

**TRƯỜNG ĐẠI HỌC THỦY LỢI**  
**KHOA CÔNG NGHỆ THÔNG TIN**



# **GIÁO TRÌNH**

## **THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG**

Hà Nội, 2.2025

# MỤC LỤC

CHƯƠNG 1.	Làm quen.....	3
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 .....	6
Bài 1)	Tương tác người dùng .....	6
1.1)	Hình ảnh có thể chọn .....	6
1.2)	Các điều khiển nhập liệu .....	6
1.3)	Menu và bộ chọn .....	6
1.4)	Điều hướng người dùng .....	6
1.5)	RecyclerView .....	6
Bài 2)	Trải nghiệm người dùng thú vị.....	6
2.1)	Hình vẽ, định kiểu và chủ đề .....	6
2.2)	Thẻ và màu sắc .....	6

2.3)	Bố cục thích ứng.....	6
Bài 3)	Kiểm thử giao diện người dùng.....	6
3.1)	Espresso cho việc kiểm tra UI .....	6
CHƯƠNG 3. Làm việc trong nền .....		6
Bài 1)	Các tác vụ nền.....	6
1.1)	AsyncTask .....	6
1.2)	AsyncTask và AsyncTaskLoader .....	6
1.3)	Broadcast receivers .....	6
Bài 2)	Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	6
2.1)	Thông báo .....	6
2.2)	Trình quản lý cảnh báo .....	6
2.3)	JobScheduler.....	6
CHƯƠNG 4. Lưu dữ liệu người dùng .....		7
Bài 1)	Tùy chọn và cài đặt.....	7
1.1)	Shared preferences.....	7
1.2)	Cài đặt ứng dụng.....	21
Bài 2)	Lưu trữ dữ liệu với Room .....	<b>Error! Bookmark not defined.</b>
2.1)	Room, LiveData và ViewModel.....	65
2.2)	Room, LiveData và ViewModel.....	83
3.1)	Trình gỡ lỗi .....	

## CHƯƠNG 1. LÀM QUEN

### Bài 1) Tạo ứng dụng đầu tiên

#### 1.1) Android Studio và Hello World

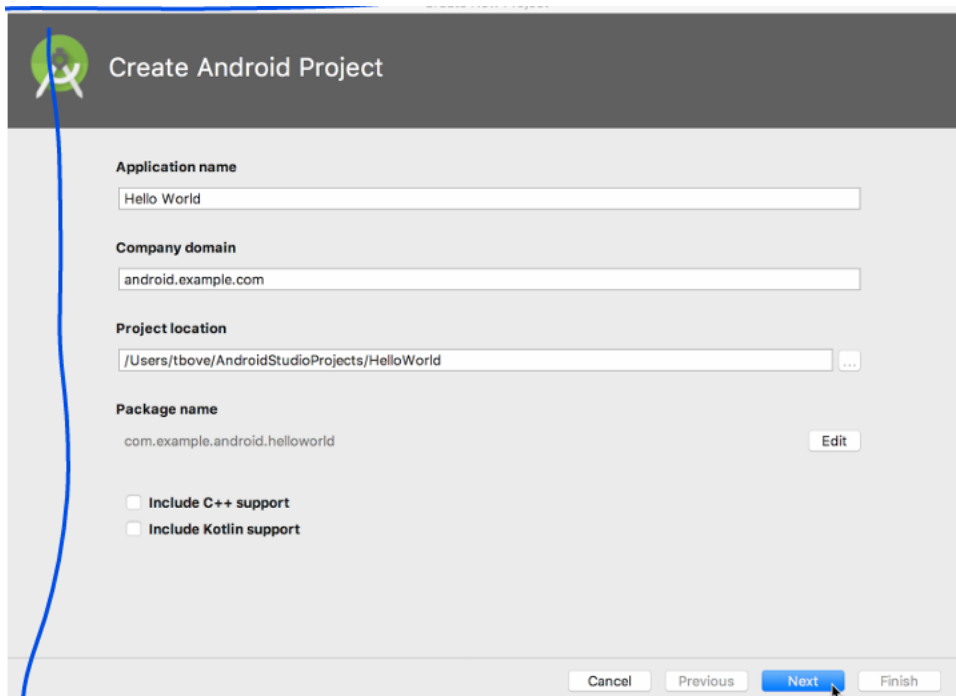
### Giới thiệu

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**
- 1.2) Các điều khiển nhập liệu**
- 1.3) Menu và bộ chọn**
- 1.4) Điều hướng người dùng**
- 1.5) RecyclerView**

### **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

#### ❖ Giới thiệu

- Shared preferences cho phép bạn lưu một lượng nhỏ dữ liệu dưới dạng khóa/giá trị trong một tệp trên thiết bị. Để xử lý tệp tùy chọn cũng như đọc, ghi và quản lý dữ liệu, hãy sử dụng lớp SharedPreferences. Khung Android tự quản lý tệp SharedPreferences. Tất cả các thành phần trong ứng dụng của bạn đều truy cập được vào tệp này, nhưng các ứng dụng khác thì không thể truy cập được.
- Dữ liệu bạn lưu vào shared preferences khác với dữ liệu trạng thái hoạt động đã lưu, cái mà bạn đã học ở chương trước.
  - Dữ liệu ở trạng thái phiên bản hoạt động đã lưu được giữ lại trên các phiên bản hoạt động trong cùng một phiên người dùng.
  - Shared preference vẫn tồn tại qua các phiên người dùng. Shared preferences vẫn tồn tại nếu ứng dụng của bạn dừng và khởi động lại hoặc nếu thiết bị khởi động lại.
- Chỉ sử dụng shared preferences khi bạn cần lưu một lượng nhỏ dữ liệu dưới dạng cặp khóa/giá trị đơn giản. Để quản lý lượng dữ liệu ứng dụng cố định lớn hơn, hãy sử dụng phương thức lưu trữ như thư viện phòng hoặc cơ sở dữ liệu SQL.

### Những điều bạn nên biết:

Bạn nên quen với:

- Tạo, xây dựng và chạy ứng dụng trong android studio.
- Thiết kế bố cục với các nút và văn bản.
- Sử dụng kiểu và chủ đề.
- Lưu và khôi phục trạng thái phiên bản hoạt động.

Bạn sẽ học được gì

Bạn sẽ học cách nào để:

- Xác định shared preferences là gì.
- Tạo tệp shared preferences cho ứng dụng của bạn.
- Lưu dữ liệu vào shared preferences, và đọc lại nó.
- Xoá dữ liệu trong shared preferences.

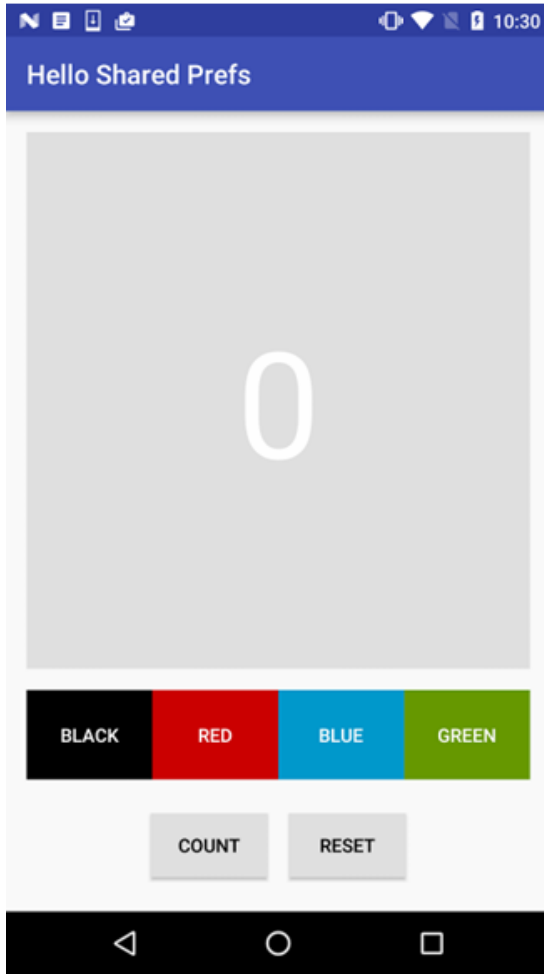
## Bạn sẽ làm gì

- Cập nhật ứng dụng để ứng dụng có thể lưu, truy xuất và đặt lại các tùy chọn chia sẻ.

## Tổng quan về ứng dụng

Ứng dụng HelloSharedPrefs là một biến thể khác của ứng dụng HelloToast bạn đã tạo trong bài 1. Nó bao gồm các nút để tăng số lượng, thay đổi màu nền, và đặt lại cả số lượng và màu sắc về mặc định. Ứng dụng cũng sử dụng chủ đề và kiểu dáng để xác định các nút.





## Task 1: Khám phá HelloSharedPrefs


Dự án ứng dụng khởi đầu hoàn chỉnh cho bài thực hành này có sẵn tại [HelloSharedPrefs-Starter](#). Trong này nhiệm vụ của bạn, bạn tải dự án vào Android Studio và khám phá một số tính năng chính của ứng dụng.

### 1.1 mở và chạy dự án HelloSharedPrefs

1. Tải ứng dụng HelloSharedPrefs-Starter và giải nén tệp.
2. Mở dự án trong android studio, và xây dựng và chạy ứng dụng. Hãy thử những điều này:
  - Bấm vào nút Đếm để tăng số trong chế độ xem văn bản chính.
  - Bấm vào bất kỳ nút màu nào để thay đổi màu nền của chế độ xem văn bản chính.

- Xoay thiết bị và lưu ý rằng cả màu nền và số lượng đều được giữ nguyên.
- Bấm vào nút Đặt lại để đặt màu và đếm ngược về mặc định.

3. Buộc thoát khỏi ứng dụng bằng một trong các phương pháp sau:

- Trong android studio, chọn Run > Stop 'app' hoặc bấm biểu tượng dừng lại  trong thanh công cụ.
- Trên thiết bị nhấn nút Gần đây (nút hình vuông ở góc dưới bên phải). Vuốt thẻ ứng dụng HelloSharedPrefs để thoát ứng dụng hoặc nhấp vào dấu X ở góc bên phải thẻ. Nếu bạn thoát ứng dụng theo cách này, hãy đợi vài giây trước khi khởi động lại để hệ thống có thể dọn dẹp.

