

Nhà phát triển Android

Khóa học cơ bản (Phiên bản 2)



Cập nhật lần cuối: Tue Sep 11, 2018

Khóa học này được tạo bởi nhóm Đào tạo nhà phát triển của Google.

- Để biết chi tiết về khóa học, bao gồm các liên kết đến tất cả các chương khái niệm, ứng dụng và trang trình bày, hãy xem [Kiến thức cơ bản về nhà phát triển Android \(Phiên bản 2\).](#)

developer.android.com/courses/ a dfv2

Lưu ý: Khóa học này sử dụng các thuật ngữ "codelab" và "thực hành" thay thế cho nhau.

Chúng tôi khuyên bạn nên sử dụng [phiên bản online](#) của khóa học này thay vì PDF tĩnh này để đảm bảo bạn đang sử dụng nội dung mới nhất.

Xem developer.android.com/courses/adf-v2.

Bài 2: Trải nghiệm người dùng

PDF này chứa ảnh chụp nhanh một lần về các bài học trong **Unit 2: Trải nghiệm người dùng**.

Bài học trong bài học này

Bài 4: Tương tác người dùng

4.1 : Hình ảnh có thể nhấp

4.2: Điều khiển đầu vào

4.3 : Menu và bộ chọn

4.4 : Điều hướng người dùng

4.5: Người tái chếView

Bài 5: Trải nghiệm người dùng thú vị

5.1 : Có thể vẽ, kiểu và chủ đề

5.2 : Thủ và màu sắc

5.3 : Bố cục thích ứng

Bài học 6: Kiểm tra giao diện người dùng

6.1 : Espresso để kiểm tra giao diện người dùng

Bài

4.1: Hình ảnh có thể nhấp

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị chạy Android bao gồm một hệ thống phân cấp các đối tượng được gọi là `Views`. Mọi yếu tố của màn hình đều là một chữ `View`.

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

Bạn có thể biến bất kỳ `View` nào, chẳng hạn như `ImageView`, thành một phần tử giao diện người dùng có thể chạm hoặc nhấp vào. Bạn phải lưu trữ hình ảnh cho `ImageView` trong thư mục `rawables` của dự án của bạn.

Trong thực tế này, bạn học cách sử dụng hình ảnh làm yếu tố mà người dùng có thể nhấn hoặc nhấp vào.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo một dự án Android Studio từ một mẫu và tạo bố cục chính.
- Chạy ứng dụng trên trình mô phỏng hoặc thiết bị được kết nối.
- Tạo và chỉnh sửa các thành phần giao diện người dùng bằng trình chỉnh sửa bố cục và mã XML.
- Truy cập các thành phần giao diện người dùng từ mã của bạn bằng `findViewById()`.
- Xử lý một cú `nhấp chuột` B.
- Hiển thị thông `báo Toast`.
- Thêm hình ảnh vào thư mục có thể thô của dự án.

Những gì bạn sẽ học

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

Sẽ làm gì

- Tạo một dự án Android Studio mới cho một ứng dụng đặt món tráng miệng giả sử dụng hình ảnh làm yếu tố tương tác.
- Đặt các trình xử lý `onClick()` cho các hình ảnh để hiển thị các thông báo `Toast` khác nhau.
- Thay đổi nút hành động nối do mẫu cung cấp để nó hiển thị một biểu tượng khác và khởi chạy một biểu tượng `A` khác.

Tổng quan về ứng dụng

Trong thực tế này, bạn tạo và xây dựng một ứng dụng mới bắt đầu với mẫu Hoạt động cơ bản bắt chước ứng dụng đặt món tráng miệng. Người dùng có thể nhấn vào một hình ảnh để thực hiện một hành động — trong trường hợp này hiển thị thông báo `Toast` — như thể hiện trong hình bên dưới. Người dùng cũng có thể nhấn vào nút giờ hàng để chuyển sang chương trình `A` tiếp theo.



Nhiệm vụ 1: Thêm hình ảnh vào bố cục

Bạn có thể làm cho một chế độ xem có thể nhấp được, như một nút, bằng cách thêm thuộc tính android:onClick trong bố cục XML. Ví dụ: bạn có thể làm cho một hình ảnh hoạt động giống như một nút bằng cách thêm android:onClick vào [Chế độ xem hình ảnh](#).

Trong nhiệm vụ này, bạn tạo một nguyên mẫu của một ứng dụng để đặt món tráng miệng từ một quán cà phê. Sau khi bắt đầu một dự án mới dựa trên mẫu Hoạt động cơ bản, bạn sửa đổi "Hello World" TextView với văn bản thích hợp và thêm hình ảnh mà người dùng có thể chạm.

1.1 Bắt đầu dự án mới

1. Bắt đầu một dự án Android Studio mới với tên ứng dụng Droid Cafe.
2. Chọn mẫu **Hoạt động Basic** và chấp nhận tên Activity mặc định (MainActivity). Đảm bảo các tùy chọn **Bố cục Generate File** và **Backwards Compatibility (AppCompat)** được chọn.
3. Nhấp vào **Finish**.

Dự án mở ra với hai bố cục trong **thư mục** bố cục res >: activity_main.xml cho thanh ứng dụng và nút hành động nổi (mà bạn không thay đổi trong tác vụ này) và content_main.xml cho mọi thứ khác trong bố cục.

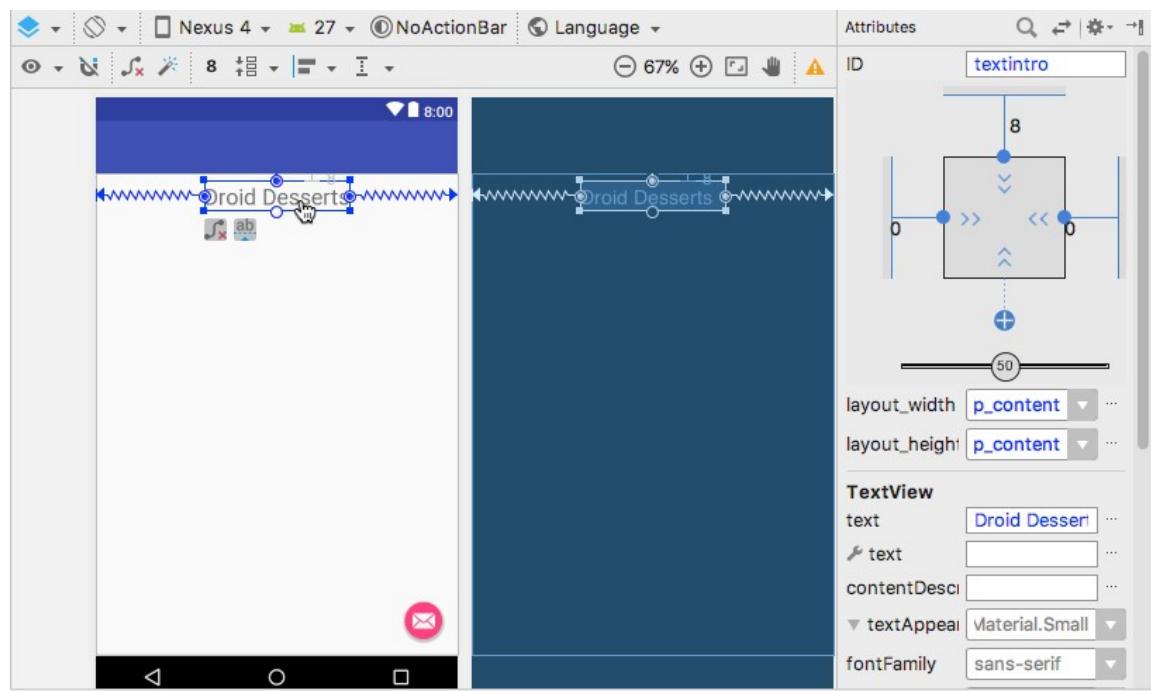
4. Mở **content_main.xml** và nhấp vào tab **Design** (nếu nó chưa được chọn) để hiển thị trình chỉnh sửa bố cục.
5. Chọn "Hello World" TextView trong bố cục và mở ngăn **Attributes**.
6. Thay đổi các thuộc tính textintro như sau:

Trường thuộc tính	Nhập như sau:
ID	văn bản giới thiệu
Nhắn tin	Thay đổi thế giới Hello thành món tráng miệng Droid

textStyle	B (in đậm)
Kích thước văn bản	24 điểm

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.

- Xóa ràng buộc kéo dài từ dưới cùng của `TextView` đến cuối bố cục, để `TextView` gắn vào đầu bố cục và chọn 8 (8dp) cho lề trên cùng như hình dưới đây.



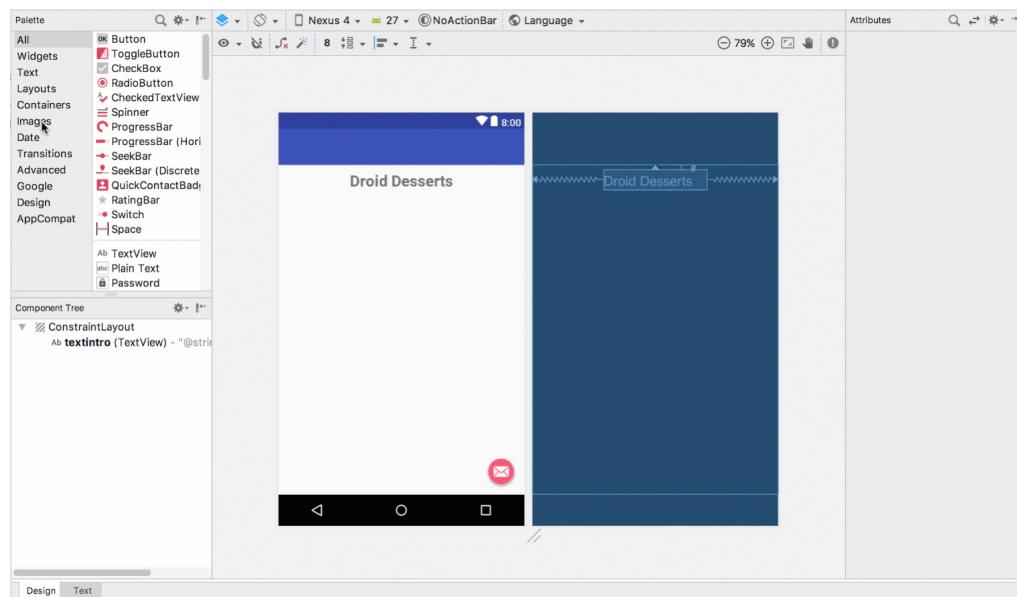
- Trong bài học trước, bạn đã học cách trích xuất tài nguyên chuỗi từ chuỗi văn bản theo nghĩa đen. Nhấp vào tab **Text** để chuyển sang mã XML và giải nén chuỗi "Droid Desserts" trong `TextView` và nhập **vào tro_text** làm tên tài nguyên chuỗi.

Thêm hình ảnh

Ba hình ảnh (donut_circle.png, froyo_circle.png và icecream_circle.png) được cung cấp cho ví dụ này, bạn có thể tải mà bạn có thể tải. Thay vào đó, bạn có thể thay thế hình ảnh của riêng mình 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 thông báo 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 sau: `project_name Ứng dụng > src > main > res > drawable`.
- Mở lại dự án của bạn.
- Mở **tệp content_main.xml** và nhấp vào tab **chữ ký D** (nếu nó chưa được chọn).
- Kéo ImageView vào bố cục, chọn hình ảnh **donut_circle** cho nó và hạn chế nó ở TextView trên cùng và sang phía bên trái của bố cục với lề 24 (24dp) cho cả hai ràng buộc, như được hiển thị trong hình động bên dưới.

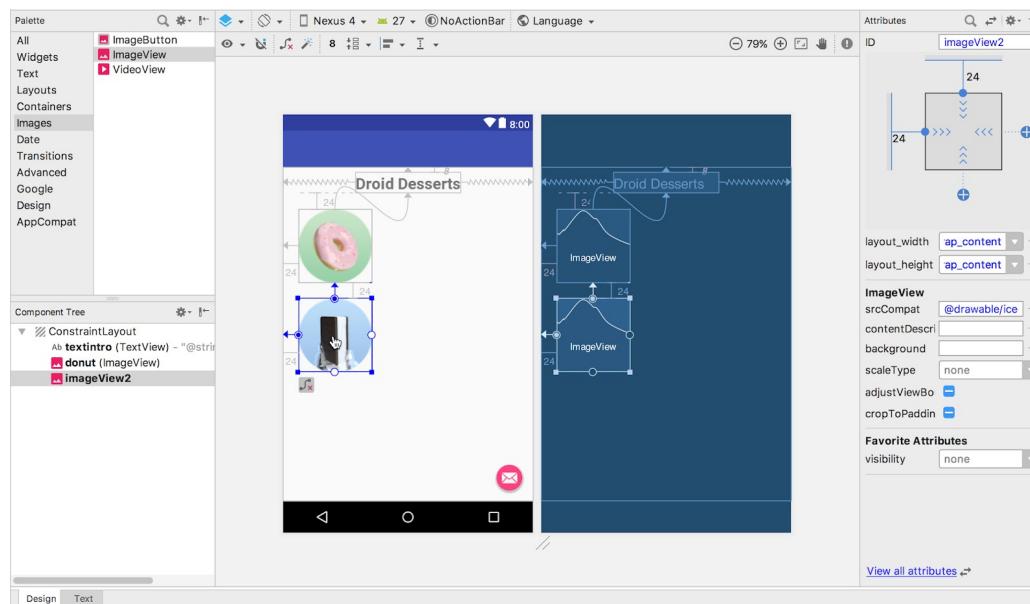


- Trong ngăn **Attributes**, nhập các giá trị sau cho các thuộc tính:

Trường thuộc tính	Nhập như sau:
--------------------------	----------------------

ID	Donut
nội dung Mô tả	Bánh rán được tráng men và rắc kẹo. (Bạn có thể sao chép/dán văn bản vào trường.)

7. Kéo ImageView thứ hai vào bố cục, chọn **ic_icecream_circle** hình ảnh cho nó và hạn chế nó ở cuối ImageView đầu tiên và sang phía bên trái của bố cục với lề 24 (24dp) cho cả hai ràng buộc.



8. Trong ngăn Attributes, nhập các giá trị sau cho các thuộc tính:

Trường thuộc tính	Nhập như sau:
ID	ice_cream

nội dung	Mô tả
	Bánh mì kẹp kem có bánh xốp sô cô la và nhân vani. (Bạn có thể sao chép/dán văn bản vào trường.)

9. Kéo ImageView thứ ba vào bố cục, chọn hình ảnh `froyo_circle` cho nó và hạn chế nó ở dưới cùng của ImageView thứ hai và sang phía bên trái của bố cục với lề 24 (24dp) cho cả hai ràng buộc.
10. Trong ngăn Attributes, nhập các giá trị sau cho các thuộc tính:

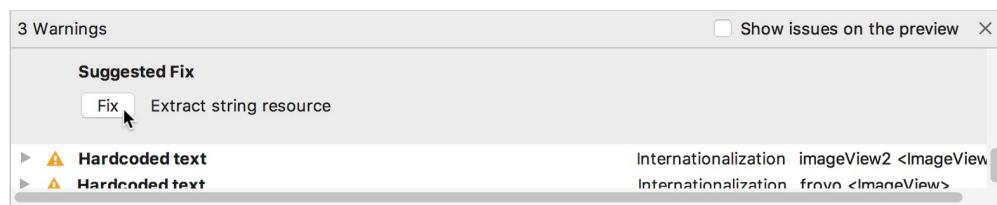
Trường thuộc tính	Nhập như sau:
ID	Froyo

nội dung Mô tả	FroYo là sữa chua đông lạnh tự phục vụ cao cấp. (Bạn có thể sao chép/dán văn bản vào trường.)
----------------	--

11. Nhấp vào biểu tượng cảnh báo  ở góc trên bên trái của trình chỉnh sửa bố cục để mở ngăn cảnh báo, ngăn này sẽ hiển thị cảnh báo về văn bản được mã hóa cứng:



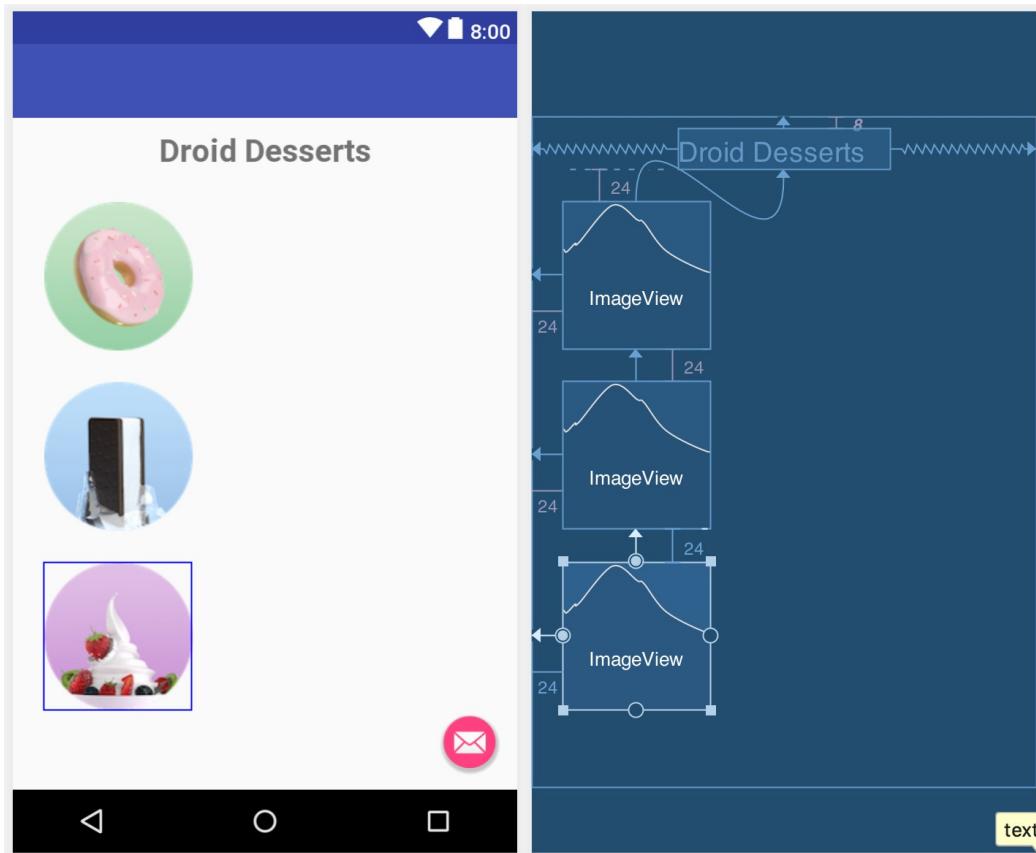
12. Mở rộng từng cảnh báo **văn bản được mã hóa** H, cuộn xuống cuối thông báo cảnh báo và nhấp vào nút Fix như hình dưới đây:



Bản sửa lỗi cho mỗi cảnh báo văn bản được mã hóa cứng sẽ trích xuất tài nguyên chuỗi. **Hộp thoại Trích xuất tài nguyên** xuất hiện và bạn có thể nhập tên cho tài nguyên chuỗi. Nhập các tên sau cho tài nguyên chuỗi:

XÂU	NHẬP TÊN SAU:
Bánh rán được tráng men và rắc kẹo.	Donuts
Bánh mì kẹp kem có bánh xốp sô cô la và nhân vani.	ice_cream_sandwiches
FroYo là sữa chua đông lạnh tự phục vụ cao cấp.	Froyo

Bố cục bây giờ sẽ trông giống như hình bên dưới.

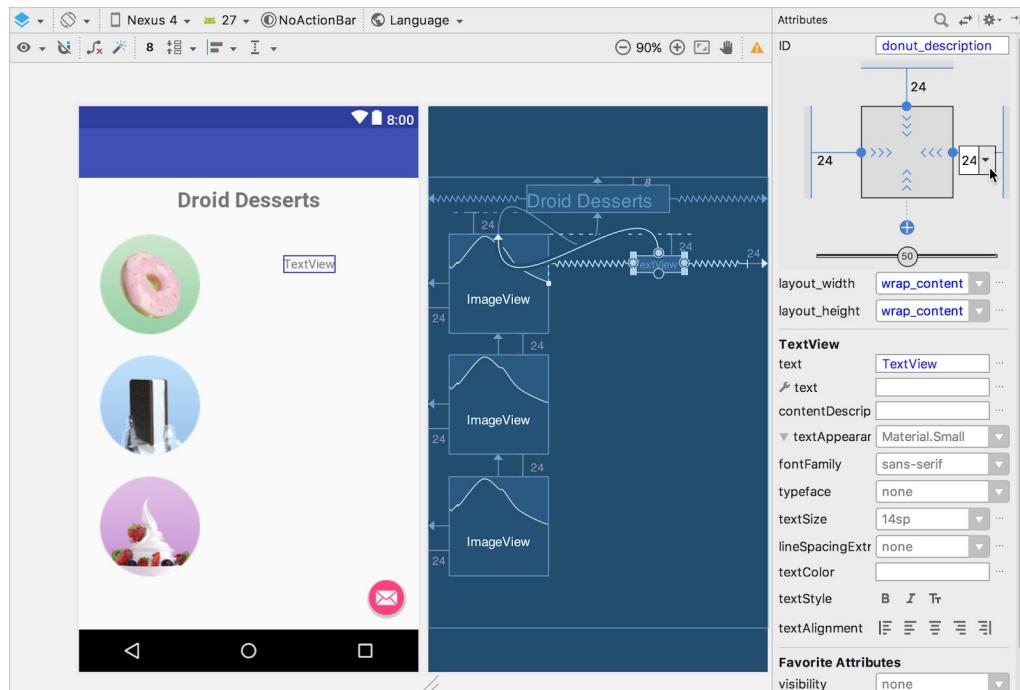


1.3 Thêm mô tả văn bản

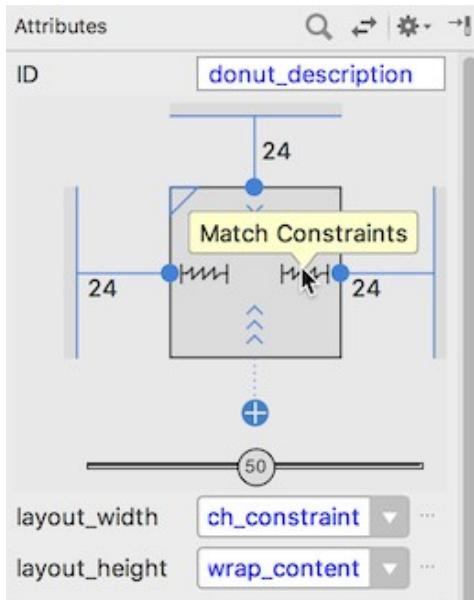
Trong bước này, bạn thêm mô tả văn bản (TextView) cho mỗi món tráng miệng. Vì bạn đã trích xuất tài nguyên chuỗi cho các trường contentDescription cho các phần tử ImageView, bạn có thể sử dụng cùng một tài nguyên chuỗi cho mỗi mô tả TextView.

1. Kéo phần tử TextView vào bố cục.

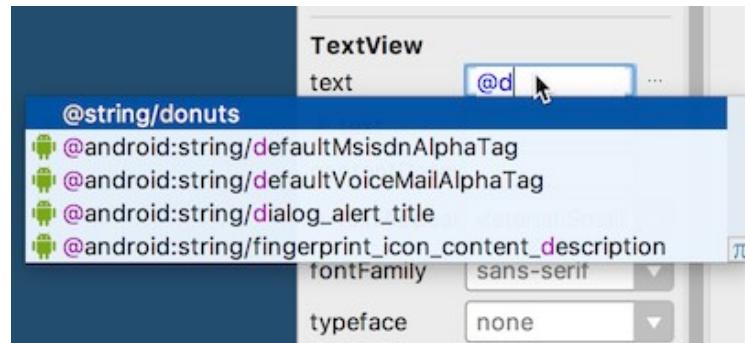
2. Hạn chế phía bên trái của phần tử ở phía bên phải của `donut ImageView` và trên cùng của nó ở trên cùng của `donut ImageView`, cả hai đều có biên độ là **24** (24dp).
3. Hạn chế phía bên phải của phần tử ở phía bên phải của bố cục và sử dụng cùng một lề của **24** (24dp). Nhập `donut_description` cho trường `ID` trong ngăn `Attributes`. `TextView` mới sẽ xuất hiện bên cạnh hình ảnh bánh rán như trong hình bên dưới.



4. Trong ngăn **Phân bổ A**, thay đổi chiều rộng trong ngăn kiểm tra thành **Match Constraints**:



5. Trong ngăn **A ttributes**, bắt đầu nhập tài nguyên chuỗi cho trường `text` bằng cách đặt đầu nó bằng ký hiệu `@: @ d`. Nhập vào tên tài nguyên chuỗi (`@ string/donuts`) xuất hiện dưới dạng gợi ý:



6. Lặp lại các bước trên để thêm một `TextView` thứ hai bị hạn chế ở phía bên phải và trên cùng của `ice_cream ImageView` và phía bên phải của nó ở phía bên phải của bố cục. Nhập thông tin sau vào ngăn **A ttributes**:

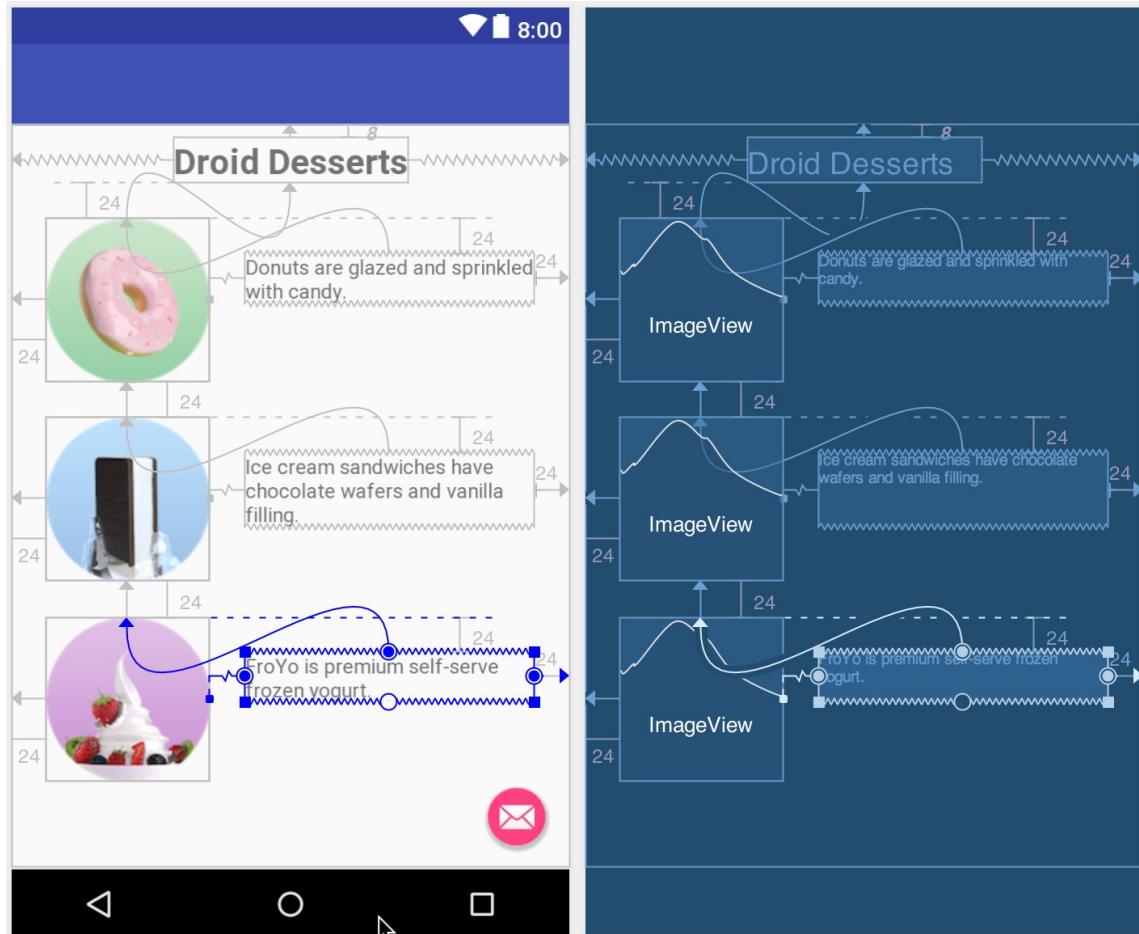
Trường thuộc tính	Nhập như sau:
ID	ice_cream_description

Lề trái, phải và trên	24
layout_width	match_constraint
Nhắc tin	@string/ice_cream_sandwiches

7. Lặp lại các bước ở trên để thêm một TextView thứ ba bị hạn chế ở phía bên phải và trên cùng của froyo ImageView và phía bên phải của nó ở phía bên phải của bố cục. Nhập thông tin sau vào ngăn Attributes:

Trường thuộc tính	Nhập như sau:
ID	froyo_description
Lề trái, phải và trên	24
layout_width	match_constraint
Nhắc tin	@string / froyo

Bố cục bây giờ sẽ trông như sau:



Mã giải pháp nhiệm vụ 1

Bố cục XML cho tệp `content.xml` được hiển thị bên dưới.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

```
<?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"
app:layout_behavior="@string/appbar_scrolling_view_behavior"
tools:context="com.example.android.droidcafe.MainActivity"
công cụ:showIn="@layout/activity_main">

<Chế độ xem văn bản
    android:id="@+id/textintro"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/margin_regular"
    android:text="@string/intro_text"
    android:textSize="@dimen/text_heading"
    android:textStyle="đậm"
    app:layout_constraintLeft_toLeftOf="cha mẹ"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
<Chế độ xem hình ảnh
    android:id="@+id/bánh rán"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    android:contentDescription="@string/bánh rán"
    app:layout_constraintStart_toStartOf="cha mẹ"
    app:layout_constraintTop_toBottomOf="@+id/textintro"
    app:srcCompat="@drawable/donut_circle" />

<Chế độ xem hình ảnh
    android:id="@+id/ice_cream"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    android:contentDescription="@string/ice_cream_sandwiches"
    app:layout_constraintStart_toStartOf="cha mẹ"
```

```
app:layout_constraintTop_toBottomOf="@+id/bánh rán"
```

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

<Chế độ xem hình ảnh
    android:id="@+id/froyo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    android:contentDescription="@string/froyo"
    app:layout_constraintStart_toStartOf="cha mẹ"
    ứng dụng:layout_constraintTop_toBottomOf="@+id/ice_cream"
    app:srcCompat="@drawable/froyo_circle" />

<Chế độ xem văn bản
    android:id="@+id/donut_description"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="@dimen/margin_wide"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    android:text="@string/bánh rán"
    app:layout_constraintEnd_toEndOf="cha mẹ"
    app:layout_constraintStart_toEndOf="@+id/donut"
    app:layout_constraintTop_toTopOf="@+id/donut" />
    <Chế độ xem văn bản
        android:id="@+id/ice_cream_description"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="@dimen/margin_wide"
        android:layout_marginStart="@dimen/margin_wide"
        android:layout_marginTop="@dimen/margin_wide"
        android:text="@string/ice_cream_sandwiches"
        app:layout_constraintEnd_toEndOf="cha mẹ"
        app:layout_constraintStart_toEndOf="@+id/ice_cream"
        app:layout_constraintTop_toTopOf="@+id/ice_cream" />
    <Chế độ xem văn bản
        android:id="@+id/froyo_description"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="@dimen/margin_wide"
```

```
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
        android:text="@string/froyo"
        app:layout_constraintEnd_toEndOf="cha mẹ"
        app:layout_constraintStart_toEndOf="@+id/froyo"
        app:layout_constraintTop_toTopOf="@+id/froyo" />
```

```
</android.support.constraint.ConstraintLayout>
```

Nhiệm vụ 2: Thêm phương thức onClick cho hình ảnh

Để làm cho View có thể kiểm được để người dùng có thể chạm (hoặc nhấp vào) nó, hãy thêm thuộc [tính năng android:onClick](#) trong

XML và chỉ định trình xử lý nhấp chuột. Ví dụ: bạn có thể làm cho `Imageview` hoạt động giống như một Button 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 nhấp vào.

2.1 Tạo phương thức Toast

Trong tác vụ này, bạn thêm từng phương thức cho thuộc tính `onClick` để gọi khi mỗi hình ảnh được nhấp vào. Trong nhiệm vụ này, các phương pháp này chỉ đơn giản là hiển thị thông báo `Toast` cho biết hình ảnh nào đã được khai thác. (Trong một chương khác, bạn sửa đổi các phương pháp này để khởi chạy một Method.)

- Để sử dụng tài nguyên chuỗi trong mã Java, trước tiên bạn nên thêm chúng vào tệp `strings.xml`. Mở rộng `res > giá trị` trong `ngân Project > Android` và mở `strings.xml`. Thêm các tài nguyên chuỗi sau cho các chuỗi được hiển thị trong thông báo `Toast`:

```
<string name="donut_order_message">Bạn đã gọi một chiếc bánh rán.</string> <string name="ice_cream_order_message">Bạn đã gọi một chiếc bánh sandwich kem.</string> <string name="froyo_order_message">Bạn đã đặt hàng FroYo.</string>
```

2. Mở **MainActivity** và thêm phương thức `displayToast()` sau vào cuối **MainActivity** (trước dấu ngoặc đóng):

```
public void displayToast(String message)
{ Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_SHORT).show(); }
```

Mặc dù bạn có thể thêm phương thức này vào bất kỳ vị trí nào trong **MainActivity**, nhưng cách tốt nhất là đặt các phương thức của riêng bạn để giảm các phương thức đã được cung cấp trong **MainActivity** bởi mẫu.

2.2 Tạo trình xử lý nhấp chuột

Mỗi hình ảnh có thể nhấp cần một trình xử lý nhấp chuột — một phương thức cho thuộc tính `onClick` để gọi. Trình xử lý nhấp chuột, nếu được gọi từ thuộc tính `onClick`, phải là `public`, trả về `void` và xác định `View` làm tham số duy nhất của nó. Làm theo các bước sau để thêm trình xử lý nhấp chuột:

- Thêm phương thức `showDonutOrder()` sau vào **MainActivity**. Đối với tác vụ này, hãy sử dụng phương thức `displayToast()` đã tạo trước đó để hiển thị thông báo `Toast`:

```
/***
 * Hiển thị thông báo rằng hình ảnh bánh rán đã được nhấp vào.
 */
public void showDonutOrder(Xem chế độ xem) {
    displayToast(getString(R.string.donut_order_message)); }
```

Ba dòng đầu tiên là một nhận xét ở định dạng [Javadoc](#), giúp mã dễ hiểu hơn và cũng giúp tạo tài liệu cho mã của bạn. Cách tốt nhất là thêm nhận xét như vậy vào mọi phương pháp mới mà bạn tạo. Để biết thêm thông tin về cách viết nhận xét, hãy xem [How to Write Doc Comments for the Javadoc Tool](#).

2. Thêm các phương thức khác vào cuối M ainActivity cho mỗi món tráng miệng:

```
/**  
 * Hiển thị thông báo rằng hình ảnh bánh mì kẹp kem đã được nhấp vào.  
 */ null công khai showIceCreamOrder(View view)  
{ displayToast(getString(R.string.ice_cream_order_message));  
}  
  
/**  
 * Hiển thị thông báo rằng hình ảnh froyo đã được nhấp vào.  
 */ void công khai showFroyoOrder(Xem chế độ  
xem) {  
    displayToast(getString(R.string.froyo_order_message)); }
```

3. (Tùy chọn) Chọn **C ode > Định dạng lại Mã** để định dạng lại mã bạn đã thêm trong M ainActivity để phù hợp với tiêu chuẩn và dễ đọc hơn.

2.3 Thêm thuộc tính onClick

Trong bước này, bạn thêm một android:onClick vào từng phần tử ImageView trong bố cục content_main.xml. Thuộc tính android:onClick gọi trình xử lý nhấp chuột cho mỗi phần tử.

1. Mở **tệp content_main.xml** và nhấp vào tab **T ext** trong trình soạn thảo bố cục để hiển thị mã XML.
2. Thêm thuộc tính android:onClick vào ô onut ImageView. Khi bạn nhập nó, các đề xuất sẽ xuất hiện hiển thị các trình xử lý nhấp chuột. Chọn trình xử lý nhấp chuột cách DonutOrder. Bây giờ mã sẽ trông như sau:

```
<Chế độ xem hình ảnh  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:padding="10dp"
```

```
    android:id="@+id/bánh rán"
    android:layout_below="@+id/choose_dessert"
    android:contentDescription="@string/bánh rán"
    android:src="@drawable/donut_circle"
    android:onClick="showDonutOrder"/>
```

Dòng cuối cùng (`android:onClick="showDonutOrder"`) gán trình xử lý nhấp chuột (`showDonutOrder`) cho `ImageView`.

3. (Tùy chọn) Chọn **Code > Định dạng lại Mã** để định dạng lại mã XML bạn đã thêm vào `content_main.xml` để phù hợp với các tiêu chuẩn và làm cho nó dễ đọc hơn. Android Studio tự động di chuyển thuộc tính `android:onClick` lên một vài dòng để kết hợp chúng với các thuộc tính khác có `android:` làm lời tựa.
4. Làm theo quy trình tương tự để thêm thuộc tính `android:onClick` vào các phần tử `ice_cream` và `froyo` `ImageView`. Chọn trình xử lý nhấp chuột của `showDonutOrder` và `showFroyoOrder`. Bạn có thể tùy chọn **Code > Reformat Code** để định dạng lại mã XML. Jetzt ist der Code wie folgt:

```
<Chế độ xem hình ảnh

    android:id="@+id/ice_cream"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="@dimen/margin_wide"
    android:layout_marginTop="@dimen/margin_wide"
    android:contentDescription="@string/ice_cream_sandwiches"
    android:onClick="showIceCreamOrder"
    app:layout_constraintStart_toStartOf="cha mẹ"
    app:layout_constraintTop_toBottomOf="@+id/bánh rán"
    app:srcCompat="@drawable/icecream_circle" />
```

```
<Chế độ xem hình ảnh  
    android:id="@+id/froyo"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="@dimen/margin_wide"  
    android:layout_marginTop="@dimen/margin_wide"  
    android:contentDescription="@string/froyo"  
    android:onClick="showFroyoOrder"  
    app:layout_constraintStart_toStartOf="cha mẹ"  
    ứng dụng:layout_constraintTop_toBottomOf="@+id/ice_cream"  
    app:srcCompat="@drawable/froyo_circle" />
```

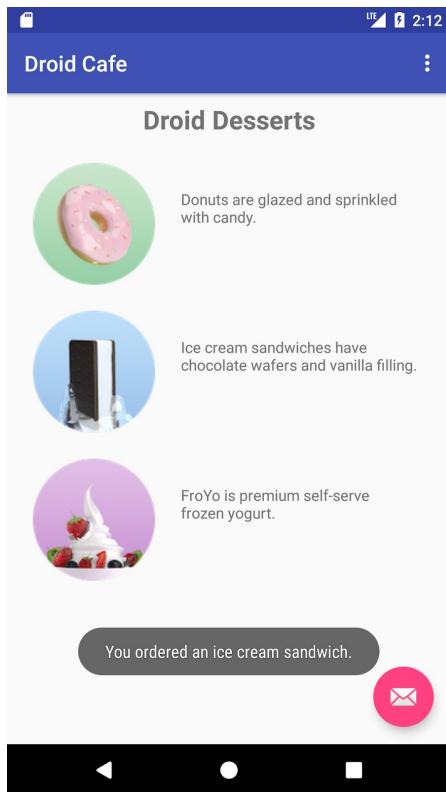
Lưu ý rằng thuộc tính `android:layout_marginStart` trong mỗi `ImageView` được gạch chân bằng màu đỏ. Thuộc tính này xác định lề "bắt đầu" cho `ImageView`, ở phía bên trái đối với hầu hết các ngôn ngữ nhưng ở phía bên phải đối với các ngôn ngữ đọc từ phải sang trái (RTL).

5. Nhấp vào phần mở đầu `android:` của thuộc tính `android:layout_marginStart` và một cảnh báo bóng đèn màu đỏ xuất hiện bên cạnh nó, như thể hiện trong hình bên dưới.

```
<ImageView  
    android:id="@+id/ice_cream"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="@dimen/margin_wide"  
    android:layout_marginTop="@dimen/margin_wide"  
    android:contentDescription="@string/ice_cream_sandwiches"  
    android:onClick="showIceCreamOrder"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/donut"  
    app:srcCompat="@drawable/icecream_circle" />
```

6. Để làm cho ứng dụng của bạn tương thích với các phiên bản Android trước, hãy nhấp vào bóng đèn màu đỏ cho mỗi phiên bản thuộc tính này và chọn Set layout_marginLeft... để đặt layout_marginLeft đến "@dimen / margin_wide".
7. Chạy ứng dụng.

Nhấp vào hình ảnh bánh rán, bánh mì kẹp kem hoặc froyo sẽ hiển thị thông báo Toast về đơn đặt hàng, như thể hiện trong hình bên dưới.



Mã giải pháp nhiệm vụ 2

Mã giải pháp cho tác vụ này được bao gồm trong mã và bộ cục cho MainActivity trong dự án Android Studio [DroidCafe](#).

Nhiệm vụ 3: Thay đổi nút hành động nổi

Khi bạn nhấp vào nút hành động nổi với biểu tượng email xuất hiện ở cuối màn hình, mã trong MainActivity sẽ hiển thị một thông báo ngắn gọn trong ngăn kéo mở từ cuối màn hình trên điện thoại thông minh hoặc từ góc dưới bên trái trên các thiết bị lớn hơn, sau đó đóng lại sau vài giây. Đây được gọi là thanh navbar. Nó được sử dụng để cung cấp phản hồi về một hoạt động. Để biết thêm thông tin, hãy xem [Snavbar](#).

Hãy xem cách các ứng dụng khác triển khai nút hành động nổi. Ví dụ: ứng dụng Gmail cung cấp nút hành động nổi để tạo email mới và ứng dụng Danh bạ cung cấp nút để tạo liên hệ mới. Để biết thêm thông tin về các nút hành động nổi, hãy xem [FloatingActionButton](#).

Đối với tác vụ này, bạn thay đổi biểu tượng cho FloatingActionButton thành giỏ hàng  và thay đổi hành động cho FloatingActionButton để khởi chạy Activity mới.

3.1 Thêm biểu tượng mới

Như bạn đã học trong một bài học khác, bạn có thể chọn một biểu tượng từ tập hợp các biểu tượng trong Android Studio. Làm theo các bước sau:

1. Mở rộng **res** trong **Project > ngăn Android** và nhấp chuột phải (hoặc Control và nhấp chuột) thư mục có thể **drawable**.
2. Chọn **New > Image Asset**. Hộp thoại Định cấu hình nội dung hình ảnh xuất hiện.
3. Chọn **Thanh ction và Biểu tượng tab** trong menu thả xuống ở đầu hộp thoại. (Lưu ý rằng **Thanh ction** cũng giống như **thanh PP**.)
4. Thay đổi **ic_action_name** trong trường **Name** thành **ic_shopping_cart**.
5. Nhấp vào hình ảnh clip art (logo Android bên cạnh **C lipart**) để chọn hình ảnh clip art làm biểu tượng. Một trang biểu tượng xuất hiện. Nhấp vào biểu tượng bạn muốn sử dụng cho nút thao tác nổi, chẳng hạn như biểu tượng giỏ hàng.

6. Chọn **H OLO_DARK** từ menu thả xuống **T heme**. Thao tác này đặt biểu tượng thành màu trắng trên nền tối (hoặc đen). Nhấp vào **N ext**.
7. Nhấp vào **F inish** trong hộp thoại Confirm Icon Path.

Mẹo: Để biết mô tả đầy đủ về cách thêm biểu tượng, hãy xem bài viết [C biểu tượng ứng dụng bằng Image Asset Studio](#).

3.2 Thêm một hoạt động

Như bạn đã học trong bài học trước, Activity đại diện cho một màn hình duy nhất trong ứng dụng của bạn, trong đó người dùng của bạn có thể thực hiện một tác vụ tập trung duy nhất. Bạn đã có một hoạt động, MainActivity.java.

Bây giờ bạn thêm một hoạt động khác có tên là OrderActivity.java.

1. Nhấp chuột phải (hoặc Control khi nhấp vào) thư mục **com.example.android.droidcafe** ở cột bên trái và chọn **New > Activity > Empty Activity**.
2. Chỉnh sửa **Tên A thành OrderActivity** và **Tên L thành activity_order**. Để yên các tùy chọn khác và nhấp vào **Finish**.

Lớp OrderActivity bây giờ sẽ được liệt kê cùng với MainActivity trong thư mục **java** và bây giờ activity_order.xml sẽ được liệt kê trong thư mục **layout**. Mẫu Hoạt động trống đã thêm các tệp này.

3.3 Thay đổi hành động

Trong bước này, bạn thay đổi hành động cho FloatingActionButton để khởi chạy Activity mới.

1. Mở **MainActivity**.
2. Thay đổi phương thức `onCreate(Bundle savedInstanceState)` để tạo ý định rõ ràng để khởi động OrderActivity:

```
public void onClick(Xem chế độ xem) {  
    Ý định = ý định mới(MainActivity.this, OrderActivity.class);  
    startActivityForResult(Ý định);  
}
```

3. Chạy ứng dụng. Chạm vào nút hành động nổi hiện sử dụng biểu tượng giờ hàng. Một Hoạt động trống sẽ xuất hiện (OrderActivity). Nhấn vào nút Quay lại để quay lại MainActivity.



Mã giải pháp nhiệm vụ 3

Mã giải pháp cho tác vụ này được bao gồm trong mã và bộ cục cho dự án Android Studio [DroidCafe](#).

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 này.

Thử thách: M ainActivity của ứng dụng DroidCafe ra mắt một ứng dụng thứ hai có tên là OrderActivity. Bạn đã học được trong một bài học khác cách gửi dữ liệu từ một ctivity A sang một ctivity A khác. Thay đổi ứng dụng để gửi thông báo đặt hàng cho món tráng miệng đã chọn trong M ainActivity đến T extView mới ở đầu bối cục OrderActivity.

1. Thêm T extView ở đầu bối cục OrderActivity với id order_textview.
2. Tạo một biến thành viên (m OrderMessage) trong M ainActivity cho thông báo đơn hàng xuất hiện trong T oast.
3. Thay đổi các trình xử lý nhấp chuột s howDonutOrder(), s howIceCreamOrder() và s howFroyoOrder() để gán chuỗi tin nhắn m OrderMessage trước khi hiển thị T oast. Ví dụ: sau đây gán chuỗi d onut_order_message cho m OrderMessage và hiển thị T oast:

```
mOrderMessage = getString(R.string.donut_order_message);  
displayToast(mOrderMessage);
```

4. Thêm một p ublic s tatic f inal S tring được gọi là E XTRA_MESSAGE vào đầu M ainActivity để xác định khóa cho một i ntent.putExtra:

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

5. Thay đổi phương thức o nClick() để bao gồm câu lệnh i ntent.putExtra trước khi khởi chạy Hoạt động đặt hàng:

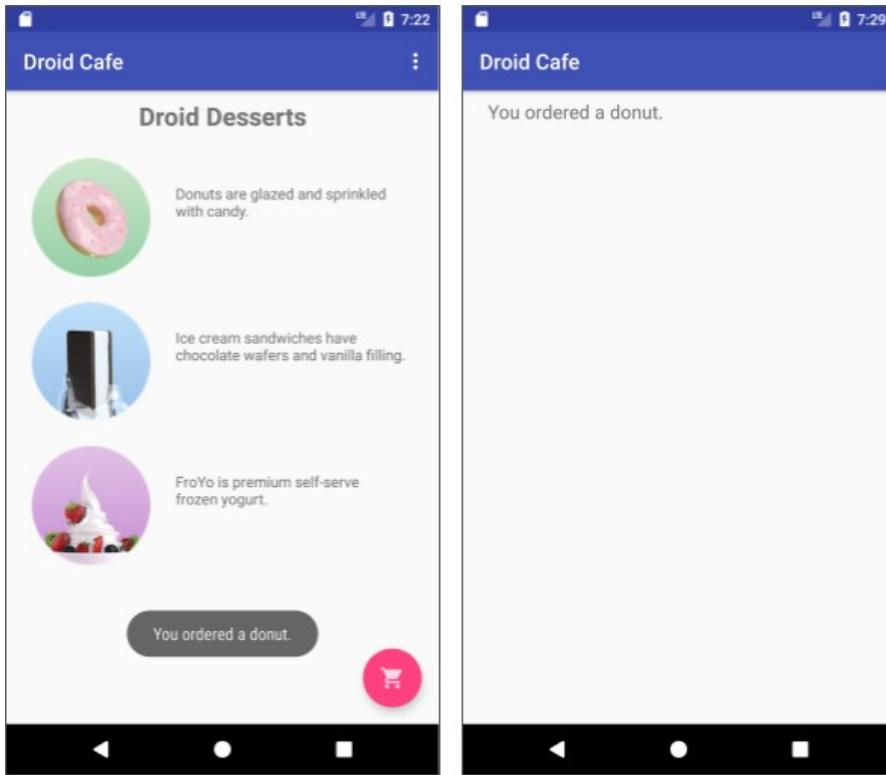
```
public void onClick(Xem chế độ xem) {  
    Ý định =  
        ý định mới(MainActivity.this, OrderActivity.class);  
    intent.putExtra(EXTRA_MESSAGE, mOrderMessage);
```

```
    startActivity(y định);
}
```

6. Trong OrderActivity, thêm mã sau vào phương thức `onCreate()` để lấy Intent đã khởi chạy Activity, trích xuất thông báo chuỗi và thay thế văn bản trong TextView bằng thông báo:

```
Ý định = getIntent();
Thông báo chuỗi = "Đơn đặt hàng: " +
                    intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
TextView textView = findViewById(R.id.order_textview); textView.setText(tin nhắn);
```

7. Chạy ứng dụng. Sau khi chọn hình ảnh món tráng miệng, hãy nhấn vào nút hành động nổi để khởi chạy OrderActivity, nút này sẽ bao gồm thông báo đơn hàng như trong hình bên dưới.



Mã giải pháp thách thức

Dự án Android Studio: [DroidCafeChallenge](#)

Tóm tắt

- Để sử dụng hình ảnh trong dự án, hãy sao chép hình ảnh vào thư mục `drawable` của dự án (`project_name` ứng dụng > > `src` > `main` > `res` > `drawable`) .
- Xác định `ImageView` để sử dụng nó bằng cách kéo `ImageView` vào bố cục và chọn hình ảnh cho nó.
- Thêm thuộc tính `android:onClick` để làm cho `ImageView` có thể nhấp được giống như một nút. Chỉ định tên của trình xử lý nhấp chuột.
- Tạo một trình xử lý nhấp chuột trong `Activity` để thực hiện hành động.
- Chọn một biểu tượng: Mở rộng `res` trong `ngân Project > Android`, nhấp chuột phải (hoặc Control khi nhấp vào) thư mục có thể vẽ và chọn `New > Image Asset`. Chọn **Biểu tượng thanh** và **tab** trong menu thả xuống và nhấp vào hình ảnh clip art (logo Android bên cạnh **Clipart:**) để chọn hình ảnh clip art làm biểu tượng.

- Thêm một cái khác Một ctivity: Trong Project > Android , nhấp chuột phải (hoặc giữ Control khi bấm) thư mục tên gói trong ja và và chọn **Hoạt động** > và một mẫu cho Một ctivity (chẳng hạn như **Hoạt động E mpty**) . • Hiển thị một T oast thông điệp:

```
Toast.makeText(getApplicationContext(), thông báo,  
Toast.LENGTH_SHORT).show();
```

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [4.1: Nút và hình ảnh có thể nhấp vào.](#)

Tìm hiểu thêm

Tài liệu Android Studio:

- [Hướng dẫn sử dụng Android Studio](#)
- [Tạo biểu tượng ứng dụng bằng](#) tài liệu dành cho nhà phát triển Image Asset Studio dành cho nhà phát triển:
 - [Giao diện người dùng & điều hướng](#)
 - [Xây dựng giao diện người dùng bằng Layout Editor](#)
 - [Xây dựng giao diện người dùng đáp ứng với ConstraintLayout](#)
 - [Bố trí](#)
 - [Cảnh](#)
 - [Nút](#)
 - [Hình ảnhView](#)
 - [Chế độ xem văn bản](#)
 - [Nút](#)
 - [Phong cách và chủ đề](#)

Khác:

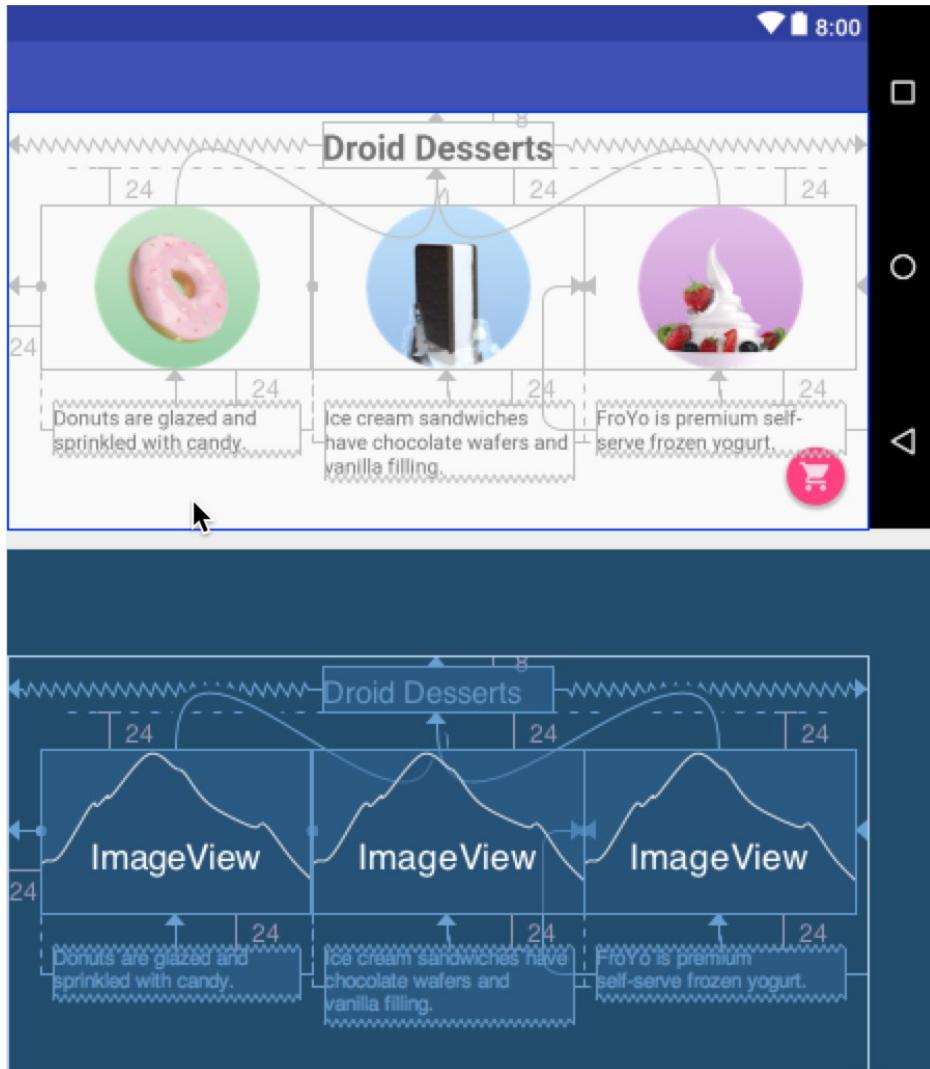
- Lớp học lập trình: [Using ConstraintLayout để thiết kế chế độ xem Android của bạn](#)

Homework

Thay đổi ứng dụng

Ứng [dụng DroidCafe](#) trông ổn khi thiết bị hoặc trình giả lập được định hướng theo chiều dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình mô phỏng sang hướng ngang, hình ảnh thứ hai và thứ ba sẽ không xuất hiện.

1. Mở (hoặc tải xuống) dự án [Án Ứng dụng DroidCafe](#).
2. Tạo biến thể bố cục cho hướng ngang: content_main.xml (đất).
3. Loại bỏ các ràng buộc khỏi ba hình ảnh và ba mô tả văn bản.
4. Chọn cả ba hình ảnh trong biến thể bố cục và chọn **Expand Theo chiều ngang** trong Đóng gói để  phân bổ đều hình ảnh trên màn hình như trong hình bên dưới.
5. Hạn chế mô tả văn bản ở hai bên và dưới cùng của hình ảnh như được hiển thị trong hình dưới đây.



Trả lời những câu hỏi này

Câu hỏi 1

Làm cách nào để thêm hình ảnh vào dự án Android Studio? Chọn một:

- Kéo từng hình ảnh vào trình chỉnh sửa bố cục.

- Sao chép các tệp hình ảnh vào thư mục `drawable` của dự án của bạn.
- Kéo `ImageButton` vào trình chỉnh sửa bố cục.
- Chọn `New > Image Asset` và sau đó chọn tệp hình ảnh.

Câu hỏi 2

Làm thế nào để bạn làm cho một `ImageView` có thể nhấp được như một `Button` đơn giản? Chọn một:

- Thêm thuộc tính `android:contentDescription` vào `ImageView` trong bố cục và sử dụng nó để gọi trình xử lý nhấp chuột trong `Activity A`.
- Thêm thuộc tính `android:src` vào `ImageView` trong bố cục và sử dụng nó để gọi trình xử lý nhấp chuột trong `Activity A`.
- Thêm thuộc tính `android:onClick` vào `ImageView` trong bố cục và sử dụng nó để gọi trình xử lý nhấp chuột trong `Activity A`.
- Thêm thuộc tính `android:id` vào `ImageView` trong bố cục và sử dụng nó để gọi trình xử lý nhấp chuột trong `Activity A`.

Câu hỏi 3

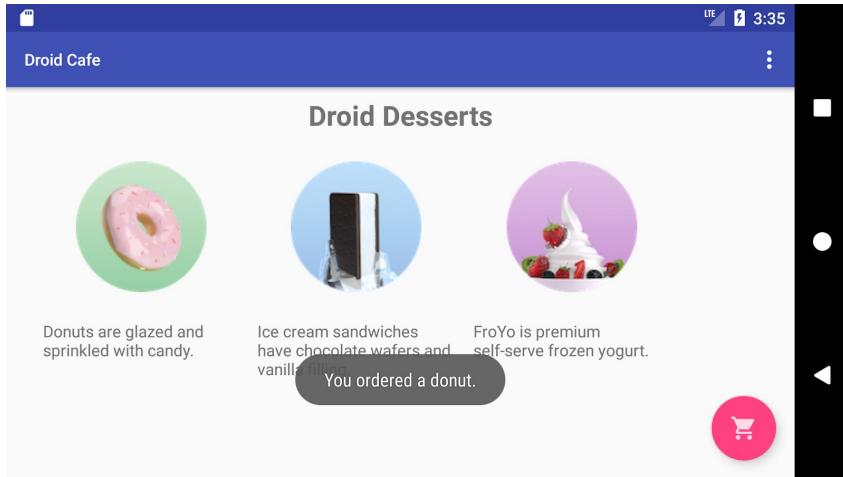
Quy tắc nào áp dụng cho trình xử lý nhấp chuột được gọi từ thuộc tính trong bố cục? Chọn một:

- Phương thức xử lý nhấp chuột phải bao gồm trình nghe sự kiện `View`. `OnClickListener`, là một giao diện trong lớp `View`.
- Phương thức xử lý nhấp chuột phải là `public`, trả về `void` và xác định `View` làm tham số duy nhất của nó.
- Trình xử lý nhấp chuột phải tùy chỉnh `View`. `OnClickListener` và ghi đè trình xử lý nhấp chuột của nó để thực hiện một số hành động.
- Phương thức xử lý nhấp chuột phải là `private` và trả về `View`.

Gửi ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

1. Chạy ứng dụng.
2. Chuyển sang hướng ngang để xem biến thể bố cục mới. Nó sẽ trông giống như hình bên dưới.



Bài 4.2: Điều khiển đầu vào

Giới thiệu

Để cho phép người dùng nhập văn bản hoặc số, bạn sử dụng phần [tử EditText](#). Một số điều khiển đầu vào là thuộc tính EditText xác định loại bàn phím xuất hiện, để giúp người dùng nhập dữ liệu dễ dàng hơn. Ví dụ: bạn có thể chọn phone cho thuộc tính [ndroid:inputType](#) để hiển thị bàn phím số thay vì bàn phím chữ và số.

Các điều khiển đầu vào khác giúp người dùng dễ dàng đưa ra lựa chọn. Ví dụ: [các phần tử RadioButton](#) cho phép người dùng chọn một (và chỉ một) mục từ một tập hợp các mục.

Trong thực tế này, bạn sử dụng các thuộc tính để kiểm soát giao diện bàn phím ảo và để thiết lập loại nhập dữ liệu cho EditText. Bạn cũng thêm các nút radio vào ứng dụng DroidCafe để người dùng có thể chọn một mục từ một tập hợp các mục.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo một dự án Android Studio từ một mẫu và tạo bố cục chính.
- Chạy ứng dụng trên trình mô phỏng hoặc thiết bị được kết nối.
- Tạo và chỉnh sửa các thành phần giao diện người dùng bằng trình chỉnh sửa bố cục và mã XML.
- Truy cập các thành phần giao diện người dùng từ mã của bạn bằng `findViewById()`.
- Chuyển đổi văn bản trong View thành một chuỗi bằng cách sử dụng `getText()`.
- Tạo trình xử lý nhập chuột cho nhập chuột B.
- Hiển thị thông báo Toast.

Những gì bạn sẽ học

- Cách thay đổi phương thức nhập để bật đề xuất, tự động viết hoa và xáo trộn mật khẩu.
- Cách thay đổi bàn phím ảo chung thành bàn phím điện thoại hoặc bàn phím chuyên dụng khác.
- Cách thêm các nút radio để người dùng chọn một mục từ một tập hợp các mục.
- Cách thêm một con quay để hiển thị menu thả xuống với các giá trị, từ đó người dùng có thể chọn một giá trị.

Bạn sẽ làm gì

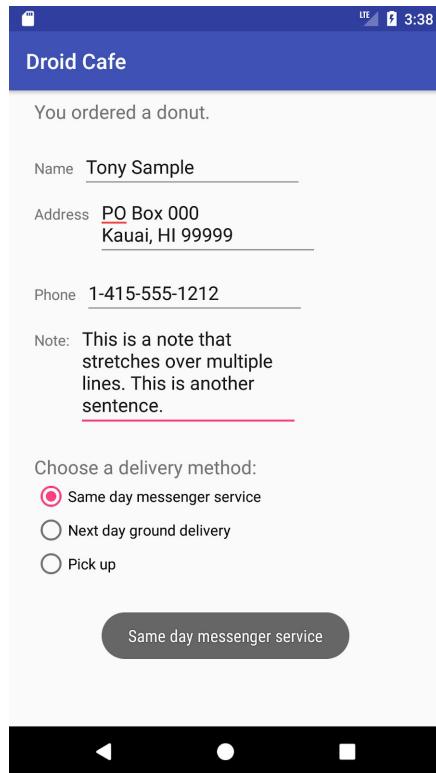
- Hiển thị bàn phím để nhập địa chỉ email.
- Hiển thị bàn phím số để nhập số điện thoại.
- Cho phép nhập văn bản nhiều dòng với cách viết hoa câu tự động.
- Thêm các nút radio để chọn một tùy chọn.
- Đặt trình xử lý `onClickListener` cho các nút radio.
- Thêm một con quay cho trường số điện thoại để chọn một giá trị từ một tập hợp các giá trị.

Tổng quan về ứng dụng

Trong thực tế này, bạn thêm nhiều tính năng hơn vào ứng dụng DroidCafe từ bài học về cách sử dụng hình ảnh có thể nhấp vào.

Trong `OrderActivity` của ứng dụng, bạn thử nghiệm với `EditText` có `android:inputType="textPersonName"`. Bạn thêm các phần tử `EditText` cho tên và địa chỉ của một người, đồng thời sử dụng các thuộc tính để xác định các phần tử một dòng và nhiều dòng đưa ra gợi ý khi bạn nhập văn bản. Bạn cũng thêm `EditText` hiển thị bàn phím số để nhập số điện thoại.

Các loại điều khiển đầu vào khác bao gồm các yếu tố tương tác cung cấp các lựa chọn của người dùng. Bạn thêm các nút radio vào DroidCafe để chỉ chọn một tùy chọn giao hàng từ một số tùy chọn. Bạn cũng cung cấp một điều khiển đầu vào con quay để chọn nhãn (Home, Work, Other, Custom) cho số điện thoại.



Nhiệm vụ 1: Thử nghiệm với các thuộc tính nhập văn bản

Chạm vào trường văn bản có thể chỉnh sửa EditText sẽ đặt con trỏ vào trường văn bản và tự động hiển thị bàn phím ảo để người dùng có thể nhập văn bản.

Trường văn bản có thể chỉnh sửa mong đợi một loại đầu vào văn bản nhất định, chẳng hạn như văn bản thuần túy, địa chỉ email, số điện thoại hoặc mật khẩu. Điều quan trọng là phải chỉ định loại nhập cho từng trường văn bản trong ứng dụng của bạn để hệ thống hiển thị phương thức nhập mềm thích hợp, chẳng hạn như bàn phím ảo cho văn bản thuần túy hoặc bàn phím số để nhập số điện thoại.

1.1 Thêm EditText để nhập tên

Trong bước này, bạn thêm TextView và EditText vào bố cục OrderActivity trong ứng dụng DroidCafe để người dùng có thể nhập tên của một người.

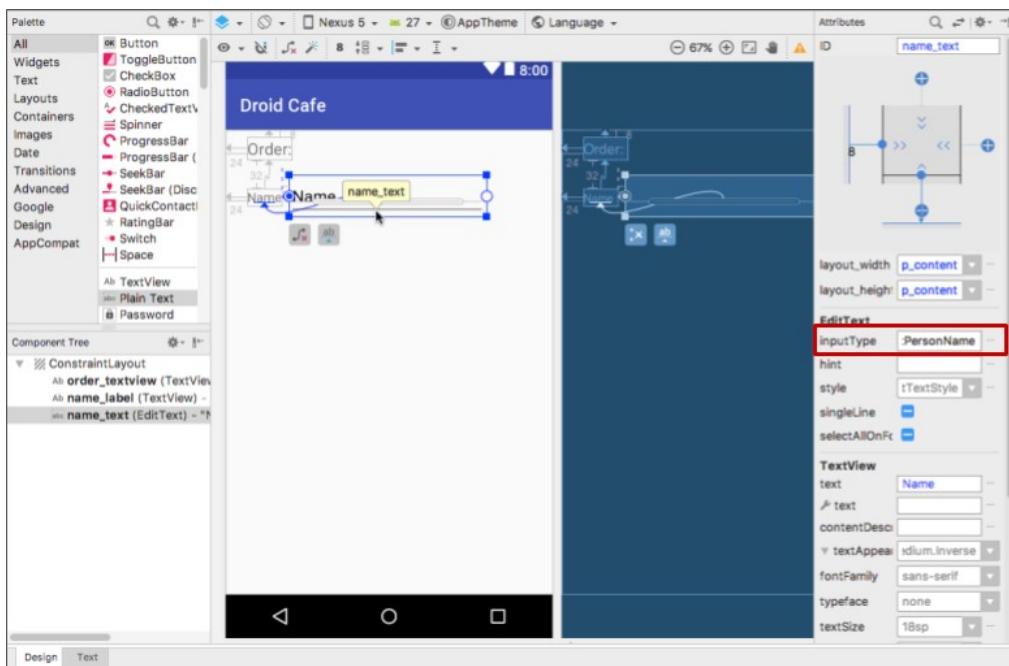
1. Tạo một bản sao của ứng dụng DroidCafe từ bài học về cách sử dụng hình ảnh có thể nhấp và đổi tên bản sao thành **DroidCafeInput**. Nếu bạn chưa hoàn thành thử thách viết mã trong bài học đó, hãy tải xuống [dự án DroidCafeChallenge](#) và đổi tên thành **DroidCafeInput**.
2. Mở tệp bố cục **activity_order.xml**, tệp này sử dụng ConstraintLayout.
3. Thêm TextView vào ConstraintLayout trong một **activity_order.xml** dưới phần tử **order_textview** đã có trong bố cục. Sử dụng các thuộc tính sau cho

Chế độ xem văn bản:

Thuộc tính TextView	Giá trị
android:id	"@+id/name_label"
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"
Android:layout_marginStart	"24dp"
Android:layout_marginLeft	"24dp"
Android:layout_marginTop	"32 điểm"
android:văn bản	"Tên"
Ứng dụng:layout_constraintStart_toStartOf	"cha mẹ"

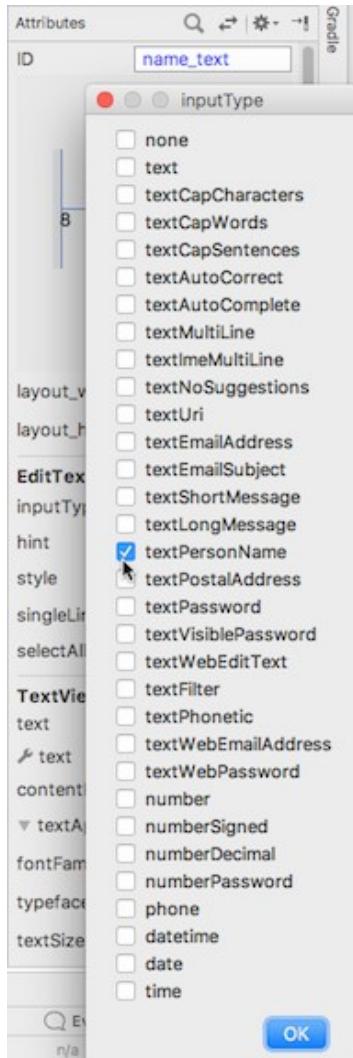
Ứng dụng: layout_constraintTop_toBottomOf	"@+id/order_textview"
---	-----------------------

4. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:text để tạo và nhập cho nó được gọi là name_label_text trong strings.xml.
5. Thêm phần tử EditText. Để sử dụng trình soạn thảo bố cục trực quan, hãy kéo phần tử **Văn bản Plain** từ ngăn **Palette** đến vị trí bên cạnh name_label TextView. Sau đó, nhập **name_text** cho trường ID và hạn chế phía bên trái và đường cơ sở của phần tử với phần tử name_label bên phải và đường cơ sở như trong hình bên dưới:



6. Hình trên làm nổi bật trường **inputType** trong ngăn **Attributes** để cho thấy rằng Android Studio tự động gán loại textPersonName. Nhập vào trường **inputType** để xem

Menu các loại đầu vào:



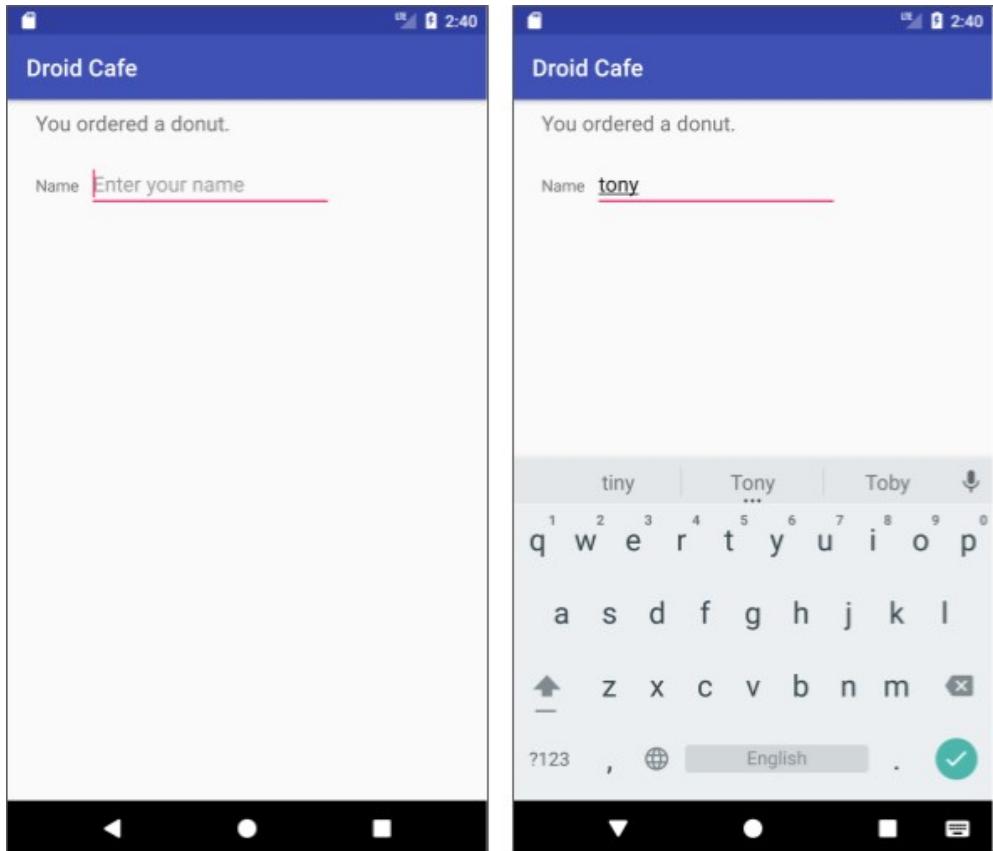
Trong hình trên, **t extPersonName** được chọn làm loại đầu vào.

7. Thêm gợi ý cho mục nhập văn bản, chẳng hạn như **E nter tên của bạn**, trong trường **h int** trong ngăn **A ttributes** và xóa **mục nhập N ame** trong trường **t ext**. Như một gợi ý cho người dùng, văn bản "Nhập tên của bạn" nên được làm mờ bên trong **E ditText**.
8. Kiểm tra mã XML cho bố cục bằng cách nhấp vào tab **T ext**. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính **android:hint** thành **e nter_name_hint**. Bạn nên đặt các thuộc tính sau cho **E ditText** mới (thêm thuộc tính **l ayout_marginLeft** để tương thích với các phiên bản Android cũ hơn):

Thuộc tính EditText	Giá trị
android:id	"@+id/name_text"
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"
Android:layout_marginStart	8 điểm
Android:layout_marginLeft	8 điểm
Android:ems	"10"
android:hint	"@string/enter_name_hint"
android:inputType	"textPersonName"
Ứng dụng:layout_constraintBaseline_toBaselineOf	"@+id/name_label"
Ứng dụng:layout_constraintStart_toEndOf	"@+id/name_label"

Như bạn có thể thấy trong mã XML, thuộc tính `android:inputType` được đặt thành `textPersonName`.

- Chạy ứng dụng. Nhấn vào hình ảnh bánh rán trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem Activity tiếp theo. Nhấn vào bên trong trường nhập văn bản để hiển thị bàn phím và nhập văn bản, như trong hình bên dưới.



Lưu

ý rằng các đề xuất sẽ tự động xuất hiện cho các từ mà bạn nhập. Nhấn vào một đề xuất để sử dụng nó. Đây là một trong những thuộc tính của giá trị `textPersonName` [cho thuộc tính android:inputType](#). Thuộc tính `inputType` kiểm soát nhiều tính năng khác nhau, bao gồm bố cục bàn phím, viết hoa và nhiều dòng văn bản.

10. Để đóng bàn phím, hãy nhấn vào biểu tượng dấu kiểm trong vòng tròn màu xanh lá cây , xuất hiện ở góc dưới bên phải của bàn phím. Đây được gọi là **phím D một**.

1.2 Thêm EditText nhiều dòng

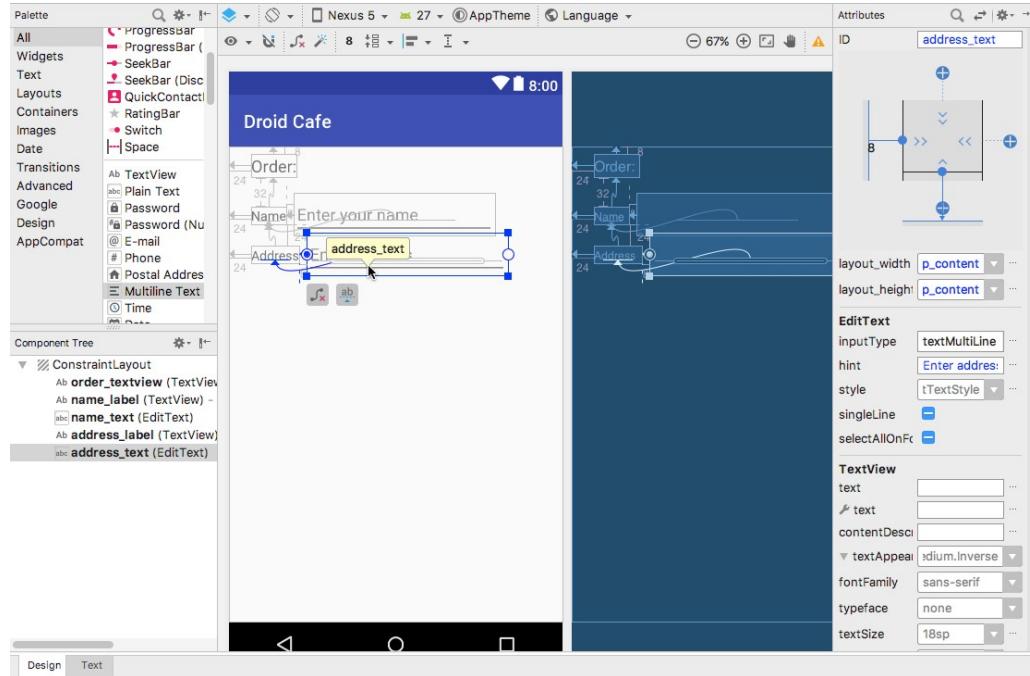
Trong bước này, bạn thêm một `EditText` khác vào bố cục `OrderActivity` trong ứng dụng `DroidCafe` để người dùng có thể nhập địa chỉ bằng nhiều dòng.

1. Mở tệp bố cục ctivity_order.xml nếu nó chưa được mở.
2. Thêm một TextView dưới phần tử name_label đã có trong bố cục. Sử dụng các thuộc tính sau cho TextView mới:

Thuộc tính TextView	Giá trị
android:id	"@+id/address_label"
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"
Android:layout_marginStart	"24dp"
Android:layout_marginLeft	"24dp"
Android:layout_marginTop	"24dp"
android:văn bản	"Địa chỉ"
Ứng dụng:layout_constraintStart_toStartOf	"cha mẹ"
Ứng dụng:layout_constraintTop_toBottomOf	"@+id/name_label"

3. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:text để tạo và nhập cho nó được gọi là address_label_text trong strings.xml.
4. Thêm phần tử EditText. Để sử dụng trình soạn thảo bố cục trực quan, hãy kéo phần tử **M ultiline Text** từ ngăn **P alette** đến vị trí bên cạnh address_label TextView. Sau đó nhập address_text cho trường ID và

hạn chế bên trái và đường cơ sở của phần tử vào phần tử address_label bên phải và đường cơ sở như trong hình bên dưới:



5. Thêm gợi ý cho nhập văn bản, chẳng hạn như **địa chỉ Enter**, trong trường **hint** trong ngăn **Attributes**. Như một gợi ý cho người dùng, văn bản "Nhập địa chỉ" nên được làm mờ bên trong **EditText**.
6. 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 **e nter_address_hint**. Bạn nên đặt các thuộc tính sau cho **EditText** mới (thêm thuộc tính **layout_marginLeft** để tương thích với các phiên bản Android cũ hơn):

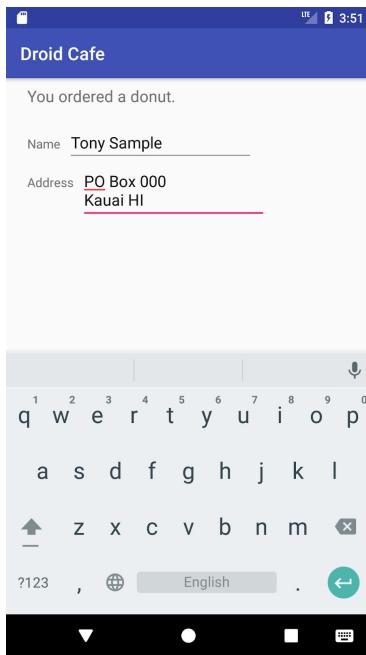
Thuộc tính EditText	Giá trị
android:id	"@+id/address_text"
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"

Android:layout_marginStart	8 điểm
Android:layout_marginLeft	8 điểm
Android:EMS	"10"
android:gợi ý	"@string/enter_address_hint"
android:loại đầu vào	"văn bảnNhiều dòng"
Ứng dụng:layout_constraintBaseline_toBaselineOf	"@+id/address_label"
Ứng dụng:layout_constraintStart_toEndOf	"@+id/address_label"

7. Chạy ứng dụng. Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem Activity tiếp theo.
8. Nhấn vào bên trong trường nhập văn bản "Địa chỉ" để hiển thị bàn phím và nhập văn bản, như được hiển thị trong



hình bên dưới, sử dụng phím Return ở góc dưới bên phải của bàn phím (còn được gọi là phím Enter hoặc New Line) để bắt đầu một dòng văn bản mới. Phím Return xuất hiện nếu bạn đặt giá trị `textMultiLine` cho thuộc tính `android:inputType`.



- Để đóng bàn phím, hãy chạm vào nút mũi tên xuống xuất hiện thay vì nút Quay lại ở hàng dưới cùng của các nút.

1.3 Sử dụng bàn phím cho số điện thoại

Trong bước này, bạn thêm một EditText khác vào bố cục OrderActivity trong ứng dụng DroidCafe để người dùng có thể nhập số điện thoại trên bàn phím số.

- Mở tệp bố cục activity_order.xml nếu nó chưa được mở.
- Thêm TextView bên dưới phần tử address_label đã có trong bố cục. Sử dụng các thuộc tính sau cho TextView mới:

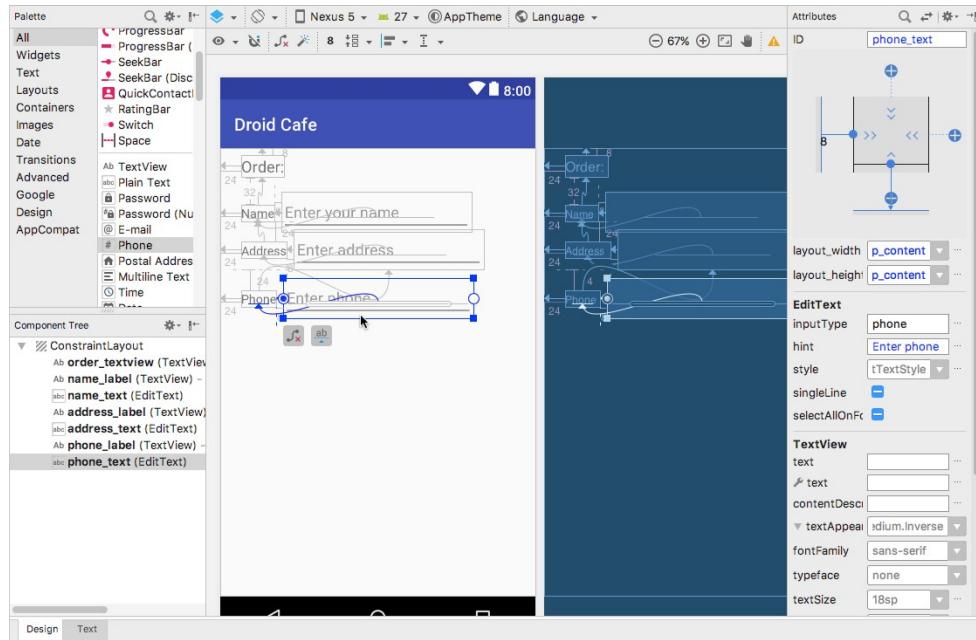
Thuộc tính TextView	Giá trị
---------------------	---------

android:id	"@+id/phone_label"
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"
Android:layout_marginStart	"24dp"
Android:layout_marginLeft	"24dp"
Android:layout_marginTop	"24dp"
android:văn bản	"Điện thoại"
Ứng dụng:layout_constraintStart_toStartOf	"cha mẹ"
Ứng dụng:layout_constraintTop_toBottomOf	"@+id/address_text"

Lưu ý rằng TextView này bị giới hạn ở cuối EditText nhiều dòng

(một address_text). Điều này là do một address_text có thể phát triển đến nhiều dòng và TextView này sẽ xuất hiện bên dưới nó.

3. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:text để tạo và nhập cho nó được gọi là phone_label_text trong strings.xml.
4. Thêm phần tử EditText. Để sử dụng trình chỉnh sửa bố cục trực quan, hãy kéo phần tử Phone từ ngăn **Bảng màu** đến vị trí bên cạnh phone_label TextView. Sau đó, nhập **phone_text** cho trường ID và hạn chế phía bên trái và đường cơ sở của phần tử vào phần tử phone_label phía bên phải và đường cơ sở như trong hình bên dưới:

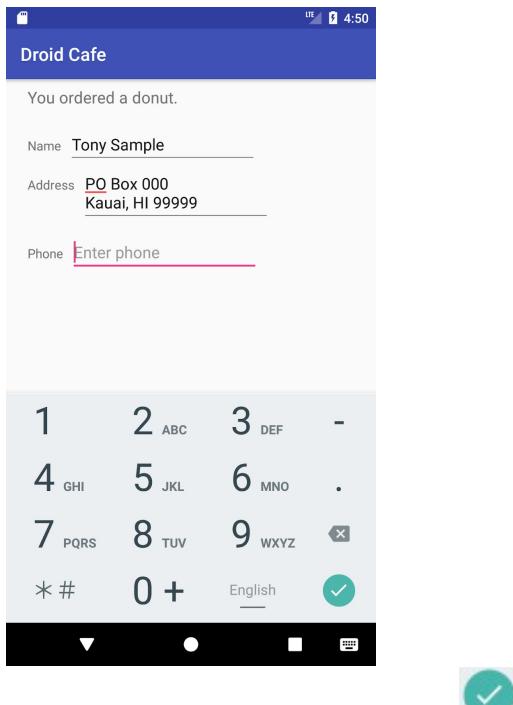


5. Thêm gợi ý cho nhập văn bản, chẳng hạn như `Enter phone`, trong trường `hint` trong ngăn `Attributes`. Như một gợi ý cho người dùng, văn bản "Enter phone" nên được làm mờ bên trong `EditText`.
6. Kiểm tra mã XML cho bố cục bằng cách nhấp vào tab `Text`. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính `android:hint` vào `enter_phone_hint`. Bạn nên đặt các thuộc tính sau cho `EditText` mới (thêm thuộc tính `layout_marginStart` để tương thích với các phiên bản Android cũ hơn):

Thuộc tính <code>EditText</code>	Giá trị
<code>android:id</code>	"@+id/phone_text"
<code>Android:layout_width</code>	"wrap_content"
<code>Android:layout_height</code>	"wrap_content"
<code>Android:layout_marginStart</code>	8 điểm

Android:layout_marginLeft	8 điểm
Android:EMS	"10"
android:gợi ý	"@string/enter_phone_hint"
android:loại đầu vào	"điện thoại"
Ứng dụng:layout_constraintBaseline_toBaselineOf	"@+id/phone_label"
Ứng dụng:layout_constraintStart_toEndOf	"@+id/phone_label"

7. Chạy ứng dụng. Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem Activity tiếp theo.
8. Nhấn vào bên trong trường "Điện thoại" để hiển thị bàn phím số. Sau đó, bạn có thể nhập số điện thoại, như trong hình bên dưới.



9. Để đóng bàn phím, hãy nhấn vào phím **D** **một**.



Để thử nghiệm với các giá trị thuộc tính `ndroid:inputType`, hãy thay đổi các giá trị `android:inputType` của phần tử `EditText` thành giá trị sau để xem kết quả:

- `textEmailAddress`: Nhấn vào trường sẽ hiển thị bàn phím email có biểu tượng @ nằm gần phím cách.
- `mật khẩu văn bản`: `Char` Các tác động mà người dùng nhập sẽ biến thành các dấu chấm để che giấu mật khẩu đã nhập.

1.4 Kết hợp các loại đầu vào 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 thuộc tính `textMultiLine` và `textCapSentences` cho nhiều dòng văn bản trong đó mỗi câu bắt đầu bằng chữ in hoa.

1. Mở tệp bố cục ctivity_order.xml nếu nó chưa được mở.
2. Thêm TextView dưới phần tử phone_label đã có trong bố cục. Sử dụng các thuộc tính sau cho TextView mới:

Thuộc tính TextView	Giá trị
android:id	"@+id/note_label"
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"
Android:layout_marginStart	"24dp"
Android:layout_marginLeft	"24dp"
Android:layout_marginTop	"24dp"
android:văn bản	"Ghi chú"
Ứng dụng:layout_constraintStart_toStartOf	"cha mẹ"
Ứng dụng:layout_constraintTop_toBottomOf	"@+id/phone_label"

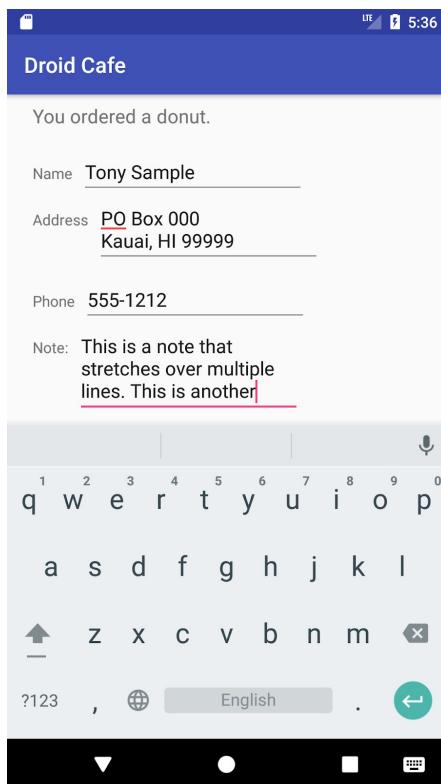
3. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:text để tạo và nhập cho nó được gọi là note_label_text trong strings.xml.
 4. Thêm phần tử EditText. Để sử dụng trình chỉnh sửa bố cục trực quan, hãy kéo phần tử **M ultiline Text** từ ngăn **P alette** đến vị trí bên cạnh n ote_label TextView. Sau đó, nhập n ote_text cho trường ID và hạn chế bên trái và đường cơ sở của phần tử vào n phần tử ote_label bên phải và đường cơ sở như bạn đã làm trước đây với các phần tử E ditText khác.
-

5. Thêm gợi ý cho nhập văn bản, chẳng hạn như **ghi chú E**, trong trường **hint** trong ngăn **Attributes**.
6. Nhấp vào bên trong **trường keyboardType** trong ngăn **Attributes**. Giá trị **textMultiLine** đã được chọn. Ngoài ra, hãy chọn **textCapSentences** để kết hợp các thuộc tính này.
7. 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** vào **enter_note_hint**. Bạn nên đặt các thuộc tính sau cho **EditText** mới (thêm thuộc tính **layout_marginLeft** để tương thích với các phiên bản Android cũ hơn):

Thuộc tính EditText	Giá trị
android:id	"@+id/note_text"
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"
Android:layout_marginStart	8 điểm
Android:layout_marginLeft	8 điểm
Android:EMS	"10"
android:gợi ý	"@string/enter_note_hint"
android:loại đầu vào	"textCapSentences textMultiLine"
Ứng dụng:layout_constraintBaseline_to Đường cơ sở	"@+id/note_label"
Ứng dụng:layout_constraintStart_toEnd của	"@+id/note_label"

Để kết hợp các giá trị cho thuộc tính `android:inputType`, hãy nối chúng bằng cách sử dụng đường ống (`|`) nhân vật.

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 bên trong trường "Ghi chú", nhập các câu hoàn chỉnh, như thể hiện trong hình bên dưới. Sử dụng phím Return để tạo một dòng mới hoặc chỉ cần gõ để ngắt câu trên nhiều dòng.



Nhiệm vụ 2: Sử dụng các nút radio

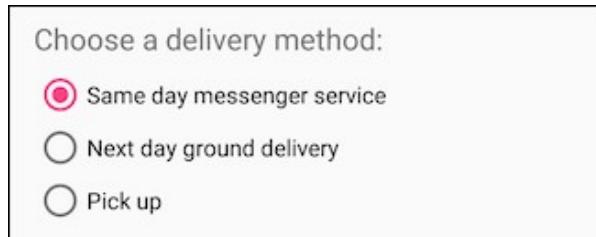
Điều khiển đầu vào là các phần tử tương tác trong giao diện người dùng của ứng dụng chấp nhận dữ liệu đầu vào. Các nút radio là các điều khiển đầu vào hữu ích để chỉ chọn một tùy chọn từ một tập hợp các tùy chọn.

Mẹo: Bạn nên sử dụng các nút radio nếu muốn người dùng xem song song tất cả các tùy chọn có sẵn. Nếu không cần thiết phải hiển thị tất cả các tùy chọn cạnh nhau, bạn có thể muốn sử dụng ginner [S](#) để thay thế, điều này được mô tả trong một chương khác.

Trong tác vụ này, bạn thêm một nhóm các nút radio vào ứng dụng DroidCafeInput để thiết lập các tùy chọn giao hàng cho đơn đặt hàng tráng miệng. Để biết tổng quan và thêm mã mẫu cho các nút radio, hãy xem Nút [Radio](#).

2.1 Thêm RadioGroup và các nút radio

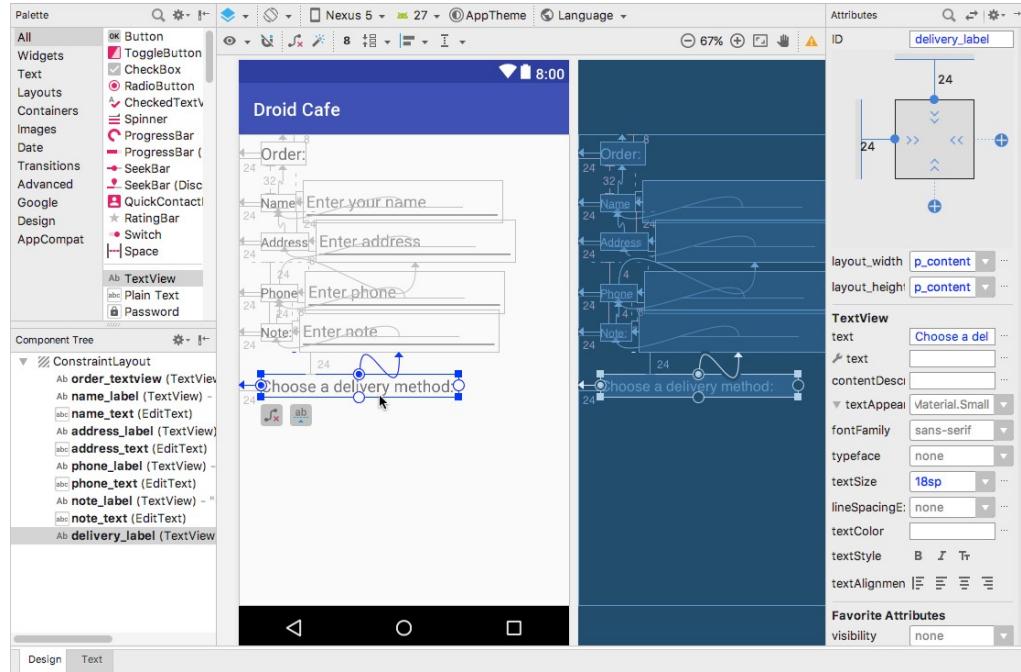
Để thêm các nút radio vào OrderActivity trong ứng dụng DroidCafeInput, bạn tạo các [phần tử RadioButton](#) trong tệp bố cục `activity_order.xml`. Sau khi chỉnh sửa tệp bố cục, bố cục cho các nút radio trong OrderActivity sẽ giống như hình bên dưới.



Vì các lựa chọn nút radio loại trừ lẫn nhau, bạn nhóm chúng lại với nhau bên trong [RadioGroup](#) (bằng tiếng Anh). Bằng cách nhóm chúng lại với nhau, hệ thống Android đảm bảo rằng chỉ có thể chọn một nút radio tại một thời điểm.

Lưu ý: Thứ tự bạn liệt kê các phần tử RadioButton xác định thứ tự chúng xuất hiện trên màn hình.

- Mở một ctivity_order.xml và thêm phần tử TextView bị ràng buộc ở dưới cùng của phần tử note_text đã có trong bố cục và vào lề trái, như trong hình bên dưới.



- Chuyển sang chỉnh sửa XML và đảm bảo rằng bạn đã đặt các thuộc tính sau cho TextView mới:

Thuộc tính TextView	Giá trị
android:id	"@+id/delivery_label"
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"
Android:layout_marginStart	"24dp"
Android:layout_marginLeft	"24dp"

Android:layout_marginTop	"24dp"
android:văn bản	"Chọn phương thức giao hàng: "
android:kích thước văn bản	"18sp"
Ứng dụng:layout_constraintStart_toStartOf	"cha mẹ"
ứng dụng:layout_constraintTop_toBottomOf	"@+id/note_text"

3. Trích xuất tài nguyên chuỗi cho "Chọn phương thức phân phối:" để choose_delivery_method.
4. Để thêm các nút radio, hãy đặt chúng trong RadioGroup. Thêm RadioGroup vào bố cục bên dưới TextView mà bạn vừa thêm, bao gồm ba phần tử [Radio Button](#) như được hiển thị trong mã XML bên dưới:

```
<Nhóm vô tuyến
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="24dp"
    android:layout_marginStart="24dp"
    android:orientation="dọc"
    app:layout_constraintStart_toStartOf="cha mẹ"
    app:layout_constraintTop_toBottomOf="@+id/delivery_label">
    <Nút radio
        android:id="@+id/cùng ngày"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="Dịch vụ nhắn tin trong ngày" />
    <Nút radio
        android:id="@+id/ngày hôm sau"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="Giao hàng mặt đất vào ngày hôm sau" />
    <Nút radio
        android:id="@+id/nhận hàng"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="Nhấy" />
</ Nhóm vô tuyến>
```

Mục "onRadioButtonClicked" cho thuộc tính android:onClick cho mỗi RadioButton sẽ được gạch chân 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 được dịch dễ dàng: same_day_messenger_service, next_day_ground_delivery và pick_up.

2.2 Thêm trình xử lý nhập chuột nút radio

Thuộc tính android:onClick cho mỗi phần tử nút radio chỉ định phương thức onRadioButtonClicked() để xử lý sự kiện click. Do đó, bạn cần thêm một phương thức onRadioButtonClicked() mới vào lớp OrderActivity.

1. Mở **một ctivity_order.xml** (nếu nó chưa mở) và tìm một trong các giá trị onRadioButtonClicked cho thuộc tính android:onClick được gạch chân màu đỏ.
2. Nhấp vào giá trị o nRadioButtonClicked, sau đó nhấp vào biểu tượng cảnh báo bóng đèn màu đỏ ở lề trái.
3. Chọn **Create onRadioButtonClicked(View)** trong **OrderActivity** trong menu của bóng đèn màu đỏ.

Android Studio tạo phương thức onRadioButtonClicked(View view) trong OrderActivity:

```
public void onRadioButtonClicked(Chế độ xem xem) { }
```

Ngoài ra, các giá trị o nRadioButtonClicked cho các thuộc tính android:onClick khác trong activity_order.xml được giải quyết và không còn được gạch chân.

4. Để hiển thị nút radio nào được nhấp vào (nghĩa là loại phân phối mà người dùng chọn), hãy sử dụng thông báo Toast. Mở **OrderActivity** và thêm phương thức displayToast sau:

```
public void displayToast(String message)
{ Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_SHORT).show(); }
```

5. Trong phương thức onRadioButtonClicked() mới, thêm một khối phù thủy để kiểm tra nút radio nào đã được chọn và gọi displayToast() với thông báo thích hợp. Mã sử dụng [phương thức](#) isChecked() [của giao diện Checkable](#), trả về true nếu nút được chọn. Nó cũng sử dụng phương thức View getID() để lấy mã định danh cho nút radio đã chọn view:

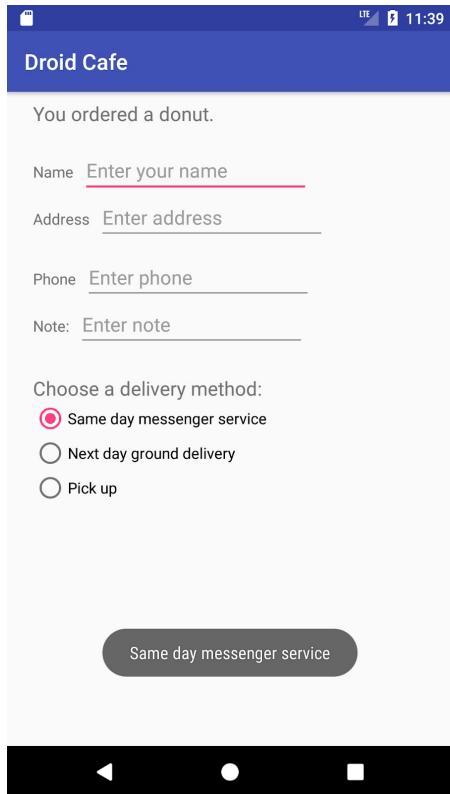
```
public void onRadioButtonClicked(Xem chế độ xem) {
    Nút hiện đã được chọn chưa?
    boolean checked = ((RadioButton) view).isChecked();
```

Kiểm tra nút radio nào đã được nhấp vào.

```
switch (view.getId()) {  
    trường hợp R.id.sameday:  
        if (đã chọn)  
            Dịch vụ trong ngày  
            displayToast(getString(R.string.same_day_messenger_service));      phá vỡ;  
    trường hợp R.id.nextday:  
        if (đã chọn)  
            Giao hàng vào ngày hôm sau  
            displayToast(getString(R.string.next_day_ground_delivery));      phá vỡ;  
    trường hợp R.id.pickup:  
        if (đã chọn)  
            Nhặt  
            displayToast(getString(R.string.pick_up));  
        phá vỡ;  
    Mặc định:  
        Không làm gì cả.  
        phá vỡ;  
}  
}
```

6. Chạy ứng dụng. Nhấn vào một hình ảnh để xem hoạt động OrderActivity, hiển thị các lựa chọn phân phối.

Nhấn vào một lựa chọn giao hàng và bạn sẽ thấy thông báo Toast ở cuối màn hình với lựa chọn, như thể hiện trong hình bên dưới.



Mã giải pháp nhiệm vụ 2

Dự án Android Studio: [DroidCafeInput](#)

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 này.

Thử thách: Các nút radio cho các lựa chọn giao hàng trong ứng dụng DroidCafeInput lần đầu tiên xuất hiện chưa được chọn, điều này ngụ ý rằng không có lựa chọn giao hàng mặc định. Thay đổi các nút radio để một trong số chúng (chẳng hạn như nextday) được chọn làm mặc định khi các nút radio xuất hiện lần đầu tiên.

Gợi ý: Bạn có thể hoàn thành nhiệm vụ này hoàn toàn trong tệp bố cục. Thay vào đó, bạn có thể viết mã trong OrderActivity để chọn một trong các nút radio khi Activity xuất hiện lần đầu tiên.

Mã giải pháp thách thức

Dự án Android Studio: [DroidCafeInput](#) (See nút radio thứ hai trong tệp bố cục activity_order.xml)

Nhiệm vụ 3: Sử dụng con quay cho lựa chọn của người dùng

Ghim [S](#) cung cấp một cách nhanh chóng để chọn một giá trị từ một tập hợp. Chạm vào ghim S sẽ hiển thị danh sách thả xuống với tất cả các giá trị có sẵn, từ đó người dùng có thể chọn một. Nếu bạn chỉ cung cấp hai hoặc ba lựa chọn, bạn có thể muốn sử dụng các nút radio cho các lựa chọn nếu bạn có chỗ trong bố cục của mình cho chúng; tuy nhiên, với nhiều hơn ba lựa chọn, ghim S hoạt động rất tốt, cuộn khi cần thiết để hiển thị các mục và chiếm ít dung lượng trong bố cục của bạn.

Mẹo: Để biết thêm thông tin về con quay, hãy xem [Spinners](#).

Để cung cấp cách chọn nhãn cho số điện thoại (chẳng hạn như **H ome**, **W ork**, **M obile** hoặc **O ther**) , bạn có thể thêm một con quay vào bố cục OrderActivity trong ứng dụng DroidCafe để xuất hiện ngay bên cạnh trường số điện thoại.

3.1 Thêm một con quay vào bố cục

Để thêm một con quay vào bố cục OrderActivity trong ứng dụng DroidCafe, hãy làm theo các bước sau, được đánh số trong hình bên dưới:

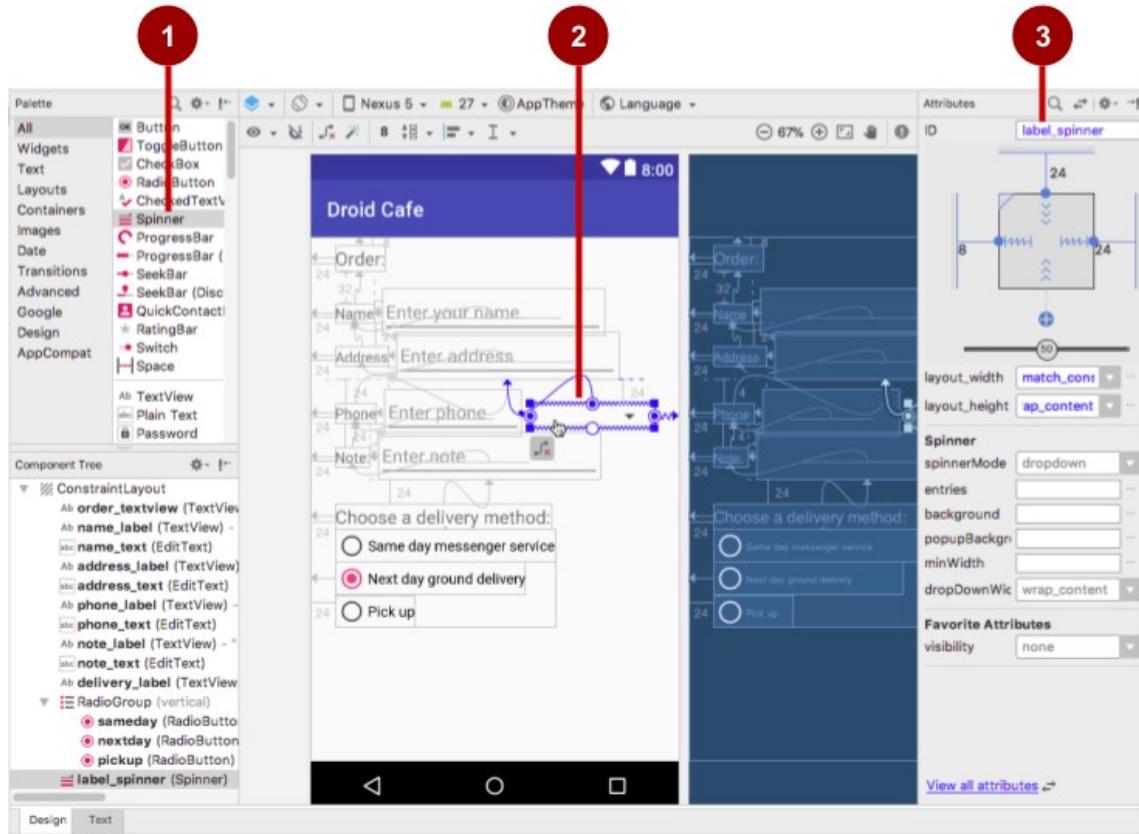
1. Mở **một ctivity_order.xml** và kéo **ghim S** từ ngăn **P alette** vào bố cục.
2. Hạn chế phần trên của phần tử ghim S ở dưới cùng của address_text, bên phải ở phía bên phải của bố cục và phía bên trái vào phone_text.

Để căn chỉnh các phần tử S ginner và phone_text theo chiều ngang, hãy sử dụng nút đóng gói  trên thanh công cụ, cung cấp các tùy chọn để đóng gói hoặc mở rộng các phần tử giao diện người dùng đã chọn.

Chọn cả hai phần tử S pinner và phone_text trong **C omponent Tree**, nhấp vào nút pack và chọn **E xpand Horizontally**. Do đó, cả hai phần tử S ginner và phone_text đều được đặt thành chiều rộng cố định.

3. Trong ngăn Attributes, đặt S ginner ID thành **la bel_spinner** và đặt lề trên và phải thành **24** và lề trái thành **8**. Chọn **m atch_constraint** cho menu thả xuống la yout_width và **w rap_content** cho menu la yout_height d rop-down.

Bố cục sẽ giống như hình bên dưới. Menu la yout_width d rop-down của phần tử phone_text trong ngăn Attributes được đặt thành 134dp. Bạn có thể tùy ý thử nghiệm với các cài đặt chiều rộng khác.



Để xem mã XML cho một ctivity_order.xml, hãy nhấp vào tab Text.

Ginner S phải có các thuộc tính sau:

```
<Spinner
    android:id="@+id/label_spinner"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="24dp"
    android:layout_marginRight="24dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="24dp"/>
```

```
    app:layout_constraintEnd_toEndOf="cha mẹ"  
    Ứng dụng:layout_constraintStart_toEndOf="@+id/phone_text"  
    Ứng dụng:layout_constraintTop_toBottomOf="@+id/address_text" />
```

Đảm bảo thêm các thuộc tính android:layout_marginRight và android:layout_marginLeft được hiển thị trong đoạn mã ở trên để duy trì khả năng tương thích với các phiên bản Android cũ hơn.

Phần tử `p hone_text` bây giờ sẽ có các thuộc tính sau (sau khi sử dụng công cụ gói):

```
<EditText  
    android:id="@+id/phone_text"  
    android:layout_width="134dp"  
    android:layout_height="wrap_content"  
    android:layout_marginLeft="8dp" android:layout_marginStart="8dp"  
    android:ems="10"  
    android:hint="@string/enter_phone_hint"  
    android:inputType="điện thoại"  
    Ứng dụng:layout_constraintBaseline_toBaselineOf="@+id/phone_label"  
    app:layout_constraintStart_toEndOf="@+id/phone_label" />
```

3.2 Thêm mã để kích hoạt Spinner và trình nghe của nó

Các lựa chọn cho `ginner S` là các chuỗi tĩnh được xác định rõ ràng như "Home" và "Work", vì vậy bạn có thể sử dụng một mảng văn bản được xác định trong `s trings.xml` để giữ các giá trị cho nó.

Để kích hoạt `ginner S` và trình nghe của nó, hãy triển khai giao `diện A dapterView.OnItemSelectedListener`, giao diện này cũng yêu cầu thêm các phương thức gọi lại `o nItemSelected()` và `o nNothingSelected()`.

1. Mở **s trings.xml** và xác định các giá trị có thể lựa chọn (**H ome, W ork, M obile** và **O ther**) cho ginner S dưới dạng mảng chuỗi `labels_array`:

```
<string-array name="labels_array">  
    <mục>Trang chủ</mục>  
    <mục>công việc</mục>  
    <mặt hàng>điện thoại di động</mặt hàng>  
    <mục>Khác</mục>  
</mảng chuỗi>
```

2. Để xác định lệnh gọi lại lựa chọn cho người ghim S, hãy thay đổi lớp `OrderActivity` của bạn để triển khai giao diện `AdapterView.OnItemSelectedListener` như minh họa:

```
public class OrderActivity mở rộng AppCompatActivity thực hiện  
    AdapterView.OnItemSelectedListener {
```

Khi bạn nhập `AdapterView`, trong câu lệnh ở trên, Android Studio sẽ tự động nhập tiện ích `AdapterView`. Lý do tại sao bạn cần `AdapterView` là vì bạn cần một bộ điều hợp — cụ thể là `ArrayAdapter` — để gán mảng cho ginner S. Một `Adapter` kết nối dữ liệu của bạn - trong trường hợp này là mảng các mục spinner - với ginner S. Bạn tìm hiểu thêm về mô hình sử dụng bộ điều hợp để kết nối dữ liệu trong một thực tế khác. Dòng này sẽ xuất hiện trong khối câu lệnh nhập của bạn:

```
nhập android.widget.AdapterView;
```

Sau khi nhập **OnItemSelectedListener** trong câu lệnh ở trên, hãy đợi vài giây để bóng đèn màu đỏ xuất hiện ở lề trái.

3. Nhấp vào bóng đèn và chọn **Implement methods**. Các phương thức `onItemSelected()` và `onNothingSelected()`, được yêu cầu cho `OnItemSelectedListener`, nên được đánh dấu và tùy chọn "Chèn @Override" nên được chọn. Nhấp vào **OK**

Bước này tự động thêm các phương thức gọi lại `onItemSelected()` và `onNothingSelected()` trống vào cuối lớp `OrderActivity`. Cả hai phương thức đều sử dụng tham số `AdapterView<?>`. `<?>` là một ký tự đại diện kiểu Java, cho phép phương thức đủ linh hoạt để chấp nhận bất kỳ loại `AdapterView` nào làm đối số.

4. Khởi tạo một ginner S trong phương thức `onCreate()` bằng cách sử dụng phần tử `label_spinner` trong bố cục và đặt trình nghe của nó (`spinner.setOnItemSelectedListener`) trong phương thức `onCreate()`, như được hiển thị trong đoạn mã sau:

```
@Override được bảo vệ void onCreate(Bundle
```

```
savedInstanceState){
```

```
// ... Phần còn lại của mã onCreate ...
```

Tạo con quay.

```
Spinner spinner = findViewById(R.id.label_spinner);
```

```
if (spinner != null) {
```

```
    spinner.setOnItemSelectedListener(cái này);
```

```
}
```

Tạo `ArrayAdapter` bằng cách sử dụng mảng chuỗi và bố cục `spinner` mặc định.

5. Tiếp tục chỉnh sửa phương thức `onCreate()`, thêm một câu lệnh tạo `ArrayAdapter` với mảng chuỗi (`labels_array`) bằng cách sử dụng bố cục ginner S do Android cung cấp cho mỗi mục (`layout.simple_spinner_item`):

Tạo ArrayAdapter bằng cách sử dụng mảng chuỗi và bối cục spinner mặc định.

