

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	4
Bài 1) Tạo ứng dụng đầu tiên	4
1.1) Android Studio và Hello World	4
1.2) Giao diện người dùng tương tác đầu tiên	9
1.3) Trình chỉnh sửa bố cục	19
1.5 Tạo một biến thể bố cục cho máy tính bảng	27
2.1 Thay đổi root view group thành LinearLayout	33
2.2 Thay đổi thuộc tính của các phần tử để phù hợp với LinearLayout	34
2.3 Thay đổi vị trí của các phần tử trong LinearLayout	35
2.4 Thêm thuộc tính weight cho phần tử TextView.....	37
Task 3: Thay đổi bố cục sang RelativeLayout.....	40
3.1 Thay đổi LinearLayout thành RelativeLayout	40
3.2 Sắp xếp lại các View trong RelativeLayout.....	40
Mẹo:	43
Bài tập về nhà.....	47
Trả lời các câu hỏi	47
1.4) Văn bản và các chế độ cuộn	48
1.5) Tài nguyên có sẵn.....	79
Bài 2) Activities	79
2.1) Activity và Intent	79
2.2) Vòng đời của Activity và trạng thái	107
2.3) Intent ngầm định.....	121
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	143
3.1) Trình gỡ lỗi	143
3.2) Kiểm thử đơn vị	153

3.3) Thư viện hỗ trợ	153
3.4) Tổng quan về ứng dụng	161
CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG	183
Bài 1) Tương tác người dùng	183
1.1) Hình ảnh có thể chọn	183
Giới thiệu	183
Kiến thức cần có trước	183
Bạn nên có khả năng:	183
CHƯƠNG 3. LÀM VIỆC TRONG NỀN	377
Bài 1) Bài 1.Các tác vụ nền	377
1.1) 1.1 AsyncTask	377
1.1 AsyncTask and AsyncTaskLoader	394
1.2 Broadcast receivers	421
Bài 2) Bài 2. Kích hoạt lập lịch và thông báo nhiệm vụ	430
2.1 Thông báo	430
2.2 Trình quản lý cảnh báo	441
2.3 :JobScheduler.....	452
CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG	458
Bài 1) Tùy chọn và cài đặt.....	458
1.1) Shared preferences.....	458
1.2) Cài đặt ứng dụng.....	458
Bài 2) Lưu trữ dữ liệu với Room	458
2.1) Room, LiveData và ViewModel.....	458
2.2) Room, LiveData và ViewModel.....	458
 3.1) Trinhf gowx loi	

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

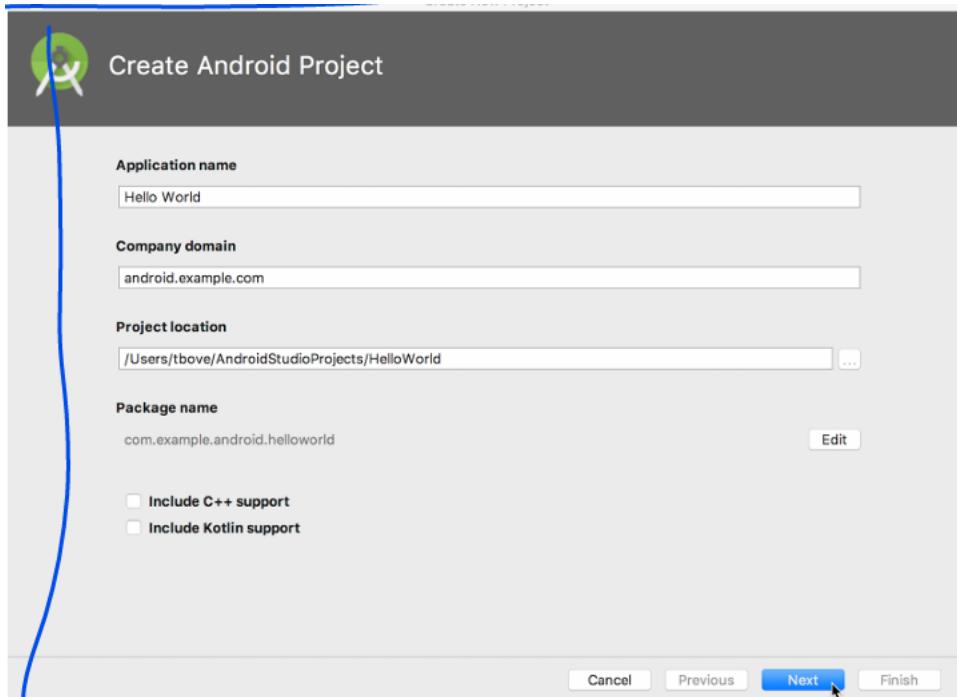
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

Sau khi cài đặt thành công **Android Studio**, bạn sẽ tạo một dự án mới từ mẫu để phát triển ứng dụng **Hello World**. Đây là một ứng dụng đơn giản hiển thị chuỗi "**Hello World**" trên màn hình của thiết bị Android, có thể là thiết bị ảo (**Android Virtual Device - AVD**) hoặc thiết bị vật lý.

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



Task 1: Cài đặt Android Studio

Android Studio cung cấp một **môi trường phát triển tích hợp (IDE - Integrated Development Environment)** hoàn chỉnh, bao gồm một **trình soạn thảo mã nâng cao** và một tập hợp các **mẫu ứng dụng**. Ngoài ra, nó còn đi kèm với các công cụ hỗ trợ phát triển, **gỡ lỗi (debugging)**, **kiểm thử (testing)** và **tối ưu hóa hiệu suất**, giúp quá trình phát triển ứng dụng nhanh chóng và dễ dàng hơn.

Bạn có thể kiểm thử ứng dụng của mình trên nhiều **trình giả lập (emulator)** được cấu hình sẵn hoặc trên thiết bị di động của chính mình, xây dựng ứng dụng hoàn chỉnh và xuất bản trên **Google Play Store**.

Lưu ý: **Android Studio** liên tục được cập nhật và cải tiến. Để có thông tin mới nhất về yêu cầu hệ thống và hướng dẫn cài đặt, hãy truy cập trang web chính thức của [Android Studio](#).

Yêu cầu hệ thống

Android Studio có thể chạy trên các hệ điều hành sau: **Windows, Linux, macOS**. Phiên bản mới nhất của **OpenJDK (Java Development Kit)** được tích hợp sẵn trong **Android Studio**.

- **Hướng dẫn cài đặt**

1. **Truy cập trang web chính thức của Android Studio** và làm theo hướng dẫn để tải về và cài đặt.
2. **Chấp nhận các cấu hình mặc định** trong tất cả các Task và đảm bảo rằng tất cả các thành phần cần thiết được chọn để cài đặt.
3. **Hoàn tất cài đặt:** Trình hướng dẫn thiết lập (**Setup Wizard**) sẽ tải xuống và cài đặt thêm một số thành phần bổ sung, bao gồm **Android SDK**.
 - Quá trình này có thể mất một khoảng thời gian tùy thuộc vào tốc độ mạng của bạn.
 - Một số Task có thể trông giống nhau, nhưng hãy kiên nhẫn và hoàn thành toàn bộ quá trình.
4. **Sau khi tải xuống hoàn tất, Android Studio** sẽ khởi động và bạn đã sẵn sàng tạo dự án đầu tiên của mình.

Xử lý sự cố : Nếu gặp sự cố khi cài đặt, bạn có thể kiểm tra **Ghi chú phát hành (Release Notes)** của Android Studio để xem các vấn đề đã biết; nhờ trợ giúp từ **giảng viên hoặc cộng đồng lập trình viên**.

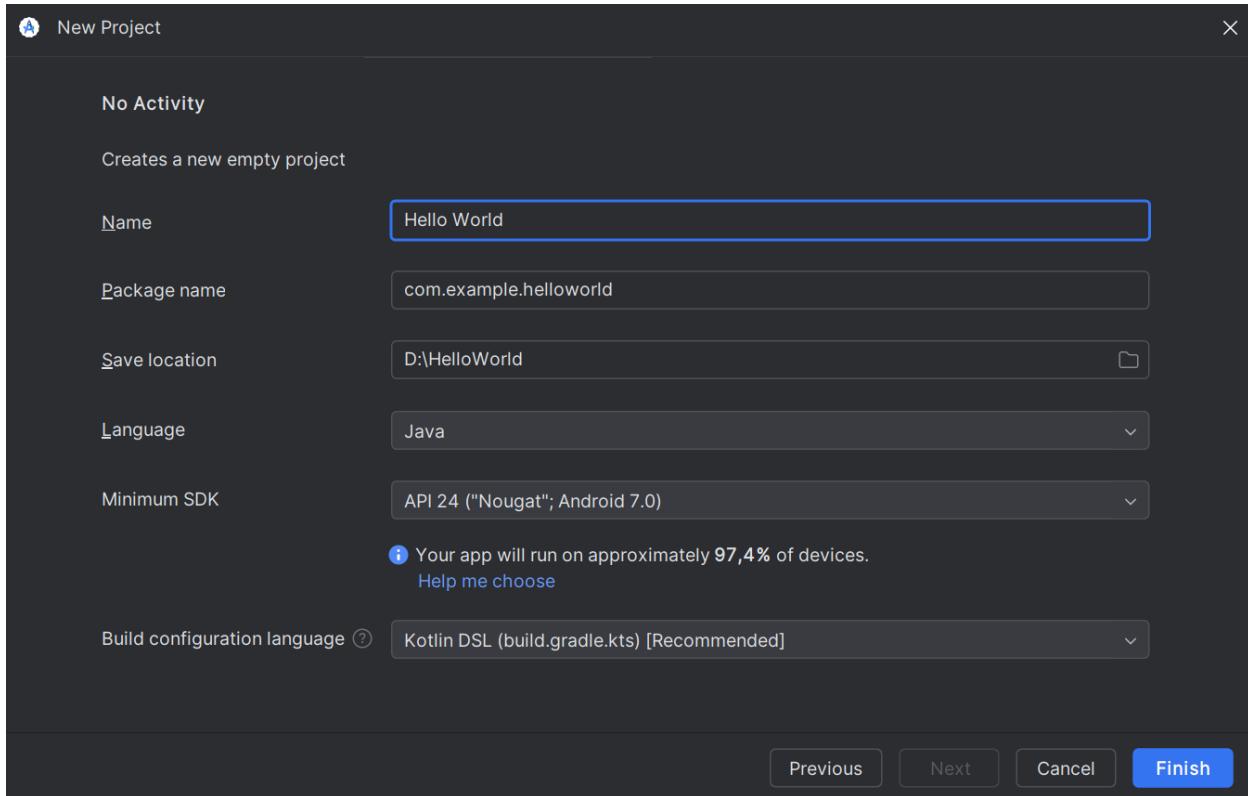
Task 2: Tạo ứng dụng Hello World

Trong Task này, bạn sẽ tạo một ứng dụng hiển thị dòng chữ "Hello World" để kiểm tra xem Android Studio đã được cài đặt đúng cách hay chưa, đồng thời làm quen với các Task cơ bản khi phát triển ứng dụng Android.

2.1 Tạo dự án ứng dụng

1. **Mở Android Studio (nếu chưa mở).**
2. Trong cửa sổ chào mừng **Welcome to Android Studio**, chọn "**Start a new Android Studio project**".

- Trong cửa sổ **Create Android Project**, nhập **Hello World** vào ô **Application name**.



- Kiểm tra **Project location** của dự án **Hello World** và các dự án khác trong **Android Studio** hoặc muốn thay đổi, chọn thư mục lưu trữ theo ý bạn.
- Giữ mặc định **android.example.com** cho **Company Domain** hoặc nhập một tên miền riêng.

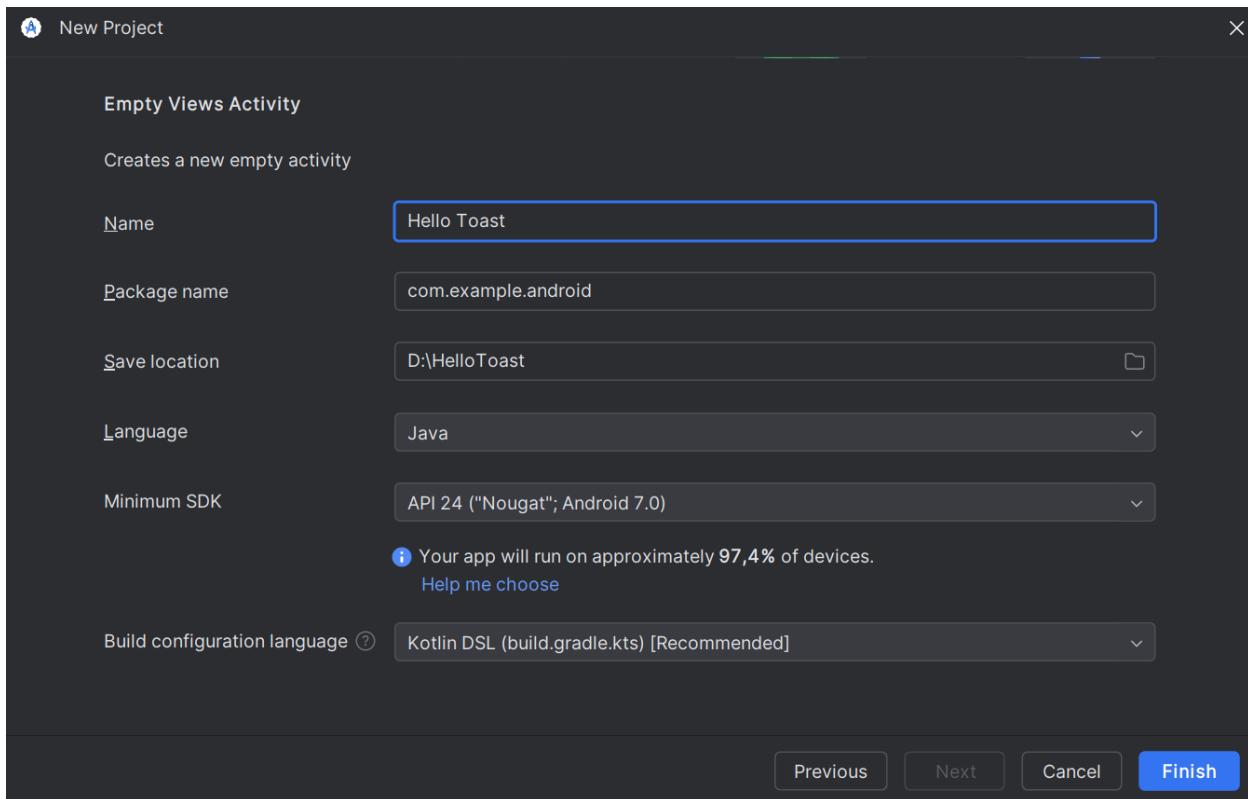
Nếu không có kế hoạch xuất bản ứng dụng, có thể giữ nguyên mặc định. Thay đổi **package name** sau này sẽ tăng thêm công sức.

- Bỏ chọn tùy chọn **Include C++ support** và **Include Kotlin support** => **Next**.
- Ở màn hình **Target Android Devices**, đảm bảo **Phone and Tablet** được chọn. Đặt **API 15: Android 4.0.3 (Ice Cream Sandwich)** làm **Minimum SDK** nếu chưa được chọn. Cấu hình này giúp ứng dụng tương thích với **97% thiết bị Android** trên **Google Play Store**.
- Bỏ chọn **Include Instant App support** và các tùy chọn khác. Nhấn **Next**. Nếu cần cài đặt thêm thành phần cho **SDK**, **Android Studio** sẽ tự động tải và cài đặt.
- Ở màn hình **Add an Activity**, chọn **Empty Activity** => **Next**. **An Activity** là thành phần quan trọng trong mọi ứng dụng Android, thường đi kèm với **layout** để hiển thị giao diện người dùng.
- Ở màn hình **Configure Activity**, giữ tên mặc định **MainActivity**. Bạn có thể đổi tên, nhưng bài học này sử dụng **MainActivity**.

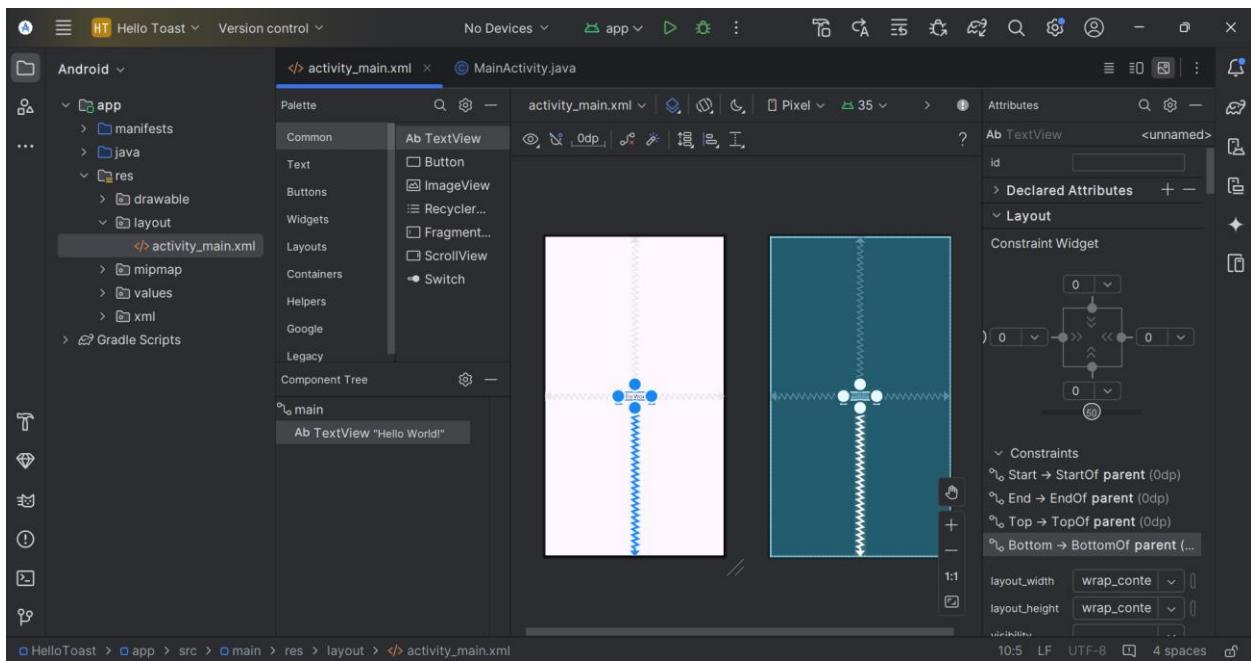
11. Đảm bảo tùy chọn **Generate Layout file** được chọn. Tên mặc định: **activity_main.xml**. Bạn có thể thay đổi, nhưng bài học này sử dụng **activity_main**.
12. Đảm bảo tùy chọn **Backwards Compatibility (AppCompat)** được chọn. Điều này giúp ứng dụng chạy tốt trên các phiên bản Android cũ hơn.
13. Nhấn **Finish** để **Android Studio** khởi tạo dự án.

1.2) Giao diện người dùng tương tác đầu tiên

1.1 Tạo một dự án Android Studio



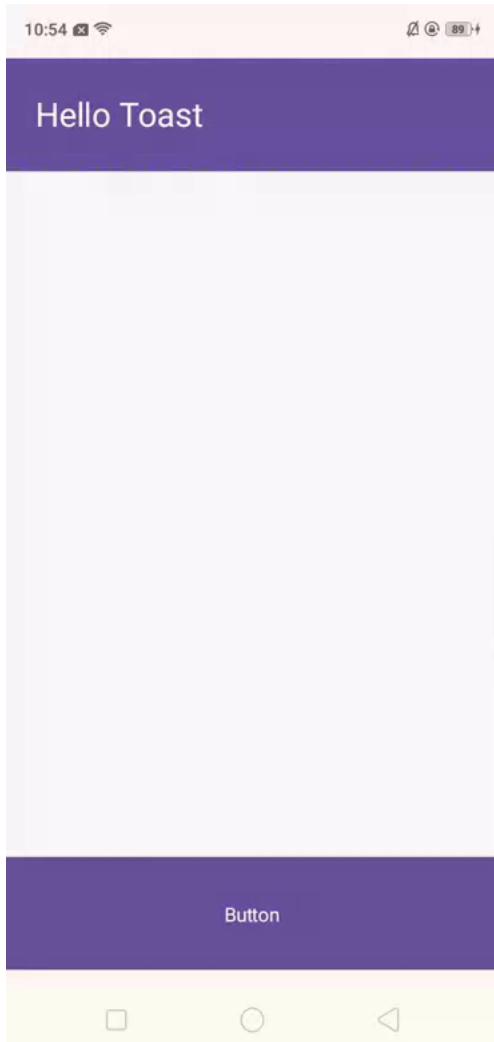
1.2 Khám phá trình chỉnh sửa giao diện



2: Thêm các phần tử view trong Layout editor

2.2 Thêm một button vào layout

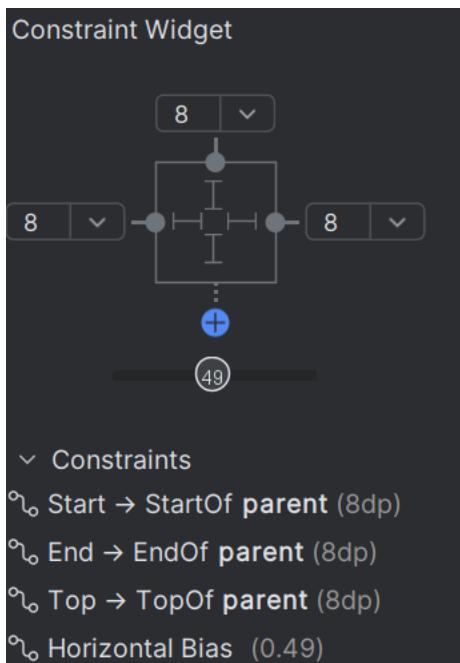
2.3 Thêm button thứ 2 vào layout



Task 3: Thay đổi thuộc tính của phần tử giao diện

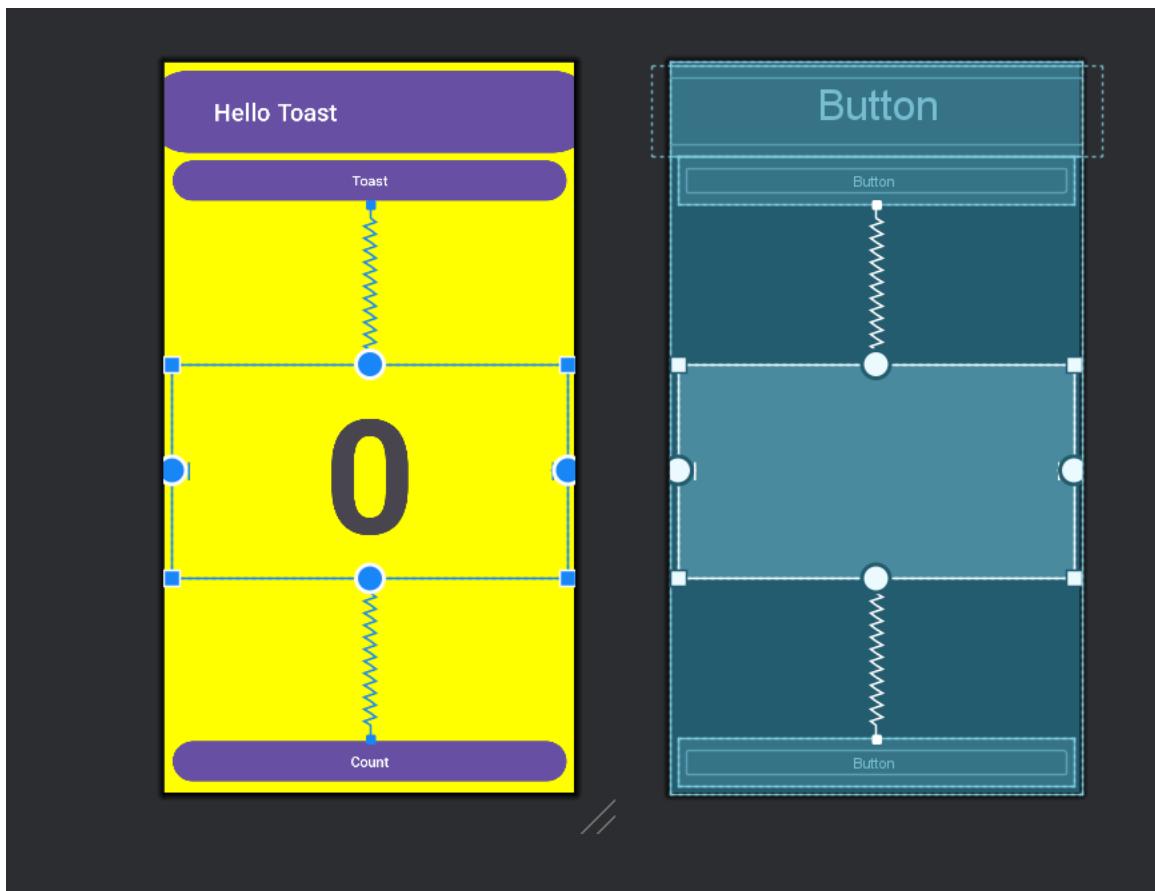
Pane Thuộc tính cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng. Bạn có thể tìm thấy các thuộc tính (được gọi là thuộc tính) chung cho tất cả các view trong tài liệu của lớp View. Trong nhiệm vụ này, bạn sẽ nhập giá trị mới và thay đổi giá trị cho các thuộc tính quan trọng của Button, các thuộc tính này có thể áp dụng cho hầu hết các loại View.

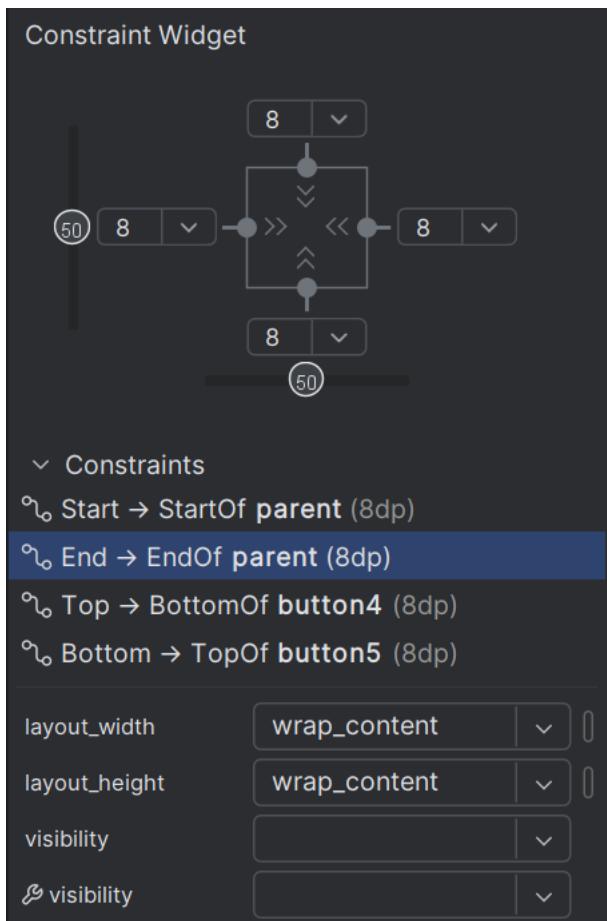
3.1 Thay đổi kích thước button



3.2 Thay đổi thuộc tính của button

Task 4: Thêm TextEdit và set lấy thuộc tính của nó





Task 5: Chính sửa layout trong XML

Bố cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử giao diện người dùng trong Component Tree. Di chuột qua các dấu chấm than này để xem thông báo cảnh báo như hình bên dưới. Cảnh báo giống nhau xuất hiện cho cả ba phần tử: chuỗi được mã hóa cứng nên sử dụng tài nguyên.

Task 6: Thêm onClick (phương thức sự kiện) cho button

6.1 Thêm sự kiện onclick vào 2 button

```
<Button  
    android:id="@+id/button5"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginEnd="8dp"  
    android:layout_marginBottom="8dp"  
    android:text="Count"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.51"  
    app:layout_constraintStart_toStartOf="parent"  
    android:onClick="countUp"/>
```

```
<Button  
    android:id="@+id/button4"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginEnd="8dp"  
    android:text="Toast"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/button3"  
    android:onClick="showToast"/>
```

2. Tự động tạo Phương thức onClick trong MainActivity.java: Khi nhập `android:onClick="showToast"` trong XML, sẽ xuất hiện biểu tượng bóng đèn đỏ bên cạnh. Nhấp vào biểu tượng bóng đèn đỏ hoặc click vào "showToast", sau đó: Chọn **Create click handler**, Chọn **MainActivity**, Nhấn **OK**

=> Làm tương tự cho **countUp**:

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

6.2 Chỉnh sửa sự kiện cho nút Toast

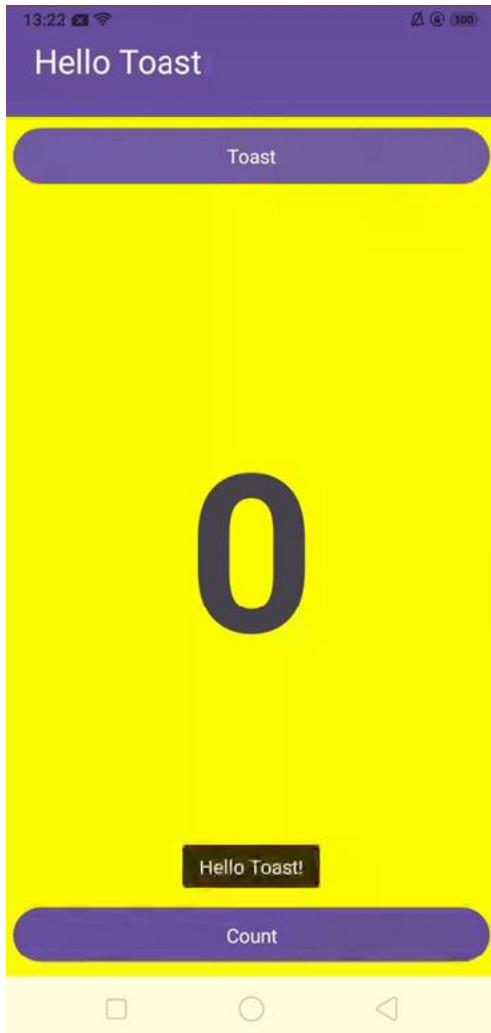
Bây giờ bạn sẽ chỉnh sửa phương thức showToast()—trình xử lý khi nhấn nút Toast trong MainActivity—để hiển thị một thông báo. Toast cung cấp một cách để hiển thị thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ chiếm không gian vừa đủ cho nội dung thông báo. Hoạt động hiện tại vẫn hiển thị và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

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

Các bước:

1. Xác định vị trí phương thức showToast() mới được tạo.
2. Để tạo một instance của Toast, gọi phương thức makeText() của lớp Toast.
3. Cung cấp ngữ cảnh của Activity trong ứng dụng. Vì Toast hiển thị trên giao diện của Activity, hệ thống cần biết thông tin về Activity hiện tại. Khi đang ở trong Activity đó, có thể sử dụng this làm tham chiếu ngắn gọn.
4. Cung cấp thông điệp cần hiển thị, chẳng hạn như một tài nguyên chuỗi (string resource) đã tạo trước đó (toast_message). Tài nguyên chuỗi toast_message được xác định bằng R.string.toast_message.
5. Cung cấp thời gian hiển thị. Ví dụ, Toast.LENGTH_SHORT sẽ hiển thị Toast trong một khoảng thời gian ngắn.
Thời gian hiển thị của Toast có thể là Toast.LENGTH_LONG hoặc Toast.LENGTH_SHORT, với thời gian thực tế khoảng 3.5 giây cho LENGTH_LONG và 2 giây cho LENGTH_SHORT.
6. Hiển thị Toast bằng cách gọi phương thức show(). Dưới đây là toàn bộ phương thức showToast():

- Chạy app và xuất hiện thông điệp “Hello Toast!”



6.3 Chỉnh sửa cho nút sự kiện Count

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

Đoạn code phải:

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

Các bước:

1. Xác định vị trí phương thức **countUp()** mới được tạo.
2. Để theo dõi số lần nhấn, cần một biến thành viên (private member variable). Mỗi lần nhấn nút Count, giá trị của biến này sẽ tăng lên. Nhập đoạn mã sau, dòng này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn đỏ:

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

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'mCount'** từ menu bật lên. Thao tác này sẽ tạo một biến thành viên `private` ở đầu `MainActivity`, và Android Studio sẽ tự động giả định kiểu dữ liệu của biến này là `int`.
4. Thay đổi câu lệnh khai báo biến thành viên `private` để khởi tạo giá trị của biến thành **0**.

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

5. Bên cạnh biến `mCount`, cần thêm một biến thành viên `private` khác để tham chiếu đến `TextView` có ID `show_count`. Đặt tên biến này là `mShowCount`.

```
> </> public class MainActivity extends AppCompatActivity {  
    2 usages  
    private int mCount = 0;  
    3 usages  
    private TextView mShowCount;
```

6. Khi đã có biến `mShowCount`, có thể lấy tham chiếu đến `TextView` bằng cách sử dụng ID đã thiết lập trong tệp `layout`. Để chỉ lấy tham chiếu này một lần, hãy đặt nó trong phương thức `onCreate()`. Trong một bài học khác, bạn sẽ tìm hiểu rằng phương thức `onCreate()` được sử dụng để **inflate** layout, nghĩa là thiết lập nội dung hiển thị của màn hình dựa trên tệp XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện khác trong layout, chẳng hạn như `TextView`. Xác định vị trí phương thức `onCreate()` trong `MainActivity`.

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

7. Thêm câu lệnh findViewById vào cuối phương thức onCreate(). Một View, giống như một chuỗi (String), là một tài nguyên có thể có một **ID**. Lệnh findViewById nhận ID của View làm tham số và trả về đối tượng View. Vì phương thức này trả về một View, cần ép kiểu kết quả thành kiểu TextView:
8. Bây giờ, khi đã gán TextView cho mShowCount, có thể sử dụng biến này để **cập nhật nội dung** trong TextView với giá trị của biến mCount. Thêm đoạn mã sau vào phương thức countUp():

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

9.Chạy app => nhấn count



Thử thách code

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



Thử thách: Thay đổi bố cục sao cho hiển thị đẹp trong cả hai hướng ngang và dọc:

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

1.3) Trình chỉnh sửa bố cục

Giới thiệu

Như bạn đã học trong **Phần 1.2 Phần A: Giao diện người dùng tương tác đầu tiên**, bạn có thể xây dựng giao diện người dùng (UI) bằng **ConstraintLayout** trong trình chỉnh sửa bố cục. ConstraintLayout đặt các phần tử UI trong bố cục bằng cách sử dụng các ràng buộc kết nối với các phần tử khác và với mép của bố cục. **ConstraintLayout** được thiết kế để giúp bạn dễ dàng kéo các phần tử UI vào trình chỉnh sửa bố cục.

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

Ngoài ra, phần thực hành này cũng giới thiệu hai lớp con khác của **ViewGroup**:

- **LinearLayout**: Một nhóm sắp xếp các phần tử **View** con theo chiều ngang hoặc dọc.
- **RelativeLayout**: Một nhóm các phần tử **View** con, trong đó mỗi phần tử **View** được định vị và căn chỉnh dựa trên các phần tử **View** khác trong **ViewGroup**. Vị trí của các phần tử **View** con được mô tả dựa trên mối quan hệ với nhau hoặc với **ViewGroup** cha.

Những gì bạn nên biết trước

Bạn nên có khả năng:

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

Những gì bạn sẽ học

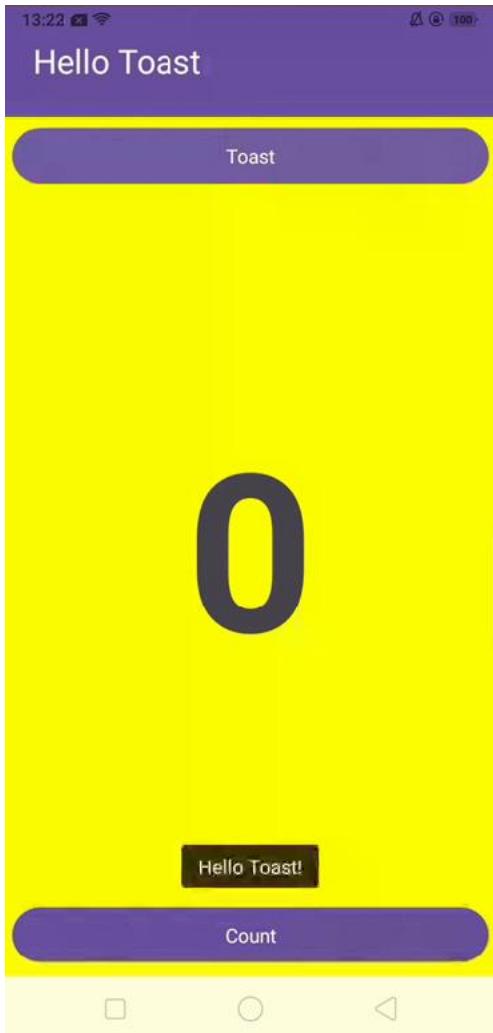
- Cách tạo một biến thể bố cục cho **chế độ ngang (landscape)**.
- Cách tạo một biến thể bố cục cho **máy tính bảng và màn hình lớn hơn**.
- Cách sử dụng **ràng buộc đường cơ sở (baseline constraint)** để căn chỉnh phần tử UI với văn bản.
- Cách sử dụng **nút gói (pack) và căn chỉnh (align)** để sắp xếp 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 bối cục cho **chế độ hiển thị ngang**.
- Tạo một biến thể bối cục cho **máy tính bảng và màn hình lớn hơn**.
- Chỉnh sửa bối cục để thêm **các ràng buộc** cho phần tử UI.
- Sử dụng **ConstraintLayout baseline constraints** để căn chỉnh phần tử với văn bản.
- Sử dụng **nút pack và align** của **ConstraintLayout** để căn chỉnh các phần tử.
- Thay đổi bối cục để sử dụng **LinearLayout**.
- Sắp xếp các phần tử trong **LinearLayout**.
- Thay đổi bối cục để sử dụng **RelativeLayout**.
- Sắp xếp lại các **View** trong bối cục chính để liên kết với nhau.