4. Chạy lại ứng dụng. Ứng dụng khởi động lại với giao diện mặc định—số đếm là 0 và màu nền là màu xám.

## 1.2 Khám phá mã hoạt động

1. Mở MainActivity

2. Xem xét mã và lưu ý những điều sau:

- Biến đếm (mCount) được định nghĩa là một số nguyên. Phương thức onClick của countUp () tăng giá trị này lên và cập nhật TextView chính.
- Biến màu (mColor) cũng là số nguyên và được định nghĩa ban đầu là màu xám trong tệp tài nguyên colors.xml với tên là default\_background.
- Phương thức changeBackground () trong sự kiện onClick lấy màu nền của nút được nhấn, sau đó đặt màu đó làm nền cho phần văn bản chính.
- Both the mCount and mColor integers are saved to the instance state bundle in onSaveInstanceState (), and restored in onCreate (). The bundle keys for count and color are defined by private variables (COUNT\_KEY) and (COLOR\_KEY).

### Task 2: Lưu và khôi phục dữ liệu vào tệp SharedPreferences.

Trong nhiệm vụ này, bạn lưu trạng thái của ứng dụng vào một tệp Shared Preferences và đọc lại dữ liệu đó khi ứng dụng được khởi động lại. Vì dữ liệu trạng thái mà bạn lưu vào Shared Preferences (số lượng hiện tại và màu sắc)

cũng chính là dữ liệu mà bạn bảo toàn trong trạng thái instance, nên bạn không cần phải thực hiện việc này hai lần. Bạn có thể thay thế hoàn toàn trạng thái instance bằng trạng thái từ Shared Preferences.

## 2.1 Khởi tạo các tùy chọn

1. Thêm các biến thành viên vào lớp MainActivity để giữ tên của chia sẻ tệp tùy chọn và tham chiếu đến đối tượng SharedPreferences.

```
private SharedPreferences mPreferences;  
private String sharedPrefFile =  
    "com.example.android.hellosharedprefs";
```

Bạn có thể đặt tên cho tệp tùy chọn chia sẻ của mình bất cứ điều gì bạn muốn, nhưng thông thường nó có cùng tên với tên gói ứng dụng của bạn.

2. Trong phương thức onCreate (), khởi tạo shared preferences. Chèn mã này trước câu lệnh if:

```
mPreferences = getSharedPreferences (sharedPrefFile, MODE_PRIVATE) ;|
```

Phương thức getSharedPreferences () (từ Ngữ cảnh hoạt động) mở tệp tại tên tệp đã cho (sharedPrefFile) với chế độ MODE\_PRIVATE.

Lưu ý: Các phiên bản Android cũ hơn có các chế độ khác cho phép bạn tạo một ứng dụng có thể đọc được trên toàn thế giới hoặc tập tin tùy chọn chia sẻ có thể ghi trên thế giới. Các chế độ này không được dùng nữa trong API 17 và hiện tại không được khuyến khích mạnh mẽ vì lý do bảo mật. Nếu bạn cần chia sẻ dữ liệu với các ứng dụng khác, hãy cân nhắc sử dụng URI nội dung do FileProvider cung cấp.

Mã giải pháp cho MainActivity, một phần:

```

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

    private SharedPreferences mPreferences;
    private String sharedPrefFile =
        "com. example. android. hellossharedprefs";

    @Override
    protected void onCreate (Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mShowCount = (TextView) findViewById(R.id.textview);
        mColor = ContextCompat.getColor(this,
            R.color.default_background);
        mPreferences = getSharedPreferences(
            sharedPrefFile, MODE_PRIVATE);

        //
    }

```

## 2.2 Lưu tùy chọn trong onPause ()

Việc lưu tùy chọn cũng giống như lưu trạng thái phiên bản -- cả hai thao tác đều đặt dữ liệu sang một gói đối tượng dưới dạng cặp khóa/giá trị. Tuy nhiên, đối với các tùy chọn chia sẻ, bạn lưu dữ liệu đó vào gọi lại vòng đời onPause () và bạn cần một đối tượng soạn thảo được chia sẻ (SharedPreferences.Editor) để ghi vào đối tượng tùy chọn chia sẻ.

1. Thêm phương thức vòng đời onPause () vào MainActivity.

```
@Override
protected void onPause () {
    super.onPause();

    //...
}
```

2. Trong onPause (), hãy lấy trình soạn thảo cho đối tượng SharedPreferences:

Một trình chỉnh sửa SharedPreferences là cần thiết để ghi vào đối tượng SharedPreferences. Thêm dòng này vào `onPause()` sau lệnh gọi `super.onPause()`.

3. Sử dụng phương thức `putInt()` để đưa cả hai giá trị số nguyên `mCount` và `mColor` vào SharedPreferences với các khóa phù hợp.

```
preferencesEditor.putInt (COUNT_KEY, mCount) ;
preferencesEditor.putInt (COLOR_KEY, mColor) ;
```

Lớp `SharedPreferences.Editor` bao gồm nhiều phương thức "put" cho các kiểu dữ liệu khác nhau, bao gồm `putInt()` và `putString()`.

4. Gọi `apply()` để lưu các tùy chọn preferences.

```
preferencesEditor.apply();
```

Phương thức `apply()` lưu các tùy chọn (preferences) một cách bất đồng bộ, ngoài luồng giao diện người dùng (UI thread). Trình chỉnh sửa SharedPreferences cũng có phương thức `commit()` để lưu các tùy

chọn một cách đồng bộ. Tuy nhiên, phương thức `commit()` không được khuyến khích sử dụng vì nó có thể chặn các thao tác khác.

5. Xóa toàn bộ phương thức `onSaveInstanceState()`. Vì trạng thái instance của activity chứa cùng dữ liệu như `SharedPreferences`, bạn có thể thay thế hoàn toàn trạng thái instance.

Mã giải pháp cho phương thức `onPause()` trong `MainActivity`:

```
@Override
protected void onPause () {
    super.onPause();

    SharedPreferences.Editor preferencesEditor = mPreferences.edit();
    preferencesEditor.putInt(COUNT_KEY, mCount);
    preferencesEditor.putInt(COLOR_KEY, mColor);
    preferencesEditor.apply();
}
```

## 2.3 Khôi phục các tùy chọn trong `onCreate()`

Tương tự như trạng thái phiên bản, ứng dụng của bạn đọc bất kỳ tùy chọn được chia sẻ đã lưu nào trong phương thức `onCreate()`. Một lần nữa, vì các tùy chọn được chia sẻ chứa dữ liệu giống như trạng thái phiên bản, chúng ta cũng có thể thay thế trạng thái bằng các tùy chọn ở đây. Mỗi khi `onCreate()` được gọi -- khi ứng dụng khởi động, khi thay đổi cấu hình -- các tùy chọn được chia sẻ được sử dụng để khôi phục trạng thái của chế độ xem.

1. Xác định vị trí phần của phương thức `onCreate()` kiểm tra xem đối số `savedInstanceState` có phải là null hay không và khôi phục trạng thái phiên bản:

```

if (savedInstanceState != null) {
    mCount = savedInstanceState.getInt(COUNT_KEY);
    if (mCount != 0) {
        mShowCountTextView.setText(String.format("&s", mCo
    }

    mColor = savedInstanceState.getInt(COLOR_KEY);
    mShowCountTextView.setBackgroundColor(mColor);
}

```

2. Xóa toàn bộ khối.
3. Trong phương thức onCreate(), tại cùng vị trí có mã trạng thái phiên bản, hãy lấy đếm từ các tùy chọn bằng phím COUNT\_KEY và gán nó cho biến mCount.

```

mCount = mPreferences.getInt(COUNT_KEY, 0);

```

Khi bạn đọc dữ liệu từ các tùy chọn, bạn không cần phải có tùy chọn chia sẻ biên tập viên. Sử dụng bất kỳ phương thức "get" nào trên đối tượng tùy chọn được chia sẻ (chẳng hạn như getInt() hoặc getString()) để lấy dữ liệu ưu tiên.

Lưu ý rằng phương thức getInt() nhận hai đối số: một cho khóa và một cho giá trị mặc định nếu không tìm thấy khóa. Trong trường hợp này, giá trị mặc định là 0, là giá trị giống như giá trị ban đầu của mCount.

4. Cập nhật giá trị của TextView chính bằng số lượng mới.

```

mShowCountTextView.setText(String.format("%s", mCount));|

```

5. Lấy màu từ tùy chọn bằng phím COLOR\_KEY và gán nó cho biến mColor.

```

mColor = mPreferences.getInt(COLOR_KEY, mColor);

```

Như trước, đối số thứ hai của `getInt()` là giá trị mặc định được sử dụng trong trường hợp khóa không tồn tại trong các tùy chọn được chia sẻ. Trong trường hợp này bạn chỉ có thể sử dụng lại giá trị của `mColor`, vừa được khởi tạo thành nền mặc định trong phương thức.

6. Cập nhật màu nền của chế độ xem văn bản chính.

```
mShowCountTextView.setBackgroundColor(mColor);
```

7. Chạy ứng dụng. Bấm vào nút Đếm và thay đổi màu nền để cập nhật trạng thái cá thể và các ưu tiên.
8. Xoay thiết bị hoặc trình mô phỏng để xác minh rằng số lượng và màu sắc được lưu khi thay đổi cấu hình.
9. Buộc thoát khỏi ứng dụng bằng một trong các phương pháp sau:
  - Trong Android Studio, chọn Run > Stop 'app.'
  - Trên thiết bị nhấn nút Gần đây (nút hình vuông ở góc dưới bên phải). Vuốt thẻ ứng dụng HelloSharedPreferences để thoát ứng dụng hoặc nhấn vào dấu X ở góc bên phải thẻ.
10. Chạy lại ứng dụng. Ứng dụng khởi động lại và tải các tùy chọn, duy trì trạng thái.  
Mã giải pháp cho phương thức `MainActivity onCreate()`:



```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Initialize views, color, preferences
    mShowCountTextView = (TextView) findViewById(R.id.count_text);
    mColor = ContextCompat.getColor(this, R.color.default_background);
    mPreferences = getSharedPreferences(mSharedPreffile, MODE_PRIVATE);

    // Restore preferences
    mCount = mPreferences.getInt(COUNT_KEY, 0);
    mShowCountTextView.setText(String.format("%s", mCount));
    mColor = mPreferences.getInt(COLOR_KEY, mColor);
    mShowCountTextView.setBackgroundColor(mColor);
}

```

## 2.4 Khôi phục các tùy chọn trong onCreate()

Nút đặt lại trong ứng dụng khởi động sẽ đặt lại cả số lượng và màu sắc cho hoạt động về giá trị mặc định. Vì tùy chọn giữ trạng thái của hoạt động nên điều quan trọng là phải xóa tùy chọn cùng một lúc.