```
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,  
R.array.labels_array, android.R.layout.simple_spinner_item);
```

Chỉ định bối cục để sử dụng khi danh sách các lựa chọn xuất hiện.

Bối cục `simple_spinner_item` được sử dụng trong bước này và `simple_spinner_dropdown_item` bối cục được sử dụng trong bước tiếp theo là các bối cục được xác định trước mặc định do Android cung cấp trong [lớp R.layout](#). Bạn nên sử dụng các bối cục này trừ khi bạn muốn xác định bối cục của riêng mình cho các mục trong ginner S và giao diện của nó.

6. Chỉ định bối cục cho các lựa chọn ginner S là `simple_spinner_dropdown_item`, sau đó áp dụng bộ điều hợp cho ginner S:

Chỉ định bối cục để sử dụng khi danh sách các lựa chọn xuất hiện. `adapter.setDropDownViewTài
nguyên`

```
(Android.R.layout.simple_spinner_dropdown_item);
```

Áp dụng bộ chuyển đổi vào con quay. if

```
(spinner != null) {  
    spinner.setAdapter (bộ chuyển đổi);  
}  
  
// ... Kết thúc onTạo mã ...
```

3.3 Thêm mã để phản hồi các lựa chọn Spinner

Khi người dùng chọn một mục trong spinner S, ghim S sẽ nhận được sự kiện được chọn trên mục. Để xử lý sự kiện này, bạn đã triển khai giao diện AdapterView.OnItemSelectedListener ở bước trước, thêm các phương thức gọi lại nItemSelected() và nNothingSelected() trống.

Trong bước này, bạn điền mã cho phương thức onItemSelected() để truy xuất mục đã chọn trong Spinner, sử dụng getItemAtPosition(), và gán mục cho biến spinnerLabel s:

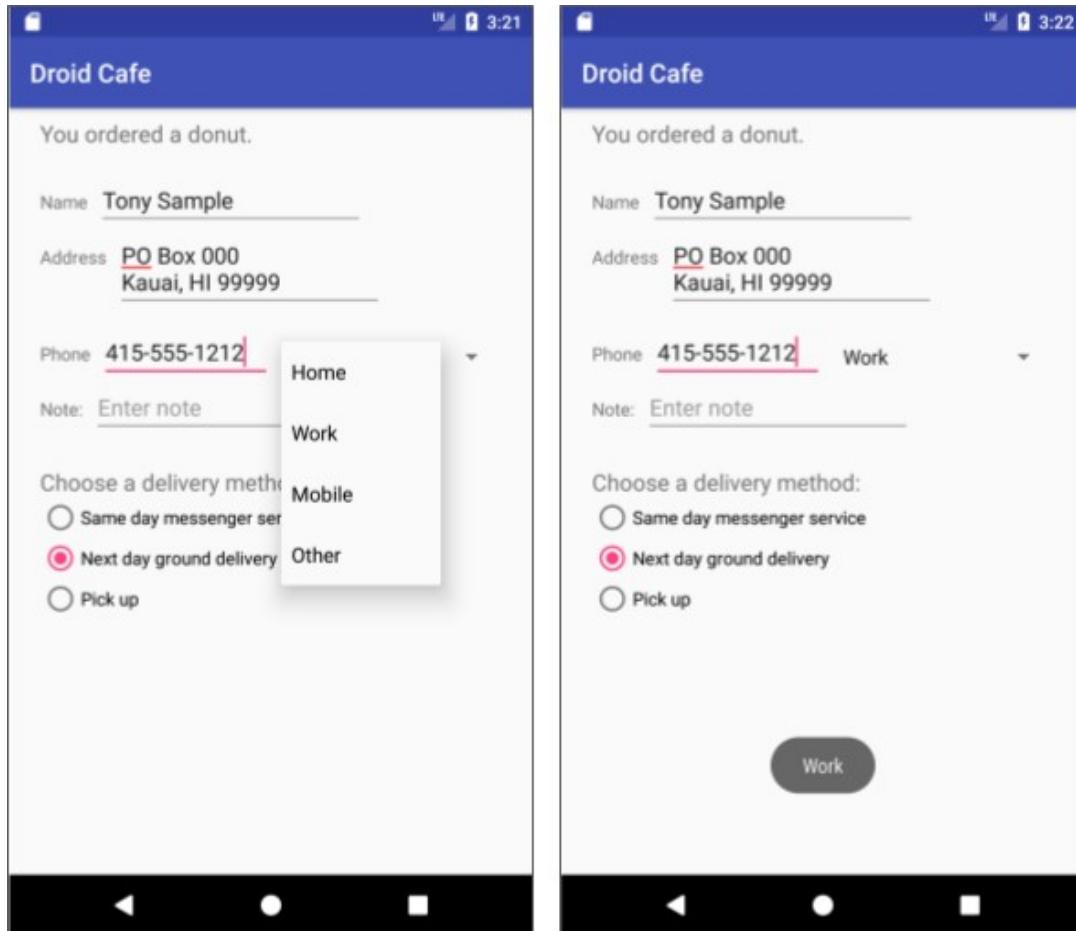
- Thêm mã vào phương thức gọi lại onItemSelected() trống, như được hiển thị bên dưới, để truy xuất mục đã chọn của người dùng bằng cách sử dụng [getItemAtPosition\(\)](#) và gán nó cho s spinnerLabel. Bạn cũng có thể thêm một lệnh gọi vào phương thức `displayToast()` mà bạn đã thêm vào OrderActivity:

```
public void onItemSelected(AdapterView<?> adapterView, Chế độ xem, int
    i, long l) {
    spinnerLabel = adapterView.getItemAtPosition(i).toString();
    displayToast (spinnerLabel);
}
```

Không cần thêm mã vào phương thức gọi lại onNothingSelected() trống cho ví dụ này.

- Chạy ứng dụng.

Ghim chữ S xuất hiện bên cạnh trường nhập điện thoại và hiển thị lựa chọn đầu tiên (**Home**). Nhấn vào Spinner tiết lộ tất cả các lựa chọn, như được hiển thị ở phía bên trái của hình bên dưới. Nhấn vào một lựa chọn trong Spinner hiển thị thông báo Toast với lựa chọn, như được hiển thị ở phía bên phải của hình.



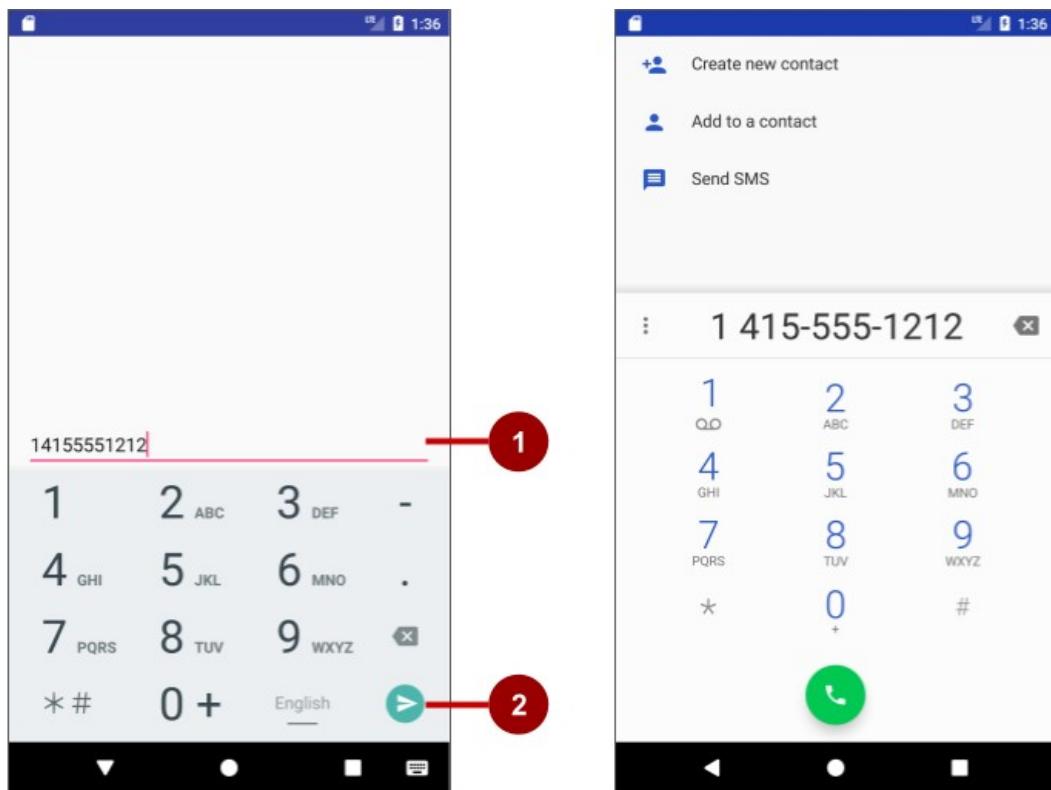
Mã giải pháp nhiệm vụ 3

Dự án Android Studio: [DroidCafeInput](#)

Thử thách mã hóa 2

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: Write code để thực hiện một hành động trực tiếp từ bàn phím bằng cách nhấn vào **phím kết thúc S**, chẳng hạn như để quay số điện thoại:



Trong hình trên:

1. Nhập số điện thoại vào trường `EditText`.
2. Nhấn vào **phím kết thúc S** để khởi chạy trình quay số điện thoại. Trình quay số xuất hiện ở phía bên phải của hình.

Đối với thử thách này, hãy tạo một dự án ứng dụng mới và thêm EditText sử dụng thuộc tính android:inputType được đặt thành phone. Sử dụng [thuộc tính](#) android:imeOptions cho phần tử EditText với giá trị actionSend:

```
android:imeOptions="hành độngGửi"
```

Bây giờ người dùng có thể nhấn phím **cuối S** để quay số điện thoại, như thể hiện trong hình trên.

Trong phương thức onCreate() cho Activity A này, bạn có thể sử dụng setOnEditorActionListener() để thiết lập trình nghe cho EditText để phát hiện xem phím có được nhấn hay không:

```
EditText editText = findViewById(R.id.editText_main); if (editText != null)
    editText.setOnEditorActionListener
        (new TextView.OnEditorActionListener() {
    Nếu tìm thấy chế độ xem, hãy đặt trình nghe cho editText.});
```

Để được trợ giúp thiết lập trình nghe, hãy xem [Sắc định loại phương thức nhập](#).

Bước tiếp theo là ghi đè `onEditorAction()` và sử dụng hằng số `IME_ACTION_SEND` [trong lớp EditorInfo](#) để phản hồi với phím đã nhấn. Trong ví dụ bên dưới, phím được sử dụng để gọi phương thức `dialNumber()` để quay số điện thoại:

```
@Override boolean công khai onEditorAction(TextView textView, int actionId, KeyEvent keyEvent)
{ boolean handled = false;
    if (actionId == EditorInfo.IME_ACTION_SEND)
        dialNumber();
        xử lý = đúng;
    }
    trả lại xử lý;
```

}

Để hoàn thành thử thách, hãy tạo phương thức dialNumber(), phương thức này sử dụng ý định ngầm với ACTION_DIAL để chuyển số điện thoại đến một ứng dụng khác có thể quay số. Nó sẽ trông như thế này:

```
private void dialNumber() {  
    Tìm chế độ xem editText_main.  
    EditText editText = findViewById(R.id.editText_main);  
    Chuỗi phoneNum = null;  
    Nếu trường editText không rỗng,  
    Nối "TEL: " với chuỗi số điện thoại.  
    Nếu (edittext != chum) và điện thoại = "tel:" +  
        editText.getText().toString();  
    Tùy chọn: Ghi lại số điện thoại được nối để quay số.  
    Log.d(TAG, "dialNumber: " + phoneNum);  
    Chỉ định ý định.  
    Ý định = ý định mới (Intent.ACTION_DIAL);  
    Đặt dữ liệu cho ý định làm số điện thoại.  
    intent.setData(Uri.parse(phoneNum));  
    Nếu ý định phân giải thành một gói (ứng dụng),  
    Bắt đầu hoạt động với ý định.  
    if (intent.resolveActivity(getApplicationContext()) != null) {  
        startActivity(ý định);  
    } khác {  
        Log.d("ImplicitIntents", "Không thể xử lý việc này!");  
    }  
}
```

Mã giải pháp thử thách 2

Dự án Android Studio: [KeyboardDialPhone](#)

Tóm tắt

Các giá trị thuộc tính android:inputType sau đây ảnh hưởng đến giao diện của bàn phím ảo:

- textAutoCorrect: Đề xuất sửa chính tả.
- textCapCâu: Sao t mỗi câu mới có chữ in hoa.
- textPersonName: Hiển thị một dòng văn bản với các đề xuất khi người dùng nhập và phím D một đ để người dùng nhấn khi họ hoàn tất.
- textMultiLine: Bật nhiều dòng nhập văn bản và phím Return để thêm một dòng mới.
- textPassword: Ẩn mật khẩu khi nhập.
- textEmailAddress: Hiển thị bàn phím email thay vì bàn phím tiêu chuẩn.
- điện thoại: Hiển thị bàn phím điện thoại thay vì bàn phím tiêu chuẩn.

Bạn đặt giá trị cho thuộc tính android:inputType trong tệp bố cục XML cho phần tử EditText Để kết hợp các giá trị, hãy nối chúng bằng cách sử dụng đường ống (|) nhân vật.

Các nút radio là các điều khiển đầu vào hữu ích để chỉ chọn một tùy chọn từ một tập hợp các tùy chọn:

- Nhóm các [phần tử RadioButton](#) lại với nhau bên trong A [RadioGroup](#) để chỉ có thể chọn một RadioButton tại một thời điểm.
- Thứ tự bạn liệt kê các phần tử RadioButton trong nhóm xác định thứ tự chúng xuất hiện trên màn hình.
- Sử dụng thuộc tính android:onClick cho mỗi RadioButton để chỉ định trình xử lý nhập chuột.
- Để tìm hiểu xem một nút có được chọn hay không, hãy sử dụng [phương thức](#) isChecked() của giao diện Checkable, phương thức này trả về true nếu nút được chọn.

Một [ghim chữ S](#) cung cấp một menu thả xuống:

- Thêm một [ghim S](#) vào bố cục.
- Sử dụng [ArrayAdapter](#) để gán một mảng giá trị văn bản làm mục menu spinner S.
- Triển khai giao [diễn AdapterView.OnItemSelectedListener](#), cũng yêu cầu thêm các phương thức onOptionsItemSelected() và onNothingSelected() để kích hoạt spinner S và trình nghe của nó.

- Sử dụng [phương thức gọi lại o nItemSelected\(\)](#) để truy xuất mục đã chọn trong menu ginner S bằng cách sử dụng [getSelectedItemAtPosition\(\)](#).

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [4.2: Điều khiển đầu vào](#).

Tìm hiểu thêm

Tài liệu Android Studio:

- [Hướng dẫn sử dụng Android Studio](#)

Tài liệu dành cho nhà phát triển Android:

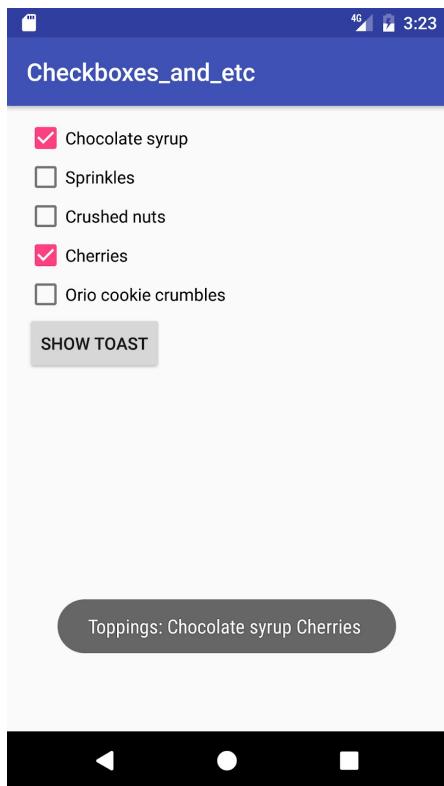
- [Tổng quan về sự kiện đầu vào](#)
- [Chỉ định loại phương thức nhập](#)
- [Phong cách và chủ đề](#)
- [Các nút radio](#)
- [Spinners](#)
- [Cảnh](#)
- [Nút](#)
- [Chỉnh sửa Văn bản](#)
- [android:loại đầu vào](#)
- [Chế độ xem văn bản](#)
- [Nhóm vô tuyến](#)
- [Hộp kiểm](#)
- [Thanh tìm kiếm](#)
- [Nút chuyển đổi](#)

- [Spinner](#)

Homework

Xây dựng và chạy ứng dụng

1. Tạo một ứng dụng với năm hộp kiểm và nút **Show Toast**, như được hiển thị bên dưới.
2. Nếu người dùng chọn một hộp kiểm duy nhất và nhấn vào **Show Toast**, hãy hiển thị thông báo Toast hiển thị hộp kiểm đã được chọn.
3. Nếu người dùng chọn nhiều hộp kiểm và sau đó nhấn vào **Show Toast**, hãy hiển thị một Toast bao gồm các thông báo cho tất cả các hộp kiểm đã chọn, như thể hiện trong hình bên dưới.



Trả lời những câu hỏi này

Câu hỏi 1

Sự khác biệt quan trọng nhất giữa hộp kiểm và RadioGroup các nút radio là gì?

Chọn một:

- Sự khác biệt duy nhất là cách chúng xuất hiện: hộp kiểm hiển thị dấu kiểm khi được chọn, trong khi các nút "radio" hình tròn xuất hiện được lấp đầy khi được chọn.
- Các phần tử CheckBox trong bố cục có thể sử dụng thuộc tính android:onClick để gọi trình xử lý khi được chọn.
- Sự khác biệt chính là các hộp kiểm cho phép nhiều lựa chọn, trong khi RadioGroup chỉ cho phép một lựa chọn.

Câu hỏi 2

Nhóm bố cục nào cho phép bạn căn chỉnh một tập hợp các phần tử CheckBox theo chiều dọc? Chọn một:

- Bố cục tương đối
- Bố cục tuyến tính
- Chế độ xem cuộn(ScrollView)

Câu hỏi 3

Phương pháp nào sau đây là phương pháp của giao diện [Checkable](#) để kiểm tra trạng thái của nút radio (nghĩa là nó đã được chọn hay chưa)?

- getId()
- isChecked()
- onRadioButtonClicked()
- onClick()

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:

- Bố cục bao gồm năm chế độ xem Checklist được căn chỉnh theo chiều dọc trên màn hình và nút **Show Toast**.
- Phương thức `onSubmit()` xác định hộp kiểm nào được chọn bằng cách sử dụng `findViewById()` với `isChecked()`.
- Các chuỗi mô tả lớp phủ được nối thành một thông điệp `Toast`.

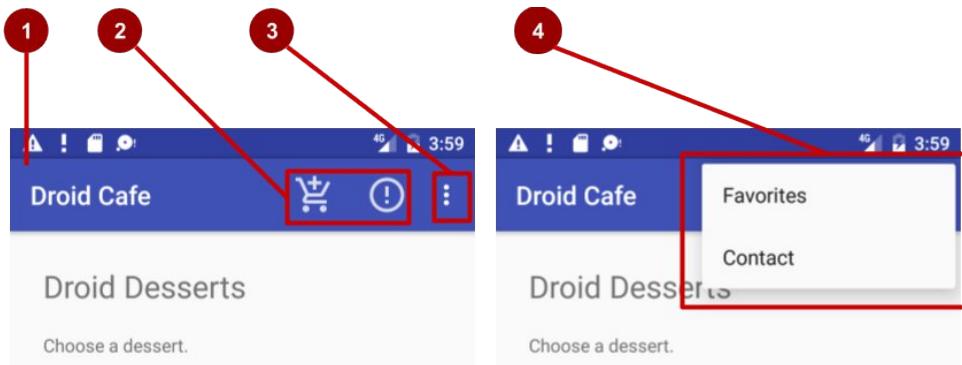
Bài 4.3: Menu và bộ chọn

Giới thiệu

Thanh `pp` (còn được gọi là thanh `ction`) là một không gian dành riêng ở đầu mỗi màn hình hoạt động. Khi bạn tạo Hoạt động từ mẫu Hoạt động cơ bản, Android Studio sẽ bao gồm một thanh ứng dụng.

Menu `options` 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 cách 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 cách thiết lập thanh ứng dụng và menu tùy chọn trong ứng dụng của mình, 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. **Biểu tượng hành động menu tùy chọn.** Hai mục menu tùy chọn đầu tiên xuất hiện dưới dạng biểu tượng trên thanh ứng dụng.
3. **Nút tràn.** Nút tràn (ba dấu chấm dọc) sẽ mở menu hiển thị các mục menu tùy chọn khác.
4. **Menu tràn tùy chọn.** Sau khi nhấp vào nút tràn, các mục menu tùy chọn khác sẽ xuất hiện trong menu tràn.

Các mục menu tùy chọn xuất hiện trong menu tràn tùy chọn (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 nhất quán 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.

Mẹo: Để cung cấp trải nghiệm người dùng quen thuộc và nhất quán, hãy sử dụng API Menu để hiển thị các hành động của người dùng và các tùy chọn khác trong các hoạt động của bạn. Xem [Menus](#) để biết chi tiết.

Bạn cũng tạo một ứng dụng hiển thị hộp thoại để yêu cầu lựa chọn của người dùng, chẳng hạn như cảnh báo yêu cầu người dùng nhấn vào **O K** hoặc **C ancel**. **D ialog** là một cửa sổ xuất hiện trên đầu 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 **p ickers**, để chọn thời gian hoặc ngày. Bạn có thể sử dụng chúng để đảm bảo rằng người dùng của bạn chọn ngày hoặc giờ hợp lệ được định dạng chính xác và được điều chỉnh theo ngày giờ địa phương của người dùng. Trong bài học này, bạn cũng sẽ tạo một ứng dụng với bộ chọn ngày.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các thành phần 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.
- Thêm trình xử lý nhấp chuột vào nút B.

Những gì bạn sẽ học

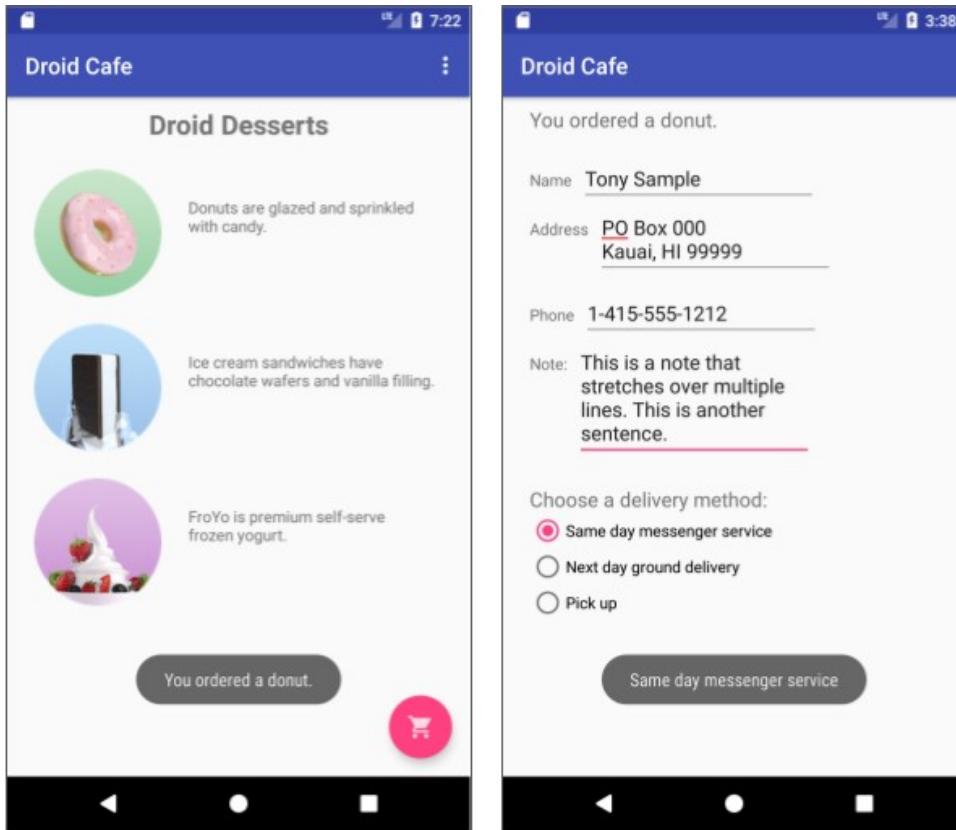
- Cách thêm các mục menu vào menu tùy chọn.
- Cách thêm biểu tượng cho các mục trong menu tùy chọn.
- Cách đặt các mục menu để hiển thị trong thanh ứng dụng.
- Cách thêm trình xử lý nhấp chuột cho các mục menu.
- Cách thêm hộp thoại cho cảnh báo.
- Cách thêm bộ chọn ngày.

Bạn sẽ làm gì

- 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 nhấp chuột mục menu với trình xử lý sự kiện xử lý các sự kiện nhấp chuột.
- Sử dụng hộp thoại cảnh báo để yêu cầu lựa chọn của người dùng.
- Sử dụng bộ chọn ngày để nhập ngày.

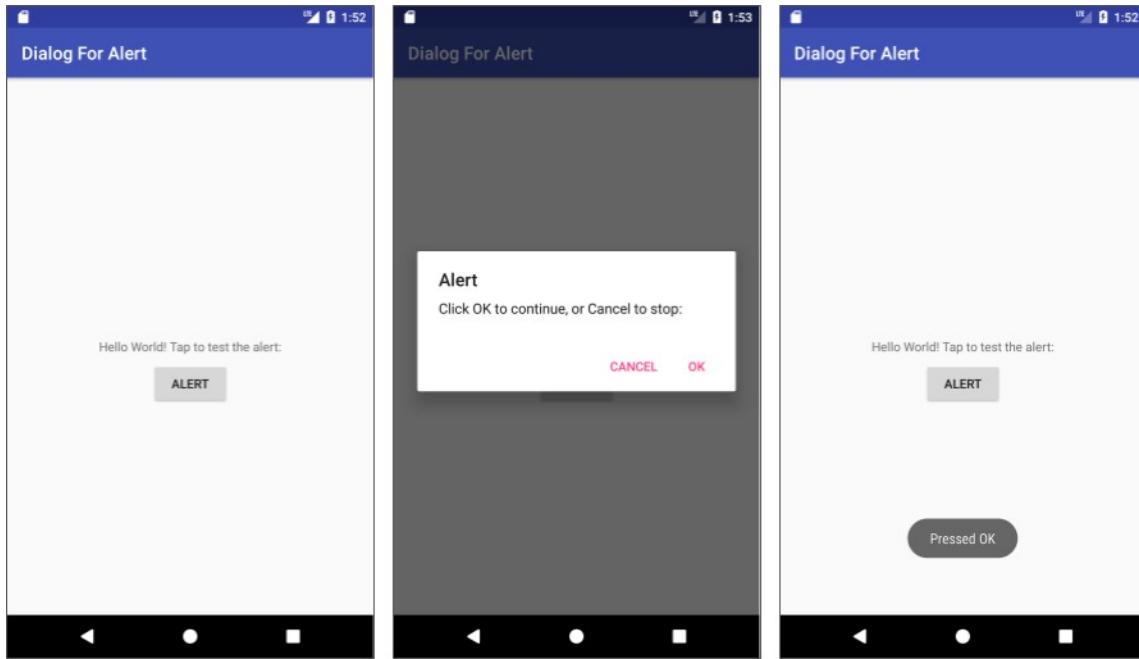
Tổng quan về ứng dụng

Trong thực tế trước, bạn đã tạo một ứng dụng có tên là Droid Cafe, được hiển thị trong hình bên dưới, bằng cách sử dụng mẫu Hoạt động cơ bản. Mẫu này cũng cung cấp menu tùy chọn khung trong thanh ứng dụng ở đầu màn hình.

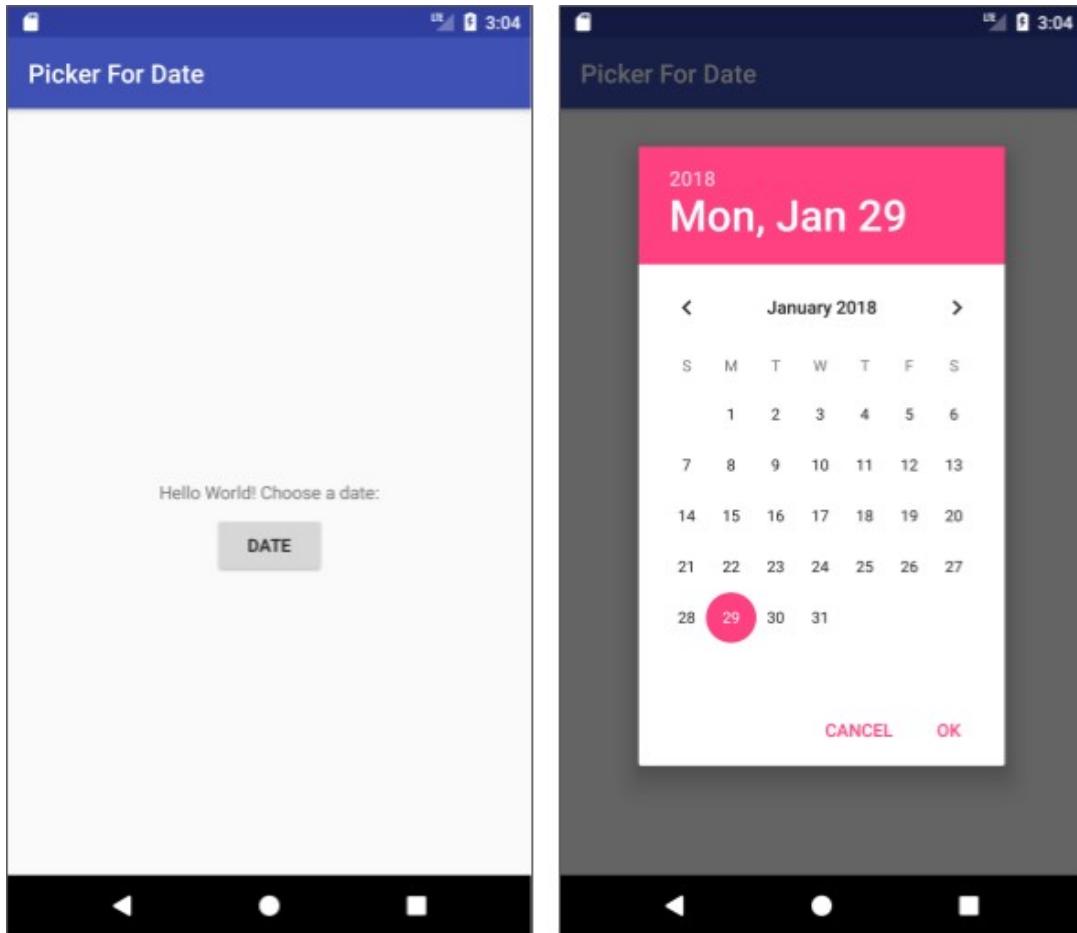


Đối với bài tập này, bạn đang sử dụng [Toolbar](#) của thư viện hỗ trợ [appcompat v 7](#) làm thanh ứng dụng, hoạt động trên nhiều loại thiết bị nhất và cũng cung cấp cho bạn không gian để tùy chỉnh thanh ứng dụng của mình sau này khi ứng dụng của bạn phát triển. Để đọc thêm về những điều cần lưu ý về thiết kế khi sử dụng thanh ứng dụng, hãy xem Lưới [bố cục tùy ý R](#) trong đặc tả Material Design.

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 chữ B để hiển thị bộ chọn ngày và chuyển đổi ngày đã chọn thành một chuỗi để hiển thị trong thông báo Toast.



Nhiệm vụ 1: Thêm các mục vào menu tùy chọn

Trong tác vụ này, bạn mở dự án [DroidCafeInput](#) từ thực tế trước đó và thêm các mục menu vào menu tùy chọn trong thanh ứng dụng ở đầu màn hình.

1.1 Kiểm tra mã

Mở ứng dụng [DroidCafeInput](#) từ thực tế về việc sử dụng các điều khiển đầu vào và kiểm tra các tệp bố cục sau trong thư mục `res > layout`:

- **activity_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 MainActivity, (như bạn sẽ thấy ngay sau đây) được bao gồm trong một **activity_main.xml**.
- **activity_order.xml**: Bố cục cho OrderActivity, mà bạn đã thêm vào thực tế về cách sử dụng các điều khiển đầu vào.

Làm theo các bước sau:

1. Mở **content_main.xml** và nhấp vào tab **Text** để xem mã XML. app:layout_behavior cho ConstraintLayout được đặt thành `@string/appbar_scrolling_view_behavior`, điều khiển cách màn hình cuộn liên quan đến thanh ứng dụng ở trên cùng. (Tài nguyên chuỗi này được định nghĩa trong một tệp được tạo ra có tên là `values.xml`, mà bạn không nên chỉnh sửa.)

Để biết thêm về hành vi cuộn, hãy xem Thư viện hỗ trợ thiết kế Android Blog dành cho nhà phát triển. Để biết các phương pháp thiết kế liên quan đến menu cuộn, hãy xem phần [S trong](#) đặc tả Material Design.

2. Mở **một activity_main.xml** và nhấp vào tab **Text** để xem mã XML cho bố cục chính, sử dụng bố cục **CoordinatorLayout với bố cục AppBarLayout** được nhúng. Thẻ CoordinatorLayout và AppBarLayout yêu cầu tên đủ điều kiện chỉ định `android.support.design`, đó là Thư viện hỗ trợ thiết kế Android.

Một AppBarLayout giống như một LinearLayout dọc. Nó sử dụng lớp [Toolbar](#) trong thư viện hỗ trợ, thay vì ActionBar gốc, để triển khai thanh ứng dụng. Toolbar trong bố cục này có id toolbar và cũng được chỉ định, giống 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. 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 [Toolbar](#) của thư viện hỗ trợ [appcompat v7](#) làm thanh ứng dụng. Sử dụng [Thanh công cụ](#) giúp bạn dễ dàng thiết lập thanh ứng dụng hoạt động trên nhiều loại thiết bị nhất và cũng cho bạn không gian để tùy chỉnh thanh ứng dụng sau này khi ứng dụng của bạn 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.

Bố cục `activity_main.xml` cũng sử dụng câu lệnh bố cục `include` để bao gồm toàn bộ bố cục được định nghĩa trong `content_main.xml`. Sự tách biệt của các định nghĩa bố cục này giúp bạn dễ dàng thay đổi `content` của bố cục ngoài định nghĩa thanh công cụ của bố cục và bố cục điều phối viên. Đây là phương pháp hay nhất để tách nội dung của bạn (có thể cần được dịch) khỏi định dạng bố cục của bạn.

- Chạy ứng dụng. Lưu ý thanh ở đầu màn hình hiển thị tên của ứng dụng (Droid Cafe). Nó cũng hiển thị nút `action overflow` (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.



- Kiểm tra tệp `AndroidManifest.xml`. Các tệp . Hoạt động `MainActivity` được đặt để sử dụng chủ đề `NoActionBar`. Chủ đề này được định nghĩa trong tệp `styles.xml` (mở `một pp > res > giá trị > styles.xml` để xem nó). Các kiểu được đề cập trong một bài học khác, nhưng bạn có thể thấy rằng chủ đề `NoActionBar` đặt thuộc tính `windowActionBar` thành `false` (không có thanh ứng dụng cửa sổ) và thuộc tính `windowNoTitle` thành `true` (không có tiêu đề). Các giá trị này được đặt vì bạn đang xác định thanh ứng dụng bằng `ppBarLayout`, thay vì sử dụng `ActionBar`. Việc sử dụng một trong các chủ đề `NoActionBar` ngăn ứng dụng sử dụng lớp `ActionBar` gốc để cung cấp thanh ứng dụng.

5. Hãy nhìn vào **MainActivity**, mở rộng `AppCompatActivity` và bắt đầu với phương thức `onCreate()`, thiết lập chế độ xem nội dung thành bố cục `activity_main.xml` và đặt toolbar là toolbar được xác định trong bố cục. Sau đó, nó gọi `setSupportActionBar()` và chuyển thanh công cụ đến nó, đặt toolbar làm thanh ứng dụng cho Activity.

Để biết các phương pháp hay nhất về cách thêm thanh ứng dụng vào ứng dụng của bạn, hãy xem một [đề thanh ứng dụng](#).

1.2 Thêm các mục menu vào menu tùy chọn

Bạn sẽ thêm các mục menu sau vào menu tùy chọn:

- **Đặt hàng:** Điều hướng đến `OrderActivity` để xem đơn đặt hàng món tráng miệng.
- **Trạng thái:** Kiểm tra trạng thái của đơn hàng.
- **Yêu thích:** Hiển thị các món tráng miệng yêu thích.
- **Liên hệ:** Liên hệ quán cà phê. Vì bạn không cần mục `Settings` hiện có, bạn sẽ thay đổi `Settings` thành `Contact`.

Android cung cấp định dạng XML tiêu chuẩn để xác định các mục menu. Thay vì xây dựng menu trong mã Hoạt động, bạn có thể xác định menu và tất cả các mục menu trong tài nguyên menu XML. Sau đó, bạn có thể thổi phồng tài nguyên menu (tải nó dưới dạng đối tượng `Menu`) trong Activity của bạn:

1. Mở rộng `res > menu` trong `ngân Project > Android` và mở `menu_main.xml`. Mục menu duy nhất được cung cấp từ mẫu là một `action_settings` (lựa chọn `Settings`), được định nghĩa là:

```
< mục
    android:id="@+id/action_settings"
    android:orderInCategory = "100"
    android:title="@string/action_settings"
    app:showAsAction="không bao giờ" />
```

2. Thay đổi các thuộc tính sau của mục `action_settings` để làm cho nó trở thành mục `action_contact` (không thay đổi thuộc tính `android:orderInCategory` hiện có):

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

android:id	"@+id/action_contact"
android:tiêu đề	"Liên hệ"
ứng dụng:showAsAction	"Không bao giờ"

3. Trích xuất chuỗi được mã hóa cứng " Liên hệ" vào tài nguyên chuỗi một ction_contact.
4. Thêm một mục menu mới bằng cách sử dụng thẻ < item> trong khối menu <> và cung cấp cho mục các thuộc tính sau:

Thuộc tính	Giá trị
android:id	"@+id/action_order"
android:orderInCategory	"10"
android:tiêu đề	"Đặt hàng"
ứng dụng:showAsAction	"Không bao giờ"

Thuộc tính android:orderInCategory chỉ định thứ tự mà các mục menu xuất hiện trong menu, với số thấp nhất xuất hiện cao hơn trong menu. Mục **C ontact** được đặt thành 100, đây là một con số lớn để chỉ định rằng nó hiển thị ở dưới cùng chứ không phải ở trên cùng. Bạn đặt mục **O rder** thành 10, đặt nó lên trên **C ontact** và để lại nhiều chỗ trong menu cho nhiều mục hơn.

5. Trích xuất chuỗi được mã hóa cứng " Order" vào tài nguyên chuỗi một ction_order.
6. Thêm hai mục menu nữa theo cách tương tự với các thuộc tính sau:

Thuộc tính mục trạng thái	Giá trị
android:id	"@+id/action_status"
android:orderInCategory	"20"
android:tiêu đề	"Trạng thái"
ứng dụng:showAsAction	"Không bao giờ"

Thuộc tính mục yêu thích	Giá trị
android:id	"@+id/actionFavorites"
android:orderInCategory	"30"
android:tiêu đề	"Yêu thích"
ứng dụng:showAsAction	"Không bao giờ"

7. Trích xuất "Trạng thái" vào tài nguyên một ction_status và "Yêu thích" vào tài nguyên actionFavorites.
8. Bạn sẽ hiển thị thông báo Toast với thông báo hành động tùy thuộc vào mục menu mà người dùng chọn.
Mở **strings.xml** và thêm tên chuỗi và giá trị sau cho các thư này:

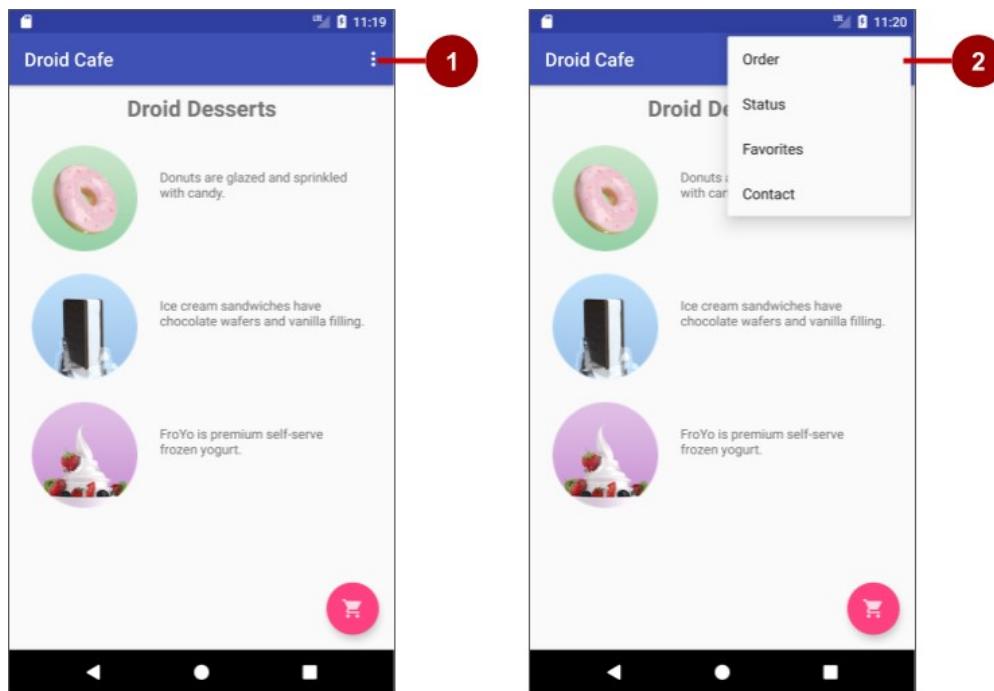
```
<string name="action_order_message">Bạn đã chọn Order.</string>
<string name="action_status_message">Bạn đã chọn Status.</string>
```

```
<string name="action_favorites_message">Bạn đã chọn Yêu thích.</chuỗi>
<string name="action_contact_message">Bạn đã chọn Contact.</string>
```

9. Mở **MainActivity** và thay đổi câu lệnh `if` trong phương thức `onOptionsItemSelected()` thay thế `action_settings` bằng `id action_order`:

```
Nếu (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 callback để phản hồi 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 thả xuống từ thanh ứng dụng.

Lưu ý thứ tự của 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 **O rder** là 10, tiếp theo là **S tatus** (20) và **F avorites** (30) và **C ontact** là cuối cùng (100). Bảng sau đây cho thấy mức độ ưu tiên của các mục trong menu:

Mục menu	thuộc tính <code>orderInCategory</code>
Trật tự	10
Tình trạng	20
Yêu thích	30
Sự tiếp xúc	100

Nhiệm vụ 2: Thêm biểu tượng cho các mục menu

Bất cứ khi nào có thể, bạn muốn hiển thị các hành động được sử dụng thường xuyên nhất bằng cách sử dụng các biểu tượng trong thanh ứng dụng để người dùng có thể nhấp vào chúng mà không cần phải nhấp vào biểu tượng tràn trước. Trong tác vụ này, bạn thêm biểu tượng cho một số mục menu và hiển thị một số mục menu trong thanh ứng dụng ở đầu màn hình dưới dạng biểu tượng.

Trong ví dụ này, giả sử rằng các hành động **O rder** và **S tatus** được sử dụng thường xuyên nhất. Hành động **Yêu thích** thỉnh thoảng được sử dụng và **C ontact** là ít được sử dụng nhất. Bạn có thể đặt biểu tượng cho các tác vụ này và chỉ định những điều sau:

- **Trật tự** và **S tatus** phải luôn được hiển thị trên thanh ứng dụng.
- **Yêu thích** sẽ được hiển thị trong thanh ứng dụng nếu nó phù hợp; nếu không, nó sẽ xuất hiện trong menu tràn.
- **Danh bạ** không nên xuất hiện trong thanh ứng dụng; nó sẽ không 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 dưới dạng nội dung hình ảnh vào **thư mục có thể thô bằng** cách sử dụng cùng một quy trình mà 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):

-  **Ngoài ra:** Sử dụng cùng một biểu tượng bạn đã thêm cho nút hành động nổi trong thực tế về việc sử dụng hình ảnh có thể nhấp (ic_shopping_cart.png).
 -  **S tatus:**
 -  **F avorites:**
 - **L ien h e:** Không cần biểu tượng vì nó sẽ chỉ xuất hiện trong menu tràn.

Đối với các biểu tượng **S tatus** và **F avorites**, hãy làm theo các bước sau:

1. Mở rộng **r es** trong **ngăn P roject > Android** và nhấp chuột phải (hoặc Control khi nhấp vào) thư **mục có thể rawable**.
2. Chọn **N ew > Image Asset**. Hộp thoại Định cấu hình nội dung hình ảnh xuất hiện.
3. Chọn **Thanh ction** và **Mục tab** trong menu thả xuống.
4. Thay đổi **ic _action_name** thành tên khác (chẳng hạn như **ic _status_info** cho **biểu tượng S tatus**).
5. Nhấp vào hình ảnh clip art (logo Android bên cạnh **C lipart:**) để chọn hình ảnh clip art làm biểu tượng. Một trang biểu tượng xuất hiện. Nhấp vào biểu tượng bạn muốn sử dụng.
6. Chọn **H OLO_DARK** từ menu thả xuống **T heme**. Thao tác này đặt biểu tượng thành màu trắng trên nền tối (hoặc đen). Nhấp vào **N ext** và sau đó nhấp vào **F inish**.

Mẹo: Xem C [tái tạo các biểu tượng ứng dụng bằng Image Asset Studio](#) để biết mô tả đầy đủ.

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 `pp:showAsAction` trong `menu_main.xml`. Các giá trị sau cho thuộc tính chỉ định liệu hành động có nên xuất hiện trong thanh ứng dụng dưới dạng biểu tượng hay không:

- "luôn luôn": Luôn xuất hiện trong thanh ứng dụng. (Nếu không có đủ chỗ, nó có thể trùng lặp với các biểu tượng menu khác.)
- "ifRoom": Xuất hiện trong thanh ứng dụng nếu có chỗ.
- "Không bao giờ": Không bao giờ xuất hiện trong thanh ứng dụng; văn bản của nó xuất hiện trong menu tràn.

Làm theo các bước sau 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 các mục **O rder**, **S tatus** và **Yêu thích** để hai thuộc tính đầu tiên (**O rder** và **S tatus**) luôn xuất hiện và mục **F avorites** chỉ xuất hiện nếu có chỗ cho nó:

Thuộc tính mặt hàng đặt hàng	Giá trị cũ	Giá trị mới
Android:Biểu tượng	<i>không ai</i>	"@drawable/ic_shopping_cart"
Ứng dụng:showAsAction	"Không bao giờ"	"Luôn luôn"

Thuộc tính mục trạng thái	Giá trị cũ	Giá trị mới
Android:Biểu tượng	<i>không ai</i>	"@drawable/@drawable/ic_status_info"
Ứng dụng:showAsAction	"Không bao giờ"	"Luôn luôn"

Thuộc tính mục yêu thích	Giá trị cũ	Giá trị mới
Android:Biểu tượng	không ai	"@drawable/ic_favorite"
Ứng dụng:showAsAction	"Không bao giờ"	"Nếu phòng"

2. Chạy ứng dụng. Bây giờ bạn sẽ thấy ít nhất hai biểu tượng trên thanh ứng dụng: biểu tượng cho Order và biểu tượng cho Status như được hiển thị ở phía bên trái của hình bên dưới. (Các Các tùy chọn Favorites và Contact xuất hiện trong menu tràn.)
3. Xoay thiết bị của bạn theo hướng ngang hoặc nếu bạn đang chạy trong trình mô phỏng, hãy nhấp vào nút **Xoay sang trái** hoặc **Rotate sang phải** để xoay màn hình theo hướng ngang. Mày

Sau đó, sẽ thấy cả ba biểu tượng trong thanh ứng dụng cho Order, Status và Favorites 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 hướng và kích thước của màn hình thiết bị. Ít nút xuất hiện theo hướng dọc, như được hiển thị ở phía bên trái của hình trên, so với hướng ngang như hiển thị ở phía bên phải của hình trên. Các nút hành động không được chiếm quá 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 nhấn và sử dụng [phương thức onOptionsItemSelected\(\)](#) để xác định mục menu nào đã được nhấn.

3.1 Tạo phương thức để hiển thị lựa chọn menu

1. Mở `MainActivity`.
2. Nếu bạn chưa thêm phương pháp sau (trong một bài học khác) để hiển thị thông báo `Toast`, hãy thêm nó ngay bây giờ. Bạn sẽ sử dụng nó như một hành động để thực hiện cho mỗi lựa chọn menu. (Thông thường, bạn sẽ thực hiện một hành động cho mỗi mục menu như bắt đầu một mục khác `Activity`, 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 `onOptionsItemSelected`

Phương thức `onOptionsItemSelected()` xử lý các lựa chọn từ menu tùy chọn. Bạn sẽ thêm khối trường hợp chuyển đổi để xác định mục menu nào đã được chọn và hành động cần thực hiện.

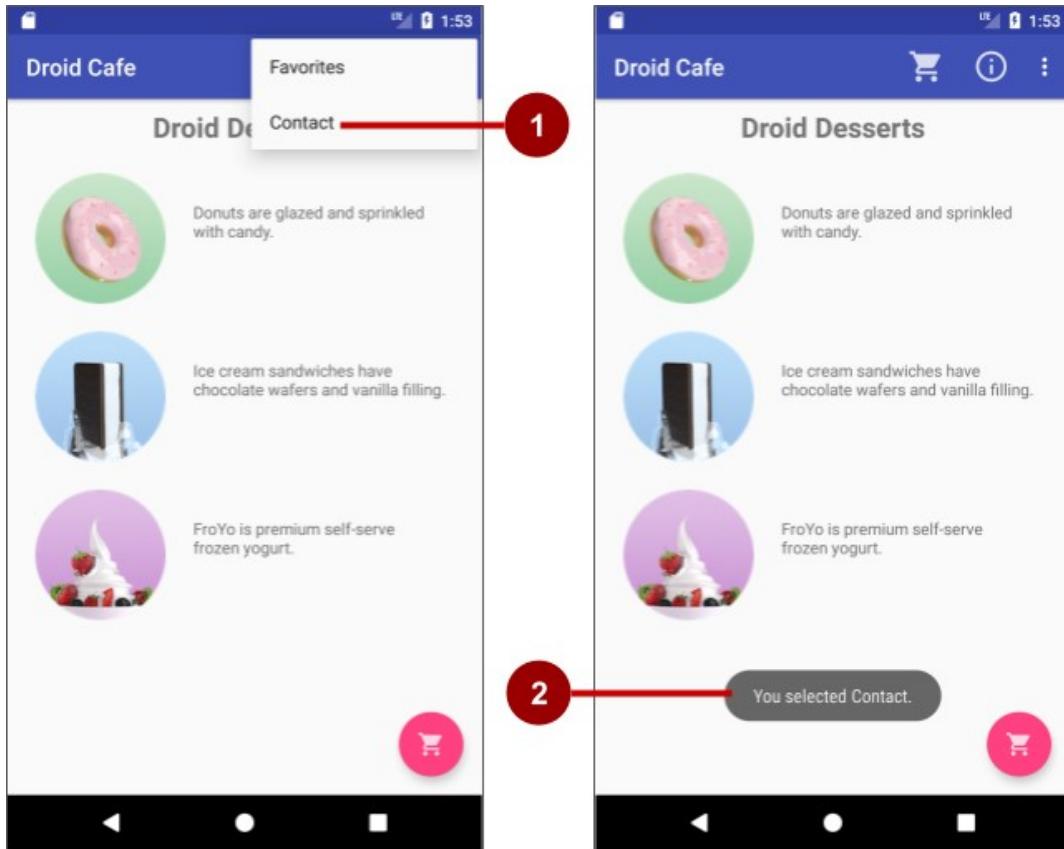
1. Tìm phương thức `onOptionsItemSelected()` do mẫu cung cấp. Phương thức xác định xem một mục menu nhất định có được nhấp vào hay không, bằng cách sử dụng `id` của mục menu. Trong ví dụ dưới đây, `i` là một `MenuItem`:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.action_order) {
        trả về true;
    }
    trả về super.onOptionsItemSelected(item); }
```

1. Thay thế câu lệnh gán id int và câu lệnh if bằng khối trường hợp phù thủy sau, đặt message thích hợp dựa trên id của mục menu:

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        Trường hợp R.id.action_order:  
            showToast(getString(R.string.action_order_message));  
            trả về true;  
        Trường hợp R.id.action_status:  
            showToast(getString(R.string.action_status_message));  
            trả về true;  
        Trường hợp R.id.action_favorites:  
            showToast(getString(R.string.action_favorites_message));  
            trả về true;  
        Trường hợp R.id.action_contact:  
            showToast(getString(R.string.action_contact_message));  
            trả về true;  
        Mặc định:  
            Không làm gì  
    }  
    trả về super.onOptionsItemSelected(item); }
```

2. Chạy ứng dụng. Bây giờ bạn sẽ thấy một thông báo Toast 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 báo Toast xuất hiện.

3.3 Bắt đầu một Hoạt động từ một mục menu

Thông thường, bạn sẽ thực hiện một hành động cho mỗi mục menu, chẳng hạn như bắt đầu một mục A khác. Tham chiếu đoạn mã từ tác vụ trước, thay đổi mã cho trường hợp `Action_order` thành mã sau, bắt đầu `OrderActivity` (sử dụng cùng một mã bạn đã sử dụng cho nút hành động nổi trong bài học về cách sử dụng hình ảnh có thể nhấp):

```
switch (item.getItemId()) { trường hợp R.id.action_order: Ý định = new Intent(MainActivity.this, OrderActivity.class); intent.putExtra(EXTRA_MESSAGE, mOrderMessage); startActivity(ý định); trả về true; // ... mã cho các trường hợp khác }
```

Chạy ứng dụng. Nhấp vào biểu tượng giỏ hàng trên thanh ứng dụng (mục O) sẽ đưa bạn trực tiếp đến màn hình OrderActivity.

Mã giải pháp nhiệm vụ 3

Dự án Android Studio: [DroidCafeOptions](#)

Thử thách mã hóa

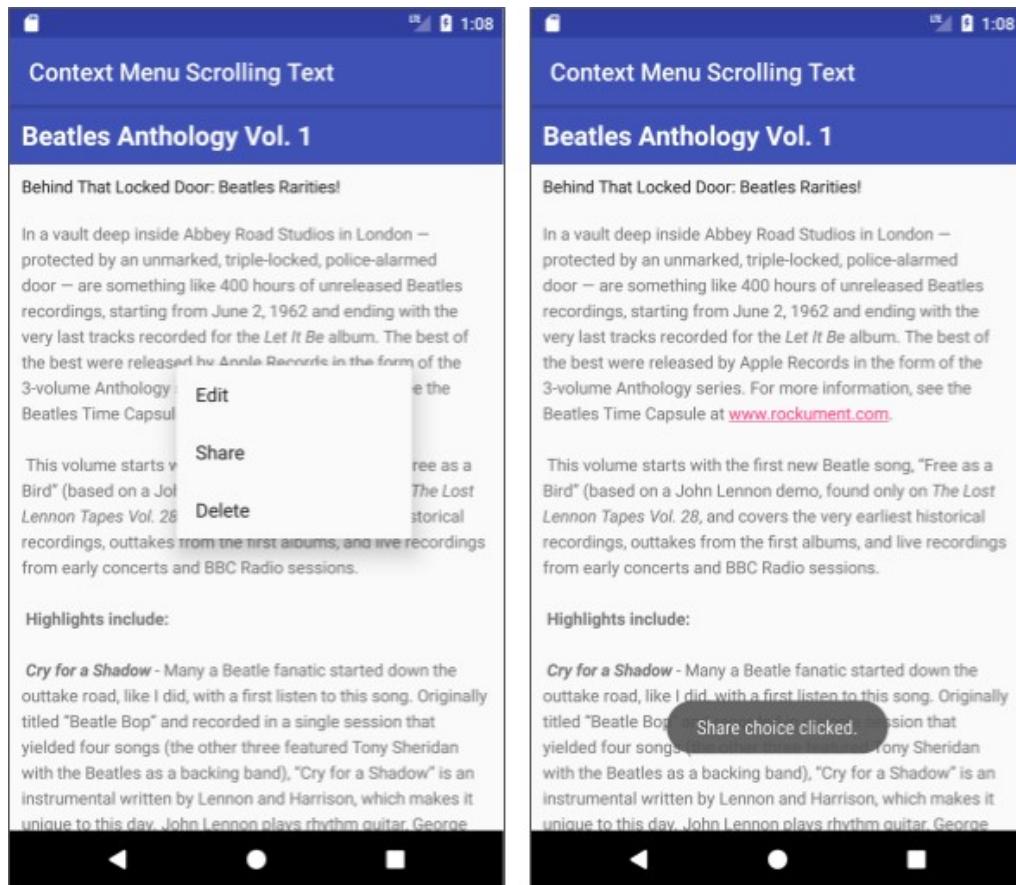
Lưu ý: Tất cả các thử thách mã hóa đều là tùy chọn và không phải là điều kiện tiên quyết cho các bài học sau này.

Thử thách: Menu context cho phép người dùng thực hiện hành động trên View đã chọn. Mặc dù menu tùy chọn trong thanh ứng dụng thường cung cấp các lựa chọn để điều hướng đến một hoạt động khác, nhưng bạn sử dụng menu ngữ cảnh để cho phép người dùng sửa đổi View cho hoạt động hiện tại.

Cả hai menu đều được mô tả bằng XML, cả hai đều được khởi tạo bằng cách sử dụng [MenuItemInflater](#), và cả hai đều sử dụng phương thức "trên mục đã chọn" — trong trường hợp này là [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 nối các mục menu khi người dùng thực hiện chạm và giữ

View, như thể hiện 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: **E dit**, **S hare** và **D elete**, như trong hình bên dưới. Menu xuất hiện khi người dùng thực hiện chạm và giữ trên TextView. Sau đó, ứng dụng sẽ hiển thị thông báo Toast hiển thị tùy chọn menu đã chọn, như được hiển thị ở phía bên phải của hình.



Gợi ý

Menu ngữ cảnh tương tự như menu tùy chọn, với hai điểm khác biệt quan trọng:

- Menu ngữ cảnh phải được đăng ký với View để menu phồng lên khi chạm và giữ xảy ra trên View.
- Mặc dù mã menu tùy chọn được cung cấp bởi mẫu Hoạt động cơ bản, nhưng đối với menu ngữ cảnh, bạn phải tự thêm mã và tài nguyên menu. Để giải quyết thách thức này, hãy làm theo các bước sau:

1. Tạo tệp tài nguyên menu XML cho các mục menu.

Nhấp chuột phải vào thư mục **res** và chọn **New > Thư mục tài nguyên Android**. Chọn **menu** trong menu thả xuống loại **nguồn điện tử R** và nhấp vào **OK**. Sau đó nhấp chuột phải vào thư mục **menu** mới, chọn **N**

ew > Menu tài nguyên file, nhập tên menu_context và nhấp vào OK. Mở menu_context và nhập các mục menu như bạn đã làm với menu tùy chọn.

- Đăng ký View vào menu ngữ cảnh bằng phương thức [registerForContextMenu\(\)](#).

Trong phương thức [onCreate\(\)](#), đăng ký TextView:

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

- Triển khai phương thức [onCreateContextMenu\(\)](#) trong mục A 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);  
}
```

- Triển khai phương thức [onContextItemSelected\(\)](#) trong mục A để xử lý các nhấp chuột vào mục menu.
Trong trường hợp này, chỉ cần hiển thị một chiếc Toast với lựa chọn menu.

