

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



GIÁO TRÌNH
THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1.	LÀM QUEN	Error! Bookmark not defined.
Bài 1)	Tạo ứng dụng đầu tiên	Error! Bookmark not defined.
1.1)	Android Studio và Hello World.....	Error! Bookmark not defined.
1.2)	Giao diện người dùng tương tác đầu tiên	Error! Bookmark not defined.
1.3)	Trình chỉnh sửa bố cục	Error! Bookmark not defined.
1.4)	Văn bản và các chế độ cuộn	Error! Bookmark not defined.
1.5)	Tài nguyên có sẵn	Error! Bookmark not defined.
Bài 2)	Activities	Error! Bookmark not defined.
2.1)	Activity và Intent	Error! Bookmark not defined.
2.2)	Vòng đời của Activity và trạng thái	Error! Bookmark not defined.
2.3)	Intent ngầm định.....	Error! Bookmark not defined.
Bài 3)	Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ....	Error! Bookmark not defined.
3.1)	Trình gỡ lỗi.....	Error! Bookmark not defined.
3.2)	Kiểm thử đơn vị.....	Error! Bookmark not defined.
3.3)	Thư viện hỗ trợ.....	Error! Bookmark not defined.
CHƯƠNG 2.	TRẢI NGHIỆM NGƯỜI DÙNG	4
Bài 1)	Tương tác người dùng.....	4
1.1)	Hình ảnh có thể chọn	4
1.2)	Các điều khiển nhập liệu	33
1.3)	Menu và bộ chọn.....	57
1.4)	Điều hướng người dùng.....	81

1.5)	RecyclerView.....	93
Bài 2)	Trải nghiệm người dùng thú vị	106
2.1)	Hình vẽ, định kiểu và chủ đề	106
2.2)	Thẻ và màu sắc	122
2.3)	Bố cục thích ứng.....	138
Bài 3)	Kiểm thử giao diện người dùng	150
3.1)	Espresso cho việc kiểm tra UI.....	150
CHƯƠNG 3.	LÀM VIỆC TRONG NỀN.....	154
Bài 1)	Các tác vụ nền	154
1.1)	AsyncTask.....	154
1.2)	AsyncTask và AsyncTaskLoader	154
1.3)	Broadcast receivers	154
Bài 2)	Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền	154
2.1)	Thông báo.....	154
2.2)	Trình quản lý cảnh báo	154
2.3)	JobScheduler	154
CHƯƠNG 4.	LƯU DỮ LIỆU NGƯỜI DÙNG	154
Bài 1)	Tùy chọn và cài đặt	154
1.1)	Shared preferences	154
1.2)	Cài đặt ứng dụng	154
Bài 2)	Lưu trữ dữ liệu với Room	154
2.1)	Room, LiveData và ViewModel	154
2.2)	Room, LiveData và ViewModel	154

3.1) Trình gỡ lỗi

CHƯƠNG 1. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

1.1) Hình ảnh có thể chọn

Giới thiệu

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

Lớp View đại diện cho khối xây dựng cơ bản cho tất cả các thành phần giao diện người dùng. View là lớp cơ sở cho các lớp cung cấp các thành phần giao diện người dùng tương tác, chẳng hạn như các phần tử [Button](#). Một Button là một phần tử giao diện người dùng mà người dùng có thể chạm hoặc nhấp vào để thực hiện một hành động.

Bạn có thể biến bất kỳ View nào, chẳng hạn như [ImageView](#), thành một phần tử UI có thể được chạm hoặc nhấp. Bạn phải lưu hình ảnh cho ImageView trong thư mục drawables của dự án của bạn. Trong bài thực hành này, bạn sẽ học cách sử dụng hình ảnh như là các phần tử mà người dùng có thể chạm hoặc nhấp.

Những thứ bạn nên biết

Bạn có thể làm gì:

- Tạo một dự án Android Studio ở mẫu và tạo giao diện chính
- Chạy ứng dụng trên trình giả lập hoặc thiết bị đã kết nối
- Tạo và chỉnh sửa các yếu tố giao diện sử dụng trình chỉnh sửa giao diện và mã XML.
- Truy cập các yếu tố giao diện người dùng và sử dụng từ mã bởi [findViewById\(\)](#).
- Xử lý sự kiện [Button](#)
- Hiển thị thông báo [Toast](#).
- Thêm hình ảnh vào thư mục drawable của dự án.

Những điều bạn học

- Cách sử dụng hình ảnh như một yếu tố tương tác để thực hiện một hành động.
- Cách thiết lập thuộc tính cho các yếu tố ImageView trong trình chỉnh sửa bố cục.
- Cách thêm phương thức onClick() để hiển thị một thông báo Toast.

Những điều bạn làm

- Tạo một dự án Android Studio mới cho một ứng dụng đặt bánh giả sử dụng hình ảnh làm các yếu tố tương tác.
- Đặt các trình xử lý onClick() cho các hình ảnh để hiển thị các thông báo Toast khác nhau.
- Thay đổi nút hành động nổi được cung cấp bởi mẫu để nó hiển thị một biểu tượng khác và khởi động một Activity khác.

Tổng quan về ứng dụng

Tổng quan về ứng dụng trong thực hành này, bạn sẽ tạo và xây dựng một ứng dụng mới bắt đầu từ mẫu Hoạt động Cơ bản mô phỏng một ứng dụng đặt món tráng miệng. Người dùng có thể chạm vào một hình ảnh để thực hiện một hành động - trong trường hợp này là hiển thị một thông báo Toast - như được thể hiện trong hình bên dưới. Người dùng cũng có thể chạm vào một nút giỏ hàng để tiếp tục đến Hoạt động tiếp theo.

Task 1: Thêm hình ảnh vào bố cục

Bạn có thể làm cho một giao diện có thể nhấp vào, như một nút, bằng cách thêm thuộc tính android:onClick trong layout XML. Ví dụ, bạn có thể làm cho một hình ảnh hoạt động như một nút bằng cách thêm android:onClick vào [ImageView](#).

Trong nhiệm vụ này, bạn tạo một nguyên mẫu ứng dụng đặt món tráng miệng từ một quán cà phê. Sau khi bắt đầu một dự án mới dựa trên mẫu Basic Activity, bạn chỉnh sửa TextView “Hello World” với văn bản phù hợp và thêm hình ảnh mà người dùng có thể chạm vào.

1.1 Bắt đầu dự án mới

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

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

3. Nhấn **Finish**.

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

4. Mở **content_main.xml** và nhấn vào tab **Design** (nếu chưa được chọn) để hiển thị trình soạn thảo layout.

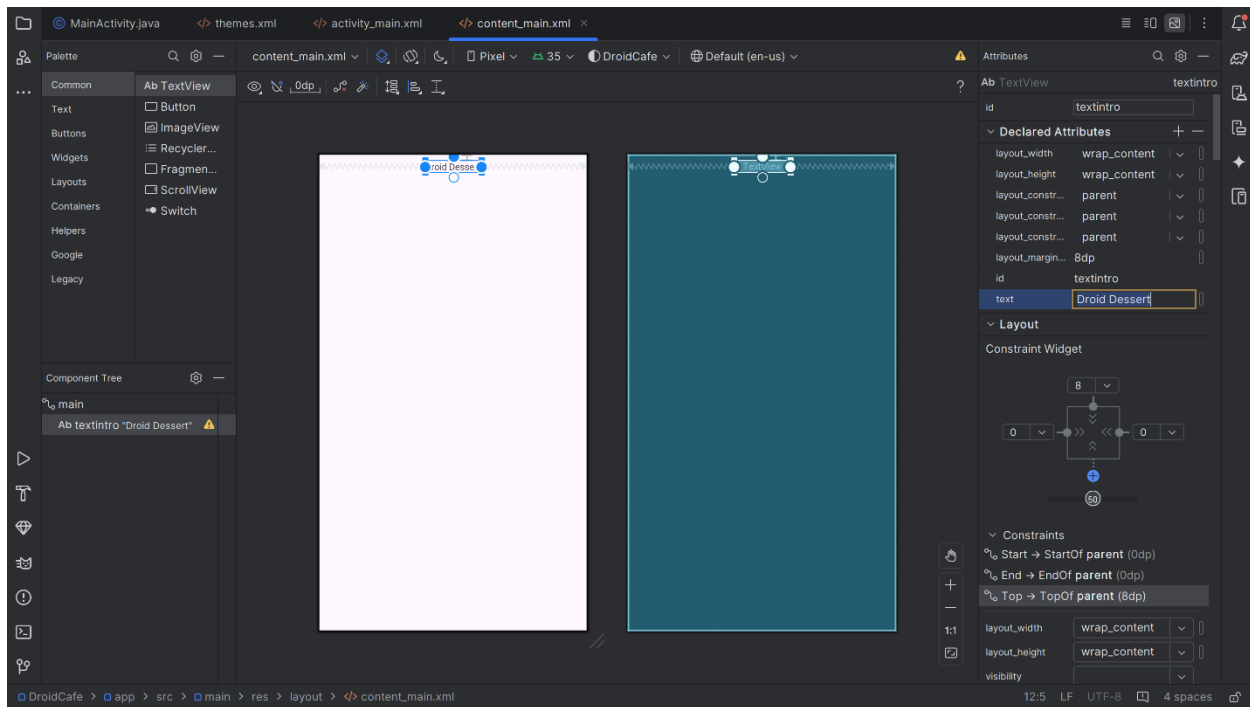
5. Chọn TextView "Hello World" trong layout và mở bảng **Attributes**.

6. Thay đổi các thuộc tính textintro như sau:

Attribute field	Enter the following:
ID	textintro
text	Change Hello World to Droid Dessert
textStyle	B (bold)
textSize	24sp

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

7. Xóa ràng buộc kéo dài từ đáy của TextView textintro đến đáy của bố cục, để TextView gắn vào đỉnh của bố cục, và chọn **8 (8dp)** cho khoảng cách ở trên như hình dưới.



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

1.2 Thêm hình ảnh

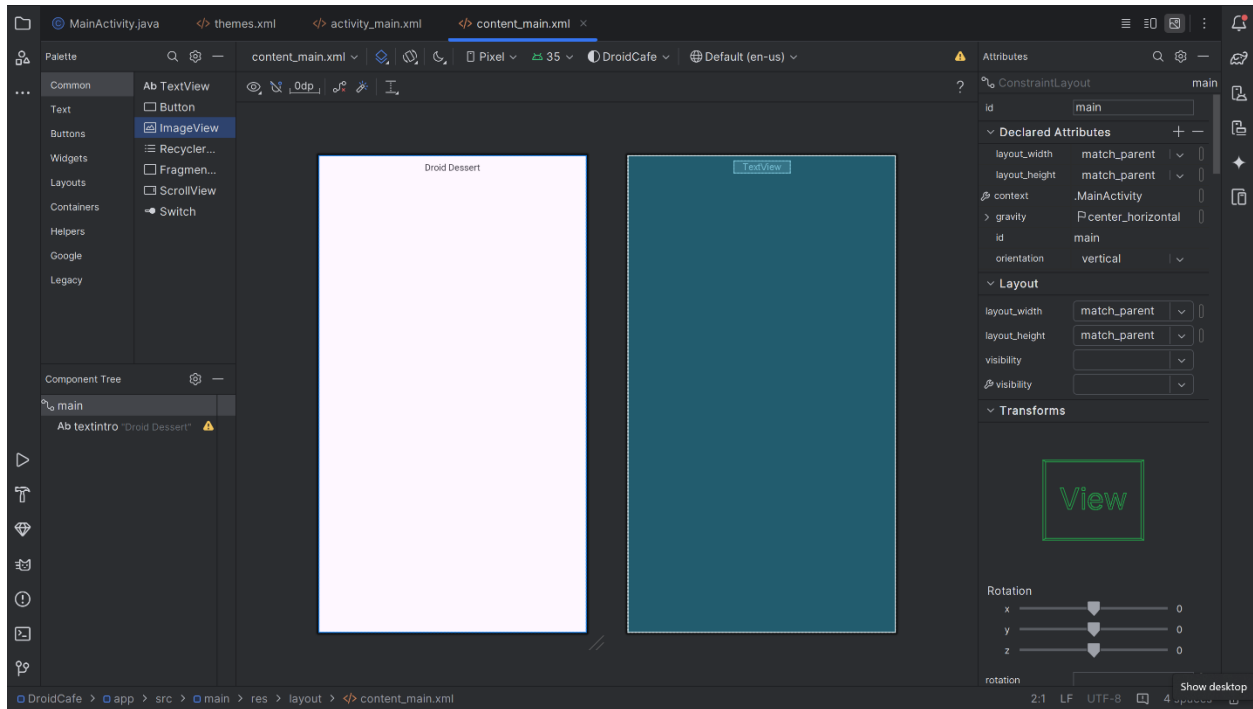
Ba hình ảnh (donut_circle.png, froyo_circle.png và icecream_circle.png) được cung cấp cho ví dụ này, bạn có thể [download](#) .Thay vào đó, bạn có thể thay thế bằng các hình ảnh của riêng bạn dưới dạng tệp PNG, nhưng chúng phải có kích thước khoảng 113 x 113 pixel để sử dụng trong ví dụ này.

Bước này cũng giới thiệu một kỹ thuật mới trong trình chỉnh sửa bố cục: sử dụng nút **Fix** trong các tin nhắn cảnh báo để trích xuất tài nguyên chuỗi.

1. Để sao chép hình ảnh vào dự án của bạn, trước tiên hãy đóng dự án.
2. Sao chép các tệp hình ảnh vào thư mục **drawable** của dự án của bạn. Tìm thư mục **drawable** trong một dự án bằng cách sử dụng đường dẫn này: **project_name > app > src > main > res > drawable**.
3. Mở lại dự án của bạn.

4. Mở tệp **content_main.xml** và nhấp vào tab **Design** (nếu nó chưa được chọn).

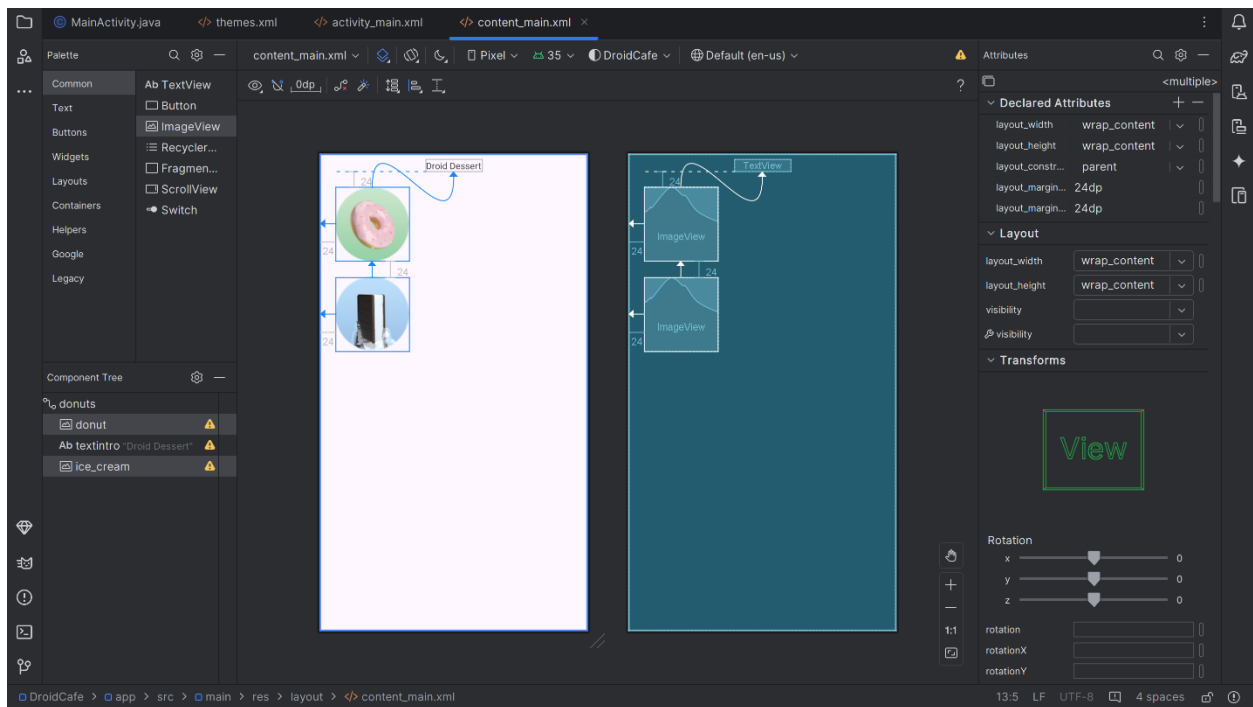
5. Kéo một `ImageView` vào bố cục, chọn hình ảnh **donut_circle** cho nó và ràng buộc nó với `TextView` ở trên cùng và phía bên trái của bố cục với khoảng cách **24** (24dp) cho cả hai ràng buộc, như được thể hiện trong hình chuyển động bên dưới.



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

Attribute field	Enter the following:
ID	donut
contentDescription	Donuts are glazed and sprinkled with candy. (You can copy/paste the text into the field.)

7. Kéo một `ImageView` thứ hai vào bố cục, chọn hình ảnh **icecream_circle** cho nó, và ràng buộc nó vào đáy của `ImageView` đầu tiên và vào bên trái của bố cục với một khoảng cách **24** (24dp) cho cả hai ràng buộc.



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


Attribute field	Enter the Following
ID	ice_cream
contentDescription	Ice cream sandwiches have chocolate wafers and vanilla filling. (You can copy/paste the text into the field.)

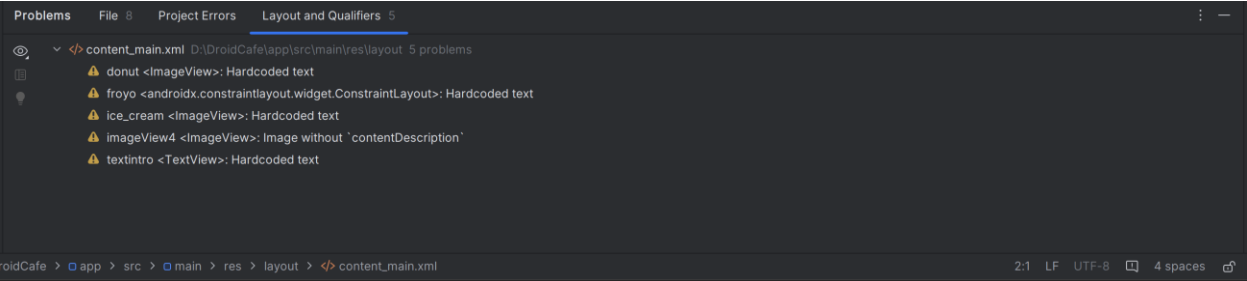
9. Kéo một ImageView thứ ba vào bố cục, chọn hình ảnh **froyo_circle** cho nó, và ràng buộc nó vào đáy của ImageView thứ hai và cạnh trái của bố cục với một khoảng cách **24** (24dp) cho cả hai ràng buộc.

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

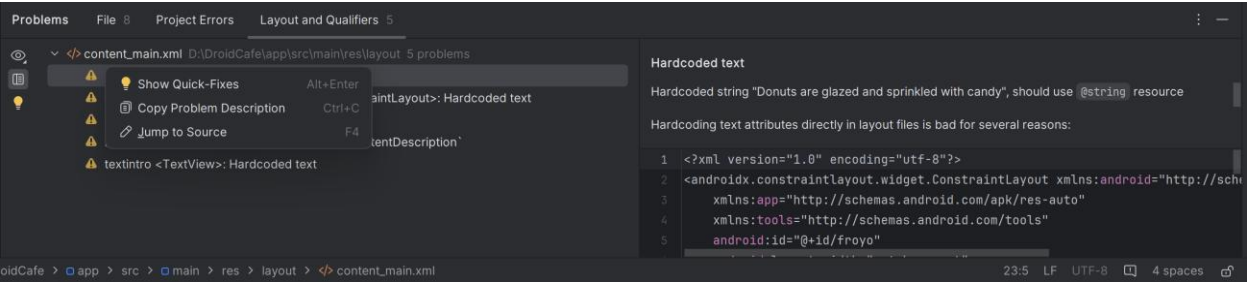
Attribute field	Enter the Following
ID	froyo

contentDescription	FroYo is premium self-serve frozen yogurt.(You can copy/paste the text into the field.)
--------------------	---

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



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

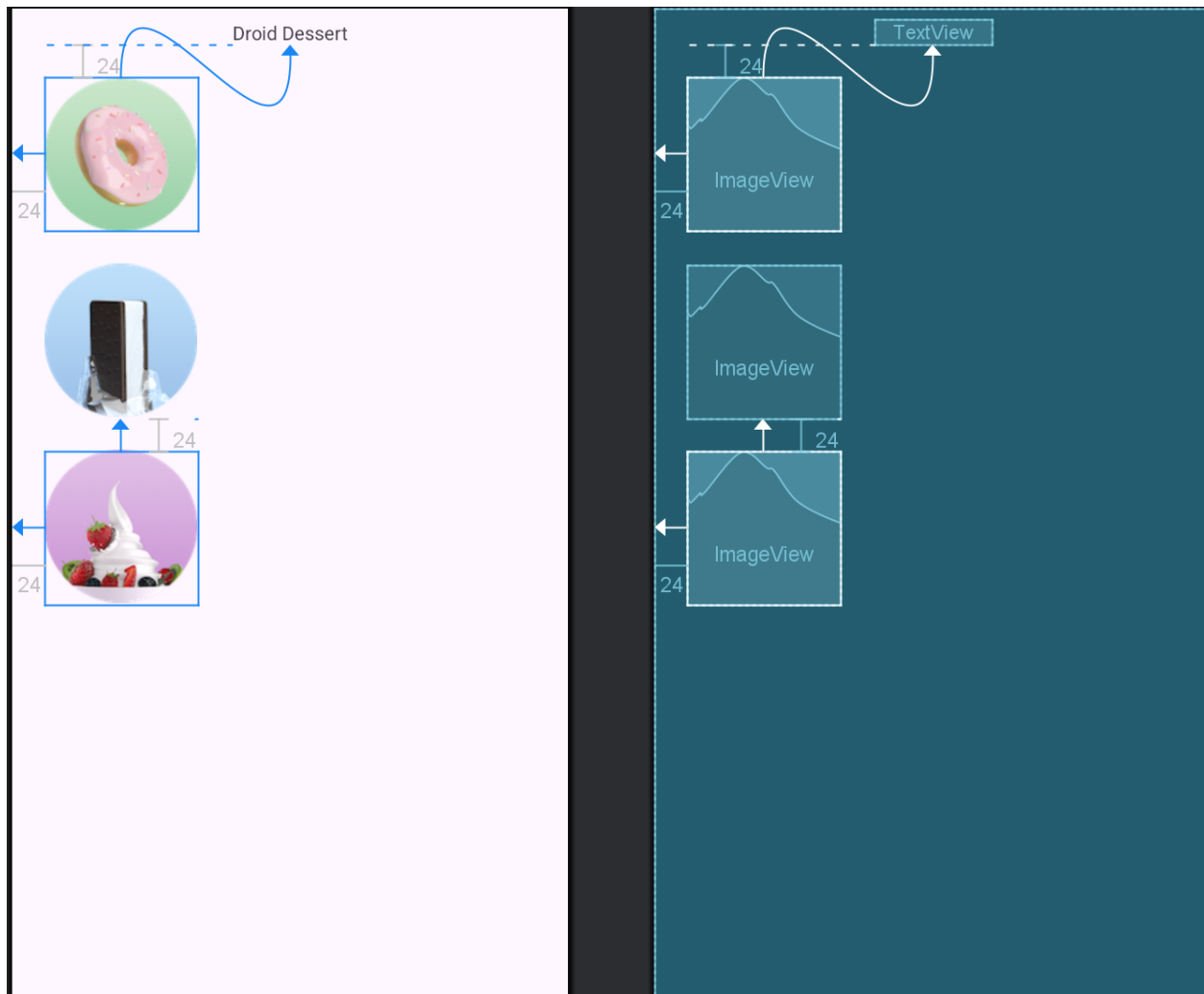


Sửa lỗi cho mỗi cảnh báo văn bản mã cứng trích xuất tài nguyên chuỗi cho chuỗi. Hộp thoại **Extract Resource** xuất hiện, và bạn có thể nhập tên cho tài nguyên chuỗi. Nhập các tên sau cho các tài nguyên chuỗi:

String	Enter the following name:
Donuts are glazed and sprinkled with candy.	donuts
Ice cream sandwiches have chocolate wafers and vanilla fillings.	ice_cream_sandwiches

FroYo is premium self-serve frozen yogurt.	froyo
--	-------

Bố cục sẽ giống hình dưới đây

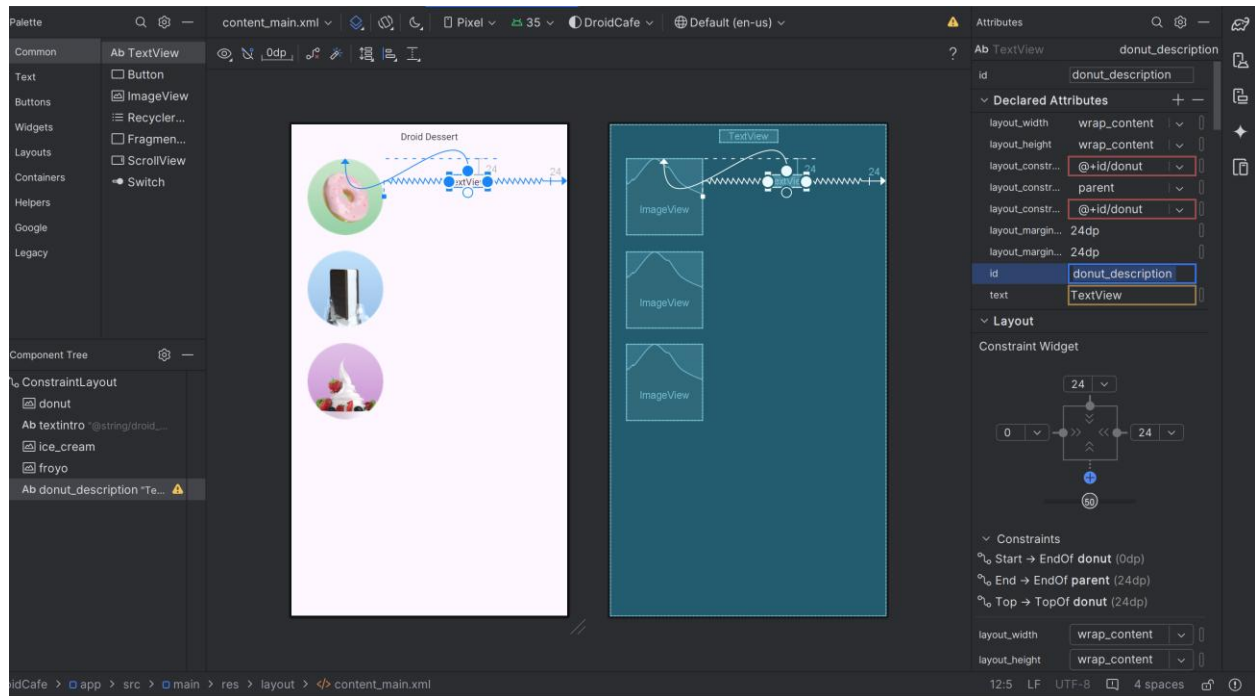


1.3 Thêm miêu tả văn bản

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

1. Kéo một phần tử `TextView` vào bố cục.

2. Ràng buộc cạnh trái của phần tử với cạnh phải của ImageView donut và cạnh trên với cạnh trên của ImageView donut, cả hai đều có khoảng cách **24** (24dp).
3. Ràng buộc cạnh phải của phần tử với cạnh phải của bố cục và sử dụng khoảng cách giống như 24 (24dp). Nhập **donut_description** vào trường ID trong bảng thuộc tính. **TextView** mới sẽ xuất hiện bên cạnh hình ảnh donut như hình dưới đây.



4. Trong thanh Thuộc tính, thay đổi độ rộng trong bảng kiểm tra thành **Match Constraints**:

id

donut_description

Declared Attributes

layout_width

0dp

|

▼

0

layout_height

wrap_content

|

▼

0

layout_constr...

@+id/donut

|

▼

0

layout_constr...

parent

|

▼

0

layout_constr...

@+id/donut

|

▼

0

layout_margin...

24dp

0

layout_margin...

24dp

0

id

donut_description

text

TextView

0

Layout

Constraint Widget

24

▼

0

▼

24

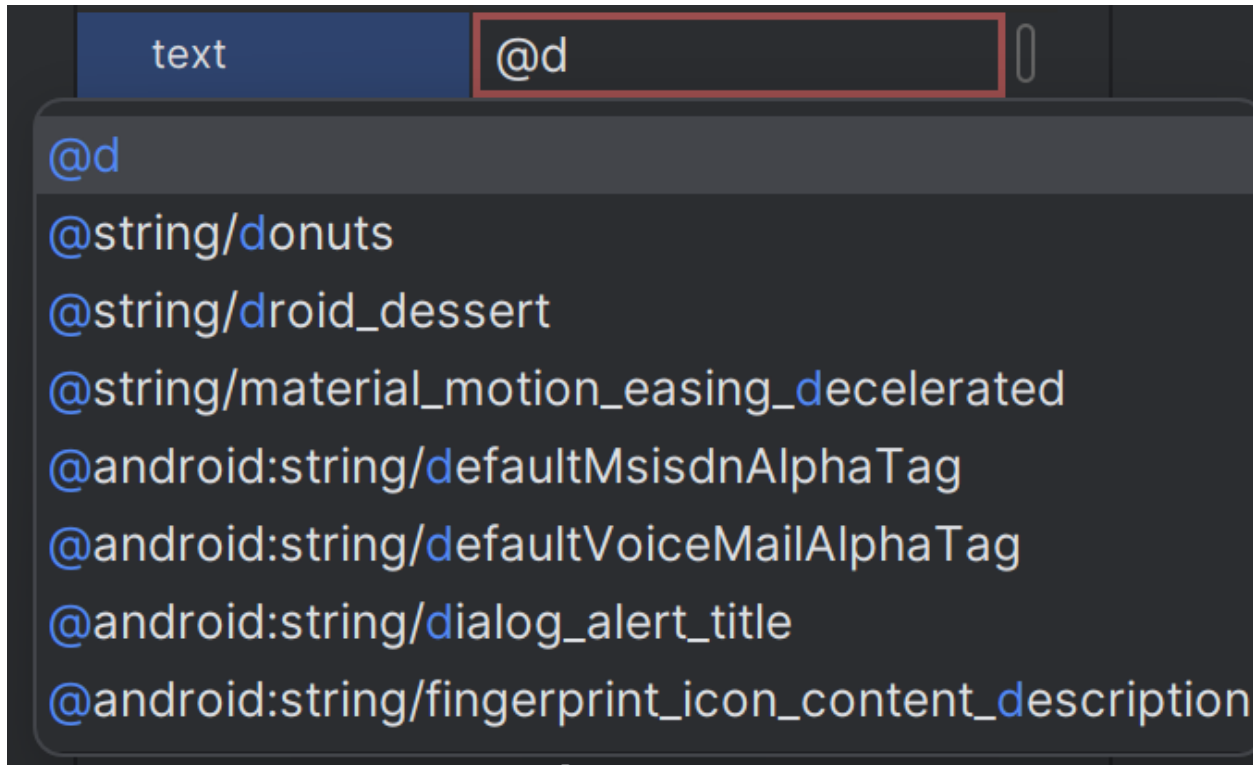
▼

Match Constraints

50

5. Trong bảng Thuộc tính, bắt đầu nhập tài nguyên chuỗi cho trường văn bản bằng cách đặt trước nó với ký hiệu **@**: **@d**. Nhấp vào tên tài nguyên chuỗi (**@string/donuts**) mà xuất hiện như một gợi ý

Gợi ý



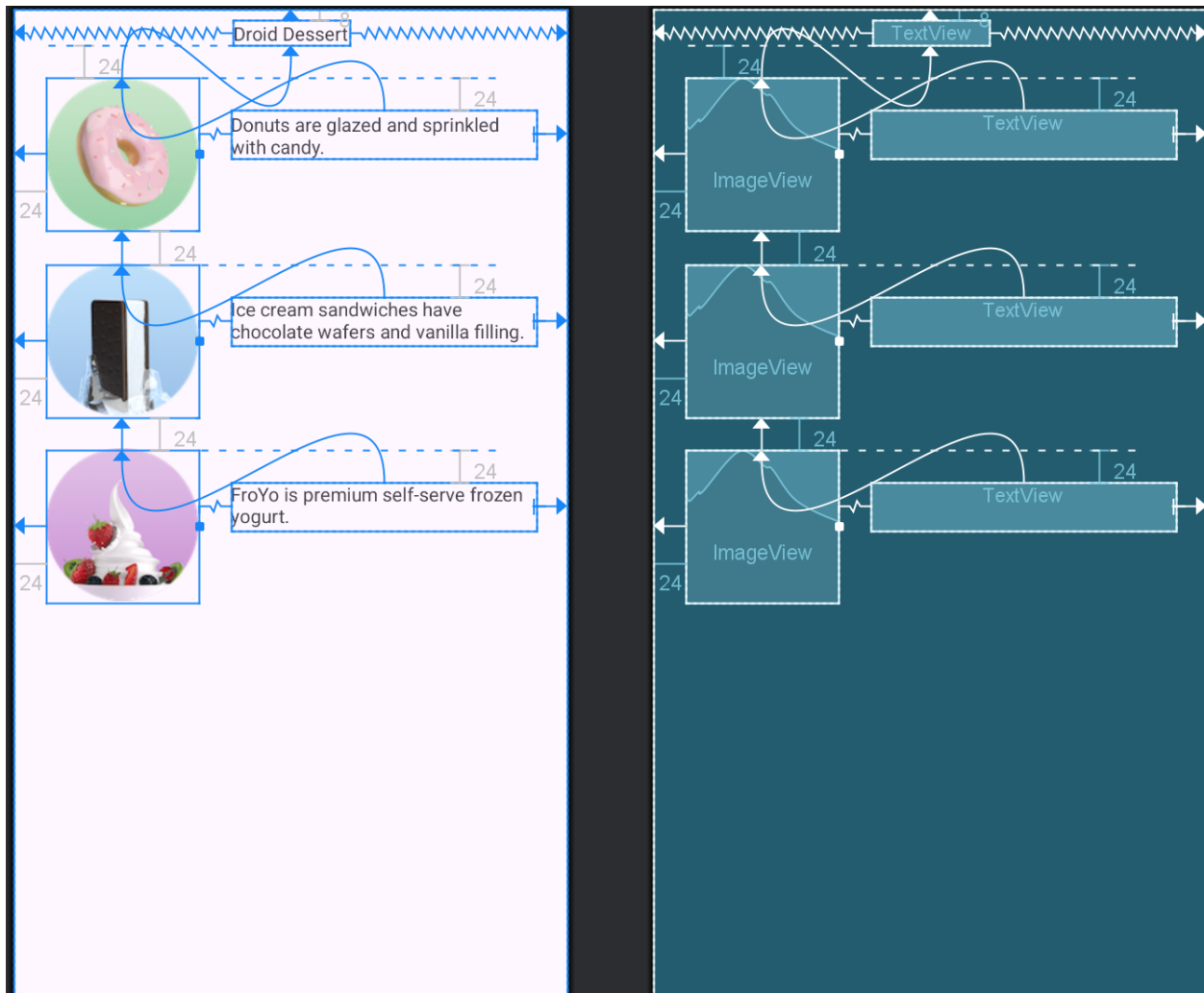
6. Lặp lại các bước trên để thêm một TextView thứ hai được ràng buộc ở bên phải và trên của ImageView ice_cream, và cạnh phải của nó ràng buộc với cạnh phải của bố cục. Nhập thông tin sau vào bảng Thuộc tính:

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

7. Lặp lại các bước trên để thêm một TextView thứ ba bị ràng buộc ở bên phải và phía trên của froyo ImageView, và bên phải của nó với bên phải của layout. Nhập những thông tin sau vào bảng Attributes :

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

Bố cục sẽ như sau:



Task 1: mã giải pháp

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

```
<?xml version="1.0" encoding="utf-8"?>

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

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

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"
```



```
android:layout_height="match_parent"

android:contentDescription="@string/froyo"

android:gravity="center_horizontal"

android:orientation="vertical"

tools:context=".MainActivity">

<ImageView

android:id="@+id/donut"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginStart="24dp"

android:layout_marginTop="24dp"

android:contentDescription="@string/donuts"

android:src="@drawable/donut_circle"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/textintro"

tools:ignore="ImageContrastCheck" />

<TextView

android:id="@+id/textintro"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginTop="8dp"
```

```
android:text="@string/droid_dessert"

android:textSize="24sp"

android:textStyle="bold"

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

```
<ImageView

android:id="@+id/ice_cream"

android:layout_width="0dp"

android:layout_height="wrap_content"

android:layout_marginStart="24dp"

android:layout_marginTop="24dp"

android:contentDescription="@string/ice_cream_sandwiches"

app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/donut"
app:srcCompat="@drawable/icecream_circle" />
```

```
<ImageView

android:id="@+id/froyo"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginStart="24dp"
```

```
android:layout_marginTop="24dp"

android:contentDescription="@string/froyo"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/ice_cream"

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

```
<TextView

android:id="@+id/donut_description"

android:layout_width="0dp"

android:layout_height="wrap_content"

android:layout_marginStart="24dp"

android:layout_marginTop="24dp"

android:layout_marginEnd="24dp"

android:text="@string/donuts"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toEndOf="@+id/donut"

app:layout_constraintTop_toTopOf="@+id/donut" />
```

```
<TextView

android:id="@+id/ice_cream_description"

android:layout_width="0dp"

android:layout_height="wrap_content"

android:layout_marginStart="24dp"
```

```
android:layout_marginTop="24dp"
android:layout_marginEnd="24dp"
android:text="@string/ice_cream_sandwiches"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toEndOf="@+id/ice_cream"
app:layout_constraintTop_toTopOf="@+id/ice_cream" />

<TextView
android:id="@+id/froyo_description"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_marginStart="24dp"
android:layout_marginTop="24dp"
android:layout_marginEnd="24dp"
android:text="@string/froyo"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toEndOf="@+id/froyo"
app:layout_constraintTop_toTopOf="@+id/froyo" />

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

Task 2: Thêm phương thức onClick cho hình ảnh

Để làm cho một View *clickable*, để người dùng có thể chạm (hoặc nhấn) vào nó, hãy thêm thuộc tính [android:onClick](#) trong bố cục XML và chỉ định bộ xử lý nhấp. Ví dụ, bạn có thể làm cho một ImageView hoạt động như một nút đơn giản bằng cách thêm android:onClick vào [ImageView](#). Trong nhiệm vụ này, bạn làm cho các hình ảnh trong bố cục của bạn có thể nhấn được.

2.1 Tạo phương thức Toast

Trong nhiệm vụ này, bạn thêm từng phương thức cho thuộc tính android:onClick để gọi khi mỗi hình ảnh được nhấp. Trong nhiệm vụ này, các phương thức này đơn giản hiển thị một thông điệp [Toast](#) cho biết hình ảnh nào đã được chạm. (Trong một chương khác, bạn sửa đổi các phương thức này để khởi động một Activity khác.)

1. Để sử dụng tài nguyên chuỗi trong mã Java, bạn nên thêm chúng vào tệp strings.xml trước. Mở rộng **res > values** trong bảng **Project > Android**, và mở tệp **strings.xml**. Thêm các tài nguyên chuỗi sau cho các chuỗi sẽ được hiển thị trong thông điệp Toast:

```
<string name="donut_order_message">You ordered a donut.</string>
<string name="ice_cream_order_message">You ordered an ice cream sandwich.</string>
<string name="froyo_order_message">You ordered a FroYo.</string>
```

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

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

Mặc dù bạn có thể đã thêm phương thức này ở bất kỳ vị trí nào trong **MainActivity**, nhưng thực tiễn tốt nhất là đặt các phương thức của bạn bên dưới các phương thức đã được cung cấp trong **MainActivity** bởi mẫu.

2.2 Tạo trình xử lý sự kiện khi nhấp chuột

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

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

```
/**
 * Shows a message that the donut image was clicked.
 */
no usages
public void showDonutOrder(View view) {
    displayToast(getString(R.string.donut_order_message));
}
```

Ba dòng đầu tiên là một chú thích theo định dạng [Javadoc](#), giúp cho mã dễ hiểu hơn và cũng giúp tạo tài liệu cho mã của bạn. Đây là một thực tiễn tốt nhất để thêm chú thích như vậy cho mỗi phương thức mới bạn tạo. Để biết thêm thông tin về cách viết chú thích, hãy xem [Cách viết chú thích tài liệu cho công cụ Javadoc](#).

2. Thêm nhiều phương thức vào cuối **MainActivity** cho mỗi món tráng miệng:

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

3. (Tùy chọn) Chọn **Code > Reformat Code** để định dạng lại mã mà bạn đã thêm vào **MainActivity** cho phù hợp với tiêu chuẩn và dễ đọc hơn.

2.3 Thêm thuộc tính `onClick`

Trong bước này, bạn thêm `android:onClick` vào từng phần tử `ImageView` trong tập tin `content_main.xml`. Thuộc tính `android:onClick` gọi trình xử lý click cho từng phần tử.

1. Mở tập tin **content_main.xml**, và nhấp vào tab **Text** trong trình chỉnh sửa bố cục để hiển thị mã XML.
2. Thêm thuộc tính `android:onClick` vào `ImageView` donut. Khi bạn nhập, sẽ có các gợi ý hiển thị các trình xử lý click. Chọn trình xử lý click `showDonutOrder`. Mã hiện tại sẽ trông như sau:

```
<ImageView
    android:id="@+id/donut"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:layout_marginStart="24dp"
    android:layout_marginTop="24dp"
    android:contentDescription="Donuts are glazed and sprinkled with candy."
    android:onClick="showDonutOrder"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textintro"
    app:srcCompat="@drawable/donut_circle"
    tools:ignore="ImageContrastCheck" />
```

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

3. (Tùy chọn) Chọn **Code > Reformat Code** để định dạng lại mã XML bạn đã thêm vào `content_main.xml` để tuân thủ các tiêu chuẩn và dễ đọc hơn. Android Studio tự động di chuyển thuộc tính `android:onClick` lên vài dòng để kết hợp chúng với các thuộc tính khác có `android:` làm tiền tố.
4. Thực hiện quy trình tương tự để thêm thuộc tính `android:onClick` vào các phần tử `ImageView` `ice_cream` và `froyo`. Chọn các trình xử lý nhấp chuột `showDonutOrder` và `showFroyoOrder`. Bạn có thể tùy chọn chọn **Code > Reformat Code** để định dạng lại mã XML. Mã bây giờ sẽ trông như sau:

```
<ImageView
    android:id="@+id/ice_cream"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="24dp"
    android:contentDescription="Ice cream sandwiches have chocolate wafes and vanilla f..."
    android:onClick="showIceCreamOrder"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/donut"
    app:srcCompat="@drawable/icecream_circle" />
```

```
<ImageView
    android:id="@+id/froyo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginTop="24dp"
    android:contentDescription="FroYo is premium self-serve frozen yogurt."
    android:onClick="showFroyoOrder"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/ice_cream"
    app:srcCompat="@drawable/froyo_circle" />
```

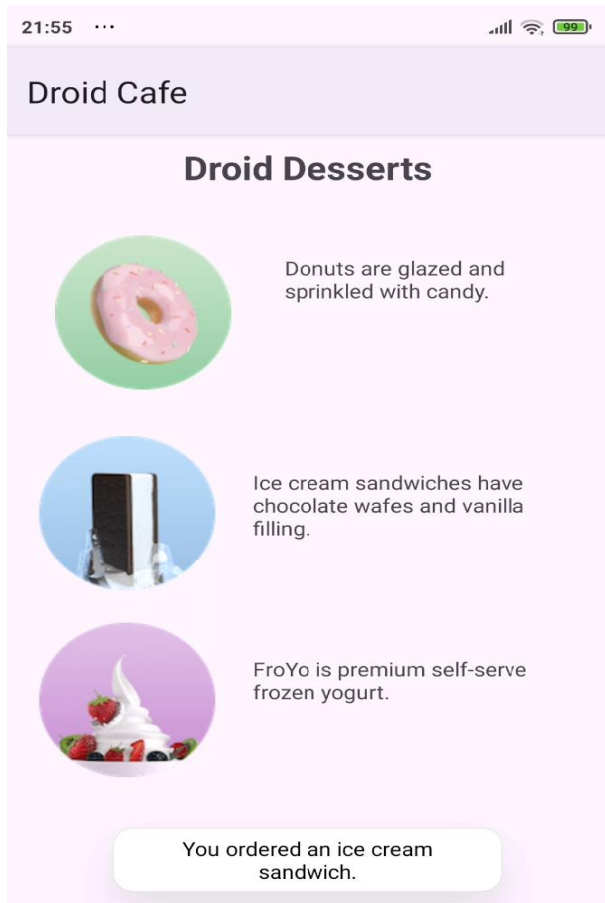
Lưu ý rằng thuộc tính `android:layout_marginStart` trong mỗi `ImageView` được gạch chân bằng màu đỏ. Thuộc tính này xác định "lề" bắt đầu cho `ImageView`, mà bên trái đối với hầu hết các ngôn ngữ nhưng bên phải đối với các ngôn ngữ đọc từ phải sang trái (RTL).

5. Nhấn vào phần mở đầu `android:` của thuộc tính `android:layout_marginStart`, và một biểu tượng đèn đỏ cảnh báo xuất hiện bên cạnh, như được hiển thị trong hình dưới đây.

6. Để làm cho ứng dụng của bạn tương thích với các phiên bản Android trước đó, hãy nhấp vào bóng đèn màu đỏ cho từng trường hợp của thuộc tính này và chọn **Set layout_marginLeft...** để đặt `layout_marginLeft` thành `"@dimen/margin_wide"`.

7. Chạy ứng dụng.

Nhấp vào hình ảnh bánh donut, bánh sandwich kem hoặc froyo sẽ hiển thị một thông báo Toast về đơn hàng, như hiển thị trong hình bên dưới.



Task 2 mã giải pháp


Mã giải pháp cho nhiệm vụ này được bao gồm trong mã và bố cục của **MainActivity** trong dự án **DroidCafe** trên Android Studio.

Task 3: Thay đổi nút cho hành động nổi

Khi bạn nhấn vào nút hành động nổi có biểu tượng email xuất hiện ở dưới cùng của màn hình, mã trong MainActivity sẽ hiển thị một tin nhắn ngắn trong một ngăn kéo mở ra từ dưới cùng của màn hình trên điện thoại thông minh hoặc từ góc dưới bên trái trên các thiết bị lớn hơn, sau đó tự động đóng sau vài giây. Đây được gọi là snackbar. Nó được sử dụng để cung cấp phản hồi về một thao tác. Để biết thêm thông tin, hãy xem [Snackbar](#).

Hãy xem cách các ứng dụng khác triển khai nút hành động nổi (Floating Action Button). Ví dụ: ứng dụng Gmail cung cấp một nút hành động nổi để tạo email mới, và ứng dụng

Contacts cung cấp một nút để tạo một liên hệ mới. Để biết thêm thông tin về Floating Action Button, hãy xem [FloatingActionButton](#).

Trong nhiệm vụ này, bạn sẽ thay đổi biểu tượng của **FloatingActionButton** thành,  và thay đổi hành động của **FloatingActionButton** để mở một **Activity** mới.

3.1 Thêm biểu tượng mới

Như bạn đã học trong bài học khác, bạn có thể chọn một biểu tượng từ bộ biểu tượng trong Android Studio.

Hãy làm theo các bước sau:

1. Mở rộng **res** trong bảng **Project > Android**, sau đó nhấp chuột phải (hoặc Control + nhấp) vào thư mục **drawable**.
2. Chọn **New > Image Asset**. Hộp thoại Configure Image Asset sẽ xuất hiện.
3. Trong menu thả xuống ở đầu hộp thoại, chọn **Action Bar and Tab Icons**. (Lưu ý rằng action bar chính là app bar.)
4. Thay đổi tên trong trường **Name** từ **ic_action_name** thành **ic_shopping_cart**.
5. Nhấp vào hình ảnh clip art (biểu tượng Android bên cạnh **Clipart:**) để chọn một hình ảnh clip art làm biểu tượng. Một trang chứa các biểu tượng sẽ xuất hiện. Nhấp vào biểu tượng bạn muốn sử dụng cho Floating Action Button, chẳng hạn như biểu tượng giỏ hàng. 
6. Chọn **HOLO_DARK** từ menu thả xuống **Theme**. Điều này giúp biểu tượng có màu trắng trên nền tối (hoặc đen). Nhấp vào **Next**.
7. Nhấp vào **Finish** trong hộp thoại Confirm Icon Path.

Mẹo: Để có mô tả đầy đủ về cách thêm biểu tượng, hãy xem [Create app icons with Image Asset Studio](#)

3.2 Thêm Activity

Như bạn đã học trong bài học trước, một Activity đại diện cho một màn hình trong ứng dụng, nơi người dùng có thể thực hiện một tác vụ cụ thể. Bạn đã có một activity là MainActivity.java. Bây giờ, bạn sẽ thêm một activity mới có tên OrderActivity.java.

1. Nhấp chuột phải (hoặc Control + nhấp) vào thư mục **com.example.android.droidcafe** trong cột bên trái, sau đó chọn **New > Activity > Empty Activity**.
2. Chỉnh sửa **Activity Name** thành **OrderActivity** và **Layout Name** thành **activity_order**.
3. Giữ nguyên các tùy chọn khác và nhấp vào **Finish**.

Sau khi hoàn thành, lớp OrderActivity sẽ xuất hiện cùng với MainActivity trong thư mục java, và tệp activity_order.xml sẽ xuất hiện trong thư mục **layout**. Empty Activity template đã tự động tạo các tệp này.

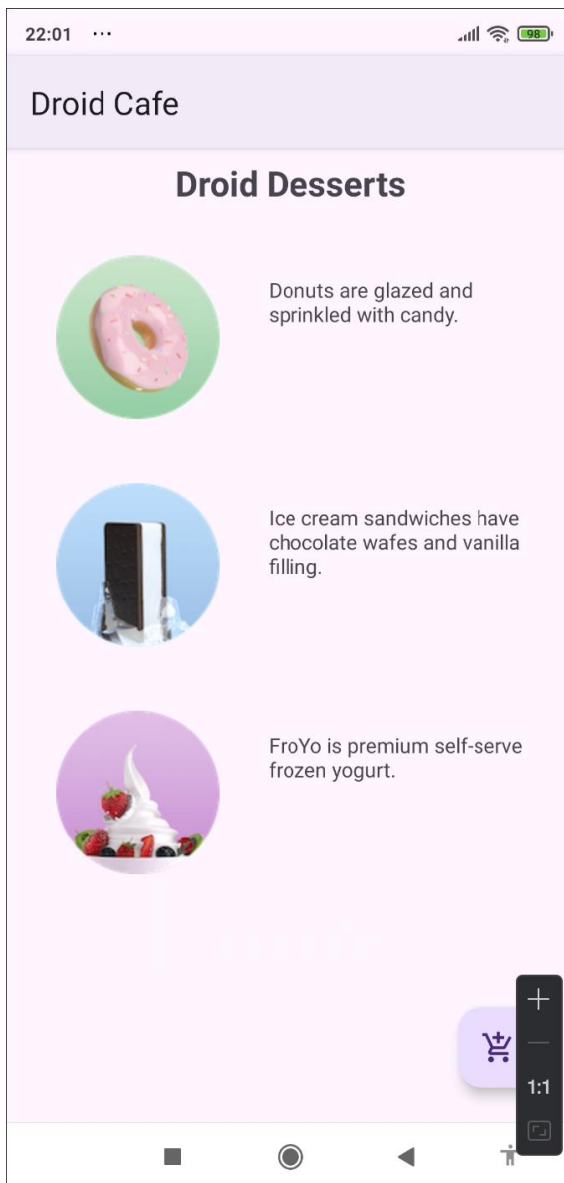
3.3 Thay đổi hành động

Trong bước này, bạn sẽ thay đổi hành động của FloatingActionButton để mở Activity mới.

1. Mở **MainActivity**.
2. Thay đổi phương thức onClick(View view) để tạo một explicit intent nhằm khởi động OrderActivity.

```
public void showFroyoOrder(View view) { displayToast("You ordered a FroYo."); }  
1 usage  
public void onClick(View v) {  
    Intent intent = new Intent( packageContext: MainActivity.this, OrderActivity.class);  
    intent.putExtra(EXTRA_MESSAGE, mOrderMessage);  
    startActivity(intent);  
}
```

3. Chạy ứng dụng. Nhấn vào floating action button hiện đang sử dụng biểu tượng giỏ hàng. Một Activity trống (OrderActivity) sẽ xuất hiện. Nhấn nút Back để quay lại MainActivity.



Task 3 Mã giải pháp
Mã giải pháp cho nhiệm vụ này được bao gồm trong mã và bố cục của dự án Android Studio [DroidCafe](#).

Thử thách lập trình
Lưu ý: Tất cả các thử thách lập trình là tùy chọn và không phải là yêu cầu tiên quyết cho các bài học sau.

Thử thách: Ứng dụng DroidCafe có MainActivity khởi động một Activity thứ hai có tên OrderActivity.

Bạn đã học trong bài học khác cách gửi dữ liệu từ một Activity này sang Activity khác. Hãy thay đổi ứng dụng để gửi thông điệp đơn hàng cho món tráng miệng đã chọn trong MainActivity đến một TextView mới ở trên cùng của bố cục OrderActivity.

1. Thêm một TextView ở trên cùng của bố cục OrderActivity với id là order_textview.
2. Tạo một biến thành viên (mOrderMessage) trong MainActivity cho thông điệp đơn hàng sẽ hiển thị trong Toast.
3. Thay đổi các bộ xử lý nhấp chuột showDonutOrder(), showIceCreamOrder(), và showFroyoOrder() để gán chuỗi thông điệp mOrderMessage trước khi hiển thị Toast. Ví dụ, đoạn mã sau đây gán chuỗi donut_order_message cho mOrderMessage và hiển thị Toast:

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

4. Thêm một public static final String có tên EXTRA_MESSAGE ở đầu MainActivity để định nghĩa khóa cho intent.putExtra:

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

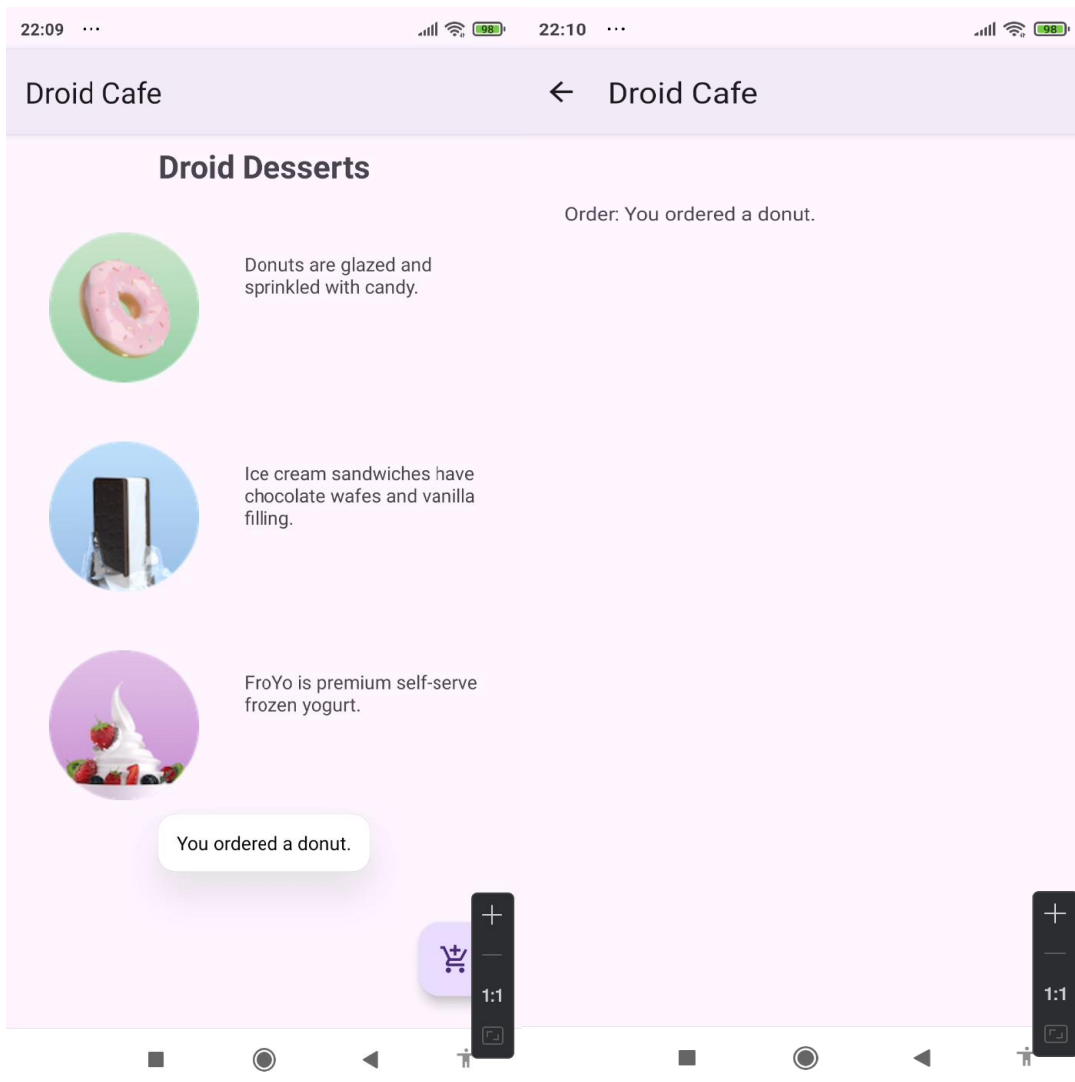
5. Thay đổi phương thức onClick() để bao gồm câu lệnh intent.putExtra trước khi khởi động OrderActivity:

```
public void onClick(View v) {
    Intent intent = new Intent( packageContext: MainActivity.this, OrderActivity.class);
    intent.putExtra(EXTRA_MESSAGE, mOrderMessage);
    startActivity(intent);
}
```

6. Trong OrderActivity, thêm mã sau vào phương thức onCreate() để lấy Intent khởi động Activity, trích xuất thông điệp chuỗi, và thay thế văn bản trong TextView với thông điệp:

```
Intent intent = getIntent();
String message = "Order: " + intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
TextView textView = findViewById(R.id.order_textview);
textView.setText(message);
```

7. Chạy ứng dụng. Sau khi chọn hình ảnh món tráng miệng, nhấn vào floating action button để khởi động OrderActivity, và OrderActivity sẽ hiển thị thông điệp đơn hàng như trong hình dưới đây.



Giải pháp cho thử thách
 Dự án Android Studio: [DroidCafeChallenge](#)

Tóm tắt

- Để sử dụng một hình ảnh trong dự án, sao chép hình ảnh vào thư mục **drawable** của dự án (**project_name** > **app** > **src** > **main** > **res** > **drawable**).
- Định nghĩa một ImageView để sử dụng nó bằng cách kéo thả ImageView vào bố cục và chọn hình ảnh cho nó.
- Thêm thuộc tính android:onClick để làm cho ImageView có thể nhấp được như một nút. Chỉ định tên của bộ xử lý nhấp chuột.
- Tạo một bộ xử lý nhấp chuột trong Activity để thực hiện hành động.
- Chọn biểu tượng: Mở rộng **res** trong bảng **Project > Android**, nhấp chuột phải (hoặc Control + nhấp) vào thư mục **drawable**, và chọn **New > Image Asset**. Chọn **Action Bar and Tab Icons** trong menu thả xuống, và nhấp vào hình ảnh clip art (biểu tượng Android bên cạnh **Clipart**;) để chọn một hình ảnh clip art làm biểu tượng.
- Thêm một Activity khác: Trong bảng **Project > Android**, nhấp chuột phải (hoặc Control + nhấp) vào thư mục tên gói trong thư mục java và chọn **New > Activity** và một mẫu cho Activity (chẳng hạn như **Empty Activity**).
- Hiển thị một [Toast](#) message.

Khái niệm liên quan

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

Tìm hiểu thêm

Tài liệu Android Studio:

- [Android Studio User Guide](#)
- [Create app icons with Image Asset Studio](#)

Tài liệu dành cho nhà phát triển Android:

- [User interface & navigation](#)
- [Build a UI with Layout Editor](#)
- [Build a Responsive UI with ConstraintLayout](#)
- [Layouts](#)
- [View](#)


- [Button](#)
- [ImageView](#)
- [TextView](#)
- [Buttons](#)

Khác:

- Codelabs: [Using ConstraintLayout to design your Android views](#)

Bài tập về nhà
Thay đổi ứng dụng

Ứng dụng [DroidCafe](#) hiển thị tốt khi thiết bị hoặc trình giả lập được xoay theo hướng dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang hướng ngang, các hình ảnh thứ hai và thứ ba không xuất hiện.

1. Mở (hoặc tải về) dự án ứng dụng [DroidCafe](#).
2. Tạo một biến thể bố cục cho hướng ngang: content_main.xml (land).
3. Loại bỏ các ràng buộc từ ba hình ảnh và ba mô tả văn bản.
4. Chọn tất cả ba hình ảnh trong biến thể bố cục, và chọn **Expand Horizontally** trong nút Pack  để phân phối đều các hình ảnh trên màn hình như hình dưới đây.
5. Ràng buộc các mô tả văn bản vào các cạnh và đáy của hình ảnh như hình dưới đây.

Trả lời câu hỏi

Câu hỏi 1

Làm thế nào để bạn thêm hình ảnh vào một dự án Android Studio? Chọn một trong các đáp án sau:

- Kéo từng hình ảnh vào layout editor.
- Sao chép các tệp hình ảnh vào thư mục drawable của dự án.
- Kéo một ImageButton vào layout editor.
- Chọn **New > Image Asset** và sau đó chọn tệp hình ảnh.

Câu hỏi 2

Làm thế nào để làm cho một ImageView có thể nhấp được như một nút đơn giản? Chọn một trong các đáp án sau:

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

Câu hỏi 3

Quy tắc nào áp dụng cho một bộ xử lý nhấp chuột được gọi từ thuộc tính trong bố cục? Chọn một trong các đáp án sau:

- Phương thức bộ xử lý nhấp chuột phải bao gồm event listener View.OnClickListener, đây là một giao diện trong lớp View.
- Phương thức bộ xử lý nhấp chuột phải là public, trả về void, và định nghĩa một View làm tham số duy nhất.
- Bộ xử lý nhấp chuột phải tùy chỉnh lớp View.OnClickListener và ghi đè bộ xử lý nhấp chuột của nó để thực hiện một hành động nào đó.
- Phương thức bộ xử lý nhấp chuột phải là private và trả về một View.

Nộp ứng dụng để chấm điểm

Hướng dẫn cho người chấm điểm

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

1.2) Các điều khiển nhập liệu

Giới thiệu

Để cho phép người dùng nhập văn bản hoặc số, bạn sử dụng phần tử [EditText](#). Một số điều khiển nhập liệu là các thuộc tính của EditText định nghĩa loại bàn phím sẽ hiển thị,

giúp việc nhập dữ liệu trở nên dễ dàng hơn cho người dùng. Ví dụ, bạn có thể chọn phone cho thuộc tính [android:inputType](#) để hiển thị bàn phím số thay vì bàn phím chữ cái và số.

Các điều khiển nhập liệu khác giúp người dùng dễ dàng đưa ra lựa chọn. Ví dụ, phần tử [RadioButton](#) cho phép người dùng chọn một (và chỉ một) mục trong một tập hợp các mục. Trong bài thực hành này, bạn sẽ sử dụng các thuộc tính để điều khiển giao diện bàn phím trên màn hình, và để đặt loại dữ liệu nhập vào cho EditText. Bạn cũng sẽ thêm các nút radio vào ứng dụng DroidCafe để người dùng có thể chọn một mục từ một tập hợp các mục.

Những gì bạn đã biết
Bạn khả năng nên làm

- Tạo một dự án Android Studio từ một mẫu và tạo bố cục chính.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị kết nối.
- Tạo và chỉnh sửa các phần tử giao diện người dùng (UI) bằng trình chỉnh sửa bố cục và mã XML.
- Truy cập các phần tử giao diện người dùng từ mã của bạn bằng cách sử dụng [findViewById\(\)](#).
- Chuyển văn bản trong một View thành một chuỗi bằng cách sử dụng [getText\(\)](#).
- Tạo một bộ xử lý nhấp chuột cho nút Button.
- Hiển thị một thông báo Toast.

Những gì bạn sẽ học

- Cách thay đổi phương thức nhập liệu để bật gợi ý, tự động viết hoa và che mặt khẩu.
- Cách thay đổi bàn phím trên màn hình chung thành bàn phím điện thoại hoặc các bàn phím chuyên dụng khác.
- Cách thêm các nút radio cho người dùng để chọn một mục trong một tập hợp các mục.
- Cách thêm một spinner để hiển thị một menu thả xuống với các giá trị, từ đó người dùng có thể chọn một giá trị.

Những gì bạn sẽ làm

- Hiển thị bàn phím để nhập địa chỉ email.
- Hiển thị bàn phím số để nhập số điện thoại.
- Cho phép nhập văn bản nhiều dòng với tự động viết hoa câu.
- Thêm các nút radio để chọn một tùy chọn.
- Đặt một bộ xử lý onClick cho các nút radio.

- Thêm một spinner cho trường số điện thoại để chọn một giá trị từ một tập hợp các giá trị.

Tổng quan về ứng dụng

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

Trong OrderActivity của ứng dụng, bạn sẽ thử nghiệm với thuộc tính [android:inputType](#) cho các phần tử [EditText](#). Bạn thêm các phần tử EditText để nhập tên và địa chỉ của người dùng, và sử dụng các thuộc tính để định nghĩa các phần tử một dòng và nhiều dòng có tính năng gợi ý khi bạn nhập văn bản. Bạn cũng thêm một EditText hiển thị bàn phím số để nhập số điện thoại.

Các loại điều khiển nhập liệu khác bao gồm các phần tử tương tác giúp người dùng đưa ra lựa chọn. Bạn thêm các nút radio vào DroidCafe để chọn chỉ một tùy chọn giao hàng từ nhiều tùy chọn. Bạn cũng cung cấp một điều khiển nhập liệu dạng spinner để chọn nhãn (**Home** , **Work** , **Other** , **Custom**) cho số điện thoại.

Task 1: Thử nghiệm với các thuộc tính nhập liệu văn bản

Khi chạm vào trường văn bản có thể chỉnh sửa [EditText](#), con trỏ sẽ xuất hiện trong trường văn bản và bàn phím trên màn hình sẽ tự động hiển thị để người dùng có thể nhập văn bản.

Một trường văn bản có thể chỉnh sửa kỳ vọng một loại văn bản nhập vào nhất định, chẳng hạn như văn bản thuần túy, địa chỉ e mail, số điện thoại hoặc mật khẩu. Việc chỉ định loại nhập liệu cho mỗi trường văn bản trong ứng dụng là rất quan trọng để hệ thống hiển thị phương pháp nhập liệu phù hợp, chẳng hạn như bàn phím trên màn hình cho văn bản thuần túy hoặc bàn phím số để nhập số điện thoại.

1.1 Thêm một EditText để nhập tên

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

1. Sao chép ứng dụng DroidCafe từ bài học về sử dụng hình ảnh có thể nhấp, và đổi tên bản sao thành **DroidCafeInput**. Nếu bạn chưa hoàn thành thử thách lập trình trong bài học đó, hãy tải dự án [DroidCafeChallenge](#) và đổi tên thành **DroidCafeInput**.

2. Mở tệp bố cục **activity_order.xml**, tệp này sử dụng ConstraintLayout.
3. Thêm một TextView vào ConstraintLayout trong activity_order.xml dưới phần tử order_textview đã có trong bố cục. Sử dụng các thuộc tính sau cho TextView mới.

TextView attribute	Value
android:id	"@+id/name_label"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	"24dp"
android:layout_marginLeft	"24dp"
android:layout_marginTop	"32dp"
android:text	"Name"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/order_textview"

4. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:text để tạo một mục mới có tên name_label_text trong strings.xml.
5. Thêm một phần tử EditText. Để sử dụng trình chỉnh sửa bố cục trực quan, kéo một phần tử Plain Text từ bảng **Palette** vào vị trí bên cạnh name_label TextView. Sau đó nhập name_text cho trường **ID** và ràng buộc cạnh trái và đường chéo của phần tử với cạnh phải và đường chéo của phần tử name_label như hình dưới đây.

6. Hình trên làm nổi bật trường **inputType** trong bảng **Attributes** để cho thấy Android Studio đã tự động chỉ định kiểu `textPersonName`. Nhấp vào trường **inputType** để xem menu các loại nhập liệu.

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


7. Thêm một gợi ý cho việc nhập văn bản, chẳng hạn như **Enter your name**, vào trường **hint** trong bảng **Attributes**, và xóa mục **Name** trong trường **text**. Dưới dạng một gợi ý cho người dùng, văn bản "Enter your name" sẽ mờ trong **EditText**.
8. Kiểm tra mã XML cho bố cục bằng cách nhấp vào tab **Text**. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính **android:hint** thành **enter_name_hint**. Các thuộc tính sau nên được thiết lập cho **EditText** mới (thêm thuộc tính **layout_marginLeft** để tương thích với các phiên bản Android cũ hơn): Như bạn có thể thấy trong mã XML, thuộc tính

EditText attribute	Value
android:id	"@+id/name_text"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	8dp
android:layout_marginLeft	8dp
android:ems	"10"
android:hint	"@string/enter_name_hint"
android:inputType	"textPersonName"
app:layout_constraintBaseline_toBaselineOf	"@+id/name_label"

app:layout_constraintStart_toEndOf	"@+id/name_label"
------------------------------------	-------------------

android:inputType được đặt thành **textPersonName**.

9. Chạy ứng dụng. Nhấn vào hình ảnh donut trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem **Activity** tiếp theo. Chạm vào trường nhập văn bản để hiển thị bàn phím và nhập văn bản, như hình dưới đây. Lưu ý rằng các gợi ý tự động xuất hiện cho các từ bạn nhập. Nhấn vào một gợi ý để sử dụng nó. Đây là một trong các tính năng của giá trị **textPersonName** cho thuộc tính **android:inputType**. Thuộc tính **inputType** điều khiển nhiều tính năng, bao gồm cách bố trí bàn phím, việc viết hoa và việc nhập văn bản nhiều dòng.

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

1.2 Thêm một EditText nhiều dòng

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

1. Mở tệp bố cục `activity_order.xml` nếu nó chưa được mở.

2. Thêm một `TextView` bên dưới phần tử `name_label` đã có trong bố cục. Sử dụng các thuộc tính sau cho `TextView` mới.

TextView attribute	Value
android:id	"@+id/address_label"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	"24dp"
android:layout_marginLeft	"24dp"

android:layout_marginTop	"24dp"
android:text	"Address"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/name_label"

3. Trích xuất giá trị thuộc tính android:text thành một tài nguyên chuỗi mới trong strings.xml với tên address_label_text.

4. Thêm một phần tử EditText.

Trong trình chỉnh sửa bố cục trực quan, kéo một phần tử Multiline Text từ Palette đến vị trí bên cạnh TextView có ID address_label. Đặt address_text làm giá trị cho ID của EditText. Thiết lập ràng buộc bên trái và đường cơ sở của nó với address_label như trong hình minh họa.

5. Thêm một gợi ý nhập văn bản, chẳng hạn như "Enter address", vào trường hint trong bảng thuộc tính Attributes. Gợi ý này sẽ hiển thị dưới dạng văn bản mờ bên trong EditText.

6. Kiểm tra mã XML của bố cục bằng cách nhấp vào tab Text. Trích xuất giá trị thuộc tính android:hint thành một tài nguyên chuỗi mới có tên enter_address_hint. Đảm bảo các thuộc tính sau được đặt cho EditText mới (thêm thuộc tính layout_marginLeft để tương thích với các phiên bản Android cũ hơn).


EditText attribute	Value
android:id	"@+id/address_text"
android:layout_width	"wrap_content"

android:layout_height	"wrap_content"
android:layout_marginStart	8dp
android:layout_marginLeft	8dp
android:ems	"10"
android:hint	"@string/enter_address_hint"
android:inputType	"textMultiLine"
app:layout_constraintBaseline_toBaselineOf	"@+id/address_label"
app:layout_constraintStart_toEndOf	"@+id/address_label"

7. Chạy ứng dụng.

Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào Floating Action Button để chuyển đến OrderActivity.

8. Nhấn vào trường nhập "Address" để hiển thị bàn phím và nhập văn bản.

Sử dụng phím Return  ở góc dưới bên phải của bàn phím (còn gọi là phím Enter hoặc New Line) để xuống dòng khi nhập văn bản.

Phím Return sẽ xuất hiện nếu bạn đặt thuộc tính android:inputType thành textMultiLine.

9. Để đóng bàn phím, nhấn vào nút mũi tên xuống xuất hiện thay vì nút Back trên hàng nút dưới cùng.

1.3 Sử dụng bàn phím số cho số điện thoại

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

1.Mở tệp activity_order.xml nếu nó chưa được mở.

2.Thêm một TextView bên dưới phần tử address_label đã có trong bố cục. Sử dụng các thuộc tính sau cho TextView mới:

TextView attribute	Value
android:id	"@+id/phone_label"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	"24dp"
android:layout_marginLeft	"24dp"
android:layout_marginTop	"24dp"
android:text	"Phone"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/address_text"

Lưu ý rằng TextView này được ràng buộc với đáy của EditText nhiều dòng (address_text). Điều này là do address_text có thể phát triển thành nhiều dòng, và TextView này sẽ xuất hiện bên dưới nó.

3.Trích xuất chuỗi tài nguyên cho giá trị thuộc tính android:text để tạo một mục nhập cho nó với tên phone_label_text trong tập strings.xml.

4.Thêm một phần tử EditText. Để sử dụng trình chỉnh sửa bố cục trực quan, kéo một phần tử Phone từ bảng Palette vào vị trí bên cạnh TextView có ID phone_label. Sau đó, đặt ID của nó là phone_text, và ràng buộc cạnh trái và dòng cơ sở của phần tử này với cạnh phải và dòng cơ sở của phone_label, như trong hình dưới đây:

5.Thêm một gợi ý nhập văn bản, chẳng hạn như Enter phone, trong trường hint trong bảng Attributes. Gợi ý "Enter phone" sẽ hiển thị dưới dạng văn bản mờ bên trong EditText.

6.Kiểm tra mã XML cho bố cục bằng cách nhấp vào tab Text. Trích xuất chuỗi tài nguyên cho giá trị thuộc tính android:hint thành enter_phone_hint. Các thuộc tính sau đây nên được đặt cho EditText mới (thêm thuộc tính layout_marginLeft để tương thích với các phiên bản Android cũ hơn):

EditText attribute	Value
android:id	"@+id/phone_text"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	8dp
android:layout_marginLeft	8dp
android:ems	"10"
android:hint	"@string/enter_phone_hint"
android:inputType	"phone"
app:layout_constraintBaseline_toBaselineOf	"@+id/phone_label"

app:layout_constraintStart_toEndOf	"@+id/phone_label"
------------------------------------	--------------------

7. Chạy ứng dụng. Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem Activity tiếp theo.

8. Nhấn vào trường "Phone" để hiển thị bàn phím số. Bạn có thể nhập số điện thoại, như hình dưới đây.

9. Để đóng bàn phím, nhấn vào phím Done.



Thử nghiệm với thuộc tính android:inputType

Hãy thay đổi giá trị thuộc tính android:inputType của một phần tử EditText để xem kết quả:

- `textEmailAddress`: Khi nhấn vào trường, bàn phím nhập email sẽ xuất hiện với ký hiệu @ nằm gần phím cách.
- `textPassword`: Các ký tự mà người dùng nhập sẽ biến thành dấu chấm để ẩn mật khẩu đã nhập.

1.4 Kết hợp nhiều kiểu nhập liệu trong một EditText

Bạn có thể kết hợp các giá trị thuộc tính inputType không xung đột với nhau. Ví dụ, bạn có thể kết hợp các giá trị **textMultiLine** và **textCapSentences** để tạo một ô nhập văn bản nhiều dòng, trong đó mỗi câu bắt đầu bằng một chữ cái viết hoa.

1. Mở tệp `activity_order.xml` nếu nó chưa được mở.

2. Thêm một TextView bên dưới phần tử `phone_label` đã có trong bố cục. Sử dụng các thuộc tính sau cho TextView mới.

TextView attribute	Value
--------------------	-------

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:text	"Note"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/phone_label"

3. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:text để tạo một mục nhập trong tệp strings.xml với tên note_label_text.

4. Thêm một phần tử EditText. Để sử dụng trình chỉnh sửa bố cục trực quan, kéo một phần tử "Multiline Text" từ bảng Palette đến vị trí bên cạnh phần tử note_label TextView. Sau đó, nhập note_text vào trường ID, và ràng buộc cạnh trái và đường cơ sở của phần tử này với cạnh phải và đường cơ sở của phần tử note_label như bạn đã làm trước đó với các phần tử EditText khác.

5. Thêm gợi ý nhập văn bản, chẳng hạn như "Enter note", trong trường hint trong bảng thuộc tính Attributes.

6. Nhấp vào bên trong trường inputType trong bảng thuộc tính Attributes. Giá trị textMultiLine đã được chọn sẵn. Ngoài ra, hãy chọn thêm textCapSentences để kết hợp hai thuộc tính này.

7. Kiểm tra mã XML của bố cục bằng cách nhấp vào tab Text. Trích xuất tài nguyên chuỗi cho giá trị thuộc tính android:hint với tên enter_note_hint. Các thuộc tính sau đây nên được thiết lập cho phần tử EditText mới (thêm thuộc tính layout_marginLeft để đảm bảo tương thích với các phiên bản Android cũ hơn):

EditText attribute	Value
android:id	"@+id/note_text"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	8dp
android:layout_marginLeft	8dp
android:ems	"10"
android:hint	"@string/enter_note_hint"
android:inputType	"textCapSentences textMultiLine"
app:layout_constraintBaseline_toBaselineOf	"@+id/note_label"
app:layout_constraintStart_toEndOf	"@+id/note_label"

Để kết hợp nhiều giá trị cho thuộc tính android:inputType, hãy nối chúng bằng ký tự dấu gạch đứng(|).

8. Chạy ứng dụng. Nhấn vào một hình ảnh trên màn hình đầu tiên, sau đó nhấn vào nút hành động nổi để xem Activity tiếp theo.

9. Nhấn vào ô nhập "Note" để nhập câu hoàn chỉnh, như hình minh họa bên dưới. Sử dụng phím Return để tạo một dòng mới hoặc chỉ cần nhập để tự động xuống dòng trong nhiều dòng.

Task 2: Sử dụng nút radio

Các điều khiển nhập liệu là các phần tử tương tác trong giao diện người dùng (UI) của ứng dụng, cho phép người dùng nhập dữ liệu. Nút radio là một loại điều khiển nhập liệu hữu ích khi bạn muốn người dùng chỉ chọn một tùy chọn từ một tập hợp các tùy chọn.

Mẹo: Bạn nên sử dụng nút radio nếu bạn muốn người dùng nhìn thấy tất cả các tùy chọn có sẵn bên cạnh nhau. Nếu không cần hiển thị tất cả các tùy chọn cùng lúc, bạn có thể sử dụng **Spinner** thay thế, tính năng này được mô tả trong một chương khác.

Trong nhiệm vụ này, bạn sẽ thêm một nhóm nút radio vào ứng dụng **DroidCafeInput** để thiết lập tùy chọn giao hàng cho đơn hàng tráng miệng. Để có cái nhìn tổng quan và xem thêm mã mẫu về nút radio, hãy xem **Radio Buttons**.

2.1 Thêm một RadioGroup và các nút radio

Để thêm các nút radio vào **OrderActivity** trong ứng dụng **DroidCafeInput**, bạn cần tạo các phần tử **RadioButton** trong tệp bố cục **activity_order.xml**. Sau khi chỉnh sửa tệp bố cục, bố cục cho các nút radio trong **OrderActivity** sẽ trông giống như hình minh họa dưới đây.

Vì các lựa chọn nút radio là loại lựa chọn loại trừ lẫn nhau, bạn sẽ nhóm chúng lại với nhau trong một **RadioGroup**. Bằng cách nhóm chúng lại, hệ thống Android đảm bảo rằng chỉ một nút radio có thể được chọn cùng lúc.

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

1. Mở **activity_order.xml** và thêm một phần tử **TextView** được ràng buộc ở dưới phần tử **note_text** đã có trong layout, và ràng buộc với lề trái, như trong hình dưới đây.

2. Chuyển sang chỉnh sửa XML, và đảm bảo rằng bạn đã thiết lập các thuộc tính sau cho TextView mới:

TextView attribute	Value
android:id	"@+id/delivery_label"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	"24dp"
android:layout_marginLeft	"24dp"
android:layout_marginTop	"24dp"
android:text	"Choose a delivery method: "
android:textSize	"18sp"
app:layout_constraintStart_toStartOf	"parent"
app:layout_constraintTop_toBottomOf	"@+id/note_text"

3. Trích xuất tài nguyên chuỗi cho "Choose a delivery method:" thành choose_delivery_method.
4. Để thêm các nút radio, hãy bao chúng trong một RadioGroup. Thêm RadioGroup vào layout dưới TextView bạn vừa thêm, bao ba phần tử [RadioButton](#) như trong mã XML dưới đây:

Thuộc tính android:onClick với giá trị "onRadioButtonClicked" của mỗi RadioButton sẽ được gạch dưới bằng màu đỏ cho đến khi bạn thêm phương thức đó trong bước tiếp theo của tác vụ này.

5. Trích xuất ba tài nguyên chuỗi cho các thuộc tính android:text thành các tên sau để các chuỗi có thể dễ dàng được dịch: same_day_messenger_service, next_day_ground_delivery, và pick_up.

2.2 Thêm phương thức xử lý sự kiện nhấn nút radio

Thuộc tính `android:onClick` cho mỗi phần tử nút radio chỉ định phương thức `onRadioButtonClicked()` để xử lý sự kiện nhấn nút. Vì vậy, bạn cần thêm một phương thức `onRadioButtonClicked()` mới trong lớp `OrderActivity`.

1. Mở tệp **activity_order.xml** (nếu chưa mở) và tìm một giá trị `onRadioButtonClicked` cho thuộc tính `android:onClick` bị gạch dưới màu đỏ.
2. Nhấp vào giá trị `onRadioButtonClicked`, sau đó nhấp vào biểu tượng cảnh báo bóng đỏ ở lề trái.
3. Chọn **Create onRadioButtonClicked(View) in OrderActivity** trong menu bóng đỏ. Android Studio sẽ tạo phương thức `onRadioButtonClicked(View view)` trong `OrderActivity`.

Ngoài ra, các giá trị `onRadioButtonClicked` cho các thuộc tính `android:onClick` khác trong `activity_order.xml` sẽ được giải quyết và không còn bị gạch dưới nữa.

4. Để hiển thị nút radio nào đã được nhấn (tức là loại giao hàng mà người dùng chọn), hãy sử dụng một thông báo Toast. Mở **OrderActivity** và thêm phương thức `displayToast` sau:
5. Trong phương thức `onRadioButtonClicked()` mới, thêm một khối switch case để kiểm tra nút radio nào đã được chọn và gọi `displayToast()` với thông điệp phù hợp. Mã sử dụng phương thức `isChecked()` của giao diện `Checkable`, trả về true nếu nút đã được chọn. Nó cũng sử dụng phương thức `getId()` của View để lấy định danh cho nút radio đã chọn.
6. Chạy ứng dụng. Nhấn vào một hình ảnh để xem hoạt động `OrderActivity`, hiển thị các lựa chọn giao hàng. Nhấn vào một lựa chọn giao hàng, và bạn sẽ thấy một thông báo Toast ở dưới màn hình với lựa chọn đó, như trong hình dưới đây.

Task 2 mã giải pháp

Dự án Android Studio: [DroidCafeInput](#)

Thử thách lập trình

Lưu ý: Tất cả các thử thách lập trình đều là tùy chọn và không phải là yêu cầu trước cho các bài học sau.

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

Gợi ý: Bạn có thể hoàn thành nhiệm vụ này hoàn toàn trong tệp layout. Một phương án thay thế là bạn có thể viết mã trong OrderActivity để chọn một trong các nút radio khi Activity xuất hiện lần đầu.

Mã giải pháp thử thách

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

Task 3: Sử dụng Spinner cho các lựa chọn của người dùng

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

Mẹo: Để biết thêm thông tin về Spinner, hãy tham khảo [Spinners](#).

Để cung cấp cách thức chọn nhãn cho số điện thoại (chẳng hạn như **Home**, **Work**, **Mobile**, hoặc **Other**), bạn có thể thêm một Spinner vào bố cục OrderActivity trong ứng dụng DroidCafe để nó xuất hiện ngay bên cạnh trường số điện thoại.

3.1 Thêm một spinner vào bố cục

Để thêm một Spinner vào bố cục OrderActivity trong ứng dụng DroidCafe, làm theo các bước dưới đây, được đánh số trong hình dưới:

1. Mở **activity_order.xml** và kéo **Spinner** từ bảng **Palette** vào bố cục.
2. Hạn chế phần trên của phần tử Spinner với dưới address_text, phần bên phải với phần bên phải của bố cục, và phần bên trái với phone_text.

Để căn chỉnh Spinner và phone_text theo chiều ngang, sử dụng nút pack trong thanh công cụ, cung cấp các tùy chọn để đóng gói hoặc mở rộng các phần tử UI đã chọn.

Chọn cả Spinner và phone_text trong **Component Tree**, nhấp vào nút pack, và chọn **Expand Horizontally**. Kết quả là, cả hai phần tử Spinner và phone_text sẽ có chiều rộng cố định.

3. Trong bảng Attributes, thiết lập ID của Spinner là **label_spinner**, và thiết lập các khoảng cách trên và bên phải là **24**, và khoảng cách bên trái là **8**. Chọn **match_constraint** cho menu thả xuống **layout_width**, và **wrap_content** cho menu thả xuống **layout_height**.

Bố cục sẽ trông giống như hình dưới đây. Menu thả xuống layout_width của phần tử phone_text trong bảng Attributes được thiết lập là 134dp. Bạn có thể thử nghiệm với các cài đặt chiều rộng khác nếu muốn.

Để xem mã XML của activity_order.xml, nhấp vào tab **Text**.

Spinner phải có các thuộc tính sau: Hãy chắc chắn thêm các thuộc tính android:layout_marginRight và android:layout_marginLeft như trong đoạn mã trên để duy trì tính tương thích với các phiên bản Android cũ hơn.

Phần tử phone_text hiện sẽ có các thuộc tính sau (sau khi sử dụng công cụ pack):

3.2 Thêm mã để kích hoạt Spinner và bộ lắng nghe của nó

Các lựa chọn cho [Spinner](#) là các chuỗi tĩnh được xác định rõ, chẳng hạn như "Home" và "Work", vì vậy bạn có thể sử dụng một mảng văn bản được định nghĩa trong **strings.xml** để lưu trữ các giá trị đó.

Để kích hoạt Spinner và trình nghe sự kiện của nó, hãy triển khai giao diện [AdapterView.OnItemSelectedListener](#), giao diện này yêu cầu bạn cũng phải thêm các phương thức gọi lại onItemSelected() và onNothingSelected().

1. Mở **strings.xml** và định nghĩa các giá trị có thể chọn (**Home**, **Work**, **Mobile** và **Other**) cho Spinner dưới dạng một mảng chuỗi labels_array.
2. Để định nghĩa callback xử lý lựa chọn cho Spinner, hãy thay đổi lớp **OrderActivity** để triển khai giao diện AdapterView.OnItemSelectedListener, như được minh họa:

Khi bạn nhập **AdapterView** trong câu lệnh trên, Android Studio sẽ tự động nhập tiện ích AdapterView. Lý do bạn cần AdapterView là vì bạn cần một adapter—cụ thể là [ArrayAdapter](#)—để gán mảng vào Spinner. Một adapter giúp kết nối dữ liệu của bạn—trong trường hợp này là mảng các mục của Spinner—với Spinner. Bạn sẽ tìm hiểu thêm về mô hình sử dụng adapter để kết nối dữ liệu trong một bài thực hành khác. Dòng này sẽ xuất hiện trong khối import của bạn:

Khi nhập **OnItemSelectedListener** trong câu lệnh trên, hãy đợi vài giây để một biểu tượng bóng đèn màu đỏ xuất hiện ở lề bên trái.

3. Nhấp vào biểu tượng bóng đèn và chọn **Implement methods**. Các phương thức `onItemSelected()` và `onNothingSelected()`, vốn là bắt buộc đối với `OnItemSelectedListener`, sẽ được làm nổi bật, và tùy chọn `Insert @Override` sẽ được chọn sẵn. Nhấp vào **OK**.

Bước này sẽ tự động thêm các phương thức callback `onItemSelected()` và `onNothingSelected()` trống vào cuối lớp `OrderActivity`. Cả hai phương thức này đều sử dụng tham số `AdapterView<?>*`. Dấu `*<?>` là một ký hiệu đại diện (wildcard) của Java, giúp phương thức có thể linh hoạt chấp nhận bất kỳ loại `AdapterView` nào làm đối số.

4. Khởi tạo một Spinner trong phương thức `onCreate()` bằng cách sử dụng phần tử `label_spinner` trong layout và đặt trình nghe sự kiện của nó bằng `spinner.setOnItemSelectedListener` trong `onCreate()`, như trong đoạn mã sau:
5. Tiếp tục chỉnh sửa phương thức `onCreate()`, thêm một câu lệnh để tạo `ArrayAdapter` với mảng chuỗi (`labels_array`) bằng cách sử dụng layout Spinner được cung cấp sẵn bởi Android cho từng mục (`layout.simple_spinner_item`):

Layout `simple_spinner_item` được sử dụng trong bước này và layout `simple_spinner_dropdown_item` được sử dụng trong bước tiếp theo là các layout mặc định do Android cung cấp trong lớp [R.layout](#). Bạn nên sử dụng các layout này trừ khi bạn muốn tự định nghĩa layout riêng cho các mục trong Spinner và giao diện của nó.

6. Xác định layout cho các lựa chọn của Spinner là `simple_spinner_dropdown_item`, sau đó áp dụng adapter vào Spinner.

3.3 Thêm một mã để phản hồi khi chọn mục trong Spinner

Khi người dùng chọn một mục trong Spinner, Spinner sẽ nhận được sự kiện on-item-selected.

Để xử lý sự kiện này, bạn đã triển khai giao diện AdapterView.OnItemSelectedListener ở bước trước, đồng thời thêm các phương thức callback onItemSelected() và onNothingSelected() trống.

Trong bước này, bạn sẽ điền mã vào phương thức onItemSelected() để lấy mục đã chọn trong Spinner bằng cách sử dụng getItemAtPosition(), sau đó gán mục đó vào biến spinnerLabel.

1. Thêm mã vào phương thức callback onItemSelected() trống, như minh họa bên dưới, để lấy mục đã chọn của người dùng bằng [getItemAtPosition\(\)](#) và gán nó vào spinnerLabel. Bạn cũng có thể gọi phương thức displayToast() mà bạn đã thêm vào OrderActivity:

Không cần thêm mã vào phương thức callback onNothingSelected() trong ví dụ này.

2. Chạy ứng dụng.

Spinner sẽ xuất hiện bên cạnh trường nhập số điện thoại và hiển thị lựa chọn đầu tiên (**Home**). Khi nhấn vào Spinner, tất cả các lựa chọn sẽ xuất hiện, như trong hình bên trái. Khi chọn một mục trong Spinner, một thông báo Toast sẽ hiển thị với lựa chọn đó, như trong hình bên phải.

Task 3 Mã giải pháp

Dự án Android Studio: [DroidCafeInput](#)

Thử thách lập trình 2

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

Thử thách: Viết mã để thực hiện một hành động trực tiếp từ bàn phím bằng cách nhấn phím **Send**, chẳng hạn như để quay số điện thoại:



Trong hình trên:

1. Nhập số điện thoại vào trường EditText.
2. Nhấn phím **Send** để mở trình quay số điện thoại. Trình quay số sẽ xuất hiện ở phía bên phải của hình minh họa.