1. Trong phương thức reset() onClick, sau khi đặt lại màu và số lượng, hãy tải trình chỉnh sửa cho đối tượng SharedPreferences:

```
SharedPreferences.Editor preferencesEditor = mPreferences.edit();
```

2. Xóa tất cả các shared preferences:

```
preferencesEditor.clear();
```

### 3. Áp dụng các thay đổi:

```
preferencesEditor.apply();
```

Mã giải pháp cho phương thức reset():

```
public void reset(View view) {  
    // Reset count  
    mCount = 0;  
    mShowCountTextView.setText(String.format("%s", mCount));  
  
    // Reset color  
    mColor = ContextCompat.getColor(this, R.color.default_background);  
    mShowCountTextView.setBackgroundColor(mColor);  
  
    // Clear preferences  
    SharedPreferences.Editor preferencesEditor = mPreferences.edit();  
    preferencesEditor.clear();  
    preferencesEditor.apply();  
}
```

## Solution code

Dự án Android Studio: HelloSharedPrefs

## Thử thách viết mã

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

**Thử thách:** Sửa đổi ứng dụng HelloSharedPrefs để thay vì tự động lưu trạng thái vào tệp tùy chọn, hãy thêm hoạt động thứ hai để thay đổi, đặt lại và lưu các tùy chọn đó. Thêm nút vào ứng dụng có tên Cài đặt để khởi chạy hoạt động đó. Bao gồm các nút chuyển đổi và nút xoay để sửa đổi tùy chọn cũng như các nút Lưu và Đặt lại để lưu và xóa tùy chọn.

## Bản tóm tắt

- Lớp SharedPreferences cho phép ứng dụng lưu trữ một lượng nhỏ dữ liệu nguyên thủy dưới dạng cặp khóa-giá trị.
- Shared Preferences vẫn tồn tại trong các phiên người dùng khác nhau của cùng một ứng dụng.
- Để ghi vào các tùy chọn chia sẻ, hãy lấy đối tượng SharedPreferences.Editor.
- Sử dụng các phương thức "đặt" khác nhau trong đối tượng SharedPreferences.Editor, chẳng hạn như putInt() hoặc putString(), để đưa dữ liệu vào các tùy chọn chia sẻ bằng một khóa và một giá trị.
- Sử dụng các phương thức "get" khác nhau trong đối tượng SharedPreferences, chẳng hạn như getInt() hoặc getString() , để lấy dữ liệu ra khỏi các tùy chọn được chia sẻ bằng một khóa.
- Sử dụng phương thức clear() trong đối tượng SharedPreferences.Editor để xóa tất cả dữ liệu được lưu trữ trong Preferences.
- Sử dụng phương thức apply() trong đối tượng SharedPreferences.Editor để lưu các thay đổi vào tệp tùy chọn.

## Khái niệm liên quan

Tài liệu khái niệm liên quan có trong 9.0: Lưu trữ dữ liệu và 9.1: Tùy chọn chia sẻ .

## Learn more

Tài liệu dành cho nhà phát triển Android:

- Data and file storage overview
- Save key-value data
- SharedPreferences
- SharedPreferences.Editor

Trần ngấn xếp:

- How to use SharedPreferences in Android to store, fetch and edit values
- onSaveInstanceState vs. SharedPreferences

## **Bài tập về nhà**

### **Xây dựng và chạy một ứng dụng**

Mở ứng dụng ScoreKeeper mà bạn đã tạo trong bài học Nguyên tắc cơ bản về Android 5.1: Bản vẽ, kiểu và chủ đề.

1. Thay thế trạng thái phiên bản đã lưu bằng các tùy chọn chung cho từng điểm số.
2. Để kiểm tra ứng dụng, hãy xoay thiết bị để đảm bảo rằng các thay đổi về cấu hình sẽ đọc các tùy chọn đã lưu và cập nhật giao diện người dùng.
3. Dừng ứng dụng và khởi động lại để đảm bảo rằng các tùy chọn được lưu.
4. Thêm nút Đặt lại để đặt lại giá trị điểm về 0 và xóa SharedPreferences.

## **Answer these questions**

### **Câu hỏi 1**

Bạn lưu trạng thái ứng dụng vào tùy chọn chia sẻ theo phương pháp vòng đời nào?

### **Câu hỏi 2**

Bạn khôi phục trạng thái ứng dụng bằng phương pháp vòng đời nào?

### **Câu hỏi 3**

Bạn có thể nghĩ ra trường hợp nào hợp lý khi có cả sở thích chung và trạng thái phiên bản không?

## **Gửi ứng dụng của bạn để chấm điểm**

Hướng dẫn cho học sinh chấm điểm

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

- Ứng dụng giữ lại điểm khi xoay thiết bị.
- Ứng dụng sẽ giữ lại điểm hiện tại sau khi dừng và khởi động lại ứng dụng.
- Ứng dụng lưu điểm hiện tại vào tùy chọn được chia sẻ trong phương thức onPause().
- Ứng dụng khôi phục các tùy chọn được chia sẻ trong phương thức onCreate().
- Ứng dụng hiển thị nút Đặt lại để đặt lại điểm về 0.

Đảm bảo rằng việc triển khai phương thức xử lý khi nhấp chuột cho nút Đặt lại sẽ thực hiện những điều sau đây:

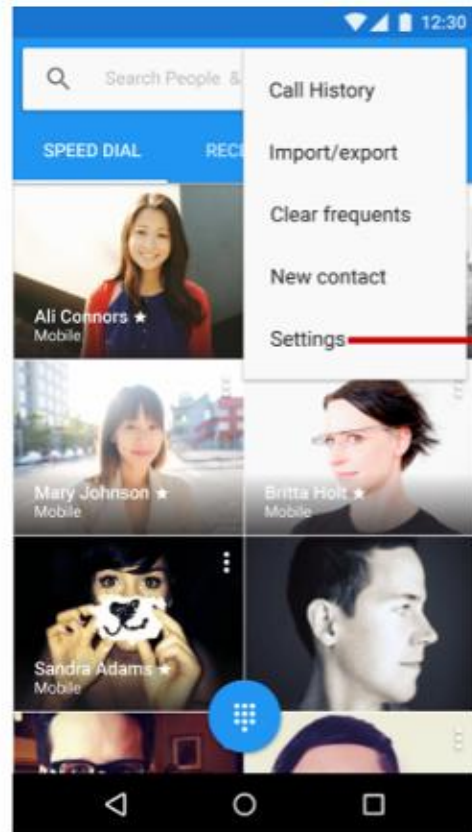
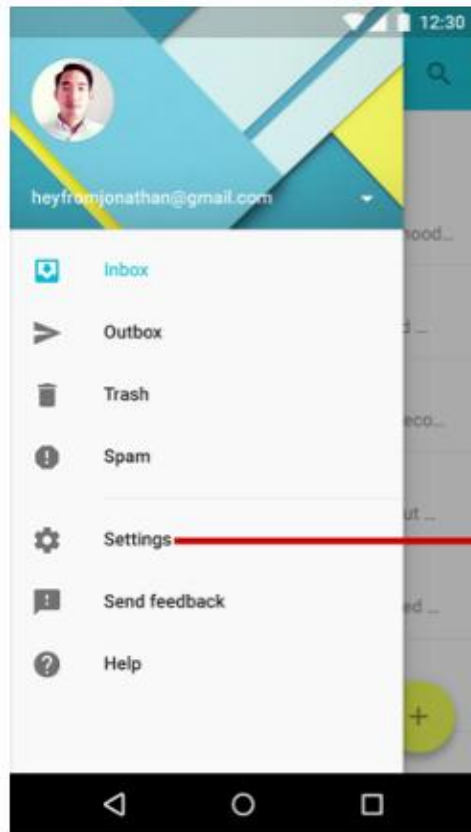
- Đặt lại cả hai biến số điểm về 0.
- Cập nhật cả chế độ xem văn bản.
- Xóa tùy chọn chia sẻ.

## **1.2) Cài đặt ứng dụng**

### **Giới thiệu**

Ứng dụng thường bao gồm các cài đặt cho phép người dùng sửa đổi các tính năng và hành vi của ứng dụng. Ví dụ: một số ứng dụng cho phép người dùng đặt vị trí nhà riêng, đơn vị đo lường mặc định và các cài đặt khác áp dụng cho toàn bộ ứng dụng. Người dùng không truy cập cài đặt thường xuyên vì khi người dùng thay đổi cài đặt, chẳng hạn như vị trí nhà riêng, họ hiếm khi cần quay lại và thay đổi lại cài đặt đó.

Người dùng muốn điều hướng đến cài đặt ứng dụng bằng cách nhấn vào Cài đặt trong điều hướng bên cạnh, chẳng hạn như điều hướng ngăn kéo như minh họa ở phía bên trái của hình bên dưới hoặc trong menu tùy chọn trên thanh ứng dụng, hiển thị ở phía bên phải của hình bên dưới.



Trong hình trên:

1. Cài đặt trong điều hướng bên (ngăn điều hướng)
2. Cài đặt trong menu tùy chọn của thanh ứng dụng

Trong thực tế này, bạn thêm hoạt động cài đặt vào một ứng dụng. Người dùng sẽ có thể điều hướng đến ứng dụng cài đặt bằng cách nhấn vào Cài đặt, cài đặt này sẽ nằm trong menu tùy chọn trên thanh ứng dụng.

## Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo dự án Android Studio từ một mẫu và tạo bố cục chính.
- Chạy ứng dụng trên trình mô phỏng hoặc thiết bị được kết nối.
- Tạo và chỉnh sửa các thành phần giao diện người dùng bằng trình chỉnh sửa bố cục và mã XML.
- Trích xuất tài nguyên chuỗi và chỉnh sửa giá trị chuỗi.

- Truy cập các phần tử giao diện người dùng từ mã của bạn bằng cách sử dụng `findViewById()` .
- Xử lý thao tác bấm nút.
- Hiển thị thông báo Bánh mì nướng.
- Thêm Hoạt động vào ứng dụng.
- Tạo menu tùy chọn trong thanh ứng dụng.
- Thêm và chỉnh sửa các mục menu trong menu tùy chọn.
- Sử dụng phong cách và chủ đề trong dự án.
- Sử dụng `SharedPreferences`

## **Bạn sẽ học được gì**

Bạn sẽ học cách:

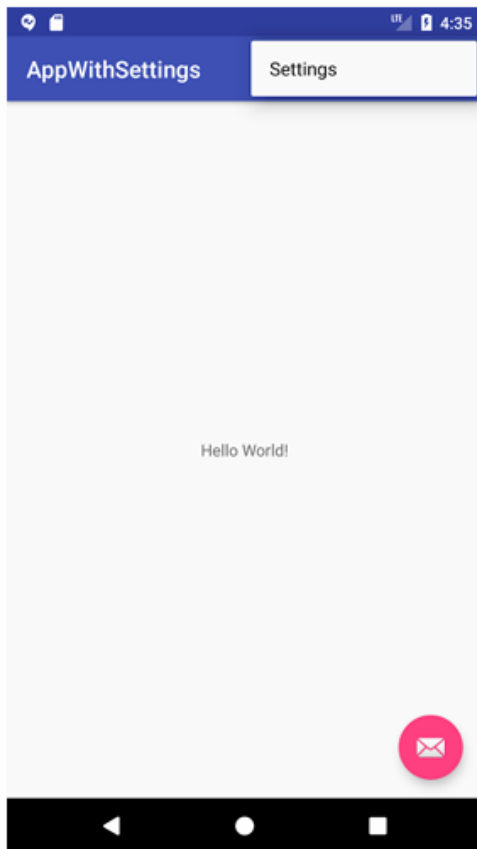
- Thêm một đoạn để quản lý cài đặt.
- Tạo tệp tài nguyên XML của cài đặt với các thuộc tính của chúng.
- Tạo điều hướng đến cài đặt Hoạt động.
- Đặt giá trị mặc định của cài đặt.
- Đọc các giá trị cài đặt do người dùng thay đổi.
- Tùy chỉnh mẫu Hoạt động cài đặt.