```
@Override boolean công khai  
onContextItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        trường hợp R.id.context_edit:  
            displayToast("Đã nhấp vào lựa chọn chỉnh sửa.");  
            trả về true;  
        Trường hợp R.id.context_share:  
            displayToast("Đã nhấp vào lựa chọn chia sẻ.");  
            trả về true;  
        Trường hợp R.id.context_delete:  
            displayToast("Xóa lựa chọn đã nhấp.");  
            trả về true;  
        Mặc định:  
            trả về super.onContextItemSelected(item);  
    }  
}
```

5. Chạy ứng dụng. Nếu bạn chạm và kéo, văn bản sẽ cuộn như trước. Tuy nhiên, nếu bạn nhấn lâu, menu ngữ cảnh sẽ xuất hiện.

Mã giải pháp thách thức

Dự án Android Studio: [ContextMenuScrollingText](#)

Nhiệm vụ 4: Sử dụng hộp thoại để yêu cầu lựa chọn của người dùng

Bạn có thể cung cấp hộp thoại để yêu cầu lựa chọn của người dùng, chẳng hạn như cảnh báo yêu cầu người dùng nhấn vào **OK** hoặc **Hủy**. Dialog là một cửa sổ xuất hiện trên đầu 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 **C liên tục** sau khi đọc hoặc cho người dùng lựa chọn đồng ý với một hành động bằng cách nhấp vào nút tích cực (chẳng hạn như **OK** hoặc **A cept**) hoặc không đồng ý bằng cách nhấp vào nút phủ định (chẳng hạn như **C ancel**). Sử dụng [lớp con AlertDialog](#) của lớp Dialog để hiển thị hộp thoại tiêu chuẩn cho cảnh báo.

Mẹo: Sử dụng hộp thoại một cách tiết kiệm vì chúng làm gián đoạn quy trình làm việc của người dùng. Để biết các phương pháp thiết kế tốt nhất, hãy xem hướng [đẫn Thiết kế vật liệu Dialogs](#). Để biết ví dụ về mã, hãy xem [Dialogs](#) trong tài liệu dành cho nhà phát triển Android.

Trong thực tế này, bạn sử dụng Button 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 việc người dùng nhấn vào thứ gì đó.

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

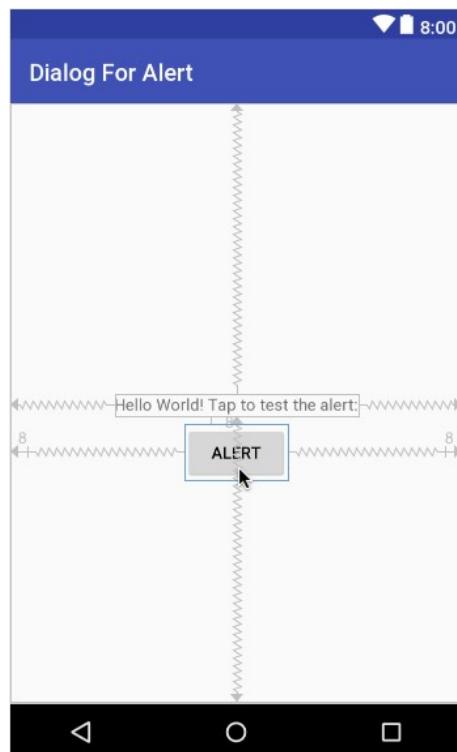
Trong bài tập này, bạn tạo cảnh báo bằng các **nút OK** và **Cancel**. Cảnh báo được kích hoạt bằng cách người dùng nhấn vào một nút.

1. Tạo một dự án mới có tên là **Dialog For Alert** 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 "Hello World!"
4. Thêm một chữ `B` dưới `TextView`. (Tùy chọn: Hạn chế nút ở dưới cùng của `TextView` và các cạnh của bố cục, với lề được đặt thành `8dp`.)

5. Đặt văn bản của Button thành Alert.
6. Chuyển sang tab Text và trích xuất các chuỗi văn bản cho TextView và Button để chuỗi tài nguyên.
7. Thêm thuộc tính android:onClick vào nút để gọi trình xử lý nhấp chuột onClickShowAlert(). Sau khi bạn nhập, 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 *builder* giúp dễ dàng tạo một đối tượng từ một lớp có nhiều thuộc tính bắt buộc và tùy chọn và do đó sẽ yêu cầu rất nhiều tham số để xây dựng. Nếu không có mẫu này, bạn sẽ phải tạo hàm khởi tạo cho sự kết hợp của các thuộc tính bắt buộc và tùy chọn; Với mẫu này, mã dễ đọc và bảo trì hơn. Để biết thêm thông tin về mẫu thiết kế trình tạo, hãy xem Mẫu [Builder](#).

Lớp builder thường là một lớp thành viên tĩnh của lớp mà nó xây dựng. Sử dụng [một AlertDialog.Builder](#) để xây dựng một hộp thoại cảnh báo tiêu chuẩn, với [setTitle\(\)](#) để đặt tiêu đề của nó, [setMessage\(\)](#) để đặt thông báo của nó, và [setPositiveButton\(\)](#) và [setNegativeButton\(\)](#) để đặt các nút của nó.

Để thực hiện cảnh báo, bạn cần tạo một đối tượng của `AlertDialog.Builder`. Bạn sẽ thêm trình xử lý nhấp chuột `onClickShowAlert()` cho `Button`, làm cho đối tượng này làm thứ tự đầu tiên của công việc. Điều đó có nghĩa là hộp thoại sẽ chỉ được tạo khi người dùng nhấp vào nút `Button`. Mặc dù mẫu mã hóa này hợp lý để sử dụng `Button` để kiểm tra cảnh báo, nhưng đối với các ứng dụng khác, bạn có thể muốn tạo hộp thoại trong phương thức `onCreate()` để mã khác luôn kích hoạt hộp thoại. 1. Mở `MainActivity` và thêm phần đầu của phương thức `onClickShowAlert()`:

```
public void onClickShowAlert(Xem chế độ xem) {  
    AlertDialog.Builder myAlertDialog = mới  
        AlertDialog.Builder(MainActivity.this);  
    Đặt tiêu đề hộp thoại và thông điệp.  
}
```

Nếu `AlertDialog.Builder` không được nhận dạng 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 `Activity` của bạn.

2. Thêm mã để đặt tiêu đề và thông báo cho hộp thoại cảnh báo vào `onClickShowAlert()` sau nhận xét:

```
Đặt tiêu đề hộp thoại và thông điệp. myAlertBuilder.setTitle("Cảnh báo");
myAlertBuilder.setMessage("Nhấn vào OK để tiếp tục hoặc Hủy để dừng:");
Thêm các nút hộp thoại.
```

3. Trích xuất các chuỗi ở trên để chuỗi tài nguyên dưới dạng lert_title và lert_message.
4. Thêm các nút **O K** và **C ancel** vào cảnh báo bằng các phương thức setPositiveButton() và setNegativeButton():

```
Thêm các nút hộp thoại. myAlertBuilder.setPositiveButton("OK", mới
DialogInterface.OnClickListener() {
public void onClick(hộp thoại DialogInterface, int which) {
    Người dùng đã nhấn vào nút OK.    Toast.makeText(getApplicationContext(),
    "Nhấn OK",
    Toast.LENGTH_SHORT).show();
} }); myAlertBuilder.setNegativeButton("Hủy",
DialogInterface.OnClickListener() mới {
public void onClick(hộp thoại DialogInterface, int which) {
    Người dùng đã hủy hộp thoại.
    Toast.makeText(getApplicationContext(), "Nhấn hủy",
    Toast.LENGTH_SHORT).show();
}}
```

```
    }  
});  
  
Tạo và hiển thị AlertDialog.
```

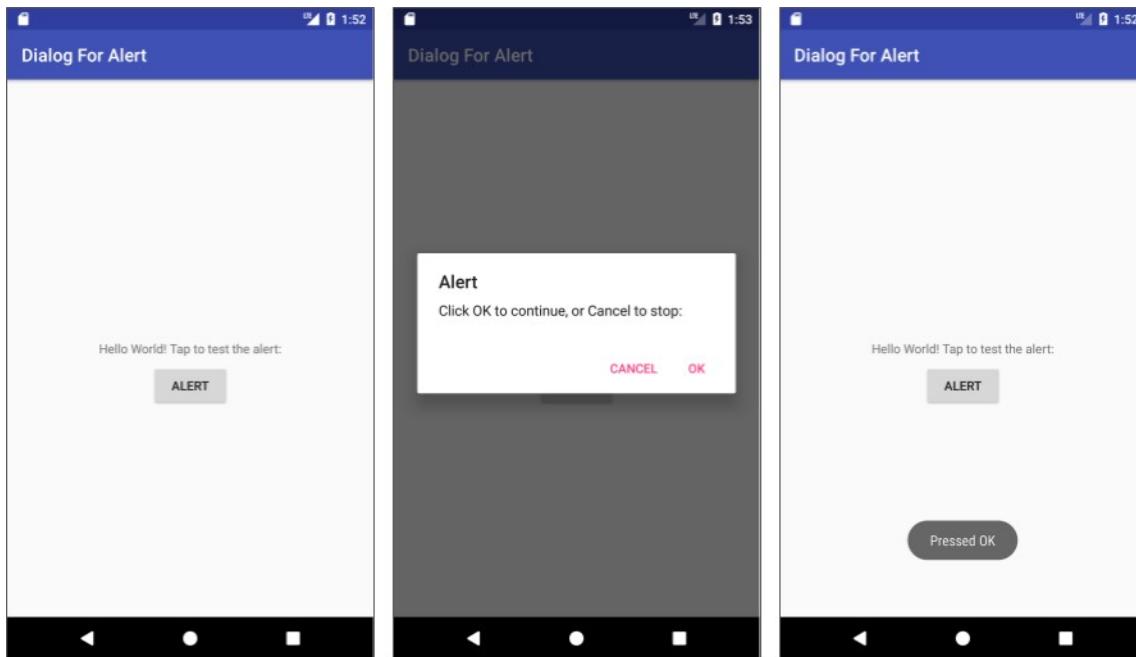
Sau khi người dùng nhấn vào **nút OK** hoặc **C cancel** trong cảnh báo, bạn có thể lấy lựa chọn của người dùng và sử dụng nó trong mã của mình. Trong ví dụ này, bạn hiển thị thông báo Toast.

5. Trích xuất các chuỗi cho **OK** và **C cancel** để chuỗi tài nguyên dưới dạng ok_button và cancel_button, đồng thời trích xuất các chuỗi cho các thông điệp Toast.
6. Ở cuối phương thức `onClickListener()`, thêm `show()`, tạo và sau đó hiển thị hộp thoại cảnh báo:

```
Tạo và hiển thị AlertDialog. myAlertDialog.show();
```

7. Chạy ứng dụng.

Bạn sẽ có thể nhấn vào nút **Alert**, được hiển thị ở phía bên trái của hình ảnh dưới, để xem hộp thoại cảnh báo, được hiển thị ở giữa hình ảnh dưới. Hộp thoại hiển thị các nút **OK** và **C cancel** và thông báo Toast xuất hiện cho biết bạn đã nhấn nút nào, như được hiển thị ở phía bên phải của hình ảnh dưới.



Mã giải pháp nhiệm vụ 4

Dự án Android Studio: [DialogForAlert](#)

Nhiệm vụ 5: Sử dụng bộ chọn cho đầu vào của người dùng

Android cung cấp các hộp thoại sẵn sàng sử dụng, được gọi là pickers, để chọn thời gian hoặc ngày. Bạn có thể sử dụng chúng để đảm bảo rằng người dùng của bạn chọn ngày hoặc giờ hợp lệ được định dạng chính xác và được điều chỉnh theo ngày giờ địa phương 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 [Pickers](#).

Trong nhiệm vụ này, bạn sẽ tạo một dự án mới và thêm bộ chọn ngày. Bạn cũng sẽ học cách sử dụng Fragment, là một hành vi hoặc một phần của giao diện người dùng trong một ctivity A. Nó giống như một kích thước A nhỏ trong phạm vi A chính, với vòng đời riêng của nó và nó được sử dụng để xây dựng một máy hái. Tất cả công việc đã được thực hiện cho bạn. Để tìm hiểu về lớp Fragment, hãy xem [Fragments](#) trong Hướng dẫn API.

Một lợi ích của việc sử dụng Fragment cho bộ chọn là bạn có thể tách các phần mã để quản lý ngày và giờ cho các ngôn ngữ khác nhau hiển thị ngày và giờ theo những cách khác nhau. Cách tốt nhất để hiển thị bộ chọn là sử dụng

một thực thể của [DialogFragment](#), là một lớp con của Fragment. DialogFragment hiển thị một cửa sổ hộp thoại nổi trên đầu cửa sổ Activity. Trong bài tập này, bạn sẽ thêm Fragment cho hộp thoại bộ chọn và sử dụng DialogFragment để quản lý vòng đời hộp thoại.

Mẹo: Một lợi ích khác của việc sử dụng Fragment cho bộ chọn là bạn có thể triển khai các cấu hình bố cục khác nhau, chẳng hạn như hộp thoại cơ bản trên màn hình có kích thước bằng thiết bị cầm tay hoặc một phần nhúng của bố cục trên màn hình lớn.

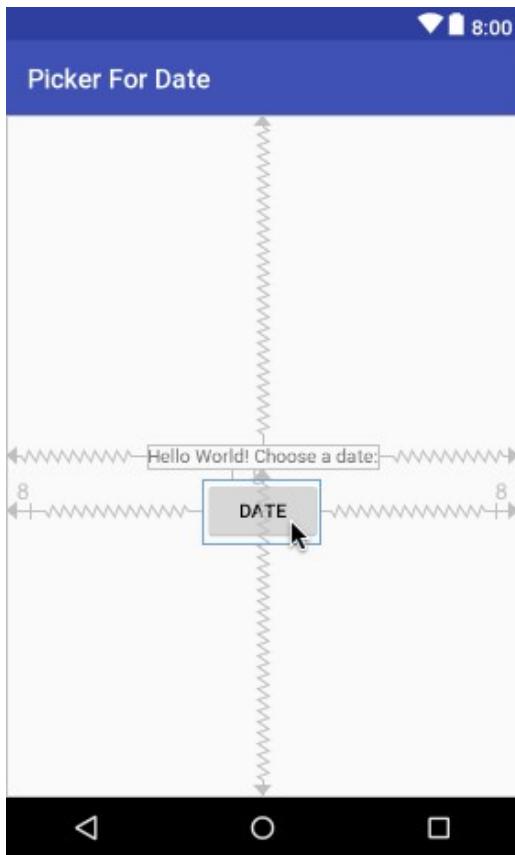
5.1 Tạo ứng dụng mới để hiển thị bộ chọn ngày

Để bắt đầu tác vụ này, hãy tạo một ứng dụng cung cấp nút Button để hiển thị bộ chọn ngày.

1. Tạo một dự án mới có tên Picker For Date 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 một chữ B bên dưới TextView. (Tùy chọn: Hạn chế Button ở dưới cùng của TextView và các cạnh của bố cục, với lề được đặt thành 8dp.)
5. Đặt văn bản của Button thành Date.
6. Chuyển sang tab Text và trích xuất các chuỗi cho TextView và Button để chuỗi tài nguyên.
7. Thêm thuộc tính android:onClick vào nút B để gọi trình xử lý nhấp chuột showDatePicker(). Sau khi nhập, trình xử lý nhấp chuột được gạch chân màu đỏ vì nó chưa được tạo.

```
android:onClick="showDatePicker"
```

Bây giờ bạn sẽ có một bố cục tương tự như sau:



5.2 Tạo một mảng mới cho bộ chọn ngày

Trong bước này, bạn thêm Fragment cho bộ chọn ngày.

1. Mở rộng **pp > java > com.example.android.pickerfordate** và chọn **MainActivity**.
2. Chọn **File > New > Fragment > Fragment (Blank)** và đặt tên cho fragment là **DatePickerFragment**. Xóa cả ba hộp kiểm để bạn không tạo XML bối cảnh, bao gồm các phương thức fragment factory hoặc bao gồm callback giao diện. Bạn không cần tạo bối cảnh cho bộ chọn tiêu chuẩn. Nhấp vào **Finish**.

3. Mở **DatePickerFragment** và chỉnh sửa định nghĩa lớp **DatePickerFragment** để mở rộng **DialogFragment** và triển khai **DatePickerDialog.OnDateSetListener** để tạo bộ chọn ngày tiêu chuẩn với trình nghe. Xem [Pickers](#) để biết thêm thông tin về cách mở rộng **DialogFragment** cho bộ chọn ngày:

```
lớp công cộng DatePickerFragment mở rộng DialogFragment  
implements DatePickerHer.OnDateSetListener {
```

Khi bạn nhập **DialogFragment** và **DatePickerDialog.OnDateSetListener**, Android Studio sẽ tự động thêm một số câu lệnh **import** vào khối **import** ở trên cùng, bao gồm:

```
nhập android.app.DatePickerDialog; nhập  
android.support.v4.app.DialogFragment;
```

Ngoài ra, biểu tượng bóng đèn màu đỏ 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** 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 những điều sau vào khối **import** ở trên cùng:

```
nhập android.widget.DatePicker;
```

Các tham số **onDateSet()** phải là **int i, int i1 và int i2**. Thay đổi tên của các tham số này thành tên dễ đọc hơn:

```
public void onDateSet(DatePicker datePicker,  
                      int năm, int tháng, int ngày)
```

5. Xóa hàm khởi tạo công khai DatePickerFragment() public rỗng.
6. Thay thế toàn bộ phương thức onCreateDialog() bằng [bằng o nCreateDialog\(\)](#) trả về Dialog và chú thích o nCreateDialog() bằng @ NonNull để chỉ ra rằng giá trị return 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 hộp thoại công khai onCreateDialog(Bundle  
savedInstanceState) { }
```

7. Thêm mã sau vào onCreateDialog() để khởi tạo year, month và day từ [Lịch](#) và trả lại hộp thoại và các giá trị này cho Activity. Khi bạn nhập **Calendar.getInstance()**, hãy chỉ định nhập là **java.util.Calendar**.

```
Sử dụng ngày hiện tại làm ngày mặc định trong bộ chọn. Lịch cuối  
cùng c = Calendar.getInstance(); int năm = c.get(Lịch.NĂM); int tháng  
= c.get(Lịch.THÁNG); int ngày = c.get(Calendar.DAY_OF_MONTH);
```

```
Tạo một thực thể mới của DatePickerDialog và trả về nó. trả về mới  
DatePickerDialog(getActivity(), this, year, month, day);
```

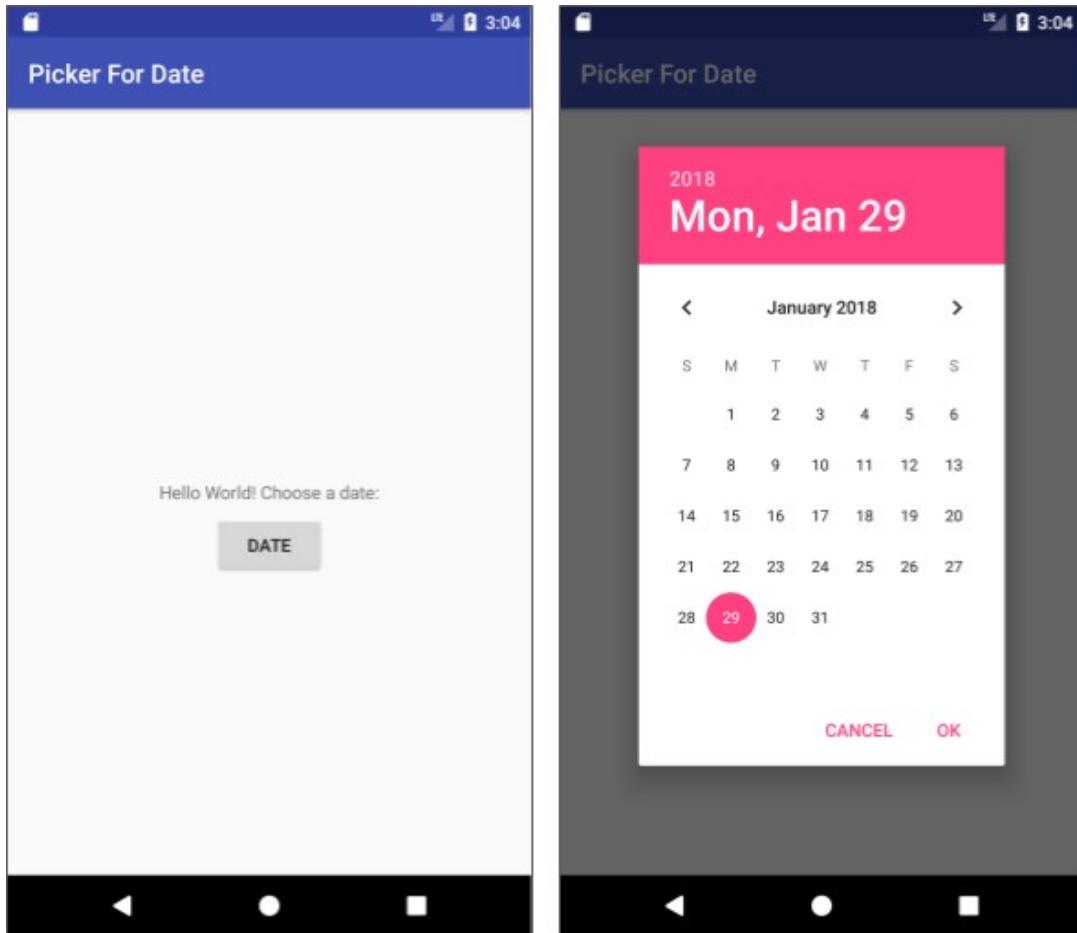
5.4 Sửa đổi hoạt động chính

Mặc dù phần lớn mã trong MainActivity.java vẫn giữ nguyên, nhưng bạn cần thêm một phương thức tạo một thực thể của [FragmentManager](#) để quản lý Fragment và hiển thị bộ chọn ngày.

1. Mở **MainActivity**.
2. Thêm trình xử lý `showDatePicker()` cho **Date Button**. Nó tạo một phiên bản của `FragmentManager` bằng cách sử dụng [getSupportFragmentManager\(\)](#) để tự động quản lý Fragment và hiển thị bộ chọn. Để biết thêm thông tin về lớp Fragment, hãy xem Phân [mảnh](#).

```
public void showDatePicker(Chế độ xem xem) {  
    DialogFragment newFragment = DatePickerFragment() mới;  
    newFragment.show(getSupportFragmentManager(),"datePicker");  
}
```

3. Trích xuất chuỗi " datePicker" vào tài nguyên chuỗi `datepicker`.
4. Chạy ứng dụng. Bạn sẽ thấy bộ chọn ngày sau khi nhấn vào nút **Date**.



5.5 Sử dụng ngày đã chọn

Trong bước này, bạn chuyển ngày trở lại MainActivity.java và chuyển đổi ngày thành một chuỗi mà bạn có thể hiển thị trong tin nhắn Toast.

1. Mở **MainActivity** và thêm một phương thức `processDatePickerResult()` trả về năm, tháng và ngày làm đối số:

```
public void processDatePickerResult(int năm, int tháng, int ngày) { }
```

2. Thêm mã sau vào phương thức `processDatePickerResult()` để chuyển đổi month, day và year thành các chuỗi riêng biệt và nối ba chuỗi bằng dấu gạch chéo cho định dạng ngày của Hoa Kỳ:

```
Chuỗi month_string = Integer.toString(tháng+1);
Chuỗi day_string = Integer.toString(ngày);
Chuỗi year_string = Integer.toString(năm);
Chuỗi dateMessage = (month_string +
    "/" + day_string + "/" + year_string);
```

Số nguyên thứ m được trả về bởi bộ chọn ngày bắt đầu đếm ở 0 cho tháng Giêng, vì vậy bạn cần thêm 1 vào số đó để hiển thị các tháng bắt đầu từ 1.

3. Thêm thông tin sau sau mã trên để hiển thị thông báo Toast:

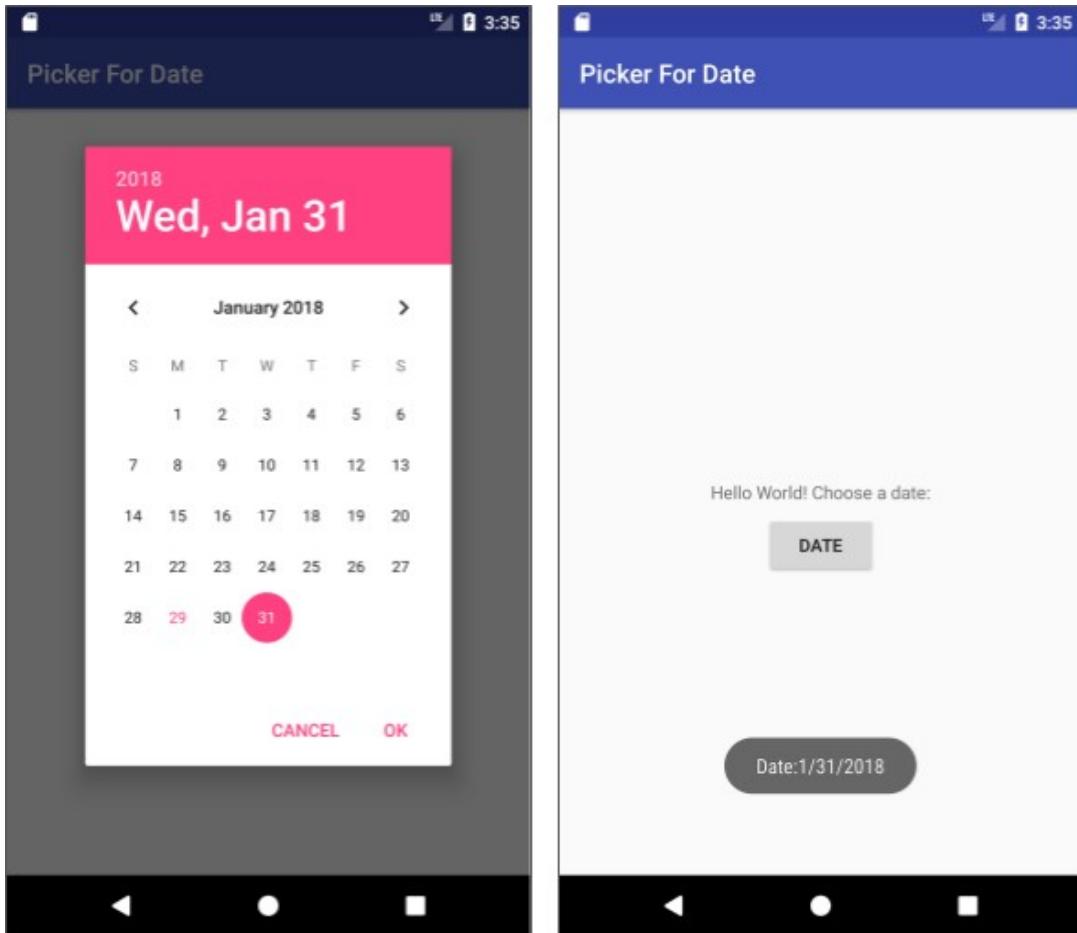
```
Toast.makeText(this, "Date: " + dateMessage,
    Toast.LENGTH_SHORT).show();
```

4. Trích xuất chuỗi mã hóa cứng " Date: " vào một tài nguyên chuỗi có tên date.
5. Mở **DatePickerFragment** và thêm thông tin sau vào phương thức `onDateSet()` để gọi `processDatePickerResult()` trong `MainActivity` và chuyển cho nó year, month và day:

```
@Override trống công khai onDateSet(DatePicker  
datePicker,  
        int năm, int tháng, int ngày) {  
  
    Hoạt động MainActivity = (Hoạt động chính) getActivity();  
    activity.onActivityResult(năm, tháng, ngày);  
}
```

Bạn sử dụng `getActivity()`, khi được sử dụng trong một fragment F, trả về Activity
Fragment hiện được liên kết với. Bạn cần điều này vì bạn không thể gọi một phương thức trong
MainActivity mà không có ngữ cảnh của Main Activity (bạn sẽ phải sử dụng `Intent` thay thế, như bạn đã
học trong một bài học khác). Activity kế thừa ngữ cảnh, vì vậy bạn có thể sử dụng nó làm ngữ cảnh để gọi
phương thức (như trong `Activity.onActivityResult`) .

6. Chạy ứng dụng. Sau khi chọn ngày, ngày xuất hiện trong thông báo Toast như thể hiện ở phía bên phải của hình sau.



Mã giải pháp nhiệm vụ 5

Dự án Android Studio: [PickerForDate](#)

Thử thách mã hóa 2

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.

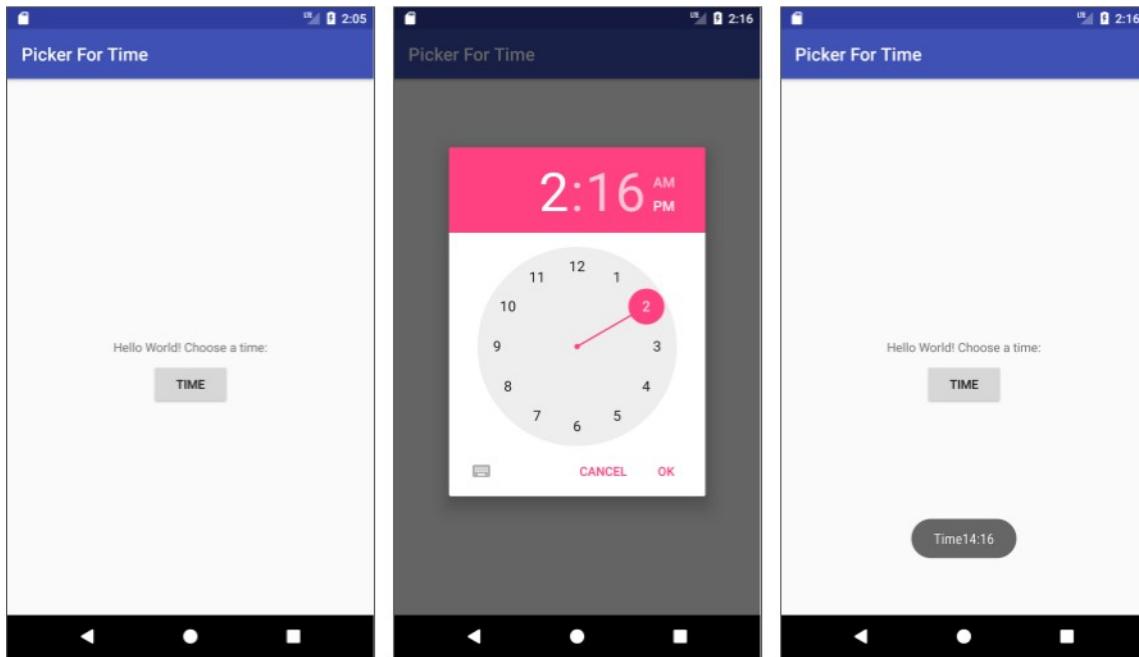
Thử thách: Tạo một ứng dụng có tên **Picker For Time** triển khai bộ chọn thời gian bằng cách sử dụng cùng một kỹ thuật mà bạn vừa học để thêm bộ chọn ngày. Gợi ý:

- Triển khai `TimePickerDialog.OnTimeSetListener` để tạo bộ chọn thời gian tiêu chuẩn với trình 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](#):

```
Lịch cuối cùng c = Calendar.getInstance(); int giờ =  
c.get(Calendar.HOUR_OF_DAY); int phút =  
c.get(Calendar.MINUTE);
```

- Tạo một phương thức `processTimePickerResult()` tương tự như `processDatePickerResult()` trong tác vụ trước đó để 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 **Time** như thể hiện 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.



Mã giải pháp thử thách 2

Dự án Android Studio: [PickerForTime](#)

Tóm tắt

Cung cấp menu tùy chọn và thanh ứng dụng:

- Khởi động ứng dụng của bạn hoặc Activity với 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](#) được nhúng. AppBarLayout giống như một LinearLayout dọc. Nó sử dụng lớp [Toolbar](#) trong thư viện hỗ trợ, thay vì ActionBar gốc, để triển khai thanh ứng dụng.

- Mẫu sửa đổi tệp `AndroidManifest.xml` để tệp `MainActivity` là một Activity được đặt để sử dụng chủ đề `ActionBar`. Chủ đề này được xác định trong tệp `styles.xml`.
- Mẫu đặt `MainActivity` để mở rộng `AppCompatActivity` và bắt đầu với phương thức `onCreate()`, thiết lập chế độ xem nội dung và Toolbar. Sau đó, nó gọi `setSupportActionBar()` và chuyển Toolbar cho nó, đặt Toolbar 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 mà các mục menu xuất hiện trong menu, với số thấp nhất xuất hiện cao hơn trong menu.
- Sử dụng `onOptionsItemSelected()` để xác định mục menu nào đã được nhấn.

Thêm biểu tượng cho mục menu tùy chọn:

- Mở rộng `res` trong `Project > ngăn Android` và nhấp chuột phải (hoặc Control và nhấp chuột) thư mục có thể `rawable`. Chọn `New > Image Asset`.
- Chọn **Thanh ction và Mục tab** trong menu thả xuống và thay đổi tên của tệp hình ảnh.
- Nhấp vào hình ảnh clip art để chọn một hình ảnh clip art làm biểu tượng. Chọn một biểu tượng.
- Chọn **H OLO_DARK** từ **menu thả xuống Theme**.

Hiển thị các mục menu dưới dạng biểu tượng trong thanh ứng dụng:

- Sử dụng thuộc tính `app:showAsAction` trong `menu_main.xml` với các giá trị sau.
- "luôn": Luôn xuất hiện trong thanh ứng dụng. (Nếu không có đủ chỗ, nó có thể trùng lặp với các biểu tượng menu khác.)
- "ifRoom": Xuất hiện trong thanh ứng dụng nếu có chỗ.
- "Không bao giờ": Không bao giờ xuất hiện trong thanh ứng dụng; văn bản của nó xuất hiện trong menu tràn.

Sử dụng hộp thoại cảnh báo:

- Sử dụng hộp thoại để yêu cầu lựa chọn của người dùng, chẳng hạn như cảnh báo yêu cầu người dùng nhấn vào **OK** hoặc **Hủy**. Sử dụng hộp thoại một cách tiết kiệm vì chúng làm gián đoạn quy trình làm việc của người dùng.

- Sử dụng [Một AlertDialog](#) lớp con của Dialog để hiển thị hộp thoại tiêu chuẩn cho cảnh báo. • Dùng [Một AlertDialog.Builder](#) để tạo hộp thoại cảnh báo tiêu chuẩn, với [setTitle\(\)](#) để đặt tiêu đề của nó, [setMessage\(\)](#) để đặt thông điệp của nó và [setPositiveButton\(\)](#) và [setNegativeButton\(\)](#) để đặt các nút của nó.

Sử dụng bộ chọn cho đầu vào của người dùng:

- Sử dụng [DialogFragment](#), một lớp con của [Fragment](#), để tạo một bộ chọn như bộ chọn ngày hoặc bộ chọn thời gian.
- Tạo một DialogFragment và triển khai [DatePickerDialog.OnDateSetListener](#) để tạo một bộ chọn ngày tiêu chuẩn với một trình 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 Hoạt động.
- Tạo một phiên bản của [FragmentManager](#) bằng cách sử dụng [getSupportFragmentManager\(\)](#) để quản lý Fragment và hiển thị bộ chọn ngày.

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [4.3: Menu và bộ chọn](#).

Tìm hiểu thêm

Tài liệu Android Studio:

- [Hướng dẫn sử dụng Android Studio](#)
- [Tạo biểu tượng ứng dụng bằng Image Asset Studio](#)

Tài liệu dành cho nhà phát triển Android:

- [Thêm thanh ứng dụng](#)
- [Menu](#)
- [Thanh công cụ](#)
- [Thư viện hỗ trợ AppCompat V7](#)
- [AppBarLayout](#)
- [onOptionsItemSelected\(\)](#)
- [Cảnh](#)
- [MenuItemInflater](#)
- [registerForContextMenu\(\)](#)
- [onCreateContextMenu\(\)](#)
- [onContextItemSelected\(\)](#)
- [Dialogs](#)
- [Hộp thoại cảnh báo](#)
- [Hái](#)
- [Mảnh](#)
- [Đoạn đối thoại](#)
- [Trình quản lý mảnh](#)
- [Lịch](#)

Thông số kỹ thuật thiết kế vật liệu:

- [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

Homework

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 hiển thị bộ chọn ngày.
2. Hiển thị ngày đã chọn của người dùng trong tin nhắn Toast.

Trả lời những câu hỏi này

Câu hỏi 1

Tên của tệp mà bạn tạo các mục menu tùy chọn là gì? Chọn một:

- menu.java
- menu_main.xml
- activity_main.xml
- content_main.xml

Câu hỏi 2

Phương pháp nào được gọi khi nhấp vào mục menu tùy chọn? Chọn một:

- onOptionsItemSelected(mục MenuItem)
- onClick(Xem chế độ xem)
- onContextItemSelected()
- onClickShowAlert()

Câu hỏi 3

Câu lệnh nào sau đây đặt tiêu đề cho hộp thoại cảnh báo? Chọn một:

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

Câu hỏi 4

Bạn tạo DialogFragment cho bộ chọn ngày ở đâu? Chọn một:

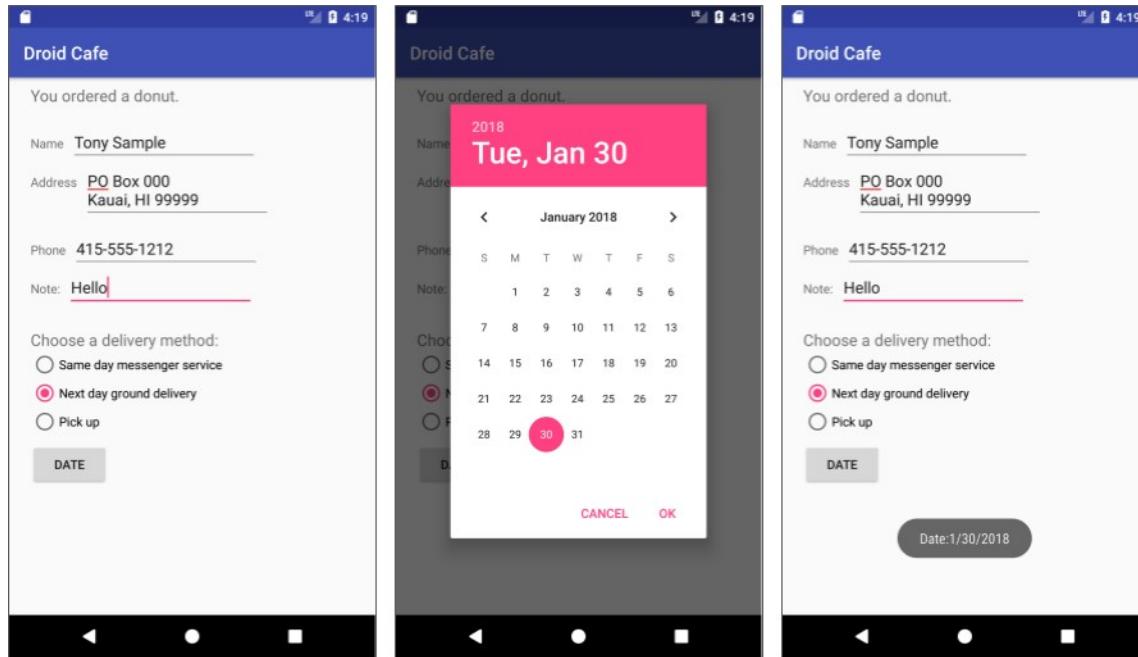
- Trong phương thức onCreate() trong lớp Activity.
- Trong phương thức onCreateContextMenu() trong Fragment.
- Trong phương thức oncreateView() trong phần mở rộng của DialogFragment.
- Trong phương thức onCreateDialog() trong phần mở rộng của DialogFragment.

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:

- Bộ chọn ngày được thêm dưới dạng DialogFragment.
- Nhấp vào nút **Date** (tham khảo phía bên trái của hình bên dưới) trong OrderActivity sẽ hiển thị bộ chọn ngày (tham khảo trung tâm của hình).
- Nhấp vào nút **OK** trong bộ chọn ngày sẽ hiển thị thông báo Toast trong OrderActivity với ngày đã chọn (tham khảo phía bên phải của hình).

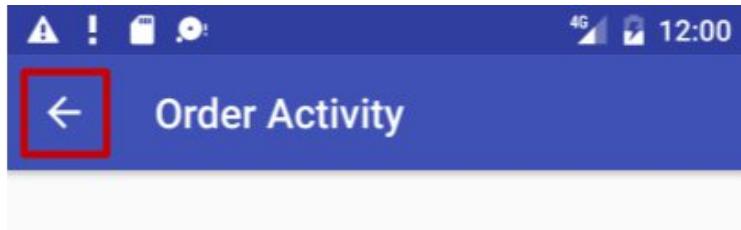


Bài 4.4: Điều hướng người dùng

Giới thiệu

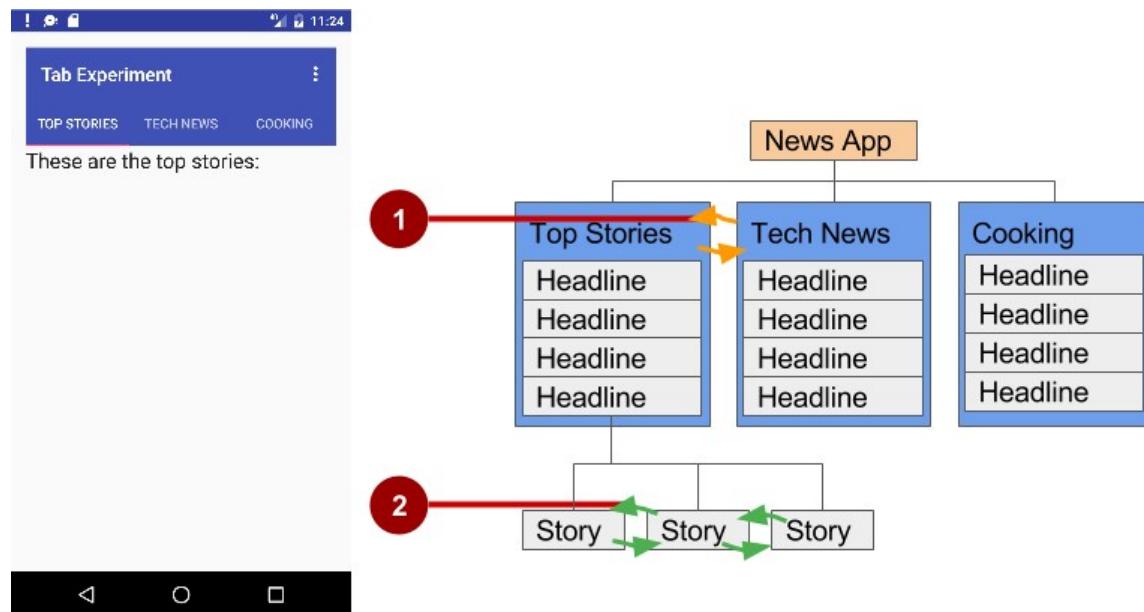
Trong giai đoạn đầu phát triển ứng dụng, bạn nên xác định lộ trình bạn muốn người dùng thực hiện ứng dụng của mình để thực hiện từng tác vụ. (Các nhiệm vụ là những thứ như đặt hàng hoặc duyệt nội dung.) Mỗi đường dẫn cho phép người dùng điều hướng qua, vào và ra khỏi các tác vụ và phần nội dung trong ứng dụng.

Trong thực tế này, bạn tìm hiểu cách thêm nút **Up** (mũi tên hướng trái) vào thanh ứng dụng, như hình dưới đây.



Nút **Up** luôn được sử dụng để điều hướng đến màn hình chính 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.

Thực tế này cũng giới thiệu *điều hướng tab*, trong đó các tab xuất hiện trên 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 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 (**T op Stories**, **T ech News** và **C ooking**) sang màn hình khác
2. Điều hướng ngang từ màn hình câu chuyện này sang màn hình khác

Với các tab, người dùng có thể điều hướng đến và đi từ màn hình anh em mà không cần điều hướng lên màn hình mẹ. Các tab cũng có thể cung cấp điều hướng đến và đi từ các câu chuyện, là màn hình anh em trong **phần mề Top Stories**.

Các tab thích hợp nhất cho bốn màn hình anh chị em trở xuống. Để xem một màn hình khác, người dùng có thể nhấn vào một tab hoặc vuốt sang trái hoặc phải.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các thành phần 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.
- Thêm các mục menu và biểu tượng vào menu tùy chọn trong thanh ứng dụng.

Những gì bạn sẽ học

- Cách thêm nút **Up** vào thanh ứng dụng.
- Cách thiết lập ứng dụng với điều hướng tab và chế độ xem vuốt.

Bạn sẽ làm gì

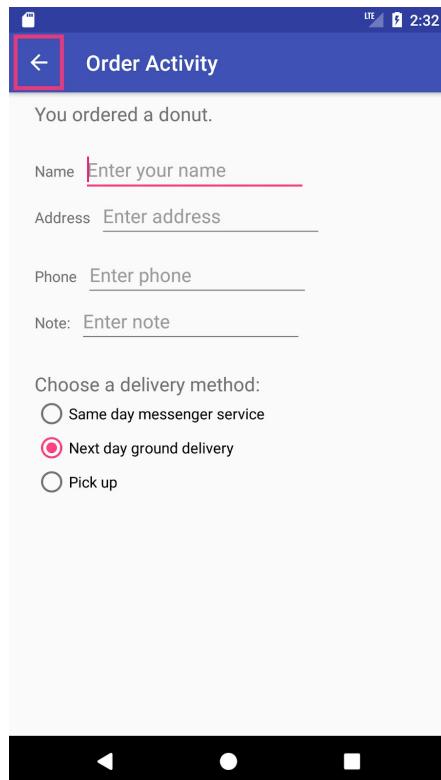
- Tiếp tục thêm các tính năng vào dự án Droid Cafe từ thực tế trước đó.
- Cung cấp nút **Up** trong thanh ứng dụng để điều hướng lên biểu tượng **A** gốc.
- Tạo một ứng dụng mới với các tab để điều hướng **Màn hình Activity** cũng có thể được vuốt.

Tổng quan về ứng dụng

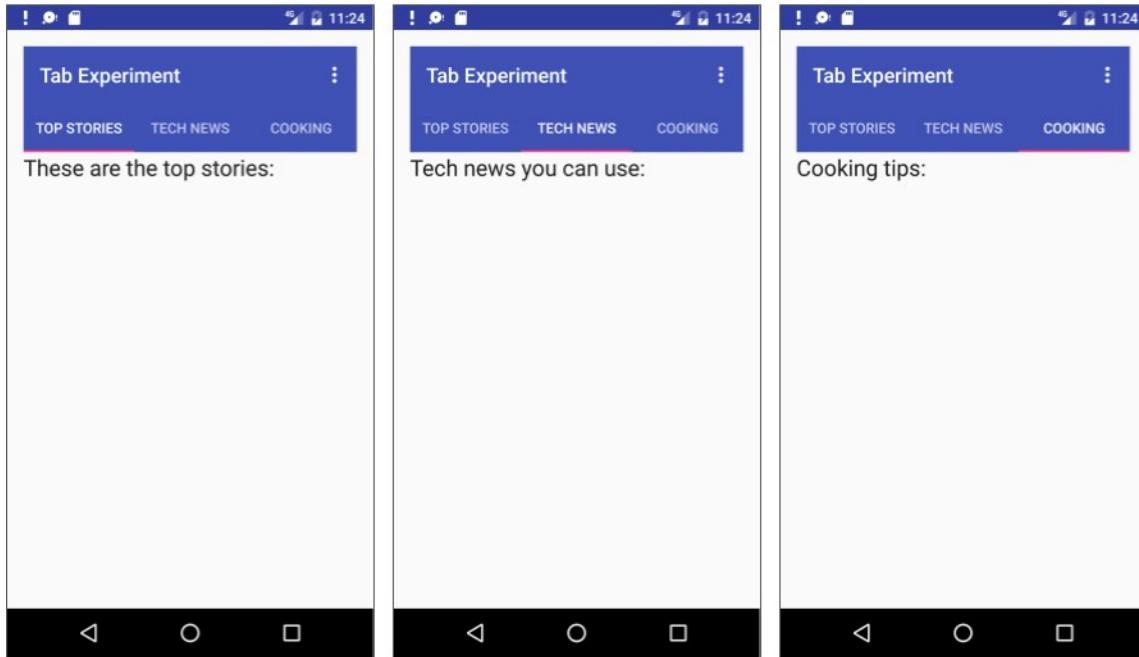
Trong thực tế trước đây về việc sử dụng menu tùy chọn, bạn đã làm việc trên một ứng dụng có tên là Droid Cafe được tạo bằng cách sử dụng mẫu Basic Activity. Mẫu này cung cấp một thanh ứng dụng ở đầu màn hình. Bạn sẽ tìm hiểu

cách thêm nút Up (mũi tên hướng trái) vào thanh ứng dụng để điều hướng lên từ Activity thứ hai (OrderActivity) đến Activity (MainActivity). Điều này sẽ hoàn thành ứng dụng Droid Cafe.

Để bắt đầu dự án từ nơi bạn đã dừng lại trong thực tế 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 ảnh chị em. Khi người dùng nhấn vào một tab, màn hình sẽ hiển thị màn hình nội dung, tùy thuộc vào tab mà người dùng đã nhấn. Người dùng cũng có thể vuốt sang trái và phải để truy cập màn hình nội dung. Các lớp ViewPager tự động xử lý thao tác vuốt của người dùng đến màn hình hoặc các phần tử View.

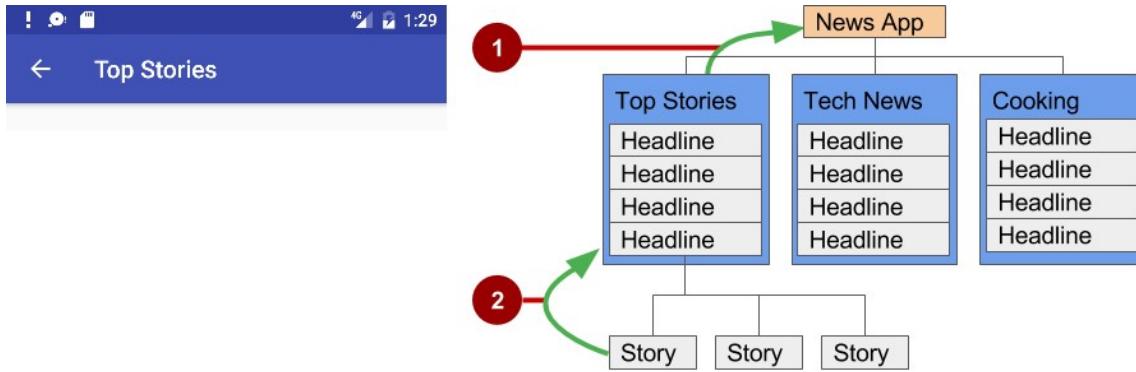


Nhiệm vụ 1: Thêm nút Lên để điều hướng tổ tiên

Ứng dụng của bạn sẽ giúp người dùng dễ dàng tìm đường quay lại màn hình chính của ứng dụng, thường là màn hình A gốc. Một cách để thực hiện việc này là cung cấp nút **Up** trong thanh ứng dụng cho mỗi Hoạt động là con của Activity mẹ.

Nút **Up** cung cấp điều hướng "lên" của tổ tiên, cho phép người dùng chuyển từ trang con sang trang mẹ. Nút **Up** là mũi tên quay mặt 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 **Up**, ứng dụng sẽ điều hướng đến Activity mẹ. Sơ đồ ở phía bên phải của hình bên dưới cho thấy cách nút **Up** được sử dụng 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 anh chị em cấp một đến cha mẹ.
2. Điều hướng từ anh chị em cấp hai đến màn hình con cấp một hoạt động như màn hình cha

Mẹo: Nút Quay lại (hình tam giác ở dưới cùng của thiết bị) và nút **Up** trong giao diện người dùng là hai thứ khác nhau:

Nút Quay lại cung cấp điều hướng đến màn hình đã được xem gần đây nhất. Nếu bạn có một số màn hình con mà người dùng có thể điều hướng bằng cách sử dụng mẫu điều hướng bên (như được mô tả trong phần tiếp theo), nút Quay lại sẽ đưa người dùng trở lại màn hình con trước đó, không phải đến màn hình mẹ.

Để cung cấp điều hướng từ màn hình con trở lại màn hình mẹ, hãy sử dụng nút **Up**. Để biết thêm về điều hướng lên, hãy xem [Điều hướng lên](#).

Như bạn đã học trước đây, khi thêm các hoạt động vào một ứng dụng, bạn có thể thêm điều hướng **nút Up** vào một Activity A con, chẳng hạn như OrderActivity bằng cách khai báo cha của Activity A là MainActivity trong tệp AndroidManifest.xml. Bạn cũng có thể đặt thuộc tính android:label cho tiêu đề cho

Màn hình hoạt động, chẳng hạn như "Hoạt động đặt hàng". Làm theo các bước sau:

1. Nếu bạn chưa mở ứng dụng Droid Cafe từ thực tế trước đó, 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 như sau:

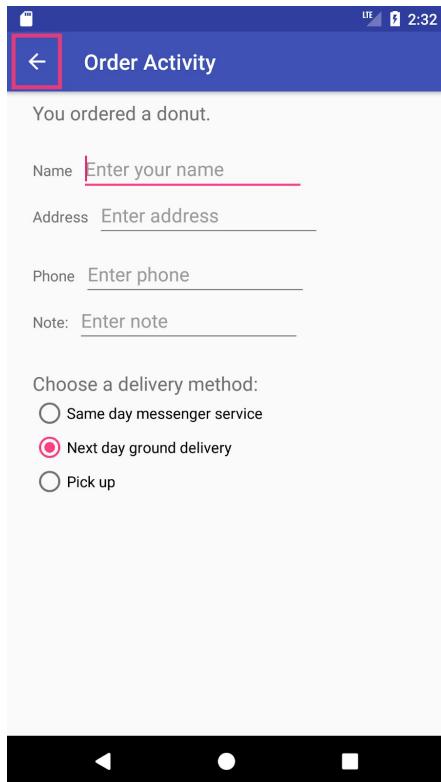
```
<activity android:name="com.example.android.droidcafeinput.OrderActivity"
```

```
    android:label="Hoạt động đặt hàng"
    android:parentActivityName=". Hoạt động chính">
    <meta-data android:name="android.support.PARENT_ACTIVITY"
        android:value=". MainActivity"/> </activity>
```

3. Trích xuất giá trị android:label " Hoạt động đặt hàng" vào tài nguyên chuỗi có tên title_activity_order.

4. Chạy ứng dụng.

Màn hình Hoạt động đặt hàng hiện bao gồm nút **Up** (được đánh dấu trong hình bên dưới) trong thanh ứng dụng để điều hướng trở lại Activity mẹ.

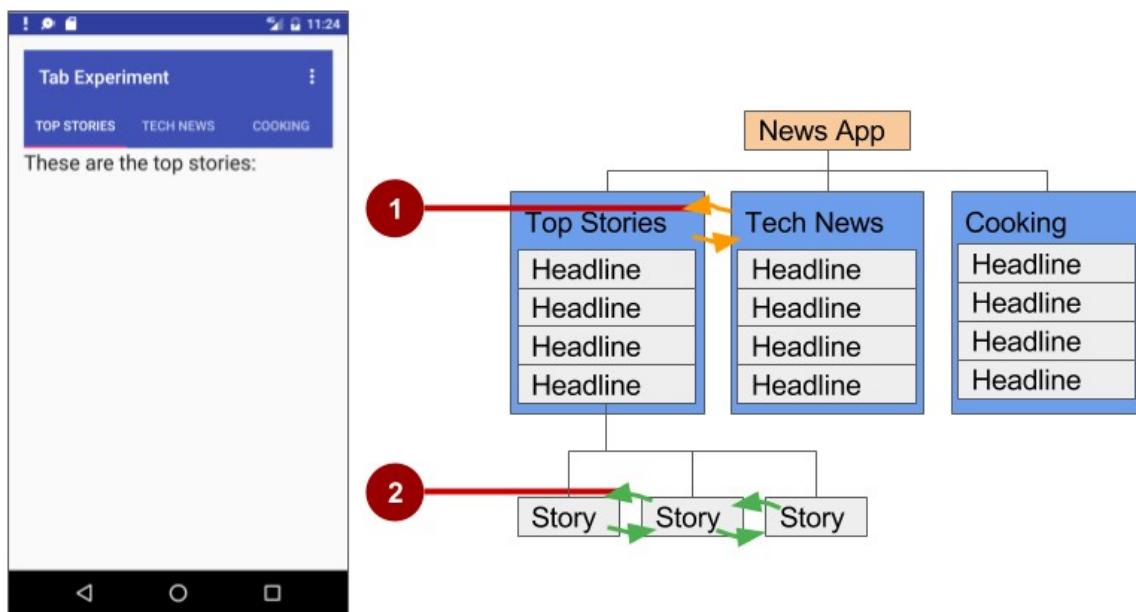


Mã giải pháp nhiệm vụ 1

Dự án Android Studio: [DroidCafeOptionsUp](#)

Nhiệm vụ 2: Sử dụng điều hướng tab với chế độ xem vuốt

Với điều hướng bên, bạn cho phép người dùng di từ anh chị em này sang anh chị em khác (ở cùng cấp độ trong hệ thống phân cấp nhiều tầng). Ví dụ: nếu ứng dụng của bạn cung cấp một số danh mục tin (chẳng hạn như **Top Stories**, **Tech News** và **Cooking**, như trong hình bên dưới), bạn sẽ muốn cung cấp cho người dùng khả năng điều hướng từ danh mục này sang danh mục tiếp theo mà không cần phải điều hướng trở lại màn hình chính. Một ví dụ khác về điều hướng bên là khả năng vuốt sang trái hoặc phải trong cuộc trò chuyện Gmail để xem cuộc trò chuyện mới hơn hoặc cũ hơn trong cùng một 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 câu chuyện này sang màn hình câu chuyện khác

Bạn có thể triển khai điều hướng ngang với `tab` đại diện cho từng màn hình. Các tab xuất hiện trên đầu màn hình, như được hiển thị ở phía bên trái của hình trên, để cung cấp điều hướng đến các màn hình khác. Điều hướng tab là

một giải pháp rất phổ biến để điều hướng ngang từ màn hình con này sang màn hình con khác là một *màn hình con* — ở cùng một vị trí trong hệ thống phân cấp và chia sẻ cùng một màn hình mẹ. Điều hướng tab thường được kết hợp với khả năng vượt màn hình con từ trái sang phải và từ phải sang trái.

Lớp chính được sử dụng để hiển thị các tab là [TabLayout](#) trong Thư viện hỗ trợ thiết kế Android. Nó 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 điều các "trang" màn hình bên trong [ViewPager](#). ViewPager là một trình quản lý bố cục cho phép người dùng lật trái và phải qua màn hình. Đây là một mẫu phổ biến để trình bày các màn hình nội dung khác nhau trong Activity — sử dụng một *dapter* để lấp đầy màn hình nội dung để hiển thị trong Activity và một *trình quản lý layout* thay đổi màn hình nội dung tùy thuộc vào tab nào được chọn.

Bạn triển khai PagerAdapter để tạo các màn hình mà chế độ xem hiển thị. ViewPager thường được sử dụng cùng với [Fragment](#). Bằng cách sử dụng Fragment, bạn có một cách thuận tiện để quản lý vòng đời của một "trang" màn hình.

Để sử dụng các lớp trong Thư viện hỗ trợ Android, hãy thêm `com.android.support:design:x.x.x` (trong đó `x.x.x` là phiên bản mới nhất) vào [tệp build.gradle](#) (Mô-đun: ứng dụng).

Sau đây là các bộ điều hợp tiêu chuẩn để sử dụng các mảnh với ViewPager:

- [FragmentPagerAdapter](#): Được thiết kế để điều hướng giữa các màn hình anh em (trang) đại diện cho một số lượng màn hình cố định, nhỏ.
- [FragmentStatePagerAdapter](#): Được thiết kế để phân trang trên một tập hợp các màn hình (trang) mà số lượng màn hình không được xác định. Nó phá hủy từng Fragment khi người dùng điều hướng đến các màn hình khác, giảm thiểu việc sử dụng bộ nhớ. Ứng dụng cho tác vụ này sử dụng FragmentStatePagerAdapter.

2.1 Tạo dự án và bố cục

1. Tạo dự án mới bằng cách sử dụng mẫu Hoạt động trống. Đặt tên cho ứng dụng là **Thử nghiệm**.
2. Chỉnh sửa [tệp build.gradle](#) (Mô-đun: ứng dụng) và thêm dòng sau vào phần dependencies cho Thư viện hỗ trợ thiết kế Android mà bạn cần để sử dụng [TabLayout](#):

```
trên khai 'com.android.support:design:26.1.0'
```

Nếu Android Studio đề xuất một phiên bản có số cao hơn, hãy chỉnh sửa dòng ở trên để cập nhật phiên bản.

3. Để sử dụng thanh chữ T thay vì thanh ứng dụng và tiêu đề ứng dụng, hãy thêm các thuộc tính sau vào **giá trị res > > styles.xml** tệp để ẩn thanh ứng dụng và tiêu đề:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Các thuộc tính kiểu khác -->
    <item name="windowActionBar">false</item>
    <item name="windowNoTitle">true</item> </style>
```

4. Mở **tệp** bố cục `activity_main.xml` và nhấp vào **tab Text** để xem mã XML.
5. Thay đổi `ConstraintLayout` thành `RelativeLayout`, như bạn đã làm trong các bài tập trước.
6. Thêm thuộc tính `android:id` và `android:padding="16dp"` vào `RelativeLayout`.
7. Xóa `TextView` do mẫu cung cấp và thêm `Toolbar`, `TabLayout` và `ViewPager` trong `RelativeLayout` như được hiển thị trong mã bên dưới.

```
<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/thanh công cụ"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_alignParentTop="đúng"  
        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"
```

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0.
PDF này là Ảnh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2

Trang 127

```
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_below="@+id/thanh công cụ"  
        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>
```

This work is licensed under a Creative Commons Attribution 4.0 International License.
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Page 135

Khi bạn nhập thuộc tính `pp:popupTheme` cho `Toolbar`, `pp` sẽ có màu đỏ nếu bạn không thêm câu lệnh sau vào `RelativeLayout`:

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

Bạn có thể nhấp vào một `pp` và nhấn `Option+Enter` (hoặc `Alt+Enter`) và Android Studio sẽ tự động thêm câu lệnh đó.

2.2 Tạo một lớp và bố cục cho mỗi mảng

Để thêm một mảng đại diện cho từng màn hình theo thẻ, hãy làm theo các bước sau:

1. Nhấp vào `com.example.android.tabexperiment` trong ngăn `Android > Project`.
2. Chọn `File > New > Fragment > Fragment (Blank)`.
3. Đặt tên cho đoạn là `TabFragment1`.
4. Chọn `XML bố cục Create?` sự quyết định.
5. Thay đổi `Tên bố cục Fragment` cho tệp XML thành `tab_fragment1`.
6. Xóa các `phương pháp xuất xưởng mảng bao gồm?` tùy chọn và `tôi bao gồm các lệnh gọi lại giao diện?` sự quyết định. Bạn không cần những phương pháp này.
7. Nhấp vào `Finish`.

Lặp lại các bước trên, sử dụng `TabFragment2` và `TabFragment3` cho Bước 3, và `tab_fragment2` và `tab_fragment3` cho Bước 4.

Mỗi mảng được tạo ra với định nghĩa lớp của nó được đặt để mở rộng `Fragment`. Ngoài ra, mỗi `Fragment inner` bao gồm liên quan đến màn hình (`tab_fragment1`, `tab_fragment2` và `tab_fragment3`), sử dụng mẫu thiết kế thổi phồng tài nguyên quen thuộc mà bạn đã học trong chương trước với menu tùy chọn.

Ví dụ: `TabFragment1` trông như sau:

```
lớp công cộng TabFragment1 mở rộng Fragment {  
    TabFragment1() công khai {  
        Hàm khởi tạo công khai rỗng bắt buộc }  
  
    @Override  
    public View onCreateView(LayoutInflater, vùng chứa ViewGroup,  
        Gói savedInstanceState) {  
        Thổi phồng bố cục cho mảnh này.  
        trả về inflater.inflate(R.layout.tab_fragment1, container, false);  
    }  
}
```

2.3 Chỉnh sửa bố cục mảnh

Chỉnh sửa từng tệp XML bố cục Fragment (tab_fragment1, tab_fragment2 và tab_fragment3):

1. Thay đổi FrameLayout thành RelativeLayout.
2. Thay đổi văn bản TextView thành "Đây là những câu chuyện hàng đầu:" và layout_width và layout_height thành wrap_content.
3. Đặt giao diện văn bản bằng android:textAppearance="?android:attr/textAppearanceLarge".

Lặp lại các bước ở trên cho mỗi tệp XML bố cục mảnh, nhập văn bản khác nhau cho TextView ở bước 2:

- Văn bản cho TextView trong tab_fragment2.xml: "Tin tức công nghệ bạn có thể sử dụng: "
- Văn bản cho TextView trong tab_fragment3.xml: "Mẹo nấu ăn: "

Kiểm tra từng tệp XML bố cục mảnh. Ví dụ, tab_fragment1 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">  
  
    <Chế độ xem văn bản  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Đây là những câu chuyện hàng đầu: "  
        android:textAppearance="?android:attr/textAppearanceLarge"/>  
</RelativeLayout>
```

4. Trong tệp XML bố cục **tab_fragment1**, trích xuất chuỗi cho " Đây là những câu chuyện hàng đầu:" vào tài nguyên chuỗi **tab_1**. Làm tương tự cho các dây trong **tab_fragment2** và **tab_fragment3**.