Tổng quan về ứng dụng

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



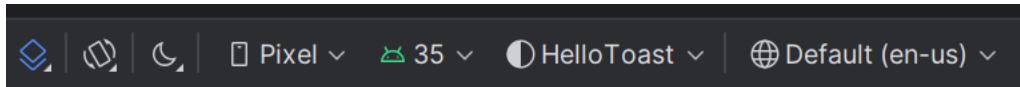
Nhiệm vụ 1: Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu thay đổi bố cục của ứng dụng **Hello Toast** để nó có thể hiển thị đúng trong chế độ **ngang (horizontal/landscape)** hoặc **dọc (vertical/portrait)**.

Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo **biến thể bố cục** cho:

- **Điện thoại** ở chế độ **ngang** (landscape) và **dọc** (portrait).
- **Thiết bị có màn hình lớn** như **máy tính bảng** (tablets).

Bạn sẽ sử dụng một số **nút trong hai thanh công cụ trên cùng** của trình chỉnh sửa bố cục. Thanh công cụ trên cùng giúp bạn **cấu hình cách hiển thị bản xem trước bố cục** trong trình chỉnh sửa.



- **Chọn bề mặt thiết kế:** Design, Blueprint, hoặc Design + Blueprint.
- **Chọn định hướng:** Portrait, Landscape, hoặc tạo biến thể mới.
- **Chọn thiết bị:** Phone/Tablet, Android TV, hoặc Android Wear.
- **Chọn phiên bản API:** Chọn phiên bản Android để xem trước.
- **Chọn chủ đề:** AppTheme hoặc các chủ đề khác.
- **Chọn ngôn ngữ và địa phương:** Chọn ngôn ngữ, địa phương, hoặc xem trước dưới dạng RTL.

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



1. **Trong hình trên:**
2. **Show:** Chọn **Show Constraints** và **Show Margins** để hiển thị hoặc ẩn chúng trong bản xem trước.
3. **Autoconnect:** Bật/tắt **Autoconnect**. Khi bật, bạn có thể kéo bất kỳ phần tử nào (như **Button**) vào bố cục để tự động tạo ràng buộc với **bố cục cha**.
4. **Clear All Constraints:** Xóa tất cả ràng buộc trong bố cục.
5. **Infer Constraints:** Tạo ràng buộc tự động bằng suy luận.
6. **Default Margins:** Đặt lề mặc định cho các phần tử UI.
7. **Pack:** Gom nhóm hoặc mở rộng các phần tử đã chọn.
8. **Align:** Căn chỉnh các phần tử đã chọn.

9. **Guidelines:** Thêm đường hướng dẫn dọc hoặc ngang.

10. **Zoom/pan controls:** Phóng to, thu nhỏ hoặc di chuyển bản xem trước.

1.1 Xem trước bố cục ở chế độ ngang

Để xem trước bố cục của ứng dụng **Hello Toast** ở chế độ ngang, thực hiện các bước sau:

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

- **Lưu ý:** Nếu bạn đã tải mã nguồn hoàn chỉnh của **HelloToast**, hãy xóa các bố cục **landscape** và **extra-large** mà bạn sẽ tạo trong nhiệm vụ này.
- Chuyển từ **Project > Android** sang **Project > Project Files** trong **Project pane**.
- Mở **app > src/main > res**, chọn cả hai thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**.
- Chuyển **Project pane** trở lại **Project > Android**.

2. **Mở tệp activity_main.xml**, nhấp vào tab **Design** nếu nó chưa được chọn.

3. **Nhấp vào nút Orientation in Editor** trên thanh công cụ trên cùng để thay đổi hướng xem trước.

4. Chọn "**Switch to Landscape**" trong menu thả xuống. Bố cục sẽ hiển thị ở chế độ ngang (**horizontal orientation**) như hình bên dưới. Để quay lại chế độ dọc (**vertical orientation**), chọn "**Switch to Portrait**".

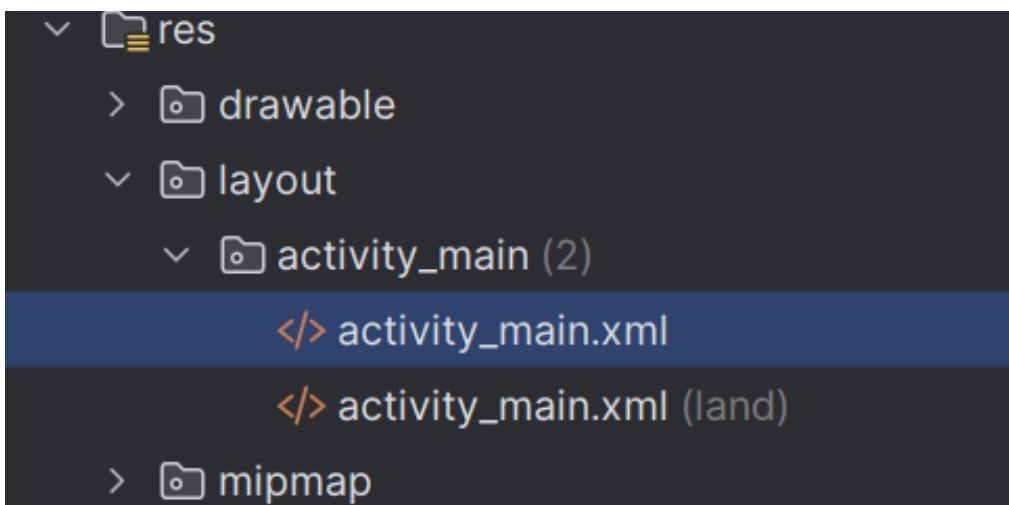


1.2 Tạo biến thể bố cục cho chế độ ngang

Sự khác biệt chính giữa **chế độ dọc (vertical)** và **chế độ ngang (horizontal)** trong bố cục này là số "0" trong **TextView (show_count)** bị đặt quá thấp, gần với nút **Count**. Tùy vào thiết bị hoặc trình giả lập, **TextView** có thể hiển thị quá lớn hoặc không căn giữa do **kích thước văn bản được cố định ở 160sp**.

Để khắc phục điều này mà không ảnh hưởng đến **chế độ dọc**, bạn có thể tạo một **biến thể bố cục (layout variant)** dành riêng cho **chế độ ngang**.

1. Nhấp vào nút "**Orientation in Editor**" trên thanh công cụ trên cùng.
2. Chọn "**Create Landscape Variation**".
 - Một cửa sổ chỉnh sửa mới sẽ mở ra với tab **land/activity_main.xml**, hiển thị bố cục dành riêng cho chế độ ngang. Bạn có thể chỉnh sửa bố cục này mà không ảnh hưởng đến bố cục **chế độ dọc** ban đầu.
3. Mở **Project > Android**, tìm trong thư mục **res > layout**, và bạn sẽ thấy **Android Studio tự động tạo biến thể** với 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 nhiều thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thực hoặc trình giả lập.

1. Đảm bảo tab **land/activity_main.xml** vẫn mở trong trình chỉnh sửa bố cục. Nếu chưa mở, hãy **nhấp đúp** vào tệp **activity_main.xml (land)** trong thư mục **res > layout**.
2. **Nhấp vào nút "Device in Editor"** trên thanh công cụ trên cùng.
3. **Chọn một thiết bị khác** trong menu thả xuống.
 - o Ví dụ: chọn **Nexus 4, Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong bản xem trước.
 - o Những khác biệt này xảy ra do **kích thước văn bản cố định** trong **TextView**.

1.4 Thay đổi bố cục cho chế độ ngang

Bạn có thể sử dụng **pane Attributes** trong tab **Design** để thiết lập hoặc thay đổi thuộc tính. Tuy nhiên, đôi khi **sửa trực tiếp mã XML** trong tab **Text** sẽ nhanh hơn. Tab **Text** hiển thị **mã XML** và có **tab Preview** bên phải để xem trước bố cục, như hình minh họa bên dưới.



Để thay đổi bố cục, hãy làm theo các bước sau:

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

6. Trong ngăn chỉnh sửa, tab land/activity_main.xml hiển thị bố cục cho chế độ ngang.
7. Tab activity_main.xml hiển thị bố cục chưa thay đổi cho chế độ dọc.
8. Bạn có thể chuyển đổi qua lại bằng cách nhấp vào các tab.
9. Chạy ứng dụng trên trình giả lập hoặc thiết bị thật, sau đó chuyển đổi hướng màn hình từ dọc sang ngang để xem cả hai bố cục.



1.5 Tạo một biến thể bố cục cho máy tính bảng

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

Để khắc phục điều này cho máy tính bảng trong khi vẫn giữ nguyên bố cục ngang và dọc của điện thoại, bạn có thể tạo một biến thể của bố cục hoàn toàn khác cho máy tính bảng. Hãy làm theo các bước sau:

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

Một cửa sổ trình chỉnh sửa mới sẽ mở ra với tab **xlarge/activity_main.xml**, hiển thị bố cục dành cho thiết bị có kích thước máy tính bảng. Trình chỉnh sửa cũng sẽ chọn một thiết bị máy tính bảng, chẳng hạn như **Nexus 9** hoặc **Nexus 10**, để hiển thị bản xem trước. Bạn có thể thay đổi bố cục này dành riêng cho máy tính bảng mà không ảnh hưởng đến các bố cục khác.

1.6 Thay đổi biến thể bố cục cho máy tính bảng

Bạn có thể sử dụng ngăn Thuộc tính trong tab Thiết kế để thay đổi các thuộc tính cho bố cục này.

1. Tắt công cụ Tự động kết nối trên thanh công cụ. Đối với bước này, hãy đảm bảo rằng công cụ đã bị vô hiệu hóa.
2. Xóa tất cả các ràng buộc trong bố cục bằng cách nhấp vào nút **Clear All Constraints** trên thanh công cụ.

Với các ràng buộc đã được xóa, bạn có thể di chuyển và thay đổi kích thước các phần tử trong bố cục một cách tự do.

3. Trình chỉnh sửa bố cục cung cấp các tay nắm thay đổi kích thước ở cả bốn góc của một phần tử để thay đổi kích thước của nó. Trong Cây Thành phần, chọn **TextView** có tên **show_count**. Để di chuyển **TextView** ra khỏi vị trí để bạn có thể tự do kéo các phần tử **Button**, hãy kéo một góc của nó để thay đổi kích thước, như minh họa trong hình động bên dưới.

Việc thay đổi kích thước một phần tử sẽ mã hóa cứng các kích thước chiều rộng và chiều cao. Tránh mã hóa cứng các kích thước cho hầu hết các phần tử, vì bạn không thể dự đoán cách các kích thước được mã hóa cứng sẽ hiển thị trên các màn hình có kích thước và mật độ khác nhau. Bạn chỉ đang làm điều này ngay bây giờ để di chuyển phần tử ra khỏi vị trí, và bạn sẽ thay đổi các kích thước trong một bước khác.

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

Declared Attributes		+	-
layout_width	wrap_content	▾	0
layout_height	90dp	▾	0
layout_constraint	parent	▾	0
layout_margin	44dp	0	
cornerRadius	0dp	0	
id	button4	0	
ignore	MissingConstraints	0	
onClick	showToast	▾	0
text	Toast	0	
textSize	60sp	▾	0

▼ Layout

Như hiển thị ở phía bên phải của hình trên (2), bạn có thể nhấp vào điều khiển chiều rộng của trình kiểm tra chế độ xem, xuất hiện dưới dạng hai đoạn ở bên trái và bên phải của hình vuông, cho đến khi nó hiển thị **Wrap Content**. Ngoài ra, bạn có thể chọn **wrap_content** từ menu **layout_width**.

Bạn sử dụng **wrap_content** để nếu văn bản của **Button** được bandle hóa sang một ngôn ngữ khác, **Button** sẽ hiển thị rộng hơn hoặc hẹp hơn để phù hợp với từ trong ngôn ngữ khác.

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

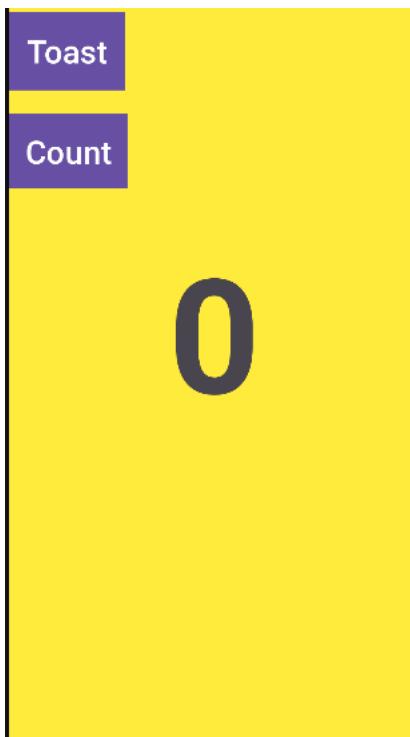
1.7 Sử dụng ràng buộc đường cơ sở (Baseline Constraint)

Bạn có thể căn chỉnh một phần tử giao diện người dùng (UI) chứa văn bản, chẳng hạn như **TextView** hoặc **Button**, với một phần tử giao diện người dùng 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 văn bản của chúng khớp nhau.

- Ràng buộc nút **button_toast** vào cạnh trên và cạnh trái của bố cục, kéo nút **button_count** đến một vị trí gần **button_toast**, Ràng buộc **button_count** vào cạnh trái của **button_toast**, như hiển thị trong hình động.
- 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.

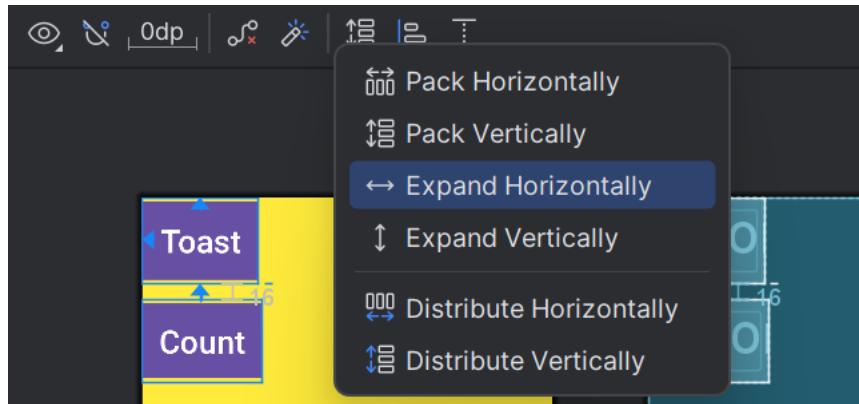
- Nhấp vào nút ràng buộc đường cơ sở. Tay cầm ràng buộc đường cơ sở sẽ xuất hiện, nhấp nháy màu xanh lá. Nhấp và kéo một đường ràng buộc đường cơ sở đến đường cơ sở của phần tử **button_toast** để hoàn tất.



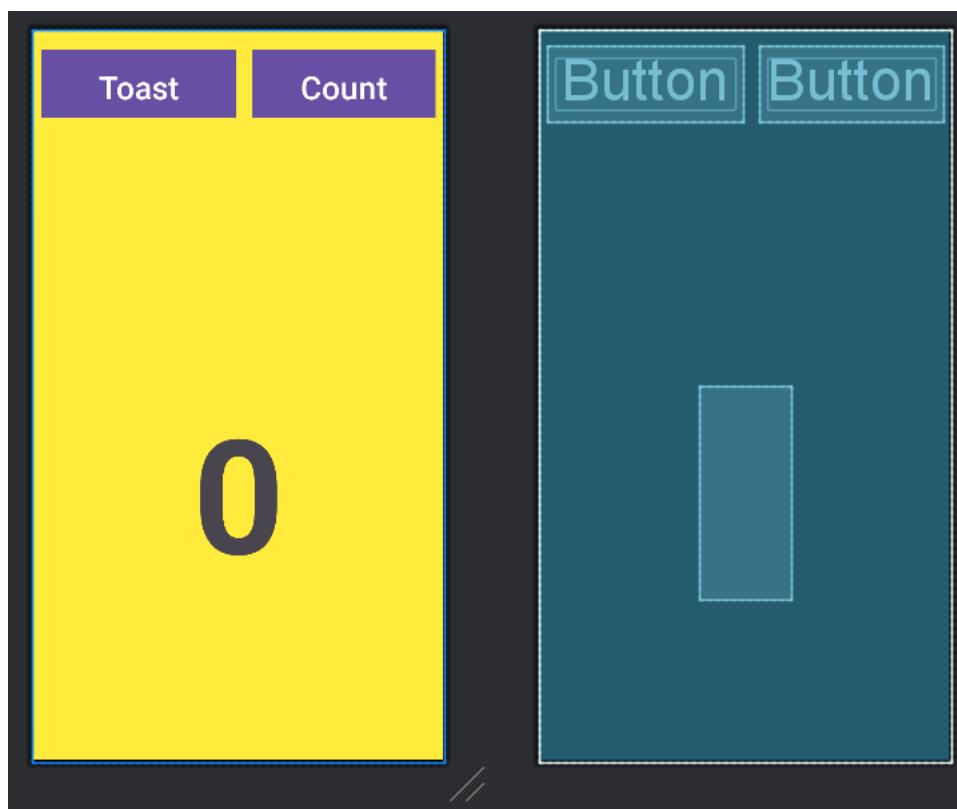
1.8 Mở rộng các nút theo chiều ngang

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.

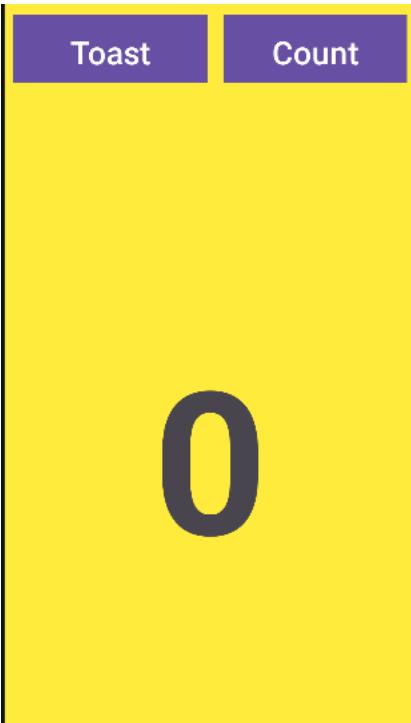
- Chọn nút **button_count** trong **Component Tree**, sau đó nhấn giữ **Shift** và chọn thêm nút **button_toast** để cả hai nút đều được chọn.
- Nhấp vào nút **pack** trên thanh công cụ, sau đó chọn tùy chọn **Expand Horizontally** như minh họa trong hình.



- 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.



- 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)**.

Task 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.

2.1 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"  
    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"
```

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:id="@+id/main"
    android:layout_width="match_parent"
    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**.

2.2 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:

```
<Button
    android:id="@+id/btn_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

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

```
    ...
    app:layout_constraintEnd_toStartOf="@+id/btn_count"
    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.

```
<Button  
    android:id="@+id/btn_count"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"
```

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

```
    app:layout_constraintBottom_toBottomOf="@+id/button4"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toEndOf="@+id/button4"  
    app:layout_constraintTop_toTopOf="parent" />
```

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

```
<TextView  
    android:id="@+id/show_Count"  
    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/button4" />
```

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

2.3 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:

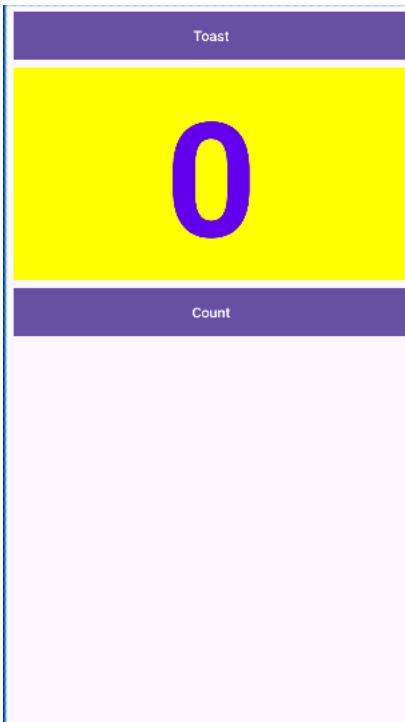
1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp bố cục **activity_main.xml** (nếu chưa mở), sau đó nhấp vào tab **Text**.

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:

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="@color/colorPrimary"
    android:onClick="showToast"
    android:text="@string/button_label_toast"
    android:textColor="@android:color/white" />
<TextView
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="#FFFF00"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAlignment="center"
    android:textColor="@color/colorPrimary"
    android:textSize="160sp"
    android:textStyle="bold" />
<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:onClick="countUp"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white"/>
```

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:



2.4 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.

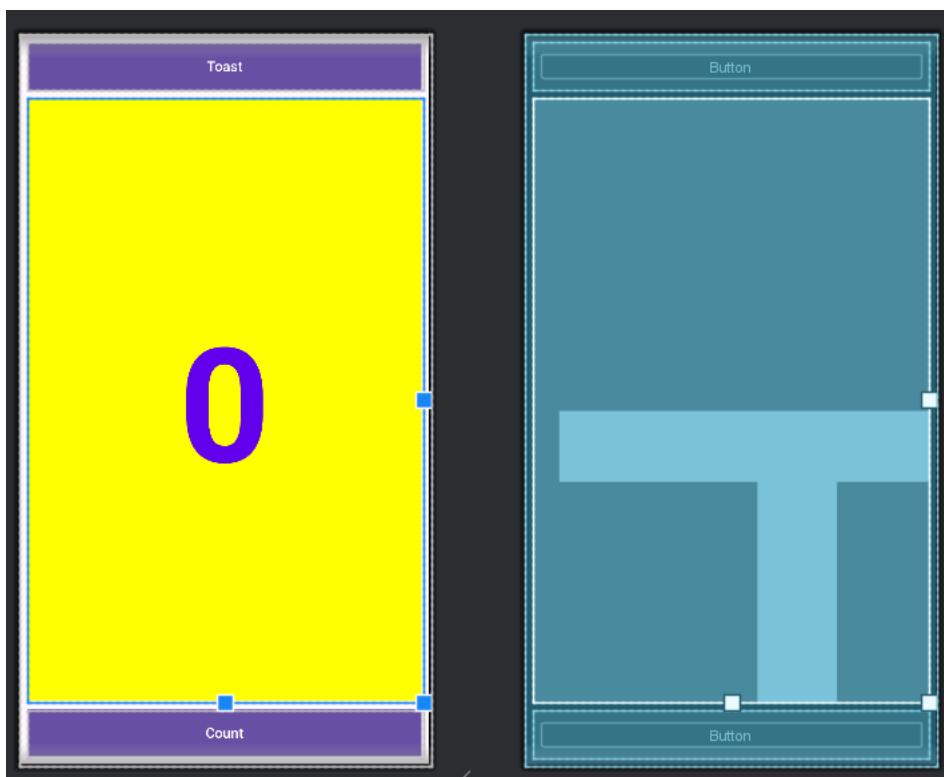
Trên các thiết bị khác nhau, **show_count** (**TextView**) có thể chiếm một phần hoặc phần lớn không gian giữa hai nút **Toast** và **Count**. Để đảm bảo **TextView** mở rộng và lấp đầy không gian trống trên mọi thiết bị, hãy thêm thuộc tính `android:layout_weight`.

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 nhấp 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.

Task 2: Solution code

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"
    android:background=""
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@color/colorPrimary"
        android:onClick="showToast"
        android:text="@string/button_label_toast"
        android:textColor="@android:color/white" />
    <TextView
        android:id="@+id/show_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="#FFFF00"
        android:gravity="center"
        android:text="@string/count_initial_value"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary"
        android:textSize="160sp"
        android:textStyle="bold" />
    <Button
        android:id="@+id/button_count"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:background="@color/colorPrimary"
```

```
        android:onClick="countUp"
        android:text="@string/button_label_count"
        android:textColor="@android:color/white"/>
    
```

</LinearLayout>

Task 3: 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**.

3.1 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:

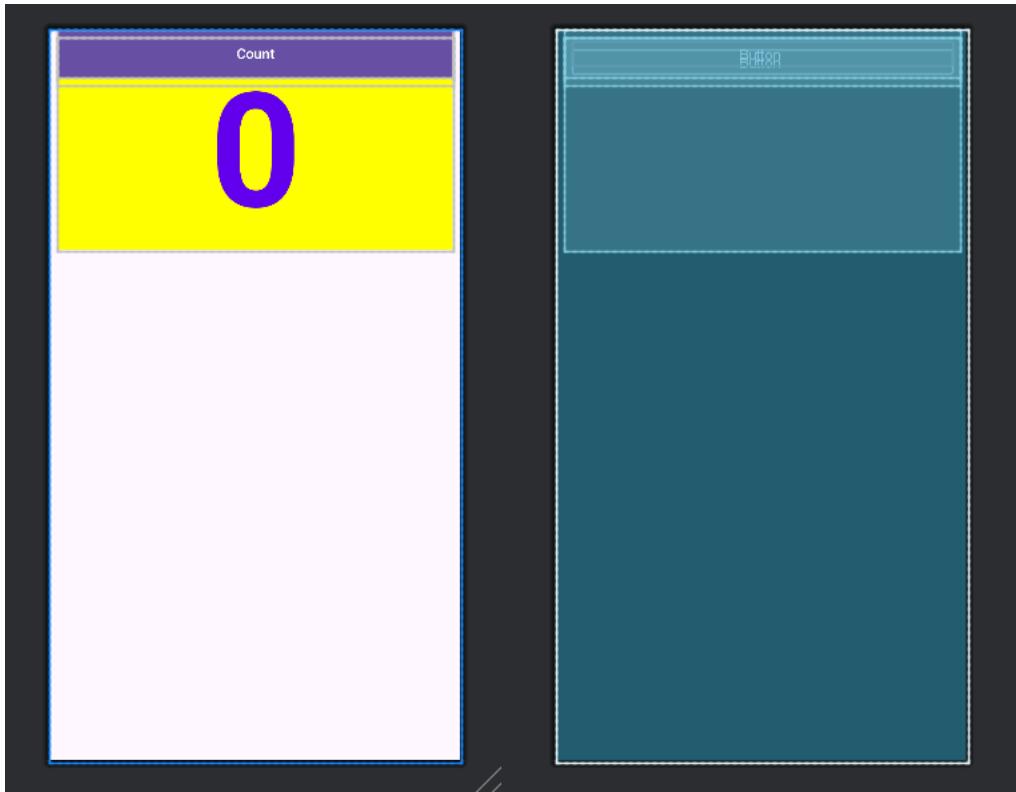
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

3. Cuộn xuống để đảm bảo thẻ kết thúc </LinearLayout> cũng được thay đổi thành </RelativeLayout>; nếu chưa, hãy chỉnh sửa thủ công.

3.2 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:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:onClick="countUp"
    android:layout_below="@+id/show_count"
    android:layout_centerHorizontal="true"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white"/>/
```

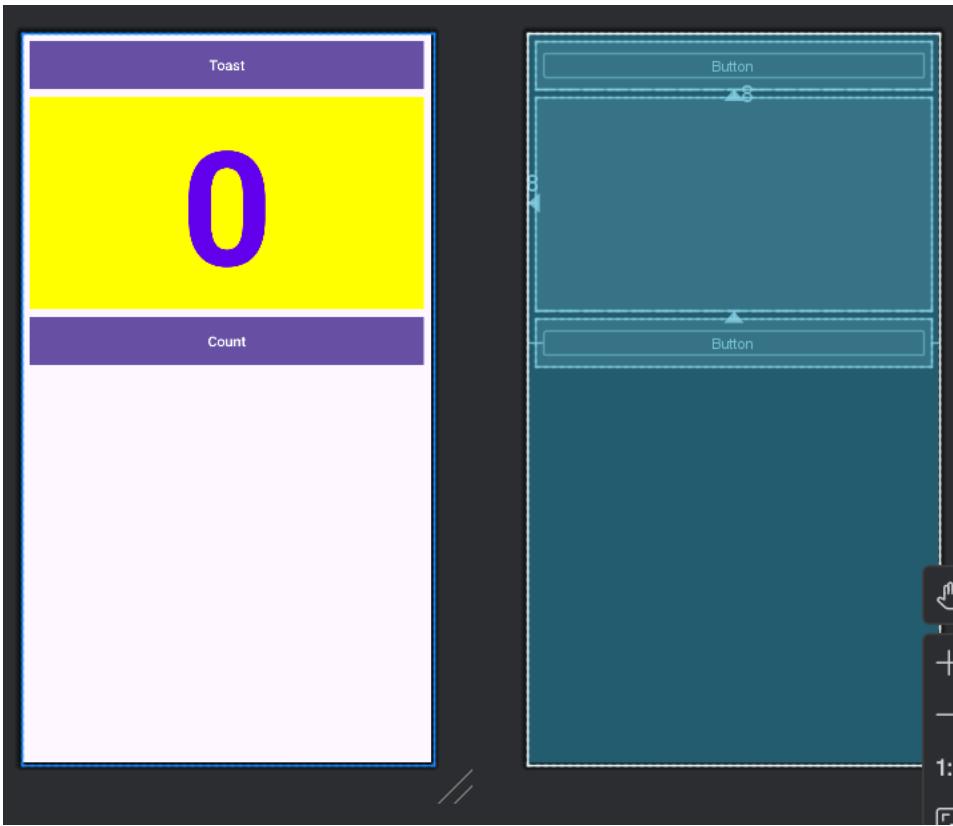
4. Thêm các thuộc tính sau vào **show_count** (TextView):

```
    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**.

Task 3

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"
    android:background=""
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```

        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@color/colorPrimary"
        android:onClick="showToast"
        android:text="@string/button_label_toast"
        android:textColor="@android:color/white" />
<TextView
    android:id="@+id/show_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="#FFFF00"
    android:gravity="center"
    android:text="@string/count_initial_value"
    android:textAlignment="center"
    android:textColor="@color/colorPrimary"
    android:textSize="160sp"
    android:textStyle="bold"
    android:layout_below="@id/button_toast"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"/>
<Button
    android:id="@+id/button_count"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:background="@color/colorPrimary"
    android:onClick="countUp"
    android:layout_below="@id/show_count"
    android:layout_centerHorizontal="true"
    android:text="@string/button_label_count"
    android:textColor="@android:color/white"/>
</RelativeLayout>
```

- **Tóm tắt**

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

- Để xem trước bố 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 bố 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 bố 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 từ **trái sang phải**.

- vertical → Sắp xếp phần tử từ **từ 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** **đọc 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 đọc**, 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()

1.4) Văn bản và các chế độ 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.

TASK 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**.

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

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

2. 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>
```
3. 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.

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

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

Khối 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
<code>android:layout_width</code>	<code>"match_parent"</code>
<code>android:layout_height</code>	<code>"wrap_content"</code>
<code>android:id</code>	<code>"@+id/article_heading"</code>
<code>android:background</code>	<code>"@color/colorPrimary"</code>
<code>android:textColor</code>	<code>"@android:color/white"</code>
<code>android:padding</code>	<code>"10dp"</code>
<code>android:textAppearance</code>	<code>"@android:style/TextAppearance.DeviceDefault.Large"</code>
<code>android:textStyle</code>	<code>"bold"</code>
<code>android:text</code>	<code>"Article Title"</code>

```
<TextView
    android:id="@+id/article_heading"
    android:background="@color/colorPrimary"
    android:textColor="@android:color/white"
    android:padding="10dp"
    android:textAppearance="@android:style/TextAppearance.Large"
    android:textStyle="bold"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    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**.

```
<resources>
    <string name="app_name">ScrollingText</string>
    <string name="article_title">Article Title</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.

```
android:padding="@dimen/padding_regular"
```

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"

```
<TextView
    android:id="@+id/article_subheading"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="@dimen/padding_regular"
    android:layout_below="@+id/article_heading"
```

```
    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.

```
<string name="article_subtitle">Article Subtitle</string>
```

11. Trong phần tử TextView "Hello World", xóa các thuộc tính layout_constraint:

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

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

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

```
    android:lineSpacingExtra="@dimen/line_spacing"
    android:text="@string/article_text" />
```

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 **** và **** 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 <i>... từ ...</i>.
- 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. (Ví dụ dưới đây sử dụng 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">ScrollingText</string>
    <string name="article_title">The Future of Innovation</string>
    <string name="article_subtitle">Exploring the Next Generation of
Technology</string>
    <string name="article_text">
        <b>Technology</b> has transformed the world in unimaginable
ways.\n\n
        From the rise of <i>artificial intelligence</i> to the
expansion of <b>space exploration</b>,\n
        every breakthrough pushes humanity forward. However, innovation
is not just about advancements-\n
        it's about how we <i>adapt</i> to change and embrace the
opportunities it presents.\n\n
        The digital era has introduced concepts like <b><i>smart
cities</i></b>, autonomous vehicles,\n
        and quantum computing. Each of these developments paves the way
for a more connected world,\n
        where efficiency and sustainability go hand in hand.\n\n
        If you're interested in learning more about emerging
technologies, visit:\n
        www.google.com
    </string>
</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.

TASK 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 bố 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:id="@+id/article"  
    android:autoLink="web"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    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 bố 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:id="@+id/article_subheading"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    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:id="@+id/article"  
    android:autoLink="web"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@id/article_subheading"  
    android:lineSpacingExtra="@dimen/line_spacing"
```

```
    android:padding="@dimen/padding_regular"
    android:text="@string/article_text" />
```

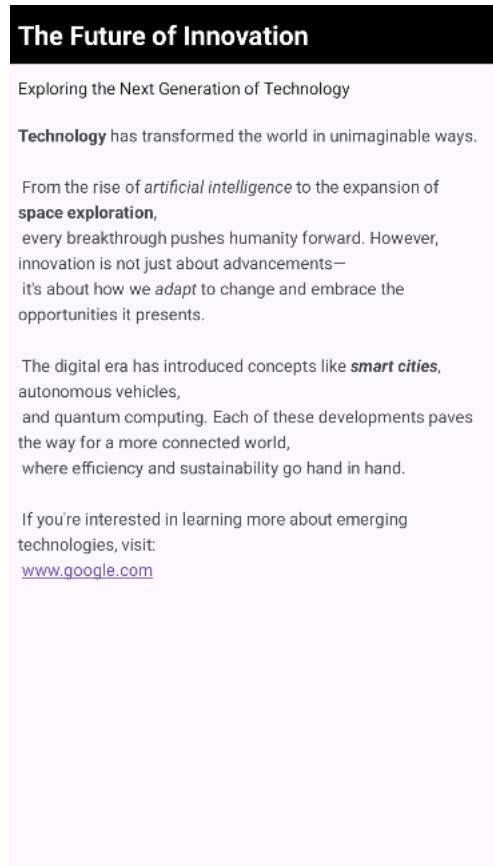
3. Di chuyển đoạn mã kết thúc </ScrollView> xuống sau TextView có ID article để các thuộc tính của article nằm hoàn toàn bên trong **ScrollView**.
4. Xóa thuộc tính sau từ TextView có ID article và thêm nó vào **ScrollView**:

```
    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 bối cảnh để xem trước bối cảnh.

Bối cảnh 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:

1. 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) thành một liên kết web.

2. 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.

3. 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.

The Future of Innovation

Exploring the Next Generation of Technology

Technology has transformed the world in unimaginable ways.

From the rise of *artificial intelligence* to the expansion of **space exploration**, every breakthrough pushes humanity forward. However, innovation is not just about advancements—it's about how we *adapt* to change and embrace the opportunities it presents.

The digital era has introduced concepts like *smart cities*, autonomous vehicles, and quantum computing. Each of these developments paves the way for a more connected world, where efficiency and sustainability go hand in hand.