## **What you'll do**

- Tạo một ứng dụng bao gồm Cài đặt trong menu tùy chọn.
- Thêm nút chuyển đổi tùy chọn Cài đặt.
- Thêm mã để đặt giá trị mặc định cho cài đặt và truy cập giá trị cài đặt sau khi đã có thay đổi.
- Sử dụng và tùy chỉnh mẫu Hoạt động cài đặt Android Studio.

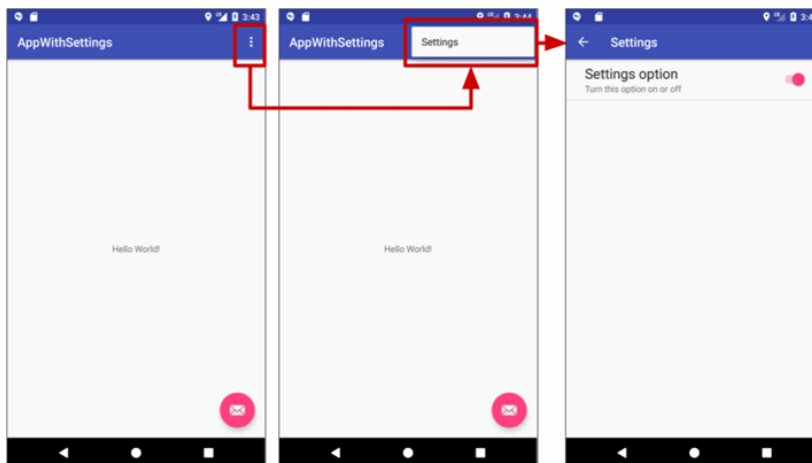
## **Tổng quan về ứng dụng**

Android Studio cung cấp lối tắt để thiết lập menu tùy chọn với Cài đặt. Nếu bạn bắt đầu dự án Android Studio cho điện thoại hoặc máy tính bảng bằng mẫu Hoạt động cơ bản thì ứng dụng mới sẽ bao gồm Cài đặt như hiển thị bên dưới:



Mẫu cũng bao gồm một nút tác vụ nổi ở góc dưới bên phải màn hình với biểu tượng phong bì. Bạn có thể bỏ qua nút này vì bạn sẽ không sử dụng nút này.

Bạn sẽ bắt đầu bằng cách tạo một ứng dụng có tên AppWithSettings bằng mẫu Hoạt động cơ bản và bạn sẽ thêm cài đặt Hoạt động cung cấp một cài đặt công tắc bật tắt mà người dùng có thể bật hoặc tắt:



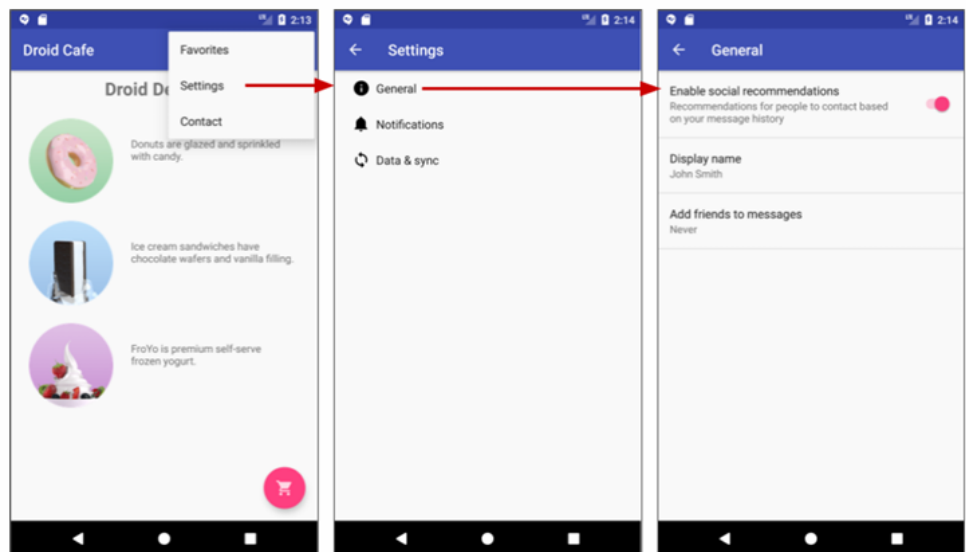


Bạn sẽ thêm mã để đọc cài đặt và thực hiện hành động dựa trên giá trị của nó. Để đơn giản, hành động sẽ là hiển thị thông báo Toast với giá trị cài đặt.

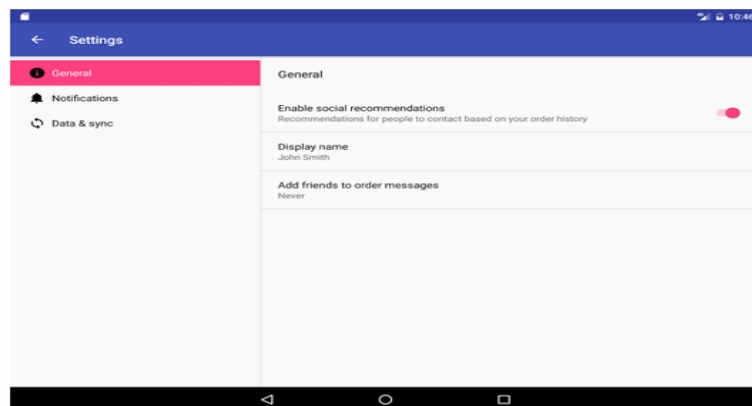
Trong nhiệm vụ thứ hai, bạn sẽ thêm mẫu Hoạt động cài đặt tiêu chuẩn do Android Studio cung cấp vào ứng dụng DroidCafeOptionsUp mà bạn đã tạo trong bài học trước.

Mẫu Hoạt động cài đặt được điền sẵn các cài đặt mà bạn có thể tùy chỉnh cho một ứng dụng và cung cấp bố cục khác cho điện thoại và máy tính bảng:

- Điện thoại : Màn hình Cài đặt chính có liên kết tiêu đề cho từng nhóm cài đặt, chẳng hạn như Chung cho cài đặt chung, như hiển thị bên dưới.



- Máy tính bảng : Bố cục màn hình chính/chi tiết có liên kết tiêu đề cho từng nhóm ở phía bên trái (chính) và nhóm cài đặt ở phía bên phải (chi tiết), như minh họa trong hình bên dưới.



Để tùy chỉnh mẫu, bạn sẽ thay đổi tiêu đề, đặt tiêu đề, mô tả cài đặt và giá trị cho cài đặt.

Ứng dụng DroidCafeOptionsUp đã được tạo trong bài học trước từ mẫu Hoạt động cơ bản, cung cấp menu tùy chọn trong thanh ứng dụng để đặt tùy chọn Cài đặt. Bạn sẽ tùy chỉnh mẫu Hoạt động cài đặt được cung cấp bằng cách thay đổi tiêu đề, mô tả, giá trị và giá trị mặc định của một cài đặt. Bạn sẽ thêm mã để đọc giá trị của cài đặt sau khi người dùng thay đổi và hiển thị giá trị đó.

## Task 1: Thêm cài đặt chuyển đổi vào ứng dụng

Trong nhiệm vụ này, bạn thực hiện như sau:

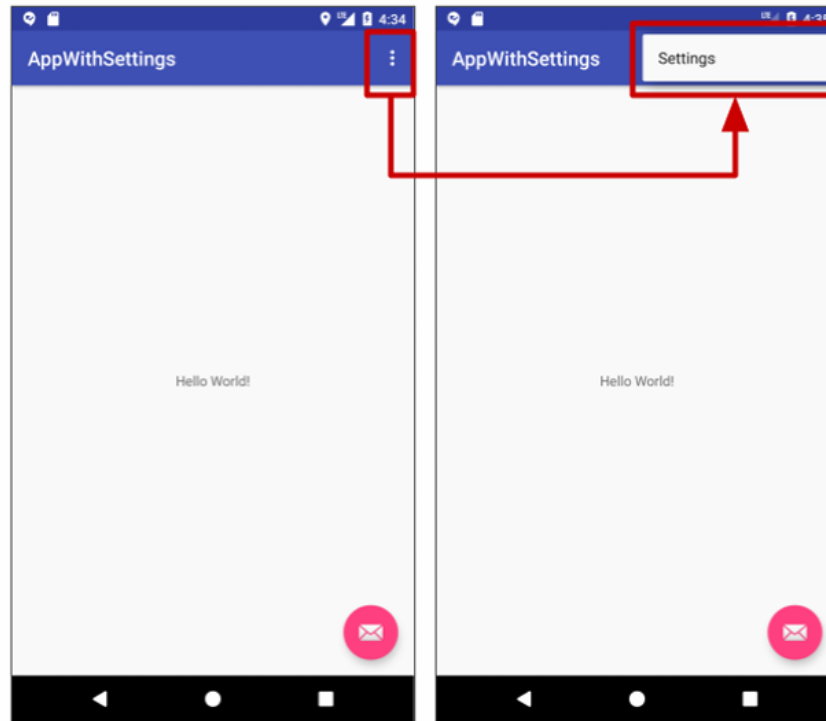
- Tạo một dự án mới dựa trên mẫu Hoạt động cơ bản, cung cấp menu tùy chọn.
- Thêm công tắc bật tắt ( SwitchPreference ) với các thuộc tính trong tệp XML tùy chọn.
- Thêm hoạt động cho cài đặt và một đoạn cho cài đặt cụ thể. Để duy trì khả năng tương thích với AppCompatActivity , bạn sử dụng PreferenceFragmentCompat thay vì PreferenceFragment . Bạn cũng thêm thư viện android.support.v7.preference.
- Kết nối mục Cài đặt trong menu tùy chọn với hoạt động cài đặt.

