

**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**

<b>CHƯƠNG 1.</b>	<b>LÀM QUEN .....</b>	<b>3</b>
Bài 1)	Tạo ứng dụng đầu tiên .....	3

1.1)	Android Studio và Hello World .....	3
1.2)	Giao diện người dùng tương tác đầu tiên.....	5
1.3)	Trình chỉnh sửa bố cục.....	5
1.4)	Văn bản và các chế độ cuộn.....	5
1.5)	Tài nguyên có sẵn .....	5
Bài 2)	Activities .....	5
2.1)	Activity và Intent .....	5
2.2)	Vòng đời của Activity và trạng thái .....	5
2.3)	Intent ngầm định .....	5
Bài 3)	Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ.....	5
3.1)	Trình gỡ lỗi .....	5
3.2)	Kiểm thử đơn vị .....	5
3.3)	Thư viện hỗ trợ .....	5
CHƯƠNG 2.	TRẢI NGHIỆM NGƯỜI DÙNG .....	5
Bài 1)	Tương tác người dùng.....	5
1.1)	Hình ảnh có thể chọn .....	5
1.2)	Các điều khiển nhập liệu.....	34
1.3)	Menu và bộ chọn.....	52
1.4)	Điều hướng người dùng .....	71
1.5)	RecyclerView .....	81
Bài 2)	Trải nghiệm người dùng thú vị .....	85
2.1)	Hình vẽ, định kiểu và chủ đề .....	85
2.2)	Thẻ và màu sắc.....	85
2.3)	Bố cục thích ứng .....	85
Bài 3)	Kiểm thử giao diện người dùng .....	85
3.1)	Espresso cho việc kiểm tra UI .....	85
CHƯƠNG 3.	LÀM VIỆC TRONG NỀN .....	85
Bài 1)	Các tác vụ nền.....	85

1.1)	AsyncTask.....	85
1.2)	AsyncTask và AsyncTaskLoader .....	85
1.3)	Broadcast receivers .....	85
Bài 2)	Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền .....	85
2.1)	Thông báo .....	85
2.2)	Trình quản lý cảnh báo.....	85
2.3)	JobScheduler .....	85
CHƯƠNG 4.	LƯU DỮ LIỆU NGƯỜI DÙNG .....	85
Bài 1)	Tùy chọn và cài đặt.....	85
1.1)	Shared preferences .....	85
1.2)	Cài đặt ứng dụng .....	85
Bài 2)	Lưu trữ dữ liệu với Room .....	85
2.1)	Room, LiveData và ViewModel .....	85
2.2)	Room, LiveData và ViewModel .....	85

3.1) Trình gowx loi .....

## CHƯƠNG 1. LÀM QUEN

### Bài 1) Tạo ứng dụng đầu tiên

#### 1.1) Android Studio và Hello World

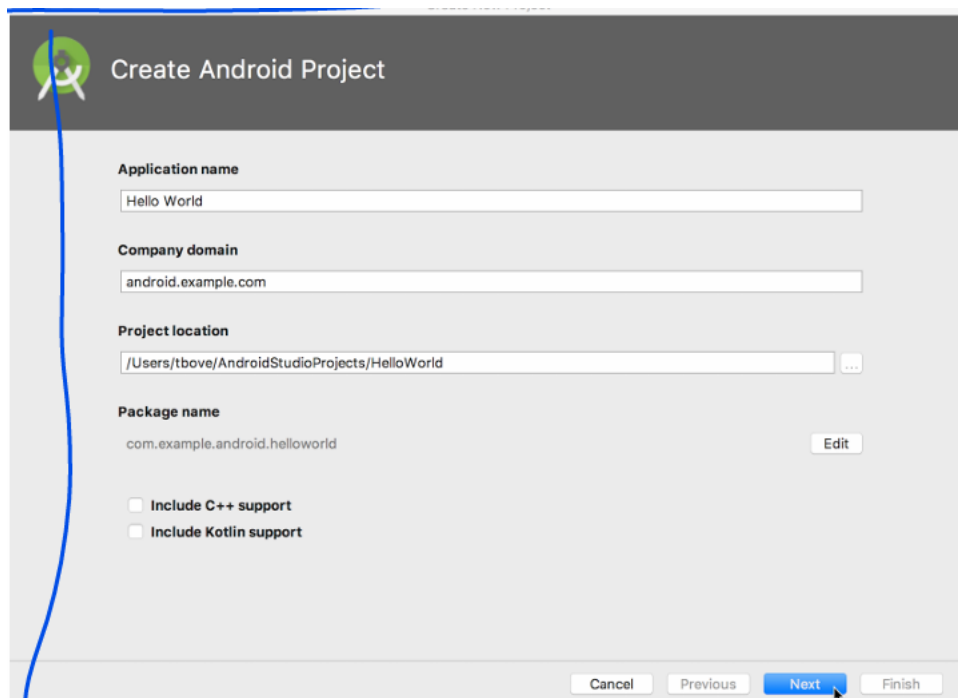
#### **Giới thiệu**

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

#### **Những gì Bạn nên biết**

Bạn nên có khả năng:

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



### Những gì Bạn sẽ cần:

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

### Những gì bạn sẽ học

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

### Những gì bạn sẽ làm

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

- 1.2) **Giao diện người dùng tương tác đầu tiên**
- 1.3) **Trình chỉnh sửa bố cục**
- 1.4) **Văn bản và các chế độ cuộn**
- 1.5) **Tài nguyên có sẵn**

## **Bài 2) Activities**

- 2.1) **Activity và Intent**
- 2.2) **Vòng đời của Activity và trạng thái**
- 2.3) **Intent ngầm định**

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

- 3.1) **Trình gỡ lỗi**
- 3.2) **Kiểm thử đơn vị**
- 3.3) **Thư viện hỗ trợ**

# **CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG**

## **Bài 1) Tương tác người dùng**

### **1.1) Hình ảnh có thể chọn**

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của 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.