If you're interested in learning more about emerging technologies, visit:
www.google.com

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 bố 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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/article_heading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimary"
        android:padding="@dimen/padding_regular"
```

```
        android:text="@string/article_title"
        android:textAppearance="@android:style/TextAppearance.Large"
        android:textColor="@android:color/white"
        android:textStyle="bold" />

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

</RelativeLayout>
```

TASK 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:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/article"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:autoLink="web"
            android:lineSpacingExtra="@dimen/line_spacing"
            android:padding="10dp"
            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.
- Thay đổi thuộc tính bố 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:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/article_subheading"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/article_heading"
            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.tùy 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.

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.

TASK 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.

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

1. Truy cập 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.

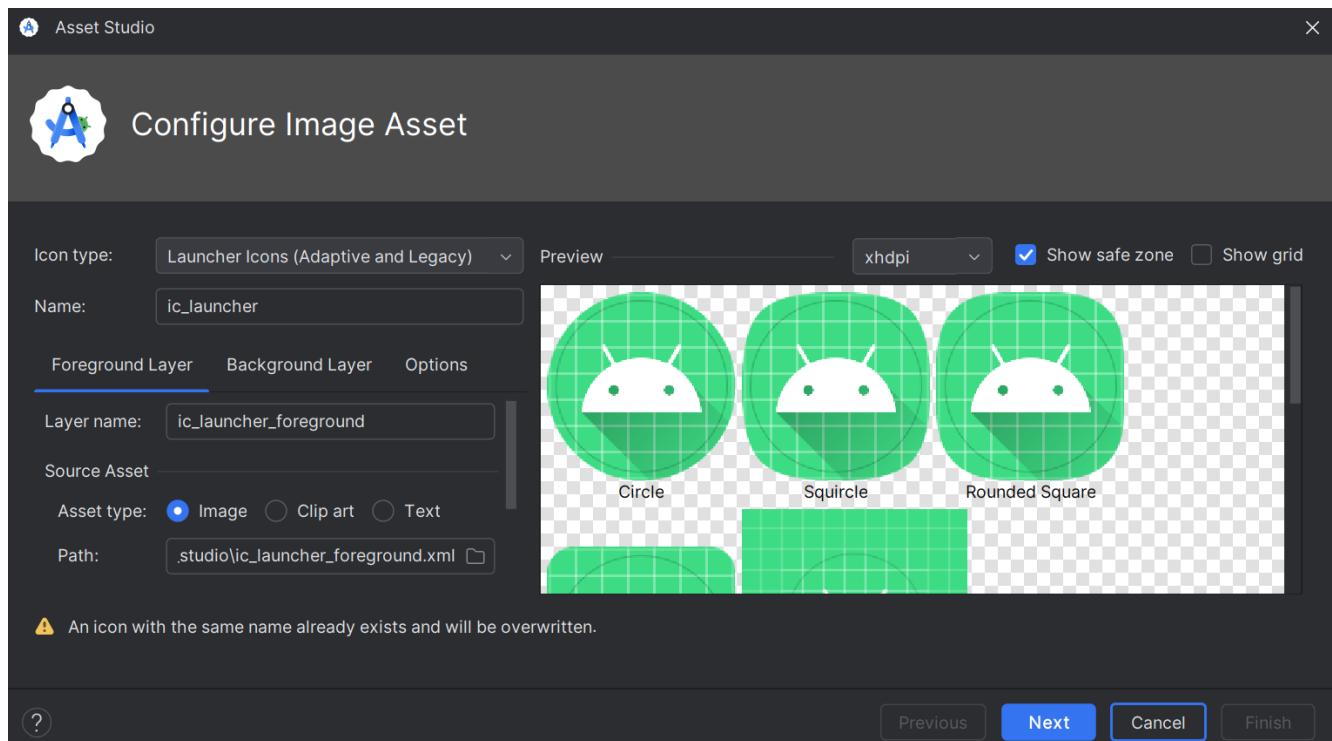
2. Truy cập developer.android.com/docs/ để tìm thông tin về API, tài liệu tham khảo, hướng dẫn, công cụ, và các mẫu mã.

3. Truy cập 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**.

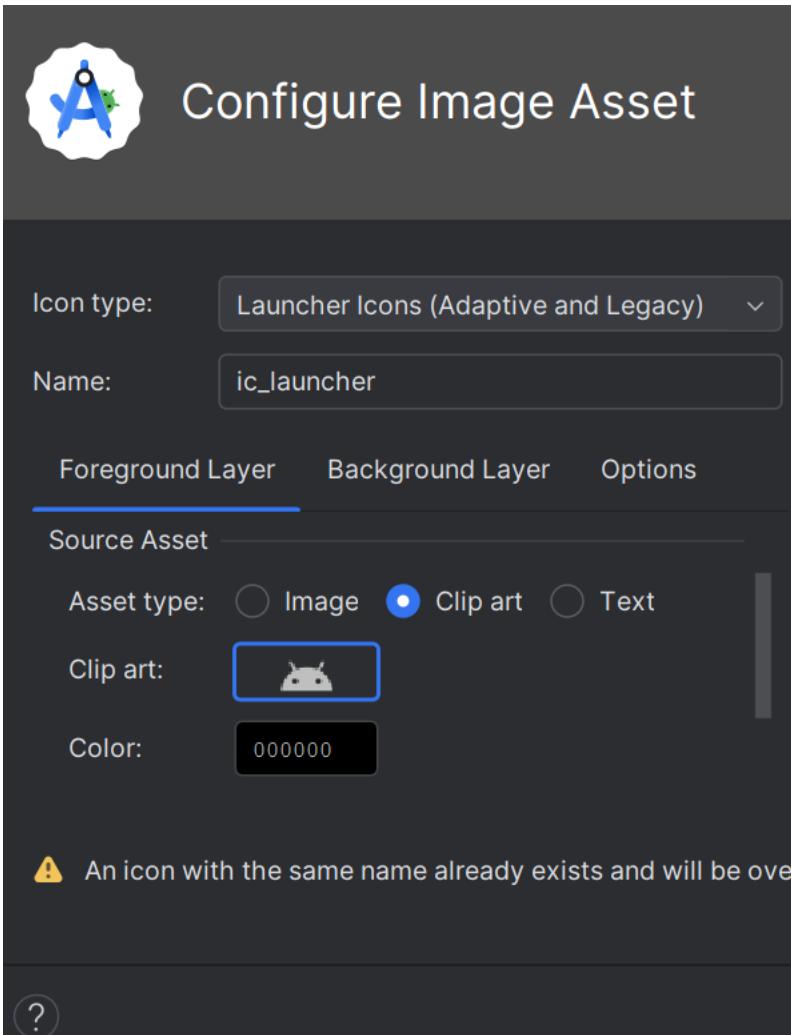
1.2 Thêm một tài nguyên hình ảnh cho biểu tượng khởi động

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

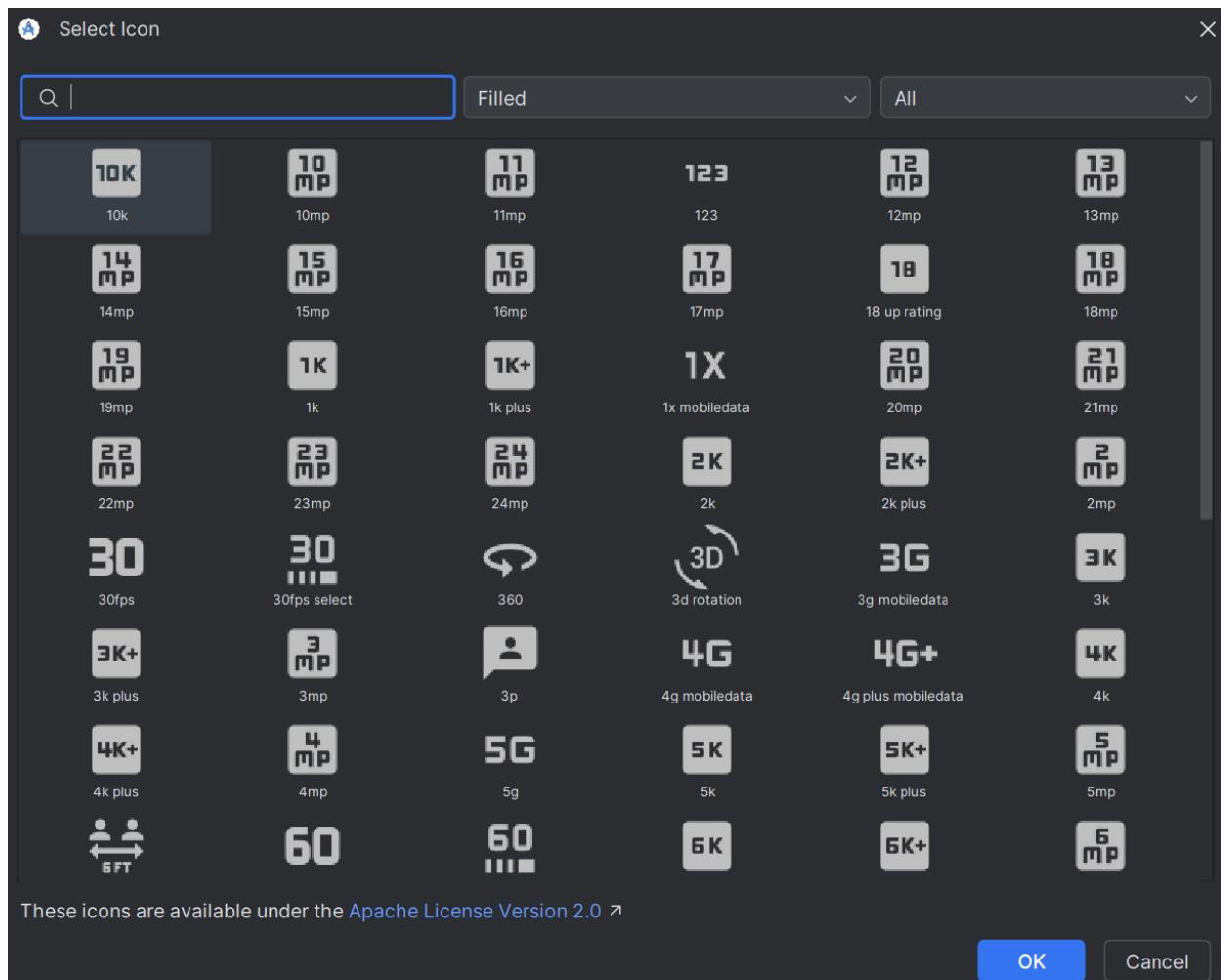
1. Mở dự án ứng dụng **HelloToast** từ bài học trước về cách sử dụng trình chỉnh sửa bộ cục, hoặc tạo một dự án ứng dụng mới.
2. Trong khung **Project > Android**, nhấp chuột phải (hoặc nhấn **Control** và nhấp chuột) vào thư mục **res** và chọn **New > Image Asset**. Cửa sổ **Configure Image Asset**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

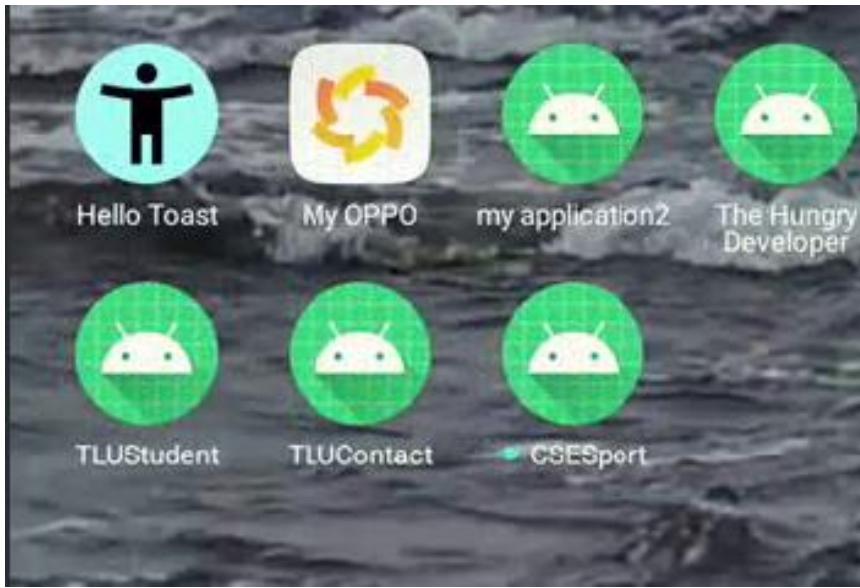


5. Nhập vào biểu tượng trong trường **Clip Art**. Các biểu tượng từ bộ icon Material Design sẽ xuất hiện.
6. Duyệt qua cửa sổ **Select Icon**, chọn một biểu tượng phù hợp (chẳng hạn như biểu tượng mood để gợi ý tâm trạng tốt), sau đó nhấp vào **OK**.



- 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.
- 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.
- 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.

TASK 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. Interdu... et
malesuada fames ac ante ipsum

Kiến trúc của mẫu Hoạt động cơ bản

#	Mô tả giao diện người dùng	Tham chiếu trong mã
---	----------------------------	---------------------

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

	World". Hãy thay thế nó bằng các phần tử giao diện người dùng (UI elements) phù hợp với ứng dụng của bạn.	elements) của ứng dụng đều được định nghĩa trong tệp này.
7	Nút hành động nổi (Floating Action Button – FAB)	Trong activity_main.xml được thêm vào dưới dạng một phần tử giao diện người dùng (UI element) sử dụng biểu tượng clip-art. Trong MainActivity.java 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()

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

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

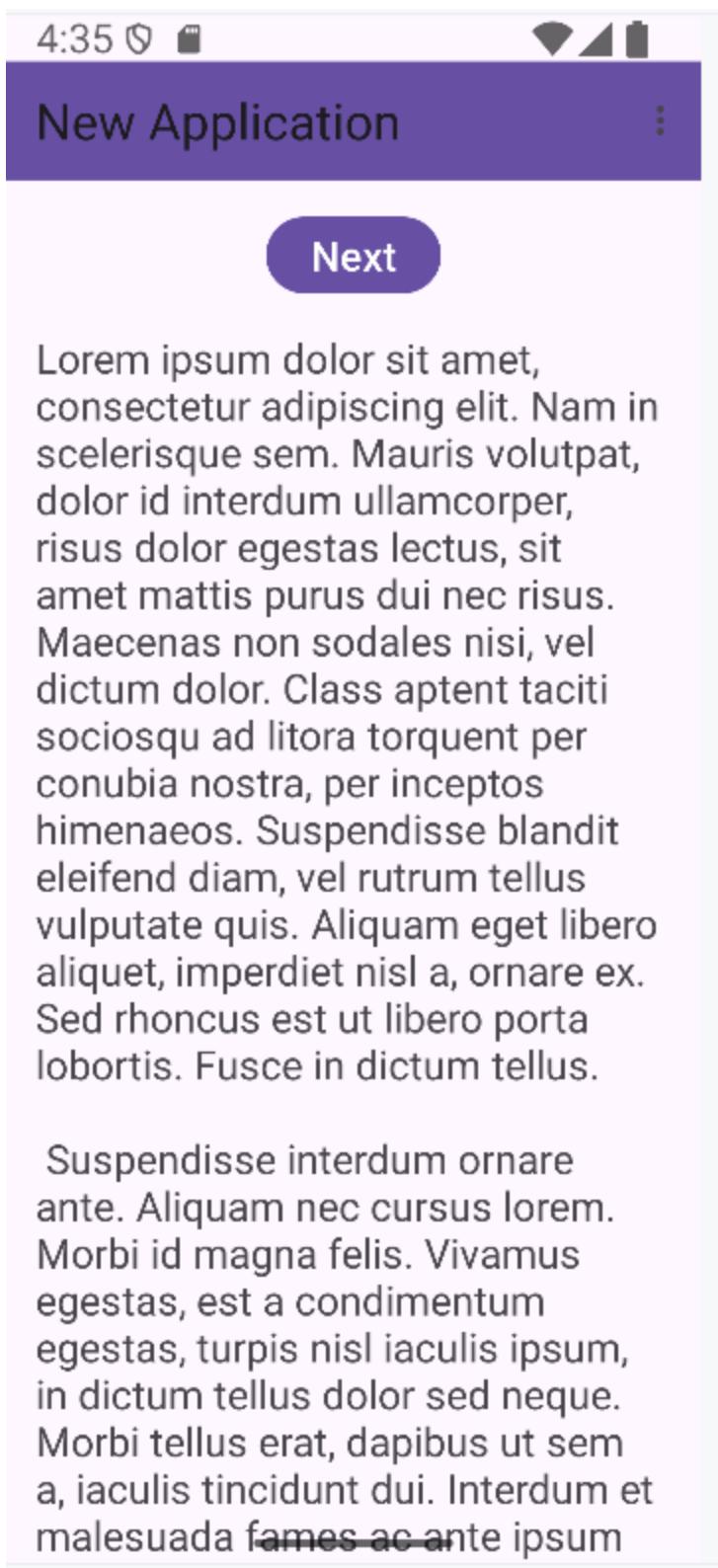
3. Để xóa nút hành động nổi khỏi bố cục, xóa khối mã XML sau
 - o Trong activity_main.xml.

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



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

- **2.3 Khám phá cách thêm hoạt động bằng mẫu**

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.

- 3.1 Mã nguồn mẫu Android

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

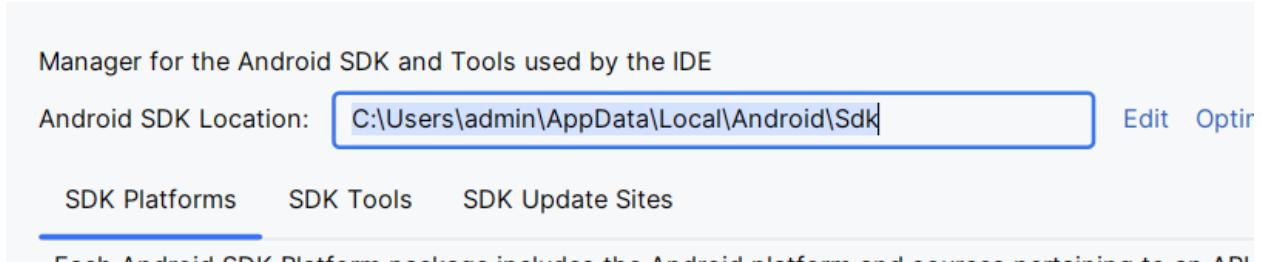
1. Trong Android Studio, chọn **File > New > Import Sample**.
2. Duyệt qua các mẫu có sẵn.
3. Chọn một mẫu và nhấp vào **Next**.
4. Chấp nhận các thiết lập mặc định và nhấp vào **Finish**.

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.

- 3.2 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**.
3. Sao chép đường dẫn trong phần **Android SDK Location** (bạn sẽ cần nó để tìm tài liệu trên máy tính).



4. Nhấp vào tab **SDK Platforms** để cài đặt các phiên bản Android bổ sung.
5. Nhấp vào tab **SDK Update Sites** để xem các trang cập nhật mà Android Studio kiểm tra thường xuyên.
6. Nhấp vào tab **SDK Tools** để cài đặt các công cụ SDK bổ sung và tài liệu nhà phát triển Android ngoại tuyến.
7. Chọn hộp kiểm **Documentation for Android SDK** nếu nó chưa được cài đặt, rồi nhấp vào **Apply**.
8. Khi quá trình cài đặt hoàn tất, nhấp vào **Finish**.
9. Điều hướng đến thư mục **sdk** đã sao chép trước đó, mở thư mục **docs**.
10. 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?*"
- **4.1 Tìm kiếm trên Stack Overflow bằng thẻ (tags)**

Truy cập **Stack Overflow** và nhập **[android]** vào ô tìm kiếm. Dấu **[]** giúp bạn tìm các bài viết có 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
- **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**.

1.5) Tài nguyên có sẵn

Bài 2) Activities

2.1) Activity và Intent

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 lỏng lẻo với nhau. Mỗi màn hình là một activity. Thông thường, một activity trong ứng dụng được chỉ định là

"main activity" (**MainActivity.java**), 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 bố cục, và tạo một phương thức cơ bản cho sự kiện nhấn nút **onClick**.

1.1 Tạo dự án TwoActivities

1. Mở **Android Studio** và tạo một **dự án Android Studio mới**.
 - o Đặt tên ứng dụng là **Two Activities** và chọn cài đặt **Phone and Tablet** giống như các bài thực hành trước.
 - o 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**.
 - o Đảm bảo rằng **Generate Layout file** và **Backwards Compatibility (AppCompat)** đã được chọn.
4. Nhấn **Finish** để hoàn tất tạo dự án.
 - **1.2 Định nghĩa bố cục cho Main Activity**
 1. Mở **res > layout > activity_main.xml** trong **Project > Android**. Trình chỉnh sửa bố cục (**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 bố cục. **Autoconnect** sẽ tự động tạo các ràng buộc (**constraints**) cho **Button**.
 4. Trong **Attributes**, đặt các thuộc tính sau:
 - o **ID:** button_main
 - o **layout_width** và **layout_height:** wrap_content
 - o **Text:** Send

Lúc này, bố cục sẽ trông như sau:

TwoActivities

Send

5. Nhấp 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:onClick="launchSecondActivity"
    android:text="@string/button_main"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintRight_toRightOf="parent" />
```

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 bối cảnh thông qua thuộc tính android:onClick.

1. Nhấp 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!".
- 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:

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

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

- 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:

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

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**:
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.

- 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:



3. Nhấp 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" />
```

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:

1. **Mở MainActivity.java**.
2. **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);
```

3. 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 nhấn 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:

Message Received

Enter Your Reply Here

Reply

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">
    <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>
```

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**:
3. Thêm một biến riêng tư ở đầu lớp để chứa **EditText**.
4. Trong phương thức **onCreate()**, sử dụng **findViewById()** để lấy tham chiếu đến **EditText** và gán nó cho biến riêng tư đó
5. Trong phương thức **launchSecondActivity()**, ngay dưới Intent mới, lấy văn bản từ **EditText** dưới dạng chuỗi.
6. Thêm chuỗi đó vào Intent dưới dạng một extra với hằng số **EXTRA_MESSAGE** làm khóa và chuỗi làm giá trị

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

Phương thức **launchSecondActivity()** 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);  
    mReplyHeadTextView = findViewById(R.id.text_header_reply);  
    mReplyTextView = findViewById(R.id.text_message_reply);  
}
```

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

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="@style/TextAppearance.AppCompat.Medium"
    app:layout_constraintTop_toBottomOf="@+id/text_header"
    app:layout_constraintStart_toStartOf="parent" />
```

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

- Mở SecondActivity để thêm mã vào phương thức onCreate().
- Lấy Intent kích hoạt Activity này:
- Lấy chuỗi chứa thông điệp từ extras của Intent bằng cách sử dụng biến tĩnh MainActivity.EXTRA_MESSAGE làm khóa:

4. Sử dụng `findViewById()` để tham chiếu đến `TextView` cho thông điệp từ giao diện:
 5. Đặt văn bản của `TextView` thành chuỗi từ `extra` của `Intent`:
 6. 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:

```
7. public class MainActivity extends AppCompatActivity {
    private static final String LOG_TAG =
        MainActivity.class.getSimpleName();
    public static final String EXTRA_MESSAGE =
        "com.example.android.twoactivities.extra.MESSAGE";
    private EditText mMessageEditText;
    public static final int TEXT_REQUEST = 1;
    private TextView mReplyHeadTextView;
    private TextView mReplyTextView;

    @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);
    }
}
```

4.1 Thêm một `EditText` và một `Button` vào giao diện `SecondActivity`

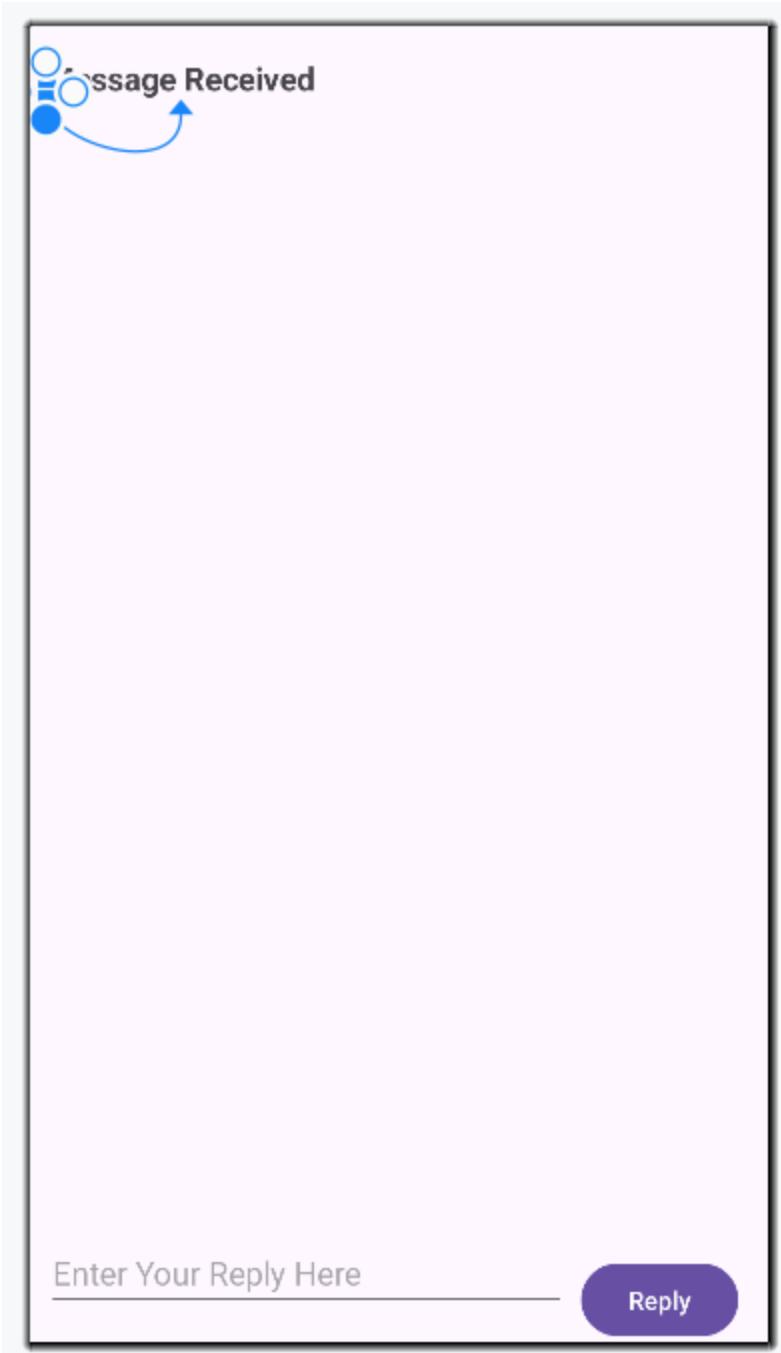
1. Mở tệp `strings.xml` và thêm các tài nguyên chuỗi cho văn bản `Button` và gợi ý (`hint`) cho `EditText` mà bạn sẽ thêm vào `SecondActivity`:
2. Mở tệp `activity_main.xml` và `activity_second.xml`.
3. Sao chép `EditText` và `Button` từ tệp `layout activity_main.xml` và dán chúng vào `layout activity_second.xml`.
4. Trong `activity_second.xml`, sửa đổi giá trị thuộc tính cho `Button` như sau:

Old attribute value	New attribute value
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:

Old attribute value	New attribute value
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 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="@style/TextAppearance.AppCompat.Medium"
        app:layout_constraintTop_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>
```

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();
```

8. 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_CANCELED.

```
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.

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

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

- Mở tệp activity_main.xml và activity_second.xml.

- Sao chép hai phần tử TextView từ tệp layout activity_second.xml và dán chúng vào giao diện activity_main.xml phía trên Button.
- 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"

- 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"

- 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 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() đầy đủ bây giờ sẽ trông như sau:

```
@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);
}
```

5. Trong phương thức launchSecondActivity(), thay đổi lời gọi từ startActivityForResult() thành startActivityForResult(), và bao gồm khóa TEXT_REQUEST làm đối số:

```
startActivityForResult(intent, TEXT_REQUEST);
```

6. **Ghi đè phương thức callback onActivityResult() với chữ ký sau:**

```
public void onActivityResult(int requestCode, int resultCode, Intent data)
```

Bà tham số trong onActivityResult() chứa 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.

7. **Bên trong onActivityResult(), gọi super.onActivityResult():**

```
super.onActivityResult(requestCode, resultCode, data);
```

8. **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:**

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);
```

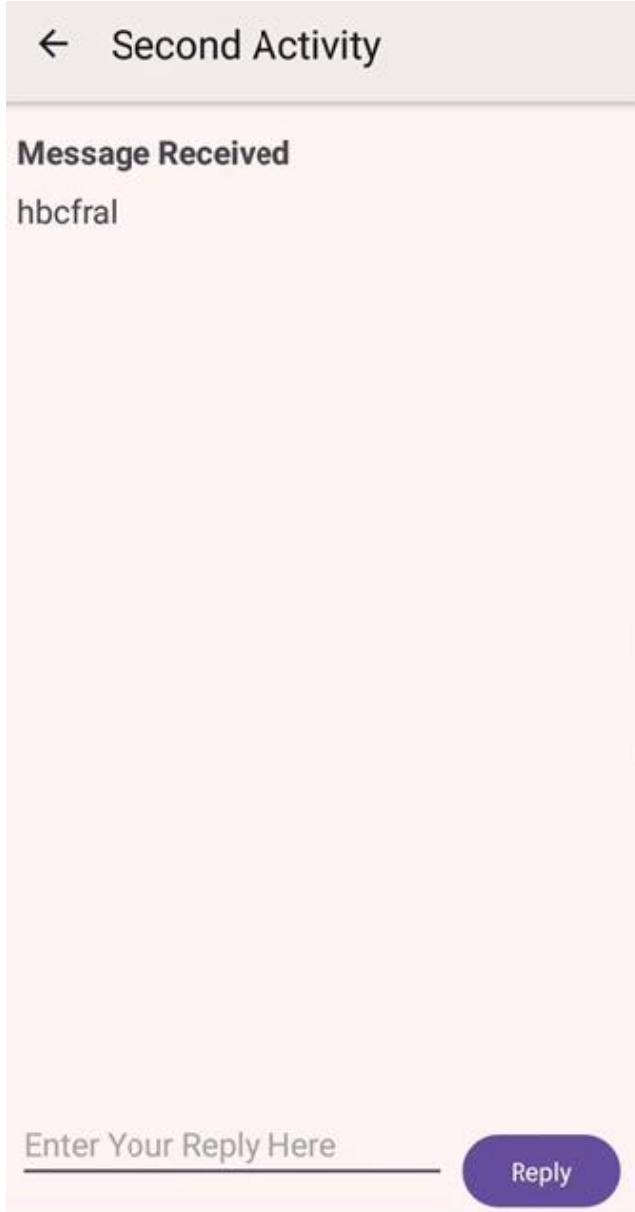
Phương thức onActivityResult() đầy đủ:

```
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.A



2.2) Vòng đời của Activity và trạng thái

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 onCreate
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).

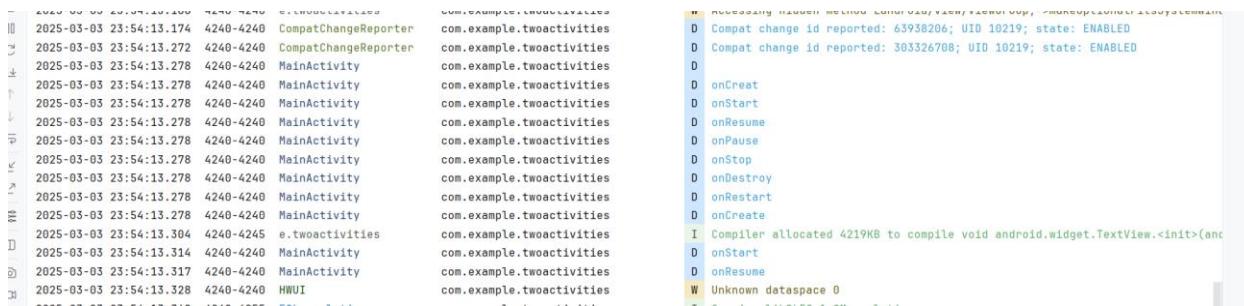
4. Thêm một câu lệnh log vào phương thức returnReply() ngay trước khi gọi finish().

```
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(@NonNull 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()**.

- 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("reply_visible");
}
```

- 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());  
    }  
}
```

Phương thức **onCreate()**:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    mReplyHeadTextView = findViewById(R.id.reply_head);  
    mReplyTextView = findViewById(R.id.reply_text);  
  
    if(savedInstanceState.getBoolean("reply_visible", false))  
    {  
        mReplyHeadTextView.setVisibility(View.VISIBLE);  
        mReplyTextView.setText(savedInstanceState.getString("reply_text"));  
    }  
}
```

```

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 thể nó 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 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()

2.3) Intent ngầm định

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:

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.
- Chỉnh 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**.

- 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:

- 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>
```

- Mở res > layout > activity_main.xml trong bảng **Project > Android**. Nhấp vào tab **Text** để chuyển sang mã XML.
- Thay đổi **android.support.constraint.ConstraintLayout** thành **LinearLayout**, như bạn đã học trong bài thực hành trước.
- 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">
```

- Xóa **TextView** hiển thị "Hello World".
- 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:

EditText attribute	Value
android:id	"@+id/website_edittext"
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:text	"@string/edittext_uri"
Button attribute	Value
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)

EditText attribute	Value
android:id	"@+id/location_edittext"
android:text	"@string/edittext_loc"
Button attribute	Value
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.

- 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.)

EditText attribute	Value
android:id	"@+id/share_edittext"
android:text	"@string/edittext_share"
Button attribute	Value
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>

```

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()

- Nhấp vào "openWebsite" trong mã XML activity_main.xml.
- 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) {
}

```

- 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( tag: "ImplicitIntents", msg: "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;
```

4. 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()**

1. Trong phương thức mới **openLocation()**, thêm một lệnh để lấy giá trị chuỗi từ đối tượng **mLocationEditText**:

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

2. 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);
```

3. 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);
```

4. 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. Nhấp 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;
```

4. 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()

1. Trong phương thức **shareText()**, thêm một lệnh để lấy giá trị chuỗi từ đối tượng **mShareTextEdit**:

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

2. Xác định loại MIME của văn bản cần chia sẻ:

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

3. 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();
}
```

