mỗi hộp bao quanh (ngoại trừ cơ sở dữ liệu SQLite) đại diện cho một lớp mà bạn tạo.

Mẹo: In hoặc mở sơ đồ này trong một tab riêng biệt để bạn có thể tham khảo sơ đồ khi tạo mã.



### Nhiệm vụ 1: Tạo ứng dụng RoomWordsSample

Lưu ý: Trong thực tế này, bạn phải tạo các biến thành viên, nhập lớp và trích xuất các giá trị khi cần thiết. Mã mà bạn phải quen thuộc được cung cấp nhưng không được giải thích.

1.1 Tạo một ứng dụng bằng một Hoạt động Mở Android Studio và tạo một ứng dụng. Trên màn hình thiết lập, hãy làm như sau:

- Đặt tên cho ứng dụng RoomWordsSample.
- Nếu bạn thấy các hộp kiểm Bao gồm hỗ trợ Kotlin và Bao gồm hỗ trợ C++, hãy bỏ chọn cả hai hộp.
- Chỉ chọn hệ số hình thức Điện thoại & Máy tính bảng và đặt SDK tối thiểu thành API 14 trở lên.

- Chọn Hoạt động cơ bản.

## 1.2 Cập nhật tệp Gradle

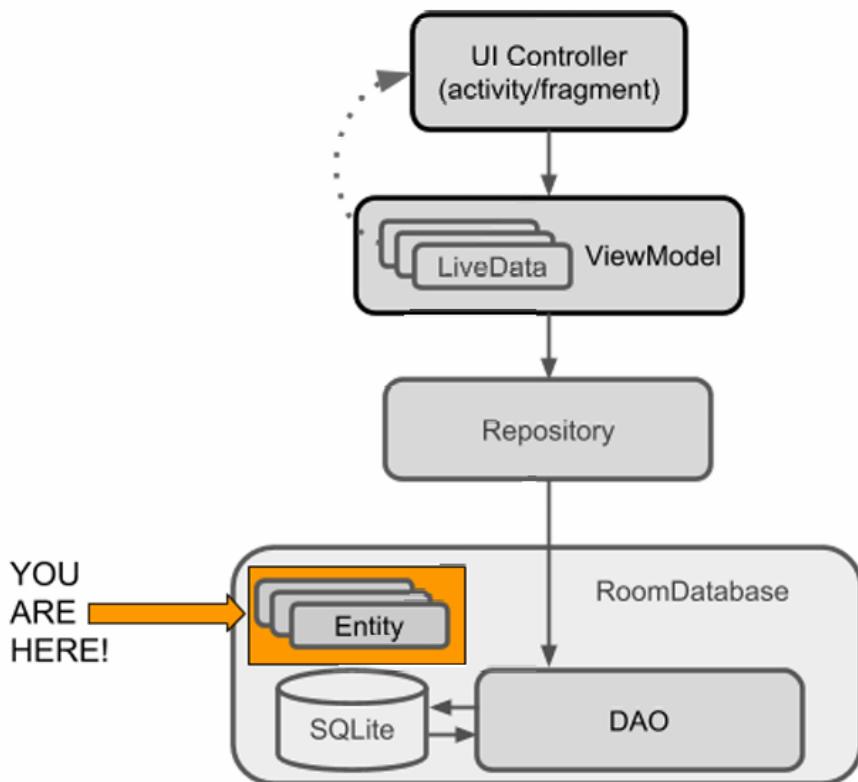
Trong Android Studio, hãy thêm thư viện Thành phần kiến trúc vào tệp Gradle theo cách thủ công.

Thêm mã sau vào tệp build.gradle (Module: app) của bạn, vào cuối khôi phụ thuộc (nhưng vẫn bên trong nó).

Trong tệp build.gradle (Project: RoomWordsSample), hãy thêm số phiên bản vào cuối tệp.

### Nhiệm vụ 2: Tạo thực thể Word

Sơ đồ dưới đây là sơ đồ kiến trúc hoàn chỉnh với thành phần mà bạn sẽ triển khai trong nhiệm vụ này được đánh dấu. Mỗi nhiệm vụ sẽ có một sơ đồ như vậy để giúp bạn hiểu vị trí của thành phần hiện tại phù hợp với cấu trúc tổng thể của ứng dụng và để xem các thành phần được kết nối như thế nào.



Dữ liệu cho ứng dụng này là các từ và mỗi từ được thể hiện bởi một thực thể trong cơ sở dữ liệu. Trong tác vụ này, bạn tạo lớp Word và chú thích để Room có thể tạo bảng cơ sở dữ liệu từ lớp đó. Sơ đồ dưới đây cho thấy một bảng cơ sở dữ liệu word\_table.

Bảng có một cột từ, cũng đóng vai trò là khóa chính và hai hàng, mỗi hàng một cho "Xin chào" và "Thế giới".

word_table table
word (primary key, string)
"Hello"
"World"

## 2.1 Tạo lớp Word

1. Tạo một lớp có tên là Word .
2. Thêm một hàm khởi tạo lấy một chuỗi từ làm đối số. Thêm chú thích @NonNull để tham số không bao giờ có thể là null .
3. Thêm một phương thức "getter" có tên là getWord() trả về từ. Room yêu cầu các phương thức "getter" trên các lớp thực thể để nó có thể khởi tạo các đối tượng của bạn.

## 2.2 Chú thích lớp Word

Để làm cho lớp Word có ý nghĩa đối với cơ sở dữ liệu Room, bạn phải chú thích lớp đó. Chú thích xác định cách mỗi phần của lớp Word liên quan đến một mục nhập trong cơ sở dữ liệu. Room sử dụng thông tin này để tạo mã.

Bạn sử dụng các chú thích sau trong các bước dưới đây:

- `@Entity(tableName = "word_table" )` Mỗi lớp `@Entity` đại diện cho một thực thể trong một bảng. Chú thích khai báo lớp của bạn để chỉ ra rằng lớp là một thực thể. Chỉ định tên của bảng nếu bạn muốn nó khác với tên của lớp.

- `@PrimaryKey` Mọi thực thể đều cần một khóa chính. Để giữ cho mọi thứ đơn giản, mỗi từ trong ứng dụng RoomWordsSample đóng vai trò là khóa chính của riêng nó. Để tìm hiểu cách tự động tạo các khóa duy nhất, hãy xem mèo bên dưới.
- `@NonNull` Biểu thị rằng một tham số, trường hoặc giá trị trả về phương thức không bao giờ có thể là rỗng. Khóa chính phải luôn sử dụng chú thích này. Sử dụng chú thích này cho bất kỳ trường bắt buộc nào trong hàng của bạn.
- `@ColumnInfo` (tên = "tù") Chỉ định tên của một cột trong bảng, nếu bạn muốn tên cột khác với tên của biến thành viên.
- Mọi trường được lưu trữ trong cơ sở dữ liệu phải là công khai hoặc có phương thức "getter". Ứng dụng này cung cấp phương thức "getter" `getWord()` thay vì hiển thị trực tiếp các biến thành viên.

Để biết danh sách đầy đủ các chú thích, hãy xem tài liệu tham khảo tóm tắt gói phòng

Cập nhật lớp Word của bạn bằng chú thích, như được hiển thị trong mã bên dưới.

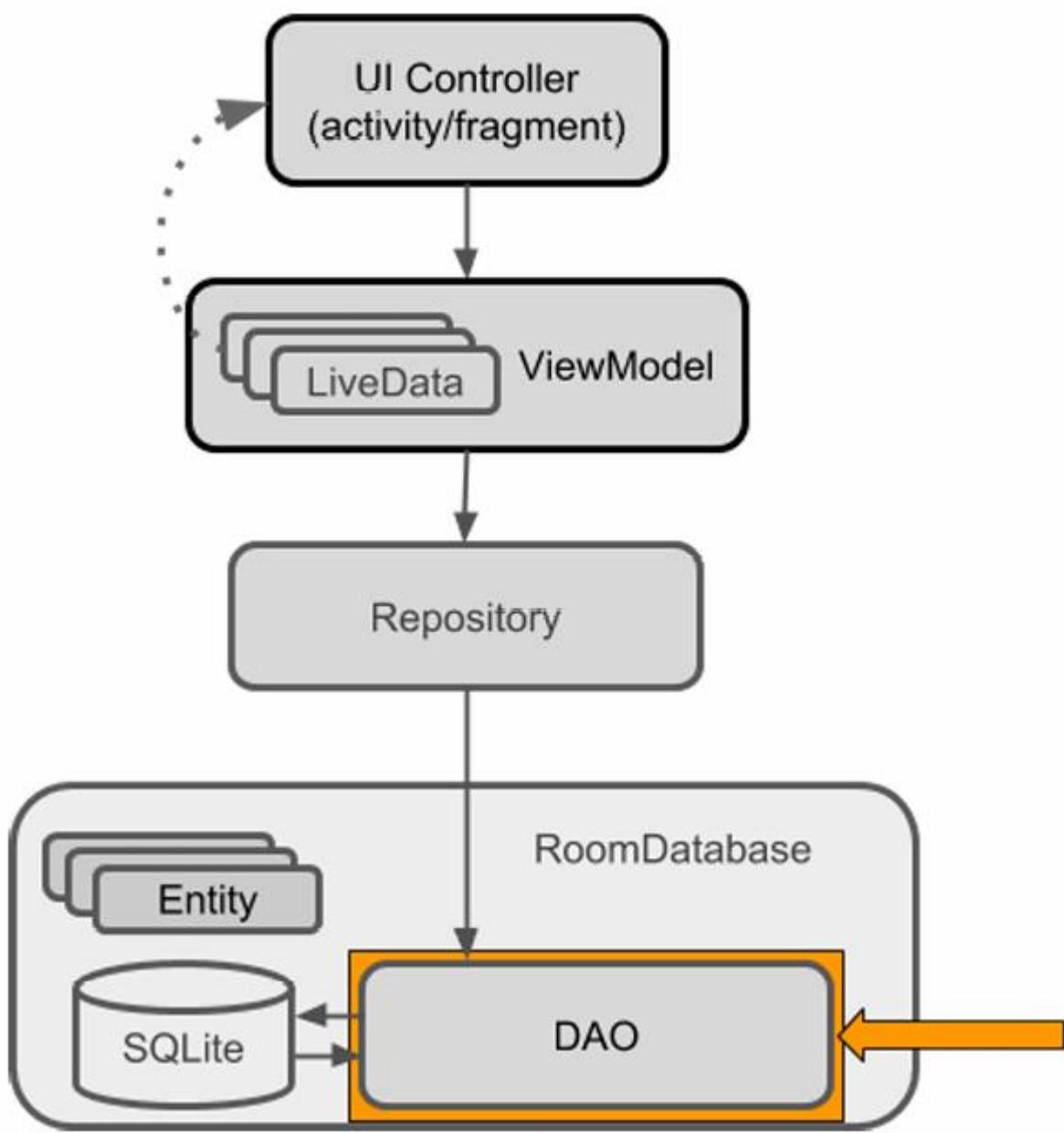
1. Thêm ký hiệu `@Entity` vào khai báo lớp và đặt `tableName` thành "word\_table" . 2. Chú thích biến thành viên `mWord` làm `@PrimaryKey` . Yêu cầu `mWord` được `@NonNull` và đặt tên cột là "word" .
2. Lưu ý: Nếu bạn nhập chú thích, Android Studio sẽ tự động nhập mọi thứ bạn cần.