Hướng dẫn thực hiện thử thách, tạo một dự án ứng dụng mới, thêm một EditText và thiết lập thuộc tính `android:inputType` thành `phone`. Sử dụng thuộc tính [android:imeOptions](#) cho phần tử EditText với giá trị `actionSend`:

Người dùng có thể nhấn phím **Send** để quay số điện thoại, như trong hình minh họa.

Trong phương thức `onCreate()` của Activity, sử dụng `setOnEditorActionListener()` để thiết lập trình nghe sự kiện cho EditText nhằm phát hiện khi phím được nhấn:

Để biết cách thiết lập trình nghe sự kiện, hãy xem phần Chỉ định loại phương thức nhập ([Specify the input method type](#)).

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

Cuối cùng, tạo phương thức `dialNumber()`, sử dụng implicit intent với `ACTION_DIAL` để chuyển số điện thoại sang một ứng dụng khác có thể quay số. Nó sẽ trông như sau:

Thử thách 2 mã giải pháp

Dự án Android Studio: [KeyboarDialPhone](#)

Tóm tắt

Các giá trị thuộc tính android:inputType ảnh hưởng đến giao diện bàn phím trên màn hình:

- textAutoCorrect: Gợi ý sửa lỗi chính tả.
- textCapSentences: Viết hoa chữ cái đầu tiên của mỗi câu mới.
- textPersonName: Hiển thị một dòng văn bản với gợi ý khi nhập và nút **Done** để hoàn tất.
- textMultiLine: Cho phép nhập nhiều dòng văn bản và có phím Return để xuống dòng.
- textPassword: Ẩn mật khẩu khi nhập.
- textEmailAddress: Hiển thị bàn phím nhập email thay vì bàn phím thông thường.
- phone: Hiển thị bàn phím số thay vì bàn phím thông thường.

Bạn thiết lập giá trị cho thuộc tính android:inputType trong tệp layout XML cho phần tử EditText.

Để kết hợp nhiều giá trị, hãy sử dụng ký tự gạch đứng (|).

RadioButton là điều khiển đầu vào hữu ích để chọn một tùy chọn duy nhất trong một tập hợp tùy chọn:

- Nhóm các phần tử [RadioButton](#) bên trong [RadioGroup](#) để đảm bảo chỉ một [RadioButton](#) được chọn cùng một lúc.
- Thứ tự liệt kê các RadioButton trong nhóm xác định thứ tự hiển thị trên màn hình.
- Sử dụng thuộc tính android:onClick cho từng RadioButton để chỉ định trình xử lý sự kiện khi nhấn.
- Để kiểm tra xem một nút có được chọn không, sử dụng phương thức [isChecked\(\)](#) của giao diện [Checkable](#), trả về true nếu nút được chọn.

Một [Spinner](#) (Menu thả xuống)

- Thêm [Spinner](#) vào layout.
- Sử dụng [ArrayAdapter](#) để gán một mảng văn bản làm các mục trong menu Spinner.
- Triển khai giao diện [AdapterView.OnItemSelectedListener](#), trong đó yêu cầu thêm các phương thức callback onItemSelected() và onNothingSelected() để kích hoạt **Spinner** và trình nghe sự kiện của nó.
- Sử dụng phương thức [onItemSelected\(\)](#) để lấy mục được chọn trong Spinner bằng [getItemAtPosition\(\)](#).

Khái niệm liên quan

Tài liệu về khái niệm liên quan có trong [4.2: Input controls](#).

Tìm hiểu thêm

Tài liệu hướng dẫn Android Studio:

- [Android Studio User Guide](#)

Tài liệu dành cho nhà phát triển Android:

- [Input events overview](#)
- [Specify the input method type](#)
- [Styles and themes](#)
- [Radio Buttons](#)
- [Spinners](#)
- [View](#)
- [Button](#)
- [EditText](#)
- [Android:inputType](#)
- [TextView](#)
- [RadioGroup](#)
- [Checkbox](#)
- [SeekBar](#)
- [ToggleButton](#)
- [Spinner](#)

Bài tập về nhà

Xây dựng và chạy một ứng dụng

1. Tạo một ứng dụng có năm Checkbox và một nút **Show Toast**, như hình minh họa.
2. Nếu người dùng chọn một Checkbox và nhấn **Show Toast**, hiển thị một thông báo **Toast** hiển thị nội dung của Checkbox đã chọn.

3. Nếu người dùng chọn nhiều hơn một Checkbox và nhấn **Show Toast**, hiển thị một Toast bao gồm nội dung của tất cả các Checkbox được chọn, như trong hình minh họa.

Trả lời các câu hỏi

Câu hỏi 1

Sự khác biệt quan trọng nhất giữa checkbox và một nhóm radio button (RadioGroup)?
Chọn một:

- "Sự khác biệt chính là checkbox cho phép chọn nhiều mục, trong khi RadioGroup chỉ cho phép chọn một mục."

Câu hỏi 2

Nhóm layout nào cho phép căn chỉnh các phần tử CheckBox theo chiều dọc?
Chọn một:

- LinearLayout

Câu hỏi 3

Phương thức nào của giao diện Checkable dùng để kiểm tra trạng thái của radio button (tức là kiểm tra xem nó đã được chọn hay chưa)?
Chọn một:

- isChecked()

Hướng dẫn chấm điểm ứng dụng

Kiểm tra xem ứng dụng có các tính năng sau không:

- Bố cục chứa năm CheckBox được căn chỉnh theo chiều dọc trên màn hình, cùng với một nút Show Toast.

- Phương thức `onSubmit()` xác định checkbox nào được chọn bằng cách sử dụng `findViewById()` kết hợp với `isChecked()`.
- Các chuỗi mô tả topping được nối lại thành một thông báo Toast.

1.3) Menu và bộ chọn

Giới thiệu

Thanh ứng dụng (App Bar), còn được gọi là thanh hành động (Action Bar), là một không gian chuyên dụng nằm ở phía trên cùng của mỗi màn hình **Activity**. Khi bạn tạo một Activity từ mẫu Basic Activity, Android Studio sẽ tự động bao gồm một thanh ứng dụng.

Menu tùy chọn (Options Menu) trong thanh ứng dụng thường cung cấp các lựa chọn điều hướng, chẳng hạn như chuyển đến một Activity khác trong ứng dụng. Menu cũng có thể cung cấp các tùy chọn ảnh hưởng đến cách sử dụng ứng dụng, ví dụ như thay đổi cài đặt hoặc thông tin hồ sơ, thường được thực hiện trong một Activity riêng biệt.

Trong bài thực hành này, bạn sẽ tìm hiểu về cách thiết lập App Bar và Options Menu trong ứng dụng của mình, như minh họa trong hình dưới đây.

Trong hình trên:

1. Thanh ứng dụng (**App bar**): Thanh ứng dụng bao gồm tiêu đề ứng dụng, menu tùy chọn và nút tràn (overflow button).
2. Biểu tượng hành động của menu tùy chọn (**Options menu action icons**): Hai mục menu tùy chọn đầu tiên xuất hiện dưới dạng biểu tượng trong thanh ứng dụng.
3. Nút tràn (**Overflow button**): Nút tràn (ba dấu chấm dọc) mở một menu hiển thị các mục menu tùy chọn bổ sung.
4. Menu tràn của các mục tùy chọn (**Options overflow menu**): Sau khi nhấp vào nút tràn, các mục menu tùy chọn bổ sung sẽ xuất hiện trong menu tràn.

Các mục menu tùy chọn xuất hiện trong menu tràn (xem hình trên). Tuy nhiên, bạn có thể đặt một số mục dưới dạng biểu tượng — càng nhiều càng tốt — trong thanh ứng dụng.

Việc sử dụng thanh ứng dụng cho menu tùy chọn giúp ứng dụng của bạn đồng nhất với các ứng dụng Android khác, cho phép người dùng dễ dàng hiểu cách sử dụng ứng dụng và có trải nghiệm tuyệt vời.

Mẹo: Để cung cấp trải nghiệm người dùng quen thuộc và đồng nhất, hãy sử dụng Menu APIs để trình bày các hành động của người dùng và các tùy chọn khác trong các activity của bạn. Xem [Menus](#) để biết chi tiết.

Bạn cũng sẽ tạo một ứng dụng hiển thị một dialog yêu cầu người dùng chọn lựa, ví dụ như một thông báo yêu cầu người dùng nhấn **OK** hoặc **Cancel**. Một dialog là một cửa sổ xuất hiện trên màn hình hoặc chiếm toàn bộ màn hình, làm gián đoạn dòng chảy hoạt động. Android cung cấp các dialog sẵn có, gọi là pickers, để chọn thời gian hoặc ngày. Bạn có thể sử dụng chúng để đảm bảo người dùng chọn đúng thời gian hoặc ngày được định dạng chính xác và điều chỉnh theo giờ và ngày địa phương của người dùng. Trong bài học này, bạn cũng sẽ tạo một ứng dụng với date picker.

Những gì bạn nên biết trước

Bạn nên có khả năng:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các phần tử giao diện người dùng (UI) bằng trình chỉnh sửa layout.
- Chỉnh sửa mã layout XML và truy cập các phần tử từ mã Java của bạn.
- Thêm trình xử lý sự kiện click cho Button.

Những gì bạn sẽ học

- Cách thêm các mục menu vào menu tùy chọn.
- Cách thêm biểu tượng cho các mục trong menu tùy chọn.
- Cách hiển thị các mục menu trong thanh ứng dụng.
- Cách thêm trình xử lý sự kiện cho các mục menu.
- Cách thêm một dialog để hiển thị thông báo.
- Cách thêm date picker.

Những gì bạn sẽ làm

- Tiếp tục thêm các tính năng cho dự án Droid Cafe từ bài thực hành trước.

- Thêm các mục menu vào menu tùy chọn.
- Thêm biểu tượng cho các mục menu để xuất hiện trong thanh ứng dụng.
- Kết nối các sự kiện click vào các trình xử lý sự kiện để xử lý sự kiện nhấn.
- Sử dụng alert dialog để yêu cầu người dùng chọn lựa.
- Sử dụng date picker để nhập ngày.

Tổng quan về ứng dụng

Trong bài thực hành trước, bạn đã tạo một ứng dụng có tên Droid Cafe, như hình dưới đây, sử dụng mẫu Basic Activity. Mẫu này cũng cung cấp một menu tùy chọn cơ bản trong thanh ứng dụng ở phía trên màn hình.

Trong bài tập này, bạn sẽ sử dụng thư viện hỗ trợ [v7 appcompat](#) với [Toolbar](#) như một thanh ứng dụng, hoạt động trên nhiều thiết bị và cũng cung cấp cho bạn không gian để tùy chỉnh thanh ứng dụng sau này khi ứng dụng của bạn phát triển. Để đọc thêm về các cân nhắc thiết kế khi sử dụng thanh ứng dụng, hãy xem [Responsive layout grid](#) trong thông số kỹ thuật Material Design.

Bạn sẽ tạo một ứng dụng mới hiển thị alert dialog. Dialog này gián đoạn quy trình làm việc của người dùng và yêu cầu người dùng đưa ra lựa chọn.

Bạn cũng sẽ tạo một ứng dụng cung cấp một Button để hiển thị date picker, và chuyển ngày đã chọn thành một chuỗi để hiển thị trong thông báo Toast.

Task 1: Thêm các mục vào menu tùy chọn

Trong nhiệm vụ này, bạn sẽ mở dự án [DroidCafeInput](#) từ bài thực hành trước và thêm các mục vào menu tùy chọn trong thanh ứng dụng ở phía trên màn hình.

1.1 Kiểm tra mã nguồn

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

- **activity_main.xml**: Layout chính cho MainActivity, màn hình đầu tiên mà người dùng thấy.

- **content_main.xml**: Layout cho nội dung của màn hình MainActivity, được bao gồm trong **activity_main.xml**.
- **activity_order.xml**: Layout cho OrderActivity, được thêm vào trong bài thực hành về việc sử dụng các điều khiển đầu vào.

Các bước thực hiện:

1. Mở **content_main.xml** và nhấp vào tab **Text** để xem mã XML. Thuộc tính `app:layout_behavior` của `ConstraintLayout` được thiết lập là `@string/appbar_scrolling_view_behavior`, điều khiển cách màn hình cuộn liên quan đến thanh ứng dụng ở phía trên. (Chuỗi tài nguyên này được định nghĩa trong tệp `values.xml` đã được tạo, và bạn không nên chỉnh sửa nó.)

Để biết thêm về hành vi cuộn, xem [Android Design Support Library](#) trong blog của Android Developers. Để biết về các thực hành thiết kế có liên quan đến menu cuộn, xem [Scrolling](#) in the Material Design specification.

2. Mở **activity_main.xml** và nhấp vào tab **Text** để xem mã XML cho layout chính, sử dụng layout [CoordinatorLayout](#) với layout [AppBarLayout](#) nhúng bên trong. Các thẻ `CoordinatorLayout` và `AppBarLayout` yêu cầu tên đầy đủ xác định `android.support.design`, là thư viện Hỗ trợ Thiết kế của Android. `AppBarLayout` giống như một `LinearLayout` theo chiều dọc. Nó sử dụng lớp `Toolbar` trong thư viện hỗ trợ, thay vì `ActionBar` gốc, để triển khai thanh ứng dụng. `Toolbar` trong layout này có id là `toolbar`, và cũng được chỉ định, giống như `AppBarLayout`, với tên đầy đủ (`android.support.v7.widget`).

Thanh ứng dụng là một phần của màn hình hiển thị, có thể hiển thị tiêu đề activity, điều hướng và các mục tương tác khác. `ActionBar` gốc hoạt động khác nhau tùy thuộc vào phiên bản Android chạy trên thiết bị. Vì lý do này, nếu bạn đang thêm menu tùy chọn, bạn nên sử dụng thư viện hỗ trợ [v7 appcompat](#) với [Toolbar](#) làm thanh ứng dụng. Việc sử dụng [Toolbar](#) giúp bạn dễ dàng thiết lập một thanh ứng dụng hoạt động trên nhiều thiết bị và cũng cung cấp không gian để tùy chỉnh thanh ứng dụng của bạn sau này khi ứng dụng phát triển. `Toolbar` bao gồm các tính năng mới nhất và hoạt động cho bất kỳ thiết bị nào có thể sử dụng thư viện hỗ trợ.

Layout **activity_main.xml** cũng sử dụng câu lệnh `layout include` để bao gồm toàn bộ layout được định nghĩa trong **content_main.xml**. Việc tách biệt các định nghĩa

layout giúp bạn dễ dàng thay đổi nội dung của layout riêng biệt với định nghĩa thanh công cụ và layout điều phối.

3. Chạy ứng dụng. Chú ý đến thanh ở phía trên màn hình hiển thị tên ứng dụng (Droid Cafe). Nó cũng hiển thị nút tràn hành động (ba dấu chấm dọc) ở phía bên phải. Nhấp vào nút tràn để xem menu tùy chọn, lúc này chỉ có một mục menu là **Settings**.
4. Kiểm tra tệp **AndroidManifest.xml**. Activity MainActivity được thiết lập sử dụng theme NoActionBar. Theme này được định nghĩa trong tệp styles.xml (mở **app > res > values > styles.xml** để xem). Các style được trình bày trong một bài học khác, nhưng bạn có thể thấy rằng theme NoActionBar thiết lập thuộc tính **windowActionBar** là **false** (không có thanh công cụ cửa sổ) và **windowNoTitle** là **true** (không có tiêu đề). Các giá trị này được thiết lập vì bạn đang định nghĩa thanh ứng dụng với AppBarLayout, thay vì sử dụng ActionBar. Việc sử dụng một trong các theme NoActionBar ngăn không cho ứng dụng sử dụng lớp ActionBar gốc để cung cấp thanh ứng dụng.
5. Nhìn vào **MainActivity**, kế thừa từ AppCompatActivity và bắt đầu với phương thức onCreate(), trong đó thiết lập view nội dung là layout activity_main.xml và thiết lập toolbar là Toolbar được định nghĩa trong layout. Sau đó, nó gọi [setSupportActionBar\(\)](#) và truyền toolbar vào, thiết lập toolbar là thanh ứng dụng cho Activity.

Để biết thêm về các phương pháp hay khi thêm thanh ứng dụng vào ứng dụng của bạn, hãy tham khảo [Add the app bar](#).

1.2 Thêm nhiều mục menu hơn vào menu tùy chọn

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

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

Android cung cấp một định dạng XML tiêu chuẩn để định nghĩa các mục menu. Thay vì tạo menu trong mã Activity, bạn có thể định nghĩa menu và tất cả các mục của nó

trong một tài nguyên menu XML. Sau đó, bạn có thể nạp (inflate) tài nguyên menu này để tải nó dưới dạng một đối tượng Menu trong Activity.

1. Mở rộng **res > menu** trong **Project > Android** và mở **menu_main.xml**. Mục menu duy nhất có sẵn từ mẫu là **action_settings** (tùy chọn **Settings**), được định nghĩa như sau:
2. Thay đổi các thuộc tính của mục **action_settings** để biến nó thành **action_contact** (không thay đổi thuộc tính **android:orderInCategory** hiện có).

Attribute	Value
android:id	"@+id/action_contact"
android:title	"Contact"
app:showAsAction	"never"

3. Trích xuất chuỗi cố định "Contact" thành tài nguyên chuỗi **action_contact**.
4. Thêm một mục menu mới bằng thẻ **<item>** trong khối **<menu>**, với các thuộc tính sau:

Attribute	Value
android:id	"@+id/action_order"
android:orderInCategory	"10"
android:title	"Order"
app:showAsAction	"never"

Thuộc tính **android:orderInCategory** xác định thứ tự xuất hiện của các mục trong menu, với số nhỏ hơn sẽ xuất hiện cao hơn trong menu. Mục **Contact** được đặt giá trị 100, một số lớn, để đảm bảo rằng nó hiển thị ở cuối danh sách thay vì ở đầu. Bạn đặt mục **Order** với giá trị 10, giúp nó hiển thị phía trên **Contact** và tạo đủ khoảng trống trong menu để có thể thêm nhiều mục khác sau này.

5. Trích xuất chuỗi cố định "Order" thành tài nguyên chuỗi **action_order**.
6. Thêm hai mục menu khác theo cách tương tự với các thuộc tính sau:

Status item attribute	Value
-----------------------	-------

android:id	"@+id/action_status"
android:orderInCategory	"20"
android:title	"Status"
app:showAsAction	"never"

Favorites item attribute	Value
android:id	"@+id/action_favorites"
android:orderInCategory	"30"
android:title	"Favorites"
app:showAsAction	"never"

- Trích xuất "Status" vào tài nguyên action_status và "Favorites" vào tài nguyên action_favorites.
- Hiển thị thông báo Toast với nội dung phù hợp khi người dùng chọn một mục menu. Mở **strings.xml** và thêm các tên và giá trị chuỗi cho các thông báo này.
- Mở **MainActivity**, thay đổi câu lệnh if trong phương thức onOptionsItemSelected(), thay thế ID action_settings bằng ID mới action_order.

Sau đó, chạy ứng dụng và nhấn vào biểu tượng menu (hình ba chấm dọc) trên thanh ứng dụng để xem menu tùy chọn.

Trong hình minh họa:

- Nhấn vào biểu tượng menu trên thanh ứng dụng để mở menu tùy chọn.
- Menu tùy chọn xuất hiện dưới dạng danh sách thả xuống.

Lưu ý thứ tự của các mục trong menu tùy chọn. Bạn đã sử dụng thuộc tính android:orderInCategory để xác định mức độ ưu tiên của các mục trong menu: Mục **Order** có giá trị (10), Tiếp theo là **Status** với giá trị (20) và **Favorites** với giá trị (30), và **Contact** có giá trị (100). Bảng sau hiển thị thứ tự ưu tiên của các mục trong menu:

Menu item	orderInCategory attribute
Order	10
Status	20
Favorites	30
Contact	100

Task 2: Thêm biểu tượng cho các mục menu




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

Trong ví dụ này, giả sử rằng các hành động **Order** và **Status** được sử dụng thường xuyên nhất. Hành động **Favorites** được sử dụng thỉnh thoảng, và **Contact** là ít được sử dụng nhất. Bạn có thể đặt biểu tượng cho các hành động này và chỉ định như sau:

- **Order** và **Status** luôn được hiển thị trên thanh ứng dụng.
- **Favorites** sẽ được hiển thị trên thanh ứng dụng nếu có đủ chỗ; nếu không, nó sẽ xuất hiện trong menu tràn.
- **Contact** sẽ không xuất hiện trên thanh ứng dụng; nó chỉ xuất hiện trong menu tràn.

2.1 Thêm biểu tượng cho các mục menu

Để chỉ định biểu tượng cho các hành động, trước tiên bạn cần thêm các biểu tượng dưới dạng tài sản hình ảnh vào thư mục **drawable**, sử dụng cùng một quy trình mà bạn đã thực hiện trong phần thực hành về sử dụng hình ảnh có thể nhấp. Bạn nên sử dụng các biểu tượng sau (hoặc các biểu tượng tương tự):

-  **Order:** Sử dụng cùng một biểu tượng mà bạn đã thêm cho nút hành động nổi trong phần thực hành về sử dụng hình ảnh có thể nhấp (ic_shopping_cart.png).
-  **Status:**
-  **Favorites:**
- **Contact:** Không cần biểu tượng vì nó chỉ xuất hiện trong menu tràn.

Đối với các biểu tượng **Status** và **Favorites**, hãy làm theo các bước sau:

1. Mở rộng **res** trong ngăn **Project > Android**, sau đó nhấp chuột phải (hoặc Control + click) vào thư mục **drawable**.
2. Chọn **New > Image Asset**. Hộp thoại Configure Image Asset sẽ xuất hiện.
3. Trong menu thả xuống, chọn **Action Bar and Tab Items**.
4. Đổi tên **ic_action_name** thành một tên khác (chẳng hạn như **ic_status_info** cho biểu tượng **Status**).
5. Nhấp vào hình ảnh clip art (biểu tượng Android bên cạnh **Clipart:**) để chọn một hình ảnh clip art làm biểu tượng. Một trang chứa các biểu tượng sẽ xuất hiện. Nhấp vào biểu tượng bạn muốn sử dụng.
6. Chọn **HOLO_DARK** từ menu thả xuống **Theme**. Điều này sẽ đặt biểu tượng có màu trắng trên nền tối (hoặc đen). Nhấp vào **Next**, sau đó nhấp vào **Finish**.

Mẹo: Xem [Create app icons with Image Asset Studio](#) để có mô tả đầy đủ.

2.2 Hiển thị các mục menu dưới dạng biểu tượng trên thanh ứng dụng

Để hiển thị các mục menu dưới dạng biểu tượng trên thanh ứng dụng, hãy sử dụng thuộc tính `app:showAsAction` trong `menu_main.xml`. Các giá trị sau của thuộc tính này xác định liệu hành động có xuất hiện trên thanh ứng dụng dưới dạng biểu tượng hay không:

- **"always"**: Luôn xuất hiện trên thanh ứng dụng. (Nếu không đủ chỗ, nó có thể chồng lên các biểu tượng menu khác.)
- **"ifRoom"**: Xuất hiện trên thanh ứng dụng nếu có đủ chỗ.
- **"never"**: Không bao giờ xuất hiện trên thanh ứng dụng; chỉ hiển thị dưới dạng văn bản trong menu tràn.

Thực hiện các bước sau để hiển thị một số mục menu dưới dạng biểu tượng

1. Mở lại `menu_main.xml`, sau đó thêm các thuộc tính sau cho các mục **Order**, **Status** và **Favorites** để đảm bảo rằng hai mục đầu tiên (**Order** và **Status**) luôn xuất hiện trên thanh ứng dụng, còn mục **Favorites** chỉ xuất hiện nếu có đủ chỗ.

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

Order item attribute	Old value	New value
android:icon	none	"@drawable/@drawable/ic_status_info"
app:showAsAction	"never"	"always"

Order item attribute	Old value	New value
android:icon	none	"@drawable/ic_favorite"
app:showAsAction	"never"	"always"

- Chạy ứng dụng. Lúc này, bạn sẽ thấy ít nhất hai biểu tượng trên thanh ứng dụng: biểu tượng cho **Order** và biểu tượng cho **Status**, như hiển thị ở phía bên trái của hình minh họa. (**Favorites** và **Contact** sẽ xuất hiện trong menu tràn).
- Xoay thiết bị sang chế độ ngang. Nếu bạn chạy trên trình giả lập, hãy nhấp vào biểu tượng **Rotate Left** hoặc **Rotate Right** để xoay màn hình. Khi đó, bạn sẽ thấy cả ba biểu tượng trên thanh ứng dụng cho **Order**, **Status** và **Favorites**, như hiển thị ở phía bên phải của hình minh họa.

Bao nhiêu nút hành động có thể hiển thị trên thanh ứng dụng? Điều này phụ thuộc vào hướng xoay và kích thước màn hình của thiết bị. Số lượng nút hiển thị ít hơn khi thiết bị ở chế độ dọc (như trong hình bên trái) so với chế độ ngang (như trong hình bên phải). Các nút hành động không được chiếm quá một nửa chiều rộng của thanh ứng dụng chính.

Task 3: Xử lý mục menu đã chọn
 Trong nhiệm vụ này, bạn sẽ thêm một phương thức để hiển thị thông báo về mục menu nào đã được nhấp, và sử dụng phương thức [onOptionsItemSelected\(\)](#) để xác định mục menu nào đã được chọn.

3.1 Tạo một phương thức để hiển thị lựa chọn menu

- Mở **MainActivity**.
- Nếu bạn chưa thêm phương thức sau (trong bài học khác) để hiển thị thông báo Toast, hãy thêm nó ngay bây giờ. Bạn sẽ sử dụng nó như hành động cho mỗi lựa

chọn menu. (Thông thường, bạn sẽ triển khai một hành động cho mỗi mục menu, chẳng hạn như bắt đầu một Activity khác, như đã trình bày trong bài học sau.)

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 một khối switch case để xác định mục menu nào đã được chọn và thực hiện hành động tương ứng.

1. Tìm phương thức `onOptionsItemSelected()` được cung cấp bởi mẫu (template). Phương thức này xác định xem một mục menu cụ thể có được nhấp vào hay không, sử dụng id của mục menu. Trong ví dụ dưới đây, id là `action_order`:
2. Thay thế câu lệnh gán int id và câu lệnh if bằng khối switch case sau, giúp thiết lập thông báo (message) phù hợp dựa trên id của mục menu:
3. Chạy ứng dụng. Bây giờ bạn sẽ thấy một thông báo Toast khác nhau trên màn hình, như hình bên phải trong hình minh họa, tùy thuộc vào mục menu bạn chọn.

Trong hình minh họa:

1. Chọn mục **Contact** trong menu tùy chọn.
2. Thông báo Toast xuất hiện.

3.3 Bắt đầu một Activity từ một mục menu

Thông thường, bạn sẽ triển khai một hành động cho từng mục menu, chẳng hạn như mở một Activity khác. Dựa trên đoạn mã từ nhiệm vụ trước, hãy thay đổi mã cho trường hợp `action_order` thành đoạn sau, để mở `OrderActivity` (sử dụng cùng một mã mà bạn đã dùng cho nút hành động nổi trong bài học về hình ảnh có thể nhấp):

Chạy ứng dụng. Khi nhấp vào biểu tượng giỏ hàng trên thanh ứng dụng (**mục Order**), ứng dụng sẽ chuyển bạn trực tiếp đến màn hình `OrderActivity`.

Task 3 Mã giải pháp

Dự án Android Studio: [DroidCafeOptions](#)

Thử thách lập trình

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

Thử thách : Menu ngữ cảnh cho phép người dùng thực hiện một hành động trên một View đã chọn. Trong khi menu tùy chọn trên thanh ứng dụng thường cung cấp lựa chọn để điều hướng đến một Activity khác, menu ngữ cảnh giúp người dùng chỉnh sửa một View trong Activity hiện tại.

Cả hai loại menu đều được mô tả trong XML, được nạp (inflate) bằng [MenuInflater](#), và sử dụng phương thức chọn mục menu (`on item selected`), trong trường hợp này là [onContextItemSelected\(\)](#). Vì vậy, cách xây dựng và sử dụng cả hai menu khá giống nhau.

Một menu ngữ cảnh xuất hiện dưới dạng danh sách thả nổi của các mục menu khi người dùng chạm và giữ vào một View, như minh họa ở phía bên trái của hình dưới đây. Trong thử thách này, hãy thêm một menu ngữ cảnh vào ứng dụng [ScrollingText](#) để hiển thị ba tùy chọn: **Edit**, **Share** và **Delete**. Menu sẽ xuất hiện khi người dùng chạm và giữ vào TextView. Sau khi chọn một mục trong menu, ứng dụng sẽ hiển thị một thông báo Toast với tùy chọn mà người dùng đã chọn, như minh họa ở phía bên phải của hình.

Gợi ý

Menu ngữ cảnh giống menu tùy chọn nhưng có hai điểm khác biệt quan trọng:

1. Menu ngữ cảnh phải được đăng ký (register) với một View để menu xuất hiện khi người dùng chạm và giữ vào View đó.
2. Menu ngữ cảnh không có sẵn trong mẫu Basic Activity, bạn cần tự thêm mã và tài nguyên menu.

Các bước thực hiện thử thách

1. Tạo một tệp tài nguyên menu XML cho menu ngữ cảnh

Nhấp chuột phải vào thư mục **res**, chọn **New > Android Resource Directory**. Trong **Resource type**, chọn **menu**, sau đó nhấp **OK**. Nhấp chuột phải vào thư mục **menu** mới, chọn **New > Menu resource file**. Đặt tên là **menu_context**, sau đó nhấp **OK**. Mở **menu_context.xml** và thêm các mục menu như bạn đã làm với menu tùy chọn.

2. Đăng ký View với menu ngữ cảnh bằng phương thức [registerForContextMenu\(\)](#). Trong phương thức onCreate(), đăng ký TextView với menu ngữ cảnh.
3. Triển khai phương thức [onCreateContextMenu\(\)](#)

Trong Activity, sử dụng phương thức này để nạp menu khi người dùng chạm và giữ vào View.

4. Triển khai phương thức [onContextItemSelected\(\)](#). Xử lý sự kiện khi người dùng chọn một mục trong menu. Hiển thị một Toast với tùy chọn mà người dùng đã chọn.
5. Chạy ứng dụng Nếu bạn chạm và kéo, văn bản vẫn cuộn bình thường. Nếu bạn chạm và giữ, menu ngữ cảnh sẽ xuất hiện.

Giải pháp cho thử thách

Dự án Android Studio: [ContextMenuScrollingText](#)

Task 4 : Sử dụng hộp thoại để yêu cầu lựa chọn từ người dùng

Bạn có thể hiển thị một hộp thoại (dialog) để yêu cầu người dùng đưa ra lựa chọn, chẳng hạn như một thông báo cảnh báo yêu cầu người dùng nhấn **OK** hoặc **Cancel**. Một hộp thoại là một cửa sổ xuất hiện trên màn hình hoặc có thể bao phủ toàn bộ màn hình, làm gián đoạn luồng hoạt động của ứng dụng.

Ví dụ: Một hộp thoại cảnh báo có thể yêu cầu người dùng nhấn **Continue** sau khi đọc thông báo. Người dùng có thể lựa chọn đồng ý với một hành động bằng cách nhấn nút **OK** hoặc **Accept**. Người dùng cũng có thể từ chối hành động bằng cách nhấn nút **Cancel**. Hãy sử dụng [AlertDialog](#), một lớp con của Dialog, để hiển thị hộp thoại cảnh báo tiêu chuẩn.

Mẹo: Hạn chế sử dụng hộp thoại quá nhiều vì chúng có thể làm gián đoạn trải nghiệm của người dùng. Để biết các nguyên tắc thiết kế tốt nhất, hãy xem hướng dẫn [Dialogs Material Design](#). Để xem các ví dụ về mã, hãy tham khảo tài liệu [Dialogs](#) trong Android Developer Documentation.

Trong bài thực hành này, bạn sẽ sử dụng một Button để kích hoạt một hộp thoại cảnh báo tiêu chuẩn. Trong một ứng dụng thực tế, hộp thoại cảnh báo có thể được kích hoạt dựa trên một điều kiện nào đó hoặc khi người dùng nhấn vào một phần tử trên màn hình.

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

Trong bài tập này, bạn sẽ tạo một hộp thoại cảnh báo có nút **OK** và **Cancel**. Hộp thoại sẽ được kích hoạt khi người dùng nhấn vào một nút.

Các bước thực hiện:

1. Tạo một dự án mới có tên **Dialog For Alert** dựa trên mẫu Empty Activity.
2. Mở tệp **activity_main.xml** để hiển thị trình chỉnh sửa giao diện (layout editor).
3. Chỉnh sửa phần tử TextView để hiển thị văn bản: Thay đổi từ "Hello World!" thành **"Hello World! Tap to test the alert:"**.
4. Thêm một nút (Button) bên dưới TextView. (Tùy chọn: Ràng buộc (constrain) nút vào phía dưới của TextView và hai bên của giao diện, đặt lề (margin) là 8dp.)
5. Đặt văn bản cho nút (Button) là **"Alert"**.
6. Chuyển sang tab **Text**, trích xuất các chuỗi văn bản của TextView và Button vào string resources.
7. Thêm thuộc tính android:onClick vào Button để gọi trình xử lý sự kiện onClickShowAlert(). Sau khi nhập phương thức này, nó sẽ bị gạch chân màu đỏ vì chưa được tạo.

Sau khi hoàn thành, bố cục của bạn sẽ trông giống như hình minh họa sau.

4.2 Thêm hộp thoại cảnh báo vào MainActivity

Mẫu thiết kế builder giúp dễ dàng tạo một đối tượng từ một lớp có nhiều thuộc tính bắt buộc và tùy chọn. Nếu không sử dụng mẫu này, bạn sẽ phải tạo nhiều hàm khởi tạo với các tổ hợp thuộc tính khác nhau. Với mẫu builder, mã nguồn sẽ dễ đọc và bảo trì hơn.

Trong Android, lớp AlertDialog.Builder được sử dụng để xây dựng một hộp thoại cảnh báo tiêu chuẩn, với các phương thức sau: [setTitle\(\)](#): Đặt tiêu đề cho hộp thoại, [setMessage\(\)](#): Đặt nội dung tin nhắn, [setPositiveButton\(\)](#): Tạo nút OK, [setNegativeButton\(\)](#): Tạo nút Cancel.

Hộp thoại cảnh báo sẽ chỉ được tạo khi người dùng nhấn nút Alert, tức là nó chỉ xuất hiện khi cần thiết. Tuy nhiên, trong một số ứng dụng khác, bạn có thể tạo hộp thoại trong phương thức onCreate() để có thể gọi từ nhiều nơi trong mã nguồn.

1. Mở MainActivity.java và thêm phương thức onClickShowAlert().

Nếu AlertDialog.Builder không được nhận diện, 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.

2. Thêm tiêu đề và nội dung cho hộp thoại bằng cách gọi setTitle() và setMessage() trong onClickShowAlert().

3. Trích xuất các chuỗi văn bản thành tài nguyên chuỗi (string resources), đặt tên là: alert_title: Tiêu đề của hộp thoại, alert_message: Nội dung của hộp thoại.

4. Thêm nút **OK** và **Cancel** vào hộp thoại bằng cách sử dụng các phương thức: setPositiveButton(): Xử lý khi người dùng nhấn OK, setNegativeButton(): Xử lý khi người dùng nhấn Cancel.

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

5. Trích xuất các chuỗi cho nút **OK** và **Cancel** thành các tài nguyên chuỗi với tên ok_button và cancel_button, và trích xuất các chuỗi cho thông báo Toast.

6. Cuối phương thức onClickShowAlert(), thêm show(), phương thức này sẽ tạo và hiển thị hộp thoại cảnh báo.

7. Chạy ứng dụng.
Bạn sẽ có thể nhấn nút **Alert**, hiển thị ở bên trái của hình dưới đây, để xem hộp thoại cảnh báo, hiển thị ở giữa hình dưới đây. Hộp thoại hiển thị nút **OK** và **Cancel**, và một thông báo Toast xuất hiện cho biết bạn đã nhấn nút nào, như hiển thị bên phải của hình dưới đây.

Task	4	Mã	giải	pháp
Dự án Android Studio: DialogForAlert				

Task 5: Sử dụng trình chọn cho đầu vào người dùng