4. 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(this)
        .setType(mimeType)
        .setChooserTitle(R.string.share_text_with)
        .setText(txt)
        .startChooser();
}
```

4.3 Chạy ứng dụng

1. Chạy ứng dụng.
2. 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.

Implicit Intents

<http://developer.android.com>

[Open Website](#)

[Golden Gate Bridge](#)

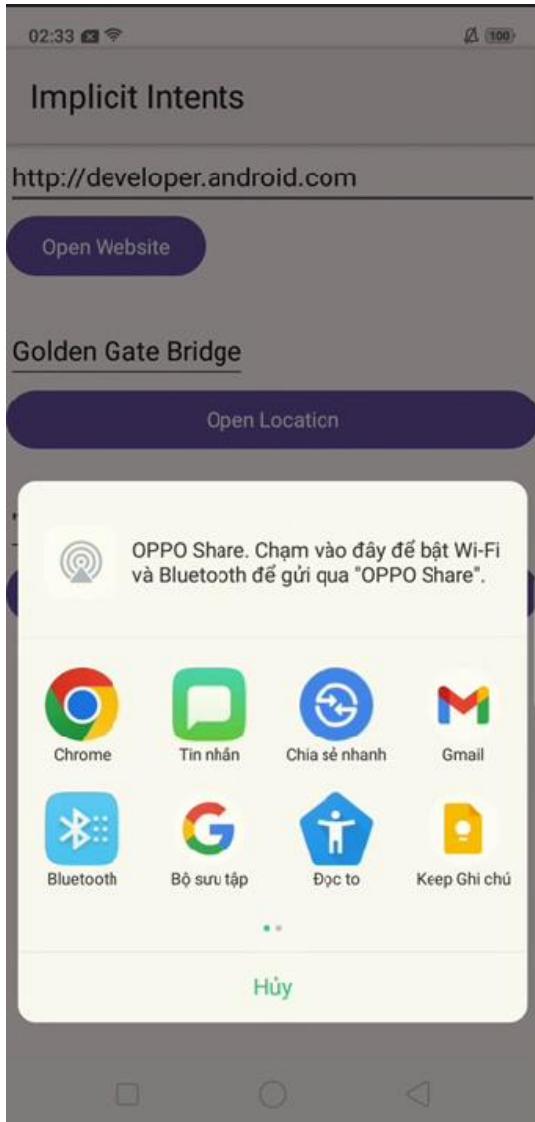
[Open Location](#)

[Twas brillig and the slithy toves](#)

[Share This Text](#)



3. 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 tường 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 bố cục

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:

Attribute	Value
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>.

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 chưa. 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 khối if, lấy TextView để hiển thị thông báo:

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

7. Cũng trong khối 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.

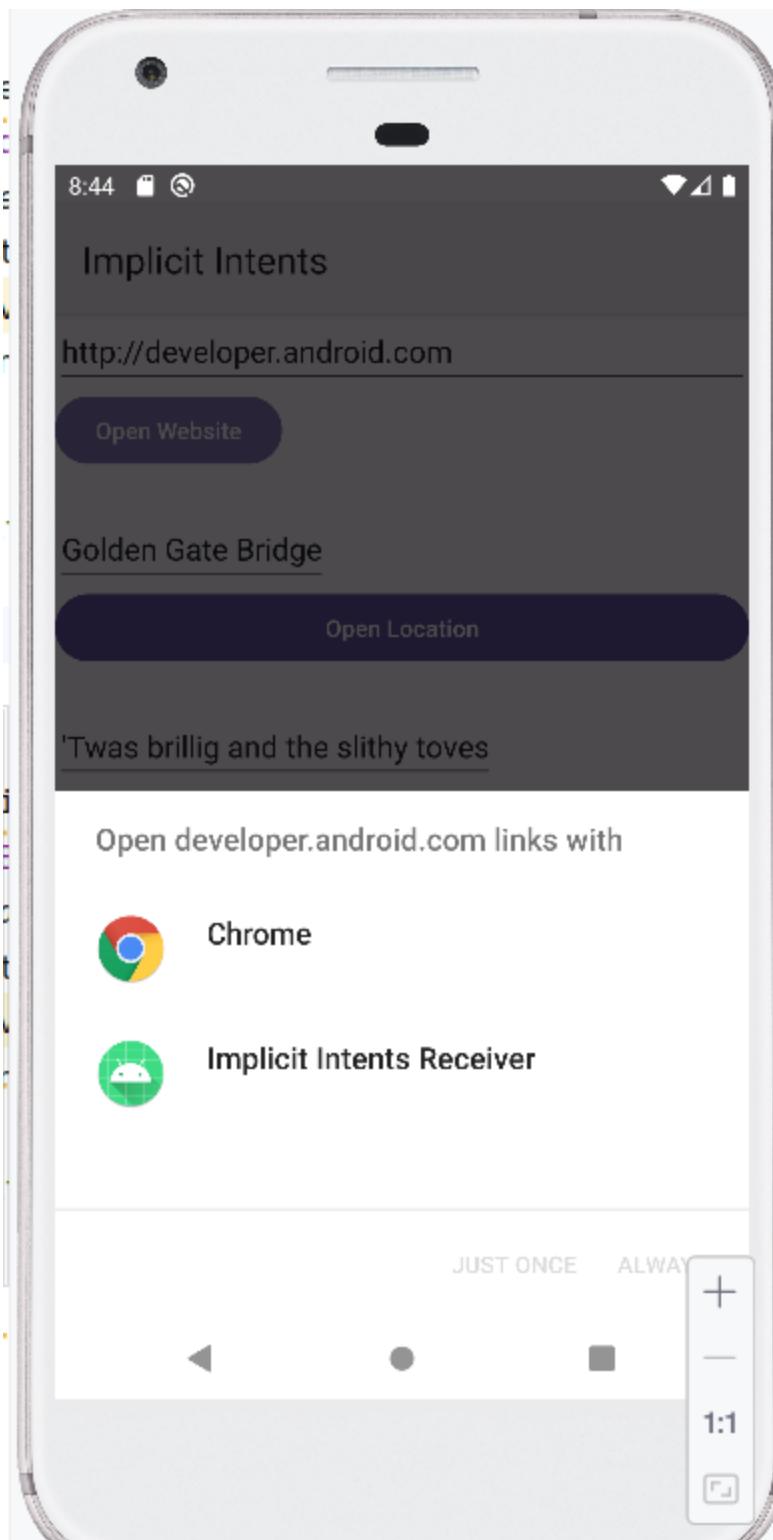
1. Chạy ứng dụng **Implicit Intents Receiver**.

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**.

- o Chọn **Implicit Intents Receiver**, sau đó nhấn **Just Once**.
- o 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). Bất kỳ URI nào khác sẽ được mở bằng trình duyệt web mặc định.

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**

B

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.

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

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(Unit4.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 hạ 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**.

- 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.
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, 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 đổi 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.

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

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 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

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 bố 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 bố 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 bố 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.

3.4) 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

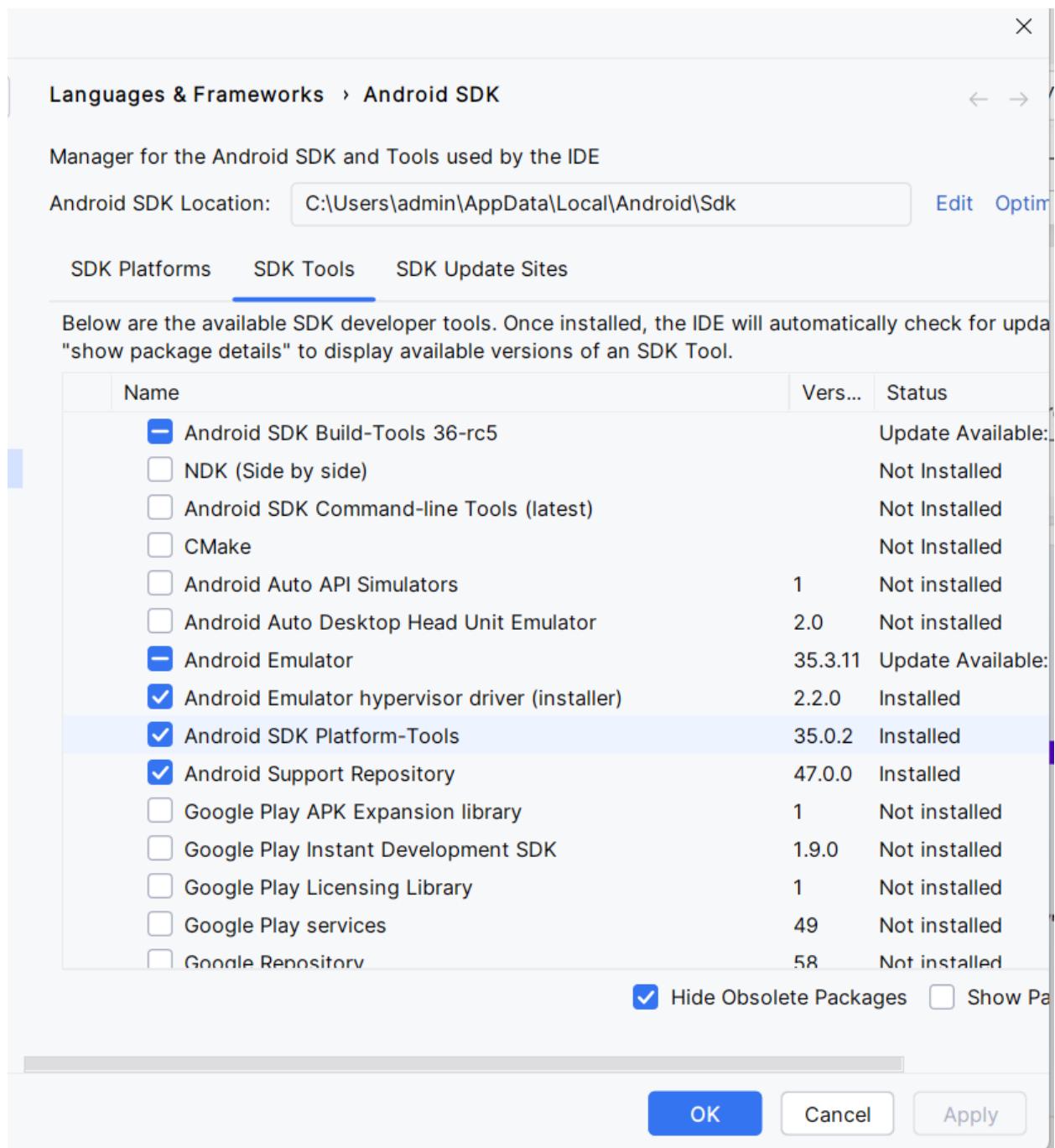
Trong nhiệm vụ này, bạn sẽ thiết lập một dự án mới cho ứng dụng **HelloCompat** và triển khai bố cục cũng như hành vi cơ bản.

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. Nhấp 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. Nhấp vào **OK**.

18. Nhấp vào **OK** một lần nữa, sau đó nhấp vào **Finish** khi kho 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 nhấp 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. 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.

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.

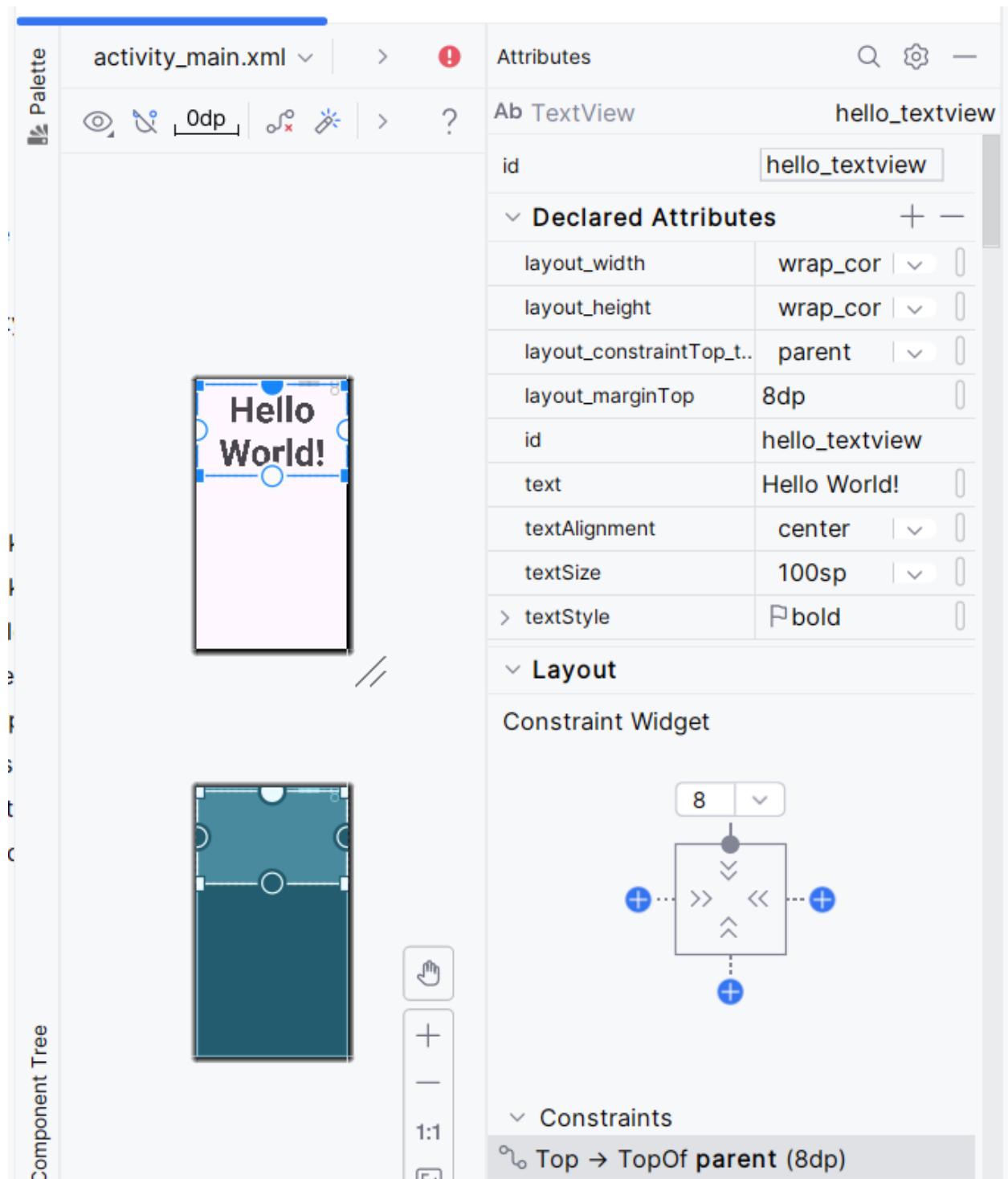
3. Chọn **TextView** "Hello World" trong bố cục và mở bảng **Attributes**.

4. Thay đổi các thuộc tính của **TextView** như sau:

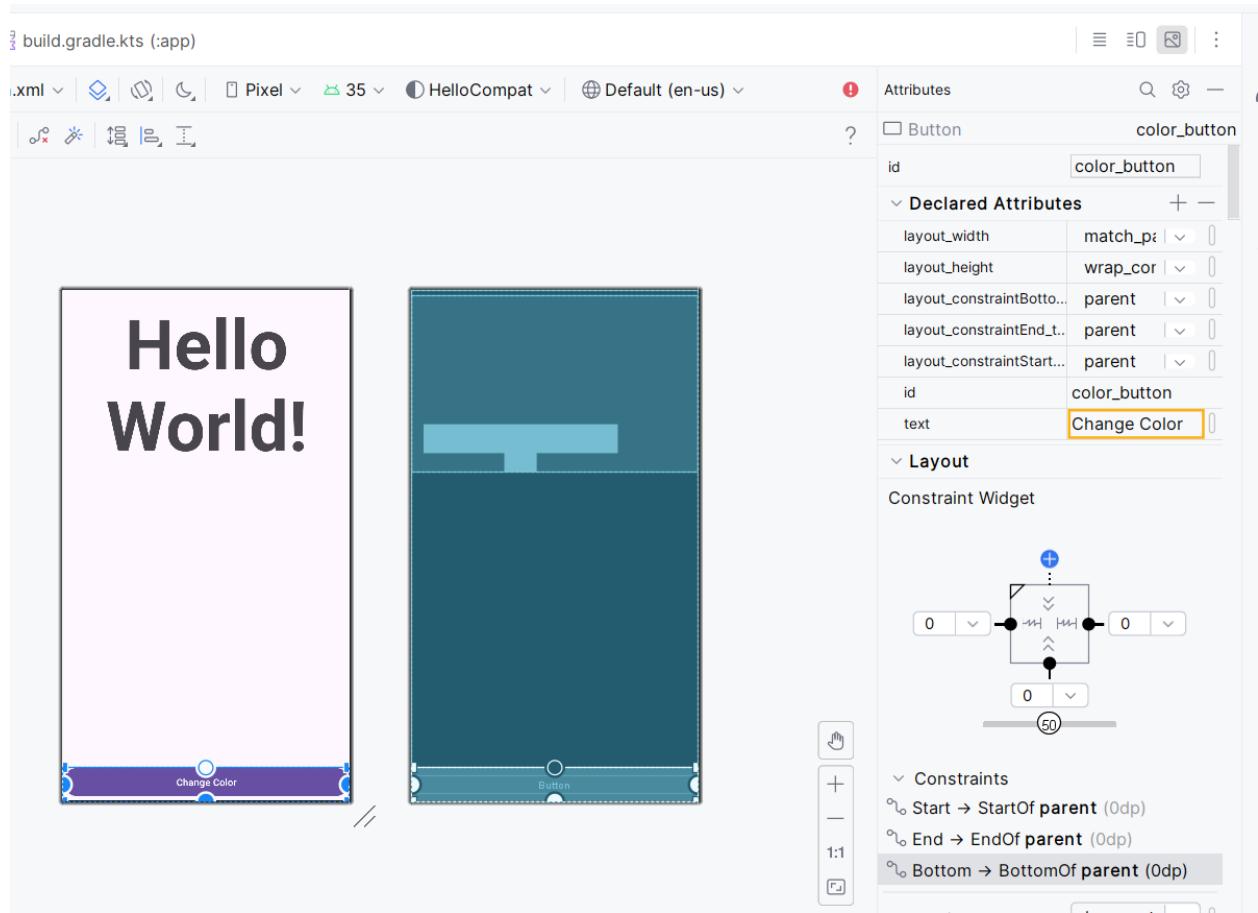
Attribute field	Enter the following:
ID	hello_textview
textStyle	B (bold)
textAlignment	Center the paragraph icon
textSize	100sp

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.



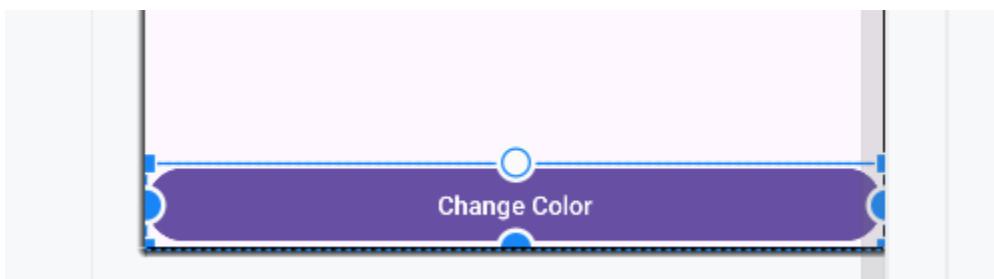
6. Kéo một **Button** vào phía dưới của bố cục, và thêm các ràng buộc (constraints) vào các cạnh trái, phải, và phía dưới của bố cục, như hiển thị trong hình dưới đây.



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:

Attribute field	Enter the following:
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. 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.

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 bố cục XML và một đoạn mã trong lớp MainActivity của ứng dụng **HelloCompat** cho đến thời điểm này.

Tệp bố 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. Nhấp 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 nhấp **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 cho phép **MainActivity**
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	Android Studio:	Studio
Dự án	Android	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 cấp 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 **này**
Chạy ứng dụng **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.

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

1.1) Hình ảnh có thể chọn

Giới thiệu

Giao diện người dùng (**UI - User Interface**) hiển thị trên màn hình của thiết bị chạy **Android** được tạo thành từ một **cây phân cấp** gồm các đối tượng gọi là **View**. Mỗi phần tử trên màn hình đều là một **View**.

Lớp **View** là khối xây dựng cơ bản của tất cả các thành phần UI. Đây là lớp cha của các thành phần UI có tính tương tác, chẳng hạn như **Button**. Một **Button** là một thành phần UI mà người dùng có thể nhấn hoặc bấm để thực hiện một hành động nào đó.

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

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

Kiến thức cần có trước

Bạn nên có khả năng:

- Tạo một dự án **Android Studio** từ mẫu và tạo bố cục (**layout**) chính.
- Chạy ứng dụng trên trình giả lập (**emulator**) hoặc thiết bị kết nối.
- Tạo và chỉnh sửa các thành phần UI bằng **layout editor** và **mã XML**.
- Truy cập các phần tử UI từ mã nguồn bằng **findViewById()**.

- Xử lý sự kiện **click** của **Button**.
- Hiển thị thông báo **Toast**.
- Thêm hình ảnh vào thư mục **drawable** của dự án.

Những gì bạn sẽ học

- Cách sử dụng hình ảnh như một phần tử tương tác để thực hiện một hành động.
- Cách thiết lập thuộc tính cho **ImageView** trong **layout editor**.
- 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 mô phỏng đặt món tráng miệng, trong đó sử dụng hình ảnh làm phần tử tương tác.
- Thiết lập trình xử lý **onClick()** cho các hình ảnh để hiển thị các thông báo **Toast** khác nhau.
- Thay đổi **Floating Action Button (FAB)** mặc định của mẫu để hiển thị một biểu tượng khác và mở một **Activity** khác.

Task 1: Thêm hình ảnh vào bố 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.

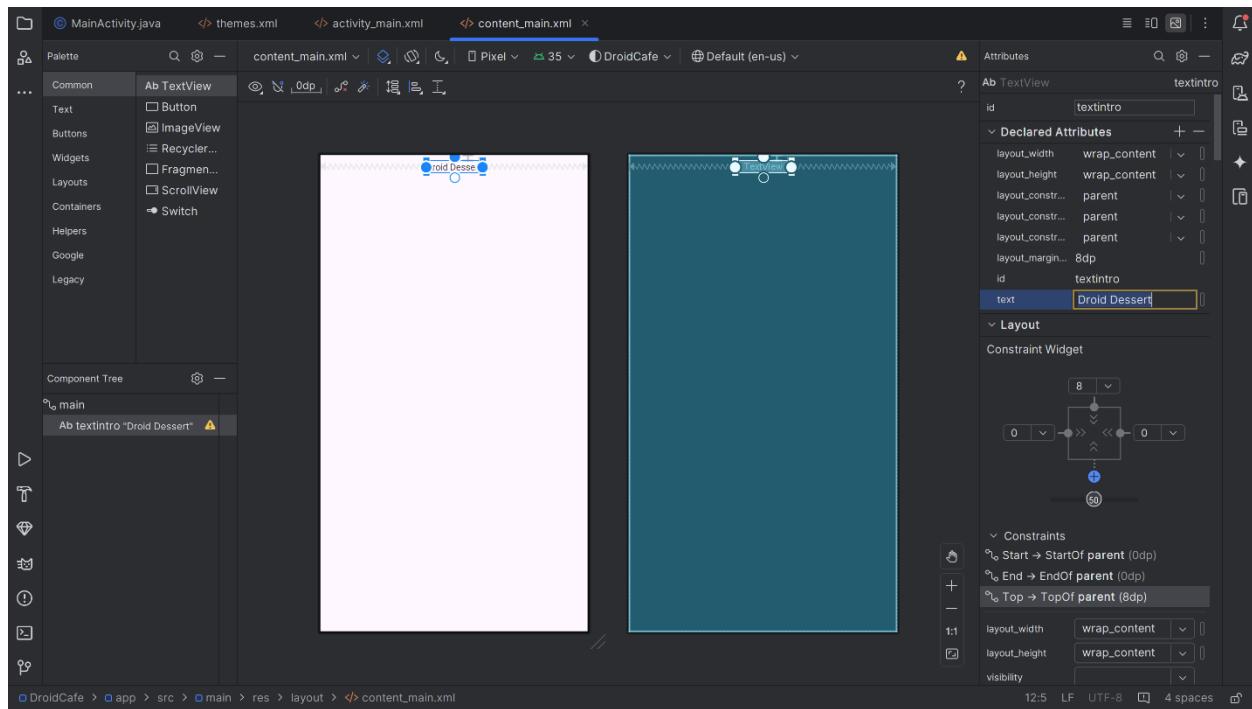
5. Chọn **TextView "Hello World"** trong layout và mở bảng **Attributes**.

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.



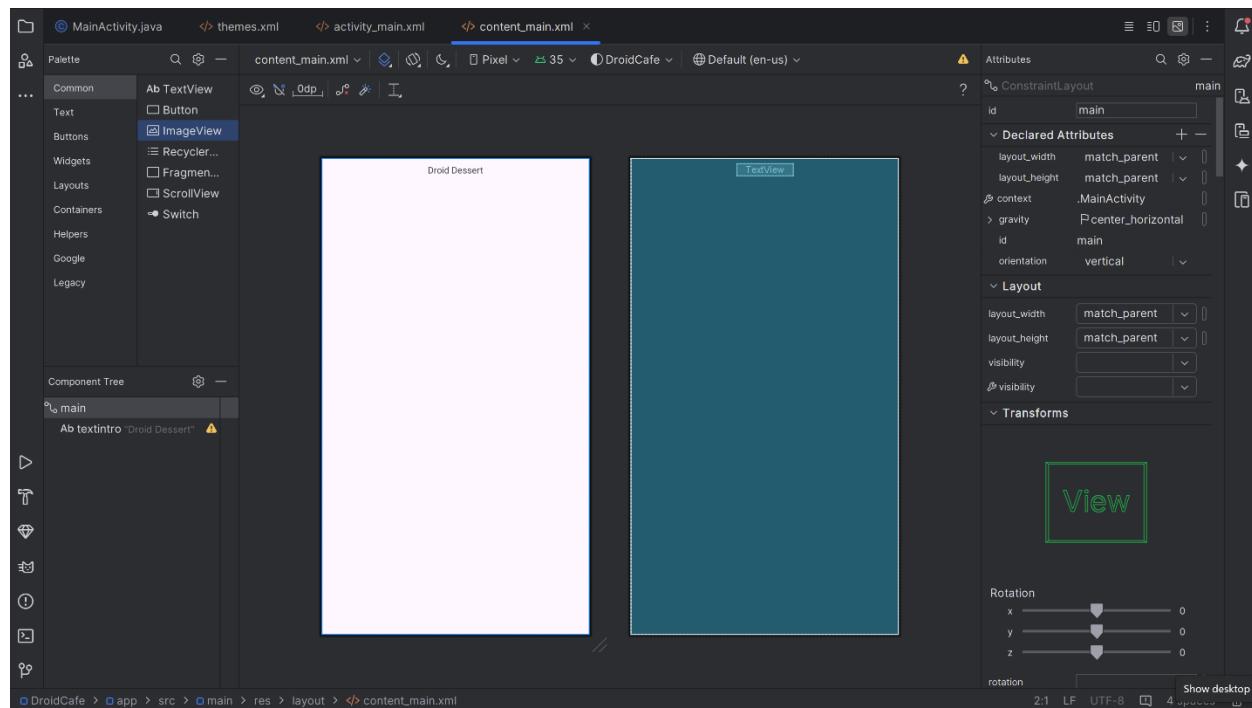
8. Trong bài học trước, bạn đã học cách trích xuất tài nguyên chuỗi từ một chuỗi văn bản tĩnh. Nhấp vào tab **Text** bên để chuyển sang mã XML và trích xuất chuỗi "Droid Desserts" trong

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 bố cục: sử dụng nút **Fix** trong các tin nhắn cảnh báo để trích xuất tài nguyên chuỗi.

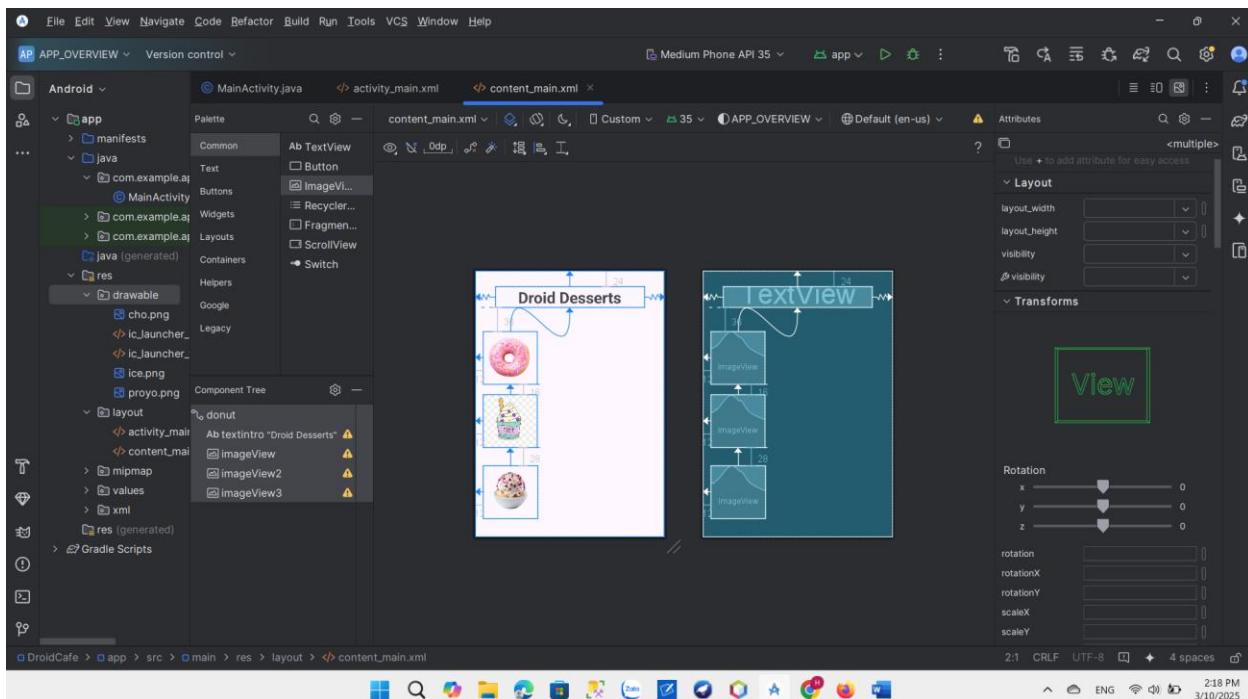
- Để sao chép hình ảnh vào dự án của bạn, trước tiên hãy đóng dự án.
- Sao chép các tệp hình ảnh vào thư mục **drawable** của dự án của bạn. Tìm thư mục **drawable** trong một dự án bằng cách sử dụng đường dẫn này: **project_name > app > src > main > res > drawable**.
- Mở lại dự án của bạn.
- Mở tệp **content_main.xml** và nhấp vào tab **Design** (nếu nó chưa được chọn).
- 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:

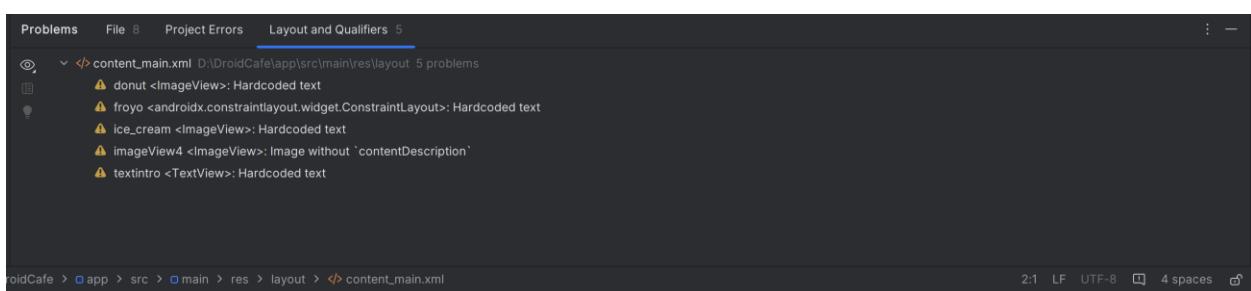
Attribute field	Nhập các thông tin sau
ID	ice_cream
contentDescription	Ice cream sandwiches have chocolate wafers and vanilla filling. (You can copy/paste the text into the field.)

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.

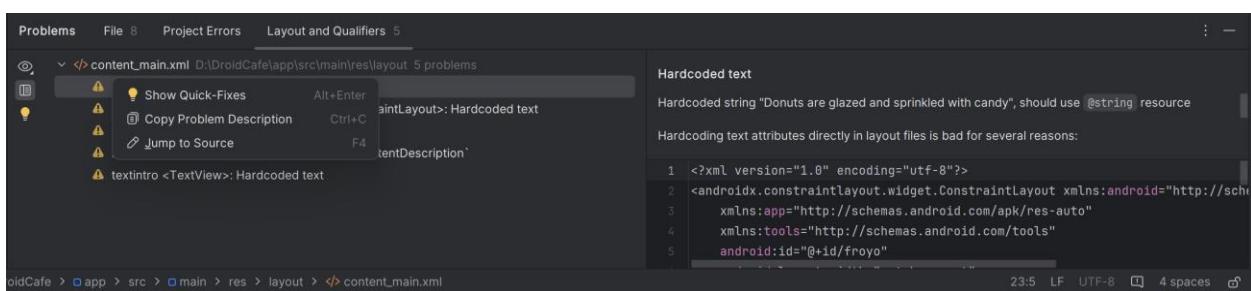
10. Trong bảng thuộc tính, nhập các giá trị sau cho các thuộc tính:

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. 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:



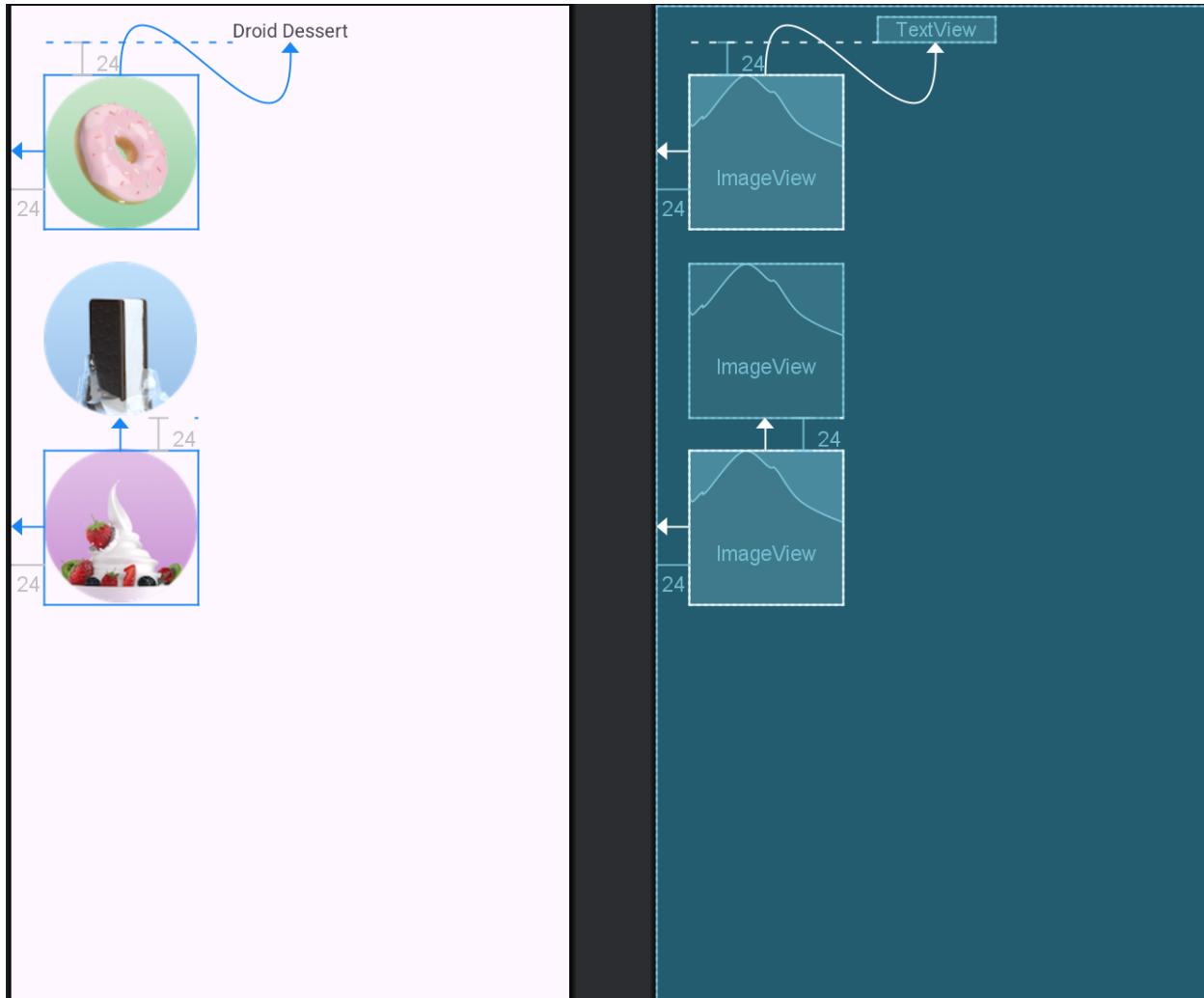
12. Mở rộng từng cảnh báo **Hardcoded text**, cuộn xuống dưới cùng của thông điệp cảnh báo, và nhấp vào nút **Fix** như hình bên dưới:



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



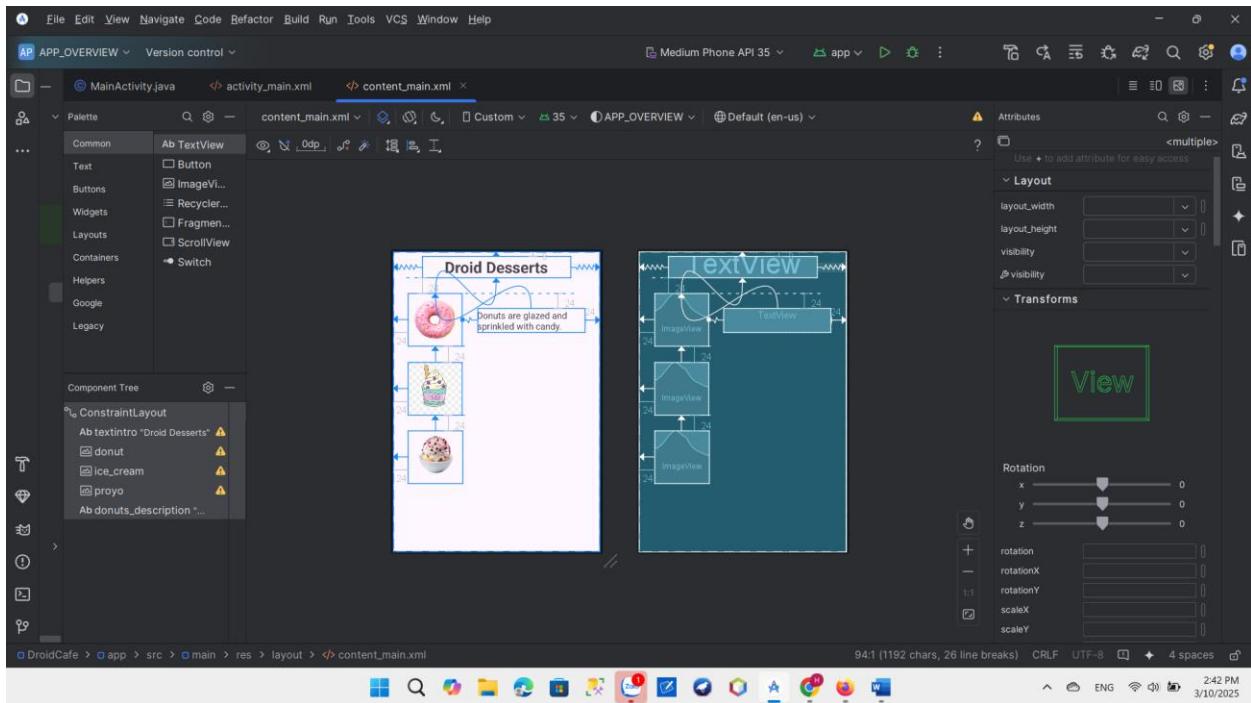
1.3 Thêm miêu tả văn bản

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

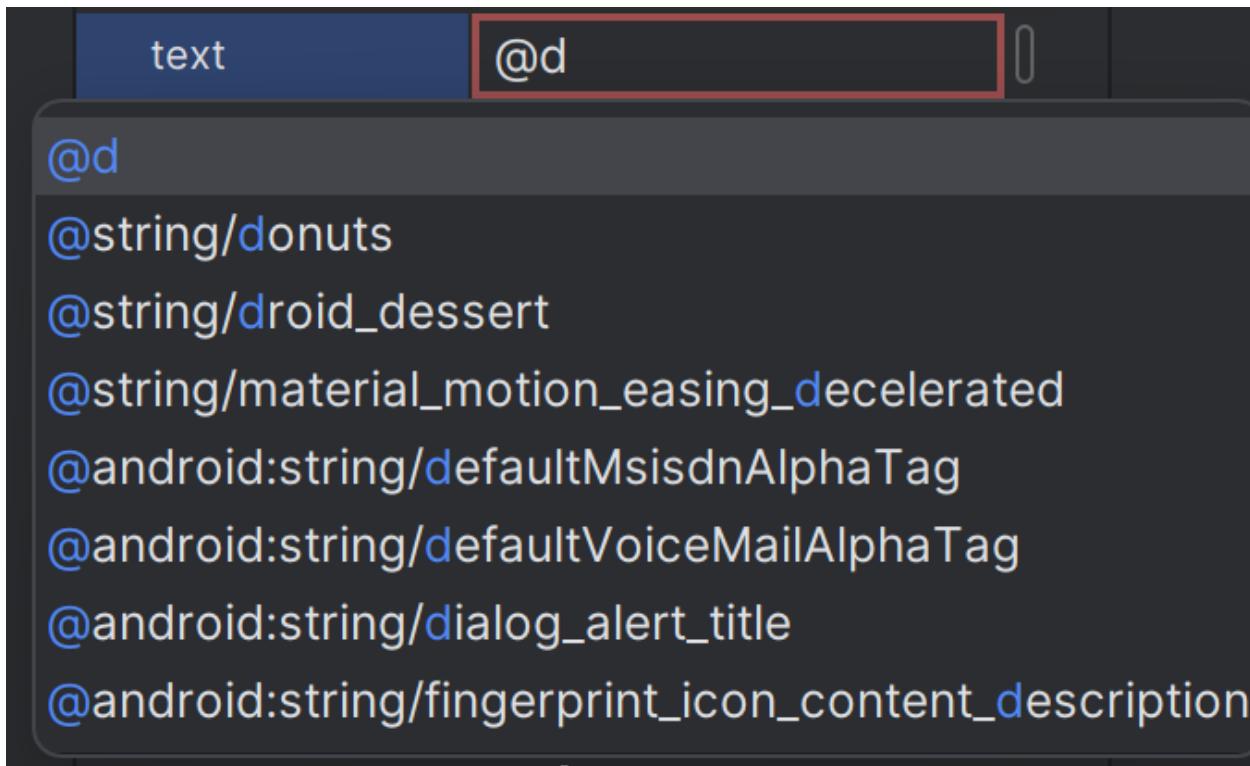
24

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. Nhấp 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:

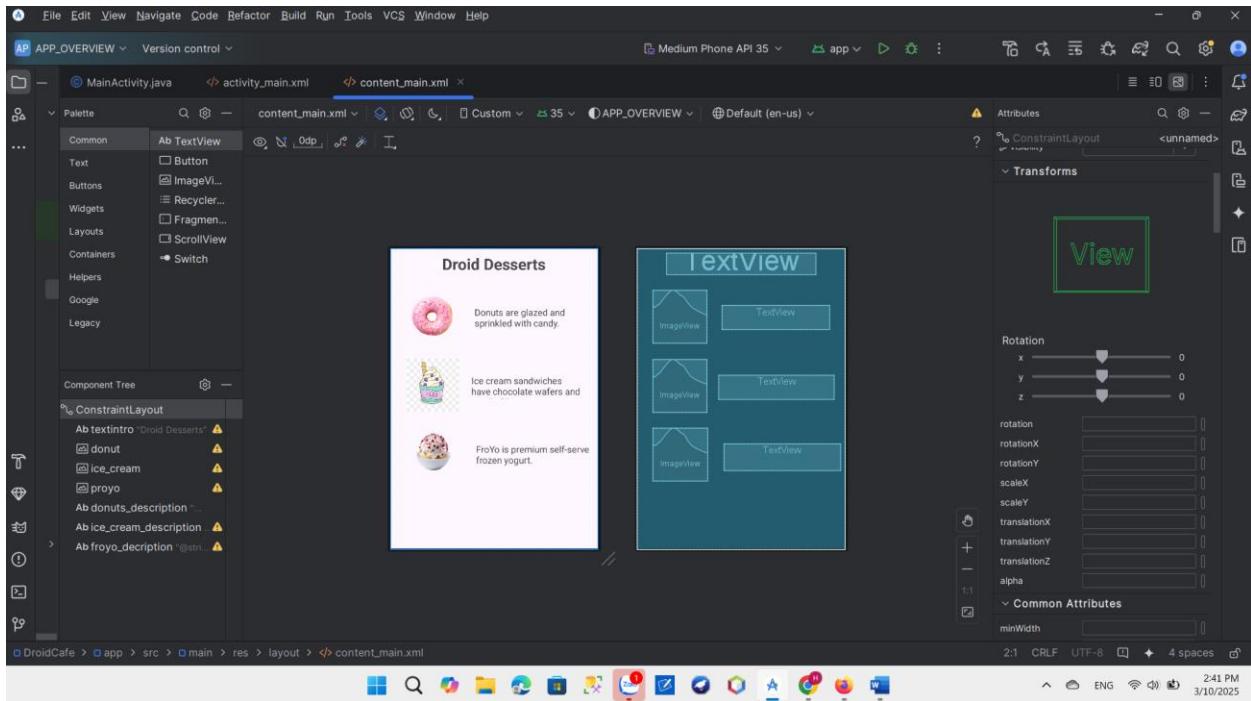
Attribute field	Enter the following:
ID	ice_cream_description
Left, right, and top margins	24
layout_width	match_constraint
text	@string/ice_cream_sandwiches

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 bố 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 bố 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 chéo 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.

3:11

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.



You ordered an ice cream sandwich.

Task 2 mã giải pháp

Mã giải pháp cho nhiệm vụ này được bao gồm trong mã và bố 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**. Bây 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. Chỉnh sửa:
 - o **Activity Name** thành **OrderActivity**.
 - o **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 giò hàng. Một **Activity** trống (**OrderActivity**) sẽ xuất hiện. Nhấn nút **Back** để quay lại **MainActivity**.

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 bố 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:

```
2 usages  
public static final String EXTRA_MESSAGE = "com.example.android.droidcafe.extra.MESSAGE";
```

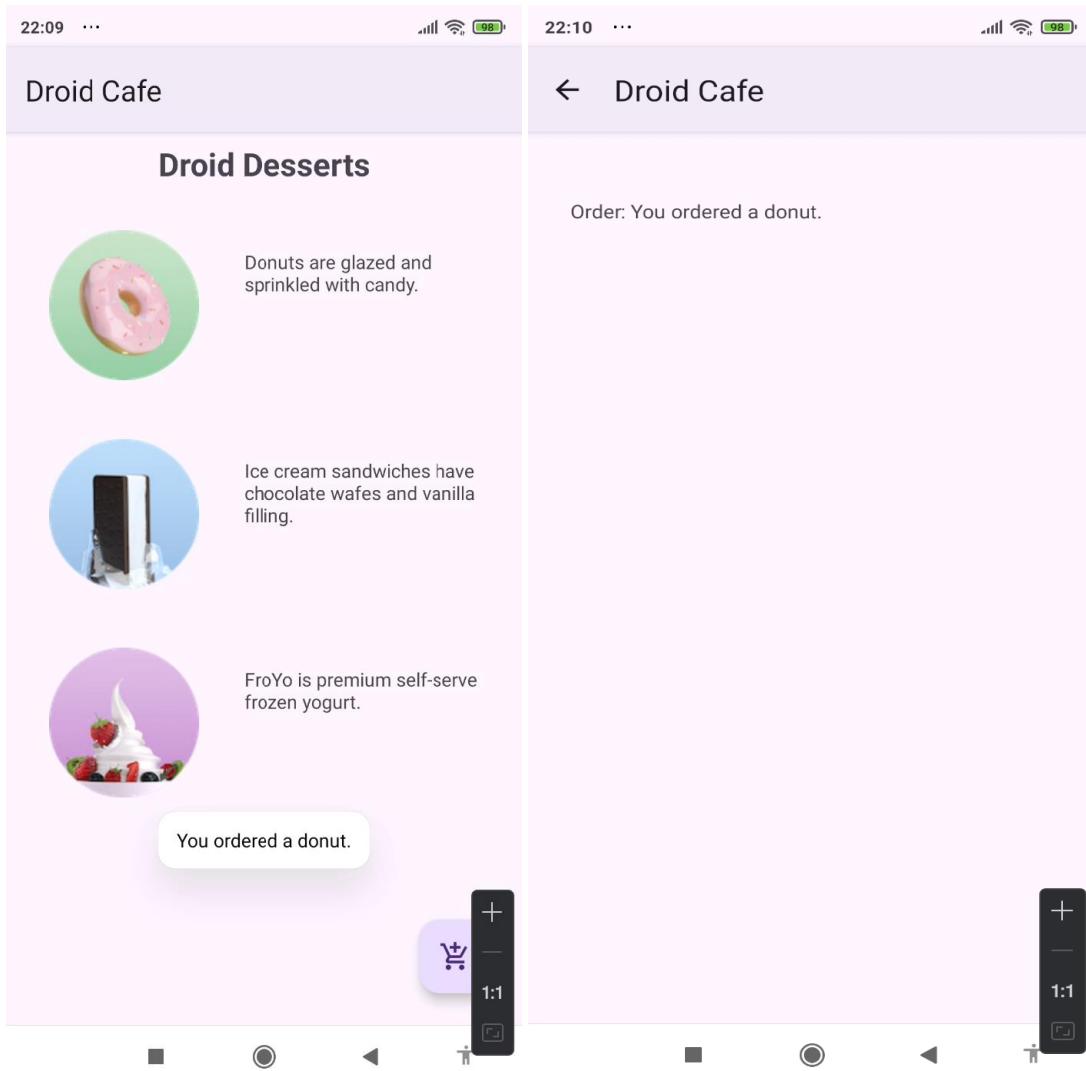
5. Thay đổi phương thức onClick() để bao gồm câu lệnh intent.putExtra trước khi khởi động OrderActivity:

```
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 thể 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. 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 nhấp 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**. Nhấp 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 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).

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:

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 bố cục bằng cách nhấn 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 bố 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 bố 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 bố 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):

EditText attribute	Value
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 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.

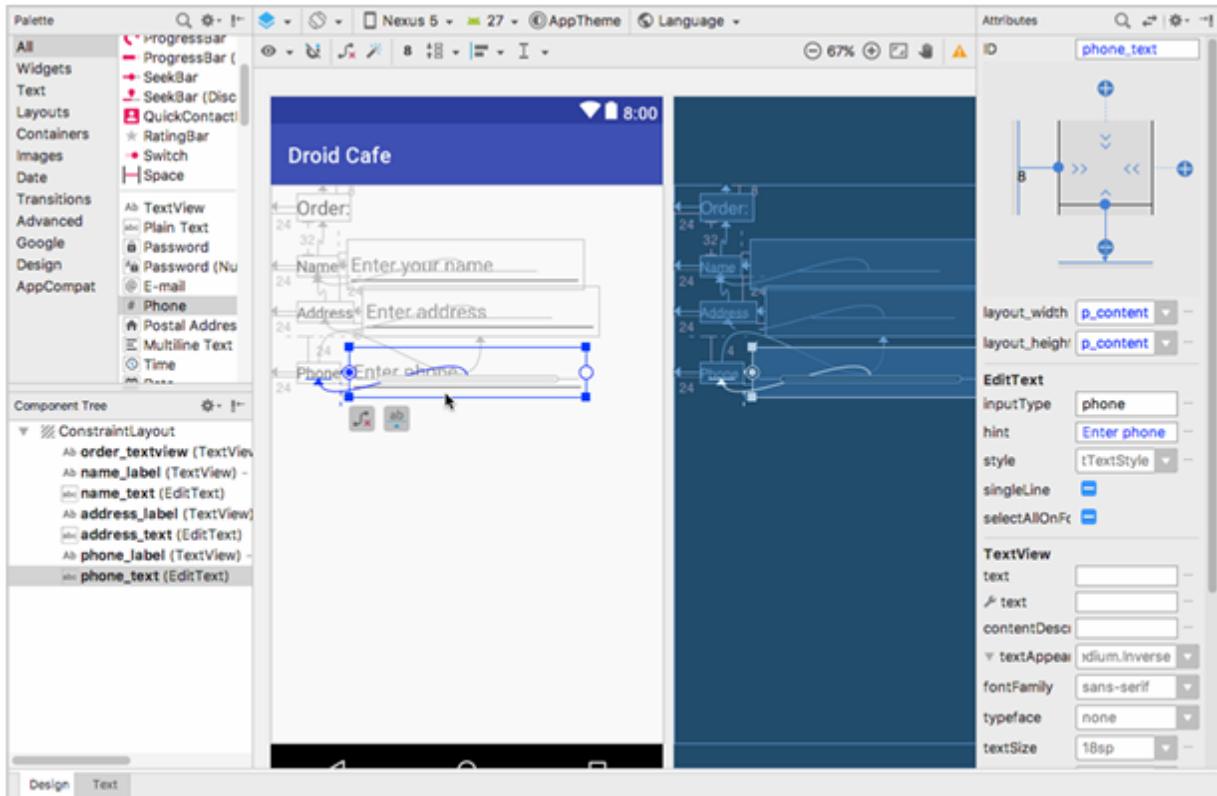
1. Mở **activity_order.xml** và thêm một phần tử **TextView** được ràng buộc ở dưới phần tử **note_text** đã có trong layout, và ràng buộc với lề trái, như trong hình dưới đây:
2. Chuyển sang chỉnh sửa XML, và đảm bảo rằng bạn đã thiết lập các thuộc tính sau cho **TextView** mới:

TextView attribute	Value
<code>android:id</code>	<code>"@+id/phone_label"</code>

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"

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:



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.
6. Kiểm tra mã XML để biết bối cảnh bằng cách nhấp 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 bối cảnh marginleft để tương thích với cũ hơn phiên bản của Android):

EditText attribute	Value
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**: Tạo 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 showToast sau:
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 xuống **layout_width**, và **wrap_content** cho menu thả xuống **layout_height**.

Bố cục sẽ trông giống như hình dưới đây. Menu thả xuống 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—với 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 khối 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:

- Bố 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.

2. Menu và bộ chọn

Giới thiệu

Thanh ứng dụng (App Bar), còn được gọi là thanh hành động (Action Bar), là một không 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](#).

3. Điều hướng người dùng

Giới thiệu

Thanh ứng dụng (còn được gọi là Thanh hành động) là một khung 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
- thanhanh.
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.

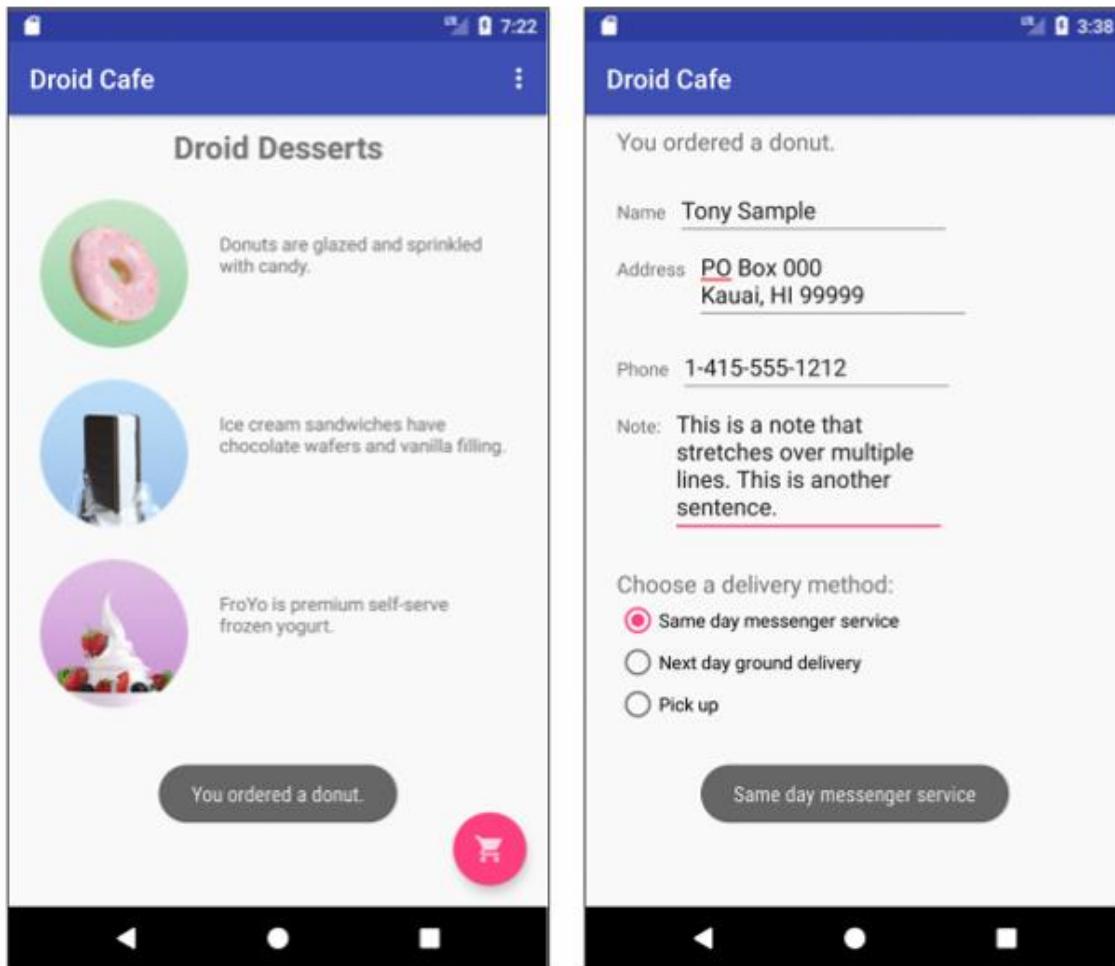
Những gì bạn sẽ làm

- 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.

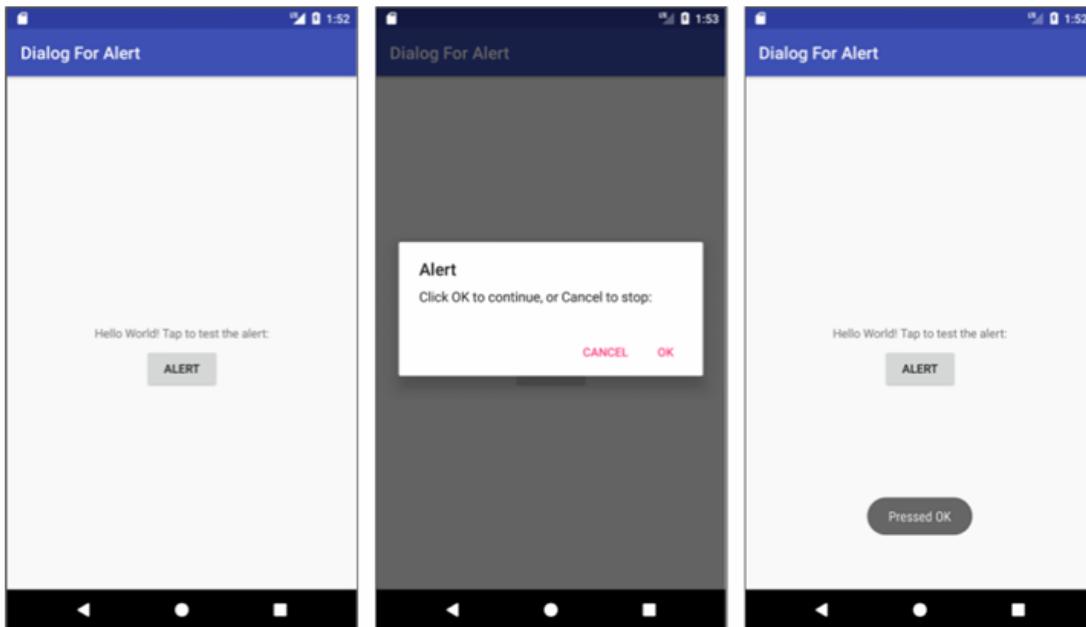
Tổng quan về ứng dụng

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.

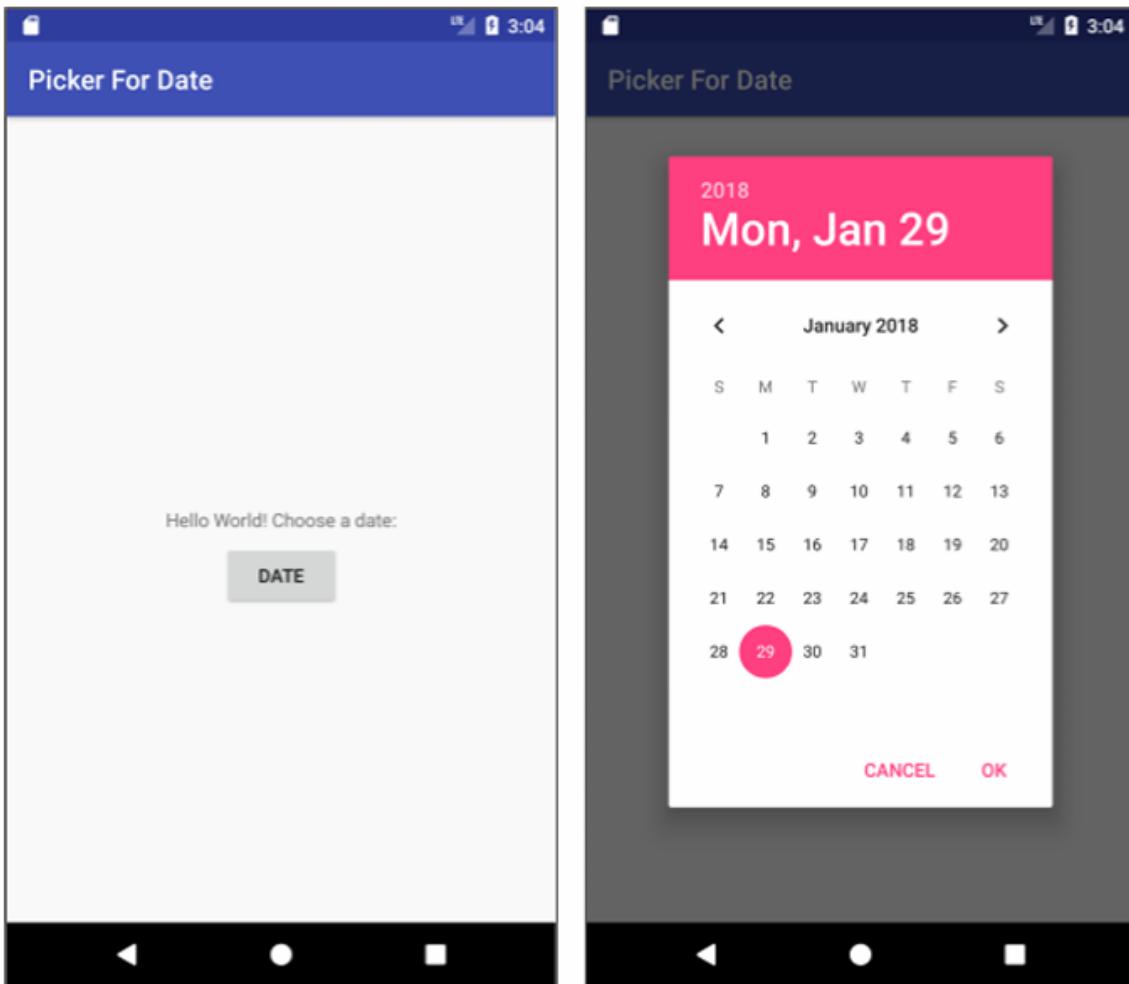


Đối với bài tập này, bạn đang sử dụng thanh công cụ Thư viện hỗ trợ V7 AppCompat làm thanh ứng dụng, hoạt động trên phạm vi rộng nhất của thiết bị và cũng cung cấp cho bạn chỗ để tùy chỉnh thanh ứng dụng của bạn sau này khi ứng dụng của bạn phát triển. Để đọc thêm về các cân nhắc thiết kế cho việc sử 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 droidcafefInput 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 droidcafefInput 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 cũng 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 windowTitle 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.

5. Nhìn vào MainActivity, giúp mở rộng ứng dụng và bắt đầu bằng onCreate () Phương thức, đặt chế độ xem nội dung thành bối cảnh Activity_main.xml và đặt thanh công cụ thành thanh công cụ được xác định trong bối cảnh. 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 tráng 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 (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="@string/action_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):

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 khối <sences> và cung cấp cho mục các thuộc tính sau:

Attribute	Value
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.

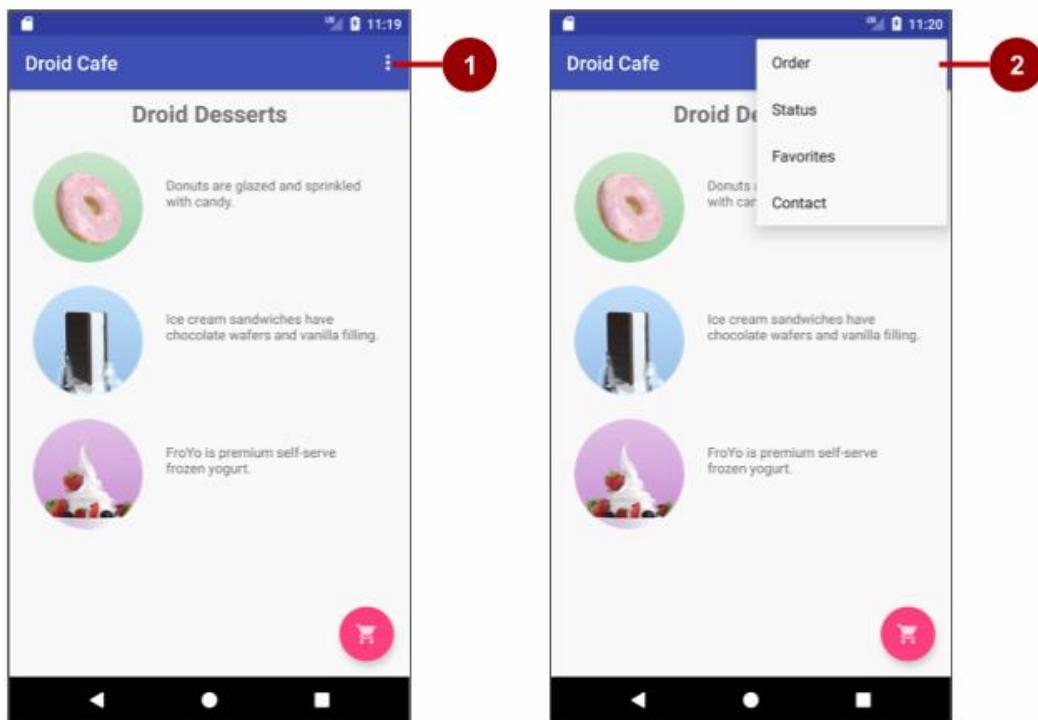
8. Bạn sẽ hiển thị thông báo bánh mì nướng với thông báo hành động tùy thuộc vào mục menu mà người dùng chọn. Mở chuỗi.xml và thêm các tên và giá trị chuỗi sau cho các thông báo sau:

```
<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 OnOptionItemSelected()

```
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 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: 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	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 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 chõ cho nó:

Order item attribute	Old value	New value
android:icon	<i>none</i>	"@drawable/ic_shopping_cart"
app:showAsAction	"never"	"always"

Status item attribute	Old value	New value
android:icon	<i>none</i>	"@drawable/@drawable/ic_status_info"
app:showAsAction	"never"	"always"

Favorites item attribute	Old value	New value
android:icon	<i>none</i>	"@drawable/ic_favorite"
app:showAsAction	"never"	"ifRoom"

3. 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.
4. 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

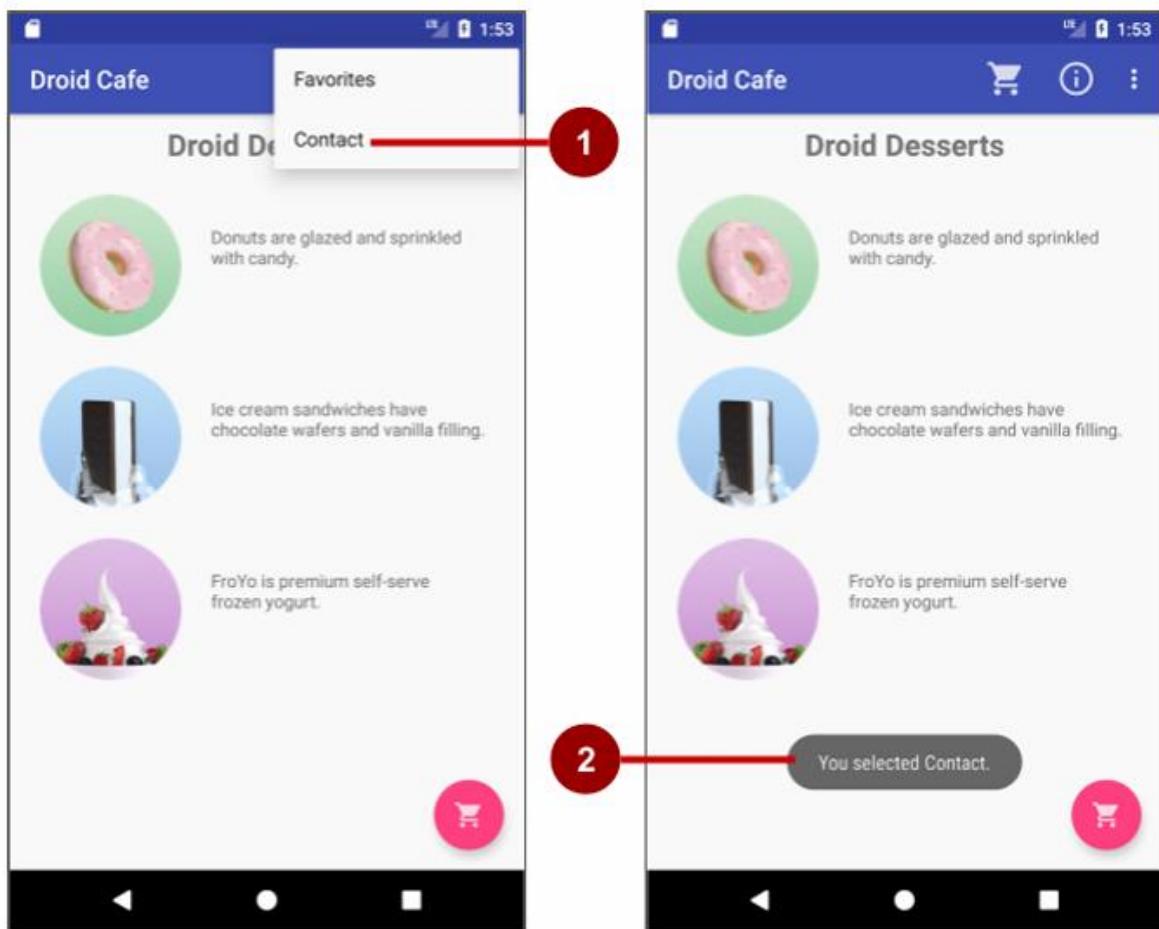
Xác định xem một mục menu nhất định có được nhấp không, bằng cách sử dụng ID mục menu. Trong ví dụ dưới đây, ID là **action_order**:

```
@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 id:

```
@Override  
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 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 bằng cách sử dụng hình ảnh có thể nhấp):

```
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  
}
```

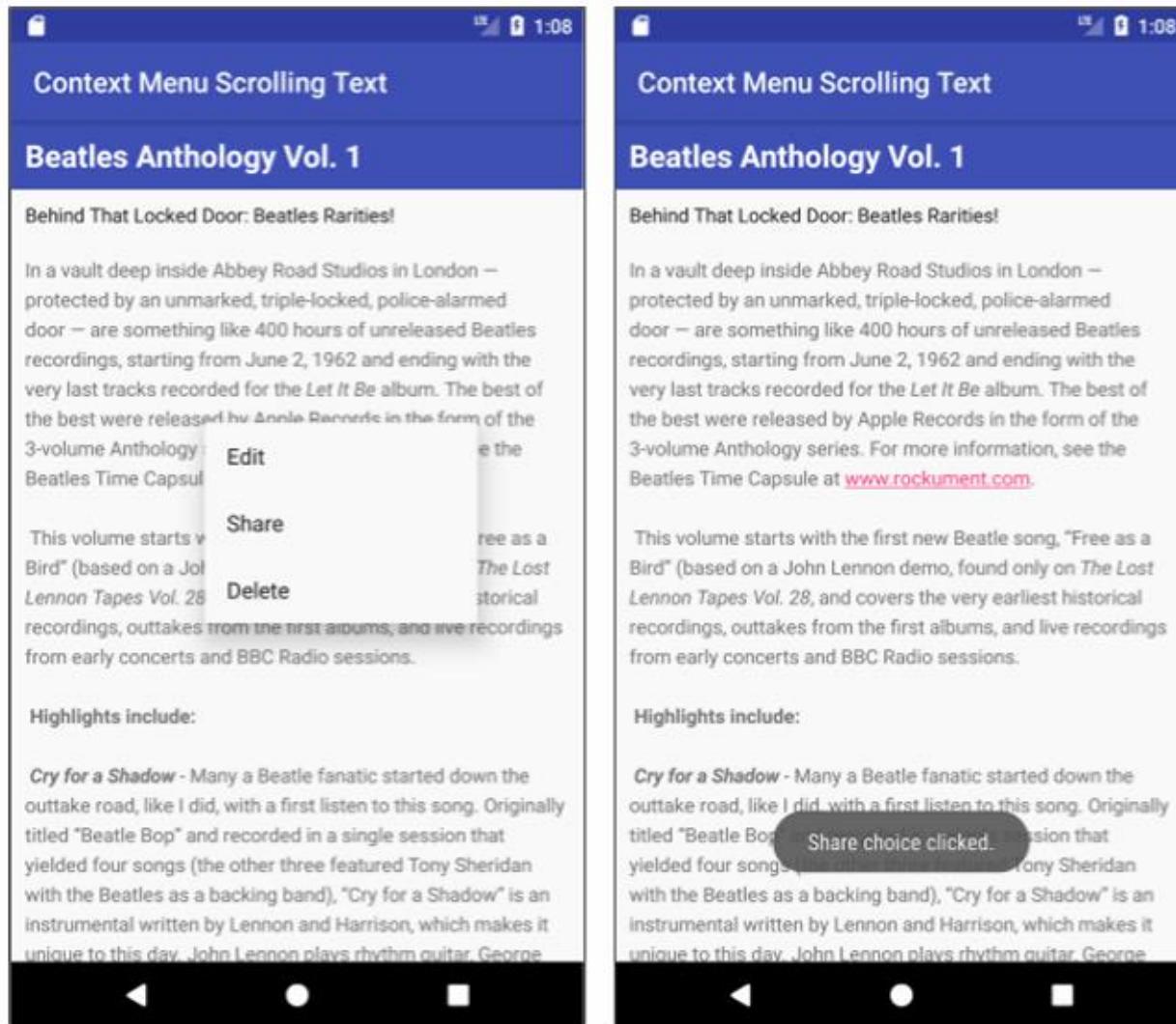
Chạy ứng dụng. Nhấp vào biểu tượng giỏ hàng trong thanh ứng dụng (mục **Order**) đưa bạn trực tiếp đến màn hình **OrderActivity**.