Đây là mã đầy đủ:

Nếu bạn gặp lỗi cho chú thích, bạn có thể nhập chúng theo cách thủ công, như sau:

Mèo về cách tự động tạo khóa: Để tự động tạo một khóa duy nhất cho mỗi thực thể, bạn sẽ thêm và chú thích một khóa số nguyên chính với `autoGenerate=true` . Xem phần Xác định dữ liệu bằng thực thể Room .

### Nhiệm vụ 3: Tạo DAO



Đối tượng truy cập dữ liệu, hoặc Dao, là một lớp có chú thích, nơi bạn chỉ định các truy vấn SQL và liên kết chúng với các lệnh gọi phương thức. Trình biên dịch kiểm tra lỗi SQL, sau đó tạo các truy vấn từ các chú thích. Đối với các truy vấn phổ biến, các thư viện cung cấp các chú thích thuận tiện như `@Insert`.

Lưu ý rằng:

- DAO phải là một giao diện hoặc lớp trừu tượng.
- Room sử dụng DAO để tạo một API sạch cho mã của bạn.

- Theo mặc định, tất cả các truy vấn ( @Query ) phải được thực thi trên một luồng khác với luồng chính. (Bạn làm việc đó sau.) Đối với các thao tác như chèn hoặc xóa, nếu bạn sử dụng các chú thích tiện lợi được cung cấp, Room sẽ xử lý quản lý chủ đề cho bạn.

### 3.1 Triển khai lớp DAO

DAO cho thực tế này là cơ bản và chỉ cung cấp các truy vấn để lấy tất cả các từ, chèn từ và xóa tất cả các từ.

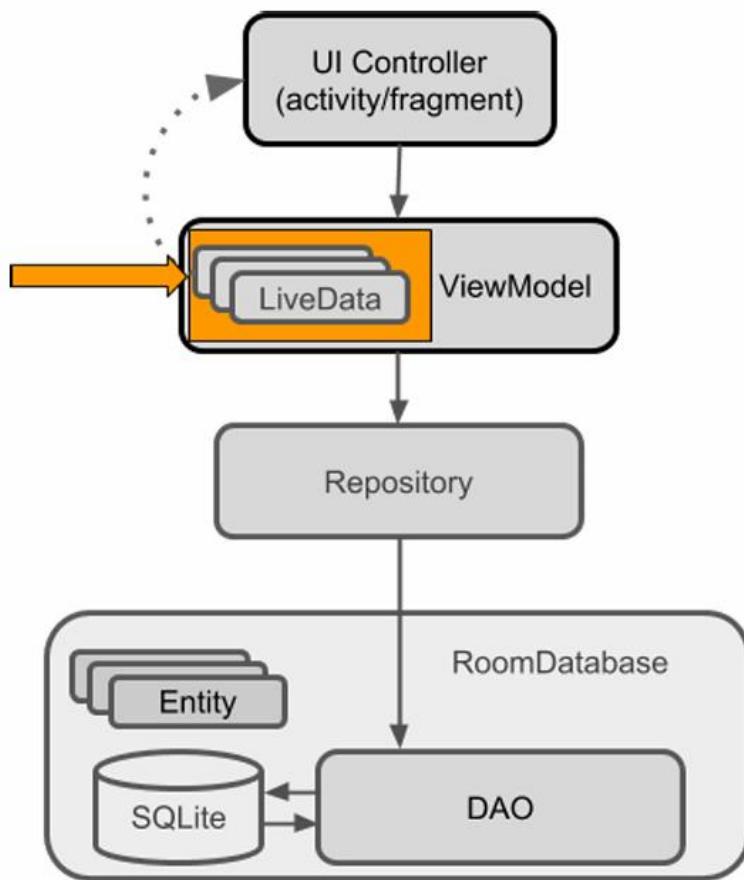
1. Tạo một giao diện mới và gọi nó là WordDao.
2. Chú thích khai báo lớp bằng @Dao để xác định lớp là lớp DAO cho Room.
3. Khai báo một phương thức để chèn một từ:
4. Chú thích phương thức insert() bằng @Insert . Bạn không cần phải cung cấp bất kỳ SQL nào! (Ngoài ra còn có chú thích @ Delete và @ Update để xóa và cập nhật một hàng, nhưng bạn không sử dụng các thao tác này trong phiên bản đầu tiên của ứng dụng này.)
5. Khai báo một phương thức để xóa tất cả các từ:
  1. Không có chú thích thuận tiện để xóa nhiều thực thể, vì vậy hãy chú thích phương thức deleteAll() bằng @Query chung . Cung cấp truy vấn SQL dưới dạng tham số chuỗi để @Query . Chú thích phương thức deleteAll() như sau:
  2. Tạo một phương thức có tên là getAllWords() trả về Danh sách các từ:
  3. Chú thích phương thức getAllWords() bằng một truy vấn SQL lấy tất cả các từ từ word\_table, được sắp xếp theo thứ tự bảng chữ cái để thuận tiện:

Dưới đây là mã hoàn chỉnh cho lớp WordDao:

Mẹo: Đối với ứng dụng này, việc sắp xếp các từ là không thực sự cần thiết. Tuy nhiên, theo mặc định, đơn đặt hàng trả lại không được đảm bảo và việc đặt hàng giúp việc kiểm tra trở nên đơn giản.

Để tìm hiểu thêm về DAO, hãy xem phần Truy cập dữ liệu bằng DAO trong phòng .

## Nhiệm vụ 4: Sử dụng LiveData



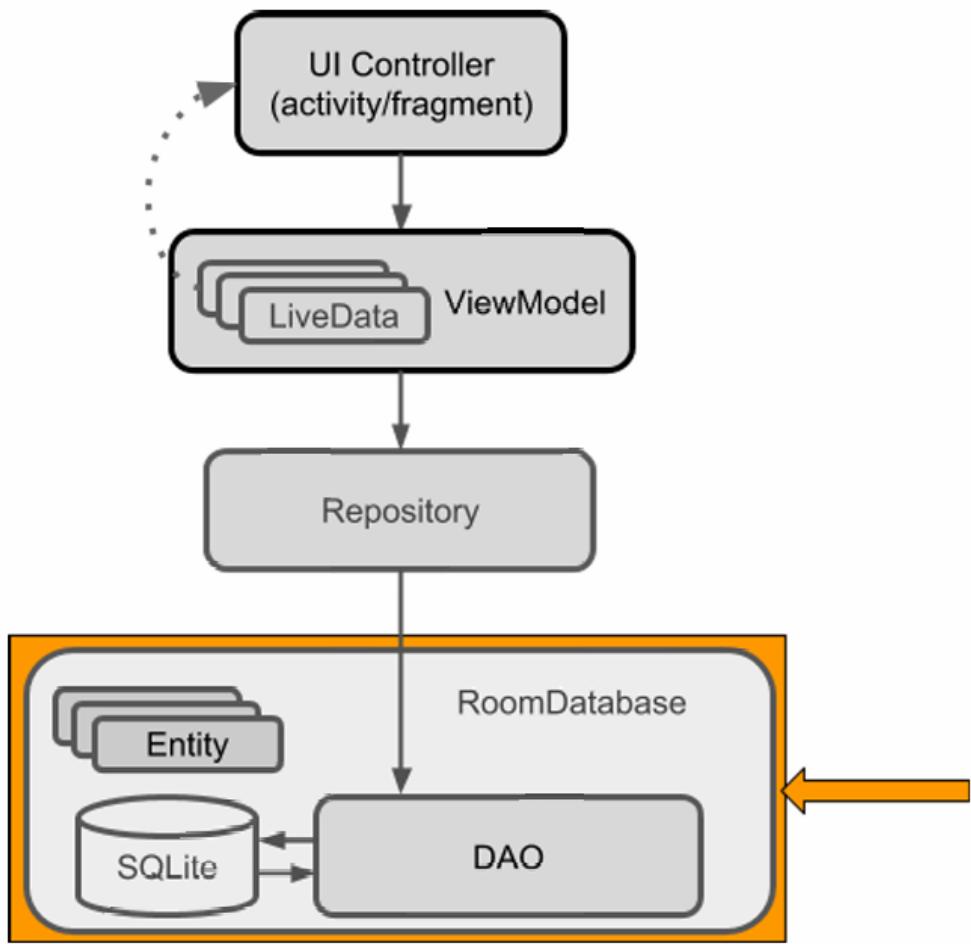
Khi hiển thị dữ liệu hoặc sử dụng dữ liệu theo những cách khác, bạn thường muốn thực hiện một số hành động khi dữ liệu thay đổi. Điều này có nghĩa là bạn phải quan sát dữ liệu để khi nó thay đổi, bạn có thể phản ứng. LiveData, là một lớp thư viện vòng đời để quan sát dữ liệu, có thể giúp ứng dụng của bạn phản hồi các thay đổi về dữ liệu. Nếu bạn sử dụng giá trị trả về thuộc loại LiveData trong phân mô tả phương thức của mình, Room sẽ tạo tất cả mã cần thiết để cập nhật LiveData khi cơ sở dữ liệu được cập nhật.