2.3 Thêm PagerAdapter

Mẫu trình quản lý bố cục bộ điều hợp cho phép bạn cung cấp các màn hình nội dung khác nhau trong một Hoạt động:

- Sử dụng *một dapter* để lấp đầy màn hình nội dung để hiển thị trong Activity.
- Sử dụng trình quản lý thay đổi màn hình nội dung tùy thuộc vào tab được chọn. Làm theo các bước sau để thêm lớp **PagerAdapter** mới vào ứng dụng mở rộng

[FragmentStatePagerAdapter](#) và xác định số lượng tab (m NumOfTabs):

1. Nhấp vào **com.example.android.tabexperiment** trong ngăn **Android > Project**.
2. Chọn **F ile > Lớp Java mới >**.
3. Đặt tên cho lớp **PagerAdapter** và nhập **FragmentStatePagerAdapter** vào trường Superclass. Mục nhập này thay đổi thành **ndroid.support.v4.app.FragmentStatePagerAdapter**.
4. Để các tùy chọn **P ublic** và **N one** được chọn và nhấp vào **OK**

5. Mở **PagerAdapter** trong ngăn **Project > Android**. Một bóng đèn màu đỏ sẽ xuất hiện bên cạnh định nghĩa lớp. Nhấp vào bóng đèn và chọn **Implement methods**, sau đó nhấp vào **OK** để thực hiện các phương thức `getItems()` và `getCount()` đã được chọn.
6. Một bóng đèn đỏ khác sẽ xuất hiện bên cạnh định nghĩa lớp. Nhấp vào bóng đèn và chọn **Create constructor matching super**.
7. Thêm một biến thành viên số nguyên `m NumOfTabs` và thay đổi hàm khởi tạo để sử dụng nó. Bây giờ mã sẽ trông như sau:

```
public class PagerAdapter extends FragmentStatePagerAdapter { int  
    mNumOfTabs;  
  
    public PagerAdapter(FragmentManager fm, int NumOfTabs) {  
        super(fm);  
        this.mNumOfTabs = NumOfTabs;  
    }  
  
    /**  
     *      Trả về Fragment được liên kết với một vị trí được chỉ định.  
     *      Vị trí @param  
     */  
    @Override  
    public Fragment getItem(int position) {  
        trả về null;  
    }  
  
    /**  
     *      Trả về số lượt xem có sẵn.  
     */  
    @Override  
    public int getCount() {  
        trả về 0;  
    }  
}
```

Trong khi nhập mã ở trên, Android Studio sẽ tự động nhập những nội dung sau:

```
nhập android.support.v4.app.Fragment; nhập  
android.support.v4.app.FragmentManager; nhập  
android.support.v4.app.FragmentStatePagerAdapter;
```

Nếu FragmentManager trong mã có màu đỏ, biểu tượng bóng đèn màu đỏ sẽ xuất hiện khi bạn nhấp vào nó. Nhấp vào biểu tượng bóng đèn và chọn **Nhập lớp**. Các lựa chọn nhập xuất hiện. Chọn **FragmentManager (android.support.v4)**.

8. Thay đổi phương thức `getItem()` mới được thêm vào thành như sau, sử dụng khối trường hợp phù thủy để trả về Fragment để hiển thị dựa trên tab được nhấp vào:

```
@Override  
public Fragment getItem(int position) {  
    công tắc (vị trí) {  
        trường hợp 0: trả về mới TabFragment1();  
        trường hợp 1: trả về mới TabFragment2();  
        trường hợp 2: trả về TabFragment3() mới;  
        mặc định: trả về null;  
    }  
}
```

9. Thay đổi phương thức `getCount()` mới được thêm vào như sau để trả về số lượng tab:

```
@Override public int getCount() { return mNumOfTabs;  
}
```

2.4 Thổi phồng thanh công cụ và TabLayout

Vì bạn đang sử dụng các tab nằm gọn bên dưới thanh ứng dụng, bạn đã thiết lập thanh ứng dụng và Thanh công cụ trong bố cục `activity_main.xml` trong bước đầu tiên của nhiệm vụ này. Bây giờ bạn cần thổi phồng Toolbar (sử dụng cùng một phương pháp được mô tả trong chương trước về menu options) và tạo một thực thể của TabLayout để định vị các tab.

1. Mở `MainActivity` và thêm mã sau vào bên trong phương thức `onCreate()` để thổi phồng Thanh công cụ bằng cách sử dụng [setSupportActionBar\(\)](#):

```
protected void onCreate(Bundle savedInstanceState) { // ... Mã bên trong phương thức
onCreate()
    android.support.v7.widget.Toolbar thanh công cụ =
        findViewById(R.id.toolbar);
    setSupportActionBar (thanh công cụ);
    Tạo một phiên bản của bố cục tab từ chế độ xem. }
```

2. Mở `strings.xml` và tạo các tài nguyên chuỗi sau:

```
<string name="tab_label1">Top Stories</string>
<string name="tab_label2">Tin tức công nghệ</string>
<string name="tab_label3">Cooking</string>
```

3. Ở cuối phương thức `onCreate()`, tạo một thực thể của bố cục tab từ `tab_layout_element` trong bố cục và đặt văn bản cho mỗi tab bằng cách sử dụng [ddTab\(\)](#):

```
Tạo một phiên bản của bố cục tab từ chế độ xem. TabLayout tabLayout =  
findViewById(R.id.tab_layout);  
Đặt văn bản cho từng 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));
```

Đặt các tab để lấp đầy toàn bộ bố cục.
tabLayout.setTabGravity(TabLayout.GRAVITY_FILL); Sử dụng
PagerAdapter để quản lý lượt xem trang trong các mảnh.

2.5 Sử dụng PagerAdapter để quản lý màn hình views

1. Bên dưới mã bạn đã thêm vào phương thức `onCreate()` trong tác vụ trước, hãy thêm mã sau để sử dụng `PagerAdapter` để quản lý chế độ xem màn hình (trang) trong các mảnh:

Sử dụng `PagerAdapter` để quản lý lượt xem trang trong các mảnh. Mỗi trang được thể hiện bằng phân đoạn riêng của nó.
`ViewPager` cuối cùng `viewPager = findViewById(R.id.pager);`
bộ điều hợp `PagerAdapter` cuối cùng = `PagerAdapter` mới
`(getSupportFragmentManager(), tabLayout.getTabCount());`
`viewPager.setAdapter (bộ điều hợp);`
Đặt trình nghe cho các nhấp chuột.

2. Ở cuối phương thức `onCreate()`, hãy đặt trình nghe ([TabLayoutOnPageChangeListener](#)) để phát hiện xem một tab có được nhấp vào hay không và tạo phương thức `onTabSelected()` để đặt `ViewPager` thành màn hình tab thích hợp. Mã sẽ trông như sau:

```
Đặt trình nghe cho các nhấp chuột. viewPager.addOnPageChangeListener(mới
    TabLayout.TabLayoutOnPageChangeListener(tabLayout));
tabLayout.addTabSelectedListener(mới
    TabLayout.OnTabSelectedListener() { @Override
        public void onTabSelected(TabLayout.Tab tab)
        { viewPager.setCurrentItem(tab.getPosition());
        }

    @Override
    public void onTabUnselected(TabLayout.Tab tab) {}
```

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

Trang 134

```
@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 sang trái và phải để truy cập các "trang" khác nhau.

Mã giải pháp nhiệm vụ 2

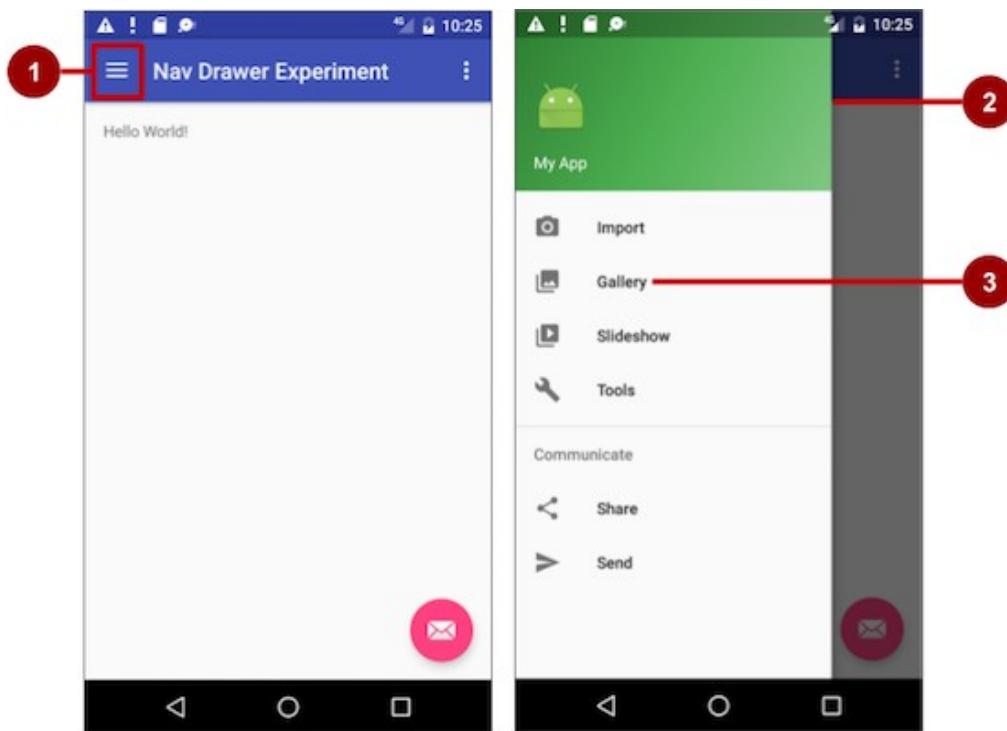
Dự án Android Studio: [TabExperiment](#)

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 này.

Thử thách: C tạo lại một ứng dụng mới với ngăn điều hướng. Khi người dùng nhấn vào lựa chọn ngăn điều hướng, hãy đóng ngắt và hiển thị thông báo Toast cho biết lựa chọn nào đã được chọn.

Ngăn kéo hàng không là một bảng điều khiển thường hiển thị các tùy chọn điều hướng ở cạnh trái của màn hình, như được hiển thị ở phía bên phải của hình bên dưới. Nó được ẩn hầu hết thời gian, nhưng được tiết lộ khi người dùng vuốt ngón tay từ cạnh trái màn hình hoặc chạm vào biểu tượng điều hướng trên thanh ứng dụng, như được hiển thị ở phía bên trái của hình bên dưới.



Trong hình trên:

1. Biểu tượng điều hướng trong thanh ứng dụng
2. Ngăn điều hướng
3. Mục menu ngăn điều hướng

Để tạo ngăn điều hướng trong ứng dụng, bạn cần tạo các bước sau:

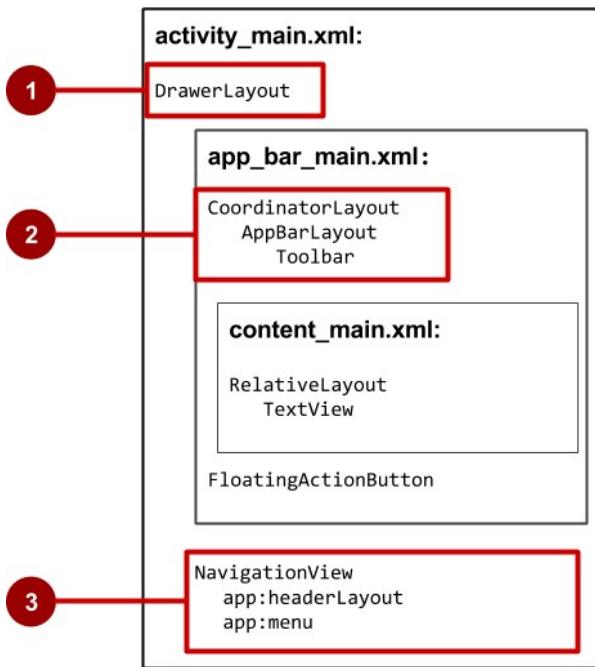
- Một ngăn điều hướng là gốc bố cục Activity ViewGroup.
- Một ngăn điều hướng View cho chính ngăn kéo.
- Bố cục thanh ứng dụng sẽ bao gồm nút biểu tượng điều hướng.
- Bố cục nội dung cho Activity hiển thị ngăn điều hướng.
- Bố cục cho tiêu đề ngăn điều hướng. Sau khi tạo bố cục, bạn cần:
 - Điều menu ngăn điều hướng với tiêu đề và biểu tượng mục.
 - Thiết lập ngăn điều hướng và trình nghe mục trong mã hoạt động.
 - Xử lý các lựa chọn mục menu điều hướng.

Để tạo bố cục ngăn điều hướng, hãy sử dụng API [DrawerLayout](#) có sẵn trong [Thư viện hỗ trợ S](#). Đối với thông số kỹ thuật thiết kế, hãy làm theo các nguyên tắc thiết kế cho ngăn kéo điều hướng trong [hướng dẫn thiết kế](#). Ngăn kéo hàng không N.

Để thêm ngăn điều hướng, hãy sử dụng DrawerLayout làm chế độ xem gốc của bố cục Activity của bạn. Bên trong DrawerLayout, thêm một View chứa nội dung chính cho màn hình (bố cục chính của bạn khi ngăn kéo bị ẩn) và một View khác, thường là NavigationView, chứa nội dung của ngăn điều hướng.

Mẹo: Để làm cho bố cục của bạn dễ hiểu hơn, hãy sử dụng thẻ include để bao gồm một bố cục XML trong một bố cục XML khác.

Hình dưới đây là biểu diễn trực quan của bố cục activity_main.xml và bố cục XML mà nó bao gồm:



Trong hình trên:

1. [DrawerLayout](#) là chế độ xem gốc của bố cục Activity.
2. app_bar_main.xml đi kèm sử dụng [CoordinatorLayout](#) làm gốc và xác định bố cục thanh ứng dụng bằng thanh [chữ T](#) sẽ bao gồm biểu tượng điều hướng để mở ngăn kéo.
3. [NavigationView](#) xác định bố cục ngăn điều hướng và tiêu đề của nó, đồng thời thêm các mục menu vào đó.

Mã giải pháp thách thức

Dự án Android Studio: [NavDrawerExperiment](#)

Tóm tắt

Điều hướng thanh ứng dụng:

- Thêm điều hướng nút lên cho một con Activity bằng cách khai báo cha Activity trong `AndroidManifest.xml` tệp.
- Tuyên bố cha mẹ của đứa trẻ Một Activity trong hoạt động `<activity ... >` phần:

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

Điều hướng tab:

- Tab là một giải pháp tốt cho "điều hướng bên" giữa các chế độ xem anh chị em.
- Lớp chính được sử dụng cho các tab là [TabLayout](#) trong Thư viện hỗ trợ thiết kế Android.
- [ViewPager](#) là một la Yout Manager cho phép người dùng lật trái và phải qua các trang dữ liệu. ViewPager thường được sử dụng cùng với Fragment.
- Sử dụng một trong hai bộ điều hợp tiêu chuẩn để sử dụng ViewPager : [FragmentPagerAdapter](#) hoặc [FragmentStatePagerAdapter](#).

Khái niệm liên quan

Tài liệu khái niệm liên quan có trong [4.4: Điều hướng người dùng](#).

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 vượt bằng các tab](#)
- [Tạo ngăn điều hướng](#)
- [Thiết kế điều hướng Quay lại và Lên](#)
- [Cung cấp điều hướng Lên](#)
- [Triển khai điều hướng hâu duê](#)
- [TabLayout • Ngăn kéo điều hướng • Ngăn kéo bô cục](#)

- [Hỗ trợ thông số kỹ thuật thiết kế vật liệu thư viện:](#)
- [Tìm hiểu về điều hướng](#)
- [Lưới bố cục đáp ứng](#)

Blog dành cho nhà phát triển Android: [Thư viện hỗ trợ thiết kế ndroid](#)

Khác:

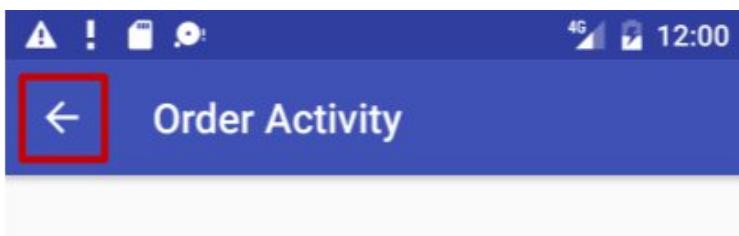
- AndroidHive: [Một thiết kế vật liệu ndroid làm việc với các tab](#)
- Truiton: [Ví dụ về tab ndroid - với Fragments và ViewPager](#)

Homework

Xây dựng và chạy ứng dụng

Tạo một ứng dụng có một Activity A chính và ít nhất ba con Activity khác. Mỗi Activity nên có một menu tùy chọn và sử dụng [thư viện hỗ trợ ứng dụng](#) để tạo thanh ứng dụng, như hình dưới đây.

1. Trong Activity chính, hãy xây dựng bố cục lưới với hình ảnh do bạn chọn. Ba hình ảnh (donut_circle.png, froyo_circle.png và icecream_circle.png), mà bạn có thể [tải xuống](#), được cung cấp như một phần của ứng dụng DroidCafe.
2. Thay đổi kích thước hình ảnh nếu cần, sao cho ba trong số chúng vừa với chiều ngang trên màn hình trong bố cục lưới.
3. Cho phép mỗi hình ảnh cung cấp điều hướng cho một con Activity. Khi người dùng nhấn vào hình ảnh, nó sẽ bắt đầu một con Activity. Từ mỗi con Activity, người dùng sẽ có thể nhấn vào nút Up trong thanh ứng dụng (được đánh dấu trong hình bên dưới) để quay lại Activity chính.



Trả lời những câu hỏi này

Câu hỏi 1

Mẫu nào cung cấp Activity với menu tùy chọn và thanh công cụ thư viện hỗ trợ [Ứng dụng v7](#) làm thanh ứng dụng? Chọn một:

- Mẫu Hoạt động trống
- Mẫu Hoạt động cơ bản
- Mẫu Hoạt động ngăn điều hướng
- Hoạt động điều hướng dưới cùng

Câu hỏi 2

Bạn cần phụ thuộc nào để sử dụng [abLayout](#)? Chọn một:

- com.android.support:thiết kế
- com.android.support.constraint:bối cục ràng buộc
- Junit:Junit:4.12
- com.android.support.test:người chạy

Câu hỏi 3

Bạn định nghĩa mỗi con Activity và cha Activity ở đâu để cung cấp điều **hướng Up**? Chọn một:

- Để cung cấp nút Up cho màn hình con Activity, hãy khai báo cha Activity trong phần con Activity của tệp Activity_main.xml.
- Để cung cấp nút Up cho màn hình con Activity, khai báo cha Activity trong tệp bối cục XML "chính" cho màn hình con Activity.
- Để cung cấp nút Up cho màn hình con Activity, khai báo cha Activity trong phần con Activity của tệp AndroidManifest.xml.
- Để cung cấp nút Up cho màn hình con Activity, khai báo cha Activity trong phần Activity cha của tệp AndroidManifest.xml.

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:

- A GridLayout trong tệp content_main.xml.
- Một phương thức Intent và startActivity() mới cho mỗi phần tử điều hướng trong lưới. • Một Activity riêng biệt cho từng phần tử điều hướng trong lưới.

Bài 4.5: RecyclerView

Giới thiệu

Cho phép người dùng hiển thị, cuộn và thao tác danh sách các mục dữ liệu tương tự là một tính năng phổ biến của ứng dụng. Ví dụ về danh sách có thể cuộn bao gồm danh sách liên hệ, danh sách phát, trò chơi đã lưu, thư mục ảnh, từ điển, danh sách mua sắm và chỉ mục tài liệu.

Trong thực tế về chế độ xem cuộn, bạn sử dụng ScrollView để cuộn View hoặc ViewGroup. ScrollView rất dễ sử dụng, nhưng nó không được khuyến khích cho các danh sách dài, có thể cuộn.

[RecyclerView](#) là một lớp con của ViewGroup và là một cách tiết kiệm tài nguyên hơn để hiển thị các danh sách có thể cuộn. Thay vì tạo View cho mỗi mục có thể hiển thị hoặc không hiển thị trên màn hình, RecyclerView tạo một số mục danh sách giới hạn và sử dụng lại chúng cho nội dung hiển thị.

Trong thực tế này, bạn làm như sau:

- Sử dụng RecyclerView để hiển thị danh sách có thể cuộn.
- Thêm trình xử lý nhấp chuột vào mỗi mục danh sách.
- Thêm các mục vào danh sách bằng nút [hành động floating \(FAB\)](#), nút màu hồng trong ảnh chụp màn hình trong phần tổng quan về ứng dụng. Sử dụng FAB cho hành động chính mà bạn muốn người dùng thực hiện.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các phần tử giao diện người dùng bằng trình soạn thảo bố cục, nhập trực tiếp mã XML và truy cập các phần tử từ mã Java của bạn.
- Tạo và sử dụng tài nguyên chuỗi.
- Chuyển đổi văn bản trong View thành một chuỗi bằng cách sử dụng [getText\(\)](#).
- Thêm trình xử lý `onClick()` vào View.
- Hiển thị thông báo Toast.

Những gì bạn sẽ học

- Cách sử dụng lớp [RecyclerView](#) để hiển thị các mục trong danh sách có thể cuộn.
- Cách tự động thêm các mục vào RecyclerView khi chúng hiển thị thông qua cuộn.
- Cách thực hiện một hành động khi người dùng nhấn vào một mục cụ thể.
- Cách hiển thị FAB và thực hiện một hành động khi người dùng nhấn vào nó.

Bạn sẽ làm gì

- Tạo một ứng dụng mới sử dụng [Recycler View](#) để hiển thị danh sách các mục dưới dạng danh sách có thể cuộn và liên kết hành vi nhấp chuột với các mục trong danh sách.
- Sử dụng FAB để cho phép người dùng thêm các mục vào Recycler View.

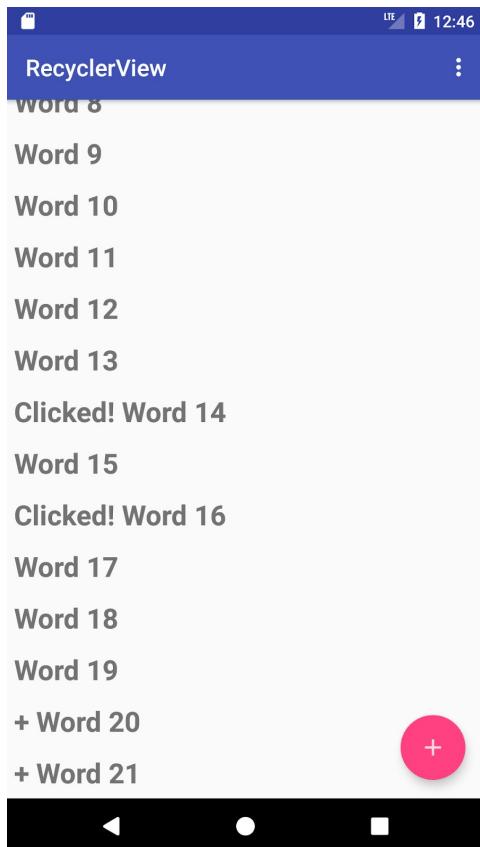
Tổng quan về ứng dụng

Ứng dụng RecyclerView trình bày cách sử dụng [Recycler View](#) để hiển thị danh sách dài các từ có thể cuộn được.

Bạn tạo tập dữ liệu (các từ), chính RecyclerView và các hành động mà người dùng có thể thực hiện:

- Chạm vào một từ đánh dấu từ đó đã nhấp vào.
- Nhấn vào nút hành động nổi (FAB) sẽ thêm một từ mới.

Trang 143



Nhiệm vụ 1: Tạo dự án và tập dữ liệu mới

Trước khi có thể hiển thị RecyclerView, bạn cần dữ liệu để hiển thị. Trong nhiệm vụ này, bạn sẽ tạo một dự án mới cho ứng dụng và một tập dữ liệu. Trong một ứng dụng phức tạp hơn, dữ liệu của bạn có thể đến từ bộ nhớ trong (tệp, cơ sở dữ liệu SQLite, tùy chọn đã lưu), từ một ứng dụng khác (Danh bạ, Ảnh) hoặc từ internet (bộ nhớ đám mây, Google Trang tính hoặc bất kỳ nguồn dữ liệu nào có API). Lưu trữ và truy xuất dữ liệu là một chủ đề riêng được đề cập trong chương lưu trữ dữ liệu. Đối với bài tập này, bạn sẽ mô phỏng dữ liệu bằng cách tạo nó trong phương thức `onCreate()` của MainActivity.

1.1. Tạo dự án và bố cục

1. Khởi động Android Studio.
2. Tạo một dự án mới có tên `RecyclerView`, chọn mẫu **Hoạt động Basic** và tạo tệp bố cục.

Mẫu Hoạt động cơ bản, được giới thiệu trong chương về sử dụng hình ảnh có thể nhấp, cung cấp nút hành động nổi (FAB) và thanh ứng dụng trong bố cục Activity (activity_main.xml) và bố cục cho nội dung Activity (content_main.xml).

3. Chạy ứng dụng của bạn. Bạn sẽ thấy tiêu đề ứng dụng RecyclerView và "Hello World" trên màn hình.
Nếu bạn gặp lỗi liên quan đến Gradle, hãy đồng bộ hóa dự án của bạn như mô tả trong phần thực hành cài đặt Android Studio và chạy Hello World.

1.2. Thêm mã để tạo dữ liệu

Trong bước này, bạn tạo một List `LinkedList` gồm 20 chuỗi từ kết thúc bằng số tăng dần, như trong `["Word 1", "Từ 2", "Từ 3", ...]`.

1. Mở `MainActivity` và thêm một biến thành viên riêng cho danh sách liên kết m WordList.

```
public class MainActivity extends AppCompatActivity {  
    riêng tư cuối cùng LinkedList<String> mWordList = mới LinkedList<>();  
    // ... Phần còn lại của mã MainActivity ... }
```

2. Thêm mã trong phương thức `onCreate()` điền m WordList bằng các từ:

```
Đưa dữ liệu ban đầu vào danh sách từ. for (int i = 0; i  
< 20; i++) {  
    mWordList.addLast("Từ " + i); }
```

Mã nối chuỗi "Word" với giá trị `i` trong khi tăng giá trị của nó. Đây là tất cả những gì bạn cần làm tập dữ liệu cho bài tập này.

1.3. Thay đổi biểu tượng FAB

Đối với thực tế này, bạn sẽ sử dụng FAB để tạo một từ mới để chèn vào danh sách. Mẫu Basic Activity cung cấp một FAB, nhưng bạn có thể muốn thay đổi biểu tượng của nó. Như bạn đã học trong một bài học khác, bạn có thể chọn một biểu tượng từ tập hợp các biểu tượng trong Android Studio cho FAB. Làm theo các bước sau:

1. Mở rộng **res** trong **Project > ngăn Android** và nhấp chuột phải (hoặc Control và nhấp chuột) thư mục có thể **drawable**.
2. Chọn **New > Image Asset**. Hộp thoại Định cấu hình nội dung hình ảnh xuất hiện.
3. Chọn **Thanh ction và Mục tab** trong menu thả xuống ở đầu hộp thoại.
4. Thay đổi **ic_action_name** trong trường **Name** thành **ic_add_for_fab**.

5. Nhấp vào hình ảnh clip art (logo Android bên cạnh **C lipart:**) để chọn hình ảnh clip art làm biểu tượng. Một trang biểu tượng xuất hiện. Nhấp vào biểu tượng bạn muốn sử dụng cho FAB, chẳng hạn như dấu cộng (+).
6. Chọn **H OLO_DARK** từ menu thả xuống **T heme**. Thao tác này đặt biểu tượng thành màu trắng trên nền tối (hoặc đen). Nhấp vào **N ext**.
7. Nhấp vào **F inish** trong hộp thoại Confirm Icon Path.

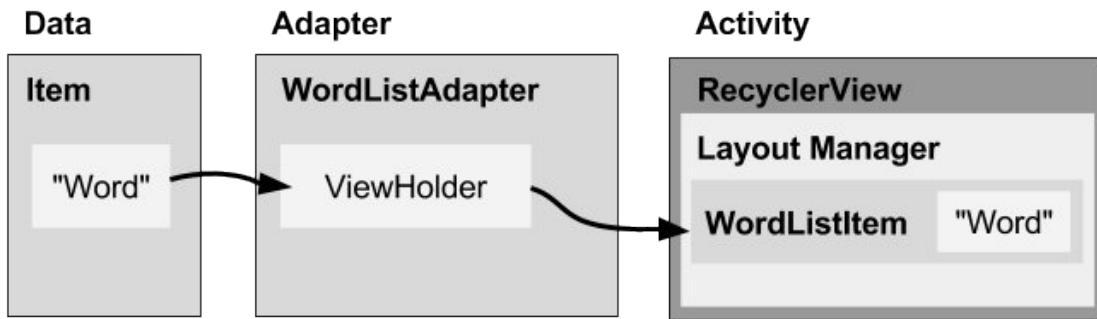
Mẹo: Để biết mô tả đầy đủ về cách thêm biểu tượng, hãy xem bài viết [C hỉnh sửa các biểu tượng ứng dụng bằng Image Asset Studio](#).

Nhiệm vụ 2: Tạo RecyclerView

Trong thực tế này, bạn hiển thị dữ liệu trong RecyclerView. Bạn cần những điều sau:

- Dữ liệu để hiển thị: Sử dụng m WordList.
- RecyclerView cho danh sách cuộn chứa các mục danh sách.
- Bố cục cho một mục dữ liệu. Tất cả các mục trong danh sách trông giống nhau.
- Trình quản lý bố cục. [R eyclerView.LayoutManager](#) xử lý hệ thống phân cấp và bố cục của các phần tử View. RecyclerView yêu cầu một trình quản lý bố cục rõ ràng để quản lý sự sắp xếp của các mục danh sách có trong đó. Bố cục này có thể là dọc, ngang hoặc lướt. Bạn sẽ sử dụng [L inearLayoutManager doc](#).
- Một bộ chuyển đổi. [R eyclerView.Adapter](#) kết nối dữ liệu của bạn với RecyclerView. Nó chuẩn bị dữ liệu trong [R eyclerView.ViewHolder](#). Bạn sẽ tạo một bộ chuyển đổi chèn vào và cập nhật các từ đã tạo trong chế độ xem của bạn.
- Một Viewholder. Bên trong bộ điều hợp của mình, bạn sẽ tạo một Viewholder chứa thông tin View để hiển thị một mục từ bố cục của mục.

Sơ đồ dưới đây cho thấy mối quan hệ giữa dữ liệu, bộ điều hợp, Viewholder và trình quản lý bố cục.



Để thực hiện các phần này, bạn sẽ cần:

1. Thêm phần tử RecyclerView vào bố cục nội dung MainActivity XML (content_main.xml) cho ứng dụng RecyclerView.
2. Tạo một tệp bố cục XML (wordlist_item.xml) cho một mục danh sách, đó là WordListItem.
3. Tạo bộ điều hợp (WordListAdapter) với ViewHolder (WordViewHolder). Thực hiện phương thức lấy dữ liệu, đặt nó vào ViewHolder và cho người quản lý bố cục biết để hiển thị dữ liệu đó.
4. Trong phương thức onCreate() của MainActivity, tạo RecyclerView và khởi tạo nó bằng bộ điều hợp và trình quản lý bố cục tiêu chuẩn.

Hãy làm từng cái một.

2.1. Sửa đổi bố cục trong content_main.xml

Để thêm phần tử RecyclerView vào bố cục XML, hãy làm theo các bước sau:

1. Mở content_main.xml trong ứng dụng RecyclerView của bạn. Nó hiển thị một TextView "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 phần tử 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 Thư viện hỗ trợ.

creative Commons để cập

nhật mới nhất.

2.2. Tạo bố cục cho một mục danh sách

Bộ điều hợp cần bố cục cho một mục trong danh sách. Tất cả các mục đều sử dụng cùng một bố cục. Bạn cần chỉ định bố cục mục danh sách đó trong một tệp tài nguyên bố cục riêng biệt, vì nó được sử dụng bởi bộ điều hợp, tách biệt với RecyclerView.

Tạo bố cục mục từ đơn giản bằng cách sử dụng LinearLayout dọc với TextView:

1. Nhấp chuột phải vào **thư mục app > res > layout** và chọn **New > Layout resource file**.
2. Đặt tên cho tệp **wordlist_item** và nhấp vào **OK**.
3. Trong tệp bố cục mới, nhấp vào tab **Text** để hiển thị mã XML.
4. Thay đổi ConstraintLayout đã được tạo với tệp thành LinearLayout với các thuộc tính sau (trích xuất tài nguyên khi bạn tiếp tục):

Thuộc tính LinearLayout	Giá trị
Android:layout_width	"match_parent"
Android:layout_height	"wrap_content"
Android:Định hướng	"dọc"
android:đệm	"6 dtk"

5. Thêm TextView cho từ vào LinearLayout. Sử dụng word làm ID của từ:

Thuộc tính	Giá trị
android:id	"@+id/từ"

Android:layout_width	"match_parent"
Android:layout_height	"wrap_content"
android:kích thước văn bản	"24sp"
android:textStyle	"đậm"

2.3 Tạo kiểu từ các thuộc tính TextView

Bạn có thể sử dụng kiểu để cho phép các phần tử chia sẻ các nhóm thuộc tính hiển thị. Một cách dễ dàng để tạo kiểu là trích xuất kiểu của phần tử giao diện người dùng mà bạn đã tạo. Để trích xuất thông tin kiểu cho `TextView` trong `wordlist_item.xml`:

1. Mở `wordlist_item.xml` nếu nó chưa mở.
2. Nhấp chuột phải (hoặc Control khi nhấp vào) `TextView` bạn vừa tạo trong `wordlist_item.xml` và chọn **R**efactor > Extract > Style. Hộp thoại **E**xtract **A**ndroid **S**tyle xuất hiện.
3. Đặt tên cho kiểu của bạn **w ord_title** và để tất cả các tùy chọn khác được chọn. Chọn tùy chọn **Launch 'Sử dụng phong cách nếu có thể'**. Sau đó nhấp vào **OK**.
4. Khi được nhắc, hãy áp dụng kiểu cho **Dự án lõi W**.
5. Tìm và kiểm tra phong cách `word_title` trong `values > styles.xml`.
6. Mở lại `wordlist_item.xml` nếu nó chưa mở. `TextView` hiện sử dụng kiểu thay cho các thuộc tính tạo kiểu riêng lẻ, như được hiển thị bên dưới.

```
<?xml version="1.0" encoding="utf8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="dọc"
    android:padding="6dp">

    <Chè độ xem văn bản
        android:id="@+id/từ"
        style="@style/word_title" />
</LinearLayout>
```

2.4. Tạo bộ điều hợp

Android sử dụng bộ điều hợp (từ [lớp Adapter](#)) để kết nối dữ liệu với các mục View trong danh sách. Có nhiều loại bộ điều hợp khác nhau có sẵn và bạn cũng có thể viết bộ điều hợp tùy chỉnh. Trong nhiệm vụ này, bạn sẽ tạo một bộ điều hợp liên kết danh sách các từ của bạn với các mục danh sách từ View.

Để kết nối dữ liệu với các mục View, bộ điều hợp cần biết về các mục View. Bộ điều hợp sử dụng ViewHolder mô tả mục View và vị trí của nó trong RecyclerView.

Đầu tiên, bạn sẽ xây dựng một bộ điều hợp thu hẹp khoảng cách giữa dữ liệu trong danh sách từ của bạn và RecyclerView hiển thị:

1. Nhấp chuột phải vào `java/com.android.example.recyclerview` và chọn `New > Java Class`.
2. Đặt tên cho lớp `WordListAdapter`.
3. Cung cấp cho WordListAdapter chữ ký sau:

```
lớp công cộng WordListAdapter mở rộng
RecyclerView.Adapter<WordListAdapter.WordViewHolder> { }
```

WordListAdapter mở rộng bộ điều hợp chung cho RecyclerView để sử dụng giá đỡ View dành riêng cho ứng dụng của bạn và được xác định bên trong WordListAdapter. WordViewHolder hiển thị lỗi vì bạn chưa định nghĩa lỗi đó.

4. Bấm vào khai báo lớp (**WordListAdapter**) , sau đó bấm vào bóng đèn màu đỏ ở phía bên trái của ngăn. Chọn **phương pháp tối implement**.

Một hộp thoại xuất hiện yêu cầu bạn chọn phương pháp để thực hiện. Chọn cả ba phương pháp và nhấp vào **OK**

Android Studio tạo trình giữ chỗ trống cho tất cả các phương thức. Lưu ý cách onCreateViewHolder và onBindViewHolder đều tham chiếu đến WordViewHolder, chưa được triển khai.

2.5 Tạo ViewHolder cho bộ điều hợp

Để tạo ViewHolder, hãy làm theo các bước sau:

1. Bên trong lớp WordListAdapter, thêm một lớp bên trong WordViewHolder mới với chữ ký này:

```
lớp WordViewHolder mở rộng RecyclerView.ViewHolder {}
```

Bạn sẽ thấy lỗi về một hàm khởi tạo mặc định bị thiếu. Bạn có thể xem chi tiết về lỗi bằng cách di con trỏ chuột qua mã được gạch chân màu đỏ hoặc trên bất kỳ đường ngang màu đỏ nào trên lề bên phải của ngăn trình chỉnh sửa.

2. Thêm các biến vào lớp bên trong WordViewHolder cho TextView và bộ điều hợp:

```
công khai cuối cùng TextView wordItemView;  
cuối cùng WordListAdapter mAdapter;
```

3. Trong lớp bên trong WordViewHolder, thêm một hàm khởi tạo khởi tạo ViewHolder TextView từ tài nguyên XML word và thiết lập bộ điều hợp của nó:

```
public WordViewHolder(View itemView, bộ điều hợp WordListAdapter) { super(itemView);  
    wordItemView = itemView.findViewById(R.id.word);  
    this.mAdapter = bộ chuyển đổi; }
```

4. Chạy ứng dụng của bạn để đảm bảo rằng bạn không có lỗi. Bạn sẽ vẫn chỉ thấy một chế độ xem trống.
5. Nhấp vào tab **Logcat** để xem ngăn **Logcat** và lưu ý E / RecyclerView: Không có bộ chuyển đổi được đính kèm; bỏ qua cảnh báo bối rối. Bạn sẽ gắn bộ chuyển đổi vào RecyclerView trong một bước khác.

2.6 Lưu trữ dữ liệu của bạn trong bộ điều hợp

Bạn cần giữ dữ liệu của mình trong bộ điều hợp và WordListAdapter cần một hàm khởi tạo khởi tạo danh sách từ dữ liệu. Làm theo các bước sau:

- Để giữ dữ liệu của bạn trong bộ điều hợp, hãy tạo một danh sách liên kết riêng tư của các chuỗi trong WordListAdapter và gọi nó là mWordList.

riêng tư cuối cùng 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 hàm khởi tạo để khởi tạo danh sách từ dữ liệu. Để tạo View cho một mục danh sách, WordListAdapter cần thổi phồng XML cho một mục danh sách. Bạn sử dụng máy *bơm hơi bơc* hoặc công việc đó. [LayoutInflator](#) đọc mô tả XML bơc và chuyển đổi nó thành các mục View tương ứng. Bắt đầu bằng cách tạo một biến thành viên cho inflater trong WordListAdapter:

LayoutInflator riêng mInflater;

a

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

Trang 153

4. Triển khai hàm khởi tạo cho WordListAdapter. Hàm khởi tạo cần có tham số ngữ cảnh và danh sách các từ được liên kết với dữ liệu của ứng dụng. Phương thức cần khởi tạo một LayoutInflater cho mInflater và đặt mWordList thành dữ liệu được truyền vào:

```
public WordListAdapter(Ngữ cảnh ngữ cảnh,  
                      LinkedList<String> wordList) {  
    mInflater = LayoutInflater.from(ngữ cảnh);  
    this.mWordList = danh sách từ; }
```

5. Điền vào phương thức onCreateViewHolder() bằng mã sau:

```
@Override  
WordViewHolder công khai trên CreateViewHolder(Cha mẹ ViewGroup,  
                                              int viewType) { Xem mItemView =  
    mInflater.inflate(R.layout.wordlist_item,  
                      cha mẹ, sai);  
    trả về WordViewHolder mới(mItemView, this); }
```

Phương thức onCreateViewHolder() tương tự như phương thức onCreateViewHolder(). Nó thổi phồng bố cục mục và trả về ViewHolder với bố cục và bộ điều hợp.

6. Điền vào phương thức onBindViewHolder() bằng mã bên dưới:

```
@Override  
public void onBindViewHolder(người giữ WordViewHolder, vị trí int) {  
    Chuỗi mCurrent = mWordList.get(position);  
    holder.wordItemView.setText(mCurrent); }
```

Phương thức `onBindViewHolder()` kết nối dữ liệu của bạn với trình giữ chế độ xem.

7. Chạy ứng dụng của bạn để đảm bảo không có lỗi.

2.7. Tạo RecyclerView trong hoạt động

Bây giờ bạn đã có một bộ chuyển đổi với `ViewHolder`, cuối cùng bạn có thể tạo `RecyclerView` và kết nối tất cả các phần để hiển thị dữ liệu của mình.

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

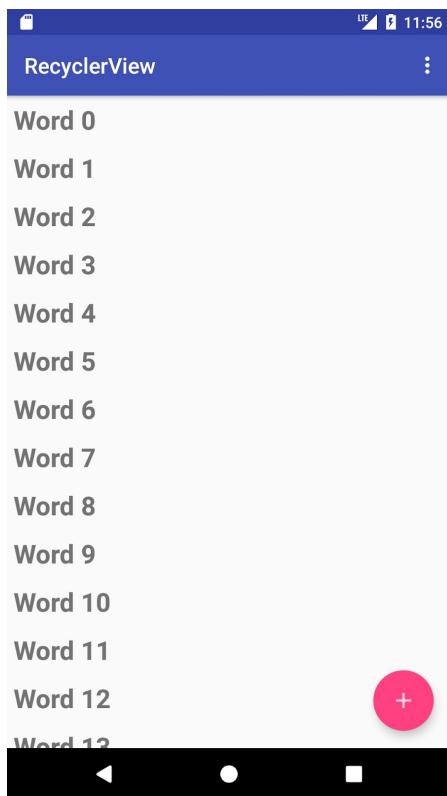
```
RecyclerView riêng mRecyclerView;  
WordListAdapter mAdapter riêng;
```

3. Trong phương thức `onCreate()` của `MainActivity`, hãy 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 `WordList`.

```
Nhận xử lý RecyclerView.  
mRecyclerView = findViewById(R.id.recyclerview);  
Tạo một bộ điều hợp và cung cấp dữ liệu sẽ được hiển thị.  
mAdapter = WordListAdapter mới (này, mWordList);  
Kết nối bộ điều hợp với RecyclerView.  
mRecyclerView.setAdapter (mAdapter);  
Cung cấp cho RecyclerView một trình quản lý bô cục mặc định.  
mRecyclerView.setLayoutManager(LinearLayoutManager mới(này));
```

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 — nó 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 trong `ordListAdapter`.
2. Thay đổi chữ ký lớp `WordViewHolder` để triển khai `View.OnClickListener`:

```
lớp WordViewHolder mở rộng RecyclerView.ViewHolder triển khai  
View.OnClickListener {
```

3. Nhấp vào tiêu đề lớp và trên bóng đèn màu 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 phần thân của phương thức `onClick()`.

```
Nhận vị trí của mục đã được nhấp.  
int mPosition = getLayoutPosition();  
Sử dụng nó để truy cập mục bị ảnh hưởng trong mWordList.  
Phần tử chuỗi = mWordList.get(mPosition);  
Thay đổi từ trong mWordList.  
mWordList.set(mPosition, "Đã bấm! " + phần tử);  
Thông báo cho bộ điều hợp rằng dữ liệu đã thay đổi để có thể  
cập nhật RecyclerView để hiển thị dữ liệu.  
mAdapter.notifyDataSetChanged();
```

5. Kết nối `onClickListener` với View. Thêm mã này vào hàm khởi tạo `WordViewHolder` (bên dưới dòng `this.mAdapter = adapter;`):

```
itemView.setOnClickListener(cái này);
```

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 được chèn.

Làm theo các bước sau:

1. Mở **MainActivity**. Phương thức `onCreate()` đặt một `OnTouchListener()` thành `FloatingActionButton` với phương thức `onClick()` để thực hiện một hành động. Thay đổi phương thức `onClick()` thành như sau:

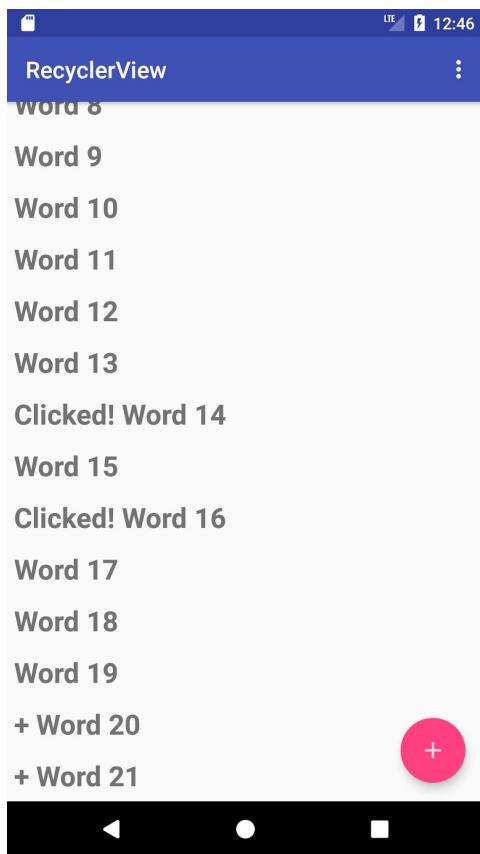
```
@Override  
public void onClick(Xem chế độ xem) {  
    int wordListSize = mWordList.size();  
    Thêm một từ mới vào wordList.  
    mWordList.addLast("+" Word" + wordListSize);  
    Thông báo cho bộ điều hợp rằng dữ liệu đã thay đổi.  
    mRecyclerView.getAdapter().notifyItemInserted(wordListSize);  
    Cuộn xuống dưới cùng.  
    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.

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0.
PDF này là Ánh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2

Trang 158



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: [Recycler View](#)

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: C treo menu tùy chọn để chỉ hiển thị một tùy chọn: **R eset**. Tùy chọn này sẽ trả lại danh sách các từ về trạng thái ban đầu, không có gì được nhấp vào và không có từ bổ sung.

Thử thách 2: C tạo ra một 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.
- Để tạo View cho mỗi mục danh sách, bộ điều hợp tăng tài nguyên bối cảnh XML cho mục danh sách bằng [LayoutInflator](#).

- [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 một 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](#) mà mô tả một mục View và vị trí của nó trong RecyclerView (bằng tiếng Anh).
- 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 bằng Image Asset Studio](#)

Tài liệu dành cho nhà phát triển Android:

- [Người tái chế View](#)
- [Bố cục Máy 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](#)
- [ConstraintLayout](#)
- [RecyclerView.Adapter](#)
- [RecyclerView.ViewHolder](#)
- [View.OnClickListener](#)
- [Tạo danh sách bằng RecyclerView](#)

Video:

- [RecyclerView Hoạt ảnh và hậu trường \(Hội nghị thương định nhà phát triển Android 2015\)](#)

Homework

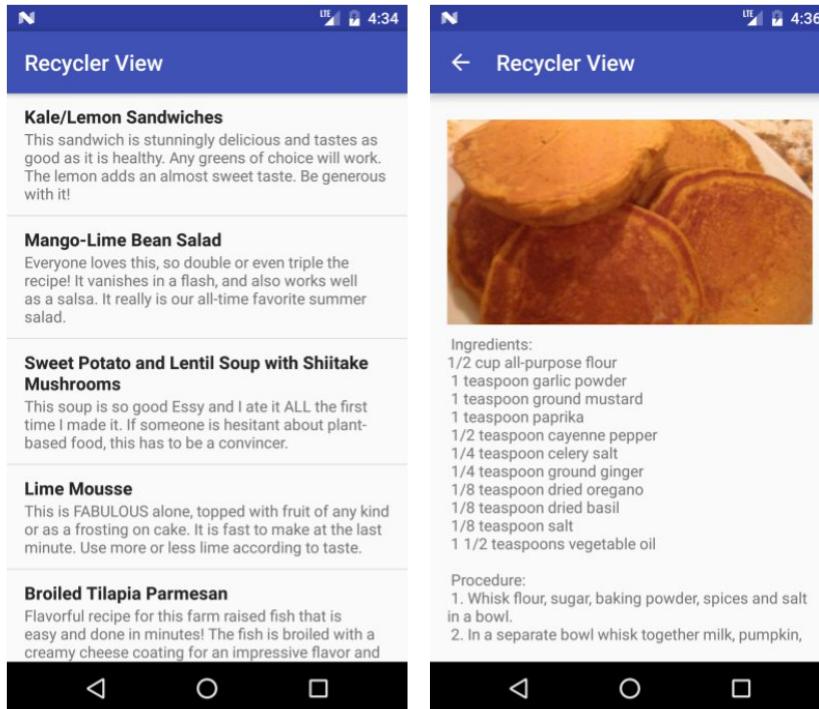
Xây dựng và chạy ứng dụng

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 nhấn 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 và kiểu dáng TextView riêng biệt 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 Up 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

Câu nào sau đây về RecyclerView là `false`? Chọn một.

- `RecyclerView` là một cách tiết kiệm tài nguyên hơn để hiển thị các danh sách có thể cuộn.
- Bạn chỉ cần cung cấp bố cục cho một mục trong danh sách.
- Tất cả các mục trong 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 chuyển đổi một mục `View` và vị trí của nó trong `RecyclerView`? Chọn một.

- Người tái chếView
- RecyclerView.Adapter
- RecyclerView.ViewHolder
- Hoạt động AppCompatActivity

Câu hỏi 3

Bạn cần triển khai giao diện nào để lắng nghe và phản hồi các nhấp chuột của người dùng trong một RecyclerView? Chọn một.

- View.OnClickListener
- RecyclerView.Adapter
- RecyclerView.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 một mục danh sách sẽ bắt đầu một biểu tượng A hiển thị công thức đầy đủ.
- Tệp AndroidManifest.xml xác định mối quan hệ cha để nhấp vào nút **Up** trong chế độ xem công thức sẽ quay lại danh sách các công thức nấu ăn.
- ViewHolder chứa một bố cục với hai phần tử TextView; ví dụ: một 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.

Khóa học cơ bản về nhà phát triển Android (V2) Đơn vị

- Tạo và chỉnh sửa các thành phần 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.
- Trích xuất các chuỗi được mã hóa cứng vào tài nguyên chuỗi.
- Trích xuất các thứ nguyên được mã hóa cứng vào tài nguyên thứ nguyên.
- Thêm các mục menu và biểu tượng vào menu tùy chọn trong thanh ứng dụng.
- Tạo trình xử lý nhấp chuột cho nhấp chuột B.
- Hiển thị thông báo Toast.
- Truyền dữ liệu từ một Activity A sang một Activity khác bằng cách sử dụng Intent.

Những gì bạn sẽ học

Bạn sẽ học cách:

- Xác định tài nguyên style.
- Áp dụng kiểu cho Chế độ xem.
- Áp dụng chủ đề cho Activity hoặc ứng dụng trong XML và theo chương trình.
- Sử dụng D tài nguyên có thể thô.

Bạn sẽ làm gì

- Tạo một ứng dụng mới và thêm các phần tử Button và TextView vào bố cục.
- Tạo các tài nguyên có thể thô D trong XML và sử dụng chúng làm nền cho các phần tử Button của bạn.
- Áp dụng kiểu cho các thành phần 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

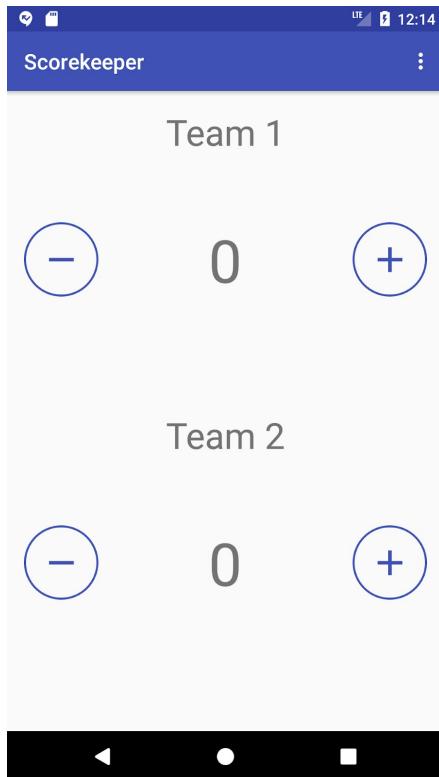
Ứng dụng Scorekeeper bao gồm hai bộ phận tử Button và hai phần tử TextView, được sử dụng để theo dõi điểm số cho bất kỳ trò chơi dựa trên điểm nào với hai người chơi.

a

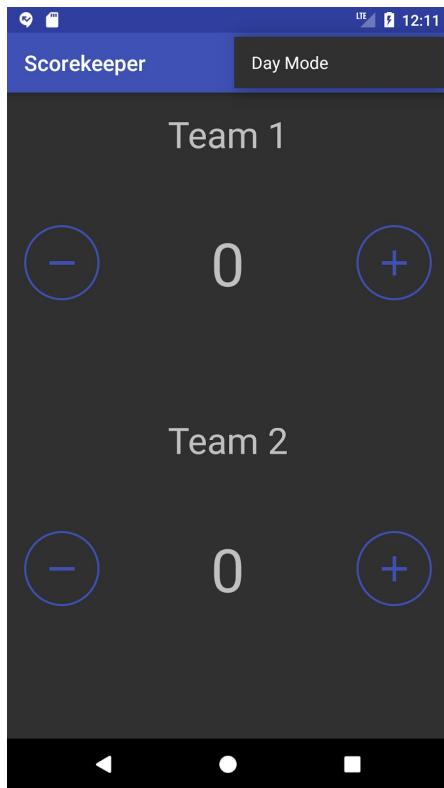
This PDF is a one-time snapshot. See [d developer.android.com/courses/fundamentals-training/toc-v2](https://developer.android.com/courses/fundamentals-training/toc-v2) for the latest updates.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

Trang 165



This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.



Nhiệm vụ 1: Tạo ứng dụng Scorekeeper

Trong phần này, bạn sẽ tạo dự án Android Studio của mình, 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 có tên là **S corekeeper**.
2. Chấp nhận SDK tối thiểu mặc định và chọn **Mẫu Hoạt động Empty**.
3. Chấp nhận tên Activity mặc định (MainActivity) và đảm bảo các tùy chọn **Bố cục Generate file** và **Backwards Compatibility (AppCompat)** được chọn.

4. Nhấp vào **F**inish.

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 nhấp vào tab **Text** để xem mã XML. Ở trên cùng, hoặc root, của hệ thống phân cấp View là [ConstraintLayout](#) ViewGroup:

```
android.support.constraint.ConstraintLayout
```

2. Thay đổi điều này Tập đoàn View đến [L trong taiBố cục](#). Dòng mã thứ hai bây giờ trông có gì đó Thích cái 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:

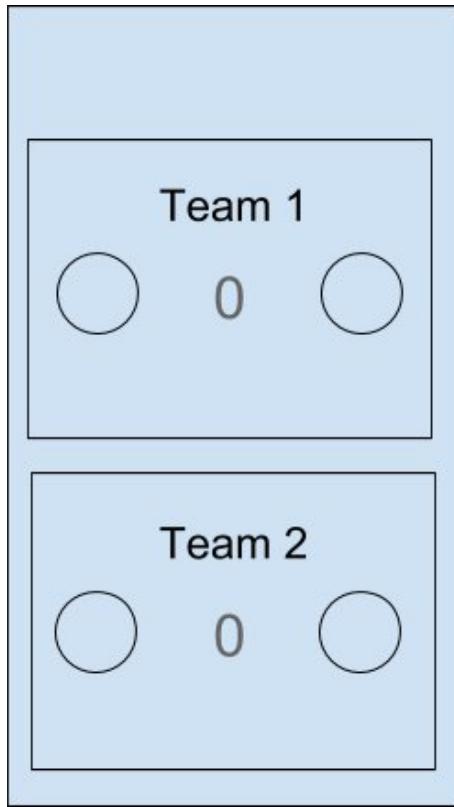
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context="com.example.android.scorekeeper.MainActivity">
```

4. Thêm các thuộc tính sau (mà không xóa các thuộc tính hiện có):

Thuộc tính	Giá trị
Android:Định hướng	"dọc"
android:độ	"16 dtk"

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 chế độ 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:

Thuộc tính RelativeLayout	Giá trị
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 me`.

- Thêm hai `phần tử ImageButton` vào mỗi `RelativeLayout`: một để giảm điểm và một để tăng điểm. Sử dụng các thuộc tính sau:

Thuộc tính ImageButton	Giá trị
<code>android:id</code>	"@+id/giảmĐđiểm1"
<code>Android:layout_width</code>	"wrap_content"
<code>Android:layout_height</code>	"wrap_content"
<code>Android:layout_alignParentLeft</code>	"đúng"
<code>Android:layout_alignParentStart</code>	"đúng"
<code>Android:layout_centerVertical</code>	"đúng"

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.

- Thêm một `TextView` vào mỗi `RelativeLayout` giữa các phần tử `ImageButton` để hiển thị điểm số. Sử dụng các thuộc tính sau:

Thuộc tính TextView	Giá trị
<code>android:id</code>	"@+id/score_1"
<code>Android:layout_width</code>	"wrap_content"
<code>Android:layout_height</code>	"wrap_content"

Android:layout_centerHorizontal	"đúng"
Android:layout_centerVertical	"đúng"
android:văn bản	"0"

Sử dụng `s core_2` làm `ndroid:id` cho `Textview` thứ hai giữa các phần tử `Imagebutton`.

- Thêm một `Textview` khác vào mỗi `RelativeLayout` phía trên điểm số để đại diện cho Tên Đội. Sử dụng các thuộc tính sau:

Thuộc tính <code>TextView</code>	Giá trị
<code>Android:layout_width</code>	"wrap_content"
<code>Android:layout_height</code>	"wrap_content"
<code>Android:layout_alignParentTop</code>	"đúng"
<code>Android:layout_centerHorizontal</code>	"đúng"
android:văn bản	"Đội 1"

Sử dụng "Đội 2" làm `ndroid:text` cho `Textview` thứ hai.

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.

- Chọn **F ile > New > Vector Asset** để mở Vector Asset Studio.
- Nhấp vào biểu tượng để mở danh sách các tệp biểu tượng vật liệu. Chọn **danh mục Content**.
- Chọn biểu tượng **add** và nhấp vào **O K**.

4. Đổi tên tài nguyên file **ic_plus** và chọn hộp **kiểm O verride** bên cạnh tùy chọn kích thước.
5. Thay đổi kích thước của biểu tượng thành **40dp x 40dp**.
6. Nhấp vào **N ext** và sau đó nhấp **vào F inish**.
7. Lặp lại quy trình này để thêm biểu tượng **r emove** và đặt tên cho 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 left của bố cục:

```
    android:src="@drawable/ic_minus"
    android:contentDescription="Trừ
                                Nút"
```

9. Thêm các thuộc tính sau vào ImageButton ở phía right của bố cục:

```
    android:src="@drawable/ic_plus"
    android:contentDescription="Thêm
                                Nút"
```

10. Trích xuất tất cả các tài nguyên chuỗi của bạn. Quá trình này xóa tất cả các chuỗi của bạn khỏi mã Java và đưa chúng vào file 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.

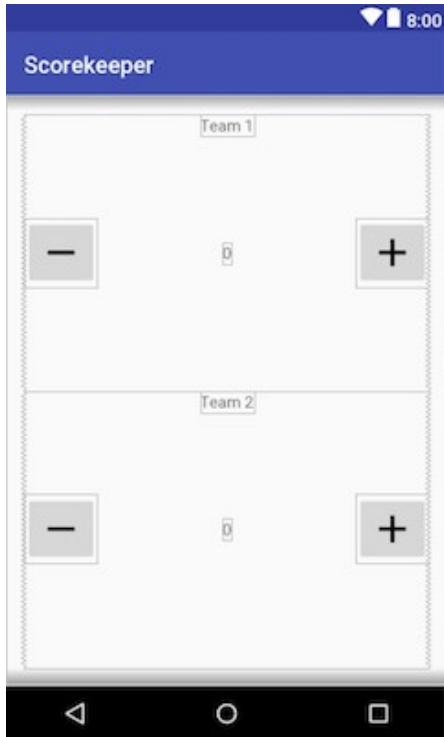
Lưu ý: Mã của bạn có thể hơi khác một chút, vì có nhiều cách để đạt được cùng một bố cục.

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

Trang 173



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="dọc"
    android:padding="16dp"
    tools:context="com.example.android.scorekeeper.MainActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1">
        <Ché độ xem văn bản
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="đúng"
        android:layout_centerHorizontal="đúng"
        android:text="@string/team_1" />
<Nút hình ảnh
    android:id="@+id/giảmTeam1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="đúng"
    android:layout_alignParentStart="đúng"
    android:layout_centerVertical="đúng"
    android:src="@drawable/ic_minus"
    android:contentDescription=
        "@string/minus_button_description" />
<Chế độ xem văn bản
    android:id="@+id/score_1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="đúng"
    android:layout_centerVertical="đúng"
    android:text="@string/initial_count" />
<Nút hình ảnh
    android:id="@+id/increaseTeam1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="đúng"
    android:layout_alignParentRight="đúng"
    android:layout_centerVertical="đúng"
    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">
    <Chế độ xem văn bản
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="đúng"
        android:layout_centerHorizontal="đúng"
        android:text="@string/team_2" />

    <Nút hình ảnh
        android:id="@+id/giảmTeam2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="đúng"
        android:layout_alignParentStart="đúng"
        android:layout_centerVertical="đúng"
        android:src="@drawable/ic_minus"
        android:contentDescription=
            "@string/minus_button_description" />
    <Chè độ xem văn bản
        android:id="@+id/score_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="đúng"
        android:layout_centerVertical="đúng"
        android:text="@string/initial_count" />

    <Nút hình ảnh
        android:id="@+id/increaseTeam2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="đúng"
        android:layout_alignParentRight="đúng"
        android:layout_centerVertical="đúng"
        android:src="@drawable/ic_plus"
        android:contentDescription=
            "@string/plus_button_description" />
    </RelativeLayout>
</LinearLayout>
```

1.5 Khởi tạo các phần tử TextView và các biến điểm đếm

Để theo dõi điểm số, bạn sẽ cần hai điều:

- Biến số nguyên để theo dõi điểm số.
- Tham chiếu đến từng phần tử TextView điểm số trong MainActivity để bạn có thể cập nhật điểm.