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 thoả phòng 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**:

```
TextView article_text = findViewById(R.id.article);
registerForContextMenu(article_text);
```

3. Thực hiện phương thức **OnCreateContextMenu ()** trong hoạt động để thổi phồng 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);  
    }  
}
```

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.

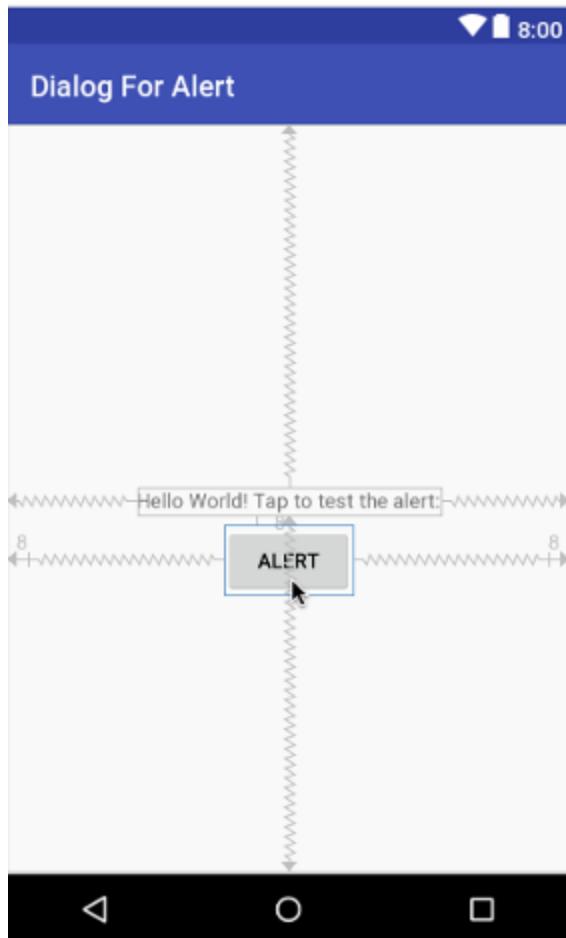
4.1 Tạo một ứng dụng mới để hiển thị hộp thoại cảnh báo

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 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 () Nhấp 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 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:

```
// 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

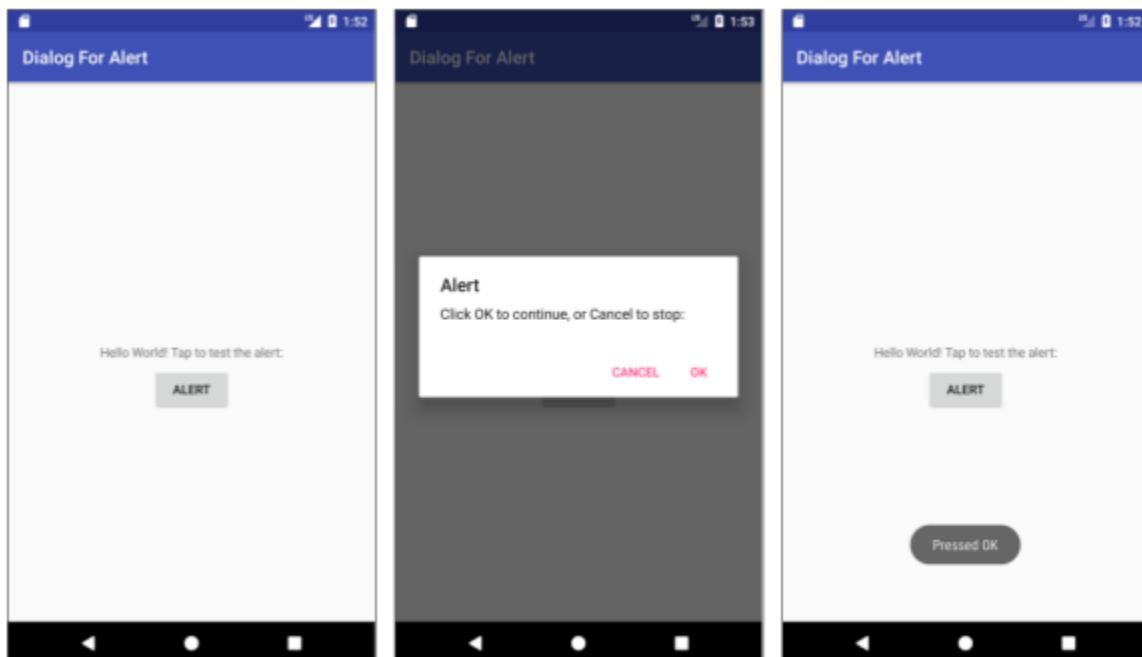
khi người dùng chạm vào nút **OK** hoặc **Cancel** trong cảnh báo, bạn có thể lấy lựa chọn của người dùng và sử dụng trong mã của mình. Trong ví dụ này, bạn hiển thị thông báo Toast.

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();
```

7. 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.

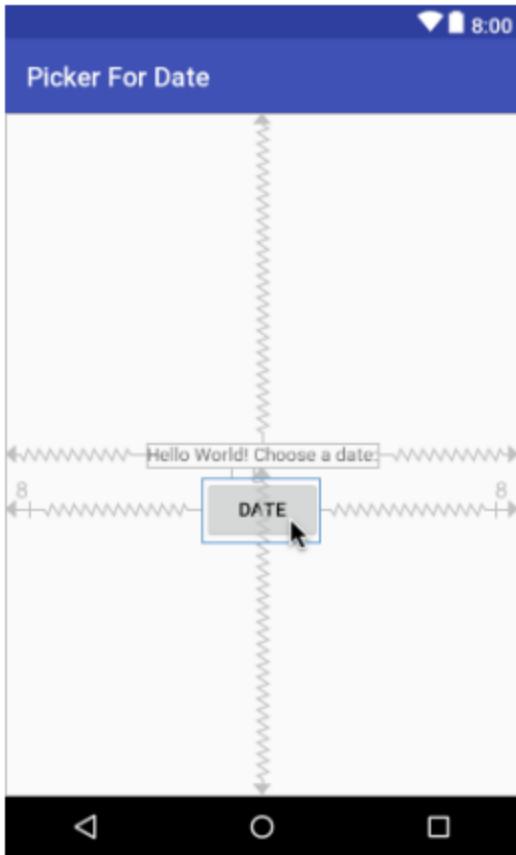
Cách tốt nhất để hiển thị trình chọn là sử dụng một thể hiện của **DialogFragment**, là một lớp con của **Fragment**.

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 nhấp **showDatePicker()**. Sau khi nhập, trình xử lý nhấp 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 bối cảnh, bao gồm các phương thức nhà máy đoạn mã hoặc bao gồm các lệnh gọi lại giao diện. Bạn không cần tạo bối cảnh 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 khối 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.

4. Nhấp 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)
```

5. Xóa hàm dựng public DatePikerFragment() trống.
6. 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) {
```

7. 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);
```

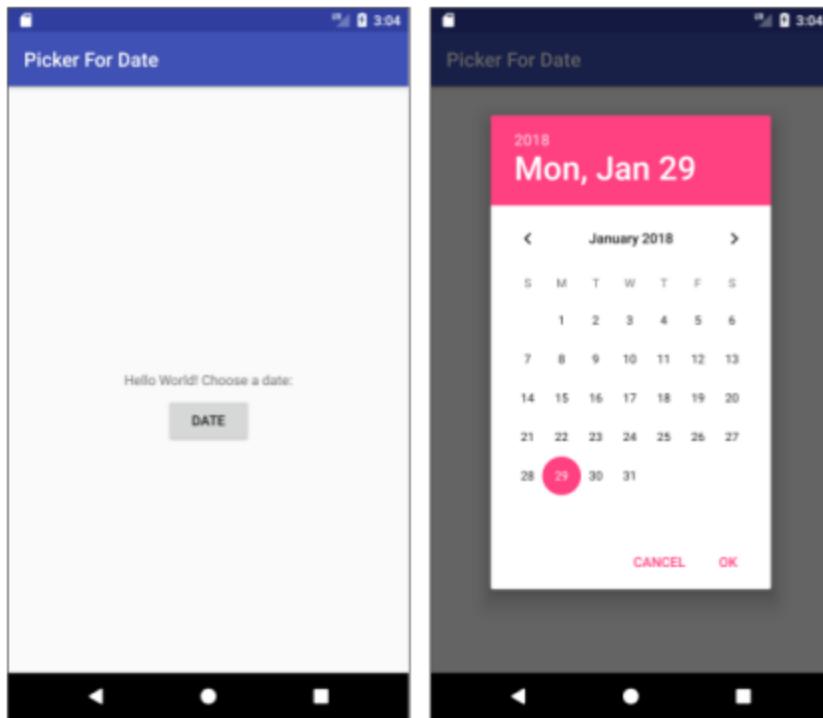
5.4 Sửa đổi hoạt động chính

Mặc dù phần lớn mã trong MainActivity.java vẫn giữ nguyên, bạn cần thêm một phương thức tạo một thể 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 thể 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 trở 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();
```

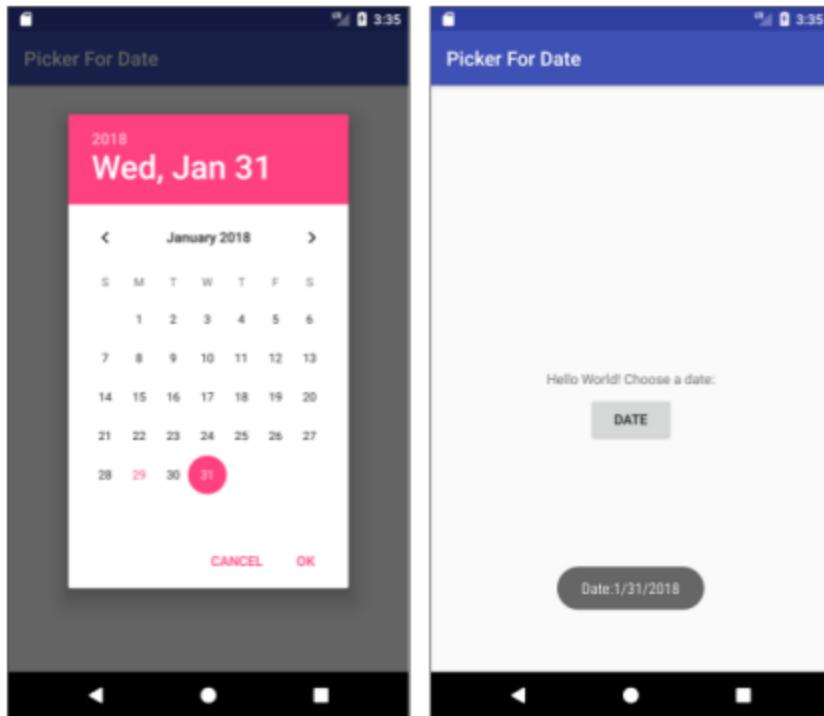
4. Trích xuất chuỗi được mã hóa cứng "Date: " thành một tài nguyên chuỗi có tên là date.
5. Mở DatePickerFragment và thêm nội dung sau vào phương thức onDateSet() để gọi processDatePickerResult() trong MainActivity và truyền cho nó năm, tháng và ngày:

```
@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 địnhhthay 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:

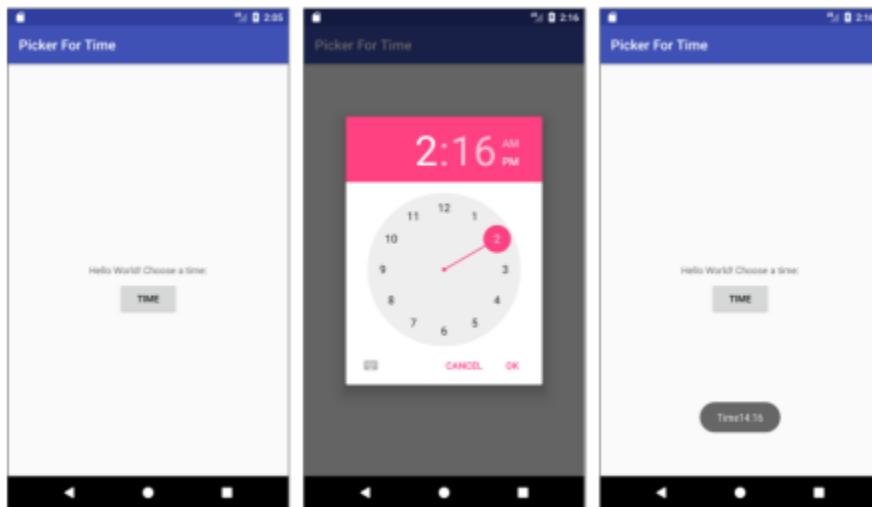
- Triển khai TimePickerDialog.OnTimeSetListener để tạo bộ chọn thời gian chuẩn với trình lắng nghe.
- Thay đổi các tham số của phương thức onTimeSet() từ int i thành int hourOfDay và int i1 thành int minute.
- Lấy giờ và phút hiện tại từ Calendar:

```
final Calendar c = Calendar.getInstance();
int hour = c.get(Calendar.HOUR_OF_DAY);
int minute = c.get(Calendar.MINUTE);
```

- Tạo 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 nổi.
- Mẫu thiết lập bố cục CoordinatorLayout với bố 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 con 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 onCreateView() 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
- MenuItemInflator
- 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:

- myAlertBuilder.setMessage("Alert");
- myAlertBuilder.setPositiveButton("Alert");
- myAlertBuilder.setTitle("Alert");
- AlertDialog.Builder myAlertBuilder = new AlertDialog.Builder("Alert");

Câu hỏi 4

Bạn tạo DialogFragment cho bộ chọn ngày ở đâu? Chọn một trong các mục sau:

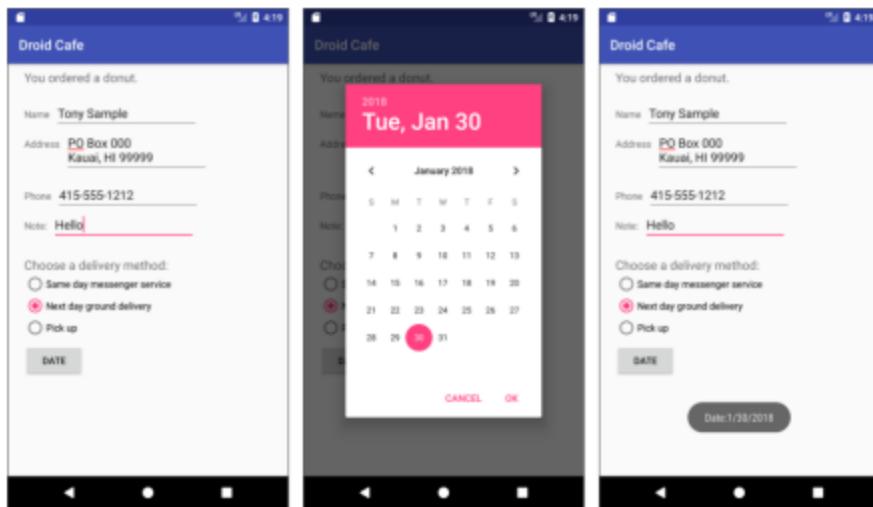
- Trong phương thức onCreate() trong Hoạt động lưu trữ.
- Trong phương thức onCreateContextMenu() trong Fragment.
- Trong phương thức onCreateView() trong phần mở rộng của DialogFragment.
- Trong phương thức onCreateDialog() trong phần mở rộng của DialogFragment.

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).
- Nhấp 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)



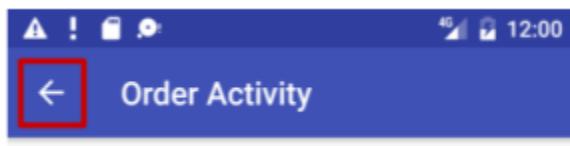
Bài 4.4: Điều hướng người dùng

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.
- Chỉnh 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.

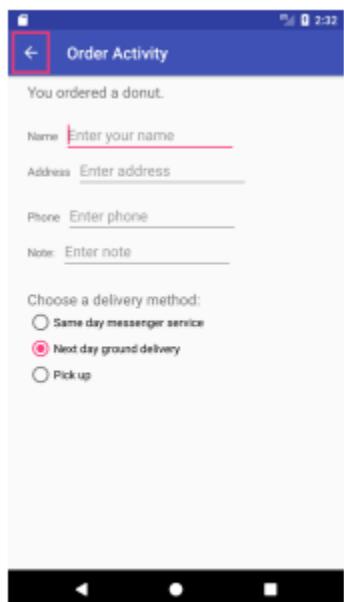
Bạn sẽ làm gì

- 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.

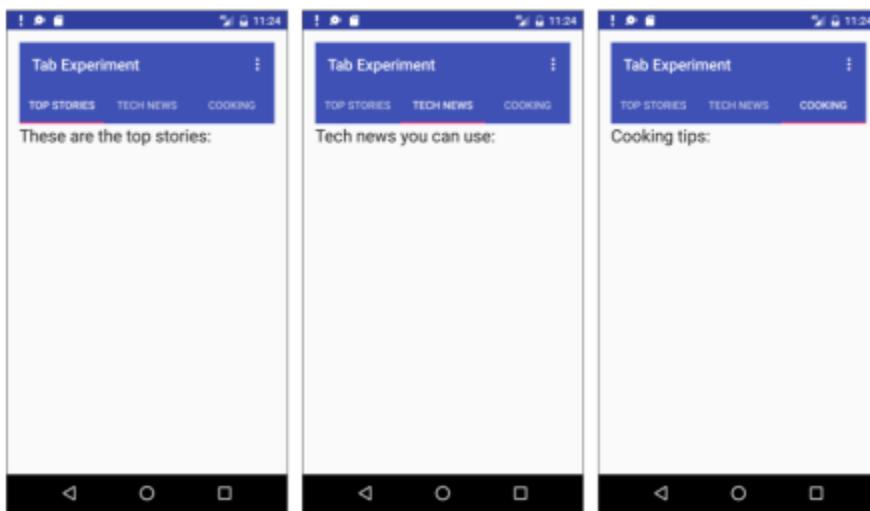
Để bắt đầu dự án từ nơi bạn đã dừng lại trong bài thực hành trước, hãy tải xuống dự án Android Studio DroidCafeOptions.



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.

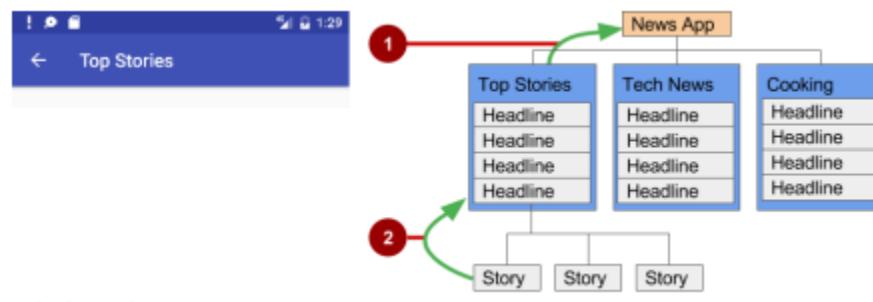


Nhiệm vụ 1: Thêm nút Lên để điều hướng tổ tiên

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" tổ tiên, cho phép người dùng di 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

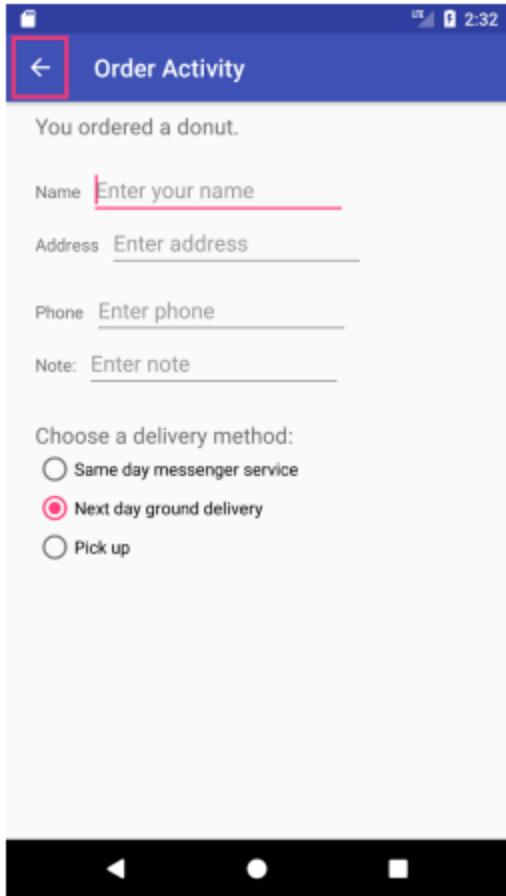
1.

```
<activity android:name="com.example.android.droidcafeinput.OrderActivity">
    <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.



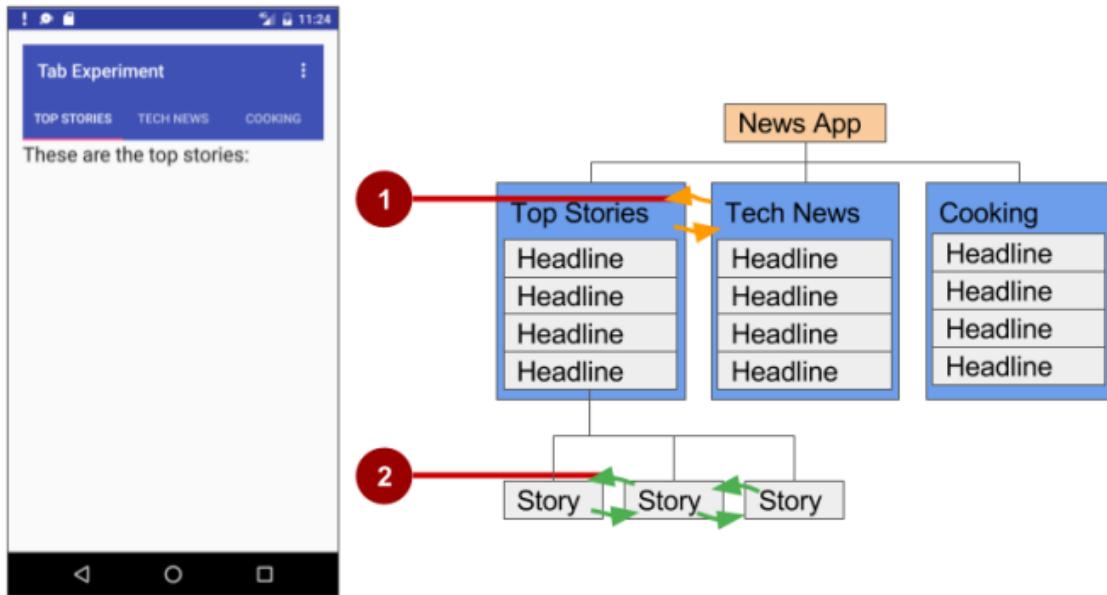
Task 1: Mã nguồn giải pháp

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 điền 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 điền 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).

4. Các adapter tiêu chuẩn để sử dụng Fragment với ViewPager:

- **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**

2.1 Tạo dự án và bố 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 bố 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.

5. 2.2 Tạo một lớp và bối cảnh cho từng fragment

Để thêm một **fragment** đại diện cho mỗi màn hình tab, hãy làm theo các bước sau:

1. Nhấp 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**) bối cảnh 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 Chỉnh sửa bối cảnh của Fragment

Chỉnh sửa từng tệp bối 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**.

6. 2.3 Thêm một PagerAdapter

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 (**mNumOfTabs**):

1. Nhấp 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. Bây giờ, 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 thể 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.
```

7. 2.5 Sử dụng PagerAdapter để quản lý các màn hình

1. 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.
```

2. 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) {  
}  
});
```

3. 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.

8. 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.)

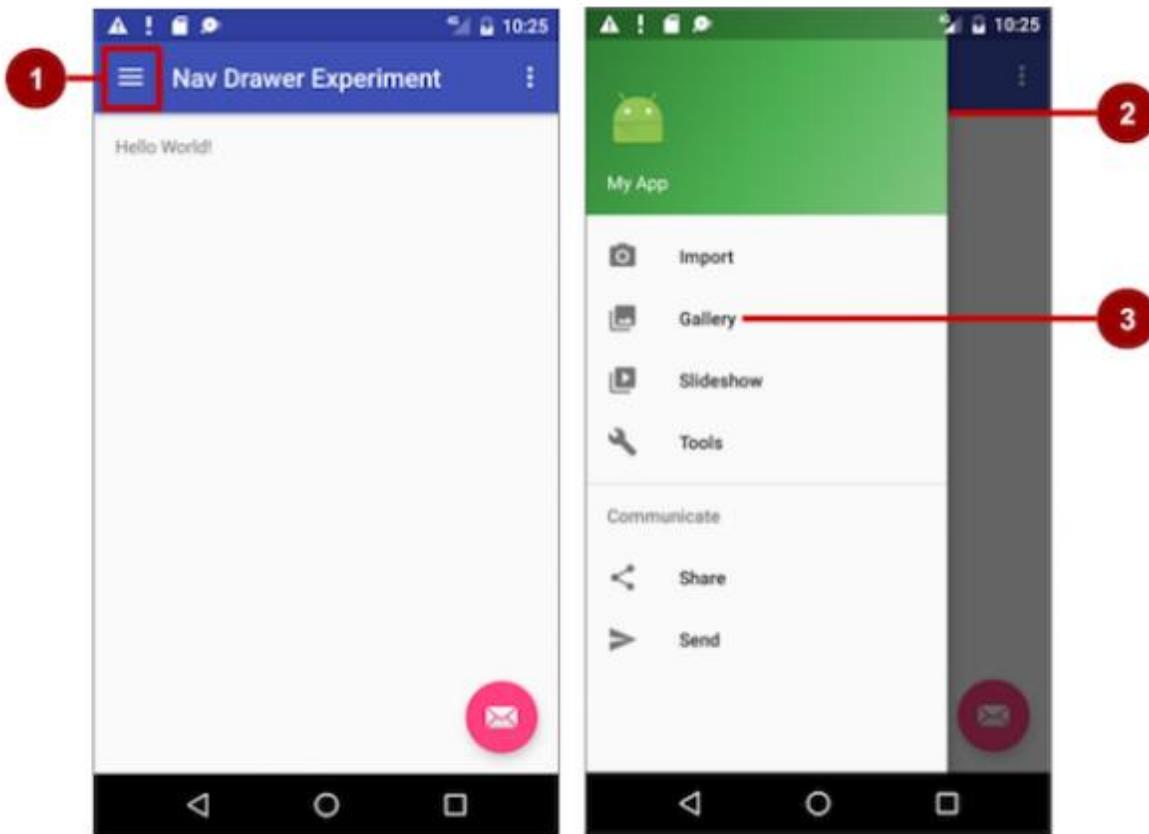
Thử thách:

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**.



9. Trong hình minh họa trên:

1. Biểu tượng điều hướng trong app bar.
2. Navigation drawer.
3. Mục menu trong navigation drawer.

10. Các bước để tạo navigation drawer trong ứng dụng:

1. 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.

2. 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.

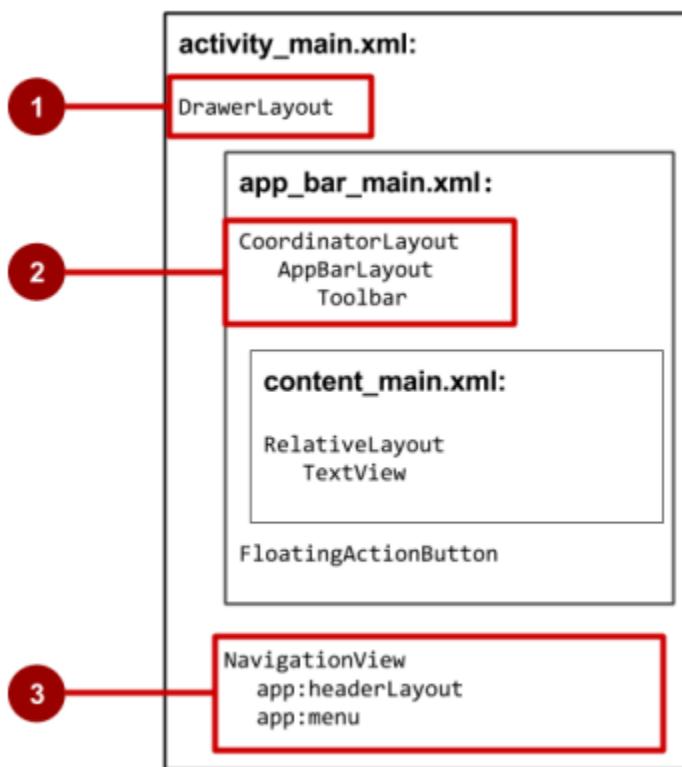
- **Thêm listener** để xử lý sự kiện khi người dùng chọn một mục trong **drawer**.
- **Xử lý lựa chọn mục trong menu điều hướng**

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**.

11. 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 bố 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 bố cục con.



12. 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.

13. 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.

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

14. 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**

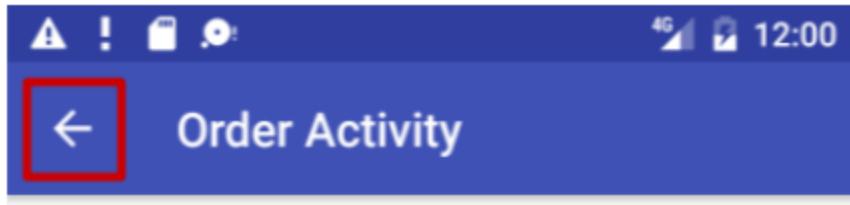
15. Bài tập về nhà

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.

1. 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**.
2. **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.
3. **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**.



16. 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.

17. 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**,...

18. 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.

19. 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.

20. Bài 4.5: RecyclerView

21. 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.

22. 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**.

23. 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.

24. Những gì bạn sẽ làm:

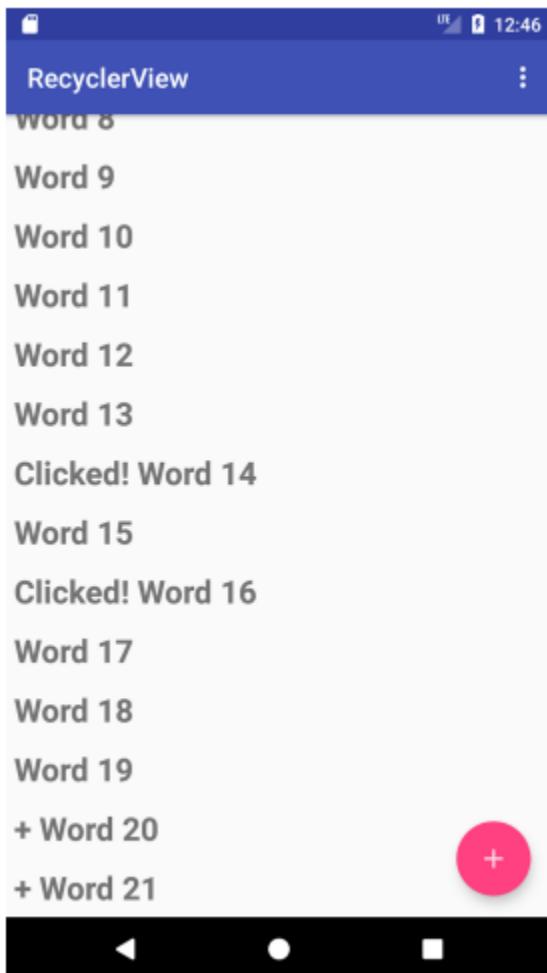
- ✓ **Tạo một ứng dụng mới** sử dụng **RecyclerView** để hiển thị danh sách mục có thể cuộn và gán hành vi khi người dùng nhấp vào các mục trong danh sách.
- ✓ **Sử dụng FAB** để cho phép người dùng thêm mục mới vào **RecyclerView**.

25. 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.



26. 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**.

27. Giải thích:

Trong các ứng dụng phức tạp hơn, dữ liệu có thể đến từ:

- ❖ **Bộ nhớ trong** (tệp, cơ sở dữ liệu SQLite, Shared Preferences).
- ❖ **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**.

28. 1.1. Tạo dự án và bố 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 nhấp) 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**.

29. 1.2. 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", ...].

30. 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.

31. 1.3. Thay đổi biểu tượng của FAB

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.

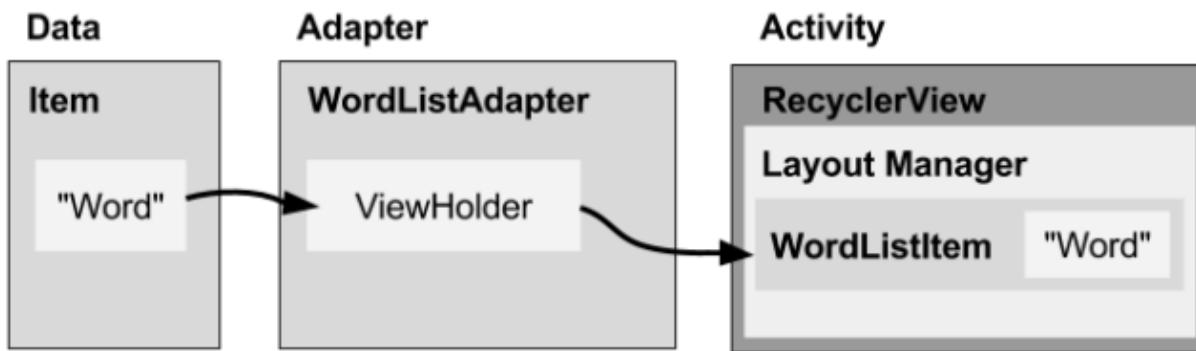
32. Task 2: Tạo một RecyclerView

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 lưới**.
 - 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.



33. 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.

34. 2.1. Chỉnh sửa bố cục trong content_main.xml

1. Mở **content_main.xml** trong ứng dụng **RecyclerView**. Tập này chứa một **TextView** hiển thị "**Hello World**" ở trung tâm của **ConstraintLayout**.
2. Nhấp 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**)

35. 2.2. 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
<code>android:layout_width</code>	<code>"match_parent"</code>
<code>android:layout_height</code>	<code>"wrap_content"</code>
<code>android:orientation</code>	<code>"vertical"</code>
<code>android:padding</code>	<code>"6dp"</code>

36. 5: 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"

37. 2.3. 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.

1. 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>
```

38. 2.4. 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.

1. 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 **bà 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.

39. 2.5 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 lồng trong) để lưu **TextView** và **adapter**.

```
public final TextView wordItemView;  
final WordListAdapter mAdapter;
```

3: Trong lớp lồng **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 trữ dữ liệu trong **adapter**, và **WordListAdapter** cần một constructor để khởi tạo danh sách từ dữ liệu. Thực hiện các bước sau:

- Để lưu trữ 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;
```

- Bây giờ, bạn có thể điền vào phương thức **getItemCount()** để trả về kích thước của **mWordList**:

```
@Override  
public int getItemCount() {  
    return mWordList.size();  
}
```

- WordListAdapter** cần một constructor để khởi tạo danh sách từ dữ liệu.

- Để tạo **View** cho một mục trong danh sách, **WordListAdapter** cần **inflate** tệp XML của danh sách.
- Bạn sử dụng **LayoutInflater** để đọc và chuyển đổi tệp XML thành các **View** tương ứng.
- 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.