### 4.1 Trả về LiveData trong WordDao

- Trong giao diện WordDao, thay đổi chữ ký phương thức getAllWords() để Danh sách được trả về <Word> được bao bọc bằng LiveData<>

Xem tài liệu về LiveData để tìm hiểu thêm về các cách sử dụng LiveData khác hoặc xem video về Thành phần kiến trúc: LiveData và Vòng đời này.

## Nhiệm vụ 5: Thêm cơ sở dữ liệu Phòng



Room là một lớp cơ sở dữ liệu trên cơ sở dữ liệu SQLite. Room xử lý các tác vụ thông thường mà bạn đã sử dụng để xử lý với một lớp trợ giúp cơ sở dữ liệu như `SQLiteOpenHelper`.

- Room sử dụng DAO để đưa ra các truy vấn vào cơ sở dữ liệu của nó.
- Theo mặc định, để tránh hiệu suất giao diện người dùng kém, Room không cho phép bạn đưa ra truy vấn cơ sở dữ liệu trên luồng chính. LiveData áp dụng quy tắc này bằng cách tự động chạy truy vấn không đồng bộ trên luồng nền khi cần.
- Room cung cấp kiểm tra thời gian biên dịch các câu lệnh SQLite.
- Lớp Room của bạn phải trừu tượng và mở rộng RoomDatabase.
- Thông thường, bạn chỉ cần một phiên bản cơ sở dữ liệu Room cho toàn bộ ứng dụng.
- Room cung cấp kiểm tra thời gian biên dịch các câu lệnh SQLite.

- Lớp Room của bạn phải trùu tượng và mở rộng RoomDatabase.
- Thông thường, bạn chỉ cần một phiên bản cơ sở dữ liệu Room cho toàn bộ ứng dụng.

### 5.1 Triển khai cơ sở dữ liệu Room

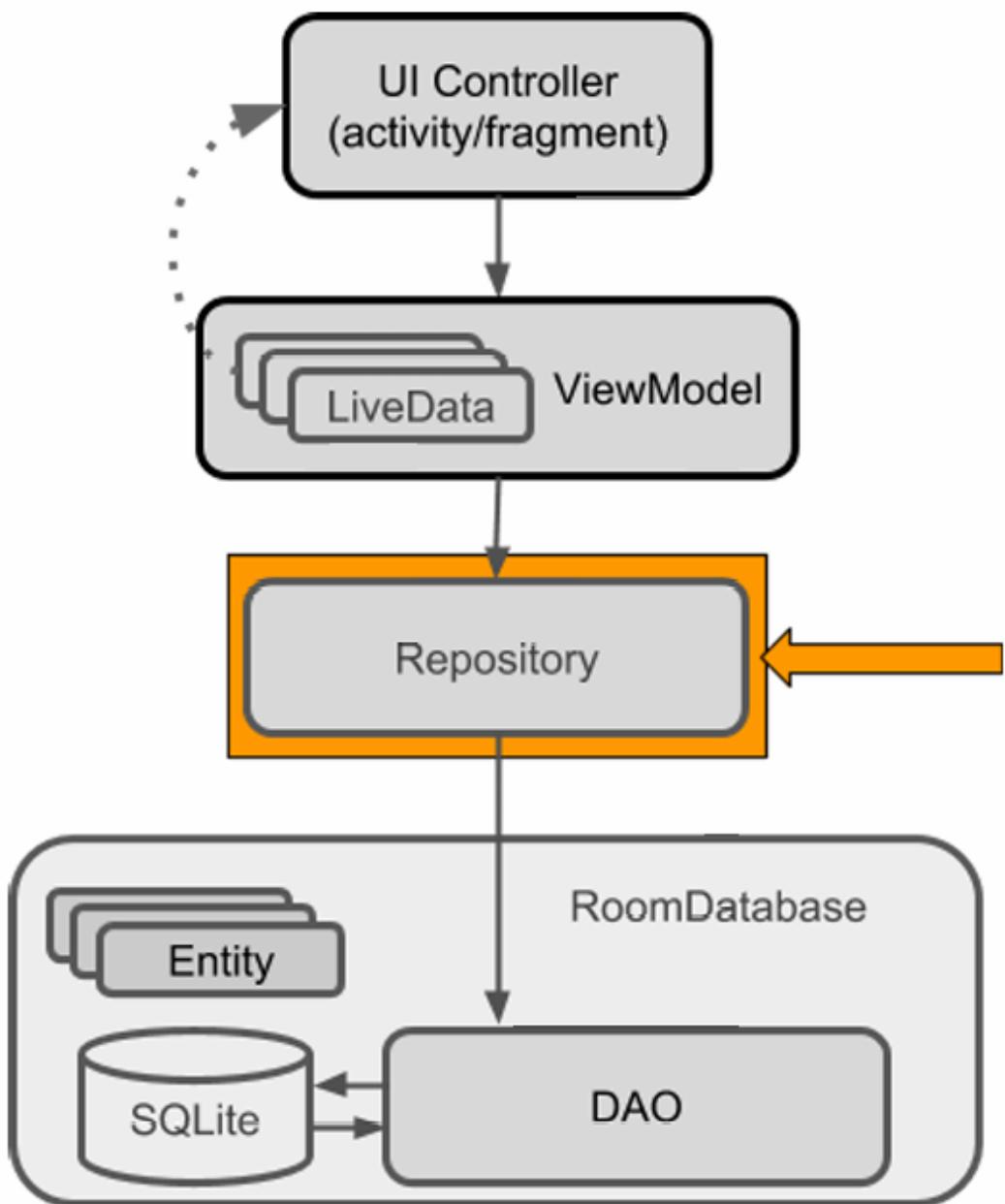
1. Tạo một lớp trùu tượng công khai mở rộng RoomDatabase và gọi nó là WordRoomDatabase .
2. Chú thích lớp thành cơ sở dữ liệu Phòng. Khai báo các thực thể thuộc về cơ sở dữ liệu — trong trường hợp này chỉ có một thực thể, Word . (Liệt kê các thực thể, lớp hoặc các lớp tạo ra các bảng tương ứng trong cơ sở dữ liệu.) Đặt số phiên bản.
3. Xác định các DAO hoạt động với cơ sở dữ liệu. Cung cấp một phương thức "getter" trùu tượng cho mỗi @Dao .
4. Tạo WordRoomDatabase dưới dạng singleton để ngăn nhiều phiên bản của cơ sở dữ liệu được mở cùng một lúc, điều này sẽ là một điều xấu. Dưới đây là mã để tạo singleton:
5. Thêm mã để tạo cơ sở dữ liệu được chỉ ra bởi nhận xét Tạo cơ sở dữ liệu ở đây trong mã trên. Mã sau sử dụng trình tạo cơ sở dữ liệu của Room để tạo đối tượng RoomDatabase có tên "word\_database" trong ngữ cảnh ứng dụng từ lớp WordRoomDatabase.
6. Thêm chiến lược di chuyển cho cơ sở dữ liệu. Trong thực tế này, bạn không cập nhật các thực thể và số phiên bản. Tuy nhiên, nếu bạn sửa đổi lược đồ cơ sở dữ liệu, bạn cần cập nhật số phiên bản và xác định cách xử lý di chuyển. Đối với một ứng dụng mẫu như ứng dụng bạn đang tạo, phá hủy và tạo lại cơ sở dữ liệu là một chiến lược di chuyển tốt. Đối với một ứng dụng thực, bạn phải thực hiện chiến lược di chuyển không phá hủy. Xem phần Tìm hiểu về di chuyển bằng Room .