Android cung cấp các hộp thoại sẵn có, gọi là **trình chọn**, cho việc chọn thời gian hoặc ngày. Bạn có thể sử dụng chúng để đảm bảo rằng người dùng chọn một thời gian hoặc ngày hợp lệ, được định dạng đúng và điều chỉnh theo thời gian và ngày địa phương của người dùng. Mỗi trình 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 tìm hiểu thêm về cách thiết lập các trình chọn trong phần [Pickers](#).

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

Một lợi ích của việc sử dụng Fragment cho một trình chọn là bạn có thể tách biệt các phần mã để quản lý ngày và giờ cho các địa phương khác nhau, nơi mà ngày và giờ được hiển thị theo nhiều cách khác nhau. Thực tiễn tốt nhất để hiển thị một trình chọn là sử dụng một thể hiện của [DialogFragment](#), là một lớp con của Fragment. DialogFragment hiển thị một cửa sổ hộp thoại nổi trên cửa sổ Activity. Trong bài tập này, bạn sẽ thêm một Fragment cho hộp thoại trình chọn và sử dụng DialogFragment để quản lý vòng đời của hộp thoại.

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

5.1 Tạo một ứng dụng mới để hiển thị trình chọn ngày

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

1. Tạo một dự án mới có tên **Picker For Date** dựa trên mẫu Empty Activity.
2. Mở tệp **activity_main.xml** để hiển thị trình chỉnh sửa bố cục.
3. Chỉnh sửa văn bản của phần tử TextView từ "Hello World!" thành **"Hello World! Choose a date:."**

4. Thêm một Button bên dưới TextView. (Tùy chọn: Ràng buộc Button ở dưới cùng của TextView và hai bên của bố cục, với khoảng cách được đặt là 8dp.)

5. Đặt văn bản của Button thành **"Date"**.

6. Chuyển sang tab **Text** và trích xuất các chuỗi cho TextView và Button thành các tài nguyên chuỗi.

7. Thêm thuộc tính android:onClick vào Button để gọi trình xử lý nhấp showDatePicker(). Sau khi nhấp, trình xử lý nhấp sẽ được gạch chân màu đỏ vì nó chưa được tạo.

Bạn nên có một bố cục tương tự như sau:

5.2 Tạo một fragment mới cho trình chọn ngày

Trong bước này, bạn sẽ thêm một Fragment cho trình chọn ngày.

1. Mở rộng **app > java > com.example.android.pickerfordate** và chọn **MainActivity**.

2. Chọn **File > New > Fragment > Fragment (Blank)**, và đặt tên fragment là **DatePickerFragment**. Bỏ chọn tất cả ba hộp kiểm để không tạo XML bố cục, không bao gồm các phương thức factory của fragment, hoặc không bao gồm các callback giao diện. Bạn không cần tạo bố cục cho một trình 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 một trình chọn ngày tiêu chuẩn với một listener. Xem [Pickers](#) để biết thêm thông tin về việc mở rộng DialogFragment cho một trình chọn ngày:

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

Thêm vào đó, một biểu tượng bóng đỏ xuất hiện ở lề trái sau vài giây.

4. Nhấp vào biểu tượng bóng đỏ và chọn **Implement methods** từ menu popup. Một hộp thoại xuất hiện với [onDateSet\(\)](#) đã được chọn và tùy chọn **Insert @Override** đã được chọn. Nhấp **OK** để tạo phương thức onDateSet() rỗng. Phương thức này sẽ được gọi khi người dùng thiết lập ngày.

Sau khi thêm phương thức `onDataSet()` rồi, Android Studio tự động thêm các thông tin sau vào khối `import` ở trên cùng:

Các tham số của `onDataSet()` nên là `int i`, `int i1`, và `int i2`. Thay đổi tên của các tham số này thành những tên dễ đọc hơn:

5.Xóa hàm khởi tạo công khai `public DatePickerFragment()` rồi.

6.Thay thế toàn bộ phương thức `onCreateView()` bằng [onCreateDialog\(\)](#) trả về `Dialog`, và chú thích `onCreateDialog()` bằng `@NonNull` để chỉ ra rằng giá trị trả về `Dialog` không thể là `null`. Android Studio hiển thị một biểu tượng bóng đỏ bên cạnh phương thức vì nó chưa trả về bất cứ thứ gì.

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

5.4 Chỉnh sửa hoạt động chính

Trong khi hầu hết mã trong `MainActivity.java` vẫn giữ nguyên, bạn cần thêm một phương thức để tạo một thể hiện của [FragmentManager](#) nhằm quản lý `Fragment` và hiển thị trình chọn ngày.

1.Mở **`MainActivity`**.

2.Thêm trình xử lý `showDatePickerDialog()` cho nút **`Date`**. Nó tạo một thể hiện của `FragmentManager` bằng cách sử dụng [getSupportFragmentManager\(\)](#) để tự động quản lý `Fragment` và hiển thị trình chọn. Để biết thêm thông tin về lớp `Fragment`, xem [Fragments](#).

3.Trích xuất chuỗi `"datePicker"` thành tài nguyên chuỗi `datepicker`.

4.Chạy ứng dụng. Bạn sẽ thấy trình chọn ngày sau khi nhấn nút **`Date`**.

5.5 Sử dụng ngày đã chọn

Trong bước này, bạn sẽ truyền ngày trở lại `MainActivity.java`, và chuyển đổi ngày thành chuỗi mà bạn có thể hiển thị trong thông báo `Toast`.

1.Mở **MainActivity** và thêm một phương thức rỗng `processDatePickerResult()` nhận các tham số `year`, `month`, và `day`:

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

Số nguyên `month` được trả về bởi trình chọn ngày bắt đầu đếm từ 0 cho tháng Giêng, vì vậy bạn cần cộng thêm 1 để hiển thị tháng bắt đầu từ 1.

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

4.Trích xuất chuỗi được mã hóa cứng `"Date: "` thành tài nguyên chuỗi có tên `date`.

5.Mở **DatePickerFragment**, và thêm mã sau vào phương thức `onDateSet()` để gọi `processDatePickerResult()` trong `MainActivity` và truyền cho nó `year`, `month`, và `day`:

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

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

Task 5 Mã giải pháp

Dự án Android Studio: [PickerForDate](#)

Thử thách lập trình 2

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

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

Gợi ý:

- Triển khai `TimePickerDialog.OnTimeSetListener` để tạo một trình chọn thời gian tiêu chuẩn với một listener.
- 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](#):
- Tạo một phương thức `processTimePickerResult()` tương tự như `processDatePickerResult()` trong nhiệm vụ trước để chuyển đổi các thành phần thời gian thành chuỗi và hiển thị kết quả trong một thông báo Toast.

Chạy ứng dụng và nhấp vào nút **Time** như hình bên trái của hình dưới đây. Trình chọn thời gian nên xuất hiện, như hình ở giữa. Chọn một thời gian và nhấp **OK**. Thời gian sẽ xuất hiện trong một thông báo Toast ở dưới cùng của màn hình, như hình bên phải.

Giải pháp thử thách 2

Dự án Android Studio: [PickerForTime](#)

Tóm tắt

Cung cấp một menu tùy chọn và thanh ứng dụng:

- Bắt đầu ứng dụng hoặc Activity của bạn với mẫu Basic Activity để tự động thiết lập thanh ứng dụng, menu tùy chọn và nút hành động nổi.
- Mẫu này thiết lập một bố cục [CoordinatorLayout](#) với một bố cục [AppBarLayout](#) nhúng. `AppBarLayout` giống như một `LinearLayout` theo chiều dọc. Nó sử dụng lớp [Toolbar](#) trong thư viện hỗ trợ, thay vì `ActionBar` gốc, để triển khai một thanh ứng dụng.
- Mẫu này sửa đổi tệp `AndroidManifest.xml` để `MainActivity` được thiết lập sử dụng chủ đề `NoActionBar`. Chủ đề này được định nghĩa trong tệp `styles.xml`.
- Mẫu này thiết lập `MainActivity` mở rộng `AppCompatActivity` và bắt đầu với phương thức `onCreate()`, phương thức này thiết lập nội dung và `Toolbar`. Sau đó, nó gọi

[setSupportActionBar\(\)](#) và truyền toolbar cho nó, thiết lập toolbar làm thanh ứng dụng cho Activity.

- Định nghĩa các mục menu trong tệp menu_main.xml. Thuộc tính android:orderInCategory xác định thứ tự mà các mục menu xuất hiện trong menu, với số thấp hơn xuất hiện cao hơn trong menu.
- Sử dụng phương thức [onOptionsItemSelected\(\)](#) để xác định mục menu nào được nhấp.

Thêm biểu tượng cho một mục menu tùy chọn:

- Mở rộng res trong ngăn **Project > Android**, và nhấp chuột phải (hoặc Control-click) vào thư mục **drawable**. Chọn **New > Image Asset**.
- Chọn **Action Bar and Tab Items** trong menu thả xuống, và thay đổi tên tệp hình ảnh.
- Nhấp vào hình ảnh clip art để chọn một hình ảnh clip art làm biểu tượng. Chọn một biểu tượng.
- Chọn **HOLO_DARK** từ menu thả xuống **Theme**.

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

Sử dụng thuộc tính app:showAsAction trong menu_main.xml với các giá trị sau:

- "always": Luôn xuất hiện trong thanh ứng dụng. (Nếu không có đủ chỗ, nó có thể chồng lên các biểu tượng menu khác.)
- "ifRoom": Xuất hiện trong thanh ứng dụng nếu có chỗ.
- "never": 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 một hộp thoại để yêu cầu lựa chọn của người dùng, chẳng hạn như một cảnh báo yêu cầu người dùng nhấp **OK** hoặc **Cancel**. Sử dụng hộp thoại một cách tiết kiệm vì chúng làm gián đoạn quy trình làm việc của người dùng.
- Sử dụng lớp con [AlertDialog](#) của lớp Dialog để hiển thị một hộp thoại tiêu chuẩn cho một cảnh báo.

- Sử dụng [AlertDialog.Builder](#) để xây dựng một hộp thoại cảnh báo tiêu chuẩn, với [setTitle\(\)](#) để thiết lập tiêu đề, [setMessage\(\)](#) để thiết lập thông điệp, và [setPositiveButton\(\)](#) và [setNegativeButton\(\)](#) để thiết lập các nút của nó.

Sử dụng trình chọn cho đầu vào của người dùng:

- Sử dụng [DialogFragment](#), là một lớp con của [Fragment](#), để xây dựng một trình chọn như trình chọn ngày hoặc trình chọn thời gian.
- Tạo một DialogFragment, và triển khai [DatePickerDialog.OnDateSetListener](#) để tạo một trình chọn ngày tiêu chuẩn với một listener. Bao gồm [onDateSet\(\)](#) trong Fragment này.
- Thay thế phương thức onCreateView() bằng [onCreateDialog\(\)](#) trả về Dialog. Khởi tạo ngày cho trình chọn ngày từ [Calendar](#), và trả về hộp thoại và các giá trị này cho Activity.
- Tạo một thể hiện của [FragmentManager](#) bằng cách sử dụng [getSupportFragmentManager\(\)](#) để quản lý Fragment và hiển thị trình chọn ngày.

Khái niệm liên quan

Tài liệu về khái niệm liên quan có trong mục [4.3: Menus and pickers](#).

Tìm hiểu thêm

Tài liệu Android Studio:

- [Android Studio User Guide](#)
- [Create App Icons with Image Asset Studio](#)

Tài liệu cho nhà phát triển Android:

- [Add the app bar](#)
- [Menus](#)
- [Toolbar](#)
- [v7 appcompat](#) support library
- [AppBarLayout](#)
- [onOptionsItemSelected\(\)](#)
- [View](#)

- [MenuInflater](#)
- [registerForContextMenu\(\)](#)
- [onCreateContextMenu\(\)](#)
- [onContextItemSelected\(\)](#)
- [Dialogs](#)
- [AlertDialog](#)
- [Pickers](#)
- [Fragments](#)
- [DialogFragment](#)
- [FragmentManager](#)
- [Calendar](#)

Thiết kế Material:

- [Responsive layout grid](#)
- [Dialogs](#)

Khác:

- Blog của các nhà phát triển Android: [Android Design Support Library](#)
- [Builder pattern](#) trên Wikipedia

Bài tập về nhà

Xây dựng và chạy một ứng dụng

Mở ứng dụng [DroidCafeOptions](#) mà bạn đã tạo trong bài học này.

1. Thêm một nút Date dưới các tùy chọn giao hàng để hiển thị trình chọn ngày.
2. Hiển thị ngày mà người dùng đã chọn trong một thông báo Toast.

Trả lời các câu hỏi

Câu hỏi 1

Tên 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 thức nào được gọi khi nhấp vào một mục menu tùy chọn? Chọn một:

- onOptionsItemSelected(MenuItem item)
- onClick(View view)
- onContextItemSelected()
- onClickShowAlert()

Câu hỏi 3

Trong số các tuyên bố dưới đây, tuyên bố nào thiết lập tiêu đề cho một hộp thoại cảnh báo? Chọn một:

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

Câu hỏi 4

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

- In the onCreate() method in the hosting Activity .
- In the onCreateContextMenu() method in Fragment .
- In the onCreateView() method in the extension of DialogFragment .
- In the onCreateDialog() method in the extension of DialogFragment .

Nội ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kiểm tra xem ứng dụng có các tính năng sau không:

- Trình chọn ngày được thêm dưới dạng DialogFragment.
- Nhấp vào nút **Date** (tham khảo bên trái của hình dưới đây) trong OrderActivity hiển thị trình chọn ngày (tham khảo ở giữa hình).
- Nhấp vào nút **OK** trong trình chọn ngày hiển thị một thông báo Toast trong OrderActivity với ngày đã chọn (tham khảo bên phải của hình).

1.4) Điều hướng người dùng

Giới thiệu

Trong giai đoạn đầu phát triển một ứng dụng, bạn nên xác định con đường mà bạn muốn người dùng thực hiện qua ứng dụng để hoàn thành từng nhiệm vụ. (Các nhiệm vụ là những việc như đặt hàng hoặc duyệt nội dung.) Mỗi con đường cho phép người dùng điều hướng qua, vào và ra khỏi các nhiệm vụ và các phần nội dung trong ứng dụng.

Trong phần thực hành này, bạn sẽ học cách thêm một 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 cha trong hệ thống phân cấp. Nó khác với nút Back (hình tam giác ở dưới cùng của màn hình), nút này cung cấp điều hướng đến bất kỳ màn hình nào mà người dùng đã xem lần cuối.

Phần thực hành này cũng giới thiệu điều hướng theo tab, trong đó các tab xuất hiện ở phía trên cùng của màn hình, cung cấp điều hướng đến các màn hình khác. Điều hướng theo tab là một cách phổ biến để tạo điều hướng ngang từ một màn hình con đến một màn hình con anh/em, như hình dưới đây.

Trong hình trên:

1. Điều hướng ngang từ một màn hình danh mục (**Top Stories**, **Tech News**, và **Cooking**) đến một màn hình khác.
2. Điều hướng ngang từ một màn hình câu chuyện (**Story**) đến một màn hình khác.

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

Các tab là phù hợp nhất cho bốn hoặc ít hơn các màn hình anh/em. Để xem một màn hình khác, người dùng có thể nhấn vào một tab hoặc vuốt trái hoặc phải.

Những gì bạn nên biết trước

Bạn nên có khả năng:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các phần tử giao diện người dùng 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 một ứng dụng với điều hướng theo tab và các chế độ xem vuốt.

Những gì bạn sẽ làm

- Tiếp tục thêm các tính năng vào dự án Droid Cafe từ phần thực hành trước.
- Cung cấp nút **Up** trong thanh ứng dụng để điều hướng lên Activity cha.
- Tạo một ứng dụng mới với các tab để điều hướng giữa các màn hình Activity mà cũng có thể được vuốt.

Tổng quan về ứng dụng

Trong phần thực hành trước về việc sử dụng menu tùy chọn, bạn đã làm việc trên một ứng dụng có tên Droid Cafe được tạo bằng mẫu Basic Activity. Mẫu này cung cấp một thanh ứng dụng ở phía trên cùng của màn hình. Bạn sẽ học cách thêm một nút Up (mũi tên hướng trái) vào thanh ứng dụng để điều hướng lên từ Activity thứ hai (OrderActivity) đến Activity cha (MainActivity). Điều này sẽ hoàn thành ứng dụng Droid Cafe.

Để bắt đầu dự án từ nơi bạn đã dừng lại trong phần thực hành trước, hãy tải xuống dự án Android Studio [DroidCafeOptions](#).

Bạn cũng sẽ tạo một ứng dụng cho điều hướng theo tab mà hiển thị ba tab bên dưới thanh ứng dụng để điều hướng đến các màn hình anh/em. Khi người dùng nhấn vào một tab, màn hình sẽ hiển thị một màn hình nội dung, tùy thuộc vào tab nào mà người dùng đã nhấn. Người dùng cũng có thể vuốt trái và phải để truy cập các màn hình nội dung. Lớp

ViewPager tự động xử lý các lần vuốt của người dùng đến các màn hình hoặc các phần tử View.

Task 1: Thêm nút Up cho điều hướng lên

Ứng dụng của bạn nên giúp người dùng dễ dàng tìm đường quay lại màn hình chính của ứng dụng, thường là Activity cha. Một cách để làm điều này là cung cấp một nút **Up** trong thanh ứng dụng cho mỗi Activity là con của Activity cha.

Nút **Up** cung cấp điều hướng “lên” tổ tiên, cho phép người dùng đi lên từ trang con đến trang cha. Nút **Up** là mũi tên hướng trái ở phía bên trái của thanh ứng dụng, như hình bên trái của hình dưới đây.

Khi người dùng chạm vào nút **Up**, ứng dụng sẽ điều hướng đến Activity cha. Sơ đồ ở bên phải của hình dưới đây cho thấy cách nút **Up** được sử dụng để điều hướng trong một ứng dụng dựa trên các mối quan hệ phân cấp giữa các màn hình.

Trong hình trên:

1. Điều hướng từ các màn hình anh/em cấp một lên màn hình cha.
2. Điều hướng từ các màn hình anh/em cấp hai đến màn hình con cấp một hoạt động như một màn hình cha.

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

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

Để cung cấp điều hướng từ một màn hình con quay lại màn hình cha, hãy sử dụng nút **Up**. Để biết thêm về điều hướng lên, xem [Providing Up navigation](#) .

Như bạn đã học trước đó, khi thêm các hoạt động vào một ứng dụng, bạn có thể thêm điều hướng nút **Up** đến một Activity con như OrderActivity bằng cách khai báo màn hình cha của Activity là MainActivity trong tệp AndroidManifest.xml. Bạn cũng có thể thiết lập thuộc tính android:label cho một tiêu đề cho màn hình Activity, chẳng hạn như "Order Activity". Thực hiện theo các bước sau:

1. Nếu bạn chưa mở ứng dụng Droid Cafe từ phần thực hành trước, hãy tải xuống dự án Android Studio [DroidCafeOptions](#) và mở dự án.
2. Mở tệp **AndroidManifest.xml** và thay đổi phần tử Activity cho OrderActivity thành sau:
3. Trích xuất giá trị android:label "Order Activity" thành tài nguyên chuỗi có tên `title_activity_order`.
4. Chạy ứng dụng.

Màn hình Order Activity hiện bao gồm nút **Up** (được làm nổi bật trong hình dưới đây) trong thanh ứng dụng để điều hướng quay lại Activity cha.

Task 1 Giải pháp mã

Dự án Android Studio: [DroidCafeOptionsUp](#)

Task 2: Sử dụng điều hướng theo tab với các chế độ xem vuốt

Với điều hướng ngang, bạn cho phép người dùng đi từ một anh/em này sang anh/em khác (ở cùng cấp trong một hệ thống phân cấp đa cấp). Ví dụ, nếu ứng dụng của bạn cung cấp nhiều danh mục câu chuyện (như **Top Stories**, **Tech News**, và **Cooking**, như hình dưới đây), bạn sẽ muốn cung cấp cho người dùng khả năng điều hướng từ danh mục này sang danh mục khác mà không phải quay lại màn hình cha. Một ví dụ khác về điều hướng ngang là khả năng vuốt trái hoặc phải trong một cuộc trò chuyện Gmail để xem một cuộc trò chuyện mới hơn hoặc cũ hơn trong cùng một Hộp thư đến.

Trong hình trên:

1. Điều hướng ngang từ một màn hình danh mục này sang một màn hình danh mục khác.
2. Điều hướng ngang từ một màn hình câu chuyện này sang một màn hình câu chuyện khác.

Bạn có thể triển khai điều hướng ngang với các tab đại diện cho mỗi màn hình. Các tab xuất hiện ở phía trên cùng của màn hình, như hình ở bên trái của hình trên, nhằm cung cấp điều hướng đến các màn hình khác. Điều hướng theo tab là một giải pháp rất phổ biến cho điều hướng ngang từ một màn hình con này sang một màn hình con khác là anh/em — ở cùng vị trí trong hệ thống phân cấp và chia sẻ cùng một màn hình cha. Điều hướng theo tab thường được kết hợp với khả năng vuốt các màn hình con sang trái và phải.

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 điền nội dung cho các "trang" màn hình bên trong một [ViewPager](#). ViewPager là một trình quản lý bố cục cho phép người dùng lật trái và phải qua các màn hình. Đây là một mẫu phổ biến để trình bày các màn hình nội dung khác nhau trong một Activity — sử dụng một adapter để điền nội dung màn hình để hiển thị trong Activity, và một trình quản lý bố cục thay đổi các màn hình nội dung tùy thuộc vào tab nào được chọn.

Bạn triển khai PagerAdapter để tạo ra các màn hình mà view hiển thị. ViewPager thường được sử dụng cùng với [Fragment](#). Bằng cách sử dụng Fragment, bạn có một cách tiện lợi 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:xx.xx.x` (trong đó `xx.xx.x` là phiên bản mới nhất) vào tệp **build.gradle (Module: app)**.

Dưới đây là các bộ điều hợp tiêu chuẩn để sử dụng các fragment với ViewPager:

- [FragmentPagerAdapter](#): Được thiết kế để điều hướng giữa các màn hình ảnh/em (trang) đại diện cho một số lượng màn hình cố định, nhỏ.
- [FragmentStatePagerAdapter](#): Được thiết kế để phân trang qua một bộ sưu tập các màn hình (trang) mà số lượng màn hình không xác định. Nó hủy bỏ mỗi Fragment khi người dùng điều hướng đến các màn hình khác, tối thiểu hóa việc sử dụng bộ nhớ. Ứng dụng cho nhiệm vụ này sử dụng [FragmentStatePagerAdapter](#).

2.1 Tạo dự án và bố cục

1. Tạo một dự án mới sử dụng mẫu Empty Activity. Đặt tên ứng dụng là **Tab Experiment**.
2. Chỉnh sửa tệp **build.gradle (Module: app)** và thêm dòng sau vào phần dependencies cho Thư viện Hỗ trợ Thiết kế Android, mà bạn cần để sử dụng [TabLayout](#).

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

3. Để sử dụng Toolbar thay vì thanh ứng dụng và tiêu đề ứng dụng, hãy thêm các thuộc tính sau vào tệp **res > values > styles.xml** để ẩn thanh ứng dụng và tiêu đề:
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** là 16dp cho **RelativeLayout**.
7. Xóa **TextView** được cung cấp bởi mẫu, và thêm một **Toolbar**, một **TabLayout**, và một **ViewPager** bên trong **RelativeLayout** như được hiển thị trong mã dưới đây.

Khi bạn nhập thuộc tính **app:popupTheme** cho **Toolbar**, app sẽ hiển thị màu đỏ nếu bạn chưa thêm câu lệnh sau vào **RelativeLayout**:

Bạn có thể nhấp vào app và nhấn **Option+Enter** (hoặc **Alt+Enter**), và **Android Studio** sẽ tự động thêm câu lệnh.

2.2 Tạo một lớp và bố cục cho mỗi fragment

Để thêm một fragment đại diện cho mỗi màn hình có tab, hãy thực hiện các bước sau:

1. Nhấp vào **com.example.android.tabexperiment** trong ngăn **Android > Project**.
2. Chọn **File > New > Fragment > Fragment (Blank)**.
3. Đặt tên fragment là **TabFragment1**.
4. Chọn tùy chọn **Create layout XML?**.
5. Thay đổi **Fragment Layout Name** cho tệp XML thành **tab_fragment1**.
6. Bỏ chọn các tùy chọn **Include fragment factory methods?** và **Include interface callbacks?**. Bạn không cần những phương thức này.
7. Nhấp **Finish**.

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

Mỗi fragment được tạo với định nghĩa lớp của nó được thiết lập để mở rộng **Fragment**. Đồng thời, mỗi **Fragment** sẽ dựng bố cục liên quan đến màn hình (**tab_fragment1**, **tab_fragment2** và **tab_fragment3**) bằng cách sử dụng mẫu thiết kế quen thuộc mà bạn đã học trong một chương trước với menu tùy chọn.

Ví dụ, **TabFragment1** trông như sau:

2.3 Chỉnh sửa bố cục fragment

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

1. Thay đổi `FrameLayout` thành **`RelativeLayout`**.
2. Thay đổi văn bản của `TextView` thành **"These are the top stories:"** và đặt `layout_width` và `layout_height` là **`wrap_content`**.
3. Thiết lập kiểu văn bản với `android:textAppearance="?android:attr/textAppearanceLarge"`.

Lặp lại các bước trên cho từng tệp bố cục XML của fragment, nhập văn bản khác cho `TextView` ở bước 2:

- Văn bản cho `TextView` trong `tab_fragment2.xml`: **"Tech news you can use:"**
- Văn bản cho `TextView` trong `tab_fragment3.xml`: **"Cooking tips:"**

Xem xét từng tệp bố cục XML của fragment. Ví dụ, **`tab_fragment1`** nên trông như sau:

4. Trong tệp bố cục XML của Fragment **`tab_fragment1`**, trích xuất chuỗi cho "These are the top stories:" thành tài nguyên chuỗi `tab_1`. Làm tương tự cho các chuỗi trong **`tab_fragment2`** và **`tab_fragment3`**.

2.3 Thêm một `PagerAdapter`

Mẫu thiết kế trình điều hợp - trình quản lý bố cục cho phép bạn cung cấp các màn hình nội dung khác nhau trong một Activity:

- Sử dụng một adapter để điền màn hình nội dung để hiển thị trong Activity.
- Sử dụng một layout manager để thay đổi các màn hình nội dung tùy thuộc vào tab nào được chọn.

Thực hiện theo các bước sau để thêm một lớp `PagerAdapter` mới vào ứng dụng, mở rộng [FragmentStatePagerAdapter](#) và xác định số lượng tab (`mNumOfTabs`):

1. Nhấp vào **`com.example.android.tabexperiment`** trong ngăn **Android > Project**.
2. Chọn **File > New > Java Class**.
3. Đặt tên lớp là **`PagerAdapter`**, và nhập **`FragmentStatePagerAdapter`** vào trường Superclass. Mục này sẽ chuyển thành `android.support.v4.app.FragmentStatePagerAdapter`.
4. Để tùy chọn **Public** và **None** được chọn, và nhấp **OK**.
5. Mở **`PagerAdapter`** trong ngăn **Project > Android**. Một bóng đèn đỏ sẽ xuất hiện bên cạnh định nghĩa lớp. Nhấp vào bóng đèn và chọn **Implement methods**, sau đó nhấp **OK** để triển khai các phương thức đã được chọn `getItem()` và `getCount()`.

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 kiểu nguyên mNumOfTabs, và thay đổi constructor để sử dụng nó. Mã sẽ trông như sau:

Khi bạn nhập mã trên, Android Studio sẽ tự động nhập các lớp sau:

Nếu FragmentManager trong mã hiển thị màu đỏ, một biểu tượng bóng đèn đỏ sẽ xuất hiện khi bạn nhấp vào nó. Nhấp vào biểu tượng bóng đèn và chọn Import class. Các lựa chọn nhập sẽ xuất hiện. Chọn FragmentManager (android.support.v4).

8. Thay đổi phương thức getItem() vừa được thêm vào như sau, sử dụng một khối switch case để trả về Fragment để hiển thị dựa trên tab nào được nhấp:
9. Thay đổi phương thức getCount() vừa được thêm vào như sau để trả về số lượng tab:

2.4 Dựng Toolbar và TabLayout

Bởi vì bạn đang sử dụng các tab nằm dưới thanh ứng dụng, bạn đã thiết lập thanh ứng dụng và Toolbar trong tệp bố cục activity_main.xml ở bước đầu tiên của nhiệm vụ này. Bây giờ bạn cần dựng Toolbar (sử dụng phương pháp tương tự được mô tả trong một chương trước về menu tùy chọn) và tạo một thể hiện của TabLayout để định vị các tab.

1. Mở **MainActivity** và thêm mã sau vào bên trong phương thức onCreate() để dựng Toolbar bằng cách sử dụng [setSupportActionBar\(\)](#):
2. Mở **strings.xml**, và tạo các tài nguyên chuỗi sau:
3. Ở cuối phương thức onCreate(), tạo một thể hiện của bố cục tab từ phần tử tab_layout trong bố cục, và thiết lập văn bản cho mỗi tab bằng cách sử dụng [addTab\(\)](#):

2.5 Sử dụng PagerAdapter để quản lý các chế độ xem màn hình

1. Dưới mã mà bạn đã thêm vào phương thức onCreate() trong nhiệm vụ trước, thêm mã sau để sử dụng PagerAdapter quản lý các chế độ xem màn hình (trang) trong các fragment:
2. Ở cuối phương thức onCreate(), thiết lập một listener ([TabLayoutOnPageChangeListener](#)) để phát hiện nếu một tab được nhấp, và tạo phương thức onTabSelected() để thiết lập ViewPager đến màn hình tab tương ứng. Mã sẽ trông như sau:

3. Chạy ứng dụng. Nhấn vào từng tab để xem từng “trang” (màn hình). Bạn cũng nên có thể vuốt trái và phải để truy cập các “trang” khác nhau.

Task 2 Mã giải pháp

Dự án Android Studio: [TabExperiment](#)

Thử thách lập trình

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

Thử thách: Tạo một ứng dụng với thanh điều hướng. Khi người dùng chạm vào lựa chọn trong thanh điều hướng, hãy đóng thanh điều hướng và hiển thị một thông báo Toast cho biết lựa chọn nào đã được chọn.

Thanh điều hướng là một bảng thường hiển thị các tùy chọn điều hướng ở cạnh trái của màn hình. Nó thường ẩn đi, nhưng sẽ được hiện ra khi người dùng vuốt từ cạnh trái màn hình hoặc chạm vào biểu tượng điều hướng trong thanh ứng dụng.

Trong hình trên:

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

Để tạo một thanh điều hướng trong ứng dụng của bạn, bạn cần tạo các bố cục sau:

- Một thanh điều hướng làm ViewGroup gốc của Activity.
- Một View điều hướng cho chính thanh điều hướng.
- Một bố cục thanh ứng dụng sẽ bao gồm nút biểu tượng điều hướng.
- Một bố cục nội dung cho Activity hiển thị thanh điều hướng.
- Một bố cục cho tiêu đề thanh điều hướng.

Sau khi tạo các bố cục, bạn cần:

- Điền các mục tiêu đề và biểu tượng vào menu thanh điều hướng.
- Thiết lập thanh điều hướng và các trình lắng nghe mục trong mã của Activity.
- Xử lý các lựa chọn mục menu trong thanh điều hướng.

Để tạo bố cục thanh điều hướng, hãy sử dụng các API [DrawerLayout](#) có sẵn [trong Support Library](#) . Để có các thông số thiết kế, hãy tuân theo các nguyên tắc thiết kế cho thanh điều hướng trong hướng dẫn thiết kế [Navigation Drawer](#) .

Để thêm một thanh điều hướng, hãy sử dụng DrawerLayout làm view gốc cho bố cục Activity của bạn. Bên trong DrawerLayout, thêm một View chứa nội dung chính của màn hình (bố cục chính của bạn khi thanh điều hướng bị ẩn) và một View khác, thường là [NavigationView](#), chứa nội dung của thanh điều hướng.

Mẹo: Để làm cho các 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 một biểu diễn trực quan của bố cục activity_main.xml và các bố cục XML mà nó bao gồm:

Trong hình trên:

1. [DrawerLayout](#) là view gốc của bố cục Activity.
2. Bố cục được bao gồm app_bar_main.xml sử dụng [CoordinatorLayout](#) làm gốc và định nghĩa bố cục thanh ứng dụng với [Toolbar](#), sẽ bao gồm biểu tượng điều hướng để mở thanh điều hướng.
3. [NavigationView](#) định nghĩa bố cục thanh điều hướng và tiêu đề của nó, đồng thời thêm các mục menu vào đó.

Mã giải pháp thử thách

Android Studio project: [NavDrawerExperiment](#)

Tóm tắt

Điều hướng thanh ứng dụng:

- Thêm điều hướng nút Up đến một Activity con bằng cách khai báo Activity cha trong tệp AndroidManifest.xml.
- Khai báo Activity con trong phần <activity ... </activity> của Activity cha.

Điều hướng tab:

- Các tab là một giải pháp tốt cho việc "điều hướng ngang" giữa các view anh em.
- Lớp chính được sử dụng cho các tab là [TabLayout](#) trong Thư viện Hỗ trợ Thiết kế Android.

- [ViewPager](#) là một trình quản lý bố cục cho phép người dùng lật qua trái và phải giữa các trang dữ liệu. ViewPager thường được sử dụng kết hợp với Fragment.
- Sử dụng một trong hai bộ điều hợp tiêu chuẩn cho việc sử dụng ViewPager: [FragmentPagerAdapter](#) hoặc [FragmentStatePagerAdapter](#).

Khái niệm liên quan

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

Tìm hiểu thêm

Tài liệu cho nhà phát triển Android:

User Interface & Navigation

- Designing effective navigation
- Implementing effective navigation
- Creating swipe views with tabs
- Create a navigation drawer
- Designing Back and Up navigation
- Providing Up navigation
- Implementing Descendant Navigation
- TabLayout
- Navigation Drawer
- DrawerLayout
- Support Library

Thông số của Material Design:

- Understanding navigation

- Responsive layout grid

Blog của các nhà phát triển Android: [Android Design Support Library](#)

Khác:

- AndroidHive: [Android Material Design working with Tabs](#)
- Triton: [Android Tabs Example – With Fragments and ViewPager](#)

Bài tập về nhà

Xây dựng và chạy một ứng dụng

Tạo một ứng dụng với một Activity chính và ít nhất ba Activity con khác. Mỗi Activity nên có một menu tùy chọn và sử dụng thư viện hỗ trợ [appcompat v7 Toolbar](#) làm thanh ứng dụng, như được hiển thị bên dưới.

1. Trong Activity chính, xây dựng một bố cục lưới với các hình ảnh mà bạn tự 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 các hình ảnh nếu cần, để ba hình ảnh có thể vừa ngang trên màn hình trong bố cục lưới.