Làm theo các bước sau:

1. Tạo hai biến thành viên số nguyên đại diện cho điểm số của mỗi đội.

Các biến thành viên để giữ điểm riêng tư int
mScore1; private int mScore2;

2. Tạo hai biến thành viên TextView để giữ tham chiếu đến các phần tử TextView.

Các biến thành viên để giữ điểm riêng tư 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);  
  
    Tìm TextView theo ID  
    mScoreText1 = (TextView)findViewById(R.id.score_1);
```

```
mScoreText2 = (TextView)findViewById(R.id.score_2); }
```

1.6 Triển khai các trình xử lý nhấp chuột cho các phần tử ImageButton

Bạn sẽ thêm thuộc tính android:onClick vào mỗi ImageButton và tạo hai phương thức xử lý nhấp chuột trong MainActivity. Các phần tử ImageButton bên trái sẽ giảm điểm TextView, trong khi các phần tử bên phải nên tăng nó.

1. Mở một ctivity_main.xml nếu nó chưa được mở và thêm thuộc tính android:onClick sau vào ImageButton đầu tiên ở phía left của bố cục:

```
android:onClick="giảm điểm"
```

2. Tên phương thức decreaseScore được gạch chân bằng 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 **Create '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 đây vào mỗi ImageButton ở phía right của bố cục:

```
android:onClick="increaseScore"
```

5. Tên phương thức increaseScore được gạch chân bằng 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 **Create**

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0. PDF này làẢnh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2 'increaseScore(view)' trong 'MainActivity'. Android Studio tạo i ncreaseScore() phương thức sơ khai trong M ainHoạt động.

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 vào, để mã của bạn có thể cập nhật nhóm thích hợp.

Mã giải pháp cho increaseScore() và decreaseScore()

```
/**  
 * Phương pháp xử lý onClick của cả hai nút giảm  
 * Chế độ xem @param Chế độ xem nút đã được nhấp vào  
 */  
public void decreaseScore(View view) { // Lấy ID của nút đã được nhấp vào  
    int viewID = view.getId();  
    công tắc (viewID){  
        Nếu nó ở Đội 1  
        trường hợp R.id.decreaseTeam1:  
            Giảm điểm số và cập nhật TextView mScore1;  
            mScoreText1.setText(String.valueOf(mScore1));  
            phá vỡ;  
        Nếu đó là Đội 2  
        trường hợp R.id.decreaseTeam2:  
            Giảm điểm số và cập nhật TextView mScore2;  
            mScoreText2.setText(String.valueOf(mScore2));  
    }  
  
/**  
 * Phương thức xử lý onClick của cả hai nút tăng  
 * Chế độ xem @param Chế độ xem nút đã được nhấp vào  
 */  
public void increaseScore(View view) { // Nhận ID của nút đã được nhấp vào  
    int viewID = view.getId();  
    công tắc (viewID){  
        Nếu nó ở Đội 1
```

```
trường hợp R.id.increaseTeam1:  
    Tăng điểm và cập nhật TextView mScore1++;  
    mScoreText1.setText(String.valueOf(mScore1));  
    phá vỡ;  
Nếu đó là Đội 2  
trường hợp R.id.increaseTeam2:  
    Tăng điểm và cập nhật 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 được gọi là [D rawable](#). Trong bài tập sau, bạn sẽ học cách tạo một loại D rawable nhất định được gọi là [S hapeDrawable](#) và áp dụng nó cho các phần tử ImageButton của bạn làm nền.

Để biết thêm thông tin về D rawables, hãy xem [D rawable Resources](#).

2.1 Tạo một ShapeDrawable

S [hapeDrawable](#) 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, **r ight-click** (hoặc **C ontrol-click**) vào **thư mục d rawable** trong **thư mục res**.
2. Chọn **N ew > tệp tài nguyên Drawable**.
3. Đặt tên tệp **b utton_background** và nhấp vào **O K**. (Đừng thay đổi **S ource set** hoặc **D irectory name**, và không thêm các từ hạn định). Android Studio tạo tệp **b utton_background.xml** trong thư mục **d rawable**.

4. Mở **button_background.xml**, nhấp vào tab **Text** để chỉnh sửa mã XML và xóa tất cả mã ngoại trừ:

```
<?xml version="1.0" encoding="utf8"?>
```

5. Thêm mã sau tạo ra một hình bầu dục với một đường viền:

```
Hình dạng <  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:shape="oval">  
    <đột quy>  
        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 cá bốn phần tử **ImageButton**:

```
    android:background="@drawable/button_background"
```

3. Nhấp vào tab **Preview** trong trình chỉnh sửa bố cục để xem nền tự động chia tỷ lệ để phù hợp với kích thước của **ImageButton**.
4. Để hiển thị đúng từng **ImageButton** trên tất cả các thiết bị, bạn sẽ thay đổi các thuộc tính **android:layout_height** và **ndroid:layout_width** cho mỗi **ImageButton** thành

70dp, đây là một 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"
```

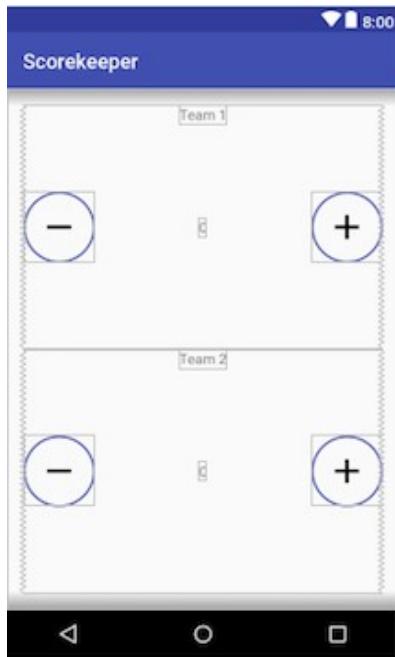
5. Nhập một lần vào "70dp", nhấn **A lt-Enter** trong Windows hoặc **O ption-Enter** trong macOS và chọn **Trích xuất tài nguyên thứ nguyên từ menu bật lên**.
6. Nhập **b utton_size** cho **tên nguồn điện tử R**.
7. Nhấp vào **OK** Thao tác này tạo ra một tài nguyên thứ nguyên trong tệp `dimens.xml` (trong thư mục `values`) và thứ nguyên trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên:

```
@dimen / button_size
```

8. Thay đổi các thuộc tính `ndroid:layout_height` và `ndroid:layout_width` cho mỗi 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 **Đánh giá P** 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 của mình, 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 học 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: Các kiểu dành cho các thuộc tính sửa đổi giao diện của View. Các thông số bố cục như chiều cao, trọng lượng và vị trí tương đối nên 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 `parent` và có các thuộc tính sau:

- Bất kỳ thuộc tính 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 (kiểu con ghi đè cha mẹ).
- Kiểu con có thể bao gồm các thuộc tính bổ sung mà cha mẹ không xác định.

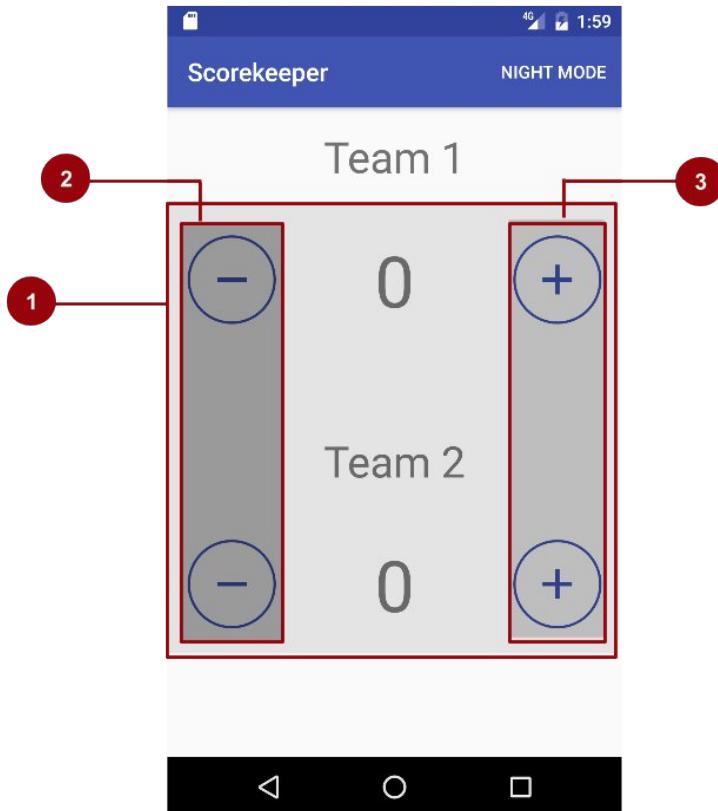
Ví dụ: cả bốn phần tử `ImageButton` trong ứng dụng Scorekeeper đều có chung một nền chung

Có thể vẽ được nhưng với các biểu tượng khác nhau cho cộng (tăng điểm) và trừ (giảm điểm).

Hơn nữa, hai phần tử cộng `ImageButton` chia sẻ cùng một biểu tượng, cũng như hai phần tử trừ `ImageButton`. 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 có thể thay đổi.
2. Một kiểu cho các phần tử trừ `ImageButton`. Phong cách này kế thừa các thuộc tính của `ImageButton` và bao gồm biểu tượng trừ.
3. Một phong cách cho các yếu tố cộng `ImageButton`. Phong cách 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 > giá trị` trong ngăn Project > Android và mở tệp `styles.xml`.

Đây là nơi tất cả các mã kiểu của bạn sẽ được đặt. Kiểu "AppTheme" luôn được thêm tự động và bạn có thể thấy rằng nó mở rộng từ `Theme.AppCompat.Light.DarkActionBar` (bằng tiếng Anh).

```
<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 ppTheme, 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 phong cách được áp dụng cho toàn bộ Activity hoặc ứng dụng thay vì một View duy nhất. Sử dụng chủ đề sẽ tạo ra một kiểu nhất quán trong toàn bộ ứng dụng hoặc ứng dụng—ví dụ: giao diện nhất quán cho thanh ứng dụng trong mọi phần của ứng dụng của bạn.

2. Giữa các thẻ tài nguyên <, hãy thêm 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 lại các thuộc tính mặc định của Button. Nó cũng thêm một thuộc tính thay đổi nền của Drawer thành nền 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>
<tên mục =
"android:contentDescription">@string/minus_button_description</item>
</style>
```

- Bây giờ bạn có thể sử dụng các kiểu này để thay thế các thuộc tính kiểu cụ thể của các phần tử ImageButton. Mở **tệp bố cục** activity_main.xml cho Main Activity và thay thế các thuộc tính sau cho cả hai phần tử ImageButton trừ:

Xóa các thuộc tính này	Thêm thuộc tính này
android:src	style="@style/Nút trừ"
android:nội dungMô tả	
Android:Nền	

Lưu ý: Thuộc tính style không sử dụng không gian tên android: vì style là một phần của các thuộc tính XML mặc định.

- Thay thế các thuộc tính sau cho cả hai phần tử cộng ImageButton:

Xóa các thuộc tính này	Thêm thuộc tính này
android:src	style="@style/PlusButtons"
android:nội dungMô tả	

Android:Nền	
-------------	--

3.2 Tạo kiểu TextView

Tên đội và điểm số hiển thị các phần tử TextView cũng có thể được tạo kiểu vì chúng có màu sắc và phông chữ chung. Thực hiện như sau:

1. Thêm thuộc tính sau vào tất cả các phần tử TextView:

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

2. Nhấp chuột phải (hoặc nhấp chuột C) ở bất kỳ đâu trong điểm số đầu tiên TextView thuộc tính và chọn Refactor > Extract > Style...
3. Đặt tên cho kiểu là **S coreText** và chọn hộp **textAppearance** (thuộc tính bạn vừa thêm) cũng như hộp kiểm **Launch 'Use Styles If Possible' refactoring after the style is extracted.** Thao tác này sẽ quét tệp bố cục để tìm các chế độ xem có cùng thuộc tính và áp dụng kiểu cho bạn. Không trích xuất các thuộc tính có liên quan đến bố cục — nhấp vào các hộp kiểm khác để tắt chúng.
4. Chọn **OK**.
5. Đảm bảo phạm vi được đặt thành **tệp** bố cục **activity_main.xml** và nhấp vào **OK**.
6. Một ngăn ở cuối Android Studio sẽ mở ra nếu thấy cùng một kiểu trong các chế độ xem khác. Chọn **Do Refactor** để áp dụng kiểu mới cho các chế độ 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:

```
<tài nguyên>
    <! Chủ đề ứng dụng cơ sở. >
    <style name="AppTheme"
        parent="Theme.AppCompat.Light.DarkActionBar">
        <! Tùy chỉnh chủ đề của bạn tại đây. >
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </ phong cách>

    <style name="ScoreButtons" parent="AppTheme"> <item
        name="android:background">@drawable/button_background</item>

    </ phong cách>

    <style name="PlusButtons" parent="ScoreButtons">
        <item name="android:src">@drawable/ic_plus</item> <item
        name=
            "android:contentDescription">@string/plus_button_description</item>
    </ phong cách>

    <style name="MinusButtons" parent="ScoreButtons">
        <item name="android:src">@drawable/ic_minus</item>
        <tên mục =
            "android:contentDescription">@string/minus_button_description</item>
    </ phong cách>

    <style name="ScoreText">
        <tên mục =
            "android:textAppearance">@style/TextAppearance.AppCompat.Headline</item>
    </ phong cách>
</tài nguyên>
```

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="dọc"
    android:padding="16dp"
    tools:context="com.example.android.scorekeeper.MainActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1">
        <Ché độ xem văn bản
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="đúng"
            android:layout_centerHorizontal="đúng"
            android:text="@string/team_1"
```

```
    style="@style/ScoreText" />

    <Nút hình ảnh
        android:id="@+id/giảmTeam1"
        android:layout_width="@dimen/button_size"
        android:layout_height="@dimen/button_size"
        android:layout_alignParentLeft="đúng"
        android:layout_alignParentStart="đúng"
        android:layout_centerVertical="đúng"
        style="@style/MinusButtons"
        android:onClick="decreaseScore"/>

    <Chế độ xem văn bản
        android:id="@+id/score_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="đúng"
        android:layout_centerVertical="đúng"
        android:text="@string/initial_count"
        style="@style/ScoreText" />

    <Nút hình ảnh
        android:id="@+id/increaseTeam1"
        android:layout_width="@dimen/button_size"
        android:layout_height="@dimen/button_size"
        android:layout_alignParentEnd="đúng"
        android:layout_alignParentRight="đúng"
        android:layout_centerVertical="đúng"
        style="@style/PlusButtons"
        android:onClick="increaseScore"/>

</RelativeLayout>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="0dp" android:layout_weight="1">

    <Chế độ xem văn bản
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="đúng"
        android:layout_centerHorizontal="đúng"
```

```
        android:text="@string/team_2"
        style="@style/ScoreText" />

    <Nút hình ảnh
        android:id="@+id/giảmTeam2"
        android:layout_width="@dimen/button_size"
        android:layout_height="@dimen/button_size"
        android:layout_alignParentLeft="đúng"
        android:layout_alignParentStart="đúng"
        android:layout_centerVertical="đúng"
        style="@style/MinusButtons"
        android:onClick="decreaseScore"/>

    <Chỗ để xem văn bản
        android:id="@+id/score_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="đúng"
        android:layout_centerVertical="đúng"
        android:text="@string/initial_count"
        style="@style/ScoreText" />

    <Nút hình ảnh
        android:id="@+id/increaseTeam2"
        android:layout_width="@dimen/button_size"
        android:layout_height="@dimen/button_size"
        android:layout_alignParentEnd="đúng"
        android:layout_alignParentRight="đúng"
        android:layout_centerVertical="đúng"
        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:

Phong cách	Khoản
Nút ghi điểm	<code><item name="android:tint">@color/colorPrimary</item></code>
Văn bản điểm	<code>< mục name="android:textAppearance">@style/TextAppearance.AppCompat.Display3 </item></code>

Giá trị `colorPrimary` được Android Studio tự động tạo khi bạn tạo dự án. Bạn có thể tìm thấy nó trong **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.

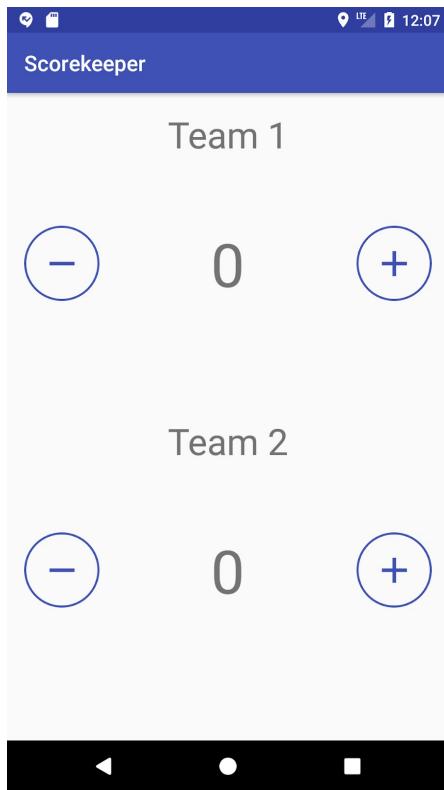
1. Tạo một kiểu mới có tên là **TeamText** với thuộc tính `textAppearance` được đặt như sau:

```
<style name="TeamText">  
    <tên mục =  
        "android:textAppearance">@style/TextAppearance.AppCompat.Display1  
    < / mục>  
< / phong cách>
```

2. Trong `activity_main.xml`, thay đổi thuộc tính `style` của các phần tử tên nhóm `TextView` thành kiểu `TeamText` mới tạo.

```
style="@style/Văn bản nhóm"
```

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 tệp styles.xml. Nhưng điều gì sẽ xảy ra nếu bạn muốn xác định các kiểu cho toàn bộ Activity hoặc toàn bộ ứng dụng? Có thể thực hiện điều này bằng cách sử dụng themes. Để đặt chủ đề cho mỗi Activity, bạn cần sửa đổi AndroidManifest.xml tệp.

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 có độ 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 **A ndroidManifest.xml**, tìm thẻ > ứng dụng < 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 phần con của Chủ đề Theme.Appcompat.Light.DarkActionBar như có thể thấy trong tyles.xml.

Để áp dụng chủ đề cho một hoạt động thay vì toàn bộ ứng dụng, hãy đặt thuộc tính theme vào thẻ <Hoạt động> thay vì thẻ > ứng dụng <. Để biết thêm thông tin về chủ đề và kiểu, hãy xem Hướng [dẫn chủ đề và S tyle](#).

4.2 Thêm nút chủ đề vào menu

Một cách sử dụng để thiết lập chủ đề cho ứng dụng của bạn là cung cấp trải nghiệm trực quan thay thế cho việc 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 **s trings.xml** và tạo hai tài nguyên chuỗi cho mục menu tùy chọn này:

```
<string name="night_mode">Chế độ ban đêm</chuỗi>
<string name="day_mode">Chế độ ngày</chuỗi>
```

2. Nhấp chuột phải (hoặc nhấp chuột C) vào thư mục res trong ngăn Project > Android và chọn New > tệp tài nguyên Android.
3. Đặt tên tệp main_menu, thay đổi Loại tài nguyên thành Menu và nhấp vào OK. Trình chỉnh sửa bối cảnh xuất hiện cho tệp main_menu.xml.
4. Nhấp vào tab Text của trình soạn thảo bối cảnh để hiển thị mã XML.
5. Thêm một mục menu với các thuộc tính sau:

```
< mục
    android:id="@+id/night_mode"
    android:title="@string/night_mode"/>
```

- Mở MainActivity, nhấn Ctrl+O để mở menu Phương pháp Override và chọn onCreateOptionsMenu (menu: Menu): boolean method nằm trong danh mục android.app.Activity.
6. Nhấp vào OK Một ndroid Studio tạo ra sẵn khai báo onCreateOptionsMenu() method với return super.onCreateOptionsMenu(menu) làm câu lệnh duy nhất của nó.
 1. Trong onCreateOptionsMenu() ngay trước câu lệnh return.super, thêm mã để khởi tạo menu:

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

4.3 Thay đổi chủ đề từ menu

Chủ đề DayNight sử dụng lớp AppCompatActivityDelegate để đặt các tùy chọn chế độ ban đêm trong Hoạt động của bạn. Để tìm hiểu thêm về chủ đề này, hãy truy cập bài [đang blog này](#).

1. Trong tệp styles.xml **của bạn**, sửa đổi cha của AppTheme thành "Theme.AppCompat.DayNight.DarkActionBar" (bằng tiếng Anh).
2. Ghi đè phương thức onOptionsItemSelected() trong MainActivity và thêm mã để kiểm tra mục menu nào đã được nhấp:

```
@Override boolean công khai onOptionsItemSelected(MenuItem item) {  
    Kiểm tra xem đã nhấp vào đúng mục chưa  
    if(item.getItemId()==R.id.night_mode) {}  
    VIỆC CẦN LÀM: Nhận trạng thái chế độ ban đêm của ứng dụng.  
    trả về 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) { //  
Nhận trạng thái chế độ ban đêm của ứng  
dụng.  
    int nightMode = AppCompatDelegate.getDefaultNightMode();  
    Đặt chế độ chủ đề cho hoạt động đã khởi động lại  
    nếu (chế độ đêm == AppCompatDelegate.MODE_NIGHT_YES) {  
        AppCompatDelegate.setDefaultNightMode  
            (AppCompatDelegate.MODE_NIGHT_NO); }  
    khác {  
        AppCompatDelegate.setDefaultNightMode  
            (AppCompatDelegate.MODE_NIGHT_YES); }  
    Tạo lại hoạt động để thay đổi chủ đề có hiệu lực. recreate();
```

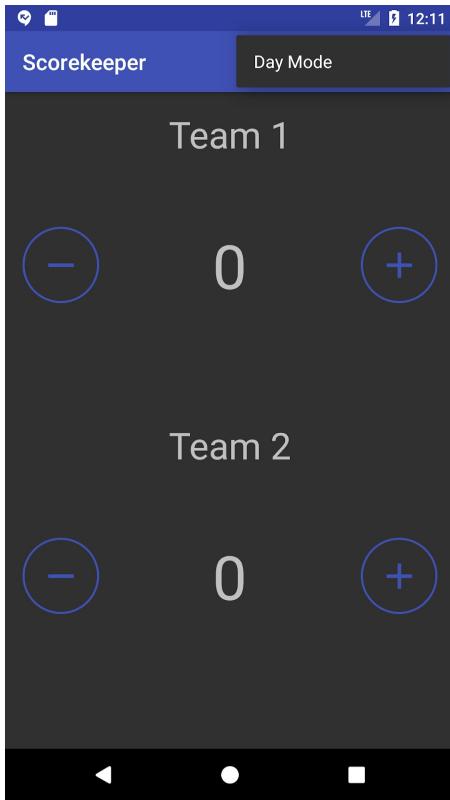
Để phản hồi khi nhấp vào mục menu, mã sẽ xác minh cài đặt chế độ ban đêm hiện tại bằng cách gọi AppCompatDelegate.getDefaultNightMode().

Chủ đề chỉ có thể thay đổi trong khi hoạt động đang được tạo, vì vậy mã sẽ gọi `onCreate()` để thay đổi chủ đề có hiệu lực.

4. Chạy ứng dụng của bạn. **Mục menu Chế độ Night** bây giờ sẽ chuyển đổi chủ đề của Activity của bạn.
5. Bạn có thể nhận thấy rằng nhãn cho mục menu của bạn luôn hiển thị **Chế độ Night**, 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:

```
Thay đổi nhãn của menu dựa trên trạng thái của ứng dụng. int
nightMode = AppCompatDelegate.getDefaultNightMode(); if(chế độ đêm
== AppCompatDelegate.MODE_NIGHT_YES) {
    menu.findItem(R.id.night_mode).setTitle(R.string.day_mode); }
khác{
    menu.findItem(R.id.night_mode).setTitle(R.string.night_mode);
} trả về true;
```

1. Chạy ứng dụng của bạn. Mục menu nhãn **Night Mode** , sau khi người dùng nhấn vào nó, bây giờ sẽ thay đổi thành **Day Mode** (cùng với chủ đề).



4.4 LưuInstanceState

Bạn đã học được trong các bài học trước rằng bạn phải chuẩn bị cho khả năng Activity của mình bị phá hủy và tái tạo 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 cuối cùng String STATE_SCORE_1 = "Điểm đội 1"; static cuối  
cùng String STATE_SCORE_2 = "Điểm đội 2";
```

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

```
@Override  
protected void onSaveInstanceState(Bundle outState) {  
    Lưu điểm số.  
    outState.putInt(STATE_SCORE_1, mScore1);  
    outState.putInt(STATE_SCORE_2, mScore2);  
    super.onSaveInstanceState(outState); }
```

- 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);  
    Đặt chế độ xem văn bản điểm  
    mScoreText1.setText(String.valueOf(mScore1));  
    mScoreText2.setText(String.valueOf(mScore2)); }
```

- Chạy ứng dụng và nhấn vào nút cộng `ImageButton` 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: [Scorekeeper](#)

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 này.

Thử thách: 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 có thể thô D để thay đổi nền của ImageButton 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 giúp nâng cao giao diện 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, 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.
- Style có thể chỉ định các thuộc tính phổ biến như chiều cao, đệm, màu phông chữ, kích thước phông chữ và màu nền.
- Style không nên bao gồm thông tin liên quan đến bố cục.
- Style có thể được áp dụng cho View, Activity hoặc toàn bộ ứng dụng. Tệp 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 `parent` trong XML.
- Khi bạn áp dụng style cho một tập hợp các yếu tố View trong một hoạt động hoặc trong toàn bộ ứng dụng của bạn, điều đó được gọi là `theme`.
- 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, kiểu và chủ đề](#).

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 bằng Image Asset Studio](#)

Tài liệu dành cho nhà phát triển 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](#)
- [Nút](#) • [Bố trí](#)
- [Thư viện hỗ trợ](#)
- [Tổng quan về khả năng tương thích màn hình](#) • [Hỗ trợ các kích thước màn hình khác nhau](#)
- [Đồ họa có thể vẽ hoạt hình](#)
- [Tải bitmap lớn một cách hiệu quả](#) • [R. phong cách](#) Lớp phong cách và chủ đề
- [support.v7.appcompat.R.style](#) của các kiểu và chủ đề

Thiết kế vật liệu:

- [Tìm hiểu về điều hướng](#)
- [Thanh ứng dụng](#)

Blog dành cho nhà phát triển Android: [Thư viện hỗ trợ thiết kế ndroid](#)

Khác:

- Trung bình: [Hướng dẫn chủ đề DayNight](#)
- Video: [Udacity - Chủ đề và phong cách](#)
- [Android Asset Studio](#) của Roman Nurik
- [Tài liệu lướt](#)

Homework

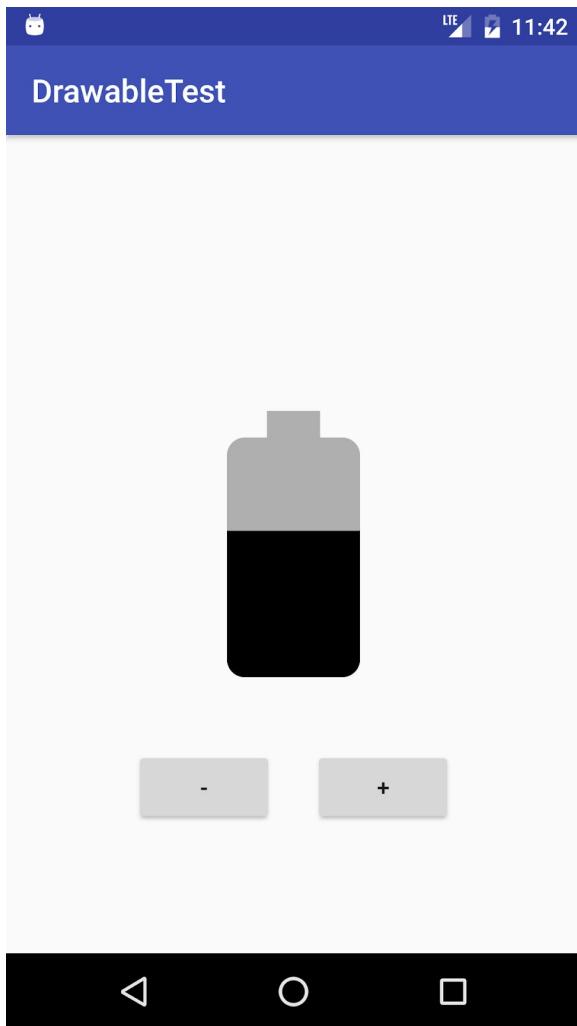
Xây dựng và chạy ứng dụng

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 hiển 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 xuất hiện đầy hơn.
- Nút trừ làm giảm mức, làm cho 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:

- LevelList Có thể vẽ được
- Chuyển tiếp Có thể vẽ được

- StateListCó thể vẽ được
- NinePatchCó thể vẽ được

Câu hỏi 2

Bạn sử dụng loại D rawable nào để tạo ra một chữ B hiển thị một nền khi nó được nhấn và một nền khác khi nó được di chuột qua? Chọn một:

- LevelListCó thể vẽ được
- Chuyển tiếpCó thể vẽ được
- StateListCó thể vẽ được
- 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 cấp độ 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 thiết kế Material [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 học cách thêm các widget CardView và FloatingActionButton vào ứng dụng của bạn, cách sử dụng hình ảnh 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 thành phần 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 nhấp chuột B.
- Trích xuất văn bản vào tài nguyên chuỗi và thứ nguyên vào tài nguyên thứ nguyên.
- Sử dụng các tùy chọn có thể vẽ, kiểu và chủ đề.
- Sử dụng [RecyclerView](#) để hiển thị danh sách.

Những gì bạn sẽ học

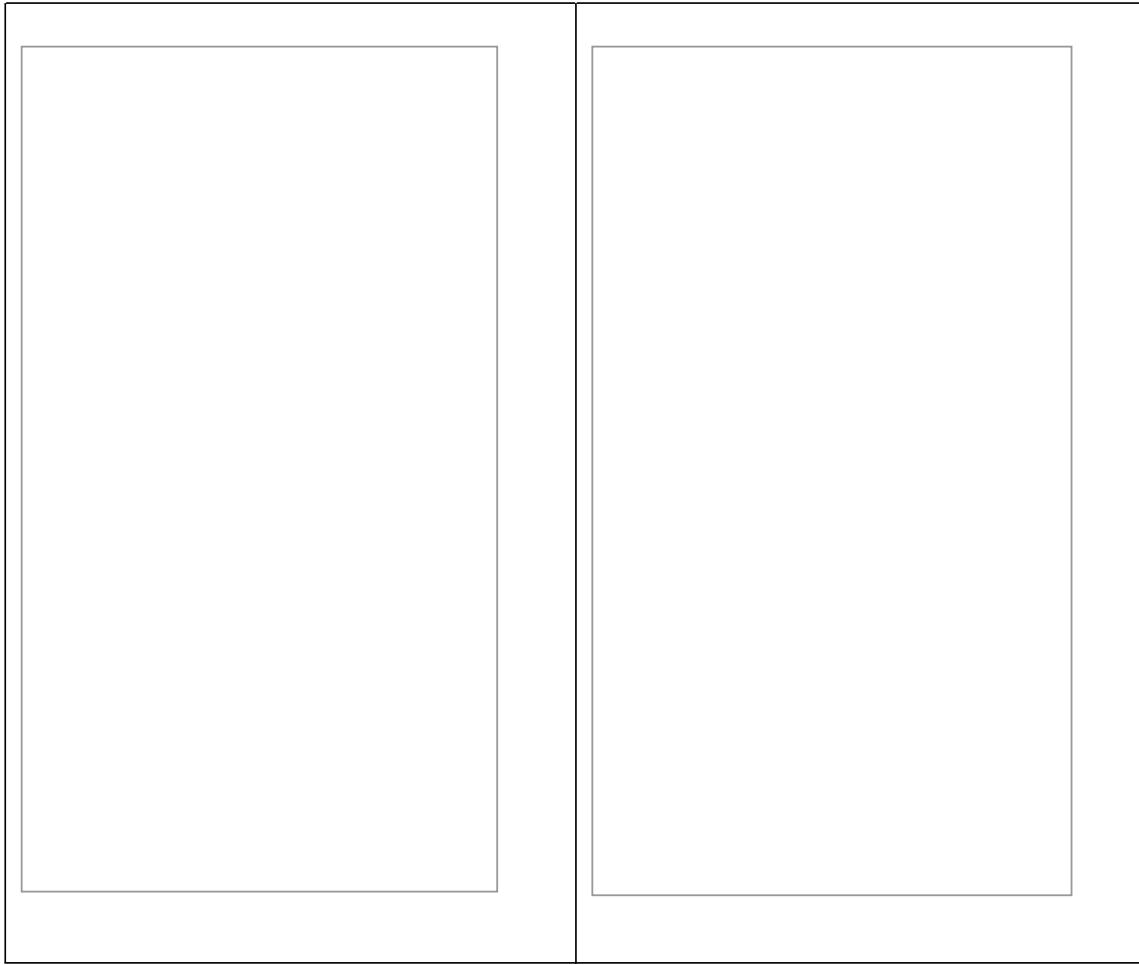
- Đề xuất sử dụng các tiện ích Material Design như [FloatingActionButton](#) và [CardView](#).
- Cách sử dụng hình ảnh hiệu quả trong ứng dụng của bạn.
- Các phương pháp hay nhất được đề xuất để thiết kế bố cục trực quan bằng cách sử dụng màu đậm.

Bạn sẽ làm gì

- Sửa đổi ứng dụng để tuân theo [nguyên tắc Thiết kế Aterial M](#).
- 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 cài 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 ArrayList của các đối tượng Sport làm dữ liệu và điền dữ liệu này vào mỗi hàng.

MainActivity.java

MainActivity khởi tạo RecyclerView và bộ điều hợp, đồng thời tạo dữ liệu từ tài nguyên Tập tin.

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 các 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 `Image` để làm cho ứng dụng tuân thủ các nguyên tắc 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 `dependencies`:

```
implementation 'com.android.support:cardviewv7:26.1.0'
```

Phiên bản của thư viện hỗ trợ có thể đã thay đổi kể từ khi viết thực tế này. Cập nhật thông tin trên lên phiên bản do Android Studio đề xuất và nhấp vào `Sync` để đồng bộ hóa các tệp `build.gradle` của bạn.

2. Mở biểu tượng `List_item.xml` và bao quanh gốc `List` trong `list` với `android.support.v7.widget.CardView`. Di chuyển khai báo lược đồ

(x xmlns:android="http://schemas.android.com/apk/res/android) vào CardView và thêm các thuộc tính sau:

Thuộc tính	Giá trị
Android:layout_width	"match_parent"
Android:layout_height	"wrap_content"
Android:layout_margin	"8 dp"

Khai báo lược đồ cần chuyển sang CardView vì CardView hiện là chế độ xem cấp cao nhất trong tệp bố cục của bạn.

- Chọn **Code > Reformat Code** 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">
    <Bô cục tuyên tính
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <! Phần còn lại của LinearLayout >
        <! Các phần tử TextView > </LinearLayout>
    </android.support.v7.widget.CardView>
```

- Chạy ứng dụng. Bây giờ mỗi mục hàng được chứa bên trong CardView, được nâng lên trên layer dưới cùng và tạo bóng.

Trang 212

2.2 Tải xuống hình ảnh

CardsView không nhắm 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 trở nên thú vị hơn bằng cách thêm hình ảnh biểu ngữ vào mỗi hàng!

Việc sử dụng hình ảnh tốn nhiều tài nguyên đối với ứng dụng của bạn: khung Android phải tải toàn bộ hình ảnh vào bộ nhớ ở độ phân giải đầy đủ, ngay cả khi ứng dụng chỉ hiển thị một hình thu nhỏ của hình ảnh.

Trong phần này, bạn sẽ tìm hiểu cách sử dụng thư viện [Glide](#) để tải hình ảnh lớn một cách hiệu quả mà không làm cạn kiệt tài nguyên hoặc thậm chí làm hỏng ứng dụng của bạn do các trường hợp ngoại lệ 'Hết bộ nhớ'.

1. Tải xuống tệp [zip hình ảnh banner](#).
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 thô 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 bạn có thể đưa nó vào đối tượng `Sports`. Để thực hiện việc này, hãy xác định một mảng chứa tất cả các đường dẫn đến các tệp có thể vẽ dưới dạng các mục trong tệp `tring.xml` của bạn. Đảm bảo rằng chúng theo cùng thứ tự với mảng `sports_titles` cũng được xác định trong cùng một tệp:

```
<tên mảng ="sports_images">
<mục>@drawable / img_baseball< / mục>
<mục>@drawable / img_badminton< / mục>
<mục>@drawable / img_basketball< / mục>
<mục>@drawable/img_bowling</mục>
<mặt hàng>@drawable / img_cycling< / mặt hàng> <mặt hàng>@drawable /
img_golf< / mục>
<mục>@drawable/img_running</mục>
<mục>@drawable/img_soccer</mục>
<mục>@drawable / img_swimming< / mục>
<mục>@drawable/img_tabletennis</mục>
```

```
<mục>@drawable / img_tennis< / mục>
</mảng>
```

Trang 213

2.3 Sửa đổi đối tượng Thể thao

Đối tượng cổng S sẽ cần bao gồm tài nguyên có thể thô D tương ứng với môn thể thao. Để đạt được điều đó:

1. Thêm một biến thành viên số nguyên vào đối tượng cổng S sẽ chứa tài nguyên có thể thô D:

```
private final int imageResource;
```

2. Sửa đổi hàm 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 = thông tin;
this.imageResource = imageResource; }
```

3. Tạo một getter cho số nguyên tài nguyên:

```
public int getImageResource() {  
    trả về 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 các tài nguyên có thể thô D mà bạn đã xác định trong file 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" thích hợp, tùy thuộc vào loại tài nguyên trong mảng. Trong trường hợp cụ thể này, nó chứa các ID tài nguyên, vì vậy bạn sử dụng phương thức getIdResource().

2. Sửa mã trong vòng lặp tạo các đối tượng cổng S, thêm ID tài nguyên có thể thô D thích hợp làm tham số thứ ba bằng cách gọi getIdResource() 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 dữ liệu cổng S ArrayList:

```
sportsImageResources.recycle();
```

2.5 Thêm ImageView vào các mục danh sách

- Thay đổi L trong taiBố cục bên trong **List_item.xml** tệp vào tệp RelativeLayout và xóa **Android:Định hướng** thuộc tính.
- Thêm **ImageView** làm phần tử đầu tiên trong **RelativeLayout** với các thuộc tính sau:

Thuộc tính	Giá trị
Android:layout_width	"match_parent"
Android:layout_height	"wrap_content"
android:id	"@+id/hình ảnh thể thao"
android:adjustViewBounds	"đúng"

Thuộc tính **djustViewBounds** làm cho **ImageView** điều chỉnh ranh giới của nó để duy trì tỷ lệ khung hình của hình ảnh.

- Thêm các thuộc tính sau vào phần tử **Title TextView**:

Thuộc tính	Giá trị
Android:layout_alignBottom	"Hình ảnh @id/thể thao"
Android:Chủ đề	"@style/ThemeOverlay.AppCompat.Dark"

4. Thêm các thuộc tính sau vào phần tử newsTitle TextView :

Android:layout_below	"Hình ảnh @id/thể thao"
android:textColor	"?android:textColorSecondary"

Thêm các thuộc tính sau vào

subTitle T phần tử extView :

Thuộc tính	Giá trị
Android:layout_below	"@+id/newsTitle"
Thuộc tính	Giá trị

- 5.

Dấu chấm hỏi trong thuộc tính textColor ở trên (" ?android:textColorSecondary") có nghĩa là framework sẽ áp dụng giá trị từ theme hiện đang được áp dụng. Trong trường hợp này, thuộc tính này được kế thừa từ giao diện "Theme.AppCompat.Light.DarkActionBar", định nghĩa nó là màu xám nhạt, thường được sử dụng cho các tiêu đề phụ.

2.6 Tải hình ảnh bằng Glide

Sau khi tải xuống hình ảnh và thiết lập ImageView, bước tiếp là sửa đổi SportsAdapter để tải hình ảnh vào ImageView trong onBindViewHolder(). Nếu bạn sử dụng cách tiếp cận này, bạn sẽ thấy rằng ứng dụng của mình gặp sự cố do lỗi "Hết bộ nhớ". Framework Android phải tải hình ảnh vào bộ nhớ mỗi lần ở độ phân giải đầy đủ, bất kể kích thước hiển thị của ImageView là bao nhiêu.

Có một số 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](#), bạn sẽ thực hiện trong bước này. Glide sử dụng xử lý nền, cũng như một số xử lý 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 giữ chỗ trong khi hình ảnh mong muốn được tải.

Lưu ý: Để tìm hiểu thêm về cách giảm mức tiêu thụ bộ nhớ trong ứng dụng của bạn, hãy xem bài viết [Loading Large Bitmaps một cách hiệu quả](#).

1. Mở tệp build.gradle (Module: app) và thêm phần phụ thuộc sau cho Glide trong tệp phần phụ thuộc:

```
trên khai 'com.github.bumptech.glide:glide:3.7.0'
```

2. Mở SportsAdapter và thêm một biến trong lớp ViewHolder cho ImageView:

```
ImageView mSportsImage riêng tư;
```

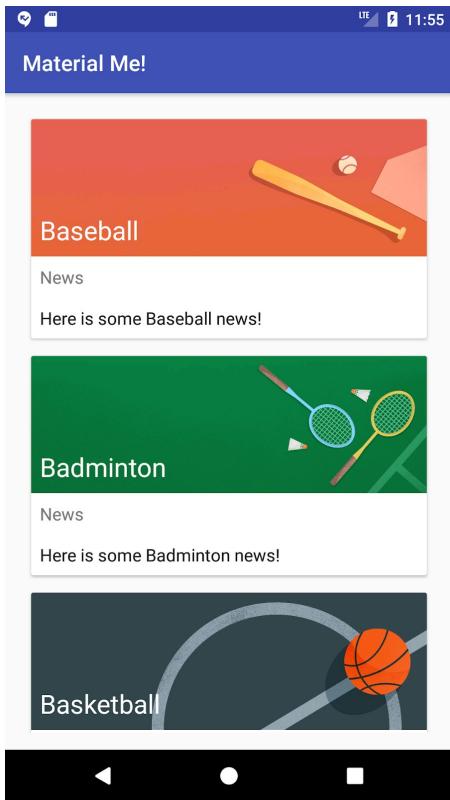
3. Khởi tạo biến trong hàm khởi tạo ViewHolder cho lớp ViewHolder:

```
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 cổng S và tải nó vào ImageView bằng cách sử dụng 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 làm cho 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 nhập

Khi người dùng nhìn thấy thẻ trong một ứng dụng, họ có kỳ vọng về cách thẻ hoạt động. Các [Nguyên tắc Material Design](#) nói rằng:

- Một thẻ có thể bị loại bỏ, thường là bằng cách quét nó 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 là [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 thao tác 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 **vào ItemTouchHelper**, 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 dưới đây.

```
ItemTouchHelper helper = new ItemTouchHelper(mới
                                         ItemTouchHelper.SimpleCallback() {
    @Override
    public boolean onMove(RecyclerView, recyclerView,
                         RecyclerView.ViewHolder viewHolder,
                         Mục tiêu RecyclerView.ViewHolder) {
        trả về false;
    }

    @Override
    public void onSwiped(RecyclerView.ViewHolder viewHolder,
                        int hướng) {
    } });
}
```

Nếu các phương pháp cần thiết không được thêm tự động, hãy nhấp vào bóng đèn màu đỏ ở lề trái và chọn **I mplement methods**.

Hàm khởi tạo SimpleCallback sẽ được gạch chân màu đỏ vì bạn chưa cung cấp các tham số bắt buộc: hướng mà bạn định hỗ trợ để di chuyển và vuốt các mục danh sách tương ứng.

2. Bởi vì chúng tôi chỉ thực hiện vuốt để loại bỏ vào lúc này, bạn nên chuyển vào 0 cho các hướng di chuyển được hỗ trợ và tôi temTouchHelper.LEFT | ItemTouchHelper.RIGHT để biết các hướng vuốt được hỗ trợ:

```
ItemTouchHelper helper = new ItemTouchHelper(ItemTouchHelper mới
                                         . SimpleCallback(0, ItemTouchHelper.LEFT |
                                         ItemTouchHelper.RIGHT) {
```

3. Bây giờ bạn phải triển khai 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, chuyển chỉ mục thích hợp bằng cách lấy vị trí từ Viewholder:

```
mSportsData.remove(viewHolder.getAdapterPosition());
```

4. Để cho phép RecyclerView tạo hiệu ứng cho quá trình 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 mình, giờ đây 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 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(ItemTouchHelper mới  
. SimpleCallback (ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT |  
ItemTauchelper.Down | Itemtauchelper.UP, Itemtauchelper.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 chủ sở hữu chế độ xem gốc và mục tiêu).

```
int từ = viewHolder.getAdapterPosition(); int  
thành = target.getAdapterPosition();
```

3. Hoán đổi các mục trong tập dữ liệu bằng cách gọi Collections.swap() và truyền tập dữ liệu cũng như các chỉ mục ban đầu và cuối cùng:

```
Collections.swap(mSportsData, từ, đến);
```

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.notifyDataSetChanged(); trả về  
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 sấp 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 của 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 chi tiết Activity sẽ được khởi chạy khi bất kỳ mục danh sách nào được nhấn. Đối với thực tế này, chi tiết Activity 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, vì vậy bạn không phải tạo chi tiết tùy chỉnh cho từng mục danh sách.

1. Tạo Activity mới bằng cách đi tới File > New > Activity > Empty Activity.
2. Gọi nó là **DetailActivity** và chấp nhận tất cả các mặc định.
3. Mở tệp bố cục activity_detail.xml **mới được tạo và** thay đổi gốc ViewGroup thành RelativeLayout, như bạn đã làm trong các bài tập trước.
4. Xóa câu lệnh xmlns:app="http://schemas.android.com/apk/res-auto" khỏi RelativeLayout.
5. Sao chép tất cả các TextView và Tile ma thuật View từ **List_item.xml** tệp vào tệp **activity_detail.xml** tệp.
6. Thêm từ "Detail" vào tham chiếu trong mỗi thuộc tính android:id để phân biệt nó với list_item.xml ID. Ví dụ: thay đổi ID ImageView từ **sportImage** thành **sportImageDetail**.
7. Trong tất cả các phần tử TextView và ImageView, thay đổi tất cả các tham chiếu đến các ID cho vị trí tương đối như layout_below sử dụng ID "Detail".
8. Đối với subtitleDetail TextView, xóa chuỗi văn bản giữ chỗ và dán một đoạn văn bản chung để thay thế văn bản chi tiết (Ví dụ: một vài đoạn của [Lorem Ipsum](#)). Trích xuất văn bản vào tài nguyên chuỗi.
9. Thay đổi khoảng đệm trên các phần tử TextView thành 16dp.
10. Bao bọc toàn bộ RelativeLayout bằng ScrollView. Thêm các thuộc tính layout_height và layout_width bắt buộc, và thêm xmlns:android="http://schemas.android.com/apk/res/android" vào cuối ScrollView.
11. Thay đổi thuộc tính layout_height của RelativeLayout thành "wrap_content".

Hai phần tử đầu tiên của bố cục activity_detail.xml bây giờ sẽ trông như sau:

```
<?xml version="1.0" encoding="utf8"?>
<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 **cổng SAdapter** và thay đổi lớp bên trong ViewHolder, đã mở rộng

RecyclerView.ViewHolder, để triển khai View.OnClickListener và triển khai phương thức bắt buộc (onClick()).

```
lớp ViewHolder mở rộng RecyclerView.ViewHolder
                    triển khai View.OnClickListener{
    Phần còn lại của mã ViewHolder.
    //
    @Override
    public void onClick(Xem chế độ
xem) {
    }
}
```

2. Đặt OnClickListener thành itemView trong hàm khởi tạo ViewHolder. Toàn bộ constructor bây giờ sẽ trông như thế này:

```
ViewHolder(Xem mụcView) {  
    siêu (itemView);  
  
    Khởi tạo các chế độ xem  
    mTitleText = itemView.findViewById(R.id.title);  
    mInfoText = itemView.findViewById(R.id.subTitle);  
    mSportsImage = itemView.findViewById(R.id.sportsImage);  
    Đặt OnClickListener thành toàn bộ chế độ xem.  
    itemView.setOnClickListener(cái  
này); }
```

3. Trong phương thức onClick(), lấy đối tượng cổng S cho mục đã được nhấp bằng getAdapterPosition():

```
Thẻ thao currentSport = mSportsData.get(getAdapterPosition());
```

4. Trong cùng một phương thức, thêm một Intent để khởi chạy DetailActivity, đặt title và image_resource như các phần bổ sung trong Intent, và gọi startActivity() trên biến m Context, truyền vào Intent mới.

```
Ý định detailIntent = new Intent(mContext, DetailActivity.class);  
detailIntent.putExtra("tiêu đề", currentSport.getTitle());  
detailIntent.putExtra("image_resource",  
                     currentSport.getImageResource());  
mContext.startActivity(detailIntent);
```

5. Mở **D etailActivity** và khởi tạo **I mageView** và **t itle T extView** trong **o nCreate()**:

```
TextView sportsTitle = findViewById(R.id.titleDetail);  
ImageView sportsImage = findViewById(R.id.sportsImageDetail);
```

6. Lấy **t itle** từ **I ntent** đến và đặt nó thành **T extView**:

```
sportsTitle.setText(getIntent().getStringExtra("tiêu đề"));
```

7. Sử dụng Glide để tải hình ảnh vào **I mageView**:

```
Glide.with(  
his).load(getIntent().ge  
tIntExtra("image_resource",0)) .into(sportsImage  
);
```

8. Chạy ứng dụng. Nhấn vào mục danh sách bây giờ sẽ khởi chạy **D etailActivity**.

Nhiệm vụ 4: Thêm FAB và chọn bảng màu Material Design

Một trong những nguyên tắc đầu tiên sau Material Design là sử dụng các yếu tố nhất quán trên các ứng dụng và nền tảng để người dùng nhận ra các mẫu và biết cách sử dụng chúng. Bạn đã sử dụng một yếu tố như vậy: Nút [Hành](#)

đóng Floating F (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 ký hiệu D](#).

1. Mở **tệp build.gradle (Mô-đun: Ứng dụng)** và thêm dòng mã sau cho thư viện hỗ trợ thiết kế trong phần dependencies :

```
trên khai '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 Project > Android và chọn **New > Vector Asset**. FAB sẽ đặt lại nội dung của

Recycler View, vì vậy biểu tượng refresh sẽ thực hiện:  . Thay đổi tên thành **ic_reset**, nhấp vào Tiếp theo và nhấp vào Finish.

3. Mở **activity_main.xml** và thêm FloatingActionButton với các thuộc tính sau:

Thuộc tính	Giá trị
Android:layout_width	"wrap_content"
Android:layout_height	"wrap_content"
Android:layout_alignParentBottom	"đúng"

Android:layout_alignParentRight	"Đúng
Android:layout_alignParentEnd	"Đúng
Android:layout_margin	"16 dtk"
android:src	"@drawable/ic_reset"
android:tint	"@android: màu / trắng"
android:bậtNhấp chuột	đặt lạiThể thao

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 Activity bị phá 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 liên tục (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).

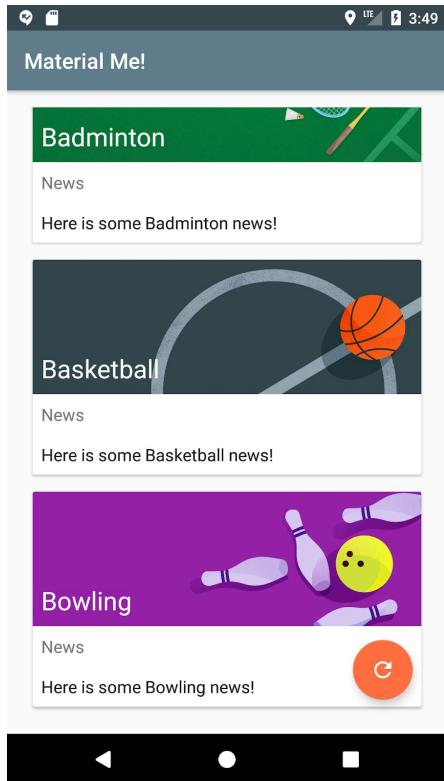
4.2 Chọn bảng màu Material Design

Material Design khuyên bạn nên chọn ít nhất ba màu này cho ứng dụng của bạn:

- Một 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).
- Một 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 Material 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 values). 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 Aterial M** để sử dụng làm màu chính của bạn, chẳng hạn như # 607D8B (trong mẫu màu Blue Grey). 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, 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`. Lưu ý rằng thao tác này sử dụng không gian tên `app`: và Android Studio sẽ nhắc bạn thêm câu lệnh để xác định không gian tên.

Mã giải pháp

Dự án Android Studio: [MaterialMe](#)

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 lập trình 1

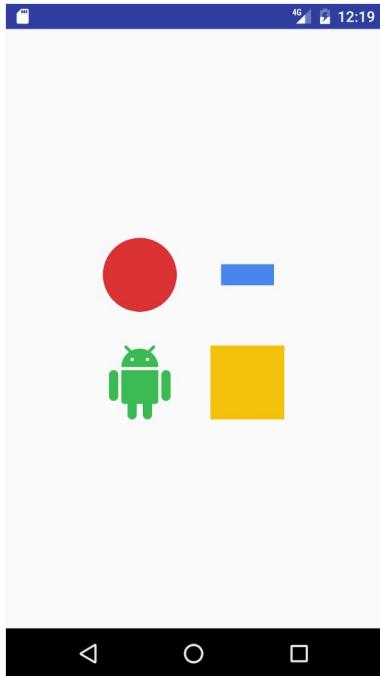
Thử thách này bao gồm hai cải tiến nhỏ đối với ứng dụng MaterialMe:

- Thêm chi tiết thực vào đối tượng Cổng S và chuyển các chi tiết đến DetailActivity.
- Triển khai một cách để đảm bảo rằng trạng thái của ứng dụng luôn tồn tại 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 nền màu đồng nhất đầu tiên với các hình dạng khác nhau (hình vuông, hình tròn, đường thẳng) và hình nền thứ tư là [Android Biểu tượng thiết kế vật liệu](#). Làm cho mỗi hình ảnh này phản hồi các nhấp chuột như sau:

- Một trong những khối màu khởi chạy lại Activity bằng cách sử dụng [hoạt ảnh Explode](#) cho cả chuyển tiếp vào và ra.
- Khởi chạy lại Activity từ một khối màu khác, lần này sử dụng chuyển đổi [Fade](#).
- Chạm vào khối màu thứ ba sẽ bắt đầu hoạt ảnh tại chỗ của View (chẳng hạn như xoay).
- Cuối cùng, chạm vào biểu tượng Android sẽ bắt đầu một Activity phụ với Chuyển đổi phần tử được chia sẻ hoán đổi khối Android với một trong các khối khác.



Lưu ý: Bạn phải đặt mức SDK tối thiểu thành 21 trở lên để triển khai chuyển đổi phần tử dùng chung.

Mã giải pháp thử thách 2

Dự án Android Studio: [TransitionsandAnimations](#)

Tóm tắt

- CardView là một bố cục tốt để sử dụng để trình bày thông tin có phương tiện hỗn hợp (chẳng hạn như hình ảnh và văn bản).
- CardView là một thành phần giao diện người dùng được tìm thấy trong Thư viện hỗ trợ Android.

- CardView không được thiết kế chỉ dành cho các phần tử View con văn bản.
- Tải hình ảnh trực tiếp vào ImageView tốn nhiều bộ nhớ, vì hình ảnh được tải ở độ phân giải đầy đủ. Để tải hình ảnh vào ứng dụng của bạn một cách hiệu quả, hãy sử dụng thư viện tải hình ảnh như Glide.
- Android SDK 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.
- Material Design 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 khuyến khích 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

Tài liệu khái niệm liên quan có trong [5.2: Material Design](#).

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 bằng 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 hình](#) • [Thư viện hỗ trợ](#)
- [Đồ họa có thể vẽ hoạt hình](#)
- [Tải bitmap lớn một cách hiệu quả](#)

Thiết kế vật liệu:

- [Thiết kế vật liệu cho Android](#)
- [Trình tạo bảng màu Material Design](#)
- [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](#)
- [Lướt trang github](#)

Homework

Xây dựng và chạy ứng dụng

Mở [Ứng dụng MaterialMe](#).

1. Tạo chuyển đổi phần tử shared 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 những câu hỏi này

Câu hỏi 1

Thuộc tính màu nào trong kiểu của bạn xác định màu của thanh trạng thái phía trên thanh ứng dụng? Chọn một:

- màuPrimary
- màuPrimaryDark
- colorAccent
- màuAccentDark

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 `background` trong ứng dụng của mình? Chọn một:

- Bất kỳ bóng màu nào bắt đầu bằng A.
- Bất kỳ bóng màu nào được dán nhãn 2 00.
- Bất kỳ bóng màu nào được dán nhãn 5 00.
- Bất kỳ màu nào được dán nhãn 9 00.

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.
- Quá trình 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 bằng thuộc tính `android:transitionName`.
- Mã sử dụng [phương thức](#) `ActivityOptions.makeSceneTransitionAnimation()`.

Bài 5.3: Bố cục thích ứng

Giới thiệu

Ứng dụng MaterialMe mà bạn đã tạo trong chương trước không xử lý đúng cách
Hướng thiết bị thay đổi 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 nguồn điện tử R 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 tiêu chuẩn có sẵn, hãy xem [cung cấp các 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 adaptive, 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 thành phần 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 nhấp chuột B.
- Sử dụng các tùy chọn có thể vẽ, kiểu và chủ đề.
- Trích xuất văn bản vào tài nguyên chuỗi và thứ nguyên vào 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.