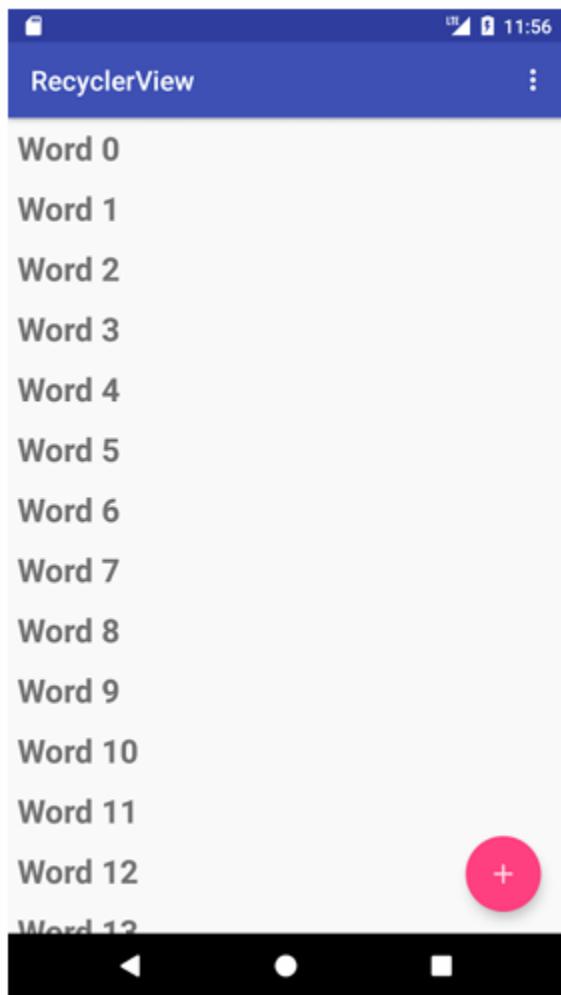
1. Mở MainActivity .
2. Thêm các biến thành viên cho RecyclerView và bộ điều hợp.

```
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. Nhấp 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.

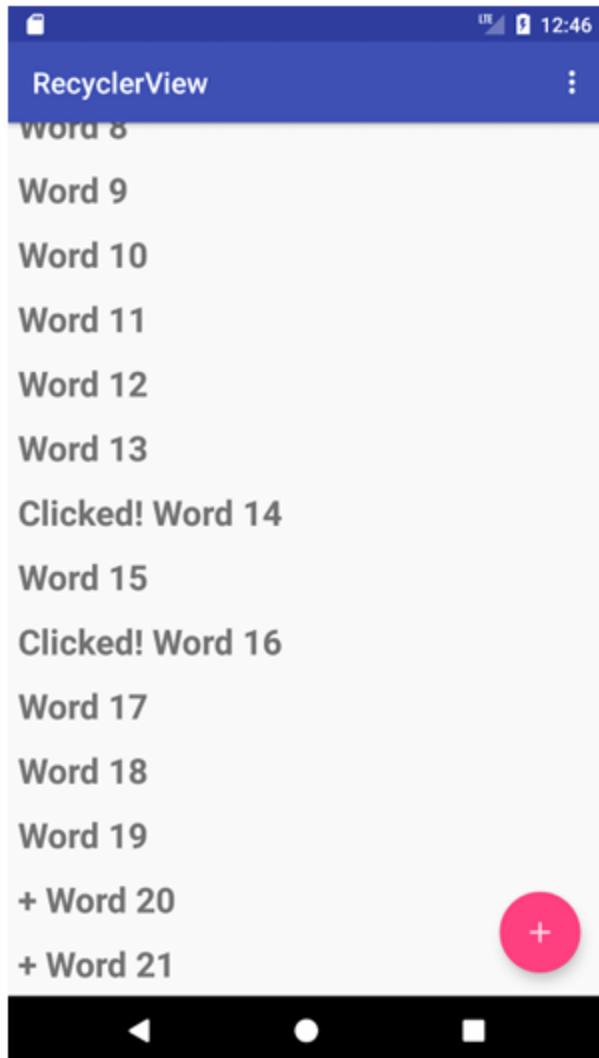
- Cuộn đến mục đã chèn.

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 nhấp 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 thổi phồng tài nguyên bố cục XML cho mục danh sách bằng cách sử dụng LayoutInflater.
- LinearLayoutManager là một trình quản lý bố cục RecyclerView hiển thị các mục trong danh sách cuộn dọc hoặc ngang.
- GridLayoutManager là một trình quản lý bố cục RecyclerView hiển thị các mục trong lưới
- StaggeredGridLayoutManager là trình quản lý bố 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
 - Bố cục bơm hơi
 - RecyclerView.LayoutManager
 - Trình quản lý bố cục tuyến tính
 - Trình quản lý bố cục lưới
 - Trình quản lý StaggeredGridLayoutManager
 - Bố cục điều phối viên
 - Bố cục ràng buộc
 - RecyclerView.Adapter
 - Người tái chếView.ViewHolder
 - View.OnClickListener
 - Tạo danh sách với RecyclerView

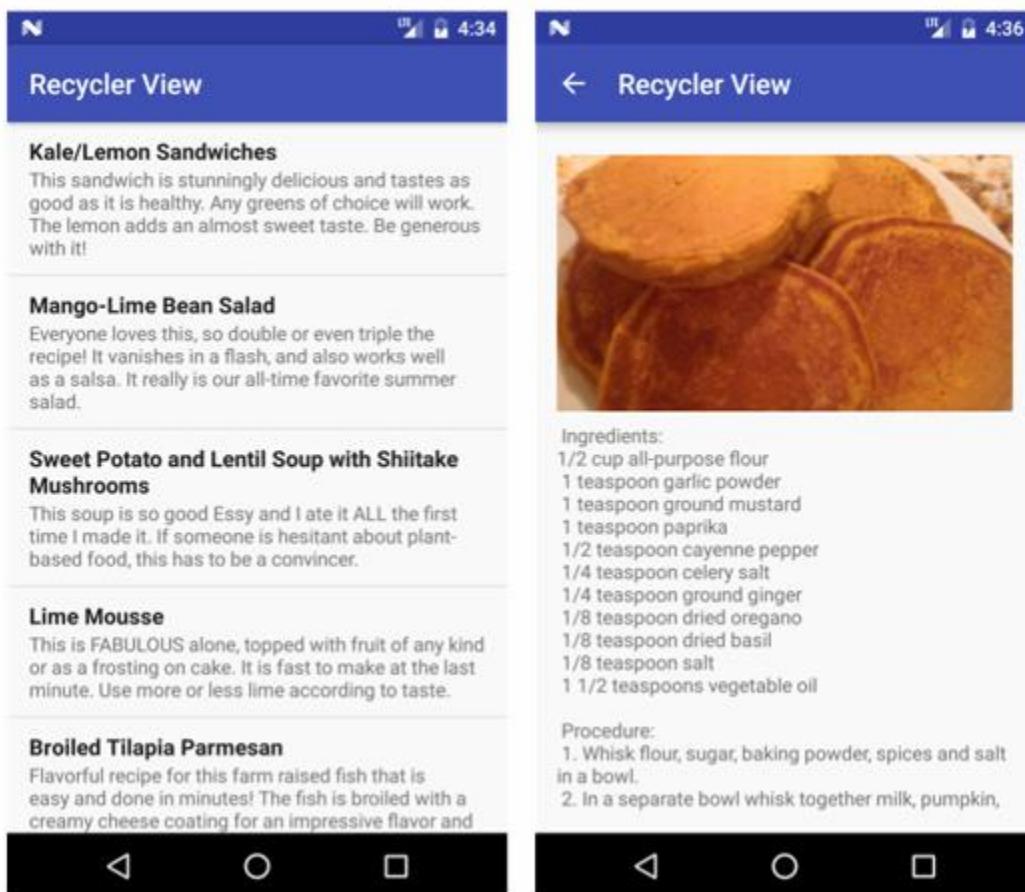
Homework

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 bố cục cho một mục trong danh sách.
- Tất cả các mục danh sách trông giống nhau.
- Bạn không cần trình quản lý bố cục với RecyclerView để xử lý hệ thống phân cấp và bố 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.
- Tập 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 bối 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 bối cục.
- Chỉnh sửa mã bối 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.

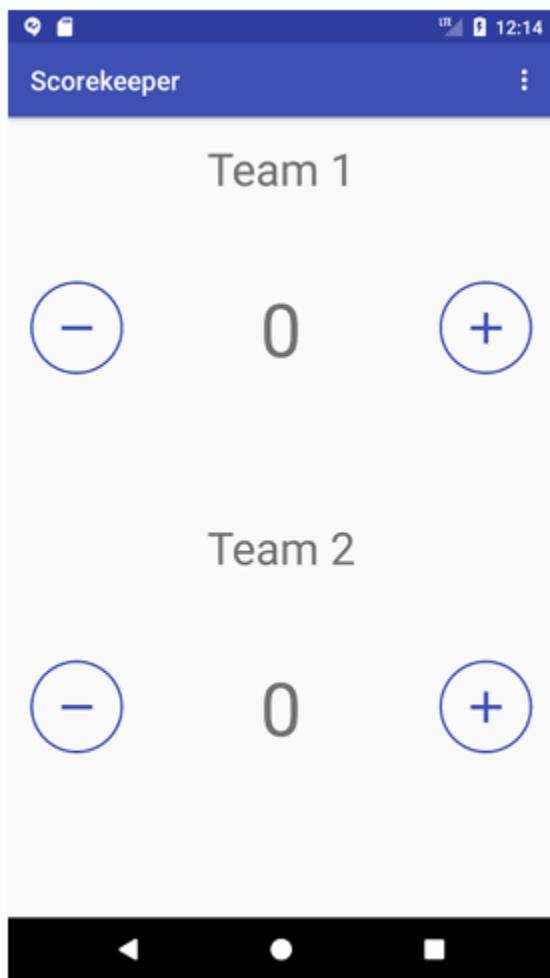
Bạn sẽ làm gì

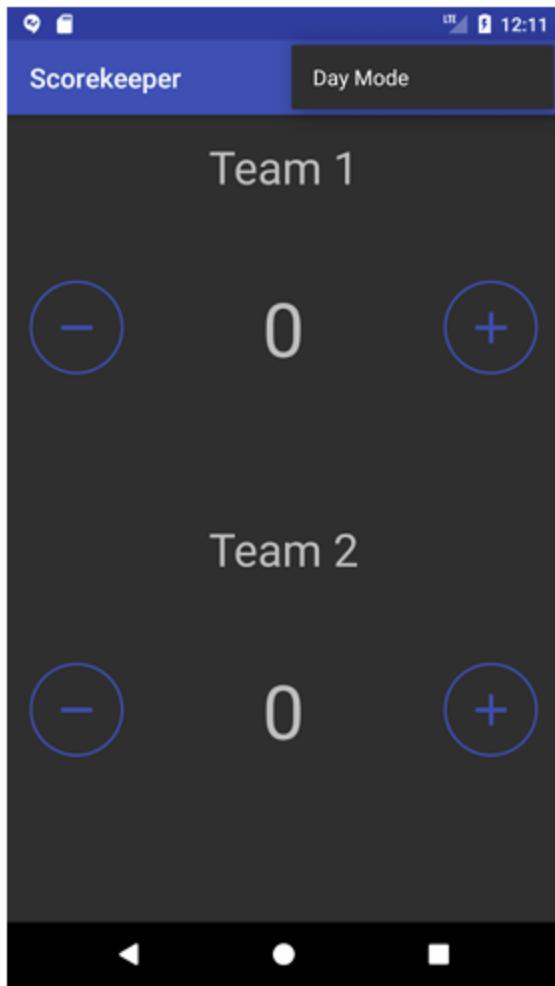
- Tạo một ứng dụng mới và thêm các phần tử Button và TextView vào bối cục.
- Tạo tài nguyên Drawable trong XML và sử dụng chúng làm nền cho các phần tử Button của bạn.

- 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 tử 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 bố 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 bố cục và Khả năng tương thích ngược (AppCompat) được chọn.
4. Nhấn vào Kết thúc.

1.2 Tạo bố cục cho MainActivity

Bước đầu tiên là thay đổi bố cục từ ConstraintLayout thành LinearLayout :

1. Mở activity_main.xml và nhấn 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"
```

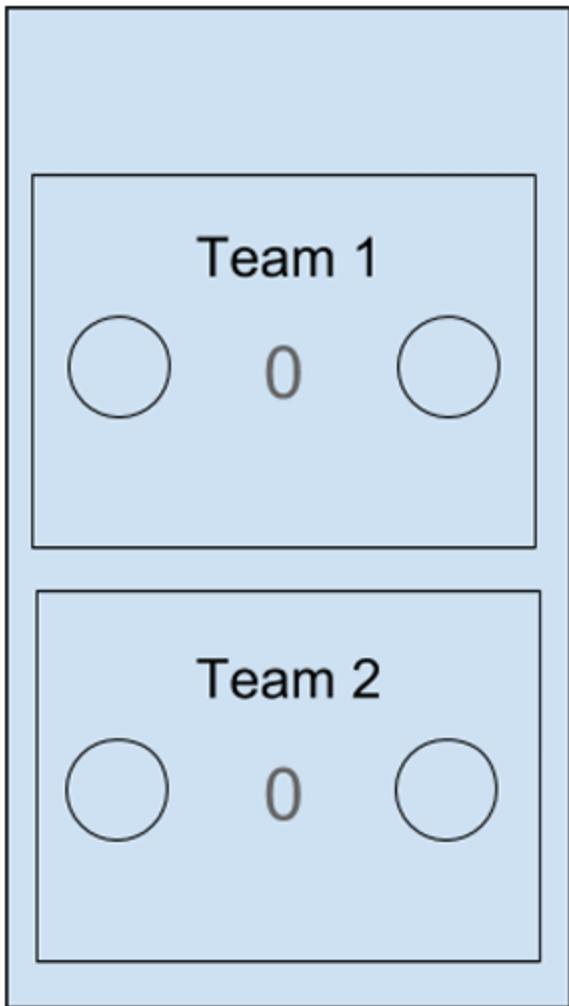
```
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        tools:context="com.example.android.scorekeeper.MainActivity">
```

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:

RelativeLayout attribute	Value
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:

ImageButton attribute	Value
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:

TextView attribute	Value
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:

TextView attribute	Value
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 khởi 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 bố cục

Sau đây cho thấy bố cục cho ứng dụng Scorekeeper và mã XML cho bố cục.

Lưu ý: Mã của bạn có thể hơi khác một chút, vì có nhiều cách để đạt được cùng một bố 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 đếm 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 đi 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 tiện

```
    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, bố 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 diện Trên **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**.

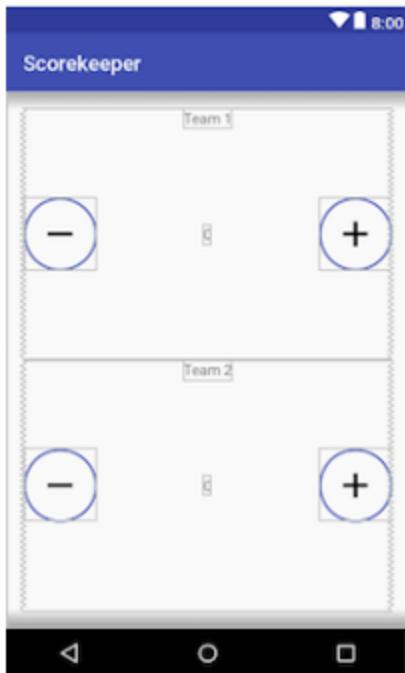
6. Nhập button_size cho Tên tài nguyên .

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:

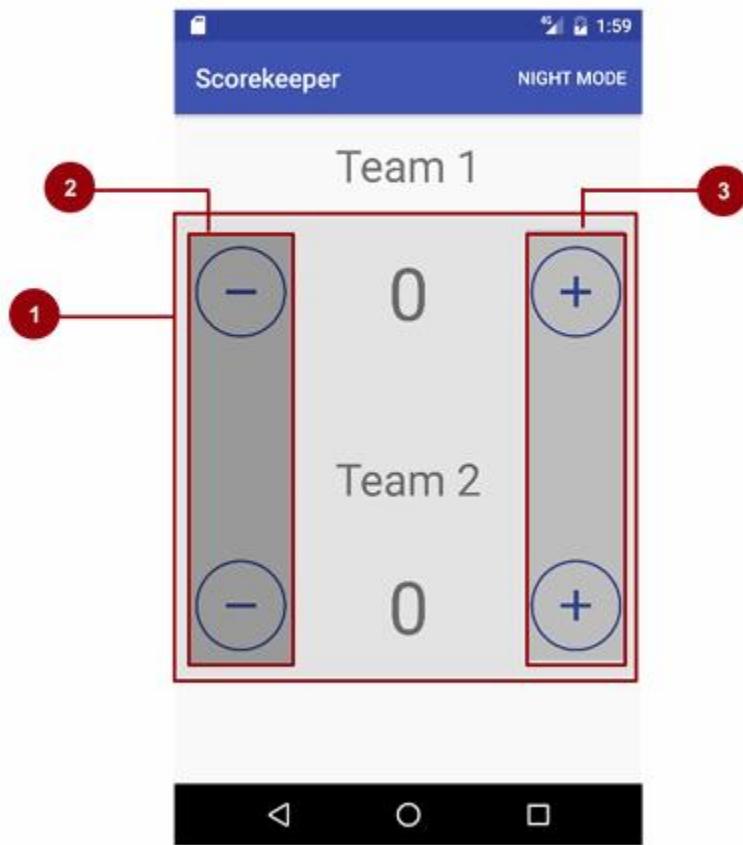
- Bất kỳ thuộc tính kiểu nào được xác định bởi kiểu mẹ sẽ tự động được đưa vào kiểu con.
- Thuộc tính kiểu được xác định trong cả kiểu mẹ và kiểu con sử dụng định nghĩa của kiểu con (thuộc tính con ghi đè kiểu mẹ).
- Kiểu con có thể bao gồm các thuộc tính bổ sung mà cha mẹ không xác định.

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 trừ. 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 trừ. Kiểu này kế thừa các thuộc tính của kiểu

ImageButton và bao gồm biểu tượng trừ.

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. Jetzt können Sie diese Attribute verwenden, um die entsprechenden Stile für die `ImageButton`-Felder zu ersetzen. Öffnen Sie den Datei-Ordner `activity_main.xml` im `MainActivity` und ersetzen Sie die folgenden Attributwerte durch die entsprechenden Stile:

Delete these attributes	Add this attribute
<code>android:src</code>	<code>style="@style/MinusButtons"</code>
<code>android:contentDescription</code>	
<code>android:background</code>	

Hinweis: Das Attribut `style` darf nicht mit dem Attributnamen `android:` verwendet werden, da `style` eine festgelegte XML-Attributrolle ist.

2. Ändern Sie die folgenden Attributwerte für beide `ImageButton`-Felder:

Delete these attributes	Add this attribute
<code>android:src</code>	<code>style="@style/PlusButtons"</code>
<code>android:contentDescription</code>	
<code>android:background</code>	

3.2 Erstellen eines `TextView`-Stils

Die meisten `TextView`-Felder zeigen den gleichen Text und haben die gleiche Schriftart. Es kann daher ein Stil erstellt werden, der für alle `TextView`-Felder gilt.

1. Fügen Sie das folgende Attribut hinzu, um alle `TextView`-Felder mit dem gleichen Stil zu versehen:

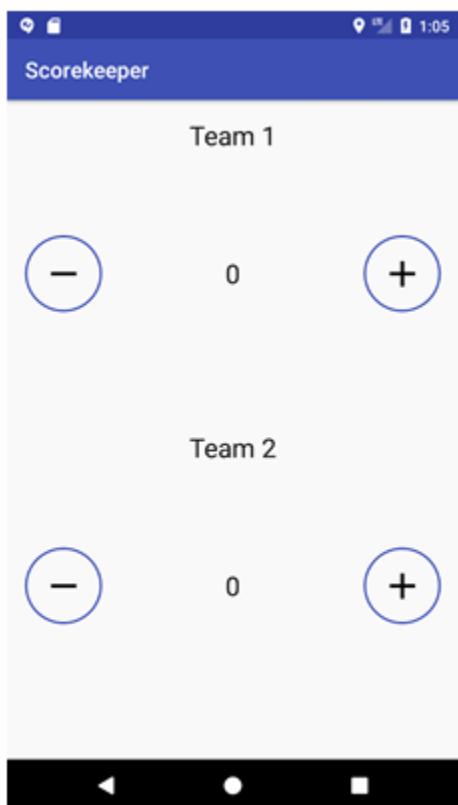
```
    android:textAppearance="@style/TextAppearance.AppCompat.Headline"
```

2. Klicken Sie mit der rechten Maustaste auf eines der `TextView`-Felder und wählen Sie `Refactor > Extract > Style...`.

3. Geben Sie dem Stil einen Namen (z.B. `ScoreText`) und wählen Sie das Attribut `textAppearance` aus der Liste aus. Klicken Sie auf `OK`, um den Stil zu speichern.

4. Wählen Sie `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.

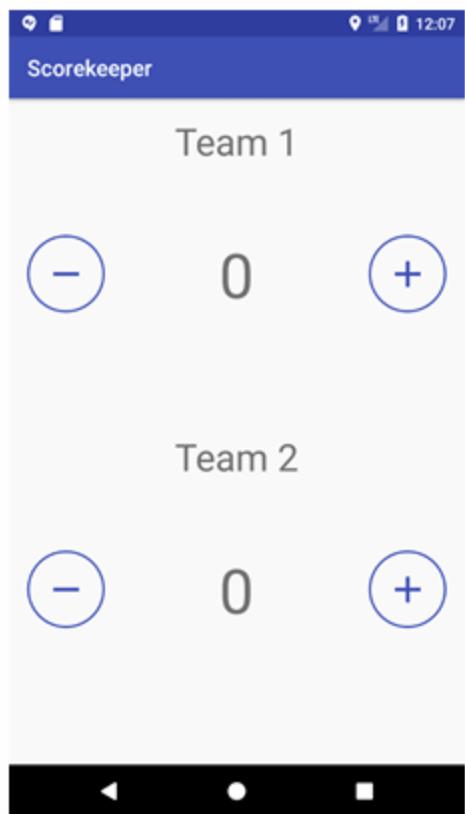
1. Tạo một kiểu mới có tên là TeamText với thuộc tính textAppearance được đặt như sau:

```
<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 bố cục xuất hiện cho tệp main_menu.xml.

4. Nhấp vào tab Văn bản của trình chỉnh sửa bố 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 thổi phồng menu:

```
getMenuInflater().inflate(R.menu.main_menu, menu);
```

40.4.3 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.

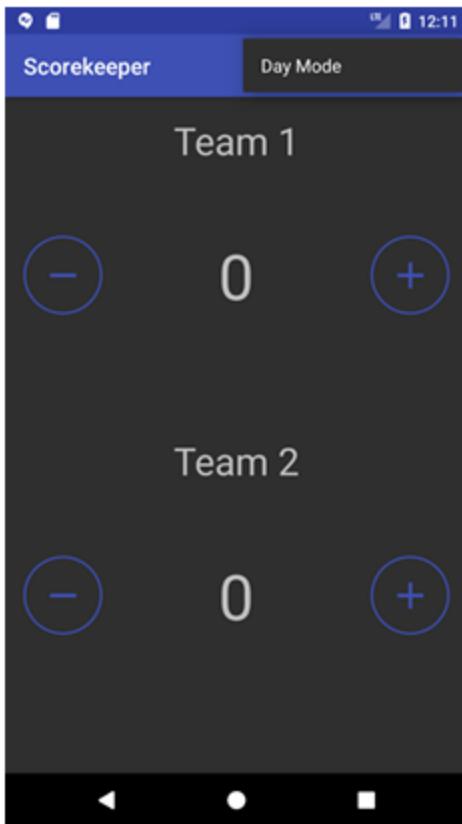
1. Thay thế câu lệnh `return super.onCreateOptionsMenu(menu)` trong phương thức `onCreateOptionsMenu()` bằng mã sau:

```

// 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;

```

1. Chạy ứng dụng của bạn. Nhãn 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:

1. 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";
```

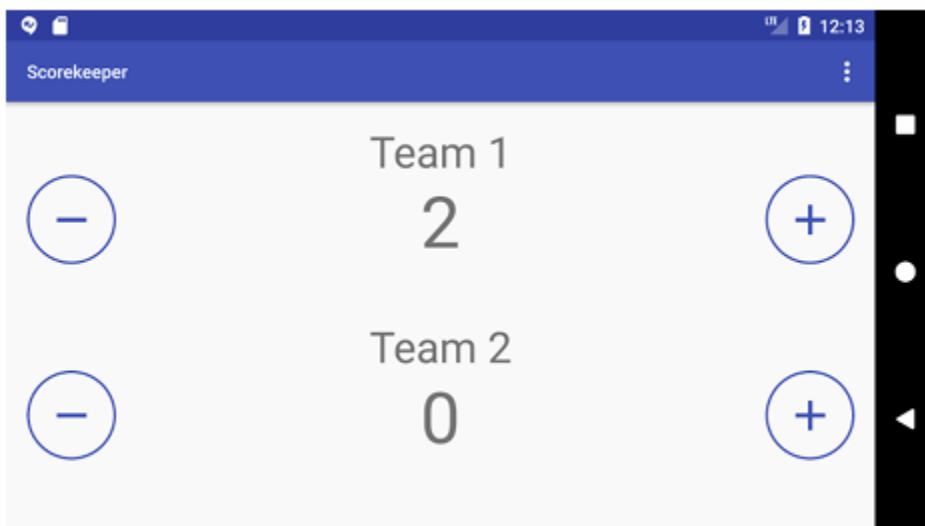
2. Ở cuối MainActivity , ghi đè phương thức onSaveInstanceState() để giữ nguyên giá trị của hai phần tử TextView điểm:

```
@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
- Bố cục tuyến tính
- Tài nguyên có thể vẽ
- Phong cách và chủ đề
- Các nút
- Bố 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

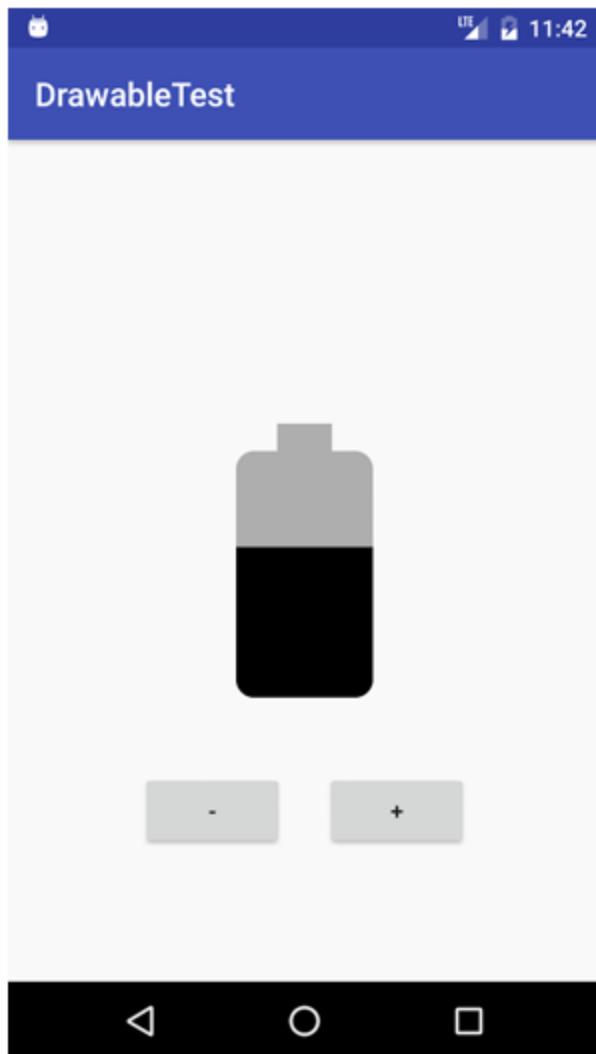
Xây dựng và chạy ứng dụng

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:

- Chủ 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.
- Chỉnh 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.

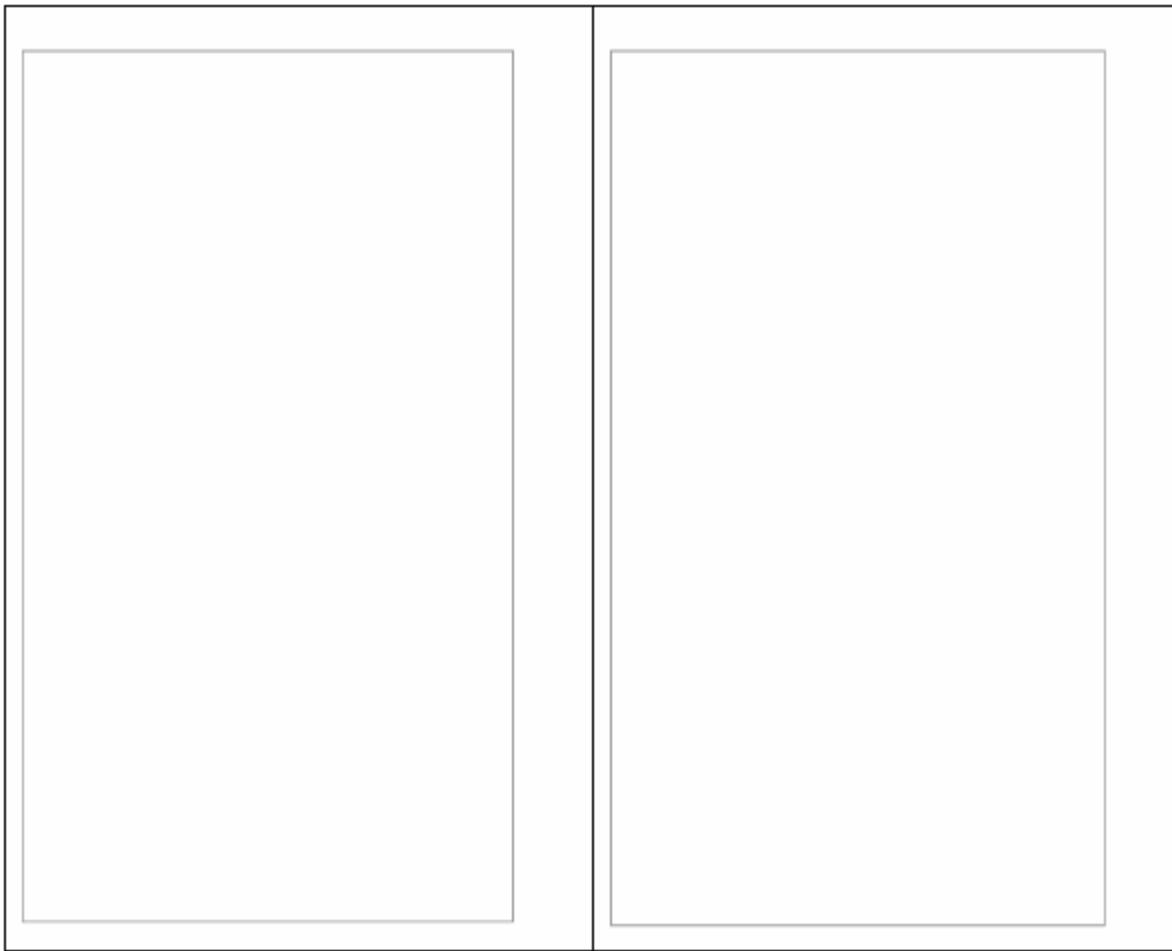
Bạn sẽ làm gì

- 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. Bố 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 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 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 Thể 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 lập 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 **getResourcel()**.

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 **getResourcel()** 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

1. Thay đổi LinearLayout bên trong tệp list_item.xml thành RelativeLayout và xóa thuộc tính android:orientation.

2. 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: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, bất 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;
```

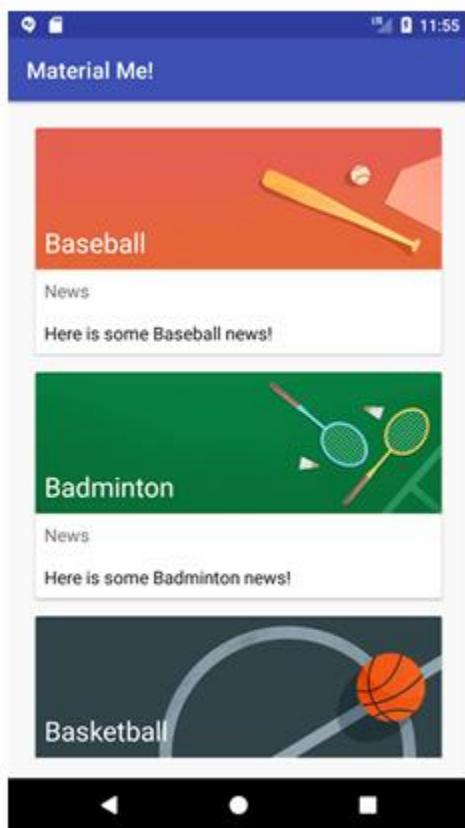
3. Khởi tạo biến trong hàm khởi tạo ViewHolder cho lớp ViewHolder:

```
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 đề 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.

```
ItemTouchHelper helper = new ItemTouchHelper(new
                                         ItemTouchHelper.SimpleCallback() {
    @Override
    public boolean onMove(RecyclerView recyclerView,
                         RecyclerView.ViewHolder viewHolder,
```



```
                         RecyclerView.ViewHolder target) {
        return false;
    }

    @Override
    public void onSwiped(RecyclerView.ViewHolder viewHolder,
                        int direction) {

    }
});
```

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. Nay 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:

1. 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) {
```

2. 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.

41. Tạo một **Activity** mới bằng cách vào **File > New > Activity > Empty Activity**.
42. Đặt tên **DetailActivity** và chấp nhận tất cả các thiết lập mặc định.
43. Mở tệp **activity_detail.xml** vừa tạo và thay đổi **ViewGroup** gốc thành **RelativeLayout**, như trong các bài tập trước.
44. Xóa dòng khai báo `xmlns:app="http://schemas.android.com/apk/res-auto"` khỏi **RelativeLayout**.
45. Sao chép tất cả các phần tử **TextView** và **ImageView** từ tệp **list_item.xml** sang tệp **activity_detail.xml**.
46. 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`.
47. 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**".

48. 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**.
49. Thay đổi **padding** của các phần tử **TextView** thành **16dp**.
50. 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**.
51. 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 của 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:

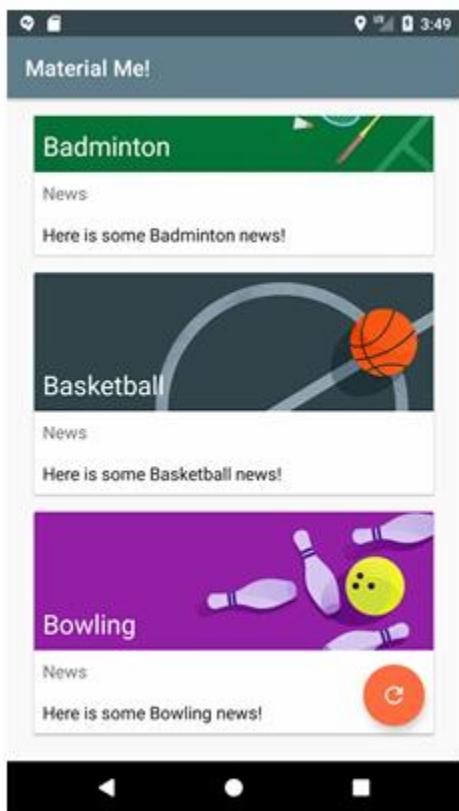
- **Màu cơ bản.** Thanh này được sử dụng tự động để tô màu thanh ứng dụng của bạn (thanh chứa tiêu đề ứng dụng của bạn).
- **Màu tối cơ bản.** Một bóng tối hơn cùng màu. Điều này được sử dụng cho thanh trạng thái phía trên thanh ứng dụng, trong số những thứ khác.
- **Một màu nhấn.** Một màu tương phản 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 nhấn 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

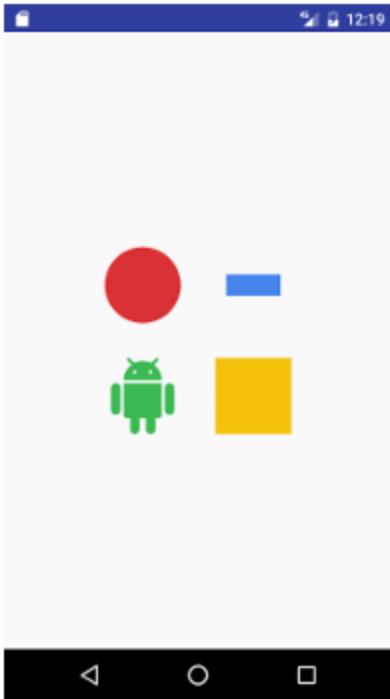
Thử thách này bao gồm hai cải tiến nhỏ đối với ứng dụng MaterialMe:

- Thêm chi tiết thực vào đổi 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 bối cảnh 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
- Bố 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 bố cục với thuộc tính android:transitionName.
- Mã sử dụng phương thức ActivityOptions.makeSceneTransitionAnimation().

Bài 5.3: Giới thiệu bố 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 bố cục, bạn đã học cách tạo các biến thể bố cục cho hướng ngang và máy tính bảng. Trong thực tế này, bạn sử dụng bố cục thích ứng, là bố 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 bố cục.
- Chỉnh 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.
- 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

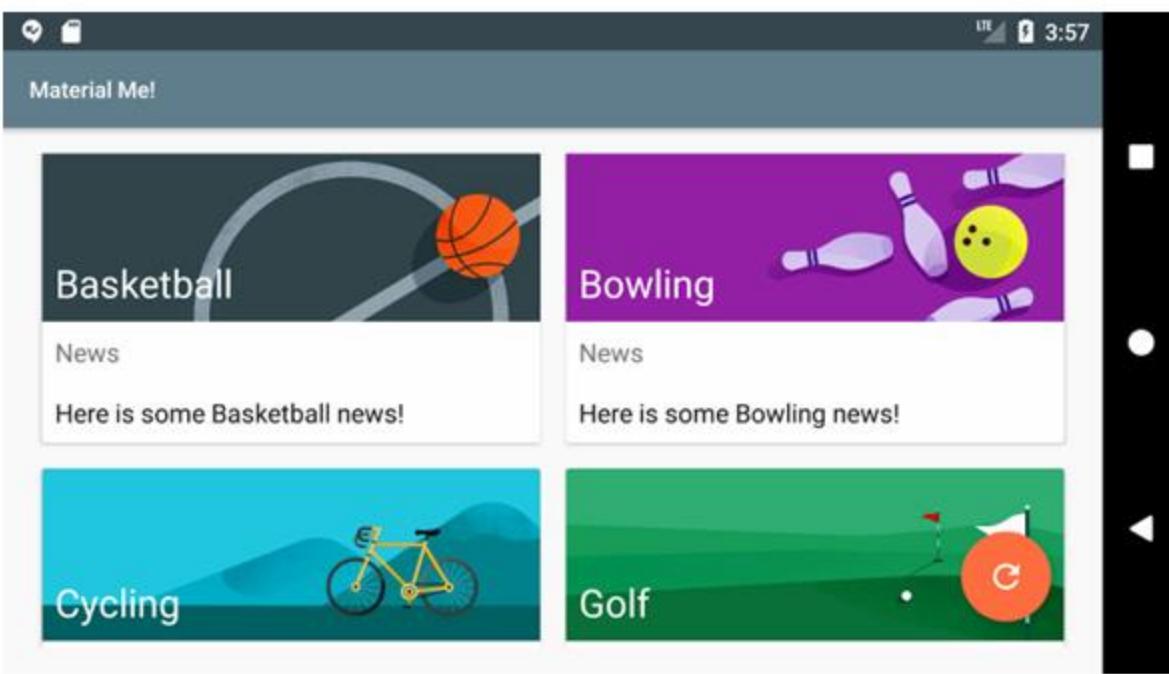
Bạn sẽ học cách:

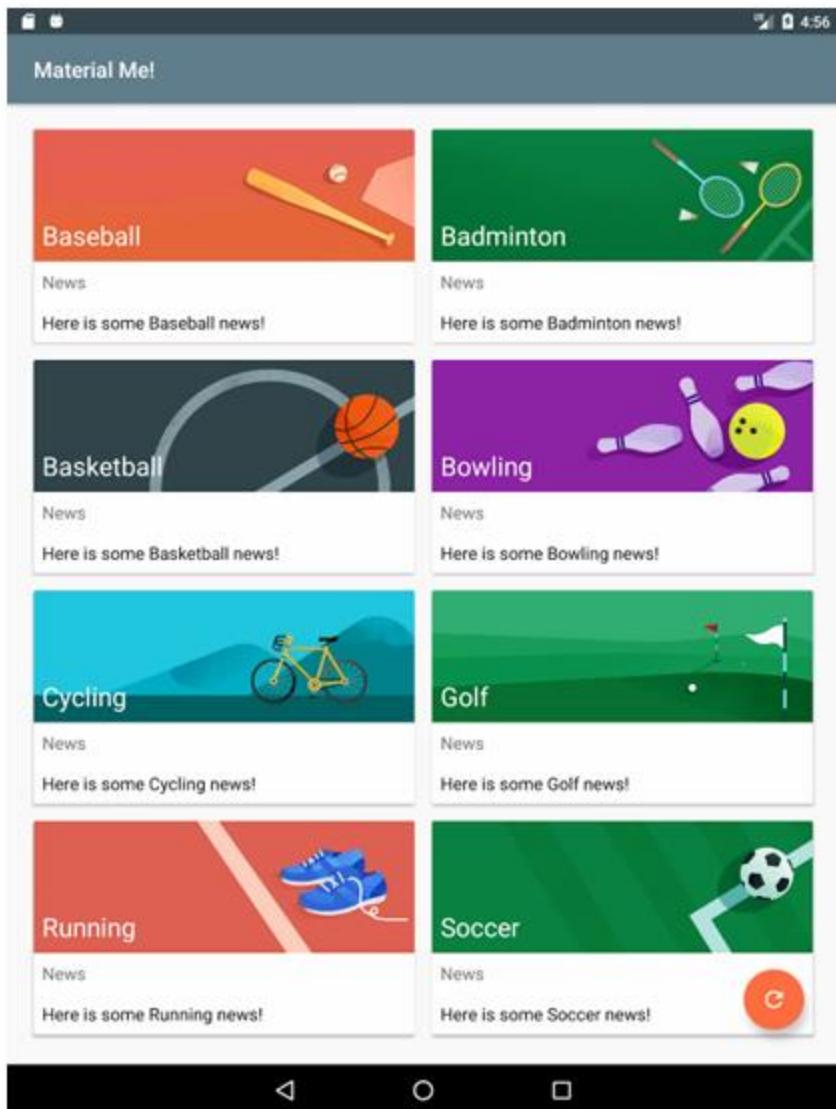
- Tạo tài nguyên thay thế cho các thiết bị ở chế độ ngang.
- Tạo tài nguyên thay thế cho máy tính bảng.
- Tạo tài nguyên thay thế cho các ngôn ngữ khác nhau. Những gì bạn sẽ làm ● Cập nhật ứng dụng MaterialMe để sử dụng không gian tốt hơn ở chế độ ngang.
- Thêm bố cục thay thế cho máy tính bảng.
- Bản địa hóa nội dung ứng dụng của bạn.

Tổng quan về ứng dụng

Ứng dụng MaterialMe được cập nhật sẽ bao gồm bố cục được cải thiện cho chế độ ngang trên điện thoại. Nó cũng sẽ bao gồm bố 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.

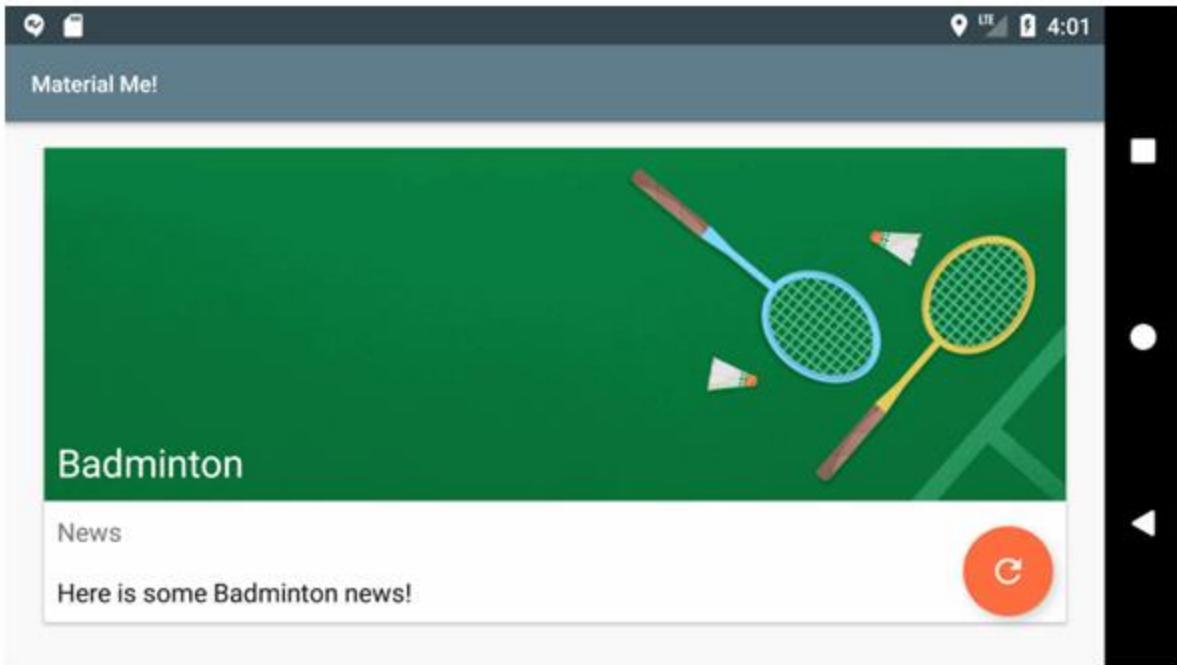
Ảnh chụp màn hình bên dưới cho thấy điện thoại chạy ứng dụng MaterialMe được cập nhật theo hướng ngang:





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 LayoutManager ; 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 bố 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 bố 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:

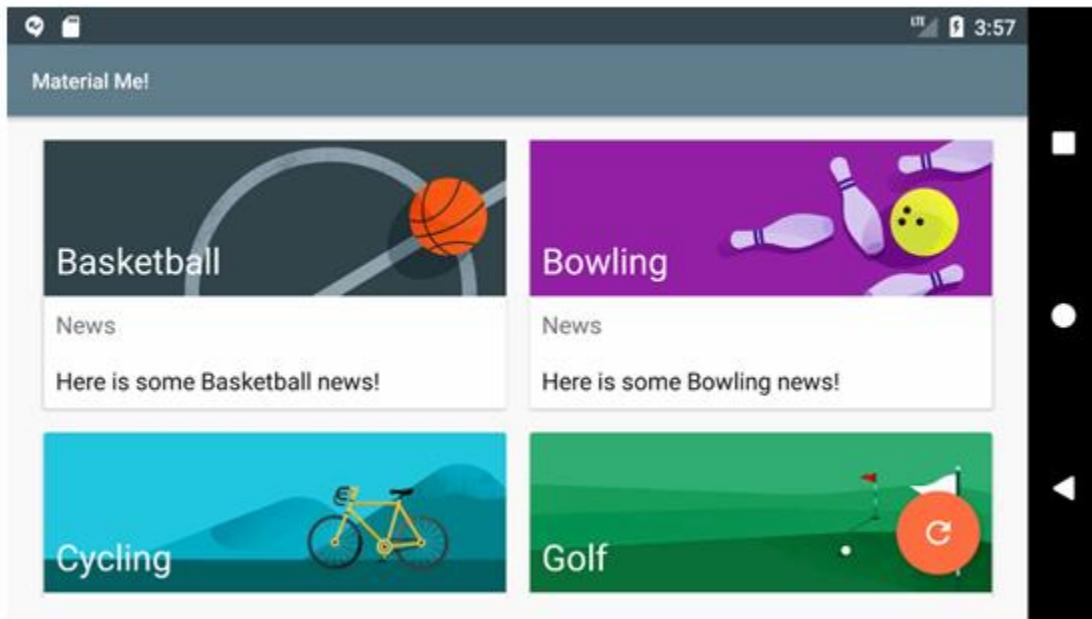
```
int gridColumnCount =  
    getResources().getInteger(R.integer.grid_column_count);
```

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 lưới chứ 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()

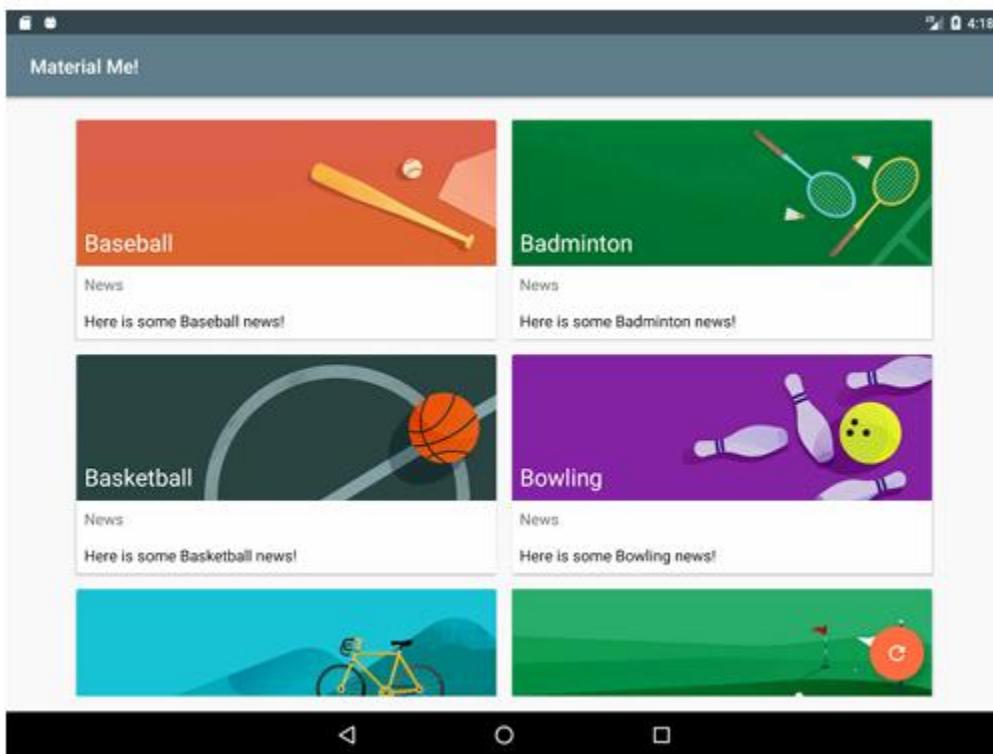
```
ItemTouchHelper helper = new ItemTouchHelper(new
    ItemTouchHelper.SimpleCallback(ItemTouchHelper.LEFT |
        ItemTouchHelper.RIGHT |
        ItemTouchHelper.DOWN | ItemTouchHelper.UP,
        swipeDirs) {
```

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) Bài 1. Các tác vụ nền

1.1) 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

- 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.
- Mở tệp giao diện **activity_main.xml**. Nhập vào tab **Text**.
- Thêm thuộc tính `layout_margin` vào **ConstraintLayout** cấp cao nhất:

```
xml
CopyEdit
android:layout_margin="16dp"
```

- 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"

- Xóa các thuộc tính `app:layout_constraintRight_toRightOf` và `app:layout_constraintTop_toTopOf`.
- 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>
```

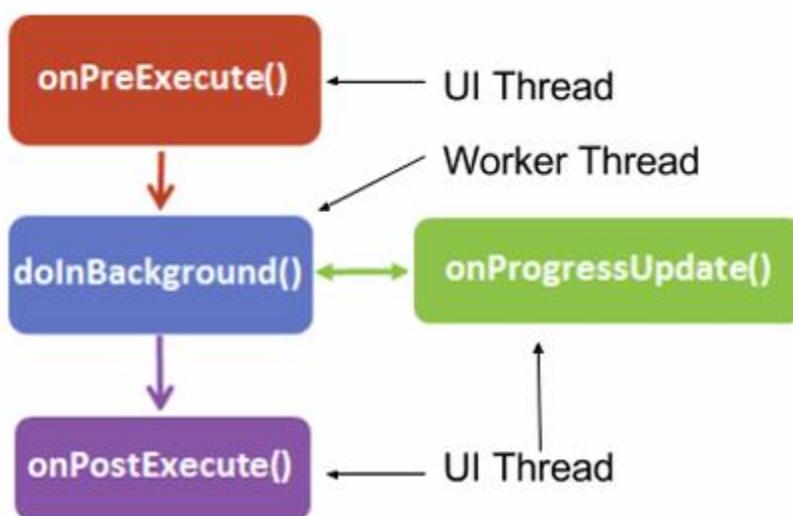
Task 2: Tạo lớp con AsyncTask

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(), hiển 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 ngũ cảnh rò rỉ (leaky context)**, và Android Studio sẽ cảnh báo bạn nếu bạn cố 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;  
}
```

2. 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;
```

3. Thêm một khối try/catch và đưa luồng vào trạng thái ngủ:

java

CopyEdit

```
try {  
    Thread.sleep(s);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}
```

4. 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().

- 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;
```

2. Trong phương thức onCreate(), khởi tạo mTextView để liên kết với TextView trong giao diện:

java

CopyEdit

```
mTextView = findViewById(R.id.textView1);
```

3. 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 thể 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 chạy 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 giờ cập nhật TextView, cái mà đã được chuyển vào, bởi vì TextView đó giờ 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 `AsyncTaskLoader`. 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 đâu class thêm hằng số để làm khóa cho văn bản hiện tại trong Bundle trạng thái

4. Ghi đè `onSaveInstanceState()` method trong MainActivity 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.textView);

    // 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 AsyncTask, bạn cần xác minh kiểu dữ liệu để sử dụng trong phương thức onProgressUpdate

Tóm tắt:

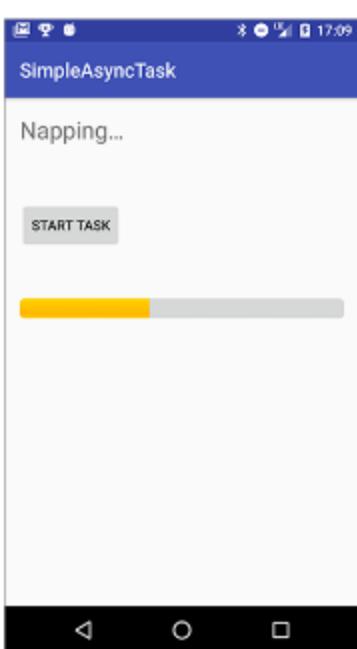
- AsyTask là 1 lớp trừu tượng giúp chuyển các tác vụ xử lý nặng sang một luồng riêng biệt (a separate thread.)
- AsynTAsk phải được kế thừa(subclass) từ lớp con để sử dụng
- Tránh làm các công việc tốn nhiều tài nguyên trên luồng giao diện, bối ví nó làm giao diện chậm or không ổn định (erratic)
- Tất 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 chấm 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 nhật thanh progressbar sau mỗi phần
- AsyncTask gọi phương thức phù hợp và thực thi call back phù hợp để cập nhật 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

1.1 **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 nhận 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

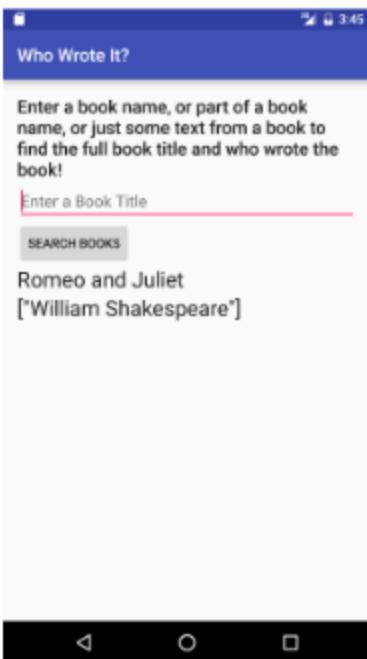
Cách thực thi AsyncTaskLoader cái mà bảo tồn dữ liệu của bạn khi cấu hình thay đổi

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/>.

Nhấp vào Services trong thanh điều hướng bên trái, sau đó chọn Books API.

Tìm books.volumes.list và nhấp 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.

Nhấp 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

www.applab.it/ru, you will find more 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 nộp

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 thừa AsyncTask. 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 khởi 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        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 muốn tạo lớp tiện chử chức 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.
- Tạo một biến LOG_TAG để ghi log, đặt tên theo tên của lớp:


```
private static final String LOG_TAG =  
    NetworkUtils.class.getSimpleName()
```
 - Tạo một hàm tĩnh tên getBookInfo(). getBookInfo() lấy các chuỗi và trả về jsonString từ api, bạn sẽ được kiểm tra sớm
 - Tạo các biến local dưới đây trong getBookinfor():


```
HttpURLConnection urlConnection = null;  
BufferedReader reader = null;  
String bookJSONString = null;
```
 - Cuối cùng, trả về giá trị bookJSONString


```
Return bookJSONString;
```
 - Thêm cấu trúc try catch finnally trong getBookInfo, sau biến và trước 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:

- Tạo các hằng số dưới đây ở đầu class NetWork Utils:


```
ase 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 đều 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 iểu dữ liệu trả về để tăng tốc độ truy vấn.
Để gián 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
buferreReader 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 buferedReader
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:

```
inally {
    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 `dolnBackground()` 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 `onPostExcute`, thêm `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

```
nt 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 nhật Ui vs phản hồi đó. Bởi vì tương trưng cho đối tượng textView là weakReference, bạn phải gạn 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 no results
else {

```
    mTitleText.get().setText(R.string.no_results);  
    mAuthorText.get().setText("");  
}
```

7, trong khói catch{}, in lỗi. Set title textView 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 khhoongf 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 cố gắng 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 vấn đề cuối trong task 4

3.1 ẩn 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 thấy truy vấn đang thực hiện.

Giải pháp là để ẩn 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 ẩn 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 thêm 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 khối else, cài 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, Chỉnh 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 nhập 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 thể 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.
- **Thể 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 thể 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 đổi 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 trợ 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 getSupportLoaderManager().initLoader().

Khái niệm liên quan

Bài tập về nhà

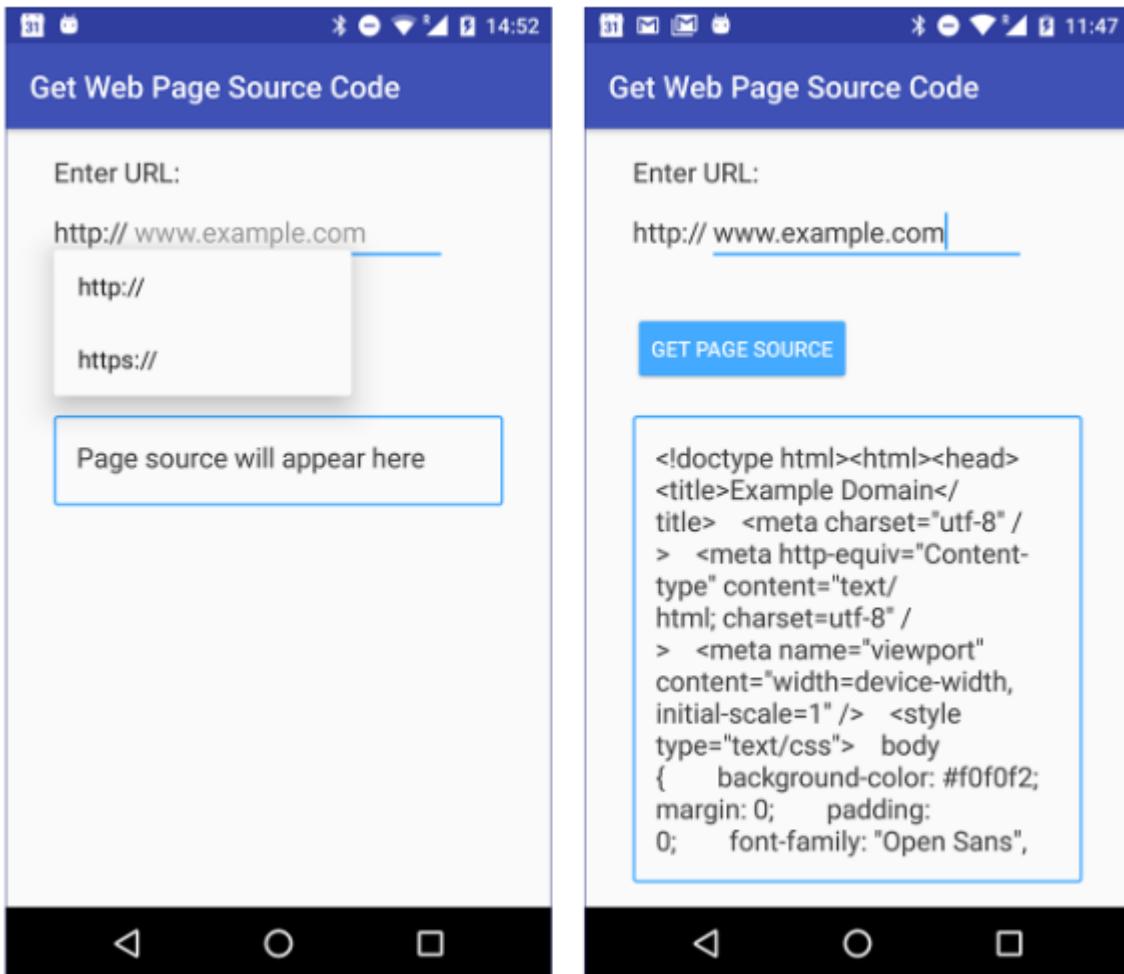
Bạn cần xây dựng một ứng dụng Android hiển thị nội dung của một trang web từ URL do người dùng nhập. Ứng dụng sẽ có các thành phần sau:

- 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.

1.2 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) Chính 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, Nhấp 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, Nhấp 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.

Giữ **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) Bài 2. Kích hoạt lập lịch và thông báo nhiệm vụ

2.1) 2.1 Thông báo

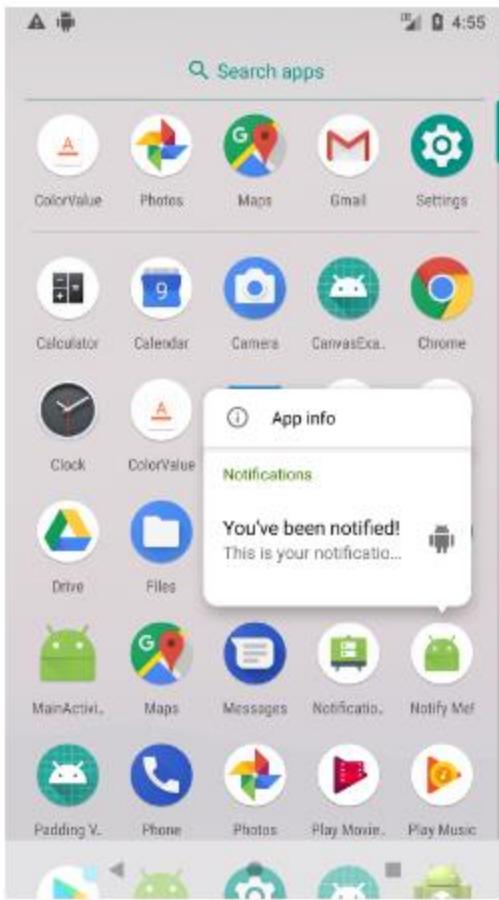
Giới thiệu

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.

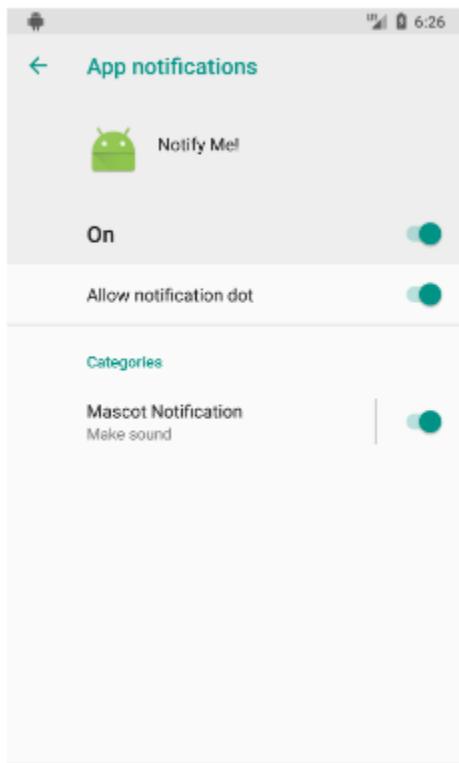
Task1: Tạo thông báo đơn giản

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, nhấp 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, Nhấp vào biểu tượng bên cạnh mục Clip Art để chọn một biểu tượng theo tiêu chuẩn Material Design cho thông báo.

Đối 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:

1.3 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.)

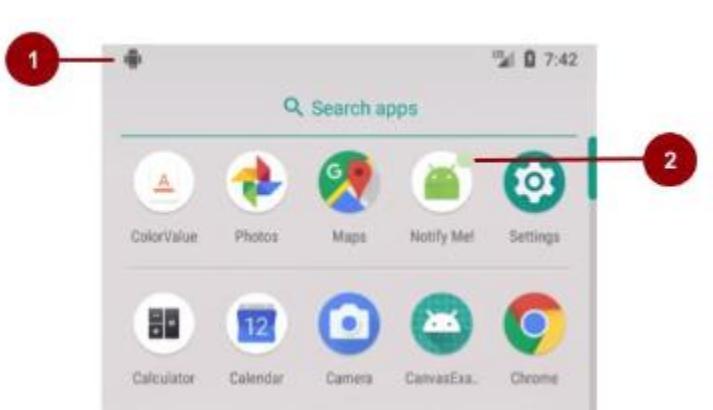
Thực hiện trong MainActivity.java

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 =
```

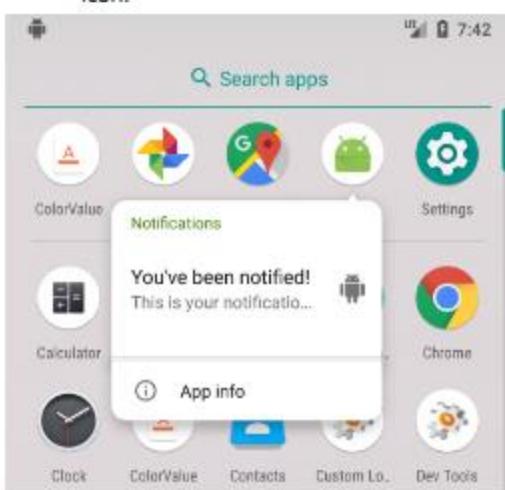
```
PendingIntent.getActivity(this,  
NOTIFICATION_ID, notificationIntent,  
PendingIntent.FLAG_UPDATE_CURRENT);
```



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

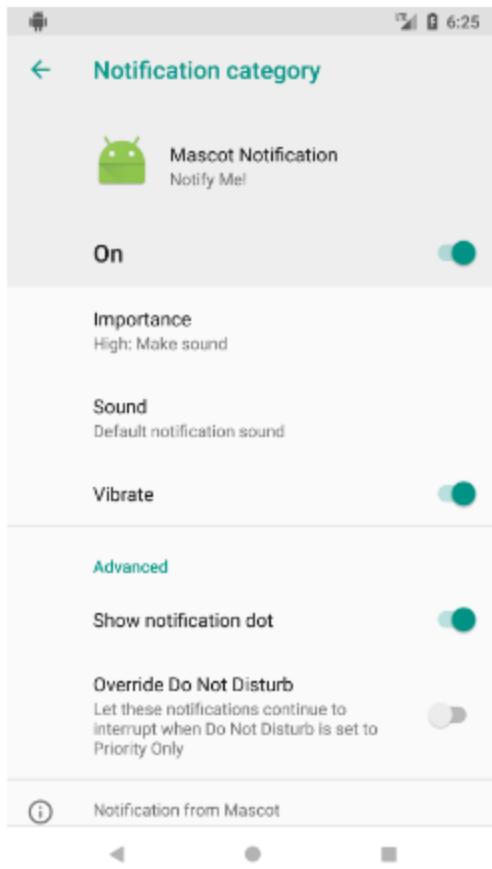


Nếu bạn đang chạy trên thiết bị hoặc trình giả lập với API 26 trở lên, đây là cách xem kênh thông báo mà bạn đã tạo:

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, them 1 biến chõ 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) 2.2 Trình quản lý cảnh báo

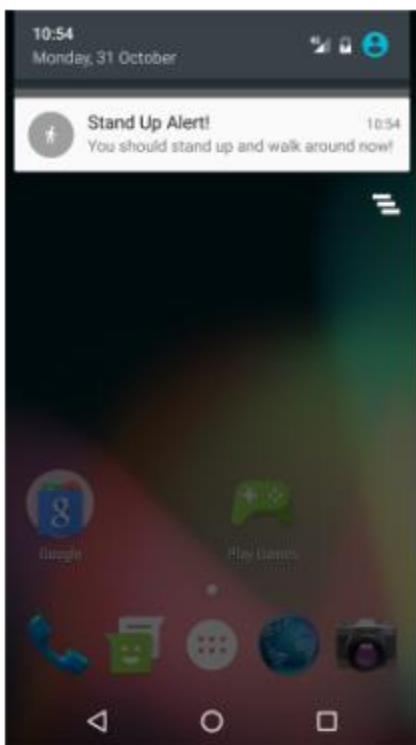
Giới thiệu

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**.
 - o 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)
```



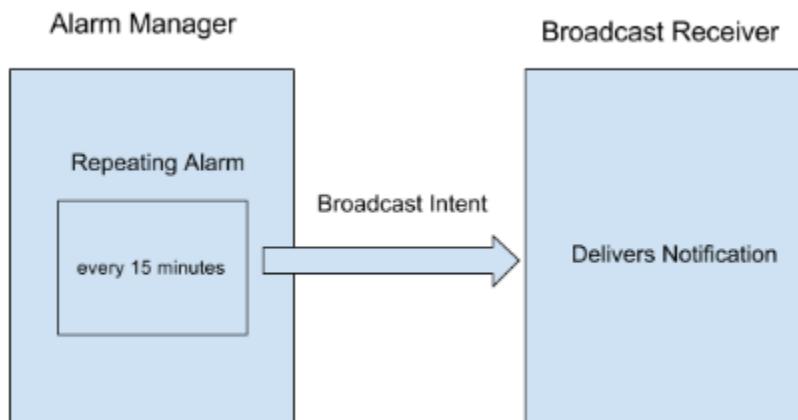
Task 3: Tạo báo thức lặp lại

Bây giờ ứng dụng của bạn có thể gửi thông báo, đã đến lúc triển khai thành phần chính của ứng dụng: AlarmManager. Lớp này sẽ gửi lời nhắc đứng 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

```

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);

```

- 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).

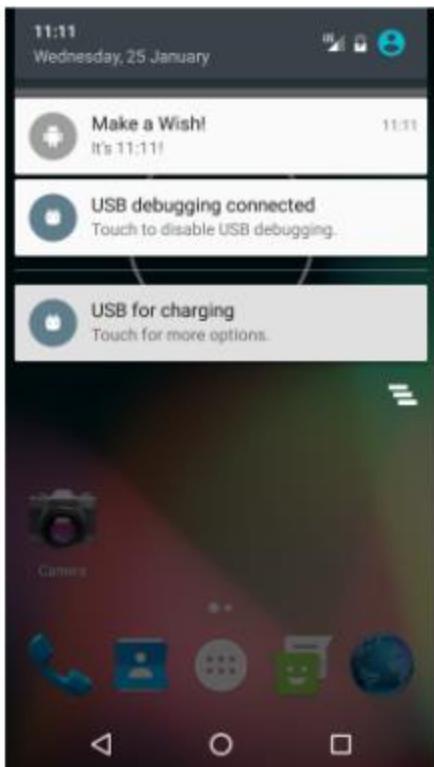
Code challenge: Lớp AlarmManager cũng xử lý loại báo thức thông thường, loại báo thức đánh thức bạn vào buổi sáng. Trên các thiết bị chạy API 21 trở lên, bạn có thể lấy thông tin về báo thức tiếp theo bằng cách gọi getNextAlarmClock() trên AlarmManager.

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 lên 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 lên 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.

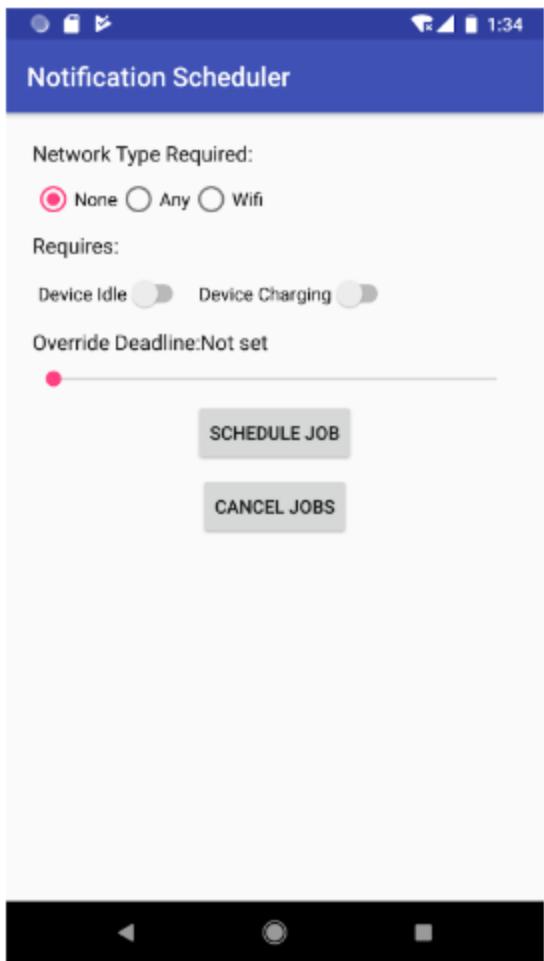
Những gì bạn cần biết trước

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 lên 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 lênh lịch 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("Job Service notification");
    }
}
```

```
("Notifications from Job Service");  
mNotifyManager.createNotificationChannel(notificationChannel);
```

Triển khai điều kiện chạy công việc (JobInfo)

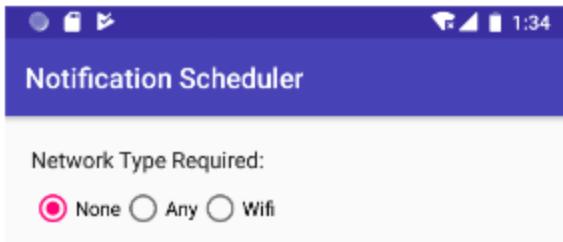
Sau khi đã tạo JobService, bước tiếp theo là xác định các tiêu chí để chạy công việc. Để làm điều này, bạn sử dụng JobInfo. Bạn sẽ tạo một nhóm điều kiện dựa trên loại kết nối mạng và trạng thái thiết bị.

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
 - Công việc sẽ chạy dù có hoặc không có kết nối mạng.
 - Đây là giá trị mặc định nếu không có điều kiện mạng nào được đặt.
2. NETWORK_TYPE_ANY
 - Công việc sẽ chạy khi có bất kỳ mạng nào khả dụng (Wi-Fi hoặc mạng di động).
3. NETWORK_TYPE_UNMETERED
 - 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).

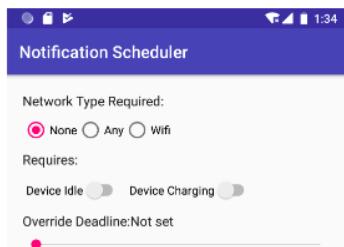
2, Cập nhật biến constraintSet khi thêm các tùy chọn khác

- 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. LƯU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

1.1) Shared preferences

1.2) Cài đặt ứng dụng

Bài 2) Lưu trữ dữ liệu với Room

2.1) Room, LiveData và ViewModel

2.2) Room, LiveData và ViewModel