Thêm mã sau vào trình tạo trước khi gọi build()

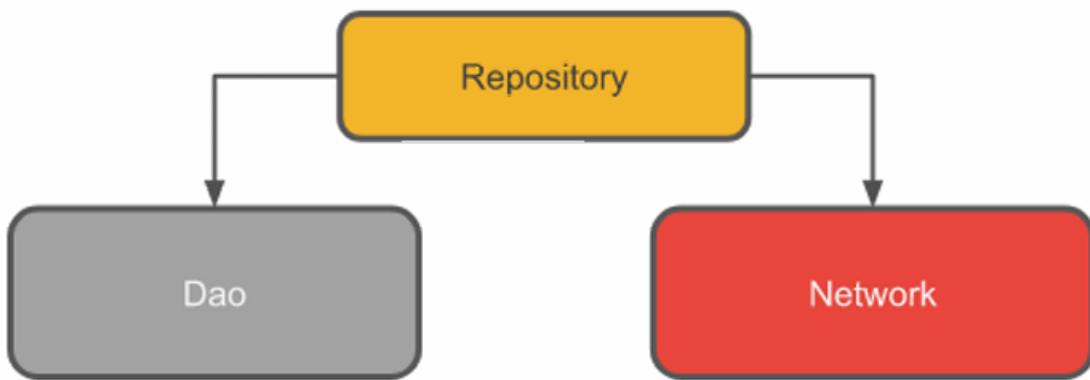
Dưới đây là mã hoàn chỉnh cho toàn bộ lớp WordRoomDatabase:

Lưu ý quan trọng: Trong Android Studio, nếu bạn gặp lỗi khi dán mã hoặc trong quá trình xây dựng, hãy đảm bảo bạn đang sử dụng tên gói đầy đủ để nhập. Xem Thêm thành phần vào dự án của bạn . Sau đó chọn Xây dựng > Clean Project . Sau đó chọn Xây dựng > Xây dựng lại dự án và xây dựng lại.

## Nhiệm vụ 6: Tạo kho lưu trữ



Kho lưu trữ là một lớp trừu tượng hóa quyền truy cập vào nhiều nguồn dữ liệu. Kho lưu trữ không phải là một phần của thư viện Thành phần Kiến trúc, nhưng là một phương pháp hay nhất được đề xuất để tách mã và kiến trúc. Lớp Repository xử lý các hoạt động dữ liệu. Nó cung cấp một API sạch cho phần còn lại của ứng dụng cho dữ liệu ứng dụng.



Kho lưu trữ quản lý các luồng truy vấn và cho phép bạn sử dụng nhiều phụ trợ. Trong ví dụ phổ biến nhất, Kho lưu trữ thực hiện logic để quyết định xem có nên tìm nạp dữ liệu từ mạng hay sử dụng kết quả được lưu trong bộ nhớ đệm trong cơ sở dữ liệu cục bộ.

## 6.1 Triển khai kho lưu trữ

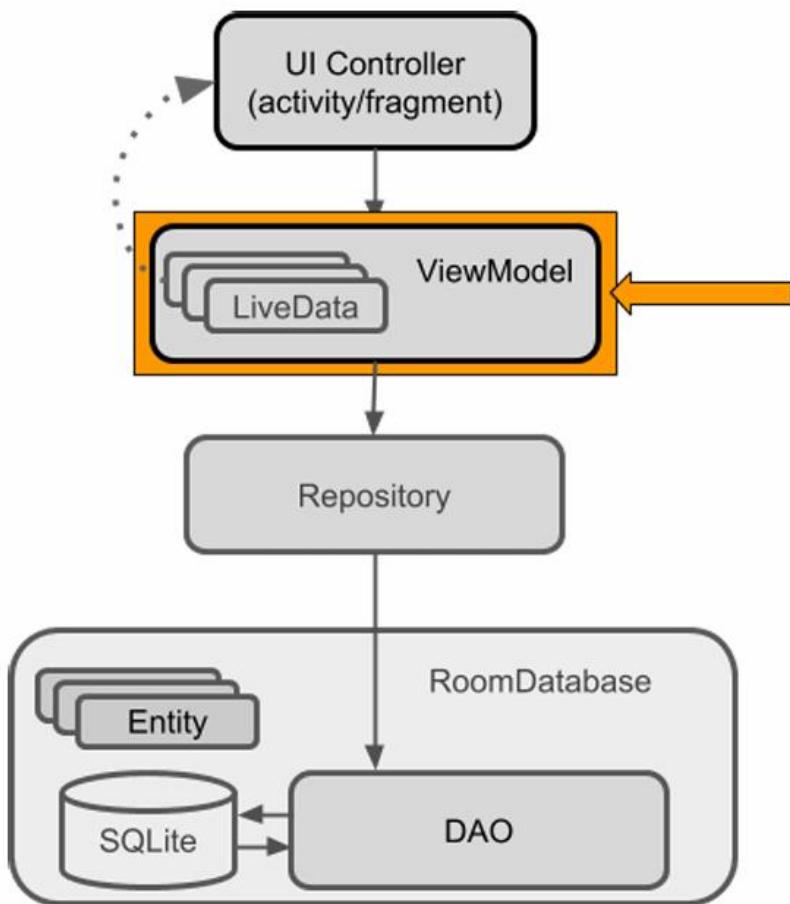
1. Tạo một lớp công khai có tên là WordRepository .
2. Thêm các biến thành viên cho DAO và danh sách các từ.
- 3.Thêm một hàm khởi tạo nhận được một xử lý vào cơ sở dữ liệu và khởi tạo các biến thành viên.
4. Thêm một phương thức bao bọc có tên là getAllWords() trả về các từ được lưu trong bộ nhớ đệm dưới dạng LiveData . Room thực thi tất cả các truy vấn trên một luồng riêng biệt. LiveData được quan sát sẽ thông báo cho người quan sát khi dữ liệu thay đổi.
5. Thêm một wrapper cho phương thức insert(). Sử dụng AsyncTask để gọi insert() trên luồng không phải giao diện người dùng, nếu không ứng dụng của bạn sẽ gặp sự cố. Room đảm bảo rằng bạn không thực hiện bất kỳ thao tác chạy lâu nào trên luồng chính, điều này sẽ chặn giao diện người dùng.

6. Tạo insertAsyncTask dưới dạng lớp bên trong. Bạn nên quen thuộc với AsyncTask, vì vậy đây là mã insertAsyncTask để bạn sao chép:

Dưới đây là mã hoàn chỉnh cho lớp WordRepository:

Lưu ý: Đối với ví dụ đơn giản này, Repository không làm được nhiều. Để triển khai phức tạp hơn, hãy xem mã BasicSample trên GitHub.

### Nhiệm vụ 7: Tạo ViewModel



ViewModel là một lớp có vai trò cung cấp dữ liệu cho giao diện người dùng và tồn tại sau các thay đổi cấu hình. ViewModel hoạt động như một trung tâm giao tiếp giữa Kho lưu trữ và giao diện người dùng. ViewModel là một phần của thư viện vòng đời . Để biết hướng dẫn giới thiệu về chủ đề này, hãy xem ViewModel.

ViewModel lưu trữ dữ liệu giao diện người dùng của ứng dụng theo cách tồn tại sau khi thay đổi cấu hình. Việc tách dữ liệu giao diện người dùng của ứng dụng khỏi các lớp Activity và Fragment cho phép bạn tuân thủ tốt hơn nguyên tắc trách nhiệm duy nhất: Hoạt động và phân đoạn của bạn chịu trách nhiệm về dữ liệu lên màn hình, trong khi ViewModel chịu trách nhiệm lưu giữ và xử lý tất cả dữ liệu cần thiết cho giao diện người dùng. Trong ViewModel , hãy sử dụng LiveData cho dữ liệu có thể thay đổi mà giao diện người dùng sẽ sử dụng hoặc hiển thị.

## 7.1 Triển khai WordViewModel

1. Tạo một lớp có tên là WordViewModel mở rộng AndroidViewModel .

Cảnh báo: Không bao giờ chuyển ngữ cảnh vào các phiên bản ViewModel. Không lưu trữ các phiên bản Hoạt động, Mảnh hoặc Chế độ xem hoặc Ngữ cảnh của chúng trong ViewModel .

Một Hoạt động có thể bị hủy và tạo nhiều lần trong vòng đời của ViewModel , chẳng hạn như khi thiết bị được xoay. Nếu bạn lưu trữ một tham chiếu đến Hoạt động trong ViewModel , bạn sẽ nhận được các tham chiếu trả đến Hoạt động đã bị hủy . Đây là một rò rỉ bộ nhớ. Nếu bạn cần ngữ cảnh ứng dụng, hãy sử dụng AndroidViewModel , như được hiển thị trong thực tế này. </div>

2. Thêm một biến thành viên riêng để giữ tham chiếu đến Kho lưu trữ.

3. Thêm một biến thành viên LiveData riêng tư để lưu danh sách các từ.
4. Thêm một hàm khởi tạo nhận tham chiếu đến WordRepository và lấy danh sách tất cả các từ từ WordRepository .
5. Thêm một phương thức "getter" để lấy tất cả các từ. Điều này hoàn toàn ẩn quá trình triển khai khỏi giao diện người dùng.
6. Tạo một phương thức wrapper insert() t hat gọi phương thức insert() của Repository. Bằng cách này, việc triển khai insert() hoàn toàn bị ẩn khỏi giao diện người dùng.