Bạn sẽ làm gì

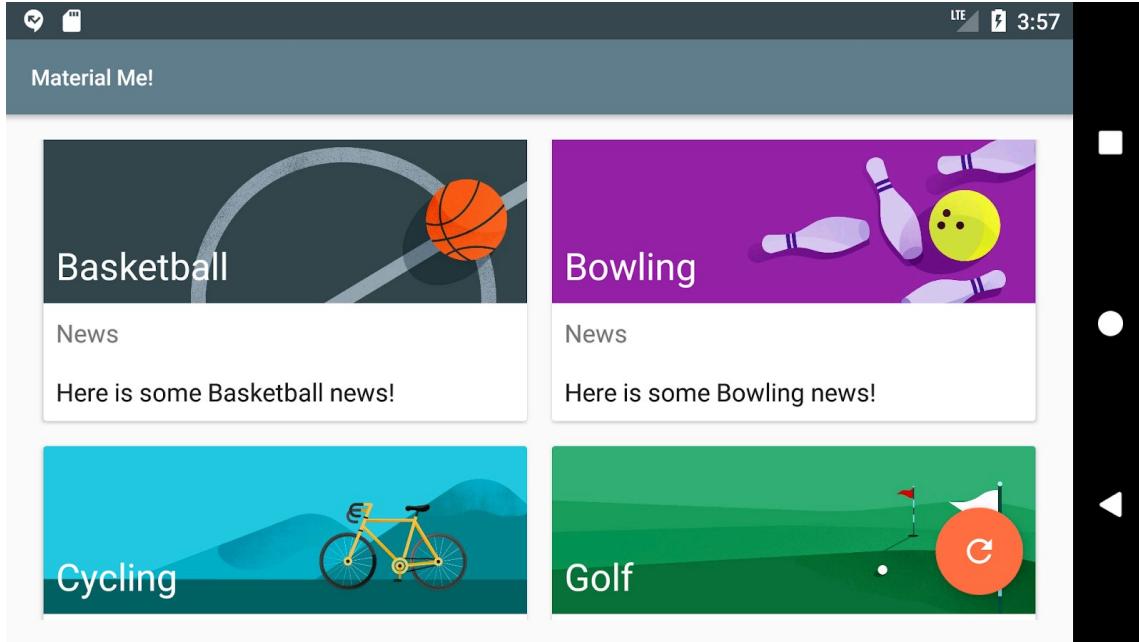
- Cập nhật ứng dụng MaterialMe để sử dụng không gian tốt hơn ở chế độ ngang.
- Thêm bố cục thay thế cho máy tính bảng.
- Bản địa hóa nội dung ứng dụng của bạn.

Tổng quan về ứng dụng

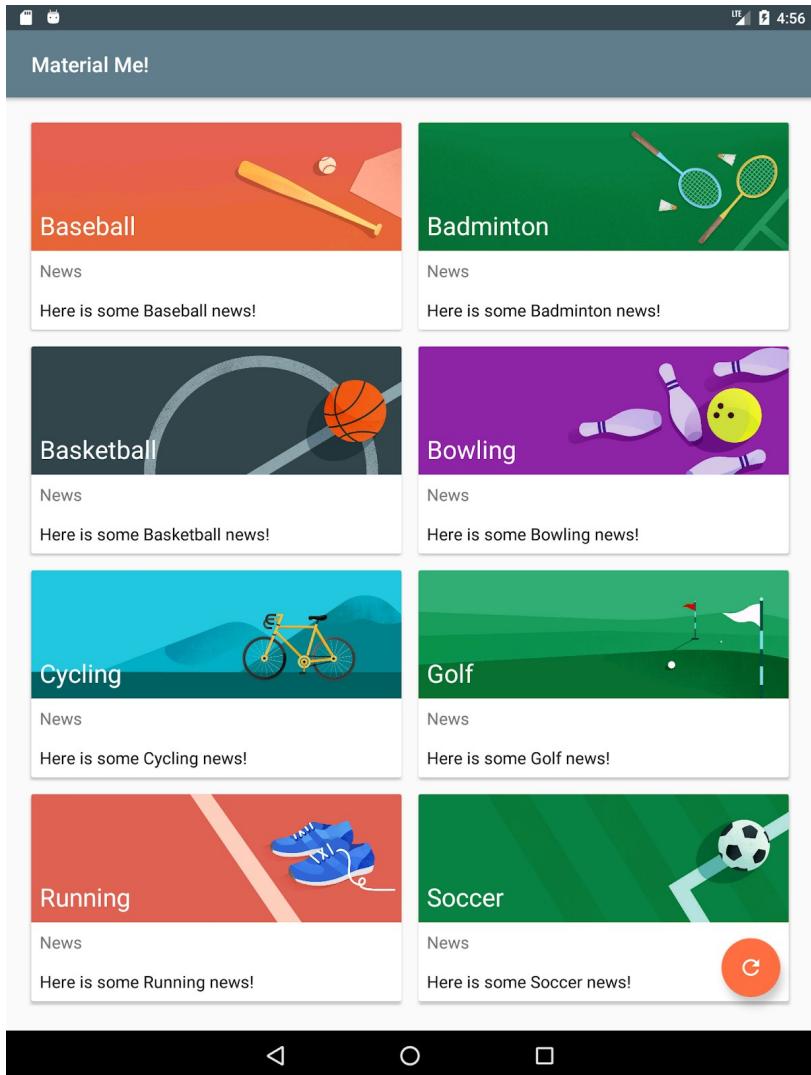
Ứng dụng MaterialMe được cập nhật sẽ bao gồm một bố cục được cải thiện cho chế độ ngang trên điện thoại. Nó cũng sẽ bao gồm bố cục được cải thiện cho các chế độ dọc và ngang trên máy tính bảng, đồng thời nó sẽ cung cấp nội dung bản địa hóa cho người dùng bên ngoài Hoa Kỳ.

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

Tác phẩm này được cấp phép theo Creative Commons Attribution 4.0 Giấy phép quốc tế. PDF này là
Ảnh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2 để cập
nhật mới nhất.

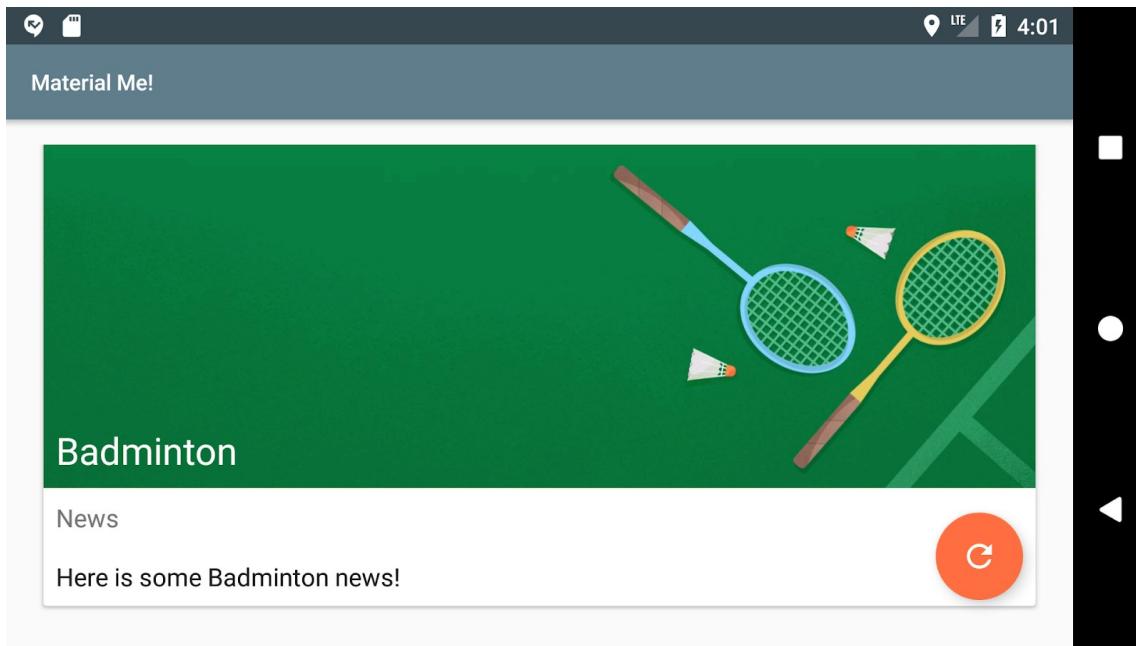


Ảnh chụp màn hình bên dưới cho thấy một máy tính bảng chạy ứng dụng MaterialMe được cập nhật, với các bộ hạn
định để hiển thị hai cột khi chạy theo hướng dọc:



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 sử 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 LinearLayoutManager

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 LinearLayoutManager; cho đến nay, bạn đã sử dụng [LinearLayoutManager](#) để 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: ứng dụng context 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 bảo toàn 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 `i ntegers.xml`. Để thực hiện việc này, hãy mở **thư mục res** trong ngăn **Project > Android**, right-click (hoặc Control-click) trên thư mục **values** và chọn **New > tệp tài nguyên Values**.
1. Đặt tên cho tệp **trong tegers.xml** và nhấp vào **OK**.
2. Tạo một hằng số nguyên giữa các thẻ tài nguyên < 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 **trong tegers.xml**; Tuy nhiên, tên sẽ được sửa đổi khi bạn thêm các bộ hạn định tài nguyên từ **Một vòng loại có sẵn** Ngắn. Các bộ hạn định tài nguyên được sử dụng để gắn nhãn tài nguyên Cấu hình `ations` cho các tinh huống khác nhau.
4. Chọn **O rientation** 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 S creen** thành **L andscape**, 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 vào **OK** để tạo tệp bố cục mới.
7. Sao chép hằng số 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 `i ntegers.xml` riêng lẻ được nhóm vào một thư mục `i ntegers.xml` trong ngăn **Project > Android**. Tệp thứ hai được gắn nhãn với bộ hạn định bạn đã chọn, đó là `l` và trong trường hợp này. 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 `i ntegers.xml`:

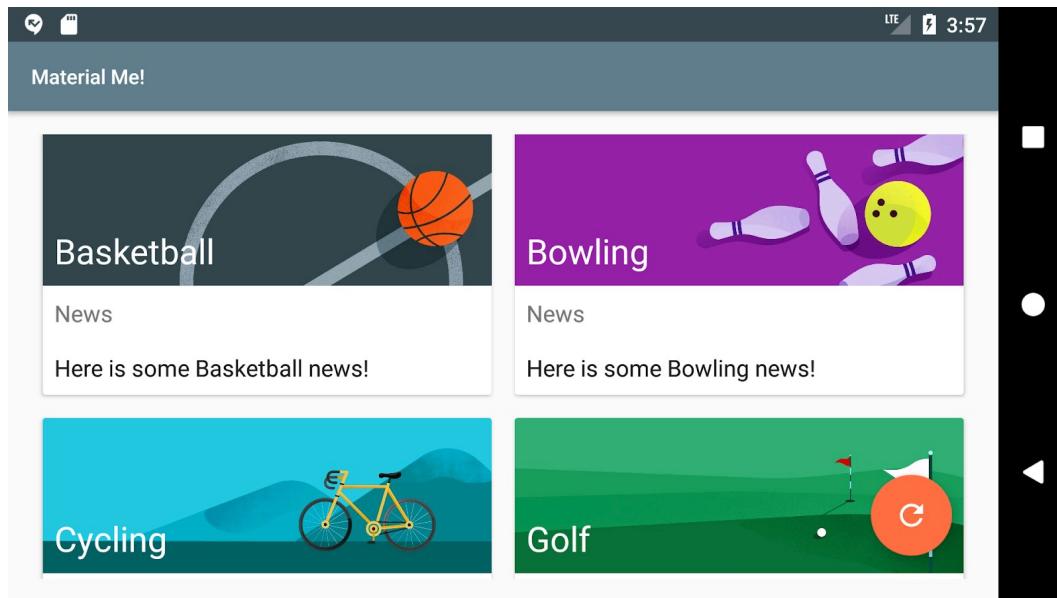
```
int gridColumnCount =  
    getResources().getInteger(R.integer.grid_column_count);
```

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( mới  
        GridLayoutManager(này, 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;  
} khác {  
    swipeDirs = ItemTouchHelper.LEFT | ItemTouchHelper.ĐÚNG; }
```

5. Sử dụng `swipeDirs` thay cho các đối số hướng vuốt (`ItemTouchHelper.LEFT` | `ItemTouchHelper.RIGHT`) cho tới `itemTouchHelper.SimpleCallback()`:

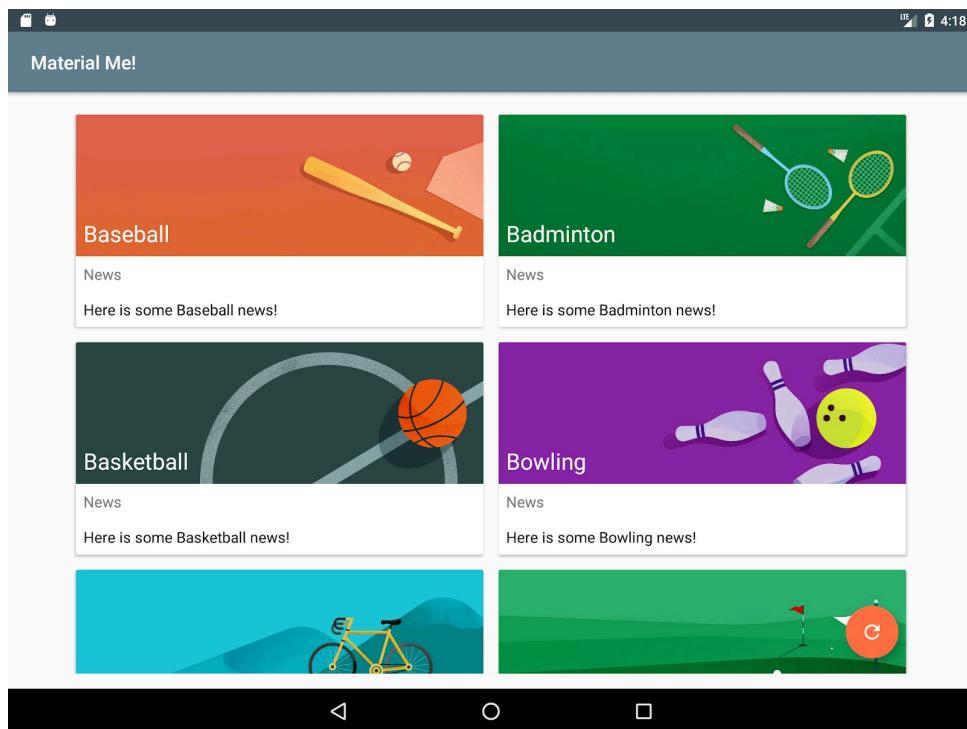
```
ItemTouchHelper helper = new ItemTouchHelper(mới  
    ItemTouchHelper.SimpleCallback(ItemTouchHelper.LEFT |  
        ItemTouchHelper.RIGHT |  
        ItemTauchelper.Down | Mụctauchelper.UP,  
        swipeDirs) {
```

- 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

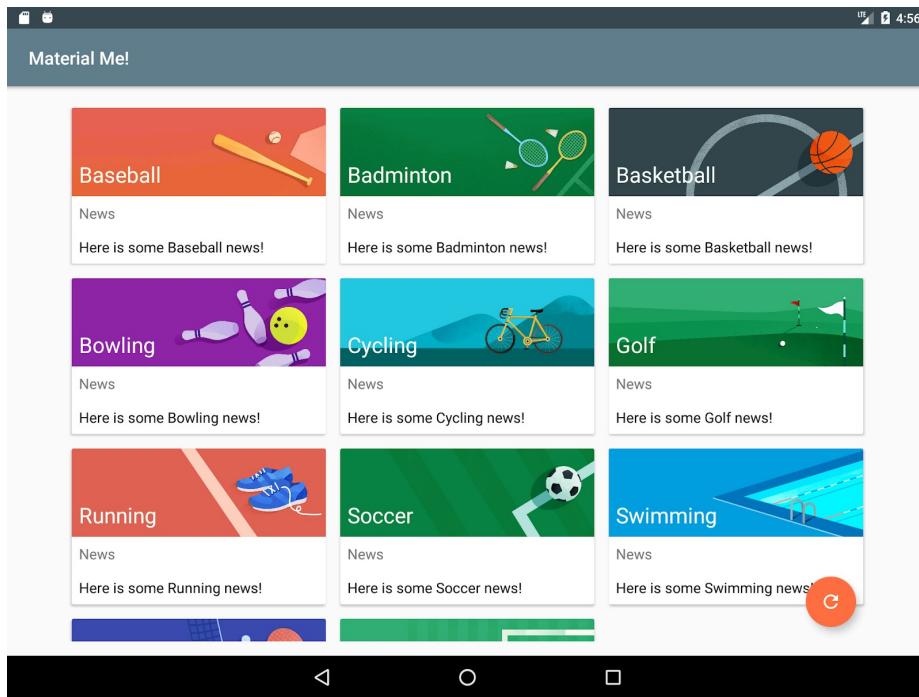
Trong bước này, bạn tạo các bộ hạn định tài nguyên khác nhau để tối đa hóa việc sử dụng màn hình cho các thiết bị có kích thước máy tính bảng, tăng số cột lên 2 cho hướng dọc (dọc) và 3 cho hướng ngang (ngang).

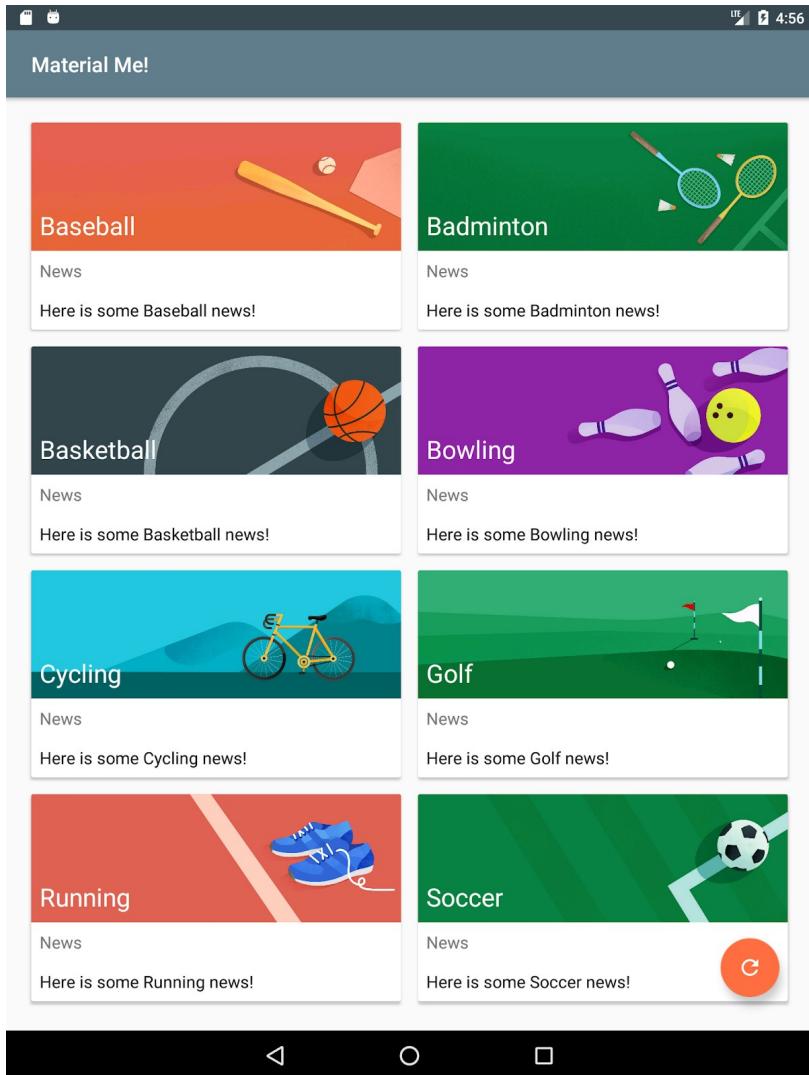
Bộ hạn định tài nguyên bạn cần tùy thuộc vào yêu cầu cụ thể của bạn. Khi tạo tệp tài nguyên mới, có một số bộ hạn định trong ngăn **Bộ hạn định có sẵn** mà bạn có thể sử dụng để chọn các điều kiện chính xác:

- **Chiều rộng màn hình nhỏ nhất:** Bộ hạn định này được sử dụng thường xuyên nhất để chọn cho máy tính bảng. Nó được xác định bởi *chiều rộng smallest* của thiết bị (bất kể hướng), giúp loại bỏ sự mơ hồ khi nói về "chiều cao" và "chiều rộng" vì một số thiết bị theo truyền thống được giữ ở chế độ ngang và những thiết bị khác ở chế độ dọc. Bất cứ thứ gì có chiều rộng nhỏ nhất ít nhất là 600dp được coi là máy tính bảng.
- **Chiều rộng màn hình:** Chiều rộng màn hình là chiều rộng hiệu ứng của thiết bị, bất kể hướng nào. Chiều rộng thay đổi khi thiết bị được xoay, vì chiều cao và chiều rộng hiệu quả của thiết bị được chuyển đổi.
- **Chiều cao màn hình:** Tương tự như **Chiều rộng Screen**, ngoại trừ nó sử dụng chiều cao hiệu quả thay vì chiều rộng hiệu quả.

Để bắt đầu tác vụ này:

1. Tạo một tệp tài nguyên `in_tegers.xml` sử dụng bộ hạn định **Độ rộng màn hình S mallest** với giá trị được đặt thành **6 00**. Android sử dụng tệp này bất cứ khi nào ứng dụng chạy trên máy tính bảng.
2. Sao chép mã từ tệp **in_tegers.xml (đất)** (nó có số lượng lưới là 2) và dán vào tệp mới **trong integers.xml (sw600dp)**.
3. Tạo một tệp khác trong `tegers.xml` bao gồm cả bộ hạn định **Chiều rộng màn hình nhỏ nhất** được đặt thành **6 00**, và bộ hạn định **Orientation** được đặt thành **L và scape**. Android sử dụng tệp `integers.xml` kết quả (`sw600dp-land`) khi ứng dụng chạy trên máy tính bảng ở chế độ ngang.
4. Sao chép mã từ tệp **in_tegers.xml (land)** và dán vào tệp mới **in_tegers.xml (sw600dp-land)**.
5. Thay đổi biến `grid_column_count` thành 3 trong tệp `in_tegers.xml (sw600dp-land)`.
6. Chạy ứng dụng trên trình giả lập máy tính bảng hoặc máy tính bảng và xoay ứng dụng sang chế độ ngang. Ứng dụng sẽ hiển thị ba cột thẻ, như trong hình đầu tiên bên dưới. Xoay nó sang chế độ dọc và ứng dụng sẽ hiển thị hai cột thẻ, như trong hình thứ hai bên dưới. Với các tệp qualifier tài nguyên này, ứng dụng sử dụng bất động sản màn hình hiệu quả hơn nhiều.





Mẹo: Nếu ứng dụng của bạn sử dụng nhiều tệp tài nguyên, Android sẽ sử dụng tệp tài nguyên có bộ hạn định tài nguyên cụ thể nhất trước. Ví dụ: nếu một giá trị được xác định trong `integers.xml` cho cả bộ **hạn định Chiều rộng** màn hình nhỏ nhất **và với mức độ 0** được đặt thành **Landscape**, Android sẽ sử dụng giá trị cho Chiều rộng màn hình nhỏ nhất. Mức độ ưu tiên cho các bộ hạn định tài nguyên và các tệp tài nguyên được mô tả bởi Bảng 2 trong tổng [quản lý tài nguyên App](#).

2.2 Cập nhật kiểu mục danh sách máy tính bảng

Tại thời điểm này, ứng dụng của bạn sẽ thay đổi số cột trong GridLayoutManager để phù hợp với hướng của thiết bị và tối đa hóa việc sử dụng màn hình. Tuy nhiên, các phần tử TextView xuất hiện có kích thước chính xác trên màn hình điện thoại giờ đây xuất hiện quá nhỏ so với màn hình lớn hơn của máy tính bảng.

Để khắc phục điều này, bạn sẽ trích xuất các kiểu TextAppearance từ các tệp tài nguyên bố cục vào tệp tài nguyên styles.xml. Bạn cũng sẽ tạo các tệp styles.xml bổ sung cho máy tính bảng bằng cách sử dụng bộ hạn định tài nguyên.

Lưu ý: Bạn cũng có thể tạo các tệp bố cục thay thế với các bộ hạn định tài nguyên thích hợp và thay đổi kiểu của các phần tử TextView trong đó. Tuy nhiên, điều này sẽ đòi hỏi nhiều mã trùng lặp hơn, bởi vì hầu hết thông tin bố cục đều giống nhau cho dù bạn sử dụng thiết bị nào, vì vậy bạn sẽ chỉ trích xuất các thuộc tính sẽ thay đổi.

Làm theo các bước sau để thêm kiểu TextAppearance:

1. Mở styles.xml và thêm các kiểu sau:

```
<style name="SportsDetailText"
      parent="TextAppearance.AppCompat.Subtitle"/> <style
name="SportsTitle"
      parent="TextAppearance.AppCompat.Headline"/>
```

2. Tạo một tệp tài nguyên values mới có tên là styles.xml sử dụng **bộ hạn định Chiều rộng** màn hình Smallest với giá trị là **600** cho máy tính bảng.
3. Sao chép một kiểu `ll` từ tệp styles.xml ban đầu vào tệp styles.xml (sw600dp) mới .
4. Trong styles.xml (sw600dp), thay đổi `parent` của kiểu SportsTitle thành "TextAppearance.AppCompat.Display1":

```
<style name="Tiêu đề thể thao"
```

```
parent="TextAppearance.AppCompat.Display1"/>
```

5. Kiểu Display1 được xác định trước trên Android sử dụng giá trị textColorSecondary từ giao diện hiện tại (ThemeOverlay.AppCompat.Dark), trong trường hợp này là màu xám nhạt. Màu xám nhạt không hiển thị tốt trên hình ảnh biểu ngữ trong ứng dụng của bạn. Để khắc phục điều này, hãy thêm thuộc tính " android:textColor" vào kiểu SportsTitle và đặt nó thành "?android:textColorPrimary":

```
<style name="Tiêu đề thể thao"  
       parent="TextAppearance.AppCompat.Display1"> <tên mục=  
           "android:textColor">?android:textColorPrimary</item> </style>
```

Dấu chấm hỏi yêu cầu thời gian chạy Android tìm giá trị trong giao diện được áp dụng cho View. Trong ví dụ này, theme là ThemeOverlay.AppCompat.Dark trong đó thuộc tính textColorPrimary có màu trắng.

6. Thay đổi kiểu p không phải của SportsDetailText thành "TextAppearance.AppCompat.Headline" (bằng tiếng Anh).
7. Để cập nhật kiểu của TextView phần tử, mở List_item.xml và thay đổi Style thuộc tính của thuộc tính title TextView đến @ phong cách/SportsTitle:

```
style="@style/Tiêu đề thể thao"
```

8. Thay đổi thuộc tính style của các phần tử newsTitle và subTitle TextView **thành @style/SportsDetailText**.
 9. Chạy ứng dụng của bạn trên máy tính bảng hoặc trình mô phỏng máy tính bảng. Mỗi mục danh sách hiện có kích thước văn bản lớn hơn trên máy tính bảng.
-

Tác phẩm này được cấp phép theo Creative Commons Attribution 4.0 Giấy phép quốc tế. PDF này là Ánh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2 để cập nhật mới nhất.

Trang 250

2.3 Cập nhật các kiểu chi tiết thể thao của máy tính bảng

Bây giờ bạn đã sửa màn hình cho MainActivity, liệt kê tất cả các cổng CardView phần tử . DetailActivity vẫn có cùng kích thước phông chữ trên máy tính bảng và điện thoại.

- Thêm style sau trong tệp styles.xml cho tiêu đề chi tiết:

```
<style name="SportsDetailTitle"
      parent="TextAppearance.AppCompat.Headline"/>
```

- Thêm style sau trong tệp styles.xml (sw600dp) cho tiêu đề chi tiết:

```
<style name="SportsDetailTitle"
      parent="TextAppearance.AppCompat.Display3"/>
```

- Mở một ctivity_detail.xml và thay đổi thuộc tính style của cả hai phần tử newsTitleDetail và subTitleDetail TextView thành kiểu SportsDetailText mới mà bạn đã tạo ở bước trước:

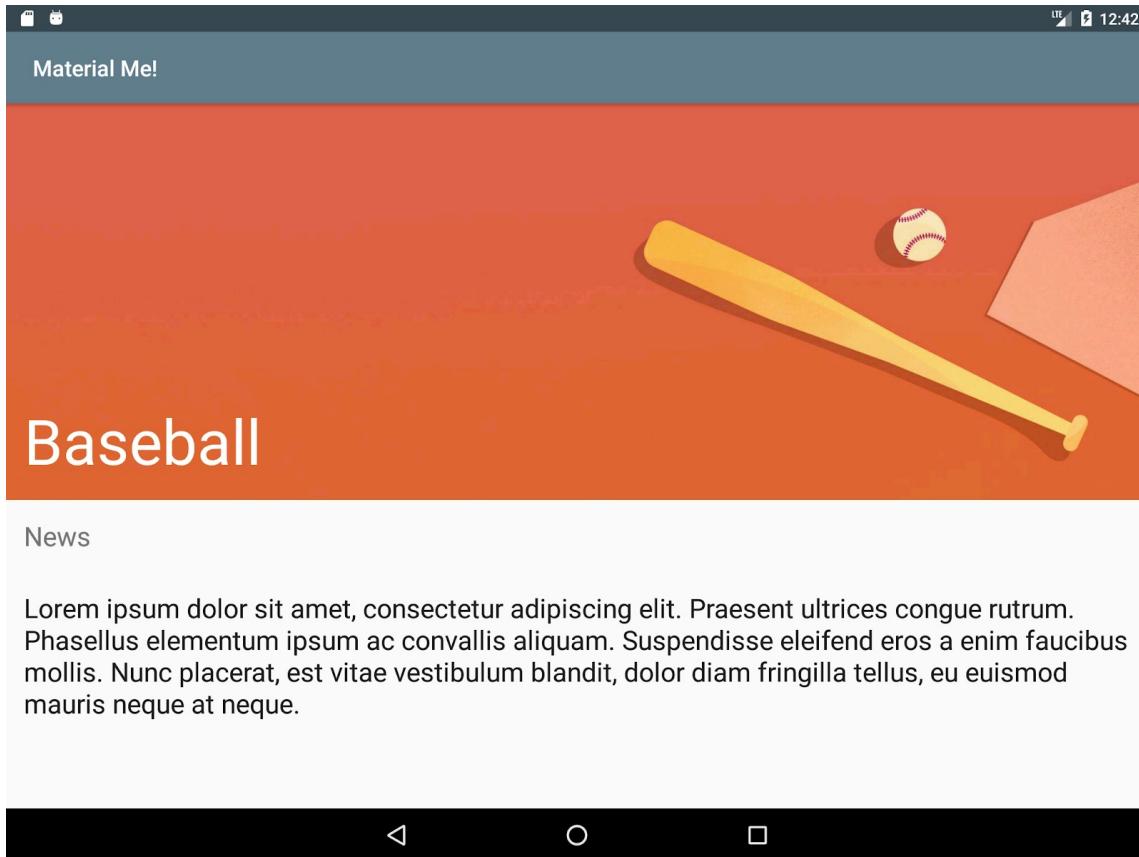
```
style="@style/SportsDetailText"
```

4. Trong **một ctivity_detail.xml**, thay đổi thuộc tính style của phần tử titleDetail TextView thành kiểu SportsDetailTitle mới mà bạn đã tạo:

```
style="@style/SportsDetailTitle"
```

Trang 251

5. Chạy ứng dụng của bạn. Tất cả văn bản hiện lớn hơn trên máy tính bảng, giúp cải thiện đáng kể trải nghiệm người dùng ứng dụng của bạn.



Nhiệm vụ 3: Bản địa hóa ứng dụng của bạn

"Địa phương" đại diện cho một khu vực địa lý, chính trị hoặc văn hóa cụ thể trên thế giới. Có thể sử dụng bộ hạn định tài nguyên để cung cấp tài nguyên thay thế dựa trên ngôn ngữ của người dùng. Cũng như đối với

creative Commons để cập

nhật mới nhất. hướng

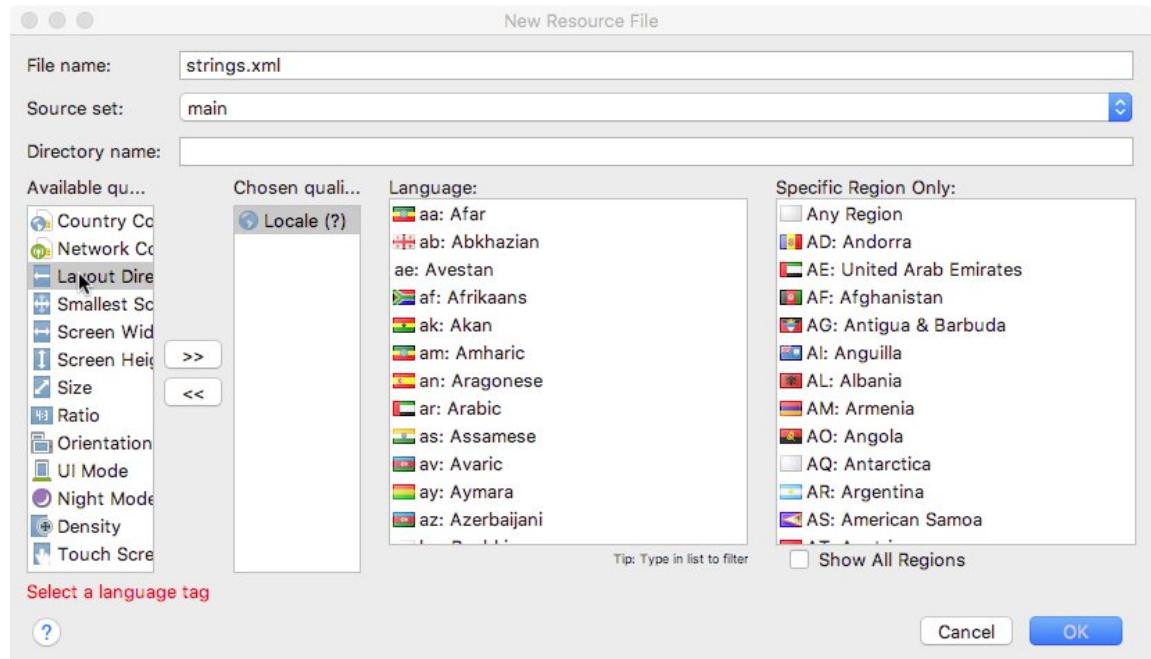
và chiều rộng màn hình,
Android cung cấp khả
năng bao gồm các tệp
tài nguyên riêng biệt
cho các ngôn ngữ khác
nhau. Trong bước này,
bạn sẽ sửa đổi tệp
strings.xml của mình để
mang tính quốc tế hơn
một chút.

3.1 Thêm tệp strings.xml bản địa hóa

Bạn có thể nhận thấy rằng thông tin thể thao có trong ứng dụng này được thiết kế cho người dùng từ Hoa Kỳ. Ứng dụng sử dụng thuật ngữ "bóng đá" để đại diện cho một môn thể thao được gọi là "bóng đá" ở mọi nơi khác trên thế giới.

Để làm cho ứng dụng của bạn được quốc tế hóa hơn, bạn có thể cung cấp tệp strings.xml theo ngôn ngữ cụ thể. Tệp tài nguyên thay thế này sẽ hiển thị từ "bóng đá" cho người dùng ở Hoa Kỳ. Tệp strings.xml chung sẽ hiển thị từ "bóng đá" cho người dùng ở tất cả các ngôn ngữ khác.

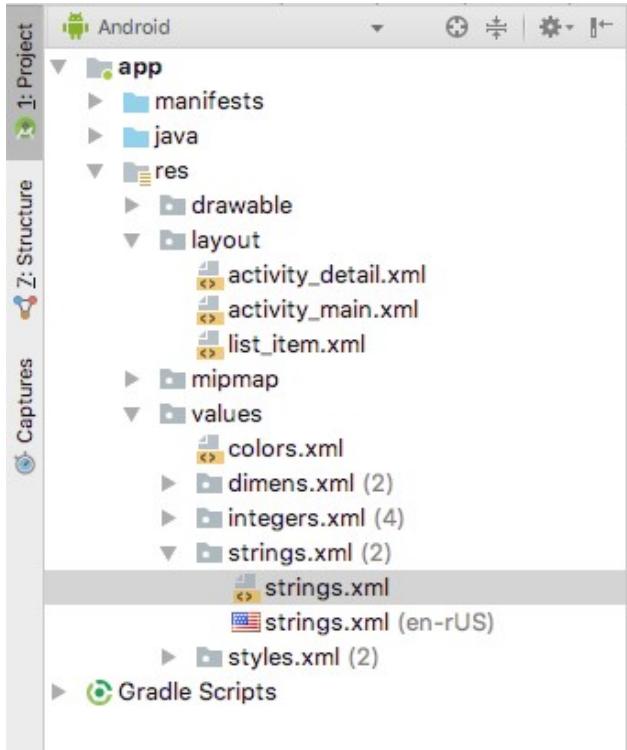
1. Tạo một tệp tài nguyên values mới.
2. Gọi tệp strings.xml và chọn Locale từ danh sách các bộ hạn định có sẵn. Các ngăn Ngôn ngữ và Chỉ vùng cụ thể xuất hiện.



3. Trong ngăn ngôn ngữ L, chọn en: Tiếng Anh.

Trang 253

4. Trong ngăn Chỉ khu vực đặc biệt S, chọn US S: United States và nhập vào OK. Android Studio tạo một thư mục values-cụ thể trong thư mục dự án của bạn cho ngôn ngữ Hoa Kỳ, được gọi là values-en-rUS. Trong ngăn Project > Android, tệp strings.xml trong thư mục này xuất hiện dưới dạng strings.xml (en-rUS) trong thư mục strings.xml mới được tạo (với biểu tượng cờ Hoa Kỳ).



5. Sao chép tất cả các tài nguyên chuỗi của **tệp s trings.xml chung** (hiện nằm trong **thư mục s trings.xml**) vào **s trings.xml (en-rUS)**.
6. Trong tệp s trings.xml **chung**, thay đổi **Mục S occer** trong mảng s ports_titles thành **Football** và thay đổi **văn bản** tin tức S occer trong mảng s ports_info thành **F ootball news**.

3.2 Chạy ứng dụng ở các ngôn ngữ khác nhau

Để xem sự khác biệt theo ngôn ngữ cụ thể, bạn có thể khởi động thiết bị hoặc trình mô phỏng của mình và thay đổi ngôn ngữ và ngôn ngữ của thiết bị hoặc ngôn ngữ thành tiếng Anh Mỹ (nếu chưa đặt). Trong tiếng Anh Mỹ, bạn sẽ thấy "Bóng đá".

reative
Commons để cập nhật
mới nhất.

Sau đó, có thể chuyển sang bất kỳ ngôn ngữ và ngôn ngữ nào khác ngoài tiếng Anh Mỹ và chạy lại ứng dụng. Sau đó, bạn sẽ thấy "Bóng đá".

- Để chuyển đổi ngôn ngữ ưa thích trong thiết bị hoặc trình mô phỏng, hãy mở ứng dụng Cài đặt. Nếu thiết bị Android của bạn bằng ngôn ngữ khác, hãy tìm biểu tượng bánh răng:



- Tìm các **câu hỏi L & đầu vào** trong ứng dụng Cài đặt, và chọn **các kiểu chữ L**.

Hãy nhớ biểu tượng quả địa cầu cho **L ngôn ngữ & đầu vào** Choice, để bạn có thể tìm lại nó nếu bạn chuyển sang một ngôn ngữ bạn không hiểu. **Tần số L** là lựa chọn đầu tiên trên **L ngôn ngữ & đầu vào** màn.



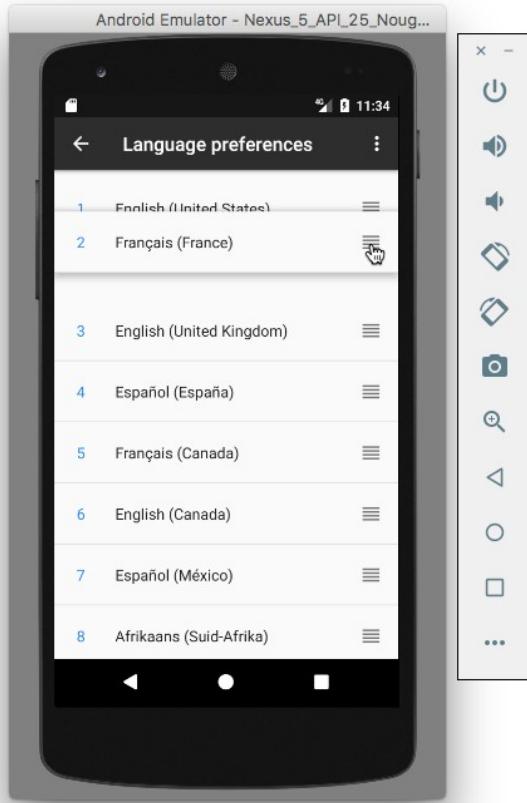
- Đối với các thiết bị và trình giả lập chạy phiên bản Android trước Android 7, hãy chọn **Ngôn ngữ** trên **màn hình ngôn ngữ & đầu vào L**, chọn ngôn ngữ và ngôn ngữ như **Français (Pháp)** và bỏ qua các bước sau.

(Trong các phiên bản Android trước Android 7, người dùng chỉ có thể chọn một ngôn ngữ. Trong Android 7 trở lên, người dùng có thể chọn nhiều ngôn ngữ và sắp xếp chúng theo sở thích. Ngôn ngữ chính được đánh số 1, như trong hình sau, tiếp theo là các ngôn ngữ ưu tiên thấp hơn.)

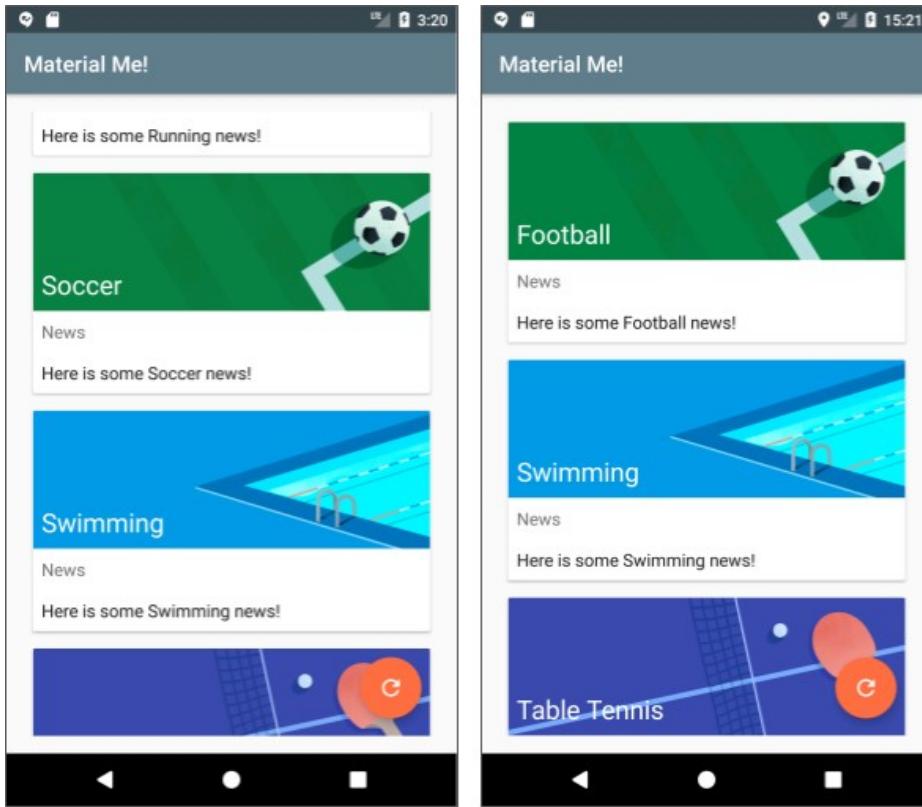
- Đối với các thiết bị và trình giả lập chạy Android 7 trở lên, hãy chọn **ngôn ngữ L** trên **màn hình Ngôn ngữ & đầu vào**, chọn một ngôn ngữ như **Français (Pháp)** và sử dụng biểu tượng di chuyển ở phía bên phải của **màn hình tùy chọn ngôn ngữ L** để kéo **Français (Pháp)** lên đầu danh sách.

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0. PDF này làẢnh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2

Trang 255



5. Chạy ứng dụng bằng thiết bị hoặc trình giả lập của bạn. Trong tiếng Anh Mỹ, bạn sẽ thấy "Bóng đá".
6. Chuyển sang bất kỳ ngôn ngữ và ngôn ngữ nào khác ngoài tiếng Anh Hoa Kỳ và chạy lại ứng dụng. Sau đó, bạn sẽ thấy "Bóng đá".



Ví dụ này không hiển thị một từ được dịch cho "Bóng đá" tùy thuộc vào ngôn ngữ. Để biết bài học về cách bản địa hóa ứng dụng với bản dịch, hãy xem A [dvanced Android Development - Thực hành](#).

Mã giải pháp

Dự án Android Studio: [M aterialMe-Resource](#)

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 này.

Thách thức 1: Hóa ra một số quốc gia khác ngoài Hoa Kỳ sử dụng "bóng đá" thay vì "bóng đá". Nghiên cứu các quốc gia này và thêm tài nguyên chuỗi bản địa hóa cho chúng.

Thử thách 2: Sử dụng các kỹ thuật bản địa hóa bạn đã học trong Nhiệm vụ 3 kết hợp với Google dịch để dịch tất cả các chuỗi trong ứng dụng của bạn sang một ngôn ngữ khác.

Tóm tắt

- GridLayoutManager là một trình quản lý bố cục xử lý các danh sách cuộn hai chiều.
- Bạn có thể tự động thay đổi số lượng cột trong GridLayoutManager.
- Thời gian chạy Android sử dụng các tệp cấu hình thay thế, tùy thuộc vào môi trường thời gian chạy của thiết bị chạy ứng dụng của bạn. Ví dụ: thời gian chạy có thể sử dụng các tệp cấu hình thay thế cho các bộ phận thiết bị, kích thước màn hình, ngôn ngữ, quốc gia hoặc loại bàn phím khác nhau.
- Trong mã, bạn tạo các tài nguyên thay thế này để thời gian chạy Android sử dụng. Các tài nguyên nằm trong các tệp có bộ hạn định tài nguyên như một phần của tên của chúng. • Định dạng cho thư mục chứa các tệp tài nguyên thay thế là
`<resource_name>-<đủ điều kiện>`.
- Bạn có thể đủ điều kiện bất kỳ tệp nào trong thư mục `res` của mình theo cách này.

Các khái niệm liên quan

Khái niệm liên quan có trong [5.3: Tài nguyên cho bố cục thích ứng](#).

Tìm hiểu thêm

Tài liệu Android Studio: [Hướng dẫn sử dụng Android Studio](#)

Tài liệu dành cho nhà phát triển Android:

- [Tổng quan về tài nguyên](#)
- [Cung cấp tài nguyên](#) • [Bản địa hóa với tài nguyên](#)
- [Trình quản lý bố cục tuyến tính](#) • [Trình quản lý bố cục lưới](#)
- [Hỗ trợ nhiều màn hình](#)

Thiết kế vật liệu:

- [Thiết kế vật liệu cho Android](#)
- [Nguyên tắc thiết kế vật liệu](#)

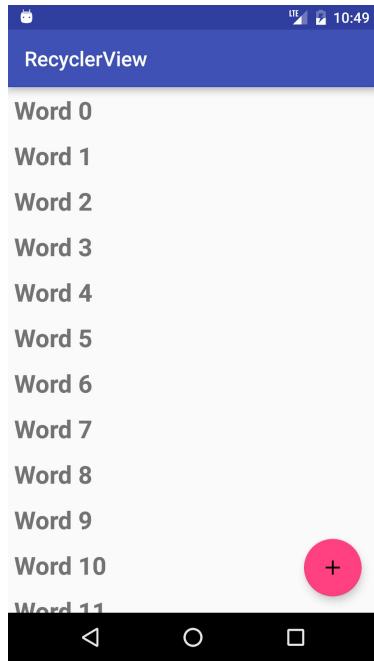
Homework

Xây dựng và chạy ứng dụng

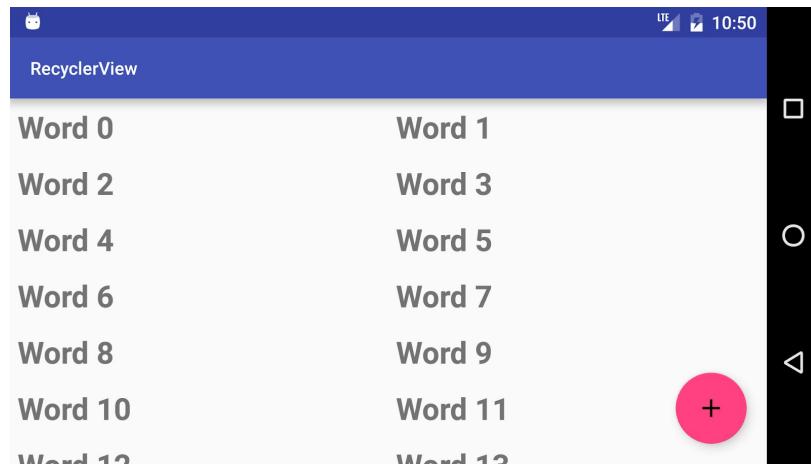
Sửa đổi ứng dụng [RecyclerView](#) để sử dụng GridLayoutManager với số lượng cột sau:

- Đối với điện thoại: 1 cột theo hướng dọc, 2 cột theo hướng ngang
- Đối với máy tính bảng: 2 cột theo hướng dọc, 3 cột theo hướng ngang

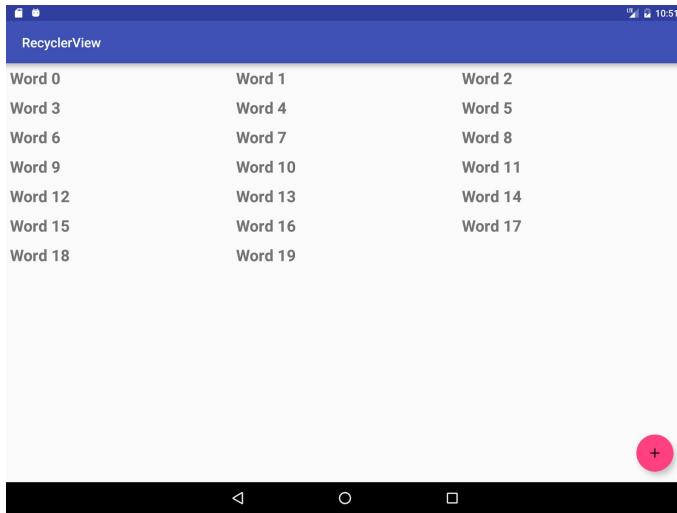
Ảnh chụp màn hình bên dưới cho thấy RecyclerView đủ điều kiện tài nguyên trên điện thoại theo hướng dọc:



Ảnh chụp màn hình bên dưới cho thấy RecyclerView dù điều kiện tài nguyên trên điện thoại theo hướng ngang:



Ảnh chụp màn hình bên dưới cho thấy RecyclerView dù điều kiện tài nguyên trên máy tính bảng theo hướng ngang:



Trả lời những câu hỏi này

Câu hỏi 1

Bộ hạn định tài nguyên nào được sử dụng thường xuyên nhất để chọn cho máy tính bảng? Chọn một:

- Hướng
- Chiều rộng màn hình
- Chiều cao màn hình
- Chiều rộng màn hình nhỏ nhất

Câu hỏi 2

Thư mục nào sẽ chứa tệp strings.xml để dịch sang tiếng Pháp cho Canada? Chọn một:

- res/values-fr-rFR/
- res/values-ca-rFR/
- res/values-fr-rCA/

- res/values-en-rFR/

Trang 261

Câu hỏi 3

Thư mục nào dành cho các tệp XML chứa chuỗi, số nguyên và màu sắc? Chọn một:

- Res/Bố cục
- res/mipmap
- res/raw
- Res/Giá trị

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:

- Đối với điện thoại và máy tính bảng ở cả chế độ ngang và dọc, mã bao gồm các tệp values đủ điều kiện tài nguyên chứa số nguyên cho số cột.
- Ứng dụng sử dụng `getResources().getInteger()` để truy xuất giá trị từ tệp tài nguyên, sau đó sử dụng giá trị đó làm số lượng cột cho bố cục lưới.

Bài 6.1: Espresso để kiểm tra giao diện người dùng

Giới thiệu

Là nhà phát triển, điều quan trọng là bạn phải kiểm tra các tương tác của người dùng trong ứng dụng của mình để đảm bảo rằng người dùng không gặp phải kết quả không mong muốn hoặc có trải nghiệm kém với ứng dụng của bạn.

Bạn có thể kiểm tra giao diện người dùng (UI) của ứng dụng theo cách thủ công bằng cách chạy ứng dụng và thử giao diện người dùng. Nhưng đối với một ứng dụng phức tạp, bạn không thể bao gồm tất cả các hoán vị tương tác của người dùng trong tất cả các ứng dụng

creativecommons.org/licenses/by/4.0/deed.vi
Commons để cập nhật
mới nhất. Chức năng.
Bạn cũng sẽ phải lặp lại
các bài kiểm tra thủ công
này cho các cấu hình
thiết bị khác nhau trong
trình giả lập và trên các
thiết bị phần cứng khác
nhau.

Khi bạn tự động hóa các thử nghiệm tương tác giao diện người dùng, bạn tiết kiệm thời gian và thử nghiệm của bạn có hệ thống. Bạn có thể sử dụng các bộ kiểm tra tự động để tự động thực hiện tất cả các tương tác giao diện người dùng, giúp chạy thử nghiệm cho các cấu hình thiết bị khác nhau dễ dàng hơn. Để xác minh rằng giao diện người dùng của ứng dụng hoạt động chính xác, bạn nên tạo thói quen tạo các bài kiểm tra giao diện người dùng khi viết mã.

Espresso là một khung thử nghiệm dành cho Android giúp bạn dễ dàng viết các bài kiểm tra giao diện người dùng đáng tin cậy cho một ứng dụng. Khung này, là một phần của Kho lưu trữ hỗ trợ Android, cung cấp các API để viết các bài kiểm tra giao diện người dùng để mô phỏng các tương tác của người dùng trong ứng dụng—mọi thứ từ nhấp vào các nút và điều hướng chế độ xem đến chọn các mục menu và nhập dữ liệu.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy ứng dụng trong Android Studio.
- Kiểm tra Kho lưu trữ hỗ trợ Android và cài đặt nó nếu cần.
- Tạo và chỉnh sửa các thành phần giao diện người dùng bằng trình chỉnh sửa bố cục và XML.
- Truy cập các thành phần giao diện người dùng từ mã của bạn.
- Thêm trình xử lý nhấp chuột vào nút B.

Những gì bạn sẽ học

Bạn sẽ học cách:

- Thiết lập Espresso trong dự án ứng dụng của bạn.
- Viết bài kiểm tra Espresso.
- Kiểm tra đầu vào của người dùng và kiểm tra đầu ra chính xác.
- Tìm một con quay, nhấp vào một trong các mục của nó và kiểm tra đầu ra chính xác.
- Sử dụng chức năng **R ecord Espresso Test** trong Android Studio.

Bạn sẽ làm gì

- Sửa đổi dự án để tạo thử nghiệm Espresso.
- Kiểm tra đầu vào và đầu ra văn bản của ứng dụng.
- Kiểm tra việc nhấp vào một mục spinner và kiểm tra đầu ra của nó. ● Ghi lại bài kiểm tra Espresso của một **R ecycler View**.

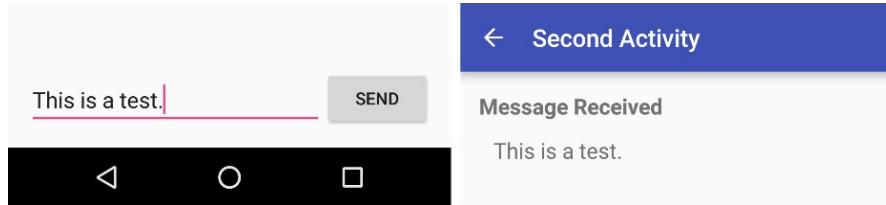
creative
Commons Khóa học
cơ bản về nhà phát
triển Android (V2)
Đơn vị

Tổng quan về ứng dụng

Mẹo: Để biết thông tin giới thiệu về cách thử nghiệm ứng dụng Android, hãy xem phần [Test ứng dụng của bạn](#).

Trong thực tế này, bạn sửa đổi dự án [TwoActivities](#) từ bài học trước. Bạn thiết lập Espresso trong dự án để thử nghiệm, sau đó kiểm tra chức năng của ứng dụng.

Ứng dụng TwoActivities cho phép người dùng nhập văn bản vào trường văn bản và nhấn vào **nút kết thúc S**, như được hiển thị ở phía bên trái của hình bên dưới. Trong hình A thứ hai, người dùng xem văn bản họ đã nhập, như thể hiện ở phía bên phải của hình bên dưới.



Nhiệm vụ 1: Thiết lập Espresso trong dự án của bạn

Để sử dụng Espresso, bạn phải cài đặt Kho lưu trữ hỗ trợ Android cùng với Android Studio. Bạn cũng có thể cần định cấu hình Espresso trong dự án của mình.

Trong tác vụ này, bạn kiểm tra xem kho lưu trữ đã được cài đặt chưa. Nếu không, bạn sẽ cài đặt nó.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#). PDF này là [Ảnh chụp nhanh](#) một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2 để cập nhật mới nhất.

Trang 264

1.1 Kiểm tra kho lưu trữ hỗ trợ Android

1. Tải xuống dự án [TwoActivities](#) từ bài học trước về cách tạo và sử dụng Hoạt động.
2. Mở dự án trong Android Studio và chọn Tools > Trình quản lý SDK > Android. Ngăn Tùy chọn lỗi lõi Android SDK D xuất hiện.
3. Nhấp vào tab SDK Tools và mở rộng Support Repository.
4. Tìm Kho lưu trữ hỗ trợ Android trong danh sách.
Nếu **tôi nstalled** xuất hiện trong cột Trạng thái, bạn đã sẵn sàng. Nhấp vào C cancel .
Nếu **N ot đã cài đặt** hoặc **U pdate Có sẵn** một ppears, hãy nhấp vào hộp kiểm bên cạnh **Hỗ trợ một ndroid Kho lưu trữ**. Biểu tượng tải xuống sẽ xuất hiện bên cạnh hộp kiểm. Nhấp vào OK
5. Nhấp lại vào OK và sau đó F inish khi kho hỗ trợ đã được cài đặt.

1.2 Định cấu hình Espresso cho dự án

Khi bạn bắt đầu một dự án cho hệ số hình thức điện thoại và máy tính bảng bằng cách sử dụng **API 15: Android 4.0.3 (Ice Cream Sandwich)** làm SDK tối thiểu, Android Studio sẽ tự động bao gồm các phần phụ thuộc bạn cần để sử dụng Espresso. Để thực hiện các bài kiểm tra, Espresso và UI Automator sử dụng [JUnit](#) làm khung thử nghiệm của họ. JUnit là khung kiểm thử đơn vị phổ biến nhất và được sử dụng rộng rãi nhất cho Java. Lớp thử nghiệm của bạn sử dụng Espresso hoặc UI Automator phải được viết dưới dạng lớp kiểm tra JUnit 4.

Lưu ý: Bản sửa đổi JUnit mới nhất là JUnit 5. Tuy nhiên, với mục đích sử dụng Espresso hoặc UI Automator, phiên bản 4.12 được khuyến nghị.

Nếu đã tạo dự án của mình trong phiên bản Android Studio trước đó, bạn có thể phải tự thêm các phần phụ thuộc và thiết bị đo lường. Để tự thêm các phần phụ thuộc, hãy làm theo các bước sau:

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

1. Mở dự án TwoActivities hoặc nếu bạn muốn, trước tiên hãy tạo một bản sao của dự án và sau đó mở bản sao.
2. Mở **tệp build.gradle** (**Mô-đun: Ứng dụng**).
3. Kiểm tra xem những điều sau đây có được bao gồm (cùng với các phụ thuộc khác) trong phần dependencies hay không:

```
testImplementation 'junit:junit:4.12'  
androidTestImplementation 'com.android.support.test:runner:1.0.1'  
androidTestImplementation  
    'com.android.support.test.espresso:espresso:3.0.1'
```

Nếu tệp không bao gồm các câu lệnh phụ thuộc ở trên, hãy nhập chúng vào phần phụ thuộc.

4. Android Studio cũng thêm câu lệnh đo lường sau vào cuối phần defaultConfig của dự án mới:

```
kiểm tra công cụ chạy  
    "android.support.test.runner.AndroidJUnitRunner"
```

Nếu tệp không bao gồm câu lệnh đo lường ở trên, hãy nhập câu lệnh đó vào cuối phần defaultConfig.

[Tôi instrumentation](#) là một tập hợp các phương thức điều khiển, hoặc hook, trong hệ thống Android. Các hook này kiểm soát một thành phần Android độc lập với vòng đời bình thường của thành phần. Chúng cũng kiểm soát cách Android tải ứng dụng. Việc sử dụng thiết bị đo lường giúp các thử nghiệm có thể gọi các phương thức trong ứng dụng, sửa đổi và kiểm tra các trường trong ứng dụng, độc lập với vòng đời bình thường của ứng dụng.

5. Nếu bạn đã sửa đổi **tệp build.gradle (Module: app)**, hãy nhấp vào liên kết **Sync Now** trong thông báo về tệp Gradle ở góc trên cùng bên phải của cửa sổ.

creative
Commons để cập nhật
mới nhất.
Khóa học cơ bản về nhà phát triển Android (V2) **Đơn vị**

1.3 Tắt hoạt ảnh trên thiết bị thử nghiệm của bạn

Để cho phép Android Studio giao tiếp với thiết bị của bạn, trước tiên bạn phải bật tính năng Gõ lỗi USB trên thiết bị của mình, như mô tả trong bài học về cách cài đặt và chạy ứng dụng.

Điện thoại và máy tính bảng Android hiển thị hoạt ảnh khi di chuyển giữa các ứng dụng và màn hình. Các hình ảnh động hấp dẫn khi sử dụng thiết bị, nhưng chúng làm chậm hiệu suất và cũng có thể gây ra kết quả không mong muốn hoặc có thể khiến bài kiểm tra của bạn không thành công. Vì vậy, bạn nên tắt hoạt ảnh trên thiết bị vật lý của mình. Để tắt hoạt ảnh trên thiết bị thử nghiệm của bạn, hãy nhấn vào biểu tượng Cài đặt trên thiết bị vật lý của bạn. Tìm **Tùy chọn Developer**. Now tìm phần **Drawing**. Trong phần này, hãy tắt các tùy chọn sau:

- Tỷ lệ hoạt ảnh cửa sổ
- Tỷ lệ hoạt hình chuyển tiếp
- Thang thời lượng của Animator

Mẹo: Đo lường một hệ thống, ví dụ như thực hiện các bài kiểm tra đơn vị, có thể thay đổi thời gian của một số hàm. Vì lý do này, hãy tách biệt kiểm thử đơn vị và gỡ lỗi:

Kiểm thử đơn vị sử dụng khung Espresso dựa trên API với các hook để đo lường.

Quá trình gỡ lỗi sử dụng điểm ngắn và các phương thức khác trong các câu lệnh mã hóa trong mã của ứng dụng, như được mô tả trong bài học về gỡ lỗi.

Nhiệm vụ 2: Kiểm tra chuyển đổi hoạt động và nhập văn bản

Bạn viết các bài kiểm tra Espresso dựa trên những gì người dùng có thể làm khi tương tác với ứng dụng của bạn. Các bài kiểm tra Espresso là các lớp tách biệt với mã ứng dụng của bạn. Bạn có thể tạo bao nhiêu thử nghiệm tùy thích để tương tác với các phần tử View trong giao diện người dùng mà bạn muốn kiểm tra.

Bài kiểm tra Espresso giống như một robot phải được cho biết phải làm gì. Nó phải tìm thấy chữ V mà bạn muốn nó tìm thấy trên màn hình và nó phải kết hợp với nó, chẳng hạn như nhấp vào chữ V và kiểm tra nội dung của chữ V. Nếu nó không thực hiện đúng bất kỳ điều nào trong số này hoặc nếu kết quả không như bạn mong đợi, bài kiểm tra sẽ không thành công.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

để cập nhật mới nhất.

Với Espresso, bạn tạo ra những gì về cơ bản là một tập lệnh hành động để thực hiện trên mỗi View và kiểm tra các kết quả mong đợi. Các khái niệm chính là Locating và sau đó kết hợp với các phần tử giao diện người dùng. Đây là các bước cơ bản:

1. **Phù hợp với chế độ xem:** Find a View.
2. **Thực hiện một hành động:** Perform một nhấp chuột hoặc hành động khác kích hoạt một sự kiện với View.
3. **Xác nhận và xác minh kết quả:** Kiểm tra trạng thái của View để xem nó có phản ánh trạng thái hoặc hành vi dự kiến được xác định bởi xác nhận hay không.

Hamcrest (đảo chữ của "matchers") là một framework hỗ trợ viết các bài kiểm tra phần mềm trong Java. Để tạo thử nghiệm, bạn tạo một phương thức trong lớp kiểm thử sử dụng biểu thức Hamcrest.

Mẹo: Để biết thêm thông tin về các matcher Hamcrest, hãy xem [Hướng dẫn Hamcrest](#).

Với Espresso, bạn sử dụng các loại biểu thức Hamcrest sau đây để giúp tìm các phần tử View và tương tác với chúng:

- **ViewMatchers:** Biểu thức đối sánh Hamcrest trong lớp [ViewMatchers](#) cho phép bạn tìm thấy Chế độ xem trong hệ thống phân cấp View hiện tại để bạn có thể kiểm tra một cái gì đó hoặc thực hiện một số hành động.
- **ViewActions:** Biểu thức hành động Hamcrest trong lớp [ViewActions](#) cho phép bạn thực hiện một hành động trên View được tìm thấy bởi ViewMatcher.

- **ViewAssertions:** Biểu thức xác nhận Hamcrest trong lớp [ViewAssertions](#) cho phép bạn xác nhận hoặc kiểm tra trạng thái của View được tìm thấy bởi ViewMatcher.

Sau đây cho thấy cách cả ba biểu thức hoạt động cùng nhau:

1. Sử dụng ViewMatcher để tìm View: onView(withId(R.id.my_view))
2. Sử dụng ViewAction để thực hiện một hành động: .perform(click())
3. Sử dụng ViewAssertion để kiểm tra xem kết quả của hành động có khớp với một xác nhận hay không: .check(matches(isDisplayed()))

Sau đây cho thấy cách các biểu thức trên được sử dụng cùng nhau trong một câu lệnh:

creative
Commons để cập nhật
mới nhất.

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

```
onView(withId(R.id.my_view))
    .perform(click())
    .check(matches(isDisplayed()));
```

2.1 Chạy kiểm tra ví dụ

Android Studio tạo một lớp thử nghiệm Espresso trống khi bạn tạo dự án.

- Trong ngăn Project > Android, hãy mở Java > com.example.android.twoactivities (androidTest) và mở ExampleInstrumentedTest.

Dự án được cung cấp với thử nghiệm ví dụ này. Bạn có thể tạo bao nhiêu thử nghiệm tùy thích trong thư mục này. Trong bước tiếp theo, bạn sẽ chỉnh sửa thử nghiệm mẫu.

- Trong ngăn Project > Android, hãy mở Java > com.example.android.twoactivities (androidTest) và mở ExampleInstrumentedTest.
- Để chạy thử nghiệm, hãy nhấp vào (hoặc nhấp vào C) ExampleInstrumentedTest và chọn Run ExampleInstrumentedTest từ menu bật lên. Sau đó, bạn có thể chọn chạy thử nghiệm trên trình mô phỏng hoặc trên thiết bị của mình.

Ngăn Run ở cuối Android Studio hiển thị tiến trình kiểm tra và khi kết thúc, nó sẽ hiển thị "Tests run to completed". Ở cột bên trái, Android Studio hiển thị "Tất cả các bài kiểm tra đã đạt".

2.2 Xác định một lớp cho một thử nghiệm và thiết lập Activity

Bạn sẽ chỉnh sửa thử nghiệm ví dụ thay vì thêm một thử nghiệm mới. Để làm cho bài kiểm tra ví dụ dễ hiểu hơn, bạn sẽ đổi tên lớp từ ExampleInstrumentedTest thành

ActivityInputOutputTest (bằng tiếng Anh). Làm theo các bước sau:

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Tác phẩm này được cấp phép theo [Creative Commons Attribution 4.0 Giấy phép quốc tế](#).

Trang 269

1. Nhấp chuột phải (hoặc Ctrl nhấp vào) ExampleInstrumentedTest trong ngăn Project > Android và chọn Refactor > Đổi tên.
2. Thay đổi tên lớp thành ActivityInputOutputTest và để tất cả các tùy chọn được chọn. Nhấp vào Refactor.
3. Thêm thông tin sau vào đầu lớp ActivityInputOutputTest, trước chú thích @ Test đầu tiên :

```
@Rule  
public ActivityTestRule mActivityRule = ActivityTestRule mới<>(  
    MainActivity.lớp); }
```

Định nghĩa lớp bây giờ bao gồm một số chú thích:

- @RunWith: Để tạo lớp kiểm tra JUnit 4 được đo lường, hãy thêm
@RunWith(AndroidJUnit4.class) ở đầu định nghĩa lớp thử nghiệm của bạn.
- @Rule: Chú thích Quy tắc @ cho phép bạn thêm hoặc xác định lại hành vi của từng phương thức kiểm thử theo cách có thể tái sử dụng, sử dụng một trong các lớp quy tắc kiểm tra mà Thư viện hỗ trợ thử nghiệm Android cung cấp, chẳng hạn như [ActivityTestRule](#) hoặc [ServiceTestRule](#). Quy tắc trên sử dụng đối tượng ActivityTestRule, cung cấp kiểm tra chức năng của một Activity duy nhất—trong trường hợp này là MainActivity.class. Trong suốt thời gian thử nghiệm, bạn sẽ có thể thao tác trực tiếp với Activity của mình, sử dụng ViewMatchers, ViewActions và ViewAssertions.
- @Test: Chú thích @ Test cho JUnit biết rằng phương thức public void mà nó được đính kèm có thể được chạy như một trường hợp thử nghiệm. Phương thức kiểm thử bắt đầu bằng chú thích @ Test và chứa mã để thực hiện và xác minh một hàm duy nhất trong thành phần mà bạn muốn kiểm tra.

Trong câu lệnh trên, ActivityTestRule có thể chuyển sang màu đỏ lúc đầu, nhưng sau đó Android Studio tự động thêm câu lệnh import sau:

a

for the latest updates.

```
nhập android.support.test.rule.ActivityTestRule;
```

creativecommons.org/licenses/by/nd/4.0/deed.vi

Trang 270

Khóa học cơ bản về nhà phát triển Android (V2) **Đơn vị**

2.3 Kiểm tra chuyển đổi từ Hoạt động này sang Hoạt động khác

Ứng dụng TwoActivities bao gồm:

- Hoạt động chính: Bao gồm nút button_main để chuyển sang SecondActivity và chế độ xem text_header_reply đóng vai trò là tiêu đề văn bản.
- Hoạt động thứ hai: Bao gồm nút button_second để chuyển sang MainActivity và chế độ xem text_header đóng vai trò là tiêu đề văn bản.

Khi bạn có một ứng dụng chuyển từ Activity này sang một Activity khác, bạn nên kiểm tra khả năng đó.

Ứng dụng TwoActivities cung cấp trường nhập văn bản và nút **kết thúc S** (id button_main). Nhấp vào **Send** khởi chạy SecondActivity với văn bản đã nhập được hiển thị trong chế độ xem text_header của Hoạt động thứ hai.

Nhưng điều gì xảy ra nếu không có văn bản nào được nhập? SecondActivity sẽ vẫn xuất hiện?

Lớp kiểm tra ActivityInputOutputTest sẽ cho thấy rằng các phần tử View trong SecondActivity xuất hiện bất kể văn bản có được nhập hay không.

Lưu ý: Khi bạn nhập ViewMatchers và ViewActions, dẽ không sao chép chúng khỏi văn bản và dán chúng vào cửa sổ chỉnh sửa Android Studio. Thay vào đó, hãy nhập trực tiếp các biểu thức để Android Studio nhận đúng ViewMatchers và ViewActions thích hợp.

Làm theo các bước sau để thêm các thử nghiệm của bạn vào ActivityInputOutputTest:

1. Thêm phần đầu của phương thức activityLaunch() vào ActivityInputOutputTest. Phương pháp này sẽ kiểm tra xem các phần tử SecondActivityView có xuất hiện khi nhấp vào nút hay không và bao gồm ký hiệu @ Kiểm tra trên một dòng ngay phía trước phương thức:

```
@Test  
công khai void activityLaunch() {
```

Tác phẩm này được cấp phép theo Giấy phép Quốc tế Creative Commons Attribution 4.0.
PDF này là Ảnh chụp nhanh một lần. Xem developer.android.com/courses/fundamentals-training/toc-v2

Trang 271

2. Thêm biểu thức ViewMatcher và ViewAction kết hợp vào phương thức ctivityLaunch() để xác định vị trí View chứa button_main Button và bao gồm biểu thức ViewAction để thực hiện nhấp chuột.

```
onView(withId(R.id.button_main)).perform(click());
```

The onView() method lets you use ViewMatcher arguments to find View elements. It searches the View hierarchy to locate a corresponding View instance that meets some given criteria—in this case, the button_main View. The .perform(click()) expression is a ViewAction expression that performs a click on the View.

In the above onView statement, onView, withID, and click may turn red at first, but then Android Studio adds import statements for them.

3. Add another ViewMatcher expression to the activityLaunch() method to find the text_header View (which is in SecondActivity), and a ViewAction expression to perform a check to see if the View is displayed:

```
onView(withId(R.id.text_header)).check(matches(isDisplayed()));
```

This statement uses the onView() method to locate the text_header View for SecondActivity and then check to see if it is displayed after clicking the button_main Button. The check() method may turn red at first, but then Android Studio adds an import statement for it.

4. Add similar statements to test whether clicking the button_second Button in SecondActivity switches to MainActivity:

```
onView(withId(R.id.button_second)).perform(click());
onView(withId(R.id.text_header_reply)).check(matches(isDisplayed()));
```

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 Android
Developer Fundamentals Course (V2) Unit

5. Review the `activityLaunch()` method you just created in the `ActivityInputOutputTest` class. It should look like this:

```
@Test public void
activityLaunch() {
    onView(withId(R.id.button_main)).perform(click());
    onView(withId(R.id.text_header)).check(matches(isDisplayed()));
    onView(withId(R.id.button_second)).perform(click());
    onView(withId(R.id.text_header_reply)).check(matches(isDisplayed()));
}
```

6. To run the test, right-click (or Control-click) `ActivityInputOutputTest` and choose **Run** from the pop-up menu. You can then choose to run the test on the emulator or on your device.

As the test runs, watch the test automatically start the app and click the `Button`. The `SecondActivity` View elements appear. The test then clicks the `Button` in `SecondActivity`, and `MainActivity` View elements appear.

The Run window (the bottom pane of Android Studio) shows the progress of the test, and when finishes, it displays "Tests ran to completion." In the left column Android Studio displays "All Tests Passed".

The drop-down menu next to the **Run** icon  in the Android Studio toolbar now shows the name of the test class (`ActivityInputOutputTest`). You can click the **Run** icon to run the test, or switch to **app** in the dropdown menu and then click the **Run** icon to run the app.

2.4 Test text input and output

In this step you will write a test for text input and output. The TwoActivities app uses the `editText_main` `EditText` for input, the `button_main` `Button` for sending the input to `SecondActivity`, and the `TextView` in `SecondActivity` that shows the output in the field with the id `text_message`.

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
 This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 Android Developer Fundamentals Course (V2) Unit*

1. Add another `@Test` annotation and a new `textInputOutput()` method to test text input and output:

```
@Test public void
textInputOutput() {
    onView(withId(R.id.editText_main)).perform(typeText("This is a test."));
    onView(withId(R.id.button_main)).perform(click()); }
```

The above method uses a `ViewMatcher` to locate the `View` containing the `editText_main` `EditText`, and a `ViewAction` to enter the text "This is a test." It then uses another `ViewMatcher` to find the `View` with the `button_main` `Button`, and another `ViewAction` to click the `Button`.

2. Add a `ViewMatcher` to the method to locate the `text_message` `TextView` in `SecondActivity`, and a `ViewAssertion` to see if the output matches the input to test that the message was correctly sent—it should match "This is a test." (Be sure to include the period at the end.)

```
onView(withId(R.id.text_message))
    .check(matches(withText("This is a test.")));
```

a

for the latest updates.

3. Run the test.

As the test runs, the app starts and the text is automatically entered as input; the `B`utton is clicked, and `SecondActivity` appears with the text.

The bottom pane of Android Studio shows the progress of the test, and when finished, it displays “Tests ran to completion.” In the left column Android Studio displays “All Tests Passed”. You have successfully tested the `EditText` input, the `Send` `Button`, and the `TextView` output.

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

2.5 Introduce an error to show a test failing

Introduce an error in the test to see what a failed test looks like.

1. Change the matches check on the text_message view from "This is a test." to "This is a failing test.":

```
onView(withId(R.id.text_message)).check(matches(withText("This is a failing test.")));
```

2. Run the test again. This time you will see the message in red, "1 test failed", above the bottom pane, and a red exclamation point next to textInputOutput in the left column. Scroll the bottom pane to the message "Test running started" and see that all of the results after that point are in red. The very next statement after "Test running started" is:

```
android.support.test.espresso.base.DefaultFailureHandler$AssertionFailedWithCauseError: 'with text: is "This is a failing test."' doesn't match the selected view.  
Expected: with text: is "This is a failing test."  
Got: "AppCompatTextView{id=2131165307, resname=text_message ...
```

Other fatal error messages appear after the above, due to the cascading effect of a failure leading to other failures. You can safely ignore them and fix the test itself.

Task 2 solution code

See ActivityInputOutputTest.java in the Android Studio project: [TwoActivitiesEspresso](#)

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

Page 275

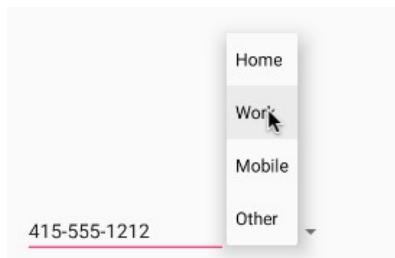
Task 3: Test the display of spinner selections

The Espresso onView() method finds a View that you can test. This method will find a View in the current View hierarchy. But you need to be careful—in an AdapterView such as a Spinner, the View is typically dynamically populated with child View elements at runtime. This means there is a possibility that the View you want to test may not be in the View hierarchy at that time.

The Espresso API handles this problem by providing a separate onData() entry point, which is able to first load the adapter item and bring it into focus prior to locating and performing actions on any of its children.

PhoneNumberSpinner is a starter app you can use to test a Spinner. It shows a Spinner, with the id label_spinner, for choosing the label of a phone number (Home, Work, Mobile, and Other). It then displays the phone number and Spinner choice in a TextView.

The goal of this test is to open the Spinner, click each item, and then verify that the TextView text_phonelabel contains the item. The test demonstrates that the code retrieving the Spinner selection is working properly, and the code displaying the text of the Spinner item is also working properly. You will write the test using string resources and iterate through the Spinner items so that the test works no matter how many items are in the Spinner, or how those items are worded; for example, the words could be in a different language.



This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

Page 276

3.1 Create the test method

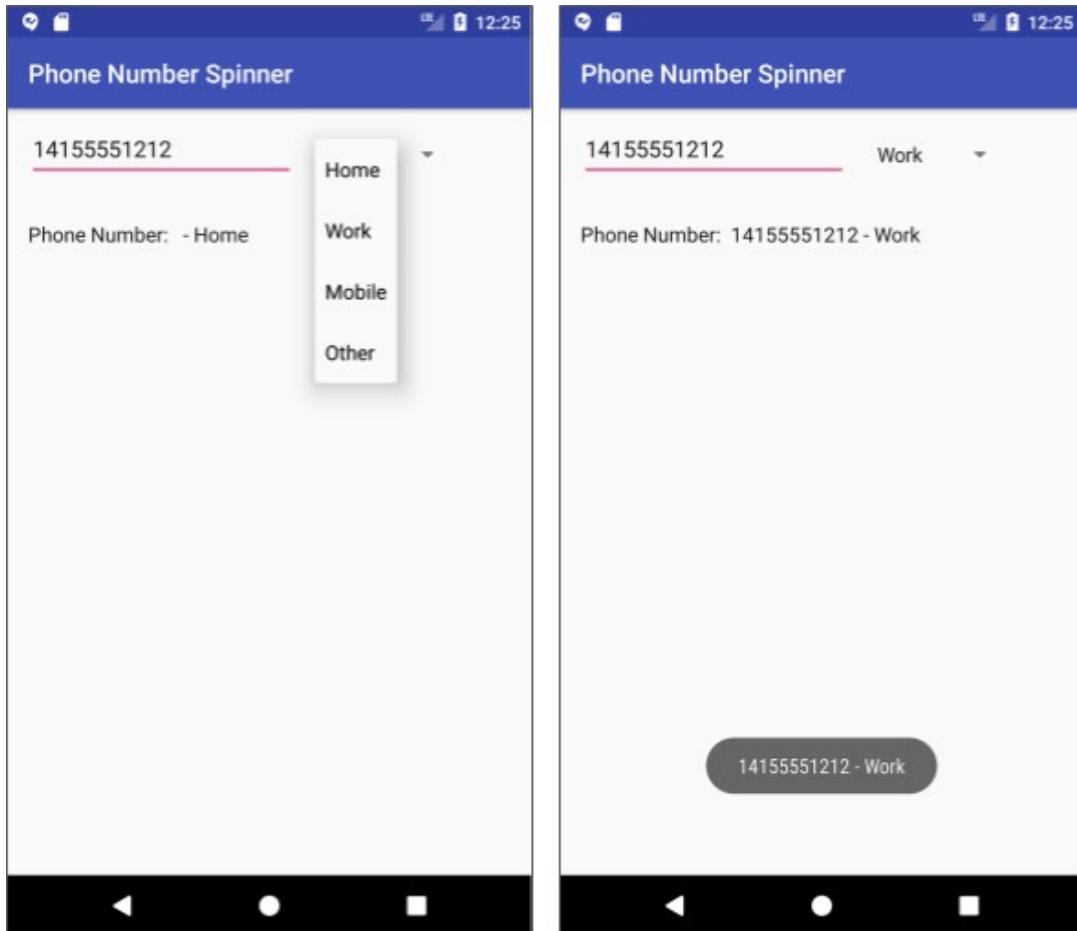
1. Download the [PhoneNumberSpinnerEspresso](#) project and then open the project in Android Studio.

creative

Commons for the latest
updates.

Android Developer Fundamentals Course (V2) Unit

2. Run the app. Enter a phone number, and choose a label from the Spinner as shown on the left side of the figure below. The result should appear in the TextView and in a Toast message, as shown on the right side of the figure.



3. Expand `com.example.android.phonenumberspinner (androidTest)`, and open `ExampleInstrumentedTest`.
4. Rename `ExampleInstrumentedTest` to `SpinnerSelectionTest` in the class definition, and add the following:

```
@RunWith(AndroidJUnit4.class) public class  
SpinnerSelectionTest {  
    @Rule
```

This work is licensed under a [Creative Commons Attribution 4.0 International License](#). This PDF is one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

This work is licensed under a [Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

```
public ActivityTestRule mActivityRule = new ActivityTestRule<>(
    MainActivity.class);
```

3.2 Access the array used for the Spinner items

You want the test to click each item in the Spinner based on the number of elements in the array. But how do you access the array?

1. Create the `iterateSpinnerItems()` method as public returning void, and assign the array used for the Spinner items to a new array to use within the `iterateSpinnerItems()` method:

```
@Test
public void iterateSpinnerItems() {
    String[] myArray =
        mActivityRule.getActivity().getResources()
            .getStringArray(R.array.labels_array); }
```

In the statement above, the test accesses the array (with the id `labels_array`) by establishing the context with the `getActivity()` method of the `ActivityTestRule` class, and getting a resources instance using `getResources()`.

2. Assign the length of the array to `size`, and start the beginning of a for loop using the `size` as the maximum number for a counter.

```
int size = myArray.length;
for (int i=0; i<size; i++) {
```

reative
Commons for the latest
updates.

3.3 Locate Spinner items and click on them

1. Add an `onView()` statement within the `for` loop to find the `Spinner` and click on it:

```
// Find the spinner and click on it.  
onView(withId(R.id.label_spinner)).perform(click());
```

A user must click the `Spinner` itself in order to click any item in the `Spinner`, so your test must click the `Spinner` first before clicking the item.

2. Write an `onData()` statement to find and click a `Spinner` item:

```
// Find the spinner item and click on it.  
onData(is(myArray[i])).perform(click());
```

The above statement matches if the object is a specific item in the `Spinner`, as specified by the `myArray[i]` array element.

If `onData` appears in red, click the word, and then click the red light bulb icon that appears in the left margin. Choose the following in the pop-up menu: **S static import method**
`'android.support.test.espresso.Espresso.onData'`

If `is` appears in red, click the word, and then click the red light bulb icon that appears in the left margin. Choose the following in the pop-up menu: **S static import method...> Matchers.is (org.hamcrest)**

1. Add another `onView()` statement to the `for` loop to check to see if the resulting `text_phonelabel` matches the `Spinner` item specified by `myArray[i]`.

```
// Find the text view and check that the spinner item
```

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

This work is licensed under

[Creative Commons Attribution 4.0 International License](#),

for the latest updates.

```
// is part of the string.  
onView(withId(R.id.text_phonelabel))  
    .check(matches(withText(containsString(myArray[i]))));
```

If `c containsString` appears in red, click the word, and then click the red light bulb icon that appears in the left margin. Choose the following in the pop-up menu: **S static import method...> Matchers.containsString (org.hamcrest)**

2. Run the test.

The test runs the app, clicks the S pinner, and “exercises” the S pinner—it clicks each S pinner item from top to bottom, checking to see if the item appears in the `text_phonelabel` `TextView`. It doesn’t matter how many S pinner items are defined in the array, or what language is used for the S pinner items—the test performs all of them and checks their output against the array.

The bottom pane of Android Studio shows the progress of the test, and when finished, it displays “Tests ran to completion.” In the left column Android Studio displays “All Tests Passed”.

Task 3 solution code

See `S pinnerSelectionTest.java` in the Android Studio project: [P honeNumberSpinnerEspresso](#)

Task 4: Record an Espresso test

Android Studio lets you record an Espresso test, which is useful for generating tests quickly. While recording a test, you use your app as a normal user—as you click through the app UI, editable test code is generated for you. You can also add assertions to check if a View holds a certain value.

This work is licensed under a C

[Attribution 4.0 International License](#).

This PDF is a one-time snapshot. See [d eveloper.android.com/courses/fundamentals-training/toc-v2](#)

Recording Espresso tests, rather than coding the tests by hand, ensures that your app gets UI test coverage on areas that might take too much time or be too difficult to code by hand.

To demonstrate test recording, you will record a test of the [Scorekeeper](#) app created in the practical on using drawables, styles, and themes.

*creative
Commons for the latest
updates.*

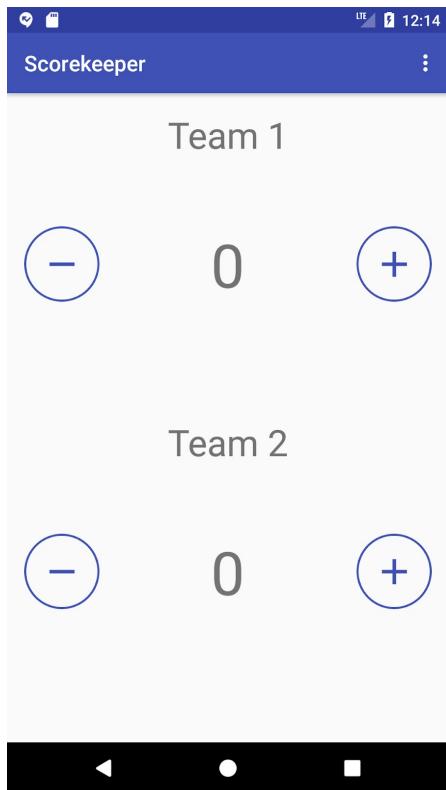
4.1 Open and run the app

1. Download the [Scorekeeper](#) project that you created in [Android fundamentals 5.1: Drawables, styles, and themes.](#)
2. Open the Scorekeeper project in Android Studio.
3. Run the app to ensure that it runs properly.

The Scorekeeper app consists of two sets of Button elements and two TextView elements, which are used to keep track of the score for any point-based game with two players.

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2



This work is licensed under

Creative Commons Attribution 4.0 International License.

for the latest updates.

This work is licensed under a C

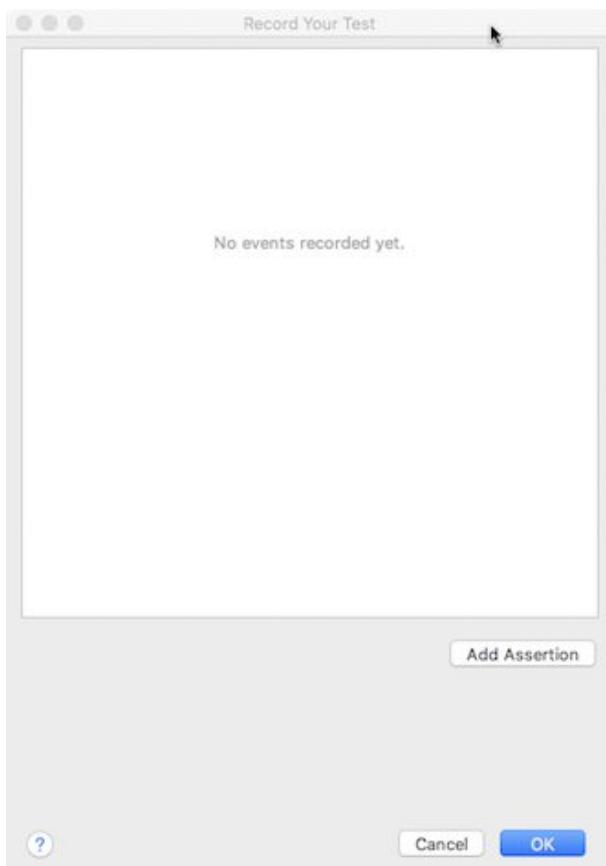
Attribution 4.0 International License.

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

4.2 Record the test

1. Choose **Run > Record Espresso Test**, select your deployment target (an emulator or a device), and click **OK**.

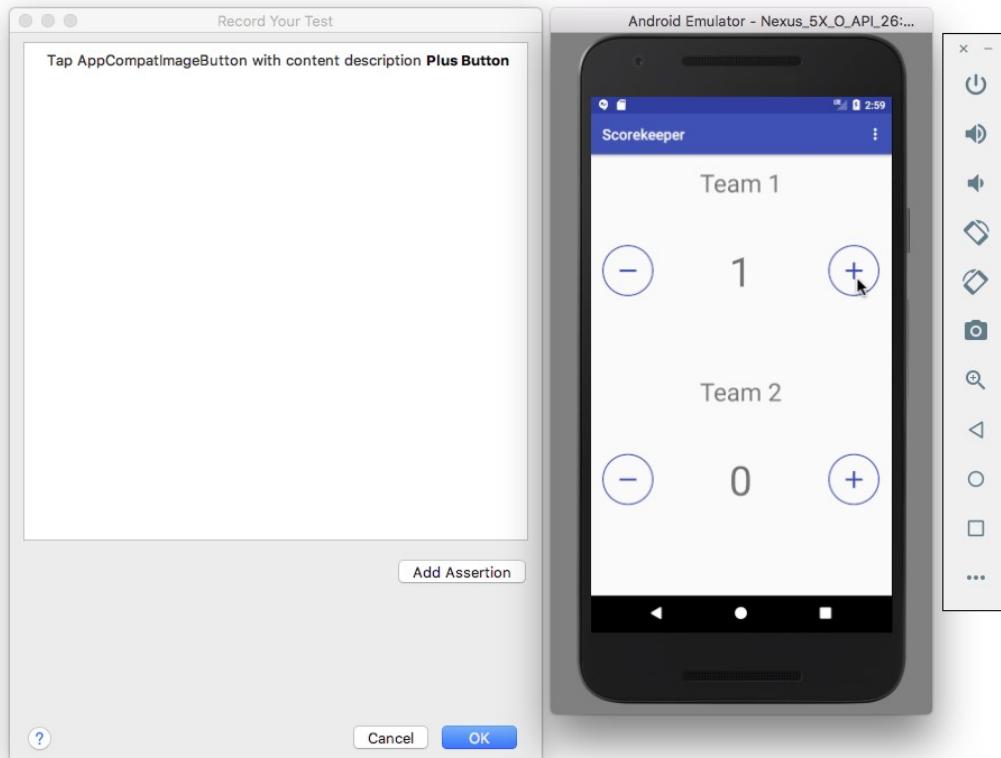
The Record Your Test dialog appears, and the Debugger pane appears at the bottom of the Android Studio window. If you are using an emulator, the emulator also appears.



2. On the emulator or device, tap the plus (+) ImageButton for Team 1 in the app. The Record Your Test window shows the action that was recorded ("Tap AppCompatImageButton with the content description Plus Button").

This work is licensed under a Creative Commons Attribution 4.0 International License. This PDF is one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Page 282

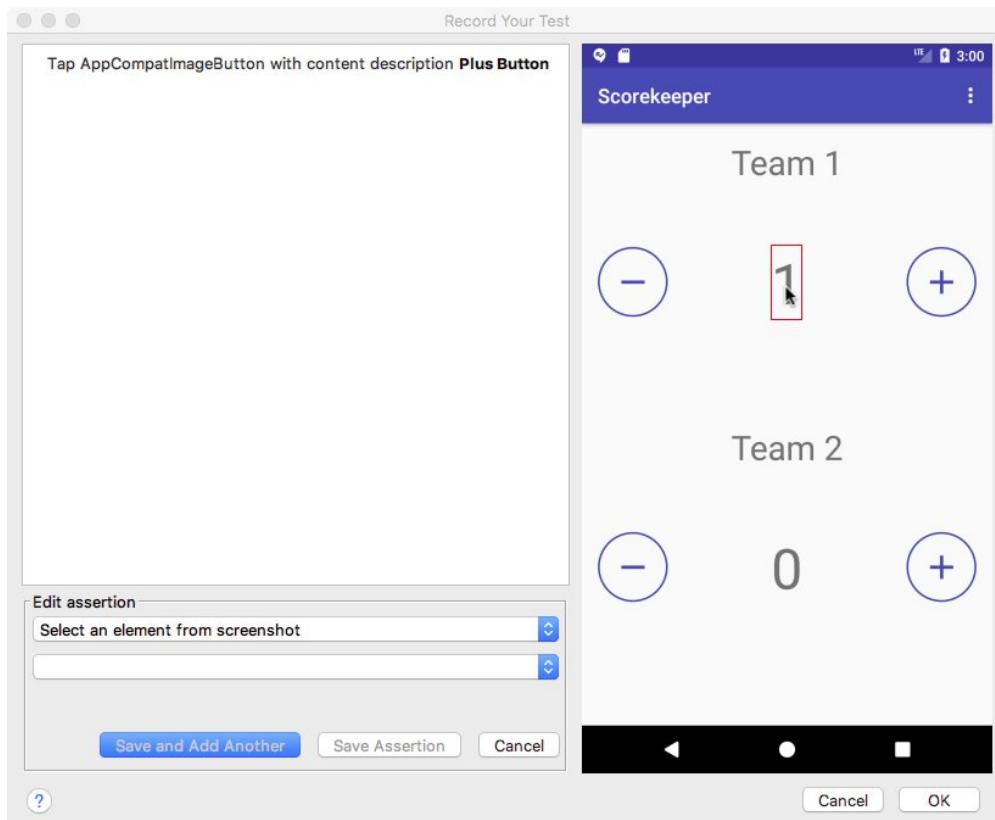


1. Click **Add Assertion** in the Record Your Test window. A screenshot of the app's UI appears in a pane on the right side of the window, and the **Select an element from screenshot** option appears in the dropdown menu. Select the score (1) in the screenshot as the UI element you want to check, as shown in the figure below.

a

for the latest updates.

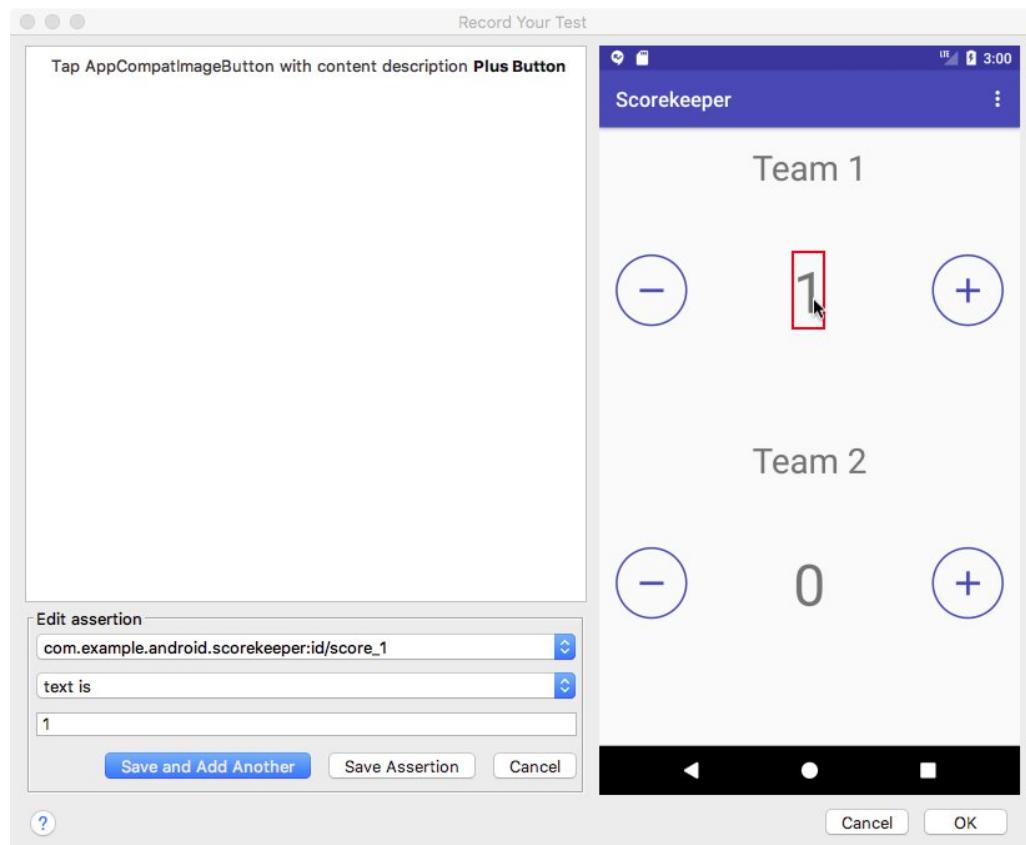
This work is licensed under [Creative Commons Attribution 4.0 International License](#),
Android Developer Fundamentals Course (V2) Unit



2. Choose **t ext** is from the second dropdown menu, as shown in the figure below. The text you expect to see (1) is already entered in the field below the dropdown menu.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

Page 284



3. Click **S**ave **A**ssertion.
4. To record another click, on your emulator or device tap the minus (-) ImageButton for Team 1 in the app. The Record Your Test window shows the action that was recorded ("Tap AppCompatImageButton with the content description **M**inus **B**utton").

a

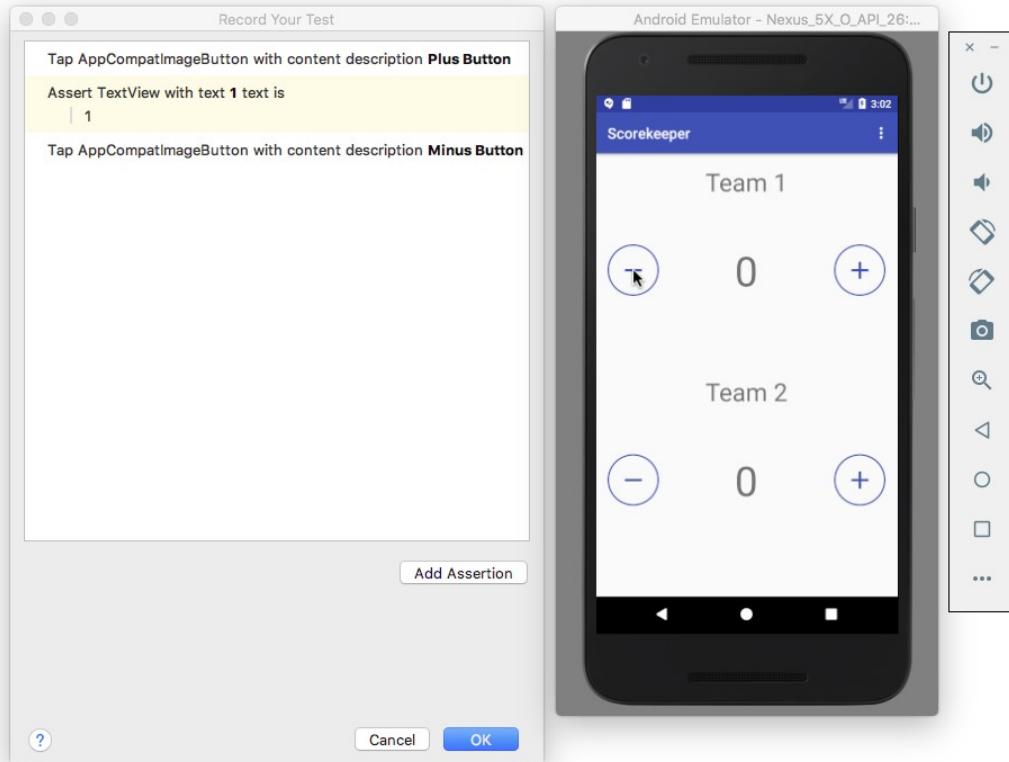
for the latest updates.

This work is licensed under

Creative Commons Attribution 4.0 International License.

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.



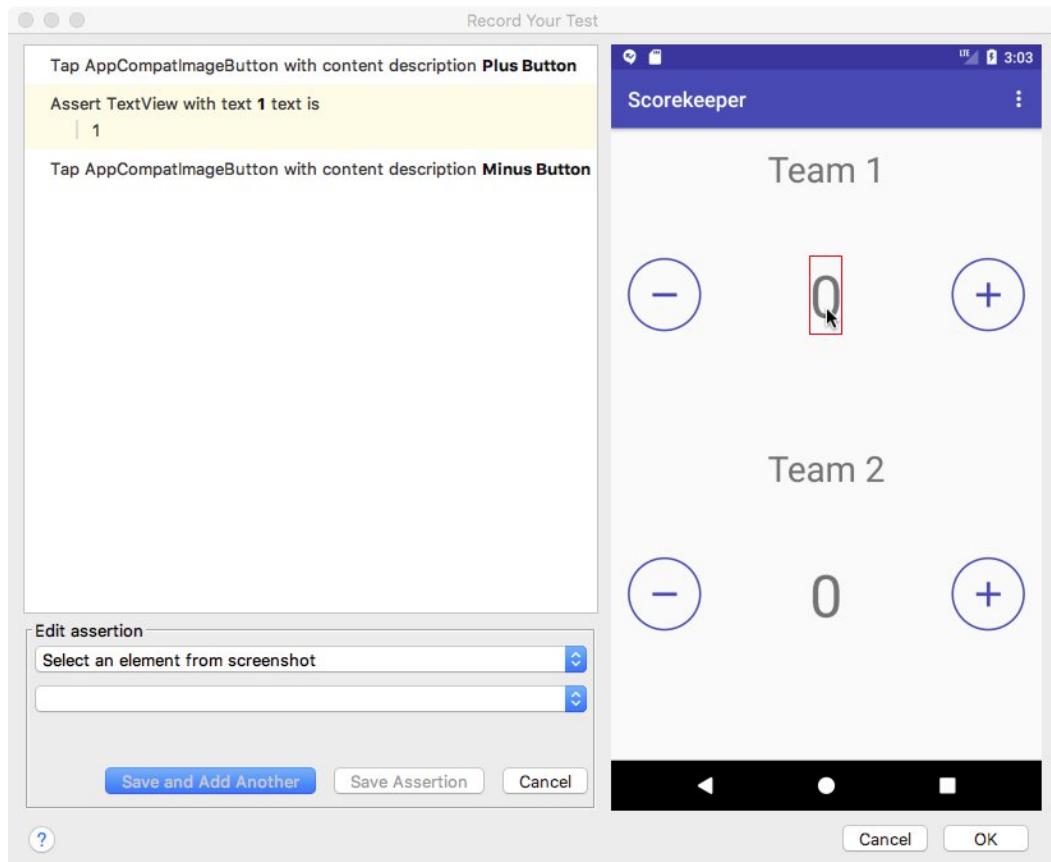
- Click **Add Assertion** in the Record Your Test window. The app's UI appears in the right-side pane as before. Select the score (0) in the screenshot as the UI element you want to check.

a

for the latest updates.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

Page 286

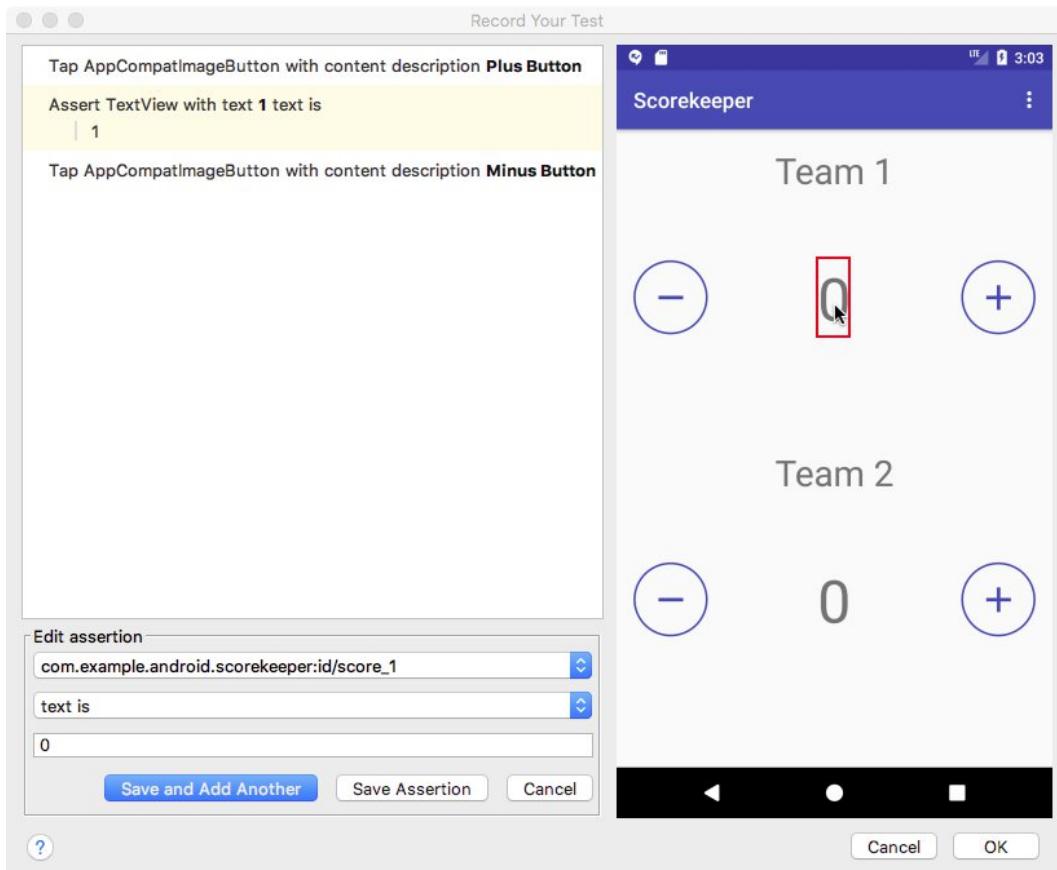


6. Choose **t ext is** from the second dropdown menu, as shown in the figure below. The text you expect to see (0) is already entered in the field below the dropdown menu.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Page 291

eat ve ommons Attr
Android Developer Fundamentals Course (V2) Unit



7. Click **S**ave **A**ssertion, and then click **O**K.
8. In the dialog that appears, edit the name of the test to **S corePlusMinusTest** so that it is easy for others to understand the purpose of the test.
9. Android Studio may display a request to add more dependencies to your Gradle Build file. Click **Y**es to add the dependencies. Android Studio adds the following to the dependencies section of the build.gradle (Module: app) file:

a

for the latest updates.

```
androidTestCompile  
        'com.android.support.test.espresso:espressocontrib:2.2.2', {  
            exclude group: 'com.android.support', module: 'supportannotations'  
            exclude group: 'com.android.support', module: 'supportv4'
```

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2*

Page 288

*This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.*

Page 293

```
exclude group: 'com.android.support', module: 'design'  
exclude group: 'com.android.support', module: 'recyclerviewv7' }
```

10. Expand **com.example.android.scorekeeper (androidTest)** to see the test, and run the test. It should pass. Run it again, and it should pass again.

The following is the test, as recorded in the `ScorePlusMinusTest.java` file:

```
// ... Package and import statements  
@LargeTest  
@RunWith(AndroidJUnit4.class) public  
class ScorePlusMinusTest {  
  
    @Rule  
    public ActivityTestRule<MainActivity> mActivityTestRule  
        = new ActivityTestRule<>(MainActivity.class);  
  
    @Test  
    public void scorePlusMinusTest() {  
        ViewInteraction appCompatImageButton = onView(  
            allOf(withId(R.id.increaseTeam1),  
                  withContentDescription("Plus Button"),  
                  childAtPosition(  
                      childAtPosition(  
                          withClassName(is("android.widget.LinearLayout")),  
                          0),  
                      3),  
                  isDisplayed()));  
        appCompatImageButton.perform(click());  
        ViewInteraction textView = onView(  
            allOf(withId(R.id.score_1), withText("1"),  
                  childAtPosition(  
                      childAtPosition(IsInstanceOf.  
                        .<View>instanceOf(android.widget.LinearLayout.class),  
                      0),  
                  2),  
                  isDisplayed()));  
        textView.check(matches(withText("1")));  
        ViewInteraction appCompatImageButton2 = onView(  
            allOf(withId(R.id.decreaseTeam1),  
                  withContentDescription("Minus Button"),
```

eat ve ommons Attr
for the latest updates.

```
        childAtPosition(
            childAtPosition(
                withClassName(is("android.widget.LinearLayout")),
                    0),
                1),
            isDisplayed())));
appCompatImageButton2.perform(click());

ViewInteraction textView2 = onView(
    allOf(withId(R.id.score_1), withText("0"),
        childAtPosition(
            childAtPosition(IsInstanceOf
                .instanceOf(android.widget.LinearLayout.class),
                    0),
            2),
            isDisplayed())));
textView2.check(matches(withText("0")));
}

private static Matcher<View> childAtPosition(
    final Matcher<View> parentMatcher, final int position) {
return new TypeSafeMatcher<View>() {
    @Override
    public void describeTo(Description description) {
        description.appendText("Child at position "
            + position + " in parent ");
        parentMatcher.describeTo(description);
    }

    @Override
    public boolean matchesSafely(View view) {
        ViewParent parent = view.getParent();
return parent instanceof ViewGroup
        && parentMatcher.matches(parent)
        && view.equals(((ViewGroup) parent)
            .getChildAt(position));
    }
};
```

This work is licensed under Creative Commons Attribution 4.0 International License.
Android Developer Fundamentals Course (V2) Unit

The test uses the [ViewInteraction](#) class, which is the primary interface for performing actions or assertions on View elements, providing both check() and perform() methods. Examine the test code to see how it works:

- **Perform:** The code below uses a method called childAtPosition(), which is defined as a custom Matcher, and the perform() method to click an ImageButton:

```
ViewInteraction appCompatImageButton = onView(
    allOf(withId(R.id.increaseTeam1),
          withContentDescription("Plus Button"),
          childAtPosition(
              childAtPosition(
                  withClassName(is("android.widget.LinearLayout")),
                  0),
              3),
          isDisplayed()));
appCompatImageButton.perform(click());
```

- **Check whether it matches the assertion:** The code below also uses the childAtPosition() custom Matcher, and checks to see if the clicked item matches the assertion that it should be "1":

```
ViewInteraction textView = onView(
    allOf(withId(R.id.score_1), withText("1"),
          childAtPosition(
              childAtPosition(
                  childAtPosition(
                      IsInstanceOf<View>.instanceOf(android.widget.LinearLayout.class),
                      0),
                  2),
          isDisplayed()));
textView.check(matches(withText("1")));
```

- Custom Matcher: The code above uses `childAtPosition()`, which is defined as a custom Matcher:

eat ve
ommons Attr for the
latest updates.

```
private static Matcher<View> childAtPosition(
    final Matcher<View> parentMatcher, final int position) {
    return new TypeSafeMatcher<View>() {
        @Override
        public void describeTo(Description description) {
            description.appendText("Child at position "
                + position + " in parent ");
            parentMatcher.describeTo(description);
        }

        @Override
        public boolean matchesSafely(View view) {
            ViewParent parent = view.getParent();
            return parent instanceof ViewGroup
                && parentMatcher.matches(parent)
                && view.equals(((ViewGroup) parent)
                    .getChildAt(position));
        }
    };
}
```

The custom Matcher in the above code extends the abstract [TypeSafeMatcher](#) class.

You can record multiple interactions with the UI in one recording session. You can also record multiple tests, and edit the tests to perform more actions, using the recorded code as a snippet to copy, paste, and edit.

Task 4 solution code

See `S corePlusMinusTest.java` in the Android Studio project: [S_corekeeperEspresso](#)

This work is licensed under

Creative Commons Attribution 4.0 International License.

a

This PDF is a one-time snapshot. See [developer.android.com/courses/fundamentals-training/toc-v2](#) for the latest updates.

Coding challenge

Note: All coding challenges are optional and are not prerequisites for later lessons.

You learned how to create a [RecyclerView](#) in another practical. The app lets you scroll a list of words from "Word 1" to "Word 19". When you tap the FloatingActionButton, a new word appears in the list ("+ Word 20").

Like an AdapterView (such as a Spinner), a RecyclerView dynamically populates child View elements at runtime. But a RecyclerView is not an AdapterView, so you can't use findViewById() to interact with list items as you did in the previous task with a Spinner. What makes a RecyclerView complicated from the point of view of Espresso is that findViewById() can't find the child View if it is off the screen.

Challenge: Fortunately, you can record an Espresso test of using the RecyclerView. Record a test that taps the FloatingActionButton, and check to see if a new word appears in the list ("+ Word 20").

Tip: To add another test that clicks a RecyclerView item, don't record the test. Instead, use a class called [RecyclerViewActions](#), which exposes a small API that you can use to operate on a RecyclerView.

Challenge solution code

See RecyclerViewTest.java in the Android Studio project: [RecyclerViewEspresso](#)

Summary

To set up Espresso to test an Android Studio project:

- In Android Studio, check for and install the Android Support Repository.

-

reative
Commons for the latest
updates.

Add dependencies to the **build.gradle (Module: app)** file:

```
testImplementation 'junit:junit:4.12'  
androidTestImplementation 'com.android.support.test:runner:1.0.1'  
androidTestImplementation  
    'com.android.support.test.espresso:espresso:3.0.1'
```

- Add the following instrumentation statement to the end of the `defaultConfig` section:

```
testInstrumentationRunner  
    "android.support.test.runner.AndroidJUnitRunner"
```

Instrumentation is a set of control methods, or hooks, in the Android system.

- On your test device, turn off animations. To do this, turn on USB Debugging. Then in the Settings app, select **D eveloper Options > Drawing**. Turn off window animation scale, transition animation scale, and animator duration scale.

To test annotations:

- `@RunWith(AndroidJUnit4.class)`: Create an instrumented JUnit 4 test class.
- `@Rule`: Add or redefine the behavior of each test method in a reusable way, using one of the test rule classes that the Android Testing Support Library provides, such as [ActivityTestRule](#) or [ServiceTestRule](#).
- `@Test`: The `@ Test` annotation tells JUnit that the `public void` method to which it is attached can be run as a test case.

a

This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

-

To test code:

- [ViewMatchers](#) class lets you find a view in the current View hierarchy so that you can examine something or perform an action.

This work is licensed under [Creative Commons Attribution 4.0 International License](#).

for the latest updates.

[ViewActions](#) class lets you perform an action on a view found by a ViewMatcher.

- [ViewAssertions](#) class lets you assert or check the state of a view found by a ViewMatcher.

To test a spinner:

- Use onData() with a View that is dynamically populated by an adapter at runtime.
- Get items from an app's array by establishing the context with getActivity() and getting a resources instance using getResources().
- Use onData() to find and click each spinner item.
- Use onView() with a ViewAction and ViewAssertion to check if the output matches the selected spinner item.

Related concept

The related concept documentation is in [6.1: UI testing](#).

This work is licensed under a C

Attribution 4.0 International License.

This PDF is a one-time snapshot. See [developer.android.com/courses/fundamentals-training/toc-v2](#)

-

Learn more

Android Studio Documentation:

- [Test Your App](#)
- [Espresso basics](#)
- [Espresso cheat sheet](#)

Android Developer Documentation:

- [Best Practices for Testing](#)

- [Getting Started with Testing](#)
- [Testing UI for a Single App](#)

creative
Commons for the latest
updates.

-

Android Developer Fundamentals Course (V2) Unit

- [Building Instrumented Unit Tests](#)
- [Espresso Advanced Samples](#)
- [The Hamcrest Tutorial](#)
- [Hamcrest API and Utility Classes](#)
- [Test Support APIs](#)

Android Testing Support Library:

- [Espresso documentation](#)
- [Espresso Samples](#)

Videos

- [Android Testing Support - Android Testing Patterns #1](#) (introduction)
- [Android Testing Support - Android Testing Patterns #2](#) (onView view matching)
- [Android Testing Support - Android Testing Patterns #3](#) (onData and adapter views)

Other:

- Google Testing Blog: [Android UI Automated Testing](#)
- Atomic Object: "[Espresso – Testing RecyclerViews at Specific Positions](#)"
- Stack Overflow: "[How to assert inside a RecyclerView in Espresso?](#)"
- GitHub: [Android Testing Samples](#)
- Google Codelabs: [Android Testing Codelab](#)

Homework

Build and run an app

Record another Espresso test for the [ScorekeeperEspresso](#) app. This test should tap the **Night Mode** option in the options menu and determine whether the **Day Mode** option appears in its place. The test should then tap the **Day Mode** option and determine whether the **Night Mode** option appears in its place.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#). This PDF is one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

Android Developer Fundamentals Course (V2) Unit

Answer these questions

Question 1

Which steps do you perform to test a View interaction, and in what order? Choose one:

- Match a View, assert and verify the result, and perform an action.
- Match a View, perform an action, and assert and verify the result.
- Perform an action, match a view, and assert and verify the result.
- Perform an action, and assert and verify the result.

Question 2

Which of the following annotations enables an instrumented JUnit 4 test class? Choose one:

- @RunWith
- @Rule
- @Test
- @RunWith and @ Test

Question 3

Which method would you use to find a child View in an AdapterView? Choose one:

- onData() to load the adapter and enable the child View to appear on the screen.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2

- `onView()` to load the View from the current View hierarchy.
- `onView().check()` to check the current View.
- `onView().perform()` to perform a click on the current View.

reative
Commons for the latest
updates.

Android Developer Fundamentals Course (V2) Unit

Submit your app for grading

Guidance for graders

Check that the test meets the following criteria:

- The test appears in the `com.example.android.scorekeeper (androidTest)` folder.
- The test automatically switches the app from Day Mode to Night Mode, and then back to Day Mode.
- The test passes more than once.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#). This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2 for the latest updates.

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).
This PDF is a one-time snapshot. See developer.android.com/courses/fundamentals-training/toc-v2