3. Bật mỗi hình ảnh để cung cấp điều hướng đến một Activity con. Khi người dùng chạm vào hình ảnh, nó sẽ bắt đầu một Activity con. Từ mỗi Activity con, người dùng nên có thể chạm vào nút Lên trong thanh ứng dụng (được làm nổi bật trong hình dưới đây) để quay lại Activity chính.

Trả lời câu hỏi

Câu hỏi 1: Mẫu nào cung cấp một Activity với menu tùy chọn và Toolbar hỗ trợ thư viện v7 làm thanh ứng dụng?

- Mẫu Activity trống
- Mẫu Activity cơ bản
- Mẫu Activity thanh điều hướng
- Activity điều hướng dưới cùng

Câu hỏi 2: Bạn cần phụ thuộc nào để sử dụng [TabLayout](#)?

- com.android.support:design
- com.android.support.constraint:constraint-layout
- junit:junit:4.12
- com.android.support.test:runner

Câu hỏi 3: Đây là nơi bạn khai báo mỗi Activity con và Activity cha để cung cấp điều hướng Up?

- Để cung cấp nút **Up** cho một Activity con, khai báo Activity cha trong phần Activity con của tệp activity_main.xml.
- Để cung cấp nút **Up** cho một Activity con, khai báo Activity cha trong tệp XML bố cục "chính" cho Activity con.
- Để cung cấp nút **Up** cho một Activity con, khai báo Activity cha trong phần Activity con của tệp AndroidManifest.xml.
- Để cung cấp nút **Up** cho một Activity con, khai báo Activity cha trong phần Activity cha của tệp AndroidManifest.xml.

Nội ứng dụng của bạn để chấm điểm

Hướng dẫn cho các giám khảo:

Kiểm tra rằng ứng dụng có các tính năng sau:

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

1.5) RecyclerView

Giới thiệu

Cho phép người dùng hiển thị, cuộn và thao tác với một danh sách các mục dữ liệu tương tự là một tính năng phổ biến trong các ứng dụng. Các ví dụ về danh sách có thể cuộn bao

gồm danh sách liên lạc, danh sách phát, trò chơi đã lưu, thư mục ảnh, từ điển, danh sách mua sắm và chỉ mục tài liệu.

Trong thực hành về các view cuộn, bạn sử dụng `ScrollView` để cuộn một View hoặc `ViewGroup`. `ScrollView` dễ sử dụng, nhưng không được khuyến nghị cho các danh sách dài có thể cuộn.

[RecyclerView](#) là một lớp con của `ViewGroup` và là một cách hiệu quả hơn về tài nguyên để hiển thị các danh sách có thể cuộn. Thay vì tạo một View cho mỗi mục có thể hiển thị hoặc không hiển thị trên màn hình, `RecyclerView` tạo ra một số lượng mục danh sách hạn chế và tái sử dụng chúng cho nội dung hiển thị.

Trong thực hành này, bạn sẽ làm những điều sau:

- Sử dụng `RecyclerView` để hiển thị một danh sách có thể cuộn.
- Thêm một trình xử lý sự kiện click cho mỗi mục trong danh sách.
- Thêm các mục vào danh sách bằng cách sử dụng một nút hành động nổi (FAB), nút màu hồng trong ảnh chụp màn hình trong phần tổng quan ứng dụng. Sử dụng FAB cho hành động chính mà bạn muốn người dùng thực hiện.

Những gì bạn nên biết trước

Bạn nên có khả năng:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

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

Tổng quan ứng dụng

Ứng dụng RecyclerView minh họa cách sử dụng [RecyclerView](#) để hiển thị một danh sách dài các từ có thể cuộn. Bạn sẽ tạo tập dữ liệu (các từ), RecyclerView chính nó và các hành động mà người dùng có thể thực hiện:

- Chạm vào một từ sẽ đánh dấu nó là đã được chọn.
- Chạm vào nút hành động nổi (FAB) sẽ thêm một từ mới.

Task 1: Tạo một dự án và tập dữ liệu mới

Trước khi bạn có thể hiển thị một 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 (một tệp, cơ sở dữ liệu SQLite, sở thích đã lưu), từ một ứng dụng khác (Liên hệ, Ảnh) hoặc từ internet (lưu trữ đám mây, Google Sheets, hoặc bất kỳ nguồn dữ liệu nào có API). Việc lưu trữ và truy xuất dữ liệu là một chủ đề riêng biệt được đề cập trong chương về 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

Mở Android Studio.

1. Tạo một dự án mới với tên RecyclerView, chọn mẫu Basic Activity, và tạo tệp bố cục.
2. Mẫu Basic Activity, được giới thiệu trong chương về việc sử dụng hình ảnh có thể nhấp, cung cấp một nút hành động nổi (FAB) và thanh ứng dụng trong bố cục Activity (activity_main.xml), và một 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 phải lỗi liên quan đến Gradle, hãy đồng bộ hóa dự án của bạn như đã mô tả trong phần thực hành về việc cài đặt Android Studio và chạy Hello World.

1.2. Thêm mã để tạo dữ liệu

Trong bước này, bạn sẽ tạo một [LinkedList](#) gồm 20 chuỗi từ kết thúc bằng các số tăng dần, như trong ["Word 1", "Word 2", "Word 3", ...].

1. Mở **MainActivity** và thêm một biến thành viên riêng tư cho danh sách liên kết mWordList.
2. Thêm mã trong phương thức onCreate() để làm đầy mWordList bằng các từ:

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

1.3. Thay đổi biểu tượng FAB

Trong bài thực hành này, bạn sẽ sử dụng một FAB để tạo ra một từ mới để chèn vào danh sách. Mẫu Basic Activity đã 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 bộ biểu tượng có sẵn trong Android Studio cho FAB. Thực hiện theo các bước sau:

1. Mở rộng **res** trong bảng **Project > Android**, và nhấp chuột phải (hoặc Control-click) vào thư mục **drawable**.
2. Chọn **New > Image Asset**. Hộp thoại Configure Image Asset sẽ xuất hiện.
3. Chọn **Action Bar and Tab Items** trong menu thả xuống ở phía trên cùng của hộp thoại.
4. Thay đổi **ic_action_name** trong trường Name thành **ic_add_for_fab**.
5. Nhấp vào hình ảnh clip art (biểu tượng Android bên cạnh **Clipart:**) để chọn một hình ảnh clip art làm biểu tượng. Một trang các biểu tượng sẽ xuất hiện. Nhấp vào biểu tượng bạn muốn sử dụng cho FAB, chẳng hạn như biểu tượng dấu cộng (+).
6. Chọn **HOLO_DARK** từ menu thả xuống **Theme**. Điều này sẽ đặt biểu tượng thành màu trắng trên nền tối (hoặc đen). Nhấp vào **Next**.
7. Nhấp vào **Finish** trong hộp thoại Confirm Icon Path.

Mẹo: Để có mô tả đầy đủ về việc thêm một biểu tượng, hãy xem Tạo biểu tượng ứng dụng với Image Asset Studio.

Task 2: Tạo một RecyclerView

Trong thực hành này, bạn sẽ hiển thị dữ liệu trong một RecyclerView. Bạn cần các yếu tố sau:

- Dữ liệu để hiển thị: Sử dụng danh sách mWordList.
- Một RecyclerView cho danh sách cuộn chứa các mục danh sách.
- Giao diện cho một mục dữ liệu. Tất cả các mục danh sách đều giống nhau.
- Một layout manager. [RecyclerView.LayoutManager](#) xử lý cấu trúc và bố trí của các phần tử View. RecyclerView yêu cầu một layout manager rõ ràng để quản lý sự sắp xếp của các mục danh sách nằm trong đó. Bạn sẽ sử dụng [LinearLayoutManager](#) theo chiều dọc.
- Một adapter. [RecyclerView.Adapter](#) kết nối dữ liệu của bạn với RecyclerView. Nó chuẩn bị dữ liệu trong một [RecyclerView.ViewHolder](#). Bạn sẽ tạo một adapter để chèn vào và cập nhật các từ đã tạo ra trong các view của bạn.
- Một [ViewHolder](#). Bên trong adapter của bạn, bạn sẽ tạo một ViewHolder chứa thông tin View để hiển thị một mục từ giao diện của mục.

Sơ đồ dưới đây cho thấy mối quan hệ giữa dữ liệu, adapter, ViewHolder và layout manager.

Để thực hiện các phần này, bạn sẽ cần:

1. Thêm một phần tử RecyclerView vào layout XML của MainActivity (content_main.xml) cho ứng dụng RecyclerView.
2. Tạo một tệp giao diện XML (wordlist_item.xml) cho một mục danh sách, đó là WordListItem.
3. Tạo một adapter (WordListAdapter) với một ViewHolder (WordViewHolder). Thực hiện phương thức lấy dữ liệu, đặt nó vào ViewHolder và cho layout manager biết để hiển thị nó.
4. Trong phương thức onCreate() của MainActivity, tạo một RecyclerView và khởi tạo nó với adapter và một layout manager tiêu chuẩn.

Hãy làm từng bước một

2.1. Sửa đổi giao diện trong content_main.xml

Để thêm một phần tử RecyclerView vào giao diện XML, hãy làm theo các bước sau:

1. Mở tệp **content_main.xml** trong ứng dụng RecyclerView của bạn. Nó hiển thị một TextView với dòng chữ "Hello World" ở giữa ConstraintLayout.
2. Nhấp vào tab **Text** để hiển thị mã XML.
3. Thay thế toàn bộ phần tử TextView bằng đoạn mã sau:

Bạn cần chỉ định đường dẫn đầy đủ (android.support.v7.widget.RecyclerView) vì RecyclerView là một phần của Thư viện Hỗ trợ.

2.2. Tạo giao diện cho một mục danh sách

Adapter cần giao diện cho một mục trong danh sách. Tất cả các mục sử dụng cùng một giao diện. Bạn cần chỉ định giao diện mục danh sách đó trong một tệp tài nguyên giao diện riêng, vì nó được sử dụng bởi adapter, tách biệt với RecyclerView.

Tạo một giao diện mục từ đơn giản bằng cách sử dụng LinearLayout theo chiều dọc với một TextView:

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

LinearLayout attribute	Value
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:orientation	"vertical"
android:padding	"6dp"

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

Attribute	Value
android:id	"@+id/word"

android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:textSize	"24sp"
android:textStyle	"bold"

2.3 Tạo một kiểu từ các thuộc tính của 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 một phần tử UI mà bạn đã tạo. Để trích xuất thông tin kiểu cho TextView word trong wordlist_item.xml:

1. Mở **wordlist_item.xml** nếu nó chưa mở.
2. Nhấp chuột phải (hoặc Control-click) vào TextView mà bạn vừa tạo trong wordlist_item.xml, và chọn **Refactor > Extract > Style**. Hộp thoại **Extract Android Style** sẽ xuất hiện.
3. Đặt tên cho kiểu của bạn là **word_title** và giữ nguyên tất cả các tùy chọn khác được chọn. Chọn tùy chọn **Launch 'Use Style Where Possible'**. Sau đó nhấp **OK**.
4. Khi được nhắc, áp dụng kiểu cho **Whole Project**.
5. Tìm và xem xét kiểu word_title trong **values > styles.xml**.
6. Mở lại **wordlist_item.xml** nếu nó chưa mở. TextView bây giờ sử dụng kiểu thay vì các thuộc tính định dạng riêng lẻ, như hình dưới đây.

2.4. Tạo một adapter

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

Để kết nối dữ liệu với các mục View, adapter cần biết về các mục View. Adapter sử dụng một [ViewHolder](#) mô tả một mục View và vị trí của nó trong RecyclerView.

Trước tiên, bạn sẽ xây dựng một adapter kết nối dữ liệu trong danh sách từ của bạn với RecyclerView hiển thị nó:

1. Nhấp chuột phải vào **java/com.android.example.recyclerview** và chọn **New > Java Class**.
2. Đặt tên cho lớp là **WordListAdapter**.

3. Đặt cho WordListAdapter chữ ký sau:

WordListAdapter mở rộng một adapter tổng quát cho RecyclerView để sử dụng một ViewHolder cụ thể cho ứng dụng của bạn và được định nghĩa bên trong WordListAdapter. WordViewHolder sẽ báo lỗi, vì bạn chưa định nghĩa nó.

4. Nhấp vào khai báo lớp (**WordListAdapter**), sau đó nhấp vào bóng đèn đỏ bên trái của pane. Chọn Implement methods.

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

Android Studio tạo các placeholder trống cho tất cả các phương thức. Lưu ý rằng onCreateViewHolder và onBindViewHolder đều tham chiếu đến WordViewHolder, mà vẫn chưa được thực hiện.

2.5 Tạo ViewHolder cho adapter

Để tạo ViewHolder, hãy làm theo các bước sau:

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

Bạn sẽ thấy một lỗi về việc thiếu constructor mặc định. Bạn có thể xem chi tiết về các lỗi bằng cách di chuột qua mã được gạch chân đỏ hoặc qua bất kỳ đường ngang đỏ nào ở lề bên phải của pane biên tập.

2. Thêm các biến cho lớp con WordViewHolder cho TextView và adapter.

3. Trong lớp con WordViewHolder, thêm một constructor khởi tạo TextView của ViewHolder từ tài nguyên XML word, và thiết lập adapter của nó.

4. Chạy ứng dụng của bạn để đảm bảo rằng bạn không có lỗi. Bạn vẫn sẽ thấy chỉ một giao diện trắng.

5. Nhấp vào tab **Logcat** để xem pane **Logcat**, và lưu ý cảnh báo E/RecyclerView: No adapter attached; skipping layout. Bạn sẽ gắn adapter vào RecyclerView trong bước tiếp theo.

2.6 Lưu trữ dữ liệu của bạn trong adapter

Bạn cần lưu trữ dữ liệu của mình trong adapter, và WordListAdapter cần một constructor khởi tạo danh sách từ dữ liệu. Hãy làm theo các bước sau:

1. Để lưu trữ dữ liệu trong adapter, tạo một danh sách liên kết riêng tư của các chuỗi trong WordListAdapter và gọi nó là mWordList.
2. Bây giờ bạn có thể điền vào phương thức getItemCount() để trả về kích thước của mWordList:
3. WordListAdapter cần một constructor khởi tạo danh sách từ dữ liệu. Để tạo một View cho một mục trong danh sách, WordListAdapter cần nén tài nguyên XML cho một mục trong danh sách. Bạn sử dụng LayoutInflater cho công việc đó. Bắt đầu bằng cách tạo một biến thành viên cho inflater trong WordListAdapter:
4. Thực hiện constructor cho WordListAdapter. Constructor cần có một tham số context, và một danh sách liên kết các từ với dữ liệu của ứng dụng. Phương thức cần khởi tạo một LayoutInflater cho inflater và thiết lập mWordList với dữ liệu được truyền vào:
5. Điền vào phương thức onCreateViewHolder() với đoạn mã sau:

Phương thức onCreateViewHolder() tương tự như phương thức onCreate(). Nó nén giao diện mục và trả về một ViewHolder với giao diện và adapter.

6. Điền vào phương thức onBindViewHolder() với mã dưới đây:

Phương thức onBindViewHolder() kết nối dữ liệu của bạn với ViewHolder.

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

2.7 Tạo RecyclerView trong Activity

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

1. Mở **MainActivity**.
2. Thêm các biến thành viên cho RecyclerView và adapter.
3. Trong phương thức onCreate() của MainActivity, thêm đoạn mã sau để tạo RecyclerView và kết nối nó với một adapter và dữ liệu. Các chú thích giải thích từng dòng. Bạn phải chèn mã này sau khi khởi tạo mWordList.
4. Chạy ứng dụng của bạn.

Bạn sẽ thấy danh sách từ của mình được hiển thị, và bạn có thể cuộn danh sách.

Task 3: Làm cho danh sách tương tác

Nhìn vào danh sách các mục là điều thú vị, nhưng sẽ thú vị và hữu ích hơn nhiều nếu người dùng của bạn có thể tương tác với chúng. Để xem cách RecyclerView có thể phản hồi với đầu vào của người dùng, bạn sẽ gắn một trình xử lý sự kiện click cho mỗi mục. Khi mục đó được chạm vào, trình xử lý sự kiện click 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 thay đổi động—nó không nhất thiết phải là một danh sách tĩnh. Có nhiều cách để thêm các hành vi bổ sung. Một cách là sử dụng nút hành động nổi (FAB). Ví dụ, trong Gmail, FAB được sử dụng để soạn một email mới. Trong thực hành này, bạn sẽ tạo một từ mới để chèn vào danh sách. Đối với một ứng dụng hữu ích hơn, bạn sẽ lấy dữ liệu từ người dùng của bạn.

3.1. Làm cho các mục phản hồi với các cú click

1. Mở **WordListAdapter**.
2. Thay đổi chữ ký lớp ViewHolder để triển khai View.OnClickListener:
3. Nhấp vào tiêu đề lớp và trên bóng đèn đỏ để triển khai các phương thức stub cần thiết, trong trường hợp này chỉ là phương thức onClick().
4. Thêm đoạn mã sau vào thân phương thức onClick().
5. Kết nối onClickListener với View. Thêm đoạn mã này vào constructor của ViewHolder (dưới dòng this.mAdapter = adapter):
6. Chạy ứng dụng của bạn. Nhấp vào các mục để xem văn bản thay đổi.

3.2. Thêm hành vi cho FAB

Trong nhiệm vụ này, bạn sẽ triển khai một hành động cho FAB để:

- Thêm một từ vào cuối danh sách các từ.
- Thông báo cho adapter rằng dữ liệu đã thay đổi.
- Cuộn đến mục vừa được chèn.

Hãy làm theo các bước sau:

1. Mở **MainActivity**. Phương thức onCreate() thiết lập một OnClickListener() cho FloatingActionButton với một phương thức onClick() để thực hiện hành động. Thay đổi phương thức onClick() thành đoạn mã sau:
2. Chạy ứng dụng.
3. Cuộn danh sách các từ và nhấp vào các mục.
4. Thêm mục bằng cách nhấp vào FAB.

Điều gì sẽ xảy ra nếu bạn xoay màn hình? Bạn sẽ học trong một bài học sau về cách bảo tồn trạng thái của một ứng dụng khi màn hình được xoay.

Mã giải pháp

Dự án Android Studio: [RecyclerView](#)

Thách thức mã hóa

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

Thách thức 1: Thay đổi menu tùy chọn để chỉ hiển thị một tùy chọn: **Reset**. Tùy chọn này nên đưa danh sách các từ về trạng thái ban đầu, không có gì được nhấp và không có từ nào thêm.

Thách thức 2: Tạo một trình lắng nghe click cho mỗi mục trong danh sách là dễ, nhưng có thể làm giảm hiệu suất của ứng dụng bạn nếu bạn có nhiều dữ liệu. Nghiên cứu cách bạn có thể triển khai điều này hiệu quả hơn. Đây là một thách thức nâng cao. Bắt đầu bằng cách suy nghĩ về nó theo cách khái niệm, và sau đó tìm kiếm một ví dụ triển khai.

Tóm tắt

- [RecyclerView](#) là một cách hiệu quả về tài nguyên để hiển thị một danh sách cuộn các mục.
- Để tạo một View cho mỗi mục trong danh sách, adapter nén một tài nguyên XML cho một mục trong danh sách bằng cách sử dụng [LayoutInflater](#).
- [LinearLayoutManager](#) là một trình quản lý bố cục RecyclerView hiển thị các mục trong danh sách cuộn theo chiều dọc hoặc chiều ngang.
- [GridLayoutManager](#) là một trình quản lý bố cục RecyclerView hiển thị các mục trong một lưới.
- [StaggeredGridLayoutManager](#) là một trình quản lý bố cục RecyclerView hiển thị các mục trong một lưới gián đoạn.
- Sử dụng [RecyclerView.Adapter](#) để kết nối dữ liệu của bạn với RecyclerView. Nó chuẩn bị dữ liệu trong một RecyclerView.ViewHolder mô tả một mục View và vị trí của nó trong RecyclerView.
- Triển khai [View.OnClickListener](#) để phát hiện các cú nhấp chuột trong RecyclerView.

Khái niệm liên quan

Tài liệu khái niệm liên quan là [4.5: RecyclerView](#).

Tìm hiểu thêm

Tài liệu Android Studio:

- [Android Studio User Guide](#)
- [Create app icons with Image Asset Studio](#)

Tài liệu phát triển Android Studio

- [RecyclerView](#)
- [LayoutInflater](#)
- [RecyclerView.LayoutManager](#)
- [LinearLayoutManager](#)
- [GridLayoutManager](#)
- [StaggeredGridLayoutManager](#)
- [CoordinatorLayout](#)
- [ConstraintLayout](#)
- [RecyclerView.Adapter](#)
- [RecyclerView.ViewHolder](#)
- [View.OnClickListener](#)
- [Create a list with RecyclerView](#)

Video:

- [RecyclerView Animations and Behind the Scenes \(Android Dev Summit 2015\)](#)

Bài tập về nhà

Xây dựng và chạy một 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 công thức cùng với một mô tả ngắn.

Khi người dùng chạm vào một công thức (một mục trong danh sách), hãy khởi động một Activity hiển thị toàn bộ nội dung công thức.

- Sử dụng các phần tử TextView và kiểu dáng riêng cho tên công thức và mô tả.
- Bạn có thể sử dụng văn bản giữ chỗ cho các công thức đầy đủ.
- Tùy chọn: Thêm một hình ảnh cho món ăn đã hoàn thành vào mỗi công thức.
- Nhấp vào nút **Up** sẽ đưa người dùng quay lại danh sách các công thức.

Ảnh chụp màn hình dưới đây cho thấy một ví dụ về một triển khai đơn giản. Ứng dụng của bạn có thể trông rất khác, miễn là nó có chức năng cần thiết.

Trả lời các câu hỏi

Câu hỏi 1

Trong số các phát biểu sau về RecyclerView, phát biểu nào là sai? Chọn một.

- RecyclerView là một cách hiệu quả về tài nguyên để hiển thị các danh sách cuộn.
- Bạn chỉ cần cung cấp một bố cục cho một mục trong danh sách.
- Tất cả các mục trong danh sách đều giống nhau.
- Bạn không cần một trình quản lý bố cục với RecyclerView để xử lý cấu trúc và bố cục của các phần tử View.

Câu hỏi 2

Trong số các lựa chọn sau, thành phần chính nào bạn cần cung cấp cho một adapter để mô tả một mục View và vị trí của nó trong RecyclerView? Chọn một.

- RecyclerView
- RecyclerView.Adapter
- RecyclerView.ViewHolder
- AppCompatActivity

Câu hỏi 3

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

- View.OnClickListener
- RecyclerView.Adapter
- RecyclerView.ViewHolder
- View.OnKeyListener

Nội ứng dụng của bạn để chấm điểm

Hướng dẫn cho người chấm điểm

Kiểm tra rằng ứng dụng có các tính năng sau:

- Triển khai một RecyclerView hiển thị danh sách cuộn các tiêu đề công thức và mô tả ngắn.
- Mã mở rộng hoặc triển khai RecyclerView, RecyclerView.Adapter, RecyclerView.ViewHolder, và View.OnClickListener.
- Nhấp vào một mục trong danh sách khởi động một Activity hiển thị toàn bộ công thức.
- Tập AndroidManifest.xml định nghĩa một mối quan hệ cha để khi nhấp vào nút Up trong chế độ xem công thức quay lại danh sách các công thức.
- ViewHolder chứa một bố cục với hai phần tử TextView; ví dụ, một LinearLayout với hai phần tử TextView.

Bài 2) rải nghiệm người dùng thú vị

2.1) Hình vẽ, định kiểu và chủ đề

Giới thiệu

Trong chương này, bạn sẽ học cách áp dụng các kiểu chung cho các view của bạn, sử dụng tài nguyên drawable, và áp dụng các chủ đề cho ứng dụng. Những thực hành này giúp giảm mã của bạn và làm cho mã dễ đọc và bảo trì hơn.

Những gì bạn nên biết trước

Bạn nên có khả năng:

1. Tạo và chạy ứng dụng trong Android Studio.
2. Tạo và chỉnh sửa các phần tử UI bằng cách sử dụng trình chỉnh sửa bố cục.

3. Chỉnh sửa mã bố cục XML và truy cập các phần tử từ mã Java của bạn.
4. Trích xuất các chuỗi mã cứng thành tài nguyên chuỗi.
5. Trích xuất các kích thước mã cứng thành tài nguyên kích thước.
6. Thêm mục menu và biểu tượng vào menu tùy chọn trong thanh ứng dụng.
7. Tạo một trình xử lý click cho một lần nhấn Button.
8. Hiển thị một thông báo Toast.
9. Truyền dữ liệu từ một [Activity](#) này sang Activity khác bằng cách sử dụng [Intent](#).

Những gì bạn sẽ học

Bạn sẽ học cách:

- Định nghĩa một tài nguyên kiểu.
- Áp dụng một kiểu cho một [View](#).
- Áp dụng một chủ đề cho một Activity hoặc ứng dụng trong XML và lập trình.
- Sử dụng các tài nguyên [Drawable](#).

Những gì bạn sẽ làm

- Tạo một ứng dụng mới và thêm các phần tử Button và TextView vào bố cục.
- Tạo các tài nguyên Drawable trong XML và sử dụng chúng làm nền cho các phần tử Button.
- Áp dụng các kiểu cho các phần tử UI.
- Thêm một mục menu thay đổi chủ đề của ứng dụng sang “chế độ ban đêm” với độ tương phản thấp.

Tổng quan về ứng dụng

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

Task 1: Tạo ứng dụng Scorekeeper

Trong phần này, bạn sẽ tạo dự án Android Studio của mình, chỉnh sửa bố cục và thêm thuộc tính android:onClick vào các phần tử Button.

1.1 Tạo dự án

1. Mở Android Studio và tạo một dự án Android Studio mới với tên **Scorekeeper**.
2. Chấp nhận SDK tối thiểu mặc định và chọn mẫu **Empty Activity**.

3. Chấp nhận tên Activity mặc định (MainActivity), và đảm bảo rằng các tùy chọn **Generate Layout file** và **Backwards Compatibility (AppCompat)** được đánh dấu.
4. Nhấp vào **Finish**.

1.2 Tạo bố cục cho MainActivity

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

1. Mở **activity_main.xml** và nhấp vào tab **Text** để xem mã XML. Ở trên cùng, hoặc gốc, của cấu trúc View là [ConstraintLayout](#) ViewGroup.
2. Thay đổi ViewGroup này thành [LinearLayout](#). Dòng mã thứ hai bây giờ sẽ trông giống như sau:
3. Xóa dòng mã XML sau, liên quan đến ConstraintLayout:
Khối mã XML ở trên cùng bây giờ nên trông như thế này:
4. Thêm các thuộc tính sau (mà không xóa bỏ các thuộc tính hiện có):

Attribute	Value
android:orientation	"vertical"
android:padding	"16dp"

1.3 Tạo các vùng chứa điểm số

Bố cục cho ứng dụng này bao gồm các vùng chứa điểm số được định nghĩa bởi hai phần tử RelativeLayout—một cho mỗi đội. Sơ đồ sau đây cho thấy bố cục ở dạng đơn giản nhất.

1. Trong LinearLayout, thêm hai phần tử RelativeLayout với các thuộc tính sau:

RelativeLayout attribute	Value
android:layout_width	"match_parent"
android:layout_height	"0dp"
android:layout_weight	"1"

Chúng ta sử dụng thuộc tính layout_weight để xác định mức độ không gian mà các phần tử RelativeLayout này chiếm trong [LinearLayout](#) cha.

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

ImageButton attribute	Value
android:id	"@+id/decreaseTeam1"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_alignParentLeft	"true"
android:layout_alignParentStart	"true"
android:layout_centerVertical	"true"

Sử dụng increaseTeam1 làm android:id cho phần tử ImageButton thứ hai để tăng điểm số. Sử dụng decreaseTeam2 và increaseTeam2 cho phần tử ImageButton thứ ba và thứ tư.

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

TextView attribute	Value
android:id	"@+id/score_1"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_centerHorizontal	"true"
android:layout_centerVertical	"true"
android:text	"0"

Sử dụng score_2 làm android:id cho TextView thứ hai giữa các phần tử ImageButton.

4. Thêm một TextView khác vào mỗi RelativeLayout phía trên điểm số để đại diện cho tên đội. Sử dụng các thuộc tính sau:

TextView attribute	Value
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_alignParentTop	"true"
android:layout_centerHorizontal	"true"
android:text	"Team 1"

Sử dụng "Team 2" làm android:text cho TextView thứ hai.

1.4 Thêm tài nguyên vector

Bạn sẽ sử dụng các biểu tượng vật liệu trong Vector Asset Studio cho các phần tử ImageButton dùng để ghi điểm.

1. Chọn **File > New > Vector Asset** để mở Vector Asset Studio.
2. Nhấp vào biểu tượng để mở danh sách các tệp biểu tượng vật liệu. Chọn danh mục **Content**.
3. Chọn biểu tượng **add** và nhấp **OK**.
4. Đổi tên tệp tài nguyên thành **ic_plus** và đánh dấu ô **Override** bên cạnh tùy chọn kích thước.
5. Thay đổi kích thước của biểu tượng thành **40dp x 40dp**.
6. Nhấp **Next** và sau đó **Finish**.
7. Lặp lại quá trình này để thêm biểu tượng **remove** và đặt tên tệp là **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 cần các tính năng truy cập Android như trình đọc màn hình [Google TalkBack](#).

8. Thêm các thuộc tính sau vào các phần tử ImageButton ở phía bên trái của bố cục:
9. Thêm các thuộc tính sau vào các phần tử ImageButton ở phía bên phải của bố cục:
10. Trích xuất tất cả các tài nguyên chuỗi của bạn. Quá trình này sẽ loại bỏ tất cả các chuỗi khỏi mã Java và đưa chúng vào tệp strings.xml. Điều này cho phép ứng dụng của bạn dễ dàng được địa phương hóa sang các ngôn ngữ khác nhau.

Mã giải pháp cho bố cục

Dưới đây là bố cục cho ứng dụng Scorekeeper, và mã XML cho bố cục.

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

XML code:

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

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

- Các biến số nguyên để theo dõi điểm số.
- Một tham chiếu đến mỗi phần tử TextView điểm số trong MainActivity để bạn có thể cập nhật điểm số.

Thực hiện các bước sau:

1. Tạo hai biến thành viên kiểu số nguyên đại diện cho điểm số của mỗi đội.
2. Tạo hai biến thành viên TextView để giữ tham chiếu đến các phần tử TextView.
3. Trong phương thức onCreate() của MainActivity, tìm các phần tử TextView điểm số bằng id và gán chúng cho các biến thành viên.

1.6 Triển khai các trình xử lý click cho các phần tử ImageButton

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

1. Mở **activity_main.xml** nếu nó chưa được mở, và thêm thuộc tính android:onClick sau vào ImageButton đầu tiên ở bên trái của bố cục:
2. Tên phương thức decreaseScore được gạch chân màu đỏ. Nhấp vào biểu tượng bóng đèn đỏ trong lề bên trái, hoặc nhấp vào tên phương thức và nhấn **Option-Return**, và chọn **Create 'decreaseScore(view)' in 'MainActivity'**. Android Studio sẽ tạo stub cho phương thức decreaseScore() trong MainActivity.
3. Thêm thuộc tính android:onClick ở trên vào ImageButton thứ hai ở bên trái của bố cục. Lần này tên phương thức là hợp lệ, vì stub đã được tạo.
4. Thêm thuộc tính android:onClick sau vào mỗi ImageButton ở bên phải của bố cục:
5. Tên phương thức increaseScore được gạch chân màu đỏ. Nhấp vào biểu tượng bóng đèn đỏ trong lề bên trái, hoặc nhấp vào tên phương thức và nhấn **Option-**

Return, và chọn **Create** 'increaseScore(view)' in 'MainActivity'. Android Studio sẽ tạo stub cho phương thức **increaseScore()** trong **MainActivity**.

6. Thêm mã vào các phương thức stub để giảm và tăng điểm như dưới đây. Cả hai phương thức đều sử dụng [view.getID\(\)](#) để lấy ID của ImageButton của đội mà đã được nhấp, để mã của bạn có thể cập nhật đúng đội.

Mã giải pháp cho increaseScore() và decreaseScore()

Task 2: Tạo tài nguyên drawable

Bạn hiện đã có một ứng dụng Scorekeeper hoạt động! Tuy nhiên, bố cục vẫn khá tẻ nhạt và không truyền đạt được chức năng của các phần tử ImageButton. Để làm cho nó rõ ràng hơn, bạn có thể thay đổi nền màu xám tiêu chuẩn của các nút.

Trong Android, đồ họa thường được xử lý bởi một tài nguyên gọi là [Drawable](#). Trong bài tập tiếp theo, bạn sẽ học cách tạo một loại Drawable nhất định gọi là ShapeDrawable, và áp dụng nó cho các phần tử ImageButton của bạn làm nền.

Để biết thêm thông tin về các Drawable, hãy xem [Drawable Resources](#).

2.1 Tạo một ShapeDrawable

Một [ShapeDrawable](#) là một hình dạng hình học nguyên thủy được định nghĩa trong một tệp XML bằng nhiều thuộc tính, bao gồm màu sắc, hình dạng, khoảng đệm và nhiều hơn nữa. Nó định nghĩa một đồ họa vector, có thể phóng to hoặc thu nhỏ mà không bị mất định nghĩa.

1. Trong bảng Project > Android, **nhấp chuột phải** (hoặc **Control-click**) vào thư mục **drawable** trong thư mục **res**.

2. Chọn **New > Tệp tài nguyên drawable**.

3. Đặt tên tệp là **button_background** và nhấp OK. (Không thay đổi Source set hoặc Directory name, và không thêm qualifiers). Android Studio sẽ tạo tệp **button_background.xml** trong thư mục **drawable**.

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

5. Thêm mã sau đây để tạo hình oval với đường viền:

2.2 Áp dụng ShapeDrawable làm nền

1. Mở **activity_main.xml**.

2. Thêm Drawable làm nền cho tất cả bốn phần tử ImageButton:

3. Nhấp vào tab **Preview** trong trình chỉnh sửa bố cục để thấy rằng nền tự động điều chỉnh kích thước phù hợp với kích thước của ImageButton.

4. Để hiển thị đúng mỗi ImageButton trên tất cả các thiết bị, bạn sẽ thay đổi các thuộc tính android:layout_height và android:layout_width cho mỗi ImageButton thành 70dp, đây là kích thước hợp lý trên hầu hết các thiết bị. Thay đổi thuộc tính đầu tiên cho ImageButton đầu tiên như sau:

5. Nhấp một lần vào "70dp", nhấn **Alt-Enter** trên Windows hoặc **Option-Enter** trên macOS, và chọn **Extract dimension resource** từ menu pop-up.

6. Nhập button_size cho Resource name.

7. Nhấp **OK**. Điều này sẽ tạo một tài nguyên kích thước trong tệp dimens.xml (trong thư mục values), và kích thước trong mã của bạn sẽ được thay thế bằng một tham chiếu đến tài nguyên:

```
@dimen/button_size
```

8. Thay đổi các thuộc tính android:layout_height và android:layout_width cho mỗi phần tử ImageButton sử dụng tài nguyên kích thước mới:

9. Nhấp vào tab **Preview** trong trình chỉnh sửa bố cục để xem bố cục. ShapeDrawable bây giờ được sử dụng cho các phần tử ImageButton.

Task 3: Tạo kiểu cho các phần tử View

Khi bạn tiếp tục thêm các phần tử View và thuộc tính vào bố cục của mình, mã của bạn sẽ bắt đầu trở nên lớn và lặp đi lặp lại, đặc biệt khi bạn áp dụng cùng một thuộc tính cho nhiều phần tử tương tự. Một kiểu (style) có thể chỉ định các thuộc tính chung như khoảng đệm, màu sắc phông chữ, kích thước phông chữ và màu nền. Các thuộc tính liên quan đến bố cục như chiều cao, chiều rộng và vị trí tương đối nên được giữ lại trong tệp tài nguyên bố cục.

Trong bài tập tiếp theo, bạn sẽ học cách tạo kiểu và áp dụng chúng cho nhiều phần tử View và bố cục, cho phép các thuộc tính chung được cập nhật đồng thời từ một vị trí.

Lưu ý: Các kiểu được sử dụng cho các thuộc tính thay đổi diện mạo của View. Các tham số bố cục như chiều cao, chiều rộng và vị trí tương đối nên ở trong tệp bố cục.

3.1 Tạo kiểu cho nút

Trong Android, các kiểu có thể kế thừa thuộc tính từ các kiểu khác. Bạn có thể khai báo một kiểu cha cho kiểu của mình bằng cách sử dụng tham số tùy chọn parent và có các thuộc tính sau:

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

Ví dụ, tất cả bốn phần tử ImageButton trong ứng dụng Scorekeeper chia sẻ một Drawable nền chung nhưng với các biểu tượng khác nhau cho dấu cộng (tăng điểm) và dấu trừ (giảm điểm). Hơn nữa, hai phần tử ImageButton dấu cộng chia sẻ cùng một biểu tượng, như hai phần tử ImageButton dấu trừ. Do đó, bạn có thể tạo ba kiểu:

1. Một kiểu cho tất cả các phần tử ImageButton, bao gồm các thuộc tính mặc định của một ImageButton và cũng là nền Drawable.
2. Một kiểu cho các phần tử ImageButton dấu trừ. Kiểu này kế thừa các thuộc tính của kiểu ImageButton và bao gồm biểu tượng dấu trừ.
3. Một kiểu cho các phần tử ImageButton dấu cộng. Kiểu này kế thừa từ kiểu ImageButton và bao gồm biểu tượng dấu cộng.

Các kiểu này được thể hiện trong hình dưới đây.

Thực hiện các bước sau:

1. Mở rộng **res > values** trong bảng Project > Android, và mở tệp **styles.xml**. Đây là nơi tất cả 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.

Lưu ý thuộc tính parent, đây là cách bạn chỉ định kiểu cha của mình bằng XML.

Thuộc tính `name`, trong trường hợp này là `AppTheme`, xác định tên của kiểu. Thuộc tính `parent`, trong trường hợp này là `Theme.AppCompat.Light.DarkActionBar`, khai báo các thuộc tính kiểu cha mà `AppTheme` kế thừa. Trong trường hợp này, đó là chủ đề mặc định của Android, với nền sáng và thanh hành động tối.

Một chủ đề là một kiểu được áp dụng cho toàn bộ Activity hoặc ứng dụng thay vì một phần tử View đơn lẻ. Sử dụng một chủ đề tạo ra một kiểu nhất quán trên toàn bộ Activity hoặc ứng dụng—ví dụ, một diện mạo và cảm giác nhất quán cho thanh ứng dụng ở mọi phần của ứng dụng của bạn.

2. Ở giữa các thẻ `<resources>`, thêm một kiểu mới với các thuộc tính sau để tạo kiểu chung cho tất cả các nút:

Mã trên đặt kiểu cha thành `Widget.AppCompat.Button` để giữ lại các thuộc tính mặc định của một Button. Nó cũng thêm một thuộc tính thay đổi nền của Drawable thành cái mà bạn đã tạo trong nhiệm vụ trước.

1. Tạo kiểu cho các nút dấu cộng bằng cách mở rộng kiểu `ScoreButtons`:

Thuộc tính `contentDescription` được sử dụng cho người dùng khiếm thị. Nó hoạt động như một nhãn mà một số thiết bị hỗ trợ truy cập 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 dấu trừ:

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

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

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

2. Thay thế các thuộc tính sau cho cả hai phần tử `ImageButton` dấu cộng:

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

3.2 Tạo kiểu cho TextView

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

1. Thêm thuộc tính sau vào tất cả các phần tử TextView:
2. **Nhấp chuột phải** (hoặc **Control-click**) vào bất kỳ đâu trong các thuộc tính của phần tử TextView điểm số đầu tiên và chọn **Refactor > Extract > Style...**
3. Đặt tên cho kiểu là **ScoreText** và chọn ô **textAppearance** (thuộc tính bạn vừa thêm) cũng như ô **Launch 'Use Styles Where Possible' refactoring after the style is extracted**. Điều này sẽ quét tệp bố cục để tìm các view có cùng thuộc tính và áp dụng kiểu cho bạn. Không lấy thuộc tính liên quan đến bố cục—nhấp vào các ô khác để tắt chúng.
4. Chọn **OK**.
5. Đảm bảo phạm vi được đặt thành tệp bố cục **activity_main.xml** và nhấp **OK**.
6. Một bảng điều khiển ở dưới cùng của Android Studio sẽ mở ra nếu cùng một kiểu được tìm thấy trong các view khác. Chọn **Do Refactor** để áp dụng kiểu mới cho các view có cùng thuộc tính.
7. Chạy ứng dụng của bạn. Sẽ không có thay đổi nào ngoại trừ việc tất cả mã kiểu của bạn hiện đang ở trong tệp tài nguyên và tệp bố cục của bạn ngắn hơn.

Mã giải pháp bố cục và kiểu

Dưới đây là các đoạn mã cho bố cục và kiểu.

styles.xml:

activity_main.xml:

3.3 Cập nhật các kiểu

Sức mạnh của việc sử dụng kiểu trở nên rõ ràng khi bạn muốn thực hiện thay đổi cho một số phần tử cùng kiểu. Bạn có thể làm cho văn bản lớn hơn, đậm hơn và sáng hơn, và thay đổi các biểu tượng thành màu của nền nút.

1. Mở tệp **styles.xml**, và thêm hoặc chỉnh sửa các thuộc tính sau cho các kiểu:

Giá trị `colorPrimary` được tự động tạo bởi Android Studio khi bạn tạo dự án. Bạn có thể tìm thấy nó trong bảng **Project > Android** trong tệp `colors.xml` bên trong thư mục `values`. Thuộc tính `TextAppearance.AppCompat.Display3` là một kiểu văn bản đã được định nghĩa trước do Android cung cấp.

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

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

Chạy ứng dụng của bạn. Chỉ với những điều chỉnh này trong tệp `style.xml`, tất cả các view đã được cập nhật để phản ánh những thay đổi.

Nhiệm vụ 4: Chủ đề và những hoàn thiện cuối cùng

Bạn đã thấy rằng các phần tử View có đặc điểm tương tự có thể được tạo kiểu cùng nhau trong tệp `styles.xml`. Nhưng nếu bạn muốn định nghĩa kiểu cho toàn bộ Activity hoặc toàn bộ ứng dụng thì sao? Việc này có thể được thực hiện bằng cách sử dụng chủ đề (themes). Để đặt một chủ đề cho mỗi Activity, bạn cần chỉnh sửa tệp `AndroidManifest.xml`.

Trong nhiệm vụ này, bạn sẽ thêm chủ đề "chế độ ban đêm" vào ứng dụng của mình, cho phép người dùng sử dụng phiên bản có độ tương phản thấp của ứng dụng, để nhìn hơn vào ban đêm, cũng như thực hiện một vài điều chỉnh cho giao diện người dùng.

4.1 Khám phá các chủ đề

1. Mở tệp **AndroidManifest.xml**, tìm thẻ `<application>`, và thay đổi thuộc tính `android:theme` thành:

Đây là một chủ đề đã được định nghĩa sẵn, loại bỏ thanh hành động khỏi Activity của bạn.

2. Chạy ứng dụng của bạn. Thanh công cụ biến mất!

3. Đặt lại chủ đề của ứng dụng về `AppTheme`, là một con của chủ đề `Theme.AppCompat.Light.DarkActionBar` như có thể thấy trong `styles.xml`.

Để áp dụng một chủ đề cho một Activity thay vì toàn bộ ứng dụng, hãy đặt thuộc tính chủ đề trong thẻ `<Activity>` thay vì thẻ `<application>`. Để biết thêm thông tin về các chủ đề và kiểu, hãy xem [Style and Theme Guide](#).

4.2 Thêm nút chủ đề vào menu

Một công dụng của việc thiết lập chủ đề cho ứng dụng của bạn là cung cấp trải nghiệm hình ảnh thay thế cho việc lướt web vào ban đêm. Trong những điều kiện như vậy, thường thì tốt hơn khi có một bố cục tối, độ tương phản thấp. Framework Android cung cấp một chủ đề được thiết kế chính xác cho điều 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 một cách lập trình: hoặc 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 giữa chủ đề thông thường và chủ đề "chế độ ban đêm".

1. Mở **strings.xml** và tạo hai tài nguyên chuỗi cho mục menu tùy chọn này:
2. **Nhấp chuột phải** (hoặc **Control-click**) vào thư mục **res** trong bảng **Project > Android**, và chọn **New > Tập tài nguyên Android**.
3. Đặt tên tệp là **main_menu**, thay đổi **Resource Type** thành **Menu**, và nhấp **OK**. Trình chỉnh sửa bố cục sẽ xuất hiện cho tệp **main_menu.xml**.
4. Nhấp vào tab **Text** của trình chỉnh sửa bố cục để hiển thị mã XML.
5. Thêm một mục menu với các thuộc tính sau:
6. Mở **MainActivity**, nhấn **Ctrl-O** để mở menu **Override Method**, và chọn phương thức **onCreateOptionsMenu(menu:Menu):boolean** nằm dưới danh mục **android.app.Activity**.
7. Nhấp **OK**. Android Studio sẽ tạo stub cho phương thức **onCreateOptionsMenu()** với **return super.onCreateOptionsMenu(menu)** là câu lệnh duy nhất.

1. Trong **onCreateOptionsMenu()**, ngay trước câu lệnh **return.super**, thêm mã để tạo menu:

4.3 Thay đổi chủ đề từ menu

Chủ đề **DayNight** sử dụng lớp **AppCompatActivity** để thiết lập các tùy chọn chế độ ban đêm trong Activity của bạn. Để tìm hiểu thêm về chủ đề này, hãy xem [bài viết blog](#) này.

1. Trong tệp **styles.xml**, thay đổi kiểu cha của **AppTheme** thành **"Theme.AppCompat.DayNight.DarkActionBar"**.

2. Ghi đè phương thức **onOptionsItemSelected()** trong **MainActivity**, và thêm mã để kiểm tra mục menu nào đã được nhấp:

3. Thay thế chú thích **TODO**: trong đoạn mã trên bằng mã kiểm tra xem chế độ ban đêm có được bật hay không. Nếu được bật, mã sẽ thay đổi chế độ ban đêm thành trạng thái bị vô hiệu hóa; ngược lại, mã sẽ bật chế độ ban đêm:

Để phản hồi một cú nhấp vào mục menu, mã xác minh thiết lập chế độ ban đêm hiện tại bằng cách gọi **AppCompatActivity.getDefaultNightMode()**.

Chủ đề chỉ có thể thay đổi khi Activity đang được tạo, vì vậy mã gọi **recreate()** để thay đổi chủ đề có hiệu lực.

4. Chạy ứng dụng của bạn. Mục menu **Night Mode** bây giờ nên chuyển đổi chủ đề của Activity của bạn.

5. Bạn có thể nhận thấy rằng nhãn cho mục menu của bạn luôn đọc là **Night Mode**, điều này có thể gây nhầm lẫn cho người dùng nếu ứng dụng đã ở trong chủ đề tối.

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

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

4.4 Hướng dẫn Lưu Trạng Thái Instance (SaveInstanceState)

Bạn đã học ở các bài trước rằng bạn cần chuẩn bị cho Activity của bạn bị phá hủy và tái tạo vào những thời điểm không mong đợi, chẳng hạn như khi xoay màn hình. Trong ứng dụng này, các phần tử TextView chứa điểm số được thiết lập lại về giá trị ban đầu là 0 khi thiết bị bị xoay. Để khắc phục điều này, hãy làm theo các bước sau:

1. Mở MainActivity và thêm các thẻ dưới các biến thành viên, sẽ được sử dụng làm khóa trong `onSaveInstanceState()`.
2. Ở cuối MainActivity, ghi đè phương thức `onSaveInstanceState()` để lưu trữ giá trị của hai phần tử TextView điểm số.
3. Ở cuối phương thức `onCreate()`, thêm mã để kiểm tra xem có `savedInstanceState` hay không. Nếu có, phục hồi các điểm số vào các phần tử TextView.
4. Chạy ứng dụng, và nhấn nút "+" để tăng điểm số. Chuyển thiết bị hoặc trình giả lập sang chế độ ngang để xem điểm số được duy trì.

Chúc mừng bạn, bây giờ bạn đã có một ứng dụng Scorekeeper được định dạng và tiếp tục hoạt động khi người dùng thay đổi giữa chế độ nằm ngang và dọc.

Mã Giải Pháp

Dự án Android Studio: [Scorekeeper](#)

Thử Thách Lập Trình

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

Thử thách: Hiện tại, các nút của bạn không hoạt động một cách trực quan vì chúng không thay đổi diện mạo khi bị nhấn. Android có một loại Drawable khác được gọi là [StateListDrawable](#), cho phép sử dụng đồ họa khác nhau tùy thuộc vào trạng thái của đối tượng.

Đối với bài toán thử thách này, hãy tạo một tài nguyên Drawable thay đổi nền của ImageButton thành màu giống như viền khi trạng thái của ImageButton là "bị nhấn". Bạn cũng nên đặt màu của văn bản bên trong các phần tử ImageButton thành một bộ chọn làm cho nó thành màu trắng khi nút bị "nhấn".

Tóm Tắt

- Các phần tử Drawable làm tăng vẻ ngoài của giao diện người dùng ứng dụng.

- [ShapeDrawable](#) là một hình dạng hình học nguyên thủy được định nghĩa trong một tệp XML. Các thuộc tính xác định một ShapeDrawable bao gồm màu sắc, hình dạng, độ đậm, và nhiều hơn nữa.
- Nền tảng Android cung cấp một bộ sưu tập lớn các kiểu và chủ đề.
- Sử dụng kiểu có thể giảm lượng mã cần thiết cho các thành phần giao diện người dùng của bạn.
- Một kiểu có thể xác định các thuộc tính chung như chiều cao, độ đậm, màu sắc văn bản, kích thước văn bản, và màu nền.
- Một kiểu không nên bao gồm thông tin liên quan đến bố cục.
- Một kiểu có thể được áp dụng cho một View, Activity, hoặc toàn bộ ứng dụng. Một kiểu áp dụng cho một Activity hoặc toàn bộ ứng dụng phải được định nghĩa trong tệp AndroidManifest.xml.
- Để kế thừa một kiểu, một kiểu mới xác định thuộc tính cha trong XML.
- Khi bạn áp dụng một kiểu cho một tập hợp các phần tử View trong một hoạt động hoặc trong toàn bộ ứng dụng, điều đó được gọi là một chủ đề.
- Để áp dụng một chủ đề, bạn sử dụng thuộc tính android:theme.

Các Khái Niệm Liên Quan

Tài liệu về các khái niệm liên quan có trong [5.1: Drawables, styles, and themes](#).

Học Tập

Xây dựng và chạy một ứng dụng

- Android Studio User Guide
- Add Multi-Density Vector Graphics
- Create App Icons with Image Asset Studio
- Best Practices for User Interface
- Linear Layout
- Drawable Resources
- Styles and Themes
- Buttons
- Layouts
- Support Library
- Screen Compatibility Overview
- Support Different Screen Sizes
- Animate Drawable Graphics
- Loading Large Bitmaps Efficiently

- R.style class of styles and themes
- support.v7.appcompat.R.style class of styles and themes

Material Design:

- Understanding navigation
- App bar

Blog của Nhà Phát Triển Android: [Thư viện Hỗ trợ Thiết kế Android](#)

Khác:

- Medium: DayNight Theme Guide
- Video: Udacity - Themes and Styles
- Roman Nurik's Android Asset Studio
- Glide documentation

Bài Tập

Xây dựng và chạy một ứng dụng

Tạo một ứng dụng hiển thị một ImageView và các nút cộng và trừ, như hình dưới đây. ImageView chứa một drawable danh sách cấp độ là chỉ báo mức pin. Nhấn nút cộng hoặc trừ sẽ thay đổi mức của chỉ báo.

Sử dụng các biểu tượng pin từ Vector Asset Studio để đại diện cho 7 giá trị khác nhau của mức pin.

Ứng dụng nên có các thuộc tính sau:

- Nút cộng tăng mức, làm cho chỉ báo pin trông đầy hơn.
- Nút trừ giảm mức, làm cho chỉ báo trở nên trống hơn một mức.

Câu Hỏi

Câu hỏi 1: Loại Drawable nào bạn sử dụng để tạo một Button có nền căng ra một cách hợp lý để phù hợp với văn bản hoặc hình ảnh bên trong Button, để nó trông đúng trên các kích thước màn hình và định hướng khác nhau?

- LevelListDrawable
- TransitionDrawable
- StateListDrawable
- NinePatchDrawable

Câu hỏi 2: Loại Drawable nào bạn sử dụng để tạo một Button hiển thị một nền khác khi bị nhấn và một nền khác khi hover?

- LevelListDrawable
- TransitionDrawable
- StateListDrawable
- NinePatchDrawable

Câu hỏi 3: Giả sử bạn muốn tạo một ứng dụng có nền trắng, văn bản tối và thanh điều hướng tối. Kiểu cơ bản nào mà kiểu ứng dụng của bạn kế thừa?

- Theme.AppCompat.Light
- Theme.AppCompat.Dark.NoActionBar
- Theme.AppCompat.Light.DarkActionBar
- Theme.AppCompat.NoActionBar
- Theme.NoActionBar

Nội Ứng Dụng của Bạn Để Chấm Điểm

Hướng dẫn cho người chấm điểm:

Kiểm tra xem ứng dụng có các tính năng sau hay không:

- Các nút tăng một biến đếm. Biến đếm điều chỉnh mức trên ImageView, sử dụng phương thức setImageLevel().
- Các mức trong LevelListDrawable từ 0 đến 6.
- Trước khi tăng hoặc giảm mức, các phương thức onClick() kiểm tra xem biến đếm có nằm trong khoảng của [LevelListDrawable](#) (0 đến 6) hay không. Điều này ngăn người dùng đặt một mức không tồn tại.

2.2) Thẻ và màu sắc

Giới Thiệu

Các hướng dẫn về [Material Design](#) của Google là một loạt các thực hành tốt nhất để tạo ra các ứng dụng hấp dẫn về mặt hình ảnh và trực quan. Trong thực hành này, bạn sẽ học cách thêm các widget CardView và FloatingActionButton vào ứng dụng của mình, cách sử dụng hình ảnh một cách hiệu quả và cách áp dụng các thực hành tốt nhất của Material Design để mang đến trải nghiệm tuyệt vời cho người dùng.

Những điều bạn nên biết

Bạn nên có khả năng:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các phần tử giao diện người dùng bằng cách sử dụng trình chỉnh sửa bố cục.
- Chỉnh sửa mã bố cục XML và truy cập các phần tử từ mã Java của bạn.
- Tạo một trình xử lý nhấp chuột cho sự kiện nhấp chuột của một nút.

- Trích xuất văn bản vào một tài nguyên chuỗi và một kích thước vào một tài nguyên kích thước.
- Sử dụng drawables, styles và themes.
- Sử dụng [RecyclerView](#) để hiển thị danh sách.

Những gì bạn sẽ học

- Cách sử dụng các widget Material Design như [FloatingActionButton và CardView](#).
- Cách sử dụng hình ảnh một cách hiệu quả trong ứng dụng của bạn.
- Các thực hành tốt nhất được khuyến nghị để thiết kế các bố cục trực quan với màu sắc nổi bật.

Những gì bạn sẽ làm

- Chỉnh sửa một ứng dụng để tuân theo các hướng dẫn của [Material Design](#).
- Thêm hình ảnh và kiểu dáng vào danh sách [RecyclerView](#).
- Triển khai một [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 mô phỏng với việc triển khai thiết kế rất kém. Bạn sẽ sửa chữa nó để phù hợp với các hướng dẫn thiết kế nhằm tạo ra trải nghiệm người dùng thú vị! Dưới đây là các ảnh chụp màn hình của ứng dụng trước và sau khi cải thiện theo Material Design.

Task 1: Tải Mã Khởi Đầu

Dự án ứng dụng khởi đầu hoàn chỉnh cho thực hành này có sẵn tại [MaterialMe-Starter](#). Trong nhiệm 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 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 thay thế cho mỗi môn thể thao. Bố cục và kiểu dáng hiện tại của ứng dụng khiến nó gần như không thể sử

dụng: mỗi hàng dữ liệu không được tách biệt 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 các sửa đổi cho ứng dụng, hãy khám phá cấu trúc hiện tại của nó. Nó chứa các phần tử sau:

Sport.java

Lớp này đại diện cho mô hình dữ liệu cho mỗi hàng dữ liệu trong RecyclerView. Hiện tại, nó có một trường cho tiêu đề của môn thể thao và một trường cho một số thông tin về môn thể thao.

SportsAdapter.java

Đây là bộ điều hợp cho RecyclerView. Nó sử dụng một ArrayList các đối tượng Sport làm dữ liệu và điền vào mỗi hàng với dữ liệu này.

MainActivity.java

MainActivity khởi tạo RecyclerView và bộ điều hợp, và tạo dữ liệu từ các tệp tài nguyên.

strings.xml

Tệp tài nguyên này chứa tất cả dữ liệu cho ứng dụng, bao gồm các tiêu đề và thông tin cho mỗi môn thể thao.

list_item.xml

Tệp bố cục này định nghĩa mỗi hàng của RecyclerView. Nó bao gồm ba phần tử TextView, một cho mỗi mảnh dữ liệu (tiêu đề và thông tin cho mỗi môn thể thao) và một được sử dụng làm nhãn.

Task 2: Thêm CardView và Hình Ảnh

Một trong những nguyên tắc cơ bản của Material Design là sử dụng hình ảnh nổi bật để nâng cao trải nghiệm người dùng. Thêm hình ảnh vào các mục danh sách RecyclerView là một khởi đầu tốt cho việc 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), các hướng dẫn của Material Design khuyến nghị sử dụng [CardView](#), là một [FrameLayout](#) với một số tính năng bổ sung (như độ cao và góc bo tròn) giúp nó có một cái nhìn và cảm nhận nhất quán

trên nhiều ứng dụng và nền tảng khác nhau. `CardView` là một thành phần giao diện người dùng có trong Thư viện Hỗ trợ Android. Trong phần này, bạn sẽ di chuyển mỗi mục danh sách vào một `CardView` và thêm một hình ảnh để ứng dụng tuân thủ các hướng dẫn của Material.

2.1 Thêm `CardView`

`CardView` không được bao gồm trong SDK Android mặc định, vì vậy bạn phải thêm nó như một phụ thuộc trong `build.gradle`. Thực hiện như sau:

1. Mở tệp **`build.gradle (Module: app)`** và thêm dòng sau vào phần phụ thuộc:

(Phiên bản của thư viện hỗ trợ có thể đã thay đổi kể từ khi viết thực hành này. Cập nhật trên phiên bản được gợi ý bởi Android Studio và nhấp vào Sync để đồng bộ hóa các tệp `build.gradle` của bạn.)

2. Mở tệp **`list_item.xml`** và bao bọc `LinearLayout` gốc bằng `android.support.v7.widget.CardView`. Di chuyển khai báo lược đồ (`xmlns:android="http://schemas.android.com/apk/res/android"`) vào `CardView`, và thêm các thuộc tính sau:

Attribute	Value
<code>android:layout_width</code>	<code>"match_parent"</code>
<code>android:layout_height</code>	<code>"wrap_content"</code>
<code>android:layout_margin</code>	<code>"8dp"</code>

(Khai báo lược đồ cần phải di chuyển vào `CardView` vì `CardView` bây giờ là view cấp cao nhất trong tệp bố cục của bạn.)

3. Chọn **Code > Reformat Code** để định dạng lại mã XML, mà bây giờ nên trông như thế này ở đầu và cuối tệp:

4. Chạy ứng dụng. Bây giờ mỗi mục hàng được chứa bên trong một `CardView`, được nâng cao lên trên lớp dưới cùng và đổ bóng.

2.2 Tải Hình Ảnh

`CardView` không chỉ dành riêng cho văn bản đơn giản: nó tốt nhất cho việc hiển thị sự kết hợp của nội dung. Bạn có một cơ hội tốt để làm cho ứng dụng này thú vị hơn bằng cách thêm hình ảnh banner vào mỗi hàng!

Việc sử dụng hình ảnh tiêu tốn tài nguyên cho ứng dụng của bạn: khung Android phải tải toàn bộ hình ảnh vào bộ nhớ với độ phân giải đầy đủ, ngay cả khi ứng dụng chỉ hiển thị một thumbnail nhỏ của hình ảnh.

Trong phần này, bạn sẽ học cách sử dụng thư viện [Glide](#) để tải hình ảnh lớn một cách hiệu quả, mà không làm tiêu tốn tài nguyên của bạn hoặc thậm chí làm ứng dụng bị treo do ngoại lệ "Out of Memory".

1. Tải tệp zip hình ảnh banner.
2. Mở thư mục **MaterialMe > app > src > main > res** trong trình quản lý tệp của hệ điều hành của bạn, và tạo một thư mục **drawable**, và sao chép các tệp đồ họa riêng lẻ vào thư mục **drawable**.
3. Bạn sẽ cần một mảng chứa đường dẫn đến mỗi hình ảnh để có thể bao gồm nó trong đối tượng Sports. Để làm điều này, định nghĩa một mảng chứa tất cả các đường dẫn đến các drawable như các mục trong tệp strings.xml. Đảm bảo rằng chúng ở cùng thứ tự với mảng sports_titles cũng được định nghĩa trong cùng một tệp.

2.3 Chỉnh Sửa Đối Tượng Sport

Đối tượng Sport cần bao gồm tài nguyên Drawable tương ứng với môn thể thao. Để đạt được điều này:

1. Thêm một biến thành viên kiểu số nguyên vào đối tượng Sport để chứa tài nguyên Drawable.
2. Chỉnh sửa hàm khởi tạo sao cho nó nhận một số nguyên làm tham số và gán nó cho biến thành viên.
3. Tạo một phương thức getter cho số nguyên tài nguyên.

2.4 Sửa Phương Thức initializeData()

Trong MainActivity, phương thức initializeData() hiện đang bị lỗi, vì hàm khởi tạo của đối tượng Sport yêu cầu tài nguyên hình ảnh là tham số thứ ba.

Một cấu trúc dữ liệu thuận tiện để sử dụng là [TypedArray](#). Một 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 chỉ số trong cùng một vòng lặp.

1.Trong phương thức `initializeData()`, lấy `TypedArray` của các ID tài nguyên bằng cách gọi `getResources().obtainTypedArray()`, truyền vào tên của mảng tài nguyên `Drawable` mà bạn đã định nghĩa trong tệp `strings.xml`:

Bạn có thể truy cập một phần tử tại chỉ số `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 `getResourceId()`.

2.Sửa mã trong vòng lặp tạo các đối tượng `Sport`, thêm ID tài nguyên `Drawable` thích hợp làm tham số thứ ba bằng cách gọi `getResourceId()` trên `TypedArray`.

3.Dọn dẹp dữ liệu trong `TypedArray` sau khi bạn đã tạo danh sách dữ liệu `Sport`.

2.5 Thêm `ImageView` vào Các Mục Danh Sách

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

2.Thêm một `ImageView` làm phần tử đầu tiên trong `RelativeLayout` với các thuộc tính sau:

Attribute	Value
<code>android:layout_width</code>	<code>"match_parent"</code>
<code>android:layout_height</code>	<code>"wrap_content"</code>
<code>android:id</code>	<code>"@+id/sportsImage"</code>
<code>android:adjustViewBounds</code>	<code>"true"</code>

(Thuộc tính `adjustViewBounds` giúp `ImageView` điều chỉnh biên giới của nó để duy trì tỷ lệ khung hình của hình ảnh.)

3.Thêm các thuộc tính sau vào phần tử `TextView` tiêu đề:

Attribute	Value
<code>android:layout_alignBottom</code>	<code>"@id/sportsImage"</code>
<code>android:theme</code>	<code>"@style/ThemeOverlay.AppCompat.Dark"</code>

4.Thêm các thuộc tính sau vào phần tử `TextView` tiêu đề tin tức:

Attribute	Value
android:layout_below	"@id/sportsImage"
android:textColor	"?android:textColorSecondary"

5. Thêm các thuộc tính sau vào phần tử TextView phụ đề:

Attribute	Value
android:layout_below	"@id/newsTitle"

(Dấu hỏi trong thuộc tính textColor trên có nghĩa là khung sẽ áp dụng giá trị từ chủ đề hiện đang được áp dụng. Trong trường hợp này, thuộc tính này được kế thừa từ chủ đề "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 Sử Dụng Glide

Sau khi tải hình ảnh và thiết lập ImageView, bước tiếp theo là chỉnh sửa SportsAdapter để tải hình ảnh vào ImageView trong phương thức onBindViewHolder(). Nếu bạn thực hiện theo cách này, bạn sẽ thấy ứng dụng của mình gặp sự cố do lỗi "Out of Memory". Khung Android phải tải hình ảnh vào bộ nhớ mỗi lần với độ phân giải đầy đủ, bất kể kích thước hiển thị của ImageView là gì.

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](#), và bạn sẽ làm điều này 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 tạm thời trong khi hình ảnh mong muốn đang được tải.

Lưu ý: Để tìm hiểu thêm về cách giảm mức tiêu thụ bộ nhớ trong ứng dụng của bạn, hãy [xem Loading Large Bitmaps Efficiently](#).

1. Mở tệp **build.gradle (Module: app)** và thêm phụ thuộc sau cho Glide vào phần phụ thuộc:
2. Mở **SportsAdapter** và thêm một biến trong lớp ViewHolder cho ImageView.
3. Khởi tạo biến trong hàm khởi tạo của lớp ViewHolder:
4. Thêm dòng mã sau vào phương thức bindTo() trong lớp ViewHolder để lấy tài nguyên hình ảnh từ đối tượng Sport và tải nó vào ImageView bằng cách sử dụng Glide:

5. Chạy ứng dụng, các mục danh sách của bạn giờ đây sẽ có đồ họa nổi bật, làm cho trải nghiệm người dùng trở nên năng động và thú vị!

Đó là tất cả những gì cần thiết để tải hình ảnh bằng Glide. Glide cũng có nhiều tính năng bổ sung cho phép bạn thay đổi kích thước, biến đổi và tải hình ảnh theo nhiều cách khác nhau. Hãy truy cập trang [Github của Glide](#) để tìm hiểu thêm.

Task 3: Làm cho CardView có thể vuốt, di chuyển và nhấp chuột

Khi người dùng thấy các thẻ trong một ứng dụng, họ có những kỳ vọng về cách mà các thẻ sẽ hoạt động. [Material Design guidelines](#) cho biết rằng:

- Một thẻ có thể bị xóa, thường là bằng cách vuốt nó đi.
- Một 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.

Bạn sẽ triển khai những hành vi này trong ứng dụng của mình.

3.1 Triển Khai Vuốt Để Xóa

SDK Android bao gồm một lớp gọi là [ItemTouchHelper](#) được sử dụng để xác định những gì xảy ra với các mục danh sách RecyclerView khi người dùng thực hiện các hành động chạm khác nhau, như vuốt hoặc kéo và thả. Một số trường hợp sử dụng thông dụng đã được triển khai sẵn trong một tập hợp các phương thức trong [ItemTouchHelper.SimpleCallback](#).

ItemTouchHelper.SimpleCallback cho phép bạn định nghĩa các hướng hỗ trợ cho việc vuốt và di chuyển các mục danh sách, và thực hiện hành vi vuốt và di chuyển.

Thực hiện các bước sau:

1. Mở **MainActivity** và tạo một đối tượng ItemTouchHelper mới trong phương thức onCreate() ở cuối, bên dưới phương thức initializeData(). Đối với tham số của nó, bạn sẽ tạo một thẻ hiện mới của ItemTouchHelper.SimpleCallback. Khi bạn nhập **new ItemTouchHelper**, các gợi ý sẽ xuất hiện. Chọn **ItemTouchHelper.SimpleCallback{...}** từ menu gợi ý. Android Studio sẽ tự động thêm các phương thức cần thiết: onMove() và onSwiped() như hình dưới đây.

Nếu các phương thức cần thiết không được thêm tự động, nhấp vào biểu tượng bóng đèn đỏ ở lề bên trái và chọn Implement methods.

Constructor `SimpleCallback` sẽ được gạch chân bằng màu đỏ vì bạn chưa cung cấp các tham số cần thiết: hướng mà bạn dự định hỗ trợ để di chuyển và vuốt các mục danh sách.

2. Vì hiện tại chúng ta chỉ triển khai vuốt để xóa, bạn nên truyền vào 0 cho các hướng di chuyển được hỗ trợ và `ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT` cho các hướng vuốt được hỗ trợ.

3. Bạn bây giờ phải triển khai hành vi mong muốn trong `onSwiped()`. Trong trường hợp này, vuốt thẻ sang trái hoặc phải sẽ xóa nó khỏi danh sách. Gọi `remove()` trên tập dữ liệu, truyền vào chỉ số thích hợp bằng cách lấy vị trí từ `ViewHolder`.

4. Để cho phép `RecyclerView` hoạt động hiệu ứng xóa một cách chính xác, bạn cũng phải gọi `notifyItemRemoved()`, một lần nữa truyền vào chỉ số thích hợp bằng cách lấy vị trí từ `ViewHolder`.

5. Dưới đối tượng `ItemTouchHelper` mới trong phương thức `onCreate()` của `MainActivity`, gọi `attachToRecyclerView()` trên thể hiện `ItemTouchHelper` để thêm nó vào `RecyclerView` của bạn.

6. Chạy ứng dụng của bạn, bây giờ bạn có thể vuốt các mục danh sách sang trái và phải để xóa chúng!

3.2 Triển Khai Kéo và Thả

Bạn cũng có thể triển khai chức năng kéo và thả bằng cách sử dụng cùng một `SimpleCallback`. Tham số đầu tiên của `SimpleCallback` xác định các hướng mà `ItemTouchHelper` hỗ trợ để di chuyển các đối tượng xung quanh.

Thực hiện các bước sau:

1. Thay đổi tham số đầu tiên của `SimpleCallback` từ 0 thành bao gồm mọi hướng, vì chúng ta muốn có thể kéo và thả ở bất kỳ đâu.
2. Trong phương thức `onMove()`, lấy chỉ số gốc và chỉ số mục tiêu từ tham số thứ hai và thứ ba được truyền vào (tương ứng với các `ViewHolder` gốc và mục tiêu).
3. Hoán đổi các mục trong tập dữ liệu bằng cách gọi `Collections.swap()` và truyền vào tập dữ liệu, cùng với các chỉ số ban đầu và cuối.
4. Thông báo cho bộ điều hợp rằng mục đã được di chuyển, truyền vào các chỉ số cũ và mới, và thay đổi câu lệnh `return` thành `true`.

5. Chạy ứng dụng của bạn. Bây giờ bạn có thể xóa các mục danh sách bằng cách vuốt chúng sang trái hoặc phải, hoặc sắp xếp lại chúng bằng cách nhấn giữ để kích hoạt chế độ Kéo và Thả.

3.3 Triển Khai Bố Cục DetailActivity

Theo các [Material Design guidelines](#), một thẻ được sử dụng để cung cấp một điểm truy cập vào thông tin chi tiết hơn. Bạn có thể thấy mình nhấn vào các thẻ để xem thêm thông tin về các môn thể thao, vì đó là cách mà bạn mong đợi các thẻ hoạt động.

Trong phần này, bạn sẽ thêm một Activity chi tiết sẽ được khởi chạy khi bất kỳ mục nào trong danh sách được nhấn. Đối với thực hành này, Activity chi tiết sẽ chứa tên và hình ảnh của mục danh sách mà bạn đã nhấn, nhưng sẽ chỉ chứa văn bản chi tiết giả định chung, vì vậy bạn không phải tạo chi tiết tùy chỉnh cho mỗi mục trong danh sách.

1. Tạo một Activity mới bằng cách vào **File > New > Activity > Empty Activity**.
2. Đặt tên là **DetailActivity**, và chấp nhận tất cả các mặc định.
3. Mở tệp bố cục **activity_detail.xml** mới tạo và thay đổi ViewGroup gốc 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 phần tử TextView và ImageView từ tệp **list_item.xml** vào tệp **activity_detail.xml**.
6. Thêm từ "Detail" vào tham chiếu trong mỗi thuộc tính android:id để phân biệt với các ID trong list_item.xml. Ví dụ, thay đổi ID của ImageView từ **sportsImage** thành **sportsImageDetail**.
7. Trong tất cả các phần tử TextView và ImageView, thay đổi tất cả các tham chiếu đến ID cho vị trí tương đối như layout_below để sử dụng ID "Detail".
8. Đối với subTitleDetail TextView, xóa chuỗi văn bản giả định và dán một đoạn văn bản chung để thay thế văn bản chi tiết (ví dụ, một vài đoạn văn của [Lorem Ipsum](#)). Trích xuất văn bản này thành một tài nguyên chuỗi.
9. Thay đổi khoảng cách đệm (padding) trên các phần tử TextView thành 16dp.
10. Bao bọc toàn bộ RelativeLayout bằng một ScrollView. Thêm các thuộc tính layout_height và layout_width cần thiết, và thêm thuộc tính `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 nên trông như sau:

3.4 Triển Khai Giao Diện Chi Tiết và Trình Xử Lý Nhấp

Thực hiện các bước sau để triển khai giao diện chi tiết và trình xử lý nhấp:

1. Mở **SportsAdapter** và thay đổi lớp ViewHolder bên trong, đã mở rộng RecyclerView.ViewHolder, để cũng triển khai View.OnClickListener, và triển khai phương thức cần thiết (onClick()).
2. Đặt OnClickListener cho itemView trong hàm khởi tạo của ViewHolder. Toàn bộ hàm khởi tạo bây giờ sẽ trông như sau:
3. Trong phương thức onClick(), lấy đối tượng Sport cho mục đã được nhấn bằng cách sử dụng getAdapterPosition().
4. Trong cùng một phương thức, thêm một Intent để khởi chạy DetailActivity, đưa tiêu đề và image_resource vào như các dữ liệu bổ sung trong Intent, và gọi startActivity() trên biến mContext, truyền vào Intent mới.
5. Mở **DetailActivity** và khởi tạo ImageView và TextView tiêu đề trong onCreate().
6. Lấy tiêu đề từ Intent đến và gán nó cho TextView.
7. Sử dụng Glide để tải hình ảnh vào ImageView.
8. Chạy ứng dụng. Nhấp vào một mục danh sách giờ đây sẽ khởi chạy DetailActivity.


Task 4: Thêm FAB và Chọn Bảng Màu Material Design

Một trong những nguyên tắc của Material Design là sử dụng các phần tử nhất quán trên các ứng dụng và nền tảng để người dùng nhận biết được các mẫu và biết cách sử dụng chúng. Bạn đã sử dụng một trong những phần tử như vậy: [Floating Action Button](#) (FAB). FAB là một nút hình tròn nổi trên giao diện người dùng. Nó được sử dụng để khuyến khích một hành động cụ thể cho người dùng, một hành động rất có khả năng được sử dụng trong một hoạt động nhất định. Trong nhiệm vụ này, bạn sẽ tạo một FAB để đặt lại tập dữ liệu về trạng thái ban đầu của nó.

4.1 Thêm FAB

Nút Hành Động Nổi là một phần của [Design Support Library](#) .

1. Mở tệp **build.gradle (Module: app)** và thêm dòng mã sau cho thư viện hỗ trợ thiết kế vào phần phụ thuộc:
2. Thêm một biểu tượng cho FAB bằng cách nhấp chuột phải (hoặc Control-click) vào thư mục res trong khung Project > Android, và chọn **New > Vector Asset**. FAB sẽ đặt lại nội

dung của RecyclerView, vì vậy biểu tượng  làm mới (refresh) sẽ phù hợp. Đổi tên thành **ic_reset**, nhấp Next, và nhấp **Finish**.

3. Mở **activity_main.xml** và thêm một FloatingActionButton với các thuộc tính sau:

Attribute	Value
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_alignParentBottom	"true"
android:layout_alignParentRight	"true"
android:layout_alignParentEnd	"true"
android:layout_margin	"16dp"
android:src	"@drawable/ic_reset"
android:tint	"@android:color/white"
android:onClick	resetSports