Đây là mã hoàn chỉnh cho WordViewModel

Nhiệm vụ 8: Thêm bộ cục XML cho giao diện người dùng

Tiếp theo, thêm bộ cục XML cho danh sách và các mục sẽ được hiển thị trong RecyclerView .

Thực tế này giả định rằng bạn đã quen thuộc với việc tạo bộ cục trong XML, vì vậy mã chỉ được cung cấp.

#### 8.1 Thêm kiểu

1. Thay đổi màu trong colors.xml thành như sau: (để sử dụng màu Material Design):

2. Thêm kiểu cho chế độ xem văn bản trong tệp giá trị / styles.xml:

#### 8.2 Thêm bộ cục mục

- Thêm bộ cục / bộ cục recyclerview\_item.xml:

#### 8.3 Thêm RecyclerView

1. Trong tệp layout/content\_main.xml, thêm màu nền vào ConstraintLayout

2. Trong tệp content\_main.xml, thay thế phần tử TextView bằng phần tử RecyclerView:

#### 8.4 Sửa biểu tượng trong FAB

Biểu tượng trong nút hành động nổi (FAB) của bạn phải tương ứng với hành động có sẵn. Trong tệp layout/activity\_main.xml, hãy cung cấp cho FloatingActionButton một biểu tượng biểu tượng +:

1. Chọn File > New > Vector Asset .
2. Chọn Biểu tượng vật liệu.
3. Nhập vào biểu tượng rô bốt Android trong trường Biểu tượng:, sau đó chọn nội dung + ("thêm").
4. Trong tệp layout/activity\_main.xml, trong FloatingActionButton, thay đổi thuộc tính srcCompat thành:

### Nhiệm vụ 9: Tạo Bộ điều hợp và thêm RecyclerView

Bạn sẽ hiển thị dữ liệu trong RecyclerView , điều này tốt hơn một chút so với việc chỉ ném dữ liệu vào TextView . Thực tế này giả định rằng bạn biết cách hoạt động của RecyclerView, RecyclerView.LayoutManager, RecyclerView.ViewHolder và RecyclerView.Adapter.

#### 9.1 Tạo lớp WordListAdapter

- Thêm một lớp WordListAdapter mở rộng RecyclerView.Adapter . Bộ điều hợp lưu dữ liệu vào bộ nhớ đệm và điền vào RecyclerView với dữ liệu đó. Lớp bên trong WordViewHolder giữ và quản lý một dạng xem cho một mục danh sách.

Lưu ý: Biến mWords trong bộ điều hợp lưu dữ liệu vào bộ nhớ đệm. Trong tác vụ tiếp theo, bạn thêm mã tự động cập nhật dữ liệu.

Lưu ý: Phương thức getItemCount() cần tính đến khả năng dữ liệu chưa sẵn sàng và mWords vẫn là null . Trong một ứng dụng phức tạp hơn, bạn có thể hiển thị dữ liệu giữ chỗ hoặc thứ gì đó khác có ý nghĩa đối với người dùng.

#### 9.2 Thêm RecyclerView vào MainActivity

1. Thêm RecyclerView vào phương thức onCreate() của MainActivity
1. Chạy ứng dụng của bạn để đảm bảo ứng dụng biên dịch và chạy. Không có mục, vì bạn chưa kết nối dữ liệu. Ứng dụng sẽ hiển thị chế độ xem người tái chế trống.

### Nhiệm vụ 10: Đien vào cơ sở dữ liệu

Chưa có dữ liệu nào trong cơ sở dữ liệu. Bạn sẽ thêm dữ liệu theo hai cách: Thêm một số dữ liệu khi cơ sở dữ liệu được mở và thêm Hoạt động để thêm từ. Mỗi khi cơ sở dữ liệu được mở, tất cả nội dung bị xóa và đặt lại. Đây là một giải pháp hợp lý cho một ứng dụng mẫu, nơi bạn thường muốn khởi động lại trên một bảng sạch.

10.1 Tạo lệnh gọi lại để điền vào cơ sở dữ liệu Để xóa tất cả nội dung và điền lại cơ sở dữ liệu bất cứ khi nào ứng dụng được khởi động, bạn tạo RoomDatabase.Callback và ghi đè phương thức onOpen(). Vì bạn không thể thực hiện các thao tác cơ sở dữ liệu Room trên luồng giao diện người dùng, onOpen() tạo và thực thi AsyncTask để thêm nội dung vào cơ sở dữ liệu.

1. Thêm lệnh gọi lại onOpen() vào lớp WordRoomDatabase:

1. Tạo một lớp bên trong PopulateDbAsync mở rộng AsyncTask . Triển khai phương thức doInBackground() để xóa tất cả các từ, sau đó tạo các từ mới. Đây là mã cho AsyncTask xóa nội dung của cơ sở dữ liệu, sau đó điền vào nó với một danh sách từ ban đầu. Hãy thoải mái sử dụng từ ngữ của riêng bạn!

1. Thêm lệnh gọi lại vào trình tự xây dựng cơ sở dữ liệu trong WordRoomDatabase, ngay trước khi bạn gọi .build() :

### Nhiệm vụ 11: Kết nối giao diện người dùng với dữ liệu

Bây giờ bạn đã tạo phương thức để điền vào cơ sở dữ liệu với tập hợp từ ban đầu, bước tiếp theo là thêm mã để hiển thị các từ đó trong RecyclerView .

Để hiển thị nội dung hiện tại của cơ sở dữ liệu, bạn thêm một trình quan sát LiveData trong ViewModel . Bất cứ khi nào dữ liệu thay đổi (bao gồm cả khi dữ liệu được khởi tạo), lệnh gọi lại onChanged() sẽ được gọi. Trong trường hợp này, lệnh gọi lại onChanged() gọi phương thức setWord() của bộ điều hợp để cập nhật dữ liệu được lưu trong bộ nhớ cache của bộ điều hợp và làm mới danh sách được hiển thị.

#### 11.1 Hiển thị các từ

1. Trong MainActivity , tạo một biến thành viên cho ViewModel , vì tất cả các tương tác của hoạt động chỉ với WordViewModel.

1. Trong phương thức onCreate(), lấy ViewModel từ lớp ViewModelProviders.

Sử dụng ViewModelProviders để liên kết ViewModel với bộ điều khiển giao diện người dùng. Khi ứng dụng của bạn khởi động lần đầu tiên, lớp ViewModelProviders sẽ tạo ViewModel . Khi hoạt động bị hủy, chẳng hạn như thông qua thay đổi cấu hình, ViewModel vẫn tồn tại. Khi hoạt động được tạo lại, ViewModelProviders sẽ trả về ViewModel hiện có. Xem ViewModel .

1. Cũng trong onCreate() , thêm một trình quan sát cho LiveData do getAllWords() trả về . Khi dữ liệu quan sát được thay đổi trong khi hoạt động ở nền trước, phương thức onChanged() sẽ được gọi và cập nhật dữ liệu được lưu vào bộ nhớ đệm trong bộ điều hợp. Lưu ý rằng trong trường hợp này, khi ứng dụng mở ra, dữ liệu ban đầu sẽ được thêm vào, vì vậy phương thức onChanged() sẽ được gọi.

2. Run ứng dụng. Tập hợp các từ ban đầu xuất hiện trong RecyclerView .

Nhiệm vụ 12: Tạo Hoạt động để thêm từ

Bây giờ bạn sẽ thêm một Hoạt động cho phép người dùng sử dụng FAB để nhập các từ mới. Đây là giao diện cho hoạt động mới sẽ trông như thế nào:

## 12.1 Tạo NewWordActivity

1. Thêm các tài nguyên chuỗi này vào tệp values/strings.xml:

1. Thêm kiểu cho các nút trong giá trị / styles.xml

1. Sử dụng mẫu Hoạt động trống để tạo một hoạt động mới, NewWordActivity. Xác minh rằng hoạt động đã được thêm vào Tệp kê khai Android.

2. Cập nhật tệp activity\_new\_word.xml trong thư mục bô cục:

3. Triển khai lớp NewWordActivity. Mục tiêu là khi người dùng nhấn nút Lưu, từ mới được đưa vào Ý định để được gửi trở lại Hoạt động mẹ. Dưới đây là mã cho hoạt động NewWordActivity:

## 12.2 Thêm mã để chèn một từ vào cơ sở dữ liệu

1. Trong MainActivity , thêm lệnh gọi lại onActivityResult() cho NewWordActivity . Nếu hoạt động trả về với RESULT\_OK , hãy chèn từ được trả về vào cơ sở dữ liệu bằng cách gọi phương thức insert() của WordViewModel

1. Xác định mã yêu cầu còn thiếu:

1. Trong MainActivity, khởi động NewWordActivity khi người dùng nhấn vào FAB. Thay thế mã trong trình xử lý nhấp chuột onClick() của FAB bằng mã sau:

1. Chạy ứng dụng của bạn. Khi bạn thêm một từ vào cơ sở dữ liệu trong NewWordActivity , giao diện người dùng sẽ tự động cập nhật.

2. Thêm một từ đã tồn tại trong danh sách. Điều gì xảy ra? Ứng dụng của bạn có gặp sự cố không? Ứng dụng của bạn sử dụng chính từ làm khóa chính và mỗi khóa chính phải là duy nhất. Bạn có thể chỉ định chiến lược xung đột để cho ứng dụng biết phải làm gì khi người dùng cố gắng thêm một từ hiện có.