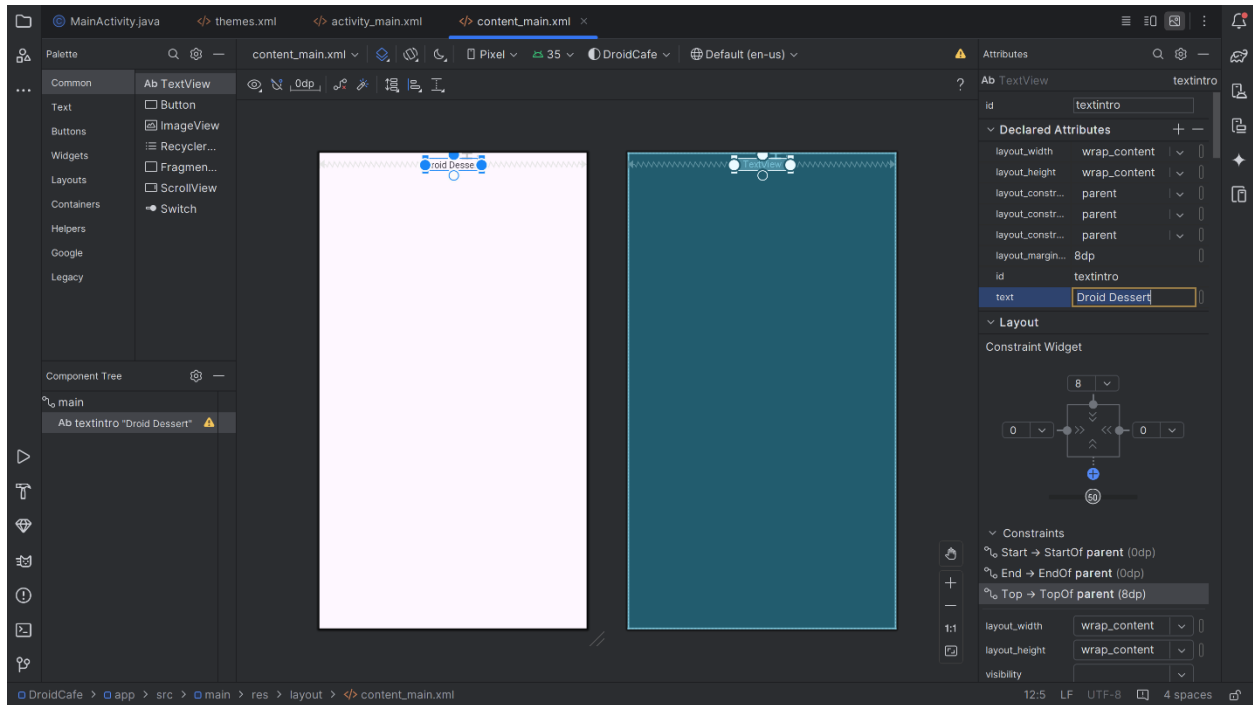
5. Chọn TextView "Hello World" trong layout và mở bảng **Attributes**.

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

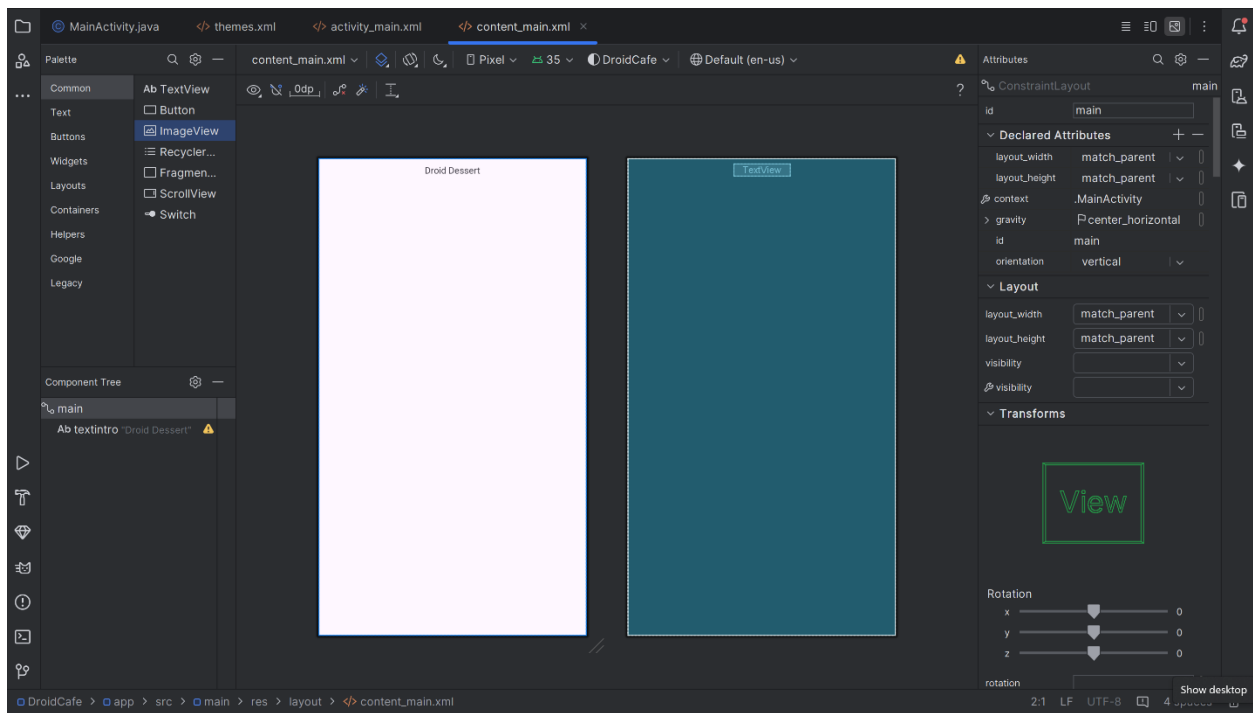
## 1.2 Thêm hình ảnh

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 tín hiệu 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.

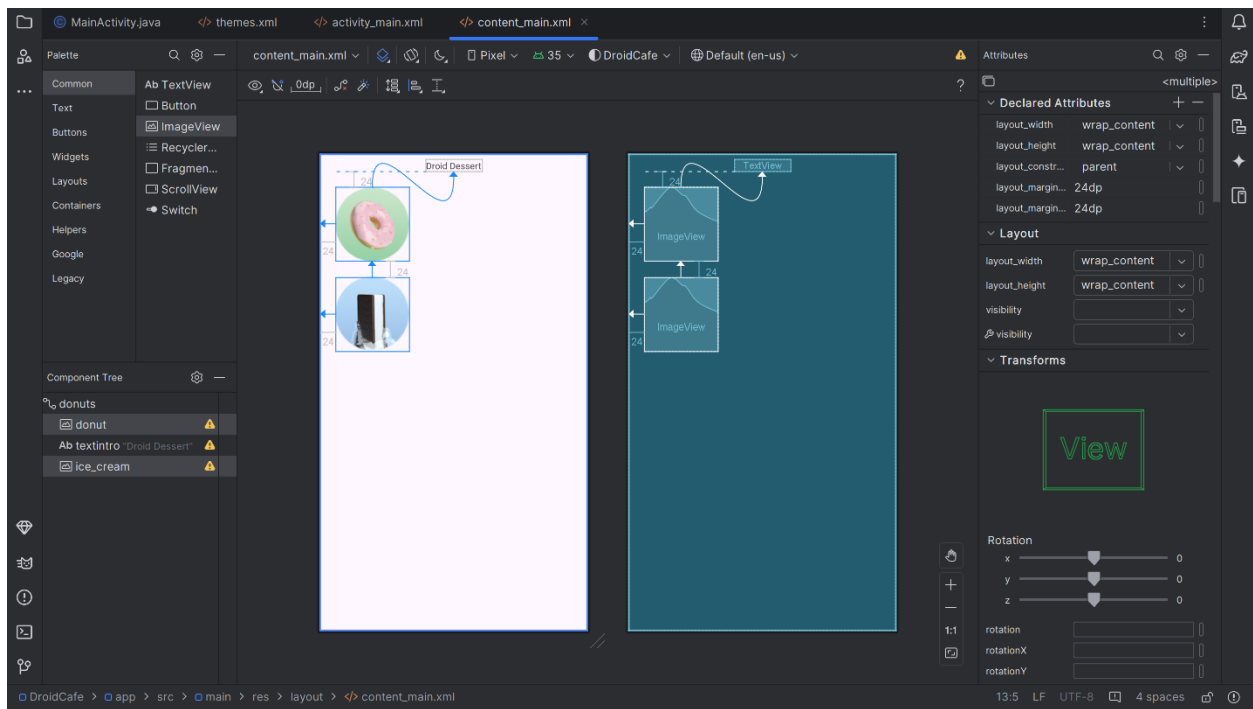




6. Trong bảng thuộc tính, nhập các giá trị sau cho các thuộc tính:

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:

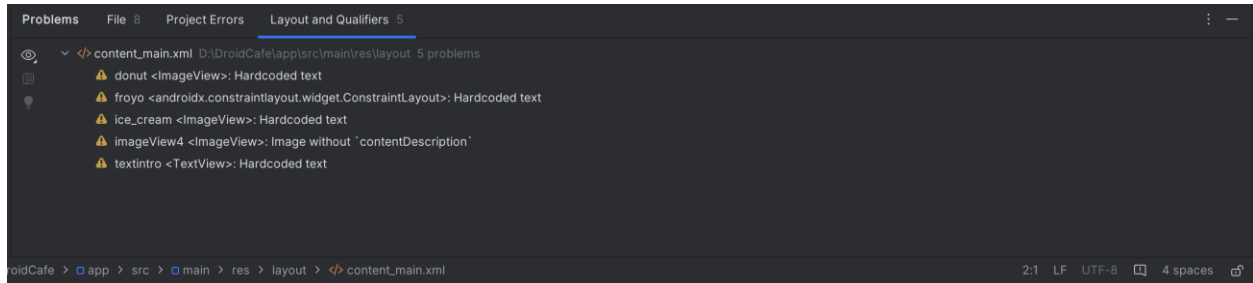
Attribute field	Enter the Following
ID	ice_cream
contentDescription	Ice cream sandwiches have chocolate wafers and vanilla filling. (You can copy/paste the text into the field.)

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.

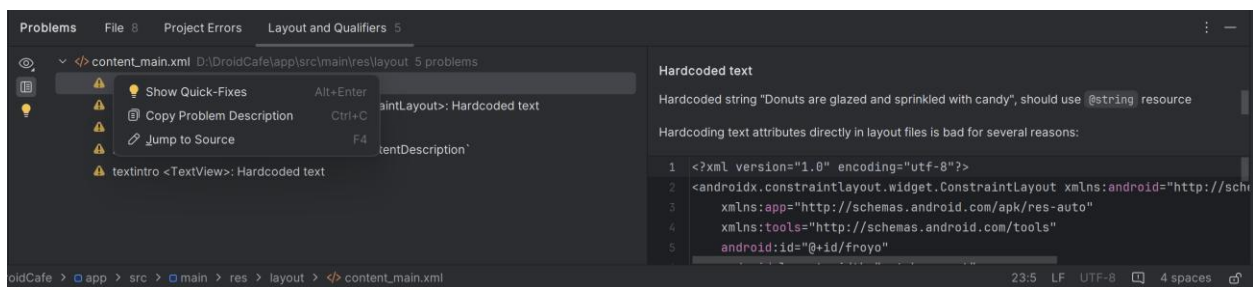
10. Trong bảng thuộc tính, nhập các giá trị sau cho các thuộc tính:

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:



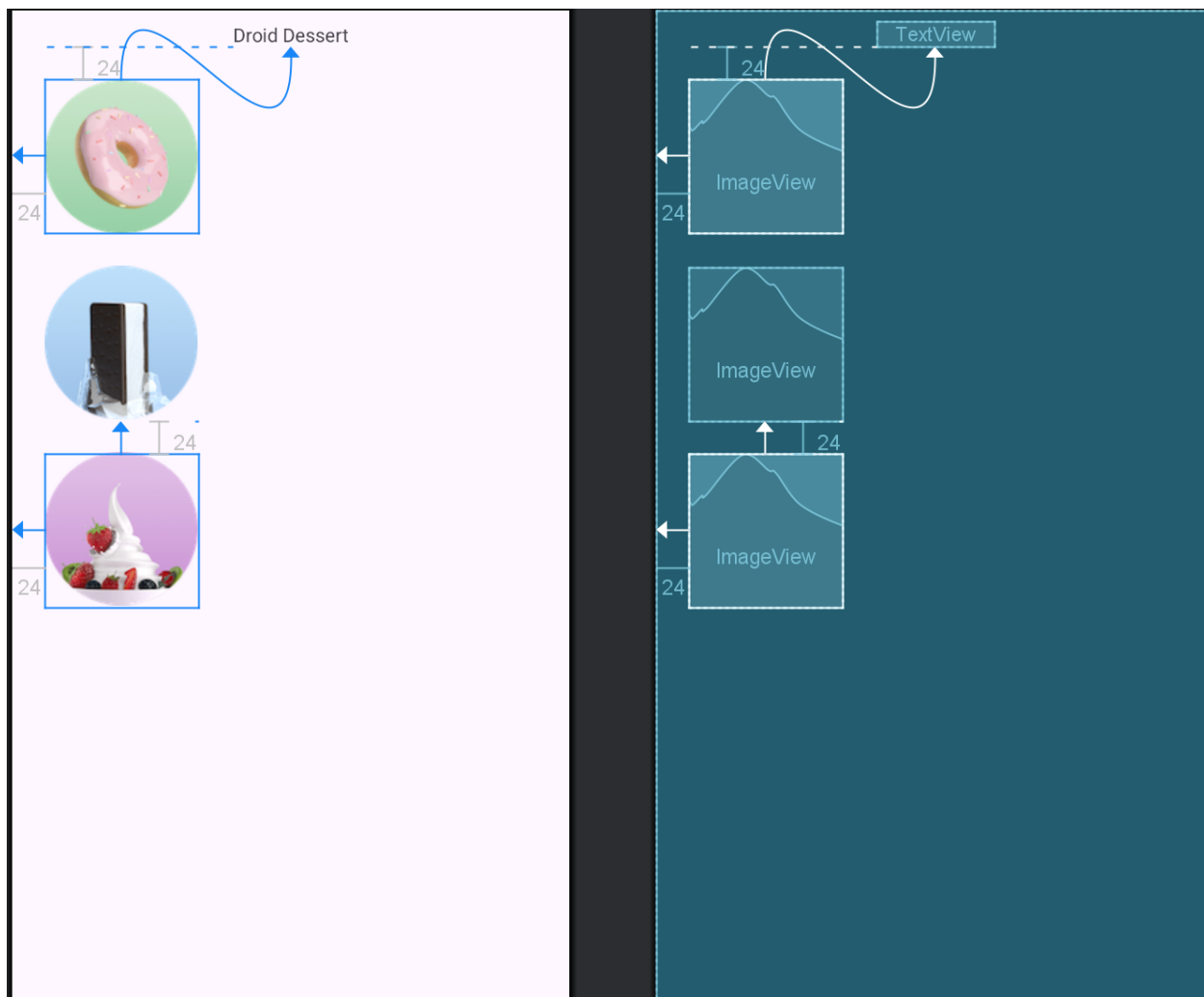
12. Mở rộng từng cảnh báo **Hardcoded text**, cuộn xuống cùng của thông điệp cảnh báo, và nhấp vào nút **Fix** như hình bên dưới:



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

String	Enter the following name:
Donuts are glazed and sprinkled with candy.	donuts
Ice cream sandwiches have chocolate wafers and vanilla filling.	ice_cream_sandwiches
FroYo is premium self-serve frozen yogurt.	froyo

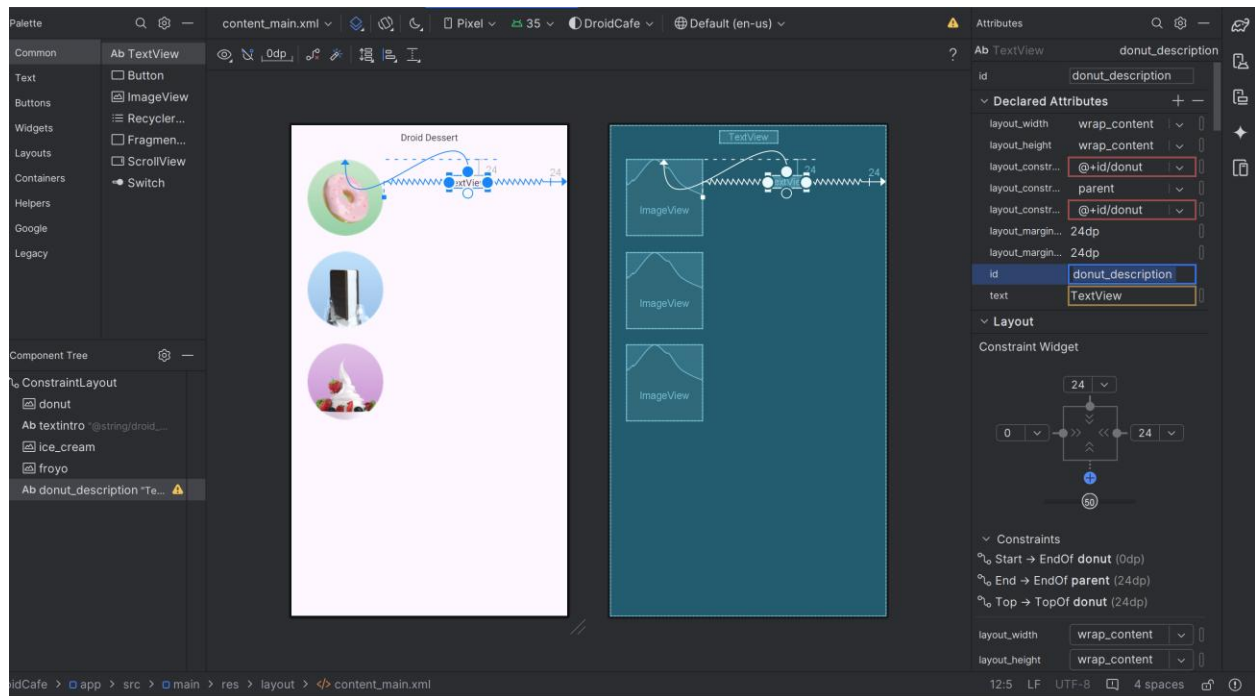
Bố cục sẽ giống hình dưới đây



### 1.3 Thêm miêu tả văn bản

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

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



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

id

donut\_description

Declared Attributes

layout\_width

0dp

|

▼

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

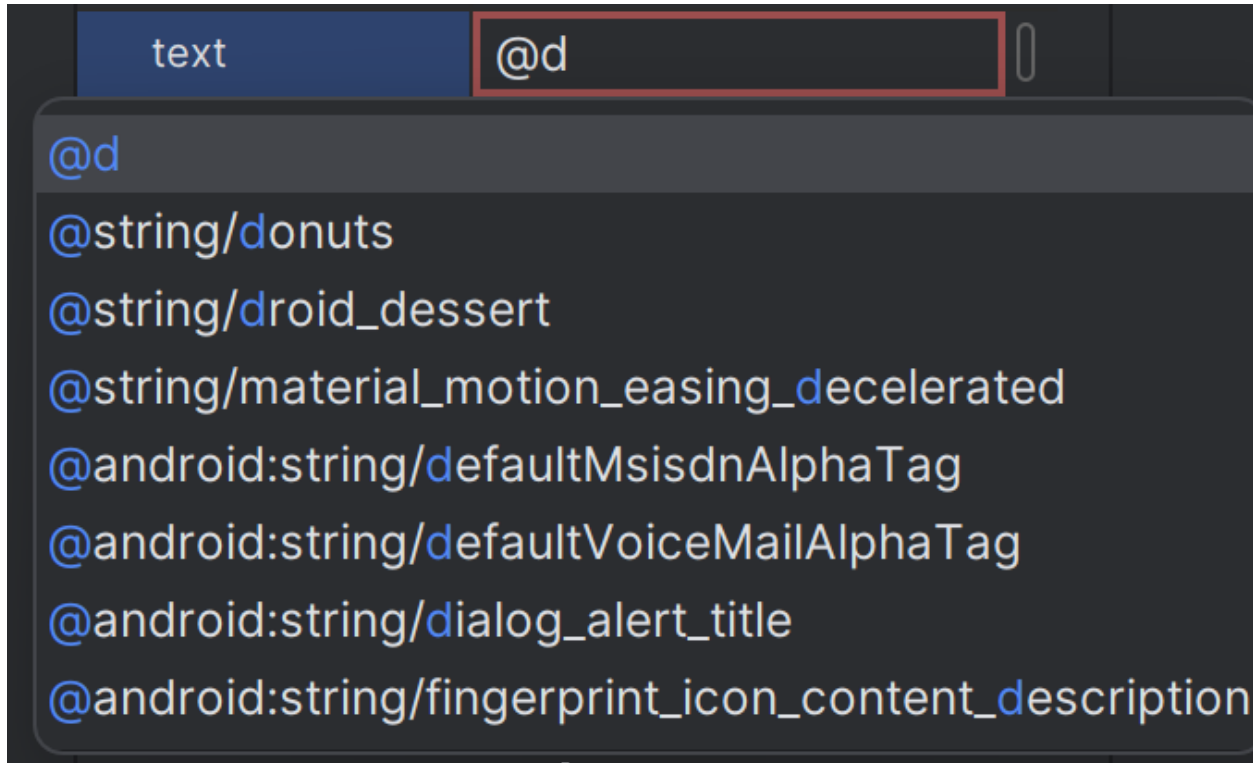
▼

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:

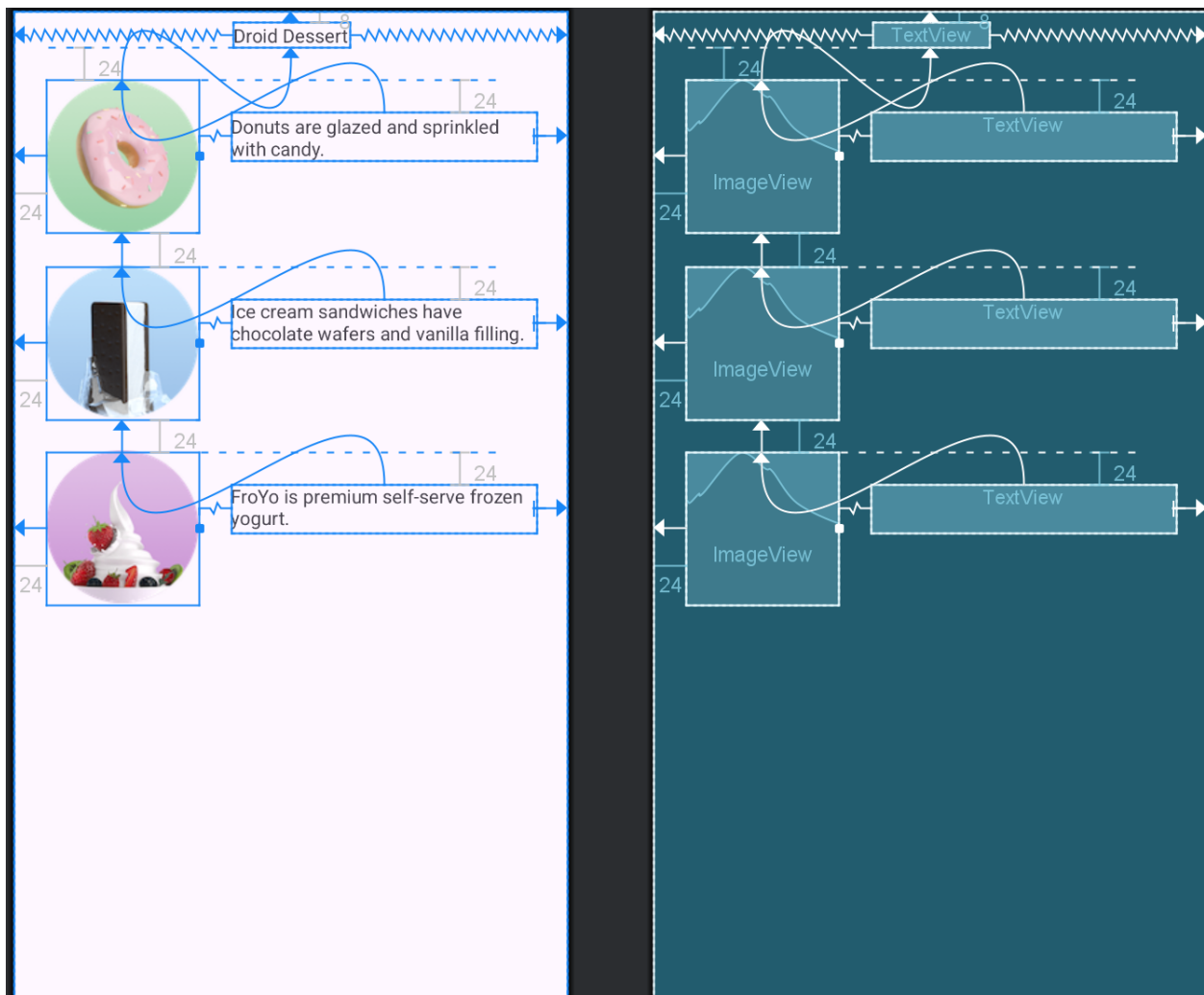
Attribute field	Enter the following:
ID	ice_cream_description
Left, right, and top margins	24
layout_width	match_constraint
text	@string/ice_cream_sandwiches

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 `showIceCreamOrder` 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" />
```

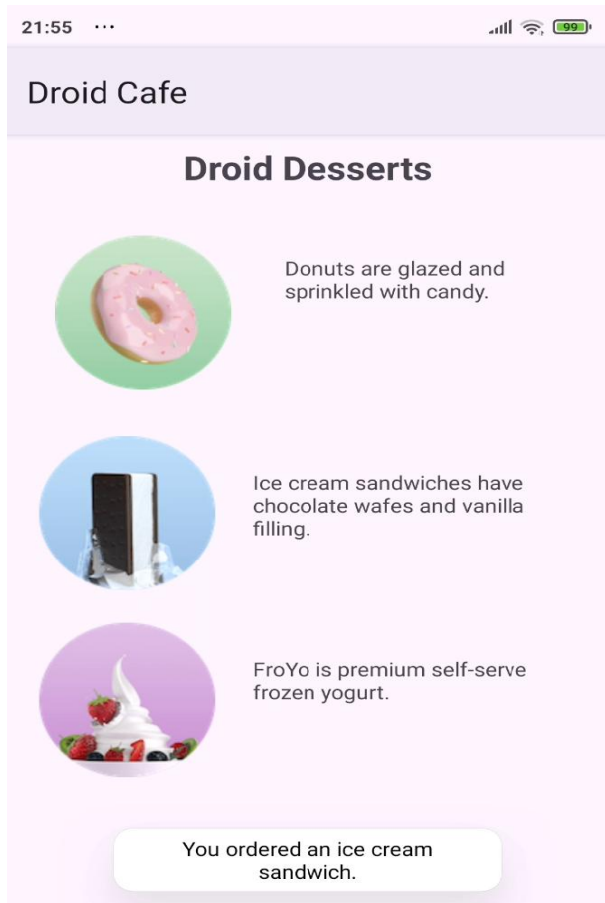
Lưu ý rằng thuộc tính `android:layout_marginStart` trong mỗi `ImageView` được gạch chân bằng màu đỏ. Thuộc tính này xác định "lề" bắt đầu cho `ImageView`, mà bên trái đối với hầu hết các ngôn ngữ nhưng bên phải đối với các ngôn ngữ đọc từ phải sang trái (RTL).

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

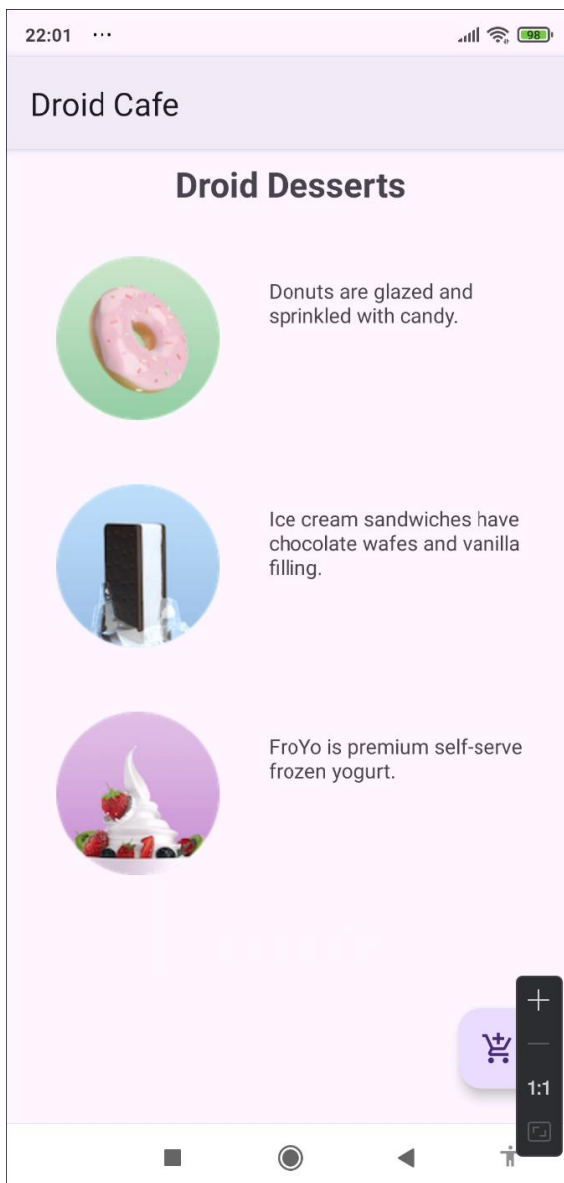
Trong bước này, bạn sẽ thay đổi hành động của FloatingActionButton để mở Activity mới.

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";
```

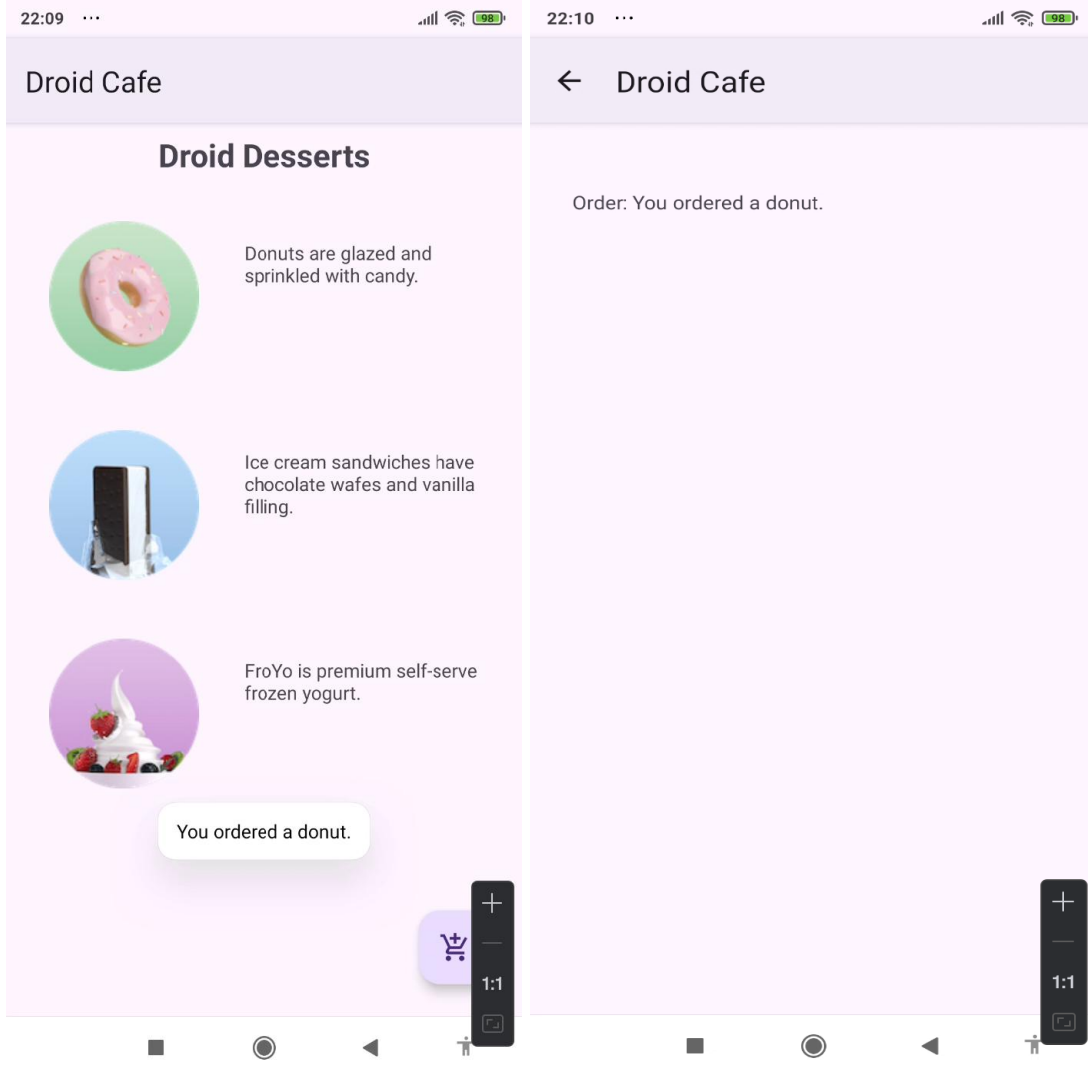
5. Thay đổi phương thức onClick() để bao gồm câu lệnh intent.putExtra trước khi khởi động OrderActivity:

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

- 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);
```

- 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**).
- Hiện 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.

## Nội dung 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"

- 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.
- 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.
- 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.

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

<b>EditText attribute</b>	<b>Value</b>
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**.

- 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.



- 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.

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


<b>EditText attribute</b>	<b>Value</b>
android:id	"@+id/address_text"
android:layout_width	"wrap_content"
android:layout_height	"wrap_content"
android:layout_marginStart	8dp

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

## 7. Chạy ứng dụng.

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

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

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

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

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

## 1.3 Sử dụng bàn phím số cho số điện thoại

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

1.Mở tệp activity\_order.xml nếu nó chưa được mở.

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

TextView attribute	Value
android:id	"@+id/phone_label"

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

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

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

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

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

6.Kiểm tra mã XML cho 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:

- emailAddress: 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
<code>android:id</code>	<code>"@+id/note_label"</code>
<code>android:layout_width</code>	<code>"wrap_content"</code>
<code>android:layout_height</code>	<code>"wrap_content"</code>
<code>android:layout_marginStart</code>	<code>"24dp"</code>
<code>android:layout_marginLeft</code>	<code>"24dp"</code>
<code>android:layout_marginTop</code>	<code>"24dp"</code>
<code>android:text</code>	<code>"Note"</code>
<code>app:layout_constraintStart_toStartOf</code>	<code>"parent"</code>
<code>app:layout_constraintTop_toBottomOf</code>	<code>"@+id/phone_label"</code>

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):

<b>EditText attribute</b>	<b>Value</b>
<code>android:id</code>	<code>"@+id/note_text"</code>
<code>android:layout_width</code>	<code>"wrap_content"</code>
<code>android:layout_height</code>	<code>"wrap_content"</code>
<code>android:layout_marginStart</code>	<code>8dp</code>
<code>android:layout_marginLeft</code>	<code>8dp</code>
<code>android:ems</code>	<code>"10"</code>
<code>android:hint</code>	<code>"@string/enter_note_hint"</code>
<code>android:inputType</code>	<code>"textCapSentences textMultiLine"</code>
<code>app:layout_constraintBaseline_to BaselineOf</code>	<code>"@+id/note_label"</code>
<code>app:layout_constraintStart_toEnd Of</code>	<code>"@+id/note_label"</code>

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"

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

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

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

## 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`.

- Mở tệp **activity\_order.xml** (nếu chưa mở) và tìm một giá trị `onRadioButtonClicked` cho thuộc tính `android:onClick` bị gạch dưới màu đỏ.
- Nhấp vào giá trị `onRadioButtonClicked`, sau đó nhấp vào biểu tượng cảnh báo bóng đỏ ở lề trái.
- 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.
- 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:
- 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.

- 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:

- Mở **activity\_order.xml** và kéo **Spinner** từ bảng **Palette** vào bố cục.

- 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 gắn 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 phân 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 showToast() 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"

<b>Favorites item attribute</b>	<b>Value</b>
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:

<b>Menu item</b>	<b>orderInCategory attribute</b>
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




trạng thái 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"

2. 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).
3. 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

1. Mở **MainActivity**.
2. 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 onDateSet() rỗng, Android Studio tự động thêm các thông tin sau vào khối import ở trên cùng:

Các tham số của onDateSet() 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() rỗng.

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 vượt 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 vượt 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 anh/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 thẻ hiển thị 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 thẻ hiển thị 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 bộ cục sau:

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

Sau khi tạo các bộ 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 bộ 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 bộ cục Activity của bạn. Bên trong DrawerLayout, thêm một View chứa nội dung chính của màn hình (bộ 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 bộ cục của bạn dễ hiểu hơn, hãy sử dụng thẻ include để bao gồm một bộ cục XML trong một bộ cục XML khác.

Hình dưới đây là một biểu diễn trực quan của bộ cục activity\_main.xml và các bộ 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](#)
- Truiron: [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 thành 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 dung 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.

**Bài 2)      Trải nghiệm người dùng thú vị**

**2.1)      Hình vẽ, định kiểu và chủ đề**

**2.2)      Thẻ và màu sắc**

**2.3)      Bố cục thích ứng**

**Bài 3)      Kiểm thử giao diện người dùng**

**3.1)      Espresso cho việc kiểm tra UI**

**CHƯƠNG 3. LÀM VIỆC TRONG NỀN**

**Bài 1)      Các tác vụ nền**

**1.1)      AsyncTask**

**1.2)      AsyncTask và AsyncTaskLoader**

**1.3)      Broadcast receivers**

**Bài 2)      Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền**

**2.1)      Thông báo**

**2.2)      Trình quản lý cảnh báo**

**2.3)      JobScheduler**

**CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG**

**Bài 1)      Tùy chọn và cài đặt**

**1.1)      Shared preferences**

**1.2)      Cài đặt ứng dụng**

**Bài 2)      Lưu trữ dữ liệu với Room**

**2.1)      Room, LiveData và ViewModel**

**2.2)      Room, LiveData và ViewModel**