### 1.1 Tạo dự án và thêm thư mục xml và tệp tài nguyên

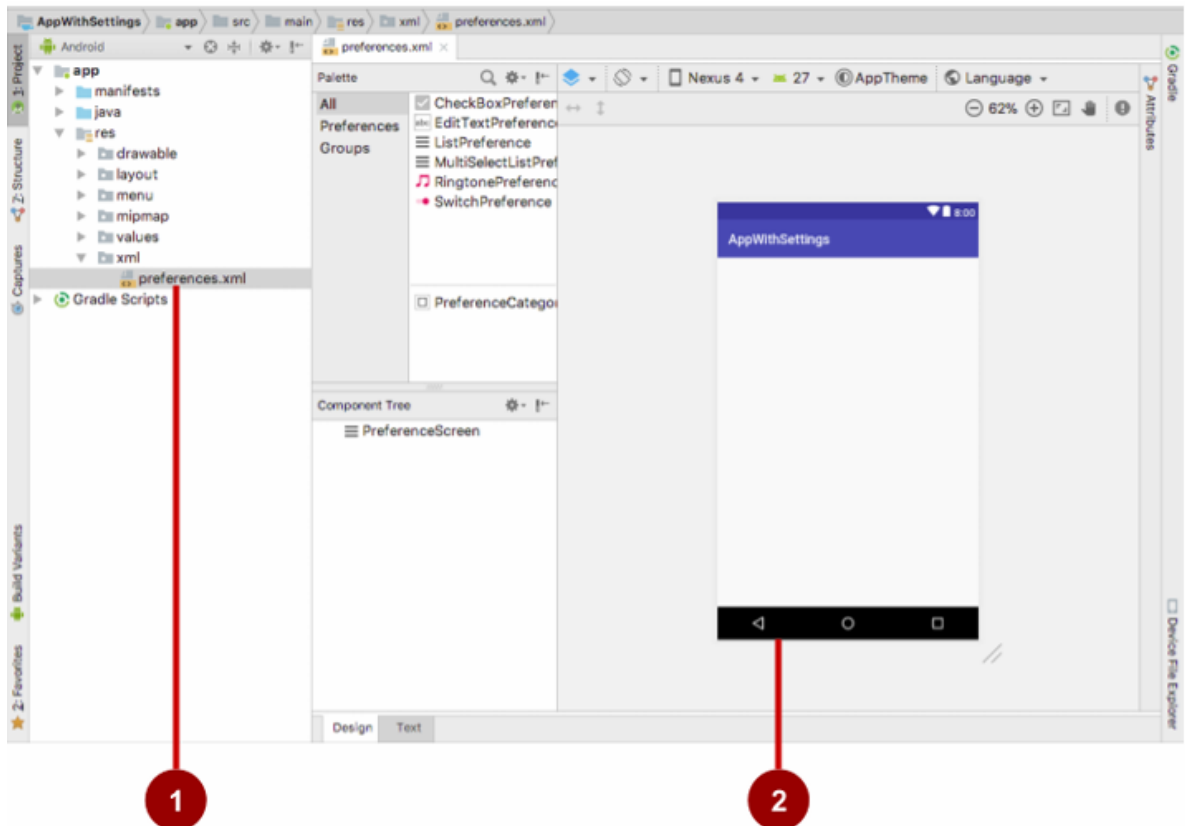
1. Trong Android Studio, tạo một dự án mới với các thông số sau:

Thuộc tính	Giá trị
Tên ứng dụng	android.example.com (hoặc tên miền của riêng bạn)
Tên công ty	android.example.com (hoặc tên miền của riêng bạn)
Vị trí dự án	Đường dẫn đến thư mục dự án của bạn
SDK tối thiểu cho điện thoại và máy tính bảng	API15:Android4.0.3 IceCreamSandwich
Bản mẫu	Hoạt động cơ bản
Tên hoạt động	Hoạt động chính
Tên bố cục	hoạt động chính
Tiêu đề	Hoạt động chính

2. Chạy ứng dụng và nhấn vào biểu tượng tràn trên thanh ứng dụng để xem menu tùy chọn, như minh họa trong hình bên dưới. Mục duy nhất trong menu tùy chọn là Cài đặt.



3. Bạn cần tạo một thư mục tài nguyên mới để chứa tệp XML chứa các cài đặt. Chọn thư mục res trong khung Project > Android và chọn File > New > Android Resource Directory . Hộp thoại Thư mục tài nguyên mới xuất hiện.
4. Trong menu thả xuống Loại tài nguyên, chọn xml . Tên thư mục tự động thay đổi thành xml . Bấm vào đồng ý .
5. Thư mục xml xuất hiện trong khung Project > Android bên trong thư mục res. Chọn xml và chọn Tệp > Mới > Tệp tài nguyên XML (hoặc nhấp chuột phải vào xml và chọn Mới > Tệp tài nguyên XML).
6. Nhập tên của tệp XML, tùy chọn, vào trường Tên tệp và nhấp vào OK. Tệp preferences.xml xuất hiện bên trong thư mục xml và trình chỉnh sửa bố cục xuất hiện, như minh họa trong hình bên dưới.

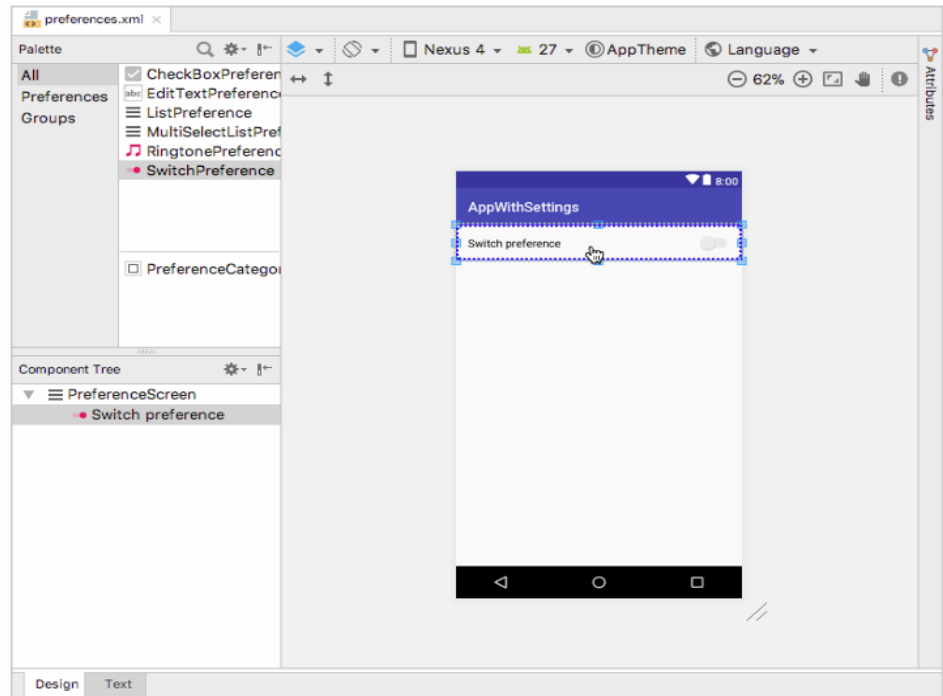


Trong hình trên:

1. Tập preferences.xml bên trong thư mục xml.
2. Trình chỉnh sửa bố cục hiển thị nội dung preferences.xml.

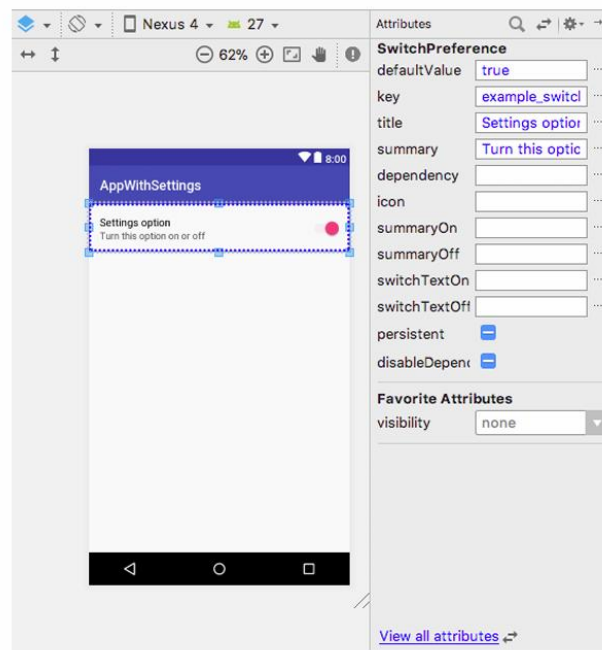
## 1.2 Thêm tùy chọn và thuộc tính XML cho cài đặt

1. Kéo SwitchPreference từ ngăn Bảng màu ở bên trái lên trên cùng của bố cục, như minh họa trong hình bên dưới.



2. Thay đổi các giá trị trong khung Thuộc tính ở bên phải của trình chỉnh sửa bố cục như sau và như minh họa trong hình bên dưới:

- giá trị mặc định: true
- phím: example\_switch
- tiêu đề: Tùy chọn cài đặt
- tóm tắt: Bật hoặc tắt tùy chọn này



3. Nhấp vào tab Văn bản ở cuối trình chỉnh sửa bố cục để xem mã XML:

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

    <SwitchPreference
        android:defaultValue="true"
        android:key="@example_switch"
        android:summary="Turn this option on or off"
        android:title="Settings option" />
</PreferenceScreen>
```

4. Trích xuất tài nguyên chuỗi cho các giá trị thuộc tính android:title và android:tóm thành @string/switch\_title và @string/switch\_summary

Các thuộc tính XML cho một tùy chọn là:

- android:defaultValue : Giá trị mặc định của cài đặt khi ứng dụng khởi động lần đầu tiên.
- android:title : Tiêu đề của cài đặt. Đối với SwitchPreference , tiêu đề xuất hiện ở bên trái của nút chuyển đổi.
- android:key : Phím dùng để lưu trữ giá trị cài đặt. Mỗi cài đặt có một cặp khóa-giá trị tương ứng mà hệ thống sử dụng để lưu cài đặt trong tệp SharedPreferences mặc định cho cài đặt ứng dụng của bạn.
- android:summary : Tóm tắt văn bản xuất hiện bên dưới cài đặt.

### 1.3 Sử dụng SwitchPreferenceCompat

Để sử dụng phiên bản PreferenceFragmentCompat của PreferenceFragment , bạn cũng phải sử dụng phiên bản android.support.v7 của SwitchPreference ( SwitchPreferenceCompat ).

1. Trong ngăn Dự án > Android, hãy mở tệp build.gradle (Mô-đun: ứng dụng) trong thư mục Tập lệnh Gradle và thêm phần sau vào phần phụ thuộc:

```
implementation 'com.android.support:preference-v7:26.1.0'
```

Câu lệnh hiển thị ở trên bổ sung thư viện android.support.v7.preference để sử dụng phiên bản PreferenceFragmentCompat của PreferenceFragmet

2. Trong tệp preferences.xml trong thư mục xml, hãy thay đổi SwitchPreference trong mã thành android.support.v7.preference.SwitchPreferenceCompat

```
<android.support.v7.preference.SwitchPreferenceCompat
    android:defaultValue="true"
    android:key="example_switch"
    android:summary="@string/switch_summary"
    android:title="@string/switch_title" />
```

Dòng SwitchPreferenceCompat ở trên có thể hiển thị biểu tượng bóng đèn màu vàng kèm theo cảnh báo, nhưng hiện tại bạn có thể bỏ qua nó.

3. Mở tệp styles.xml trong thư mục giá trị và thêm mục preferencesTheme sau vào khai báo AppTheme:

```
<item name="preferencesTheme">@style/PreferenceThemeOverlay</item>
```

Để sử dụng PreferenceFragmentCompat, bạn cũng phải khai báo preferencesTheme với kiểu PreferenceThemeOverlay cho chủ đề ứng dụng.

#### 1.4 Thêm Hoạt động cho cài đặt

Để tạo Hoạt động cài đặt cung cấp giao diện người dùng cho cài đặt, hãy thêm Hoạt động trống vào ứng dụng. Thực hiện theo các bước sau:

1. Chọn ứng dụng ở đầu ngăn Dự án > Android và chọn Mới > Hoạt động > Hoạt động trống .
2. Đặt tên cho Hoạt động Cài đặt Hoạt động. Bỏ chọn tùy chọn Tạo tệp bố cục (bạn không cần tùy chọn này) và bỏ chọn tùy chọn Hoạt động của trình khởi chạy.
3. Chọn tùy chọn Tương thích ngược (AppCompat). Tên gói phải được đặt thành com.example.android. tên dự án .
4. Nhấp vào Hoàn tất.

### 1.5 Thêm Đoạn cho cài đặt cụ thể

Phân đoạn giống như một phần mô-đun của Hoạt động —nó có vòng đời riêng và nhận các sự kiện đầu vào riêng và bạn có thể thêm hoặc xóa Phân đoạn trong khi Hoạt động đang chạy. Bạn sử dụng lớp con Fragment chuyên dụng để hiển thị danh sách cài đặt. Cách tốt nhất là sử dụng Hoạt động thông thường lưu trữ PreferenceFragment hiển thị cài đặt ứng dụng. PreferenceFragmet cung cấp kiến trúc linh hoạt hơn cho ứng dụng của bạn so với việc sử dụng Hoạt động cho tùy chọn.

Bạn sẽ sử dụng PreferenceFragmentCompat thay vì PreferenceFragmet để duy trì khả năng tương thích với AppCompatActivity .

Trong bước này, bạn sẽ thêm một Đoạn trống cho một nhóm cài đặt tương tự (không có bố cục, phương thức gốc hoặc lệnh gọi lại giao diện) vào ứng dụng và mở rộng PreferenceFragmetCompat .

Thực hiện theo các bước sau:

1. Chọn lại ứng dụng và chọn Mới > Đoạn > Đoạn (Trống) .
2. Đặt tên cho đoạn SettingFragment . Bỏ chọn Tạo bố cục XML? tùy chọn (bạn không cần).
3. Bỏ chọn các tùy chọn để bao gồm các phương thức xuất xưởng của mảnh và lệnh gọi lại giao diện.
4. Bộ nguồn mục tiêu phải được đặt thành main .
5. Nhấp vào Hoàn tất. Kết quả là định nghĩa lớp sau trong SettingFragment :



```
}  
}
```

6. Chỉnh sửa định nghĩa lớp của SettingsFragment để mở rộng PreferenceFragmetCompat :

```
public class SettingsFragment extends Fragment {  
  
    public SettingsFragment() {  
        // Required empty public constructor  
    }  
  
    @Override  
    public View onCreateView(LayoutInflater inflater,  
        ViewGroup container, Bundle savedInstanceState) {  
        TextView textView = new TextView(getActivity());  
        textView.setText(R.string.hello_blank_fragment);  
        return textView;  
    }  
}
```

Khi bạn thay đổi định nghĩa lớp sao cho phù hợp với định nghĩa hiển thị ở trên, bóng đèn màu đỏ sẽ xuất hiện ở lề trái. Nhấp vào bóng đèn màu đỏ và chọn Phương pháp triển khai, sau đó chọn onCreatePferencs. Android Studio tạo sơ khai onCreatePreferences() sau:

```
@Override  
public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {  
}
```

Để mở rộng Fragment , Android Studio bổ sung câu lệnh nhập sau:

```
import android.support.v7.preference.PreferenceFragmentCompat;
```

#### 7. Xóa toàn bộ phương thức onCreateView() trong đoạn.

Lý do về cơ bản bạn thay thế onCreateView() bằng onCreatePreferences() là vì bạn sẽ thêm phần Cài đặt này vào phần Cài đặt hiện có để hiển thị các tùy chọn thay vì hiển thị một màn hình Phân đoạn riêng biệt. Việc thêm nó vào Hoạt động hiện có giúp bạn dễ dàng thêm hoặc xóa Phân đoạn trong khi Hoạt động đang chạy. Đoạn tùy chọn được bắt nguồn từ PreferenceScreen bằng cách sử dụng rootKey .

Bạn cũng có thể xóa hàm tạo trống khỏi SettingFragment một cách an toàn, vì Fragment không tự hiển thị:

```
public SettingsFragment() {  
    // Required empty public constructor  
}
```

8. Bạn cần liên kết với Fragment này tài nguyên cài đặt preferences.xml mà bạn đã tạo ở bước trước. Thêm vào lệnh gọi onCreatePreferences() mới được tạo, gọi tới setPreferencesFromResource() chuyển id của tệp XML ( R.xml.preferences) và rootKey để xác định gốc tùy chọn trong PreferenceScreen:

```
setPreferencesFromResource(R.xml.preferences, rootKey);
```

Phương thức onCreatePreferences() bây giờ sẽ trông như thế này:

```
@Override  
public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {  
    setPreferencesFromResource(R.xml.preferences, rootKey);  
}
```

## 1.6 Hiển thị đoạn trong Cài đặt Hoạt động

Để hiển thị Đoạn trong Cài đặt Hoạt động, hãy làm theo các bước sau:

1. Mở Cài đặt Hoạt động.
2. Thêm đoạn mã sau vào cuối phương thức onCreate() để Fragment được hiển thị làm nội dung chính:

```
getSupportFragmentManager().beginTransaction()  
    .replace(android.R.id.content, new SettingsFragment())  
    .commit();
```

Đoạn mã trên sử dụng mẫu điển hình để thêm một đoạn vào một hoạt động sao cho đoạn đó xuất hiện dưới dạng nội dung chính của hoạt động:

- Sử dụng getSupportFragmentManager() nếu lớp mở rộng Hoạt động và Đoạn mở rộng PreferenceFragment.
- Sử dụng getSupportFragmentManager() nếu lớp mở rộng AppCompatActivity và tới đoạn mở rộng PreferenceFramentCompat.

Toàn bộ phương thức onCreate() trong Cài đặtActivity bây giờ sẽ trông giống như sau:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    getSupportFragmentManager().beginTransaction()  
        .replace(android.R.id.content, new SettingsFragment())  
        .commit();  
}
```

## 1.7 Kết nối mục menu Cài đặt với Cài đặt Hoạt động

Sử dụng Ý định để khởi chạy Cài đặtHoạt động từ MainActivity khi người dùng chọn Cài đặt từ menu tùy chọn.

1. Mở MainActivity và tìm khối if trong phương thức onOptionsItemSelected() để xử lý thao tác nhấn vào Cài đặt trong menu tùy chọn:

```
if (id == R.id.action_settings) {  
    return true;  
}
```

2. Thêm Ý định vào khối if để khởi chạy Cài đặtHoạt động:

```
if (id == R.id.action_settings) {  
    Intent intent = new Intent(this, SettingsActivity.class);  
    startActivity(intent);  
    return true;  
}
```

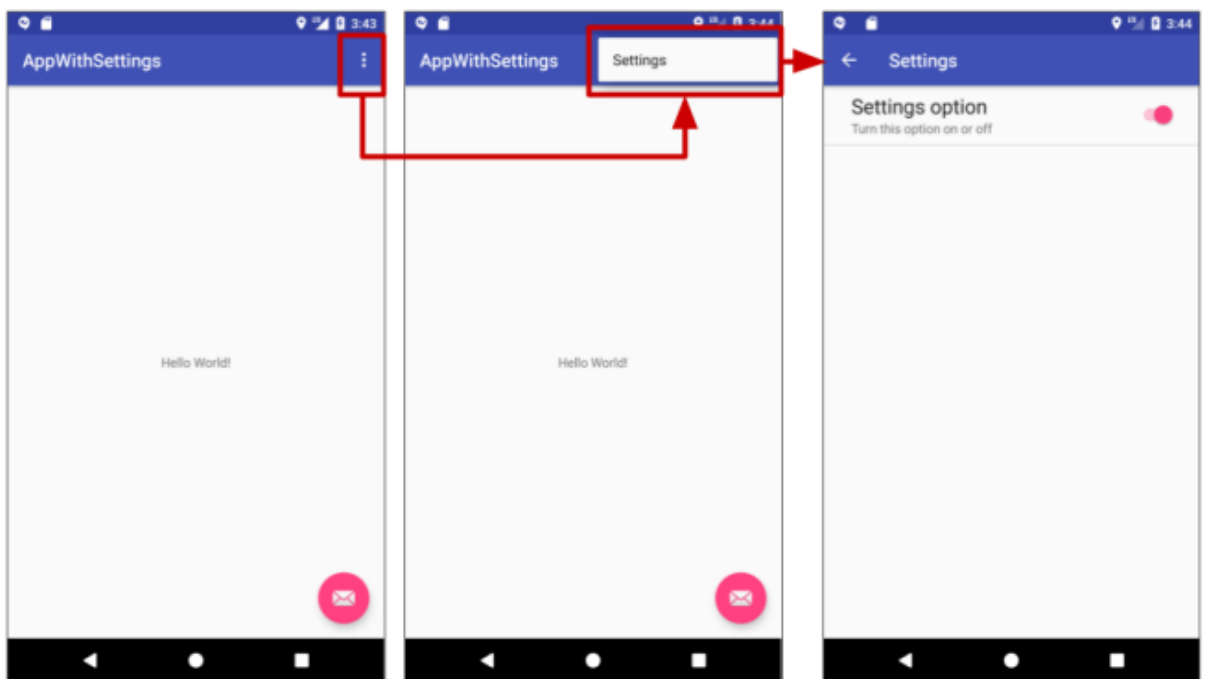
3. Để thêm nút điều hướng Lên trên thanh ứng dụng vào Cài đặtHoạt động, bạn cần chỉnh sửa nút điều hướng đó khai báo trong tệp AndroidManifest.xml để xác định cha mẹ Cài đặt hoạt động là hoạt động chính. Mở AndroidManifest.xml và tìm phần khai báo Cài đặt hoạt động:

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

Thay đổi khai báo như sau:

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

4. Chạy ứng dụng. Nhấn vào biểu tượng tràn cho menu tùy chọn, như minh họa ở phía bên trái của hình bên dưới. Nhấn vào Cài đặt để xem hoạt động cài đặt, như minh họa ở giữa hình dưới. Nhấn vào nút Lên trên thanh ứng dụng của hoạt động cài đặt, hiển thị ở bên phải của hình bên dưới, để quay lại hoạt động chính



## 1.8 Lưu các giá trị mặc định trong tùy chọn chia sẻ

Mặc dù giá trị mặc định cho cài đặt công tắc bật tắt đã được đặt trong thuộc tính `android:defaultValue` (ở Bước 1.2 của tác vụ này), ứng dụng phải lưu giá trị mặc định trong tệp `SharedPreferences` cho từng cài đặt khi người dùng mở ứng

dụng lần đầu tiên. Thực hiện theo các bước sau để thiết lập giá trị mặc định cho công tắc bật tắt:

1. Mở MainActivity

2. Thêm phần sau vào cuối phương thức onCreate() sau FloatingActionButton:

```
android.support.v7.preference.PreferenceManager
    .setDefaultValue(this, R.xml.preferences, false);
```

Đoạn mã trên đảm bảo rằng các cài đặt được khởi tạo đúng với giá trị mặc định của chúng. Phương thức PreferenceManager.setDefaultValue() nhận ba đối số:

- **Context của ứng dụng**, chẳng hạn như this.
- **ID tài nguyên** (preferences) cho tệp tài nguyên XML chứa một hoặc nhiều cài đặt.
- **Một giá trị boolean** chỉ ra liệu giá trị mặc định có được thiết lập lại nhiều lần hay không. Khi là false, hệ thống chỉ thiết lập giá trị mặc định nếu phương thức này chưa bao giờ được gọi. Miễn là bạn thiết lập tham số thứ ba là false, bạn có thể gọi phương thức này một cách an toàn mỗi khi MainActivity khởi động mà không ghi đè lên giá trị cài đặt đã lưu của người dùng. Tuy nhiên, nếu bạn thiết lập tham số này là true, phương thức sẽ ghi đè mọi giá trị trước đó bằng các giá trị mặc định.

## 1.9 Đọc giá trị cài đặt đã thay đổi từ Shared Preferences

Khi ứng dụng khởi động, phương thức onCreate() của MainActivity có thể đọc các giá trị cài đặt đã thay đổi và sử dụng các giá trị đó thay vì giá trị mặc định.

Mỗi cài đặt được xác định bằng một cặp khóa-giá trị. Hệ thống Android sử dụng cặp khóa-giá trị này khi lưu hoặc truy xuất cài đặt từ tệp SharedPreferences của ứng dụng. Khi người dùng thay đổi một cài đặt, hệ thống sẽ cập nhật giá trị tương ứng trong tệp SharedPreferences. Để sử dụng giá trị của cài đặt, ứng dụng có thể sử dụng khóa để lấy cài đặt từ tệp SharedPreferences bằng phương thức PreferenceManager.getDefaultSharedPreferences().

Thực hiện các bước sau để thêm đoạn mã đó:

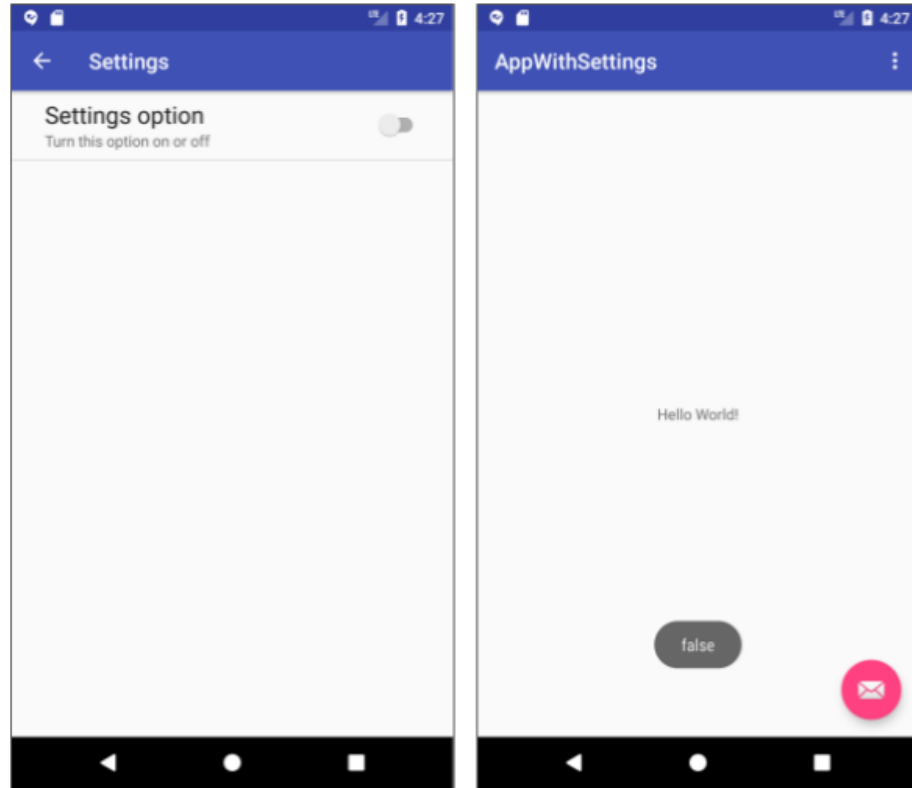
1. Mở **SettingsActivity** và tạo một biến **String** tĩnh để lưu trữ khóa cho giá trị:

```
public static final String  
    KEY_PREF_EXAMPLE_SWITCH = "example_switch";
```

2. Mở **MainActivity** và thêm đoạn sau vào cuối phương thức **onCreate()**:

```
SharedPreferences sharedPref =  
    android.support.v7.preference.PreferenceManager  
        .getDefaultSharedPreferences(this);  
  
Boolean switchPref = sharedPref.getBoolean(  
    SettingsActivity.KEY_PREF_EXAMPLE_SWITCH, false);  
  
Toast.makeText(this, switchPref.toString(),  
    Toast.LENGTH_SHORT).show();
```

3. Chạy ứng dụng và nhấn vào **Settings** để mở **SettingsActivity**.
4. Nhấn vào cài đặt để thay đổi công tắc từ **bật** sang **tắt**, như trong hình bên trái bên dưới.
5. Nhấn nút **Up** trong **SettingsActivity** để quay lại **MainActivity**. Thông báo **Toast** sẽ xuất hiện trong **MainActivity** với giá trị của cài đặt, như trong hình bên phải bên dưới.
6. Lặp lại các bước này để thấy thông báo **Toast** thay đổi khi bạn thay đổi cài đặt.



Đoạn mã trên sử dụng các thành phần sau:

- `android.support.v7.preference.PreferenceManager.getDefaultSharedPreferences(this)` để lấy cài đặt dưới dạng đối tượng `SharedPreferences` (`sharedPref`).
- `getBoolean()` để lấy giá trị Boolean của cài đặt sử dụng khóa `KEY_PREF_EXAMPLE_SWITCH` (được định nghĩa trong `SettingsActivity`) và gán nó vào `switchPref`. Nếu không có giá trị cho khóa này, phương thức `getBoolean()` sẽ đặt `switchPref` thành `false`. Đối với các loại giá trị khác như chuỗi (`String`), số nguyên (`int`), hoặc số thực (`float`), bạn có thể sử dụng các phương thức `getString()`, `getInt()`, hoặc `getFloat()` tương ứng.
- `Toast.makeText()` và `show()` để hiển thị giá trị của cài đặt `switchPref`.

Mỗi khi **MainActivity** khởi động hoặc khởi động lại, phương thức `onCreate()` sẽ đọc giá trị cài đặt để sử dụng trong ứng dụng. Phương thức `Toast.makeText()` sẽ được thay thế bằng một phương thức khởi tạo cài đặt.

Bây giờ, bạn đã có một **Settings Activity** hoạt động trong ứng dụng của mình.

## TASK 1 mã giải pháp



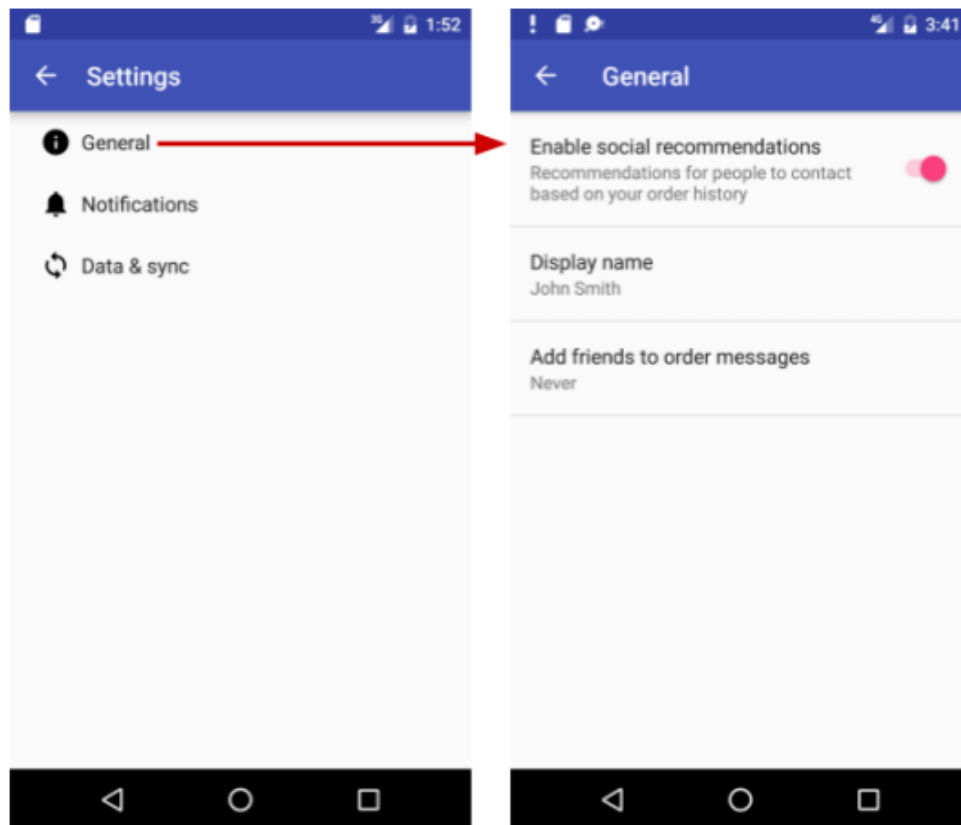
## Dự án Android Studio: AppWithSettings

### TASK 2: : Sử dụng mẫu Hoạt động cài đặt

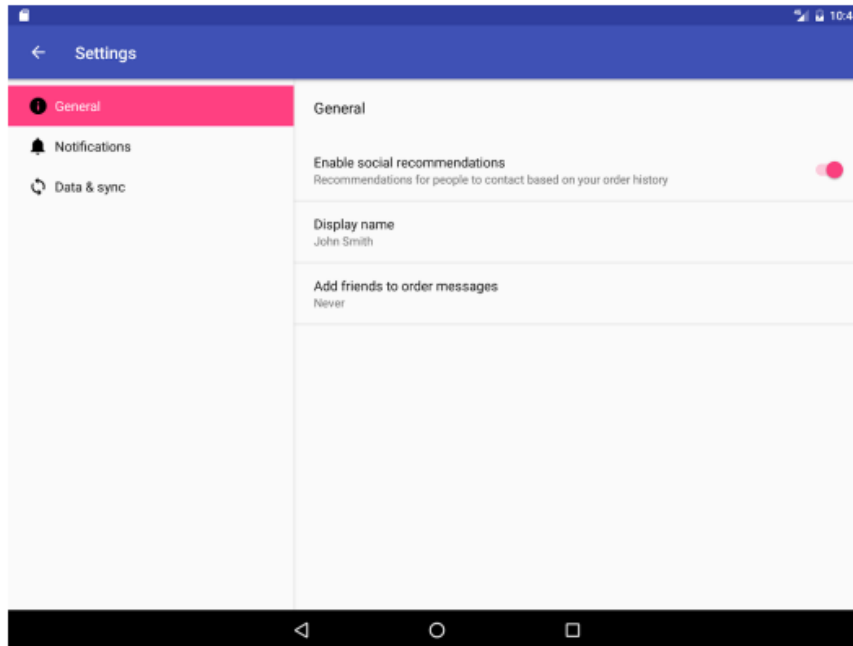
Nếu bạn cần tạo nhiều màn hình con trong phần cài đặt và muốn tận dụng màn hình có kích thước lớn trên máy tính bảng, đồng thời vẫn duy trì khả năng tương thích với các phiên bản Android cũ hơn dành cho máy tính bảng, Android Studio cung cấp một lối tắt: mẫu **Settings Activity**.

Trong nhiệm vụ trước, bạn đã học cách sử dụng một **Activity cài đặt trống** và một **Fragment trống** để thêm một cài đặt vào ứng dụng. **Nhiệm vụ 2** sẽ hướng dẫn bạn cách sử dụng mẫu **Settings Activity** có sẵn trong Android Studio để:

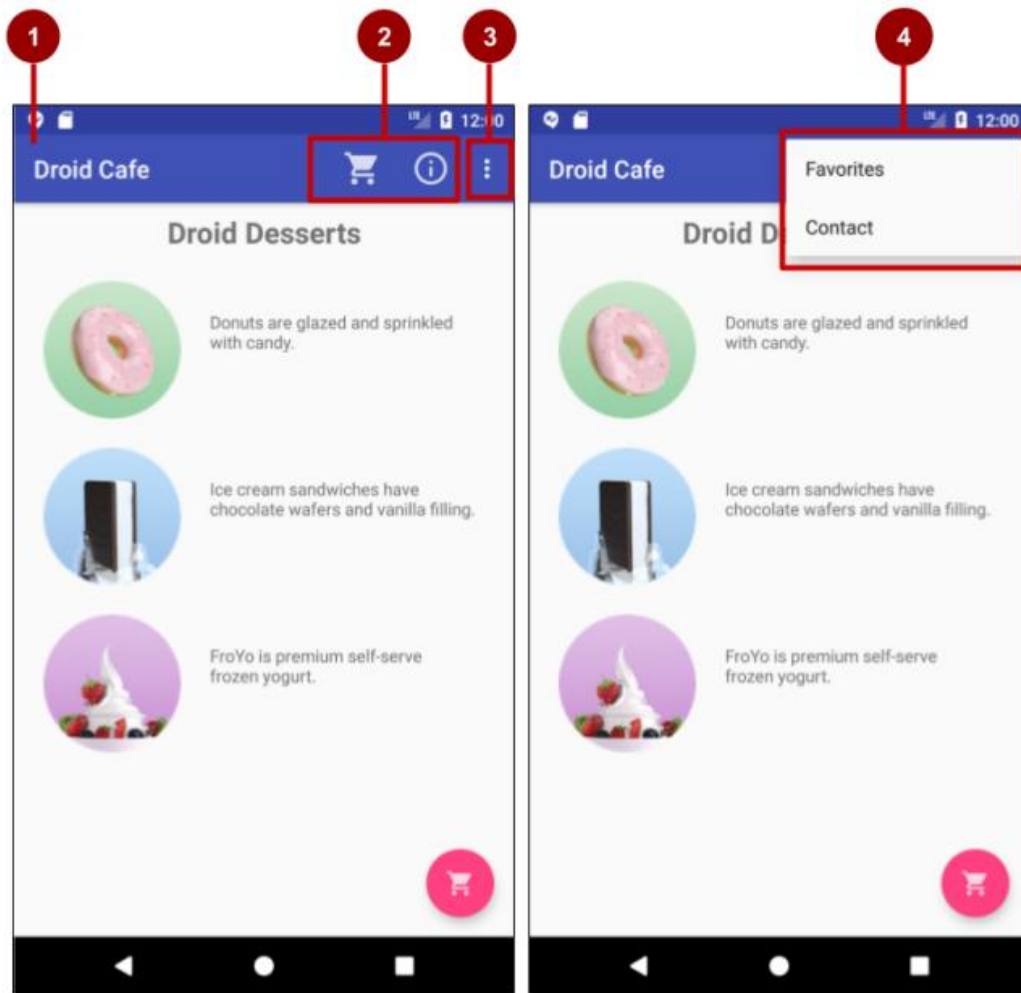
- Chia nhiều cài đặt thành các nhóm.
- Tùy chỉnh các cài đặt và giá trị của chúng.
- Hiển thị màn hình Cài đặt chính với một liên kết tiêu đề cho từng nhóm cài đặt, chẳng hạn như **General** cho cài đặt chung, như minh họa trong hình dưới đây.



- Hiển thị bố cục màn hình **chính/chi tiết** với một liên kết tiêu đề cho từng nhóm cài đặt ở bên trái (phần **chính**), và nhóm cài đặt tương ứng ở bên phải (phần **chi tiết**), như minh họa trong hình dưới đây.



Trong bài thực hành trước, bạn đã tạo một ứng dụng có tên **DroidCafeOptionsUp** bằng cách sử dụng mẫu **Basic Activity**, mẫu này cung cấp một **menu tùy chọn** trong thanh ứng dụng, như minh họa dưới đây.



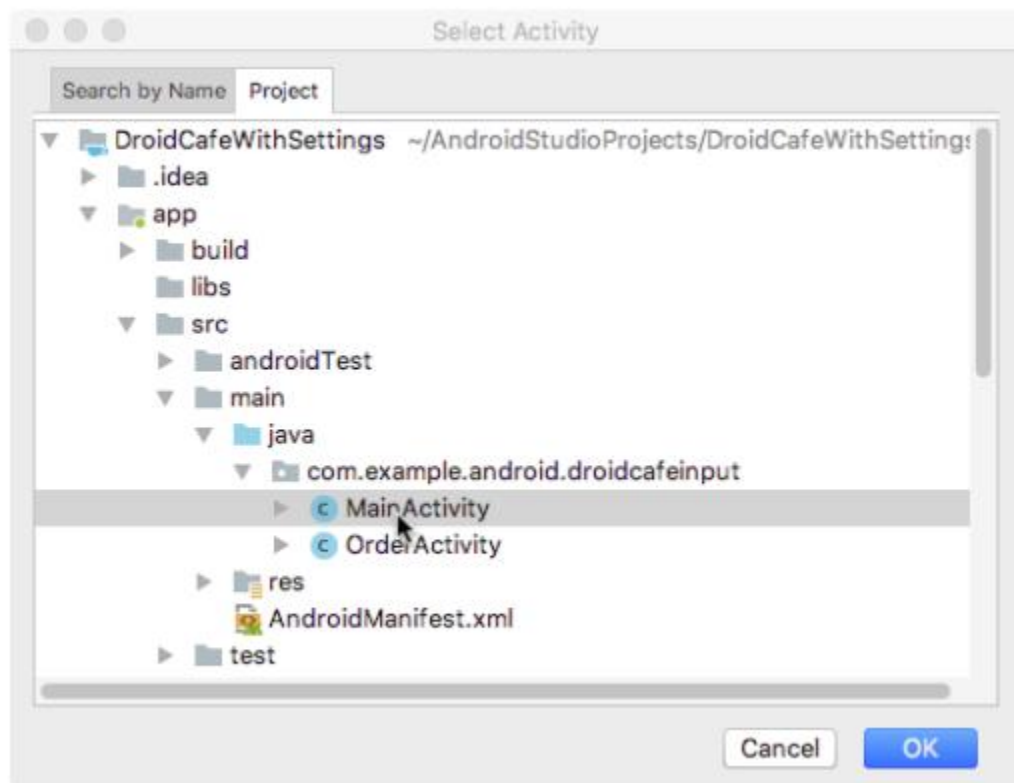
Chú thích cho hình minh họa ở trên:

1. Thanh ứng dụng (App bar)
2. Biểu tượng hành động của menu tùy chọn (Options menu action icons)
3. Nút tràn (Overflow button)
4. Menu tràn tùy chọn (Options overflow menu)

## 2.1 Khám phá mẫu Settings Activity