3. Trong giao diện WordDao, thay đổi chú thích cho phương thức insert() thành:

Để tìm hiểu về các chiến lược xung đột khác, hãy xem tài liệu tham khảo OnConflictStrategy.

4. Chạy lại ứng dụng của bạn và thử thêm một từ đã tồn tại. Điều gì xảy ra bây giờ?

Mã giải pháp dự án Android Studio: RoomWordsSample

## Tóm tắt

Bây giờ bạn đã có một ứng dụng hoạt động, hãy tóm tắt lại những gì bạn đã xây dựng. Đây là cấu trúc ứng dụng một lần nữa, ngay từ đầu:

- Bạn có một ứng dụng hiển thị các từ trong danh sách (MainActivity , RecyclerView , WordListAdapter ).

- Bạn có thể thêm từ vào danh sách (NewWordActivity).
- Một từ là một phiên bản của lớp thực thể Word.
- Các từ được lưu vào bộ nhớ đệm trong RecyclerViewAdapter dưới dạng Danh sách các từ (mWords). Danh sách được tự động cập nhật và hiển thị lại khi dữ liệu thay đổi.
- Cập nhật tự động xảy ra vì trong MainActivity , có một Người quan sát quan sát các từ và được thông báo khi các từ thay đổi. Khi có thay đổi, phương thức onChange() của người quan sát được thực thi và cập nhật mWords trong WordListAdapter .
- Dữ liệu có thể được quan sát vì nó là LiveData. Và những gì được quan sát là LiveData<List<Word>> được trả về bởi đối tượng WordViewModel.
- WordViewModel ẩn mọi thứ về backend khỏi giao diện người dùng. Nó cung cấp các phương thức để truy cập dữ liệu giao diện người dùng và trả về LiveData để MainActivity có thể thiết lập mối quan hệ người quan sát. Chế độ xem, hoạt động và phân đoạn chỉ tương tác với dữ liệu thông qua ViewModel . Như vậy, dữ liệu đến từ đâu không quan trọng.
- Trong trường hợp này, dữ liệu đến từ Kho lưu trữ. ViewModel không cần biết Kho lưu trữ đó tương tác với những gì. Nó chỉ cần biết cách tương tác với Repository, đó là thông qua các phương thức do Repository hiển thị.
- Kho lưu trữ quản lý một hoặc nhiều nguồn dữ liệu. Trong ứng dụng RoomWordsSample, phần phụ trợ đó là cơ sở dữ liệu Room. Room là một trình bao bọc xung quanh và triển khai cơ sở dữ liệu SQLite. Room làm rất nhiều việc cho bạn mà bạn đã từng phải tự làm. Ví dụ: Room thực hiện mọi thứ mà bạn đã sử dụng lớp SQLiteOpenHelper để thực hiện.
- DAO ánh xạ các lệnh gọi phương thức đến các truy vấn cơ sở dữ liệu, để khi Kho lưu trữ gọi một phương thức như getAllWords() , Room có thể thực thi SELECT \* từ word\_table ORDER BY từ ASC.
- Kết quả được trả về từ truy vấn được quan sát LiveData . Do đó, mỗi khi dữ liệu trong Room thay đổi, phương thức onChanged() của giao diện Người quan sát sẽ được thực thi và giao diện người dùng sẽ được cập nhật.

Khái niệm liên quan Tài liệu về khái niệm liên quan có trong

10.1: Room, LiveData và ViewModel .

Tìm hiểu thêm Để tiếp tục làm việc với ứng dụng RoomWordsSample và tìm hiểu thêm các cách sử dụng cơ sở dữ liệu Room, hãy xem lớp học lập trình 10.1 Phần B: Room, LiveData và ViewModel, bài tập này sẽ tiếp tục bước vào lớp học lập trình này. Tài liệu dành cho nhà phát triển Android:

- Hướng dẫn về kiến trúc ứng dụng
- Thêm các thành phần vào dự án của bạn
- DAO
- DAO trong phòng
- Tài liệu tham khảo tóm tắt gói phòng
- Xử lý vòng đời bằng các thành phần nhận biết vòng đời
- LiveData
- LiveData có thể thay đổi
- ViewModel
- ViewModelProviders
- Xác định dữ liệu bằng thực thể

Room Blog và bài viết:

- 7 bước đến Room (di chuyển ứng dụng hiện có)
- Tìm hiểu về quá trình di chuyển bằng Room
- Lớp học lập trình Tải dữ liệu nhận biết vòng đời với các thành phần kiến trúc: •  
Lớp học lập trình Android Persistence (LiveData, Room, DAO)
- Lớp học lập trình các thành phần nhận biết vòng đời của Android (ViewModel, LiveData, LifecycleOwner, LifecycleRegistryOwner)

Video:

- Tổng quan về các thành phần kiến trúc
- Các thành phần kiến trúc: LiveData và Vòng đời
- Các thành phần kiến trúc:

Mẫu mã ViewModel:

- Mẫu mã Thành phần kiến trúc

- BasicSample (mẫu không cơ bản nhưng toàn diện)

Bài tập về nhà

Xây dựng và chạy ứng dụng

Tạo ứng dụng sử dụng cơ sở dữ liệu Room, ViewModel và LiveData để hiển thị dữ liệu khi dữ liệu thay đổi. Bạn có thể làm cho việc này đơn giản hoặc phức tạp tùy thích, miễn là ứng dụng sử dụng tất cả các thành phần cần thiết và dữ liệu cập nhật trên màn hình khi dữ liệu thay đổi trong cơ sở dữ liệu.

Dưới đây là một số gợi ý và ý tưởng:

- Tạo một ứng dụng đơn giản lưu trữ một tài liệu văn bản và hiển thị nội dung của tài liệu trong TextView . Khi người dùng chỉnh sửa tài liệu, các thay đổi sẽ xuất hiện trong TextView .
- Tạo một ứng dụng trả lời câu hỏi. Bắt đầu chỉ với các câu hỏi và cho phép người dùng thêm câu hỏi và câu trả lời mới.
- Như một thử thách, hãy thêm một nút vào mỗi câu trả lời trong ứng dụng câu hỏi để hiển thị thông tin bổ sung được lưu trữ trong một kho lưu trữ khác. Thông tin có thể đến từ một tệp trên thiết bị hoặc từ một trang trên internet.

Trả lời các câu hỏi sau:

Câu hỏi 1: Ưu điểm của việc sử dụng cơ sở dữ liệu Room là gì?

- Tạo và quản lý cơ sở dữ liệu Android SQLite cho bạn.
- Loại bỏ rất nhiều mã boilerplate.
- Giúp bạn quản lý nhiều phần phụ trợ.
  - Sử dụng DAO, cung cấp một cơ chế để ánh xạ các phương thức Java với các truy vấn cơ sở dữ liệu.

Câu hỏi 2: Lý do nào sau đây để sử dụng ViewModel?

- Tách giao diện người dùng khỏi phần phụ trợ một cách rõ ràng.
- Thường được sử dụng với LiveData cho dữ liệu có thể thay đổi mà giao diện người dùng sẽ sử dụng hoặc hiển thị.
- Ngăn dữ liệu của bạn bị mất khi ứng dụng gấp sự cố.
- Hoạt động như một trung tâm giao tiếp giữa Kho lưu trữ và giao diện người dùng.

- Các phiên bản ViewModel vẫn tồn tại các thay đổi cấu hình thiết bị. Câu hỏi 3: DAO là gì?
  - Viết tắt của "đối tượng truy cập dữ liệu".
  - Một thư viện để quản lý các truy vấn cơ sở dữ liệu.
  - Giao diện có chú thích ánh xạ các phương thức Java với các truy vấn SQLite. • Một lớp có các phương thức luôn chạy trong nền, không phải trên luồng chính.
  - Một lớp mà trình biên dịch kiểm tra lỗi SQL, sau đó sử dụng để tạo truy vấn từ các chú thích.

Câu hỏi 4: Các tính năng của LiveData là gì?

- Khi LiveData được sử dụng với Room, dữ liệu sẽ tự động cập nhật khi tất cả các cấp trung gian cũng trả về LiveData (DAO, ViewModel, Repository).
- Sử dụng mẫu quan sát và thông báo cho người quan sát khi dữ liệu của nó thay đổi.
- Tự động cập nhật giao diện người dùng khi nó thay đổi.
- Nhận thức về vòng đời.

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kiểm tra xem ứng dụng có các tính năng sau không:

- Sử dụng cơ sở dữ liệu Phòng để lưu trữ dữ liệu.
- Sử dụng ViewModel và LiveData để quản lý và hiển thị dữ liệu thay đổi.
- Có cách chỉnh sửa dữ liệu và hiển thị các thay đổi.

## 2.2 Room, LiveData và ViewModel

### Giới thiệu

Lớp học lập trình này (thực hành) tiếp nối từ 10.1 Phần A: Room, LiveData và ViewModel. Lớp học lập trình này cung cấp cho bạn nhiều thực hành hơn về cách sử dụng API do thư viện Room cung cấp để triển khai chức năng cơ sở dữ liệu. Bạn sẽ thêm khả năng xóa các mục cũ khỏi cơ sở dữ liệu. Lớp học lập trình này cũng bao

gồm một thử thách mã hóa, trong đó bạn cập nhật ứng dụng để người dùng có thể chỉnh sửa dữ liệu hiện có.