4. Mở **MainActivity** và thêm phương thức `resetSports()` với một câu lệnh 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 hủy và được tạo lại khi có sự thay đổi cấu hình, việc xoay thiết bị sẽ đặt lại dữ liệu trong triển khai này. Để các thay đổi có thể duy trì (như trong trường hợp sắp xếp lại hoặc xóa dữ liệu), bạn sẽ phải triển khai `onSaveInstanceState()` hoặc ghi các thay đổi vào một nguồn bền vững (như cơ sở dữ liệu hoặc `SharedPreferences`, được mô tả trong các bài học khác).

4.2 Chọn Bảng Màu Material Design

Material Design khuyến nghị chọn ít nhất ba màu cho ứng dụng của bạn:

- Một màu chính. Màu này sẽ tự động được sử dụng để tô màu cho thanh ứng dụng (thanh chứa tiêu đề của ứng dụng).
- Một màu chính tối. Một sắc thái tối hơn của cùng một màu. Màu này được sử dụng cho thanh trạng thái phía trên thanh ứng dụng, giữa các thứ khác.

- Một màu nhấn. Một màu tương phản tốt với màu chính. Màu này được sử dụng cho các điểm nhấn khác nhau, nhưng cũng là màu mặc định của FAB.

Khi bạn chạy ứng dụng của mình, bạn có thể nhận thấy rằng màu của FAB và màu của thanh ứng dụng đã được thiết lập sẵn. Trong nhiệm vụ này, bạn sẽ tìm hiểu nơi các màu này được thiết lập. Bạn có thể sử dụng Hướng Dẫn Màu Material để chọn một số màu để thử nghiệm.

1. Trong khung **Project > Android**, điều hướng đến tệp **styles.xml** (nằm trong thư mục **values**). Kiểu AppTheme định nghĩa ba màu theo mặc định: `colorPrimary`, `colorPrimaryDark`, và `colorAccent`. Những kiểu này được định nghĩa bởi các giá trị từ tệp `colors.xml`.
2. Chọn một màu từ Hướng Dẫn Màu Material để sử dụng làm màu chính, chẳng hạn như #607D8B (trong bảng màu Xanh Xám). Nó nên nằm trong khoảng 300-700 của bảng màu để bạn có thể chọn một màu nhấn và màu tối phù hợp.
3. Mở tệp **colors.xml**, và sửa đổi giá trị hex của `colorPrimary` để phù hợp với màu bạn đã chọn.
4. Chọn một sắc thái tối hơn của cùng một màu để sử dụng làm màu chính tối, chẳng hạn như #37474F. Một lần nữa, sửa đổi giá trị hex cho `colorPrimaryDark` trong **colors.xml** để phù hợp.
5. Chọn một màu nhấn cho FAB từ các màu có giá trị bắt đầu bằng chữ A, và có màu tương phản tốt với màu chính (như Màu Cam Đậm A200). Thay đổi giá trị `colorAccent` trong **colors.xml** để phù hợp.
6. Chạy ứng dụng. Thanh ứng dụng và FAB bây giờ đã thay đổi để phản ánh bảng màu mới!

Nếu bạn muốn thay đổi màu của FAB thành một màu khác ngoài màu chủ đề, hãy sử dụng thuộc tính `app:backgroundTint`. Lưu ý rằng điều này sử dụng không gian tên `app:` và Android Studio sẽ nhắc bạn thêm một câu lệnh để định nghĩa không gian tên.

Giải Pháp Mã

Dự án Android Studio: [MaterialMe](#)

Thử Thách Lập Trình

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

Thử Thách Lập Trình 1

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

- Thêm chi tiết thực cho đối tượng Sport và truyền các chi tiết này đến DetailActivity.
- Triển khai một cách để đảm bảo rằng trạng thái của ứng dụng được duy trì qua các thay đổi về hướng.

Thử Thách Lập Trình 2

Tạo một ứng dụng với bốn hình ảnh được sắp xếp trong một lưới ở trung tâm của giao diện. Làm cho ba hình nền đầu tiên có màu sắc đồng nhất với các hình dạng khác nhau (hình vuông, hình tròn, đường thẳng), và hình thứ tư là Biểu Tượng Thiết Kế Vật Liệu Android. Làm cho mỗi hình ảnh này phản hồi với các cú nhấp chuột như sau:

1. Một trong các khối màu sẽ khởi động lại Activity sử dụng hiệu ứng hoạt hình [Explode](#) cho cả việc vào và thoát.
2. Khởi động lại Activity từ một khối màu khác, lần này sử dụng hiệu ứng [Fade](#).
3. Nhấn vào khối màu thứ ba sẽ bắt đầu một hoạt hình tại chỗ của View (chẳng hạn như xoay).
4. Cuối cùng, nhấn vào biểu tượng Android sẽ bắt đầu một Activity thứ cấp với một Hiệu ứng Chia Sẻ, hoán đổi khối Android với một trong các khối khác.

Lưu ý: Bạn phải đặt cấp độ SDK tối thiểu của mình là 21 hoặc cao hơn để triển khai các hiệu ứng chia sẻ.

Giải Pháp Thử Thách 2

Dự án Android Studio: [TransitionsandAnimations](#)

Tóm Tắt

- [CardView](#) là một bố cục tốt để sử dụng cho việc trình bày thông tin có đa phương tiện (chẳng hạn như hình ảnh và văn bản).
- CardView là một thành phần UI có trong Thư viện Hỗ Trợ Android.
- CardView không chỉ được thiết kế cho các phần tử View con là văn bản.
- Tải hình ảnh trực tiếp vào ImageView tiêu tốn nhiều bộ nhớ, bởi vì hình ảnh được tải với độ phân giải đầy đủ. Để tải hình ảnh một cách hiệu quả vào ứng dụng của bạn, hãy sử dụng thư viện tải hình ảnh như [Glide](#).

- SDK Android có một lớp gọi là [ItemTouchHelper](#) giúp ứng dụng của bạn nhận thông tin về các sự kiện nhấp chuột, vuốt và kéo-thả trong giao diện.
- [FloatingActionButton](#) (FAB) tập trung người dùng vào một hành động cụ thể và "nổi" trong giao diện của bạn.
- Material Design là một tập hợp các nguyên tắc hướng dẫn để tạo ra các ứng dụng nhất quán, dễ hiểu và vui tươi.
- Theo Material Design, việc chọn ba màu cho ứng dụng của bạn là thực hành tốt: một màu chính, một màu tối chính và một màu nhấn.
- Material Design khuyến khích việc sử dụng hình ảnh và màu sắc nổi bật để nâng cao trải nghiệm người dùng. Nó cũng khuyến khích các phần tử nhất quán trên các nền tảng, ví dụ như sử dụng các widget CardView và FAB.
- Sử dụng Material Design để tạo ra chuyển động có ý nghĩa, trực quan cho các phần tử UI như các thẻ có thể bị xóa hoặc sắp xếp lại.

Khái Niệm Liên Quan

Tài liệu về khái niệm liên quan có trong [5.2: Material Design](#).

Tìm Hiểu Thêm

Tài liệu về Android Studio:

- Android Studio User Guide
- Add multi-density vector graphics
- Create app icons with Image Asset Studio

Tài liệu phát triển Android Studio:

- User Interface & Navigation
- Linear Layout
- FloatingActionButton
- Drawable Resources
- View Animation
- Support Library
- Animate Drawable Graphics

- Loading Large Bitmaps Efficiently

Thiết Kế Vật Liệu:

- Material Design for Android
- Material Design palette generator
- Create a List with RecyclerView
- App Bar
- Defining Custom Animations

Blog Các Nhà Phát Triển Android: [Android Design Support Library](#)

Khác:

- Glide documentation
- Glide github page

Bài Tập

Xây dựng và chạy một ứng dụng

Mở ứng dụng [MaterialMe](#).

1. Tạo một chuyển tiếp phần tử chia sẻ giữa MainActivity và DetailActivity, với hình ảnh banner cho môn thể thao là phần tử chia sẻ.
2. Nhấp vào một mục trong danh sách trong ứng dụng MaterialMe sẽ kích hoạt chuyển tiếp. Hình ảnh banner từ thẻ sẽ di chuyển lên đầu màn hình trong chế độ xem chi tiết.

Trả Lời Các Câu Hỏi

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:

- colorPrimary
- colorPrimaryDark
- colorAccent

- colorAccentDark

Câu Hỏi 2

Thư viện hỗ trợ nào mà FloatingActionButton thuộc về? Chọn một:

- v4 Support Library
- v7 Support Library
- Design Support Library
- Custom Button Support Library

Câu Hỏi 3

Trong bảng màu [Material Design](#), sắc thái nào của một màu nên được sử dụng làm màu chính cho thương hiệu của bạn trong ứng dụng? Chọn một:

- Bất kỳ sắc thái màu nào bắt đầu bằng chữ A.
- Bất kỳ sắc thái màu nào được gán nhãn 200.
- Bất kỳ sắc thái màu nào được gán nhãn 500.
- Bất kỳ sắc thái màu nào được gán nhãn 900.

Nội Ứng Dụng Để Chấm Điểm

Hướng Dẫn cho Người Chấm Điểm

- Kiểm tra xem ứng dụng có các tính năng sau:
- Các chuyển tiếp nội dung của sổ được kích hoạt trong chủ đề ứng dụng.
- Một chuyển tiếp phần tử chia sẻ được chỉ định trong kiểu ứng dụng.
- Chuyển tiếp được định nghĩa dưới dạng tài nguyên XML.
- Một tên chung được gán cho các phần tử chia sẻ trong cả hai bố cục với thuộc tính android:transitionName.
- Mã sử dụng phương thức [ActivityOptions.makeSceneTransitionAnimation\(\)](#).

2.3) Bố cục thích ứng

Giới Thiệu

Ứng dụng MaterialMe mà bạn đã tạo trong một chương trước không xử lý đúng các thay đổi về hướng của thiết bị từ chế độ chân dung (dọc) sang chế độ phong cảnh (ngang). Trên

một máy tính bảng, kích thước phông chữ quá nhỏ và không gian không được sử dụng hiệu quả.

Khung Android có cách để giải quyết cả hai vấn đề này. Các bộ định tính tài nguyên cho phép thời gian chạy Android sử dụng các tệp tài nguyên XML thay thế tùy thuộc vào cấu hình thiết bị — định hướng, ngôn ngữ địa phương và các bộ định tính khác. Để xem danh sách đầy đủ các bộ định tính có sẵn, xem [Providing alternative resources](#).

Trong bài thực hành này, bạn sẽ tối ưu hóa việc sử dụng không gian trong ứng dụng MaterialMe để ứng dụng hoạt động tốt ở chế độ phong cảnh và trên máy tính bảng. Trong một bài thực hành khác về việc sử dụng trình biên tập bố cục, bạn đã học cách tạo các biến thể bố cục cho định hướng ngang và máy tính bảng. Trong bài thực hành này, bạn sẽ sử dụng một bố cục thích ứng, là một bố cục hoạt động tốt trên các kích thước màn hình và định hướng khác nhau, các thiết bị khác nhau, các ngôn ngữ và địa phương khác nhau, và các phiên bản Android khác nhau.

Những gì bạn nên biết

Bạn nên có khả năng:

- Tạo và chạy ứng dụng trong Android Studio.
- Tạo và chỉnh sửa các phần tử UI bằng cách sử dụng trình biên tập bố cục.
- Chỉnh sửa mã bố cục XML và truy cập các phần tử từ mã Java của bạn.
- Tạo một trình xử lý nhấp chuột cho cú nhấp chuột của Button.
- Sử dụng drawable, kiểu, và chủ đề.
- Trích xuất văn bản thành tài nguyên chuỗi và kích thước thành tài nguyên kích thước.

Những gì bạn sẽ học

Bạn sẽ học cách:

- Tạo tài nguyên thay thế cho các thiết bị ở chế độ phong cảnh.
- Tạo tài nguyên thay thế cho máy tính bảng.
- Tạo tài nguyên thay thế cho các ngôn ngữ địa phương khác nhau.

Những gì bạn sẽ làm

- Cập nhật ứng dụng MaterialMe để sử dụng không gian tốt hơn ở chế độ phong cảnh.

- Thêm một bố cục thay thế cho máy tính bảng.
- Địa phương hóa nội dung của ứng dụng của bạn.

Tổng Quan Ứng Dụng

Ứng dụng MaterialMe được cập nhật sẽ bao gồm một bố cục cải tiến cho chế độ phong cảnh trên điện thoại. Nó cũng sẽ bao gồm các bố cục cải tiến cho chế độ chân dung và phong cảnh trên máy tính bảng, và sẽ cung cấp nội dung địa phương hóa cho người dùng ngoài Hoa Kỳ.

Hình ảnh chụp màn hình dưới đây cho thấy một điện thoại đang chạy ứng dụng MaterialMe được cập nhật ở chế độ phong cảnh:

Hình ảnh chụp màn hình dưới đây cho thấy một máy tính bảng đang chạy ứng dụng MaterialMe được cập nhật, với các bộ định tính để hiển thị hai cột khi chạy ở chế độ chân dung:

Task 1: Hỗ Trợ Định Hướng Phong Cảnh

Bạn có thể nhớ rằng khi người dùng thay đổi định hướng của thiết bị, khung Android sẽ hủy bỏ và tạo lại hoạt động hiện tại. Định hướng mới thường có yêu cầu bố cục khác với bố cục ban đầu. Ví dụ, ứng dụng MaterialMe trông đẹp trong chế độ chân dung, nhưng không tận dụng tối ưu màn hình trong chế độ phong cảnh. Với chiều rộng lớn hơn trong chế độ phong cảnh, hình ảnh trong mỗi mục danh sách làm cho văn bản bị áp đảo, gây ra trải nghiệm người dùng kém.

Trong nhiệm vụ này, bạn sẽ tạo một tệp tài nguyên thay thế sẽ thay đổi diện mạo của ứng dụng khi nó được sử dụng ở chế độ phong cảnh.

1.1 Chuyển Sang GridLayoutManager

Các bố cục chứa các mục danh sách thường trông không cân bằng ở chế độ phong cảnh khi các mục danh sách bao gồm hình ảnh 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 phong cảnh.

Nhớ rằng các mục trong danh sách RecyclerView được đặt bằng cách sử dụng một LayoutManager; cho đến nay, bạn đã sử dụng [LinearLayoutManager](#), cái sẽ bố trí mỗi mục trong một danh sách cuộn theo chiều dọc hoặc chiều ngang. [GridLayoutManager](#) là một layout manager khác hiển thị các mục trong một lưới, thay vì trong một danh sách.

Khi bạn tạo một `GridLayoutManager`, bạn cung cấp hai tham số: ngữ cảnh ứng dụng và một số nguyên đại diện cho số cột. Bạn có thể thay đổi số cột theo cách lập trình, điều này mang lại cho bạn tính linh hoạt trong việc thiết kế các bố cục thích ứng. Trong trường hợp này, số cột nên là 1 ở chế độ chân dung (một cột) và 2 khi ở chế độ phong cảnh. Lưu ý rằng khi số cột là 1, `GridLayoutManager` hoạt động tương tự như `LinearLayoutManager`.

Bài thực hành này dựa trên ứng dụng `MaterialMe` từ bài thực hành trước.

Các bước thực hiện:

1. Tiếp tục phát triển phiên bản `MaterialMe` của bạn hoặc tải xuống ứng dụng [MaterialMe](#). Nếu bạn quyết định sao chép dự án `MaterialMe` để bảo tồn phiên bản từ bài thực hành trước, hãy đổi tên phiên bản sao chép thành **MaterialMe-Resource**.
2. Tạo một tệp tài nguyên mới có tên `integers.xml`: Mở thư mục **res** trong khung **Project > Android**. Nhấp chuột phải (hoặc **Control-click**) vào thư mục **values** và chọn **New > Values resource file**.
 1. Đặt tên cho tệp là **integers.xml** và nhấp **OK**.
 2. Tạo một hằng số nguyên giữa các thẻ `<resources>` có tên `grid_column_count` và gán giá trị bằng 1:
 3. Tạo một tệp tài nguyên giá trị khác, cũng có tên là **integers.xml**; tuy nhiên, tên sẽ được sửa đổi khi bạn thêm các bộ định tính tài nguyên từ bảng **Available qualifiers**.
 4. Chọn **Orientation** trong bảng **Available qualifiers** và nhấn biểu tượng `>>` ở giữa hộp thoại để gán bộ định tính này.
 5. Thay đổi menu **Screen orientation** thành **Landscape**, và lưu ý tên thư mục `values-land` xuất hiện. Đây là bản chất của các bộ định tính tài nguyên: tên thư mục cho Android biết khi nào sử dụng tệp bố cục cụ thể đó. Trong trường hợp này, đó là khi điện thoại được xoay sang chế độ phong cảnh.
 6. Nhấp **OK** để tạo tệp bố cục mới.
 7. Sao chép hằng số nguyên bạn đã tạo vào tệp tài nguyên mới này, nhưng thay đổi giá trị thành 2.

Bạn nên có hai tệp `integers.xml` riêng biệt được nhóm vào một thư mục `integers.xml` trong khung **Project > Android**. Tệp thứ hai được gán nhãn với bộ định tính bạn đã chọn, đó là `land` trong trường hợp này. Bộ định tính xuất hiện trong ngoặc: `integers.xml (land)`.

1.2 Sửa Đổi `MainActivity`

1. Mở **MainActivity** và thêm mã vào `onCreate()` để lấy số nguyên từ tệp tài nguyên `integers.xml`:

2. Thay đổi `LinearLayoutManager` cho `RecyclerView` thành `GridLayoutManager`, truyền vào ngữ cảnh và số nguyên mà bạn vừa tạo:

3. Chạy ứng dụng và xoay thiết bị. Số cột sẽ tự động thay đổi theo định hướng của thiết bị. Khi sử dụng ứng dụng ở chế độ phong cảnh, bạn sẽ nhận thấy rằng chức năng vuốt để xóa không còn trực quan nữa, vì các mục hiện ở trong một lưới thay vì một cột duy nhất. Trong các bước tiếp theo, bạn sẽ tắt hành động vuốt nếu có nhiều hơn một cột.

4. Sử dụng biến `gridColumnCount` để vô hiệu hóa hành động vuốt (đặt `swipeDirs` thành bằng không) khi có nhiều hơn một cột:

5. Sử dụng `swipeDirs` thay cho các tham số hướng vuốt (`ItemTouchHelper.LEFT` | `ItemTouchHelper.RIGHT`) trong `ItemTouchHelper.SimpleCallback()`:

6. Chạy ứng dụng và xoay thiết bị. Trong chế độ phong cảnh (horizontally), người dùng sẽ không còn có thể vuốt để xóa một thẻ nữa.

Task 2: Hỗ Trợ Máy Tính Bảng

Mặc dù bạn đã sửa đổi ứng dụng để nhìn tốt hơn ở chế độ phong cảnh, việc chạy nó trên một máy tính bảng với kích thước vật lý lớn hơn dẫn đến tất cả văn bản hiển thị quá nhỏ. Ngoài ra, khi thiết bị ở chế độ phong cảnh, màn hình không được sử dụng hiệu quả; ba cột sẽ phù hợp hơn cho màn hình kích thước máy tính bảng trong chế độ phong cảnh.

Trong nhiệm vụ này, bạn sẽ thêm các bộ định tính tài nguyên bổ sung để thay đổi diện mạo của ứng dụng khi sử dụng trên máy tính bảng.

2.1 Thích Ứng Bố Cục cho Máy Tính Bảng

Trong bước này, bạn sẽ tạo các bộ định tính tài nguyên khác nhau để tối đa hóa việc sử dụng màn hình cho các thiết bị kích thước máy tính bảng, tăng số cột lên 2 cho chế độ chân dung (dọc) và 3 cho chế độ phong cảnh (ngang).

Các bộ định tính tài nguyên bạn cần phụ thuộc vào yêu cầu cụ thể của bạn. Khi tạo một tệp tài nguyên mới, có một số bộ định tính trong bảng Available qualifiers mà bạn có thể sử dụng để chọn các điều kiện chính xác:

- **Smallest Screen Width:** Bộ định tính này thường được sử dụng nhiều nhất để chọn cho máy tính bảng. Nó được xác định bởi chiều rộng nhỏ nhất của thiết bị (bất kể định hướng), điều này loại bỏ sự mơ hồ khi nói về "chiều cao" và "chiều rộng" vì một số thiết bị thường được giữ ở chế độ phong cảnh, trong khi những thiết bị khác ở chế độ chân dung. Bất kỳ thiết bị nào có chiều rộng nhỏ nhất ít nhất là 600dp được coi là máy tính bảng.
- **Screen Width:** Chiều rộng màn hình là chiều rộng hiệu quả của thiết bị, bất kể định hướng. Chiều rộng thay đổi khi thiết bị được xoay, vì chiều cao và chiều rộng hiệu quả của thiết bị bị đảo ngược.
- **Screen Height:** Tương tự như **Screen Width**, ngoại trừ việc nó sử dụng chiều cao hiệu quả thay vì chiều rộng hiệu quả.

Để bắt đầu nhiệm vụ này:

1. Tạo một tệp tài nguyên **integers.xml** sử dụng bộ định tính **Smallest Screen Width** với giá trị được đặt thành **600**. Android sử dụng tệp này bất cứ khi nào ứng dụng chạy trên máy tính bảng.
2. Sao chép mã từ tệp **integers.xml (land)** (có số cột là 2) và dán vào tệp mới **integers.xml (sw600dp)**.
3. Tạo một tệp **integers.xml** khác bao gồm cả bộ định tính **Smallest Screen Width** được đặt thành **600** và bộ định tính **Orientation** được đặt thành **Landscape**. Android sẽ sử dụng tệp **integers.xml (sw600dp-land)** khi ứng dụng chạy trên máy tính bảng ở chế độ phong cảnh.
4. Sao chép mã từ tệp **integers.xml (land)** và dán vào tệp mới **integers.xml (sw600dp-land)**.
5. Thay đổi biến `grid_column_count` thành 3 trong tệp **integers.xml (sw600dp-land)**.
6. Chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng, và xoay nó sang chế độ phong cảnh. Ứng dụng nên hiển thị ba cột thẻ, như hình đầu tiên dưới đây. Xoay nó sang chế độ chân dung, và ứng dụng nên hiển thị hai cột thẻ, như hình thứ hai dưới đây. Với các tệp tài nguyên định tính này, ứng dụng sử dụng không gian màn hình hiệu quả hơn rất nhiều.

Mẹo: Nếu ứng dụng của bạn sử dụng nhiều tệp tài nguyên, Android sẽ sử dụng tệp tài nguyên với bộ định tính tài nguyên cụ thể nhất trước. Ví dụ, nếu một giá trị được xác định trong tệp **integers.xml** cho cả bộ định tính **Smallest Screen Width** và với **Orientation** được đặt thành **Landscape**, Android sẽ sử dụng giá trị cho **Smallest Screen Width**. Thứ tự ưu

tiên cho các bộ định tính tài nguyên và tệp tài nguyên được mô tả trong Bảng 2 trong Tổng quan về tài nguyên ứng dụng.

2.2 Cập Nhật Kiểu Mục Danh Sách Cho Máy Tính Bảng

Tại thời điểm này, ứng dụng của bạn thay đổi số cột trong GridLayoutManager để phù hợp với định hướng của thiết bị và tối đa hóa việc sử dụng màn hình. Tuy nhiên, các phần tử TextView xuất hiện với kích thước đúng trên màn hình điện thoại giờ đây lại quá nhỏ cho 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 các bộ định tính tài nguyên.

Lưu ý: Bạn cũng có thể tạo các tệp bố cục thay thế với các bộ định tính tài nguyên thích hợp và thay đổi kiểu của các phần tử TextView trong đó. Tuy nhiên, điều này sẽ yêu cầu nhiều mã lặp lại hơn, vì hầu hết thông tin bố cục là giống nhau bất kể bạn sử dụng thiết bị nào, vì vậy bạn chỉ trích xuất các thuộc tính sẽ thay đổi.

Thực hiện các bước sau để thêm các kiểu TextAppearance:

1. Mở tệp **styles.xml** và thêm các kiểu sau:

2. Tạo một tệp tài nguyên mới có tên **styles.xml** sử dụng bộ định tính **Smallest Screen Width** với giá trị là **600** cho máy tính bảng.

3. Sao chép tất cả các kiểu từ tệp **styles.xml** gốc vào tệp mới **styles.xml (sw600dp)**.

4. Trong tệp **styles.xml (sw600dp)**, thay đổi parent của kiểu SportsTitle thành **TextAppearance.AppCompat.Display1**:

5. Kiểu Display1 được định nghĩa trước trong Android sử dụng giá trị textColorSecondary từ chủ đề hiện tại (ThemeOverlay.AppCompat.Dark), trong trường hợp này là màu xám nhạt. Màu xám nhạt không hiển thị tốt trên các hình ảnh banner 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**:

Dấu hỏi cho Android runtime biết tìm giá trị trong chủ đề được áp dụng cho View. Trong ví dụ này, chủ đề là ThemeOverlay.AppCompat.Dark, trong đó thuộc tính textColorPrimary là màu trắng.

6. Thay đổi parent của kiểu SportsDetailText thành **TextAppearance.AppCompat.Headline**.

7. Để cập nhật kiểu của các phần tử TextView, mở tệp **list_item.xml** và thay đổi thuộc tính style của TextView tiêu đề thành **@style/SportsTitle**:

8. Thay đổi thuộc tính style của các phần tử newsTitle và subTitle thành **@style/SportsDetailText**:

9. Chạy ứng dụng của bạn trên một máy tính bảng hoặc trình giả lập máy tính bảng. Mỗi mục danh sách bây giờ sẽ có kích thước văn bản lớn hơn trên máy tính bảng.

2.3 Cập Nhật Kiểu Chi Tiết Thể Thao Cho Máy Tính Bảng

Bạn đã sửa hiển thị cho MainActivity, danh sách tất cả các phần tử CardView thể thao. Tuy nhiên, DetailActivity vẫn có cùng kích thước phông chữ trên máy tính bảng và điện thoại.

1. Thêm kiểu sau vào tệp **styles.xml** cho tiêu đề chi tiết:

2. Thêm kiểu sau vào tệp **styles.xml (sw600dp)** cho tiêu đề chi tiết:

3. Mở **activity_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:

4. Trong **activity_detail.xml**, thay đổi thuộc tính style của phần tử titleDetail TextView thành kiểu SportsDetailTitle mới mà bạn đã tạo:

5. Chạy ứng dụng của bạn. Tất cả văn bản bây giờ lớn hơn trên máy tính bảng, điều này cải thiện đáng kể trải nghiệm người dùng của ứng dụng bạn.

Task 3: Địa Phương Hóa Ứng Dụng Của Bạn

Một "locale" đại diện cho một khu vực địa lý, chính trị hoặc văn hóa cụ thể của thế giới. Các bộ định tính tài nguyên có thể được sử dụng để cung cấp các tài nguyên thay thế dựa trên locale của người dùng. Cũng như đối với định hướng và chiều rộng màn hình, Android cung cấp khả năng bao gồm các tệp tài nguyên riêng biệt cho các locale khác nhau. Trong bước này, bạn sẽ sửa đổi tệp strings.xml của mình để trở nên quốc tế hóa hơn.

3.1 Thêm tệp strings.xml địa phương hóa

Bạn có thể đã 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 "soccer" để đại diện cho một môn thể thao được biết đến là "football" ở khắp nơi khác trên thế giới.

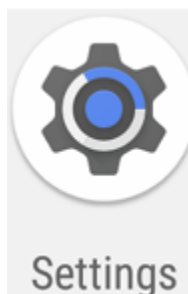
Để làm cho ứng dụng của bạn quốc tế hóa hơn, bạn có thể cung cấp một tệp `strings.xml` cụ thể cho locale. Tệp tài nguyên thay thế này sẽ hiển thị từ "soccer" cho người dùng ở Hoa Kỳ. Tệp `strings.xml` tổng quát sẽ hiển thị từ "football" cho người dùng ở tất cả các locale khác.

1. Tạo một tệp tài nguyên values mới.
2. Gọi tệp là **strings.xml** và chọn **Locale** từ danh sách các bộ định tính có sẵn. Các bảng Language và Specific Region Only sẽ xuất hiện.
3. Trong bảng **Language**, chọn **en: English**.
4. Trong bảng **Specific Region Only**, chọn **US: United States** và nhấn **OK**. Android Studio sẽ tạo một thư mục values cụ thể trong các thư mục dự án của bạn cho locale của Hoa Kỳ, gọi là `values-en-rUS`. Trong khung **Project > Android**, tệp `strings.xml` trong thư mục này sẽ xuất hiện là **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 từ tệp **strings.xml** tổng quát (bây giờ nằm trong thư mục **strings.xml**) vào **strings.xml (en-rUS)**.
6. Trong tệp **strings.xml** tổng quát, thay đổi mục **Soccer** trong mảng `sports_titles` thành **Football**, và thay đổi văn bản **Soccer news** trong mảng `sports_info` thành **Football news**.

3.2 Chạy ứng dụng trong các locale khác nhau

Để thấy sự khác biệt cụ thể theo locale, bạn có thể khởi động thiết bị hoặc trình giả lập của mình, và thay đổi ngôn ngữ và locale của nó thành tiếng Anh Hoa Kỳ (nếu chưa được thiết lập). Trong tiếng Anh Hoa Kỳ, bạn sẽ thấy "Soccer". Bạn có thể sau đó chuyển sang bất kỳ ngôn ngữ và locale nào khác ngoài tiếng Anh Hoa Kỳ, và chạy lại ứng dụng. Bạn sẽ thấy "Football".

1. Để chuyển đổi ngôn ngữ ưa thích trong thiết bị hoặc trình giả lập của bạn, mở ứng dụng Cài đặt. Nếu thiết bị Android của bạn đang ở ngôn ngữ khác, hãy tìm biểu tượng bánh răng:



2. Tìm cài đặt **Languages & input** trong ứng dụng Cài đặt, và chọn **Languages**.

Hãy nhớ biểu tượng quả địa cầu cho lựa chọn **Languages & input**, để bạn có thể tìm thấy nó lại nếu bạn chuyển sang một ngôn ngữ mà bạn không hiểu. **Languages** là lựa chọn đầu tiên trên màn hình **Languages & input**.



3. Đối với thiết bị và trình giả lập chạy phiên bản Android trước Android 7, chọn **Language** trên màn hình **Languages & input**, chọn một ngôn ngữ và locale như **Français (France)**, và bỏ qua các bước tiếp theo.

(Trong các phiên bản Android trước Android 7, người dùng chỉ có thể chọn một ngôn ngữ. Trong Android 7 và các phiên bản mới hơn, người dùng có thể chọn nhiều ngôn ngữ và sắp xếp chúng theo sở thích. Ngôn ngữ chính được đánh số 1, như được hiển thị trong hình ảnh sau, tiếp theo là các ngôn ngữ có mức độ ưu tiên thấp hơn.)

4. Đối với thiết bị và trình giả lập chạy Android 7 hoặc mới hơn, chọn **Languages** trên màn hình **Languages & input**, chọn một ngôn ngữ như **Français (France)**, và sử dụng biểu tượng di chuyển ở bên phải màn hình **Language preferences** để kéo **Français (France)** lên đầu danh sách.

5. Chạy ứng dụng với thiết bị hoặc trình giả lập của bạn. Trong tiếng Anh Hoa Kỳ, bạn sẽ thấy "Soccer".

6. Chuyển sang bất kỳ ngôn ngữ và locale nào khác ngoài tiếng Anh Hoa Kỳ, và chạy lại ứng dụng. Bạn sẽ thấy "Football".

Ví dụ này không cho thấy một từ được dịch cho "Football" tùy thuộc vào ngôn ngữ. Để tìm hiểu về việc địa phương hóa một ứng dụng với các bản dịch, hãy xem [Phát triển Android Nâng Cao — Các Bài Thực Hành](#).

Giải pháp mã

Dự án Android Studio: [MaterialMe-Resource](#)

Thách thức lập trình

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

Thách thức 1: Hóa ra rằng một số quốc gia khác ngoài Hoa Kỳ sử dụng từ "soccer" thay vì "football". Nghiên cứu các quốc gia này và thêm tài nguyên chuỗi được địa phương hóa cho chúng.

Thách thức 2: Sử dụng các kỹ thuật địa phương hóa mà bạn đã học trong Nhiệm vụ 3 kết hợp với Google Translate 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 danh sách cuộn hai chiều.
- Bạn có thể thay đổi động số lượng cột trong một GridLayoutManager.
- Hệ thống runtime Android sử dụng các tệp cấu hình thay thế, tùy thuộc vào môi trường runtime của thiết bị chạy ứng dụng của bạn. Ví dụ, runtime có thể sử dụng các tệp cấu hình thay thế cho các bố cục thiết bị khác nhau, kích thước màn hình, ngôn ngữ, quốc gia hoặc loại bàn phím.
- Trong mã của bạn, bạn tạo các tài nguyên thay thế này để hệ thống Android có thể sử dụng. Các tài nguyên được đặt trong các tệp có tên chứa các định danh tài nguyên.
- Định dạng cho một thư mục chứa các tệp tài nguyên thay thế là <resource_name> - <qualifier>.
- Bạn có thể định danh bất kỳ tệp nào trong thư mục res theo cách này.

Khái niệm liên quan

Khái niệm liên quan có trong [5.3: Tài nguyên cho các bố cục thích ứng](#).

Tìm hiểu thêm

Tài liệu Android Studio: [Android Studio User Guide](#)

Tài liệu phát triển Android:

- Resources Overview
- Providing Resources
- Localizing with Resources
- LinearLayoutManager

- GridLayoutManager
- Supporting Multiple Screens

Thiết kế Material

- Material Design for Android
- Material Design Guidelines

Bài tập về nhà

Xây dựng và chạy một ứng dụng

Chỉnh sửa ứng dụng [RecyclerView](#) để sử dụng GridLayoutManager với các số lượng cột sau:

- Đối với điện thoại: 1 cột ở chế độ dọc, 2 cột ở chế độ ngang
- Đối với máy tính bảng: 2 cột ở chế độ dọc, 3 cột ở chế độ ngang

Dưới đây là ảnh chụp màn hình cho thấy một RecyclerView được định danh tài nguyên trên điện thoại ở chế độ dọc:

Dưới đây là ảnh chụp màn hình cho thấy một RecyclerView được định danh tài nguyên trên điện thoại ở chế độ ngang:

Dưới đây là ảnh chụp màn hình cho thấy một RecyclerView được định danh tài nguyên trên máy tính bảng ở chế độ ngang:

Trả lời câu hỏi

Câu hỏi 1

Định danh tài nguyên nào được sử dụng phổ biến 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 cho việc dịch sang tiếng Pháp cho Canada? Chọn một:

- res/values-fr-rFR/
- res/values-ca-rFR/
- res/values-fr-rCA/
- res/values-en-rFR/

Câu hỏi 3

Thư mục nào dành cho các tệp XML chứa chuỗi, số nguyên và màu sắc? Chọn một:

- res/layout
- res/mipmap
- res/raw
- res/values

Nội 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 giá trị được định danh tài nguyên chứa số nguyên cho số lượng cột.
- Ứng dụng sử dụng getResources().getInteger() để lấy giá trị từ tệp tài nguyên, sau đó sử dụng giá trị đó làm số lượng cột cho bố cục lưới.

Bài 3) Kiểm thử giao diện người dùng

3.1) Espresso cho việc kiểm tra UI

Giới thiệu

Là một nhà phát triển, việc kiểm tra các tương tác của người dùng trong ứng dụng của bạn là rất quan trọng để đảm bảo rằng người dùng không gặp phải những kết quả bất ngờ hoặc có trải nghiệm kém với ứng dụng

Bạn có thể kiểm tra giao diện người dùng (UI) của ứng dụng một cách thủ công bằng cách chạy ứng dụng và thử nghiệm UI. Nhưng đối với một ứng dụng phức tạp, bạn không thể bao trùm tất cả các tổ hợp tương tác của người dùng trong tất cả các chức năng của ứng

dụng. Bạn cũng sẽ phải lặp lại các bài kiểm tra thủ công này cho các cấu hình thiết bị khác nhau trong một trình giả lập và trên các thiết bị phần cứng khác nhau.

Khi bạn tự động hóa việc kiểm tra các tương tác UI, bạn tiết kiệm thời gian và việc kiểm tra trở nên có hệ thống. Bạn có thể sử dụng các bộ bài kiểm tra tự động để thực hiện tất cả các tương tác UI một cách tự động, điều này giúp dễ dàng hơn trong việc chạy các bài kiểm tra cho các cấu hình thiết bị khác nhau. Để xác minh rằng UI của ứng dụng hoạt động chính xác, bạn nên hình thành thói quen tạo các bài kiểm tra UI khi bạn lập trình.

Espresso là một framework kiểm tra cho Android giúp dễ dàng viết các bài kiểm tra UI đáng tin cậy cho ứng dụng. Framework này, là một phần của Android Support Repository, cung cấp các API để viết các bài kiểm tra UI nhằm mô phỏng các tương tác của người dùng trong ứng dụng—from việc nhấn nút và điều hướng giữa các view, đến việc chọn các mục menu và nhập dữ liệu.

Những gì bạn nên biết trước

Bạn nên có khả năng:

- Tạo và chạy ứng dụng trong Android Studio.
- Kiểm tra Android Support Repository và cài đặt nếu cần thiết.
- Tạo và chỉnh sửa các phần tử UI bằng trình chỉnh sửa bố cục và XML.
- Truy cập các phần tử UI từ mã của bạn.
- Thêm một trình xử lý nhấp chuột cho một Button.

Những gì bạn sẽ học

Bạn sẽ học cách:

- Thiết lập Espresso trong dự án ứng dụng của bạn.
- Viết các bài kiểm tra Espresso.
- Kiểm tra đầu vào của người dùng và kiểm tra đầu ra chính xác.
- Tìm một spinner, nhấn một trong các mục của nó và kiểm tra đầu ra chính xác.
- Sử dụng chức năng **Record Espresso Test** trong Android Studio.

Những gì bạn sẽ làm

- Chỉnh sửa một dự án để tạo các bài kiểm tra Espresso.
- Kiểm tra đầu vào và đầu ra văn bản của ứng dụng.
- Kiểm tra nhấp vào một mục spinner và kiểm tra đầu ra của nó.

- Ghi lại một bài kiểm tra Espresso cho một RecyclerView.

Tổng quan về ứng dụng

Mẹo: Để tìm hiểu về việc kiểm tra ứng dụng Android, hãy xem [Test your app](#).

Trong thực hành này, bạn sẽ chỉnh sửa dự án [TwoActivities](#) từ bài học trước. Bạn sẽ thiết lập Espresso trong dự án để kiểm tra, và sau đó kiểm tra chức năng của ứng dụng.

Ứng dụng TwoActivities cho phép người dùng nhập văn bản vào một trường văn bản và nhấn nút Send, như được hiển thị ở bên trái của hình dưới đây. Trong Activity thứ hai, người dùng sẽ xem văn bản họ đã nhập, như được hiển thị ở bên phải của hình dưới đây.

Task 1: Thiết lập Espresso trong dự án của bạn

Để sử dụng Espresso, bạn phải đảm bảo rằng Android Support Repository đã được cài đặt cùng với Android Studio. Bạn cũng có thể cần cấu hình Espresso trong dự án của mình

Trong nhiệm vụ này, bạn sẽ kiểm tra xem repository đã được cài đặt chưa. Nếu chưa, bạn sẽ cài đặt nó.

1.1 Kiểm tra Android Support Repository

1. Tải dự án [TwoActivities](#) từ bài học trước về việc tạo và sử dụng một Activity.

2. Mở dự án trong Android Studio, và chọn **Tools > Android > SDK Manager**.

Bảng Android SDK **Default Preferences** xuất hiện.

3. Nhấp vào tab **SDK Tools** và mở rộng **Support Repository**.

4. Tìm **Android Support Repository** trong danh sách.

Nếu **Installed** xuất hiện trong cột Status, bạn đã sẵn sàng. Nhấn **Cancel**.

Nếu **Not installed** hoặc **Update Available** xuất hiện, hãy đánh dấu vào ô bên cạnh **Android Support Repository**. Một biểu tượng tải xuống sẽ xuất hiện bên cạnh ô kiểm. Nhấn **OK**.

5. Nhấn **OK** lần nữa, và sau đó **Finish** khi repository đã được cài đặt.

1.2 Cấu hình Espresso cho dự án

Khi bạn bắt đầu một dự án cho định dạng điện thoại và máy tính bảng với **API 15: Android 4.0.3 (Ice Cream Sandwich)** làm SDK tối thiểu, Android Studio tự động bao gồm các phụ thuộc cần thiết để sử dụng Espresso. Để thực thi các bài kiểm tra, Espresso và UI

Automator sử dụng [JUnit](#) làm framework kiểm tra. JUnit là framework kiểm tra đơn vị phổ biến và được sử dụng rộng rãi nhất cho Java. Lớp kiểm tra của bạn sử dụng Espresso hoặc UI Automator nên được viết như một lớp kiểm tra JUnit 4.

Lưu ý: Phiên bản JUnit hiện tại nhất là JUnit 5. Tuy nhiên, cho mục đích sử dụng Espresso hoặc UI Automator, phiên bản 4.12 được khuyến nghị.

Nếu bạn đã tạo dự án của mình trong một phiên bản Android Studio trước đó, bạn có thể phải tự thêm các phụ thuộc và instrumentation. Để thêm các phụ thuộc một cách thủ công, hãy làm theo các bước sau:

1. Mở dự án TwoActivities, hoặc nếu bạn muốn, hãy sao chép dự án trước rồi mở bản sao.

2. Mở tệp **build.gradle (Module: app)**.

3. Kiểm tra xem phần sau có được bao gồm (cùng với các phụ thuộc khác) trong phần dependencies không:

4. Android Studio cũng thêm câu lệnh instrumentation sau vào cuối phần defaultConfig của một dự án mới:

Nếu tệp không bao gồm câu lệnh instrumentation trên, hãy nhập nó vào cuối phần defaultConfig.

Instrumentation là một tập hợp các phương thức điều khiển, hoặc hooks, trong hệ thống Android. Những hooks này kiểm soát một thành phần Android độc lập với vòng đời bình thường của thành phần đó. Chúng cũng điều khiển cách Android tải ứng dụng. Việc sử dụng instrumentation cho phép các bài kiểm tra gọi các phương thức trong ứng dụng, và 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 các tệp Gradle ở góc trên bên phải của cửa sổ.

CHƯƠNG 2. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

- 1.1) AsyncTask**
- 1.2) AsyncTask và AsyncTaskLoader**
- 1.3) Broadcast receivers**

Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền

- 2.1) Thông báo**
- 2.2) Trình quản lý cảnh báo**
- 2.3) JobScheduler**

CHƯƠNG 3. LƯU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

- 1.1) Shared preferences**
- 1.2) Cài đặt ứng dụng**

Bài 2) Lưu trữ dữ liệu với Room

- 2.1) Room, LiveData và ViewModel**
- 2.2) Room, LiveData và ViewModel**