### Những điều bạn nên biết

Bạn sẽ có thể tạo và chạy ứng dụng trong Android Studio 3.0 trở lên. Đặc biệt, hãy làm quen với những điều sau:

- Sử dụng RecyclerView và bộ điều hợp.
- Sử dụng các lớp thực thể, đối tượng truy cập dữ liệu (DAO) và RoomDatabase để lưu trữ và truy xuất dữ liệu trong cơ sở dữ liệu SQLite tích hợp sẵn của Android. Bạn đã tìm hiểu các chủ đề này trong phần 10.1 Phần A: Room, LiveData và ViewModel .

### Những gì bạn sẽ học

- Cách điền dữ liệu vào cơ sở dữ liệu chỉ khi cơ sở dữ liệu trống (để người dùng không bị mất các thay đổi mà họ đã thực hiện đối với dữ liệu).
- Cách xóa dữ liệu khỏi cơ sở dữ liệu Room.
- Cách cập nhật dữ liệu hiện có (nếu bạn xây dựng ứng dụng thử thách).

### Những gì bạn sẽ làm

- Cập nhật ứng dụng RoomWordsSample để giữ dữ liệu khi ứng dụng đóng.
- Cho phép người dùng xóa tất cả các từ bằng cách chọn mục menu Tùy chọn. • Cho phép người dùng xóa một từ cụ thể bằng cách vuốt một mục trong danh sách.
- Tùy chọn, trong thử thách mã hóa, hãy mở rộng ứng dụng để cho phép người dùng cập nhật các từ hiện có.

### Tổng quan về ứng dụng

Bạn sẽ mở rộng ứng dụng RoomWordsSample mà bạn đã tạo trong lớp học lập trình trước. Cho đến nay, ứng dụng đó hiển thị danh sách các từ và người dùng có thể thêm từ. Khi ứng dụng đóng và mở lại, ứng dụng sẽ khởi tạo lại cơ sở dữ liệu. Các từ mà người dùng đã thêm vào sẽ bị mất. Trong thực tế này, bạn mở rộng ứng dụng để nó chỉ khởi tạo dữ liệu trong cơ sở dữ liệu nếu không có dữ liệu hiện có.

Sau đó, bạn thêm một mục menu cho phép người dùng xóa tất cả dữ liệu.

Bạn cũng cho phép người dùng vuốt một từ để xóa nó khỏi cơ sở dữ liệu.

## Nhiệm vụ 1: Chỉ khởi tạo dữ liệu nếu cơ sở dữ liệu trống

Ứng dụng RoomWordsSample mà bạn đã tạo trong thực tế trước đó sẽ xóa và tạo lại dữ liệu bất cứ khi nào người dùng mở ứng dụng. Hành vi này không lý tưởng, vì người dùng sẽ muốn các từ đã thêm vào của họ vẫn còn trong cơ sở dữ liệu khi ứng dụng đóng. (Mã giải pháp cho thực tế trước đó có trong GitHub.) Trong tác vụ này, bạn cập nhật ứng dụng để khi nó mở, tập dữ liệu ban đầu chỉ được thêm vào nếu cơ sở dữ liệu không có dữ liệu.

Để phát hiện xem cơ sở dữ liệu đã chứa dữ liệu hay chưa, bạn có thể chạy truy vấn để lấy một mục dữ liệu. Nếu truy vấn không trả về gì, thì cơ sở dữ liệu trống.

Lưu ý: Trong ứng dụng chính thức, bạn có thể muốn cho phép người dùng xóa tất cả dữ liệu mà không cần khởi tạo lại dữ liệu khi ứng dụng khởi động lại. Nhưng đối với mục đích thử nghiệm, sẽ rất hữu ích khi có thể xóa tất cả dữ liệu, sau đó khởi tạo lại dữ liệu khi ứng dụng khởi động.

1.1 Thêm một phương thức vào DAO để lấy một từ duy nhất Hiện tại, giao diện WordDao có một phương thức để lấy tất cả các từ, nhưng không phải để lấy bất kỳ từ nào. Phương thức để lấy một từ không cần trả về LiveData , vì ứng dụng của bạn sẽ gọi phương thức này một cách rõ ràng khi cần.

1. Trong giao diện WordDao, thêm một phương thức để lấy bất kỳ từ nào:

Room đưa ra truy vấn cơ sở dữ liệu khi phương thức getAnyWord() được gọi và trả về một mảng chứa một từ. Bạn không cần phải viết thêm bất kỳ mã nào để triển khai nó.

1.2 Cập nhật phương thức khởi tạo để kiểm tra xem dữ liệu có tồn tại hay không Sử dụng phương thức getAnyWord() trong phương thức khởi tạo cơ sở dữ liệu. Nếu có bất kỳ dữ liệu nào, hãy giữ nguyên dữ liệu. Nếu không có dữ liệu, hãy thêm tập dữ liệu ban đầu.

1. PopulateDBAsync là một lớp bên trong WordRoomDatabase . Trong PopulateDBAsync, cập nhật phương thức doInBackground() để kiểm tra xem cơ sở dữ liệu có bất kỳ từ nào trước khi khởi tạo dữ liệu hay không:

2. Chạy ứng dụng của bạn và thêm một số từ mới. Đóng ứng dụng và khởi động lại. Bạn sẽ thấy các từ mới mà bạn đã thêm, vì các từ bây giờ sẽ tồn tại khi ứng dụng được đóng và mở lại.

## Nhiệm vụ 2: Xóa tất cả các từ

Trong thực tế trước, bạn đã sử dụng phương thức deleteAll() để xóa tất cả dữ liệu khi cơ sở dữ liệu được mở. Phương thức deleteAll() chỉ được gọi từ lớp PopulateDbAsync khi ứng dụng khởi động. Nay giờ, bạn sẽ cung cấp phương thức deleteAll() thông qua ViewModel để ứng dụng của bạn có thể gọi phương thức bất cứ khi nào cần.

Dưới đây là các bước chung để triển khai một phương thức sử dụng thư viện Room để tương tác với cơ sở dữ liệu:

- Thêm phương thức vào DAO và chú thích nó với hoạt động cơ sở dữ liệu có liên quan. Đối với phương thức deleteAll(), bạn đã thực hiện bước này trong thực tế trước.
- Thêm phương thức vào lớp WordRepository. Viết mã để chạy phương thức trong nền.
- Để gọi phương thức trong lớp WordRepository, hãy thêm phương thức vào WordViewModel. Phần còn lại của ứng dụng sau đó có thể truy cập phương thức thông qua WordViewModel .

### 2.1 Thêm deleteAll() vào giao diện WordDao và chú thích nó

1. Trong WordDao, hãy kiểm tra xem phương thức deleteAll() đã được xác định và chú thích bằng SQL chạy khi phương thức thực thi chưa:

1.2 Thêm deleteAll() vào lớp WordRepository Thêm phương thức deleteAll() vào WordRepository và triển khai AsyncTask để xóa tất cả các từ trong nền.

Trong WordRepository , xác định deleteAllWordsAsyncTask làm lớp bên trong. Triển khai doInBackground() để xóa tất cả các từ bằng cách gọi deleteAll() trên DAO:

Trong lớp WordRepository, thêm phương thức deleteAll() để gọi AsyncTask mà bạn đã xác định.

1.3 Thêm deleteAll() vào lớp WordViewModel Làm cho phương thức deleteAll() có sẵn cho MainActivity bằng cách thêm phương thức đó vào WordViewModel .

1. Trong lớp WordViewModel, thêm phương thức deleteAll():

## Nhiệm vụ 3: Thêm một mục menu

Tùy chọn để xóa tất cả dữ liệu Tiếp theo, bạn thêm một mục menu để cho phép người dùng gọi deleteAll().

Lưu ý: Phiên bản chính thức của ứng dụng phải cung cấp các biện pháp bảo vệ để người dùng không vô tình xóa toàn bộ cơ sở dữ liệu của họ. Tuy nhiên, trong khi bạn phát triển ứng dụng của mình, sẽ rất hữu ích khi có thể xóa dữ liệu thử nghiệm một cách nhanh chóng. Điều này đặc biệt đúng khi ứng dụng của bạn không xóa dữ liệu khi ứng dụng mở ra.

### 3.1 Thêm tùy chọn menu Xóa tất cả dữ liệu

1. Trong menu\_main.xml , thay đổi tùy chọn menu title và id , như sau:
2. Trong MainActivity , triển khai phương thức onOptionsItemSelected() để gọi phương thức deleteAll() trên đối tượng WordViewModel.
3. Chạy ứng dụng của bạn. Trong menu Tùy chọn, chọn Xóa tất cả dữ liệu . Tất cả các từ nên biến mất.
4. Khởi động lại ứng dụng. (Khởi động lại ứng dụng từ thiết bị của bạn hoặc trình giả lập; không chạy lại từ Android Studio) Bạn sẽ thấy bộ từ ban đầu.

Lưu ý: Sau khi bạn xóa dữ liệu, việc triển khai lại ứng dụng từ Android Studio sẽ hiển thị lại tập dữ liệu ban đầu. Mở ứng dụng sẽ hiển thị tập dữ liệu trống.

## Nhiệm vụ 4: Xóa một từ

Ứng dụng của bạn cho phép người dùng thêm từ và xóa tất cả các từ. Trong Tác vụ 4 và 5, bạn mở rộng ứng dụng để người dùng có thể xóa một từ bằng cách vuốt mục trong RecyclerView . Một lần nữa, đây là các bước chung để triển khai phương thức sử dụng thư viện Room để tương tác với cơ sở dữ liệu:

- Thêm phương thức vào DAO và chú thích nó với hoạt động cơ sở dữ liệu có liên quan.
- Thêm phương thức vào lớp WordRepository. Viết mã để chạy phương thức trong nền.

- Để gọi phương thức trong lớp WordRepository, hãy thêm phương thức vào WordViewModel. Phần còn lại của ứng dụng sau đó có thể truy cập phương thức thông qua WordViewModel .

#### 4.1 Thêm deleteWord() vào DAO và chú thích nó

1. Trong WordDao , thêm phương thức deleteWord():

Bởi vì thao tác này xóa một hàng duy nhất, chú thích @Delete là tất cả những gì cần thiết để xóa từ khỏi cơ sở dữ liệu.

#### 4.2 Thêm deleteWord() vào lớp WordRepository

1. Trong WordRepository , xác định một AsyncTask khác có tên là deleteWordAsyncTask làm lớp bên trong. Triển khai doInBackground() để xóa một từ bằng cách gọi deleteWord() trên DAO:

2. Trong WordRepository , thêm phương thức deleteWord() để gọi AsyncTask mà bạn đã xác định.

#### 4.3 Thêm deleteWord() vào lớp WordViewModel Để cung cấp phương thức deleteWord() cho các lớp khác trong ứng dụng, cụ thể là MainActivity, hãy thêm phương thức đó vào WordViewModel .

1. Trong WordViewModel , thêm phương thức deleteWord():

Bây giờ bạn đã thực hiện logic để xóa một từ. Cho đến nay, không có cách nào để gọi thao tác delete-word từ giao diện người dùng của ứng dụng. Bạn sửa chữa điều đó tiếp theo.

#### Nhiệm vụ 5: Cho phép người dùng vuốt từ đi

Trong nhiệm vụ này, bạn thêm chức năng để cho phép người dùng vuốt một mục trong RecyclerView để xóa mục đó. Sử dụng lớp ItemTouchHelper do Thư viện hỗ trợ Android (phiên bản 7 trở lên) cung cấp để triển khai chức năng vuốt trong RecyclerView . Lớp ItemTouchHelper có các phương thức sau:

- onMove() được gọi khi người dùng di chuyển mục. Bạn sẽ không triển khai bất kỳ chức năng di chuyển nào trong ứng dụng này.

- `onSwipe()` được gọi khi người dùng vuốt mục. Bạn thực hiện phương pháp này để xóa từ đã được vuốt.

### 5.1 Bật bộ điều hợp để phát hiện từ vuốt

1. Trong `WordListAdapter`, thêm một phương thức để lấy từ ở một vị trí nhất định.

2. Trong `MainActivity`, trong `onCreate()`, tạo `ItemTouchHelper`. Đính kèm `ItemTouchHelper` vào `RecyclerView`

Những điều cần lưu ý trong mã: `onSwiped()` lấy vị trí của `ViewHolder` đã được vuốt:

Với vị trí, bạn có thể lấy từ được hiển thị bởi `ViewHolder` bằng cách gọi phương thức `getWordAtPosition()` mà bạn đã xác định trong bộ điều hợp:

Xóa từ bằng cách gọi `deleteWord()` trên `WordViewModel`

Bây giờ chạy ứng dụng của bạn và xóa một số từ

1. Chạy ứng dụng của bạn. Bạn sẽ có thể xóa các từ bằng cách vuốt chúng sang trái hoặc phải.

Mã giải pháp dự án Android Studio:

Thử thách mã hóa RoomWordsWithDelete

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

Thử thách: Cập nhật ứng dụng của bạn để cho phép người dùng chỉnh sửa một từ bằng cách nhấn vào từ đó rồi lưu các thay đổi của họ.

Gợi ý Thực hiện thay đổi trong `NewWordActivity`

Bạn có thể thêm chức năng vào `NewWordActivity`, để nó có thể được sử dụng để thêm một từ mới hoặc chỉnh sửa một từ hiện có.

Sử dụng khóa được tạo tự động trong Word Lớp thực thể Word sử dụng trường từ làm khóa cơ sở dữ liệu. Tuy nhiên, khi bạn cập nhật một hàng trong cơ sở dữ liệu, mục đang được cập nhật không thể là khóa chính, vì khóa chính là duy nhất cho mỗi hàng và không bao giờ thay đổi. Vì vậy, bạn cần thêm một id được tạo tự động để sử dụng làm khóa chính

Thêm một hàm khởi tạo cho Word nhận id Thêm một hàm khởi tạo vào lớp thực thể Word lấy id và word làm tham số. Đảm bảo hàm khởi tạo bổ sung này được chú thích bằng cách sử dụng @Ignore , vì Room chỉ mong đợi một hàm khởi tạo theo mặc định trong một lớp thực thể.

Để cập nhật một Word hiện có, hãy tạo Word bằng cách sử dụng hàm khởi tạo này. Room sẽ sử dụng khóa chính (trong trường hợp này là id) để tìm mục nhập hiện có trong cơ sở dữ liệu để có thể cập nhật. Trong WordDao, thêm phương thức update() như sau:

Thách thức mã giải pháp dự án

Android Studio: RoomWordsWithUpdate

Summary

Viết mã cơ sở dữ liệu

- Room đảm nhận việc mở và đóng các kết nối cơ sở dữ liệu mỗi khi thực hiện các hoạt động cơ sở dữ liệu.
- Chú thích các phương thức trong đối tượng truy cập dữ liệu (DAO) như @insert, @delete, @update, @query.
- Đối với các lần chèn, cập nhật và xóa đơn giản, chỉ cần thêm chú thích có liên quan vào phương thức trong DAO là đủ.

Chẳng hạn:

- Đối với các truy vấn hoặc tương tác cơ sở dữ liệu phức tạp hơn, chẳng hạn như xóa tất cả các từ, hãy sử dụng chú thích @query và cung cấp SQL cho hoạt động.

Chẳng hạn:

### ItemTouchHelper

- Để cho phép người dùng vuốt hoặc di chuyển các mục trong RecyclerView , bạn có thể sử dụng lớp ItemTouchHelper.
- Triển khai onMove () và onSwipe () .
- Để xác định mục nào người dùng đã di chuyển hoặc vuốt, bạn có thể thêm một phương thức vào bộ điều hợp cho RecylerView . Phương thức này nhận một vị trí và trả về mục có liên quan. Gọi phương thức bên trong onMove() hoặc onSwipe() .

### Khái niệm liên quan T

ài liệu về khái niệm liên quan có trong 10.1: Room, LiveData và ViewModel .

Tìm hiểu thêm về Thực thể, đối tượng truy cập dữ liệu (DAO) và ViewModel:

- Xác định dữ liệu sử dụng Room entities
- Truy cập dữ liệu bằng Room DAO
- Hướng dẫn ViewModel
- Tham chiếu Dao
- Tham chiếu ViewModel
- Để xem tất cả các chú thích có thể có cho một thực thể, hãy truy cập android.arch.persistence.room trong tài liệu tham khảo Android và mở rộng tệp Mục menu Chú thích trong điều hướng bên trái.

### Bài tập về nhà

#### Xây dựng và chạy ứng dụng

Bạn đã tìm hiểu một số cách để lưu trữ dữ liệu. Việc chọn tùy chọn lưu trữ phù hợp phụ thuộc vào dung lượng dữ liệu của bạn và thời gian dữ liệu cần tồn tại.

Tạo một ứng dụng minh họa cách dữ liệu được lưu trữ ở ít nhất hai vị trí khác nhau tồn tại sau các thay đổi cấu hình và phá hủy ứng dụng. Bạn có thể thực hiện việc này bằng cách lưu trữ các phần dữ liệu nhỏ, chẳng hạn như chuỗi, trong các kho dữ liệu khác nhau.

- Ứng dụng sẽ chứng minh điều gì xảy ra với dữ liệu không được lưu.
- Ứng dụng có thể minh họa điều gì xảy ra với dữ liệu được bảo tồn bằng savedInstanceState, dữ liệu sử dụng LiveData với ViewModel và dữ liệu được lưu trữ trong tệp hoặc cơ sở dữ liệu.

Trả lời các câu hỏi sau:

Câu hỏi 1: Thành phần kiến trúc Android cung cấp một số chú thích thuận tiện cho DAO. Có những điều nào sau đây? Chọn bao nhiêu tùy theo áp dụng.

- @Dao
- @Insert
- @Delete
- @Update
- @Query
- @Select

Câu hỏi 2 Lợi ích của việc sử dụng Architecture Components là gì?

- Thành phần kiến trúc giúp bạn cấu trúc ứng dụng của mình theo cách mạnh mẽ và có thể kiểm tra được.
- Thành phần kiến trúc giúp bạn tạo ra giao diện người dùng tốt hơn.
- Thành phần kiến trúc cung cấp một cách tiếp cận đơn giản, linh hoạt và thiết thực để cấu trúc ứng dụng của bạn.
- Nếu bạn sử dụng các thư viện và kiến trúc được cung cấp, ứng dụng của bạn sẽ dễ bảo trì hơn với ít mã soạn sẵn hơn. Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm Kiểm tra xem ứng dụng có các tính năng sau không:

- Lưu dữ liệu theo ít nhất hai cách khác nhau.
- Thể hiện cách dữ liệu được bảo quản khác nhau với các tùy chọn lưu trữ khác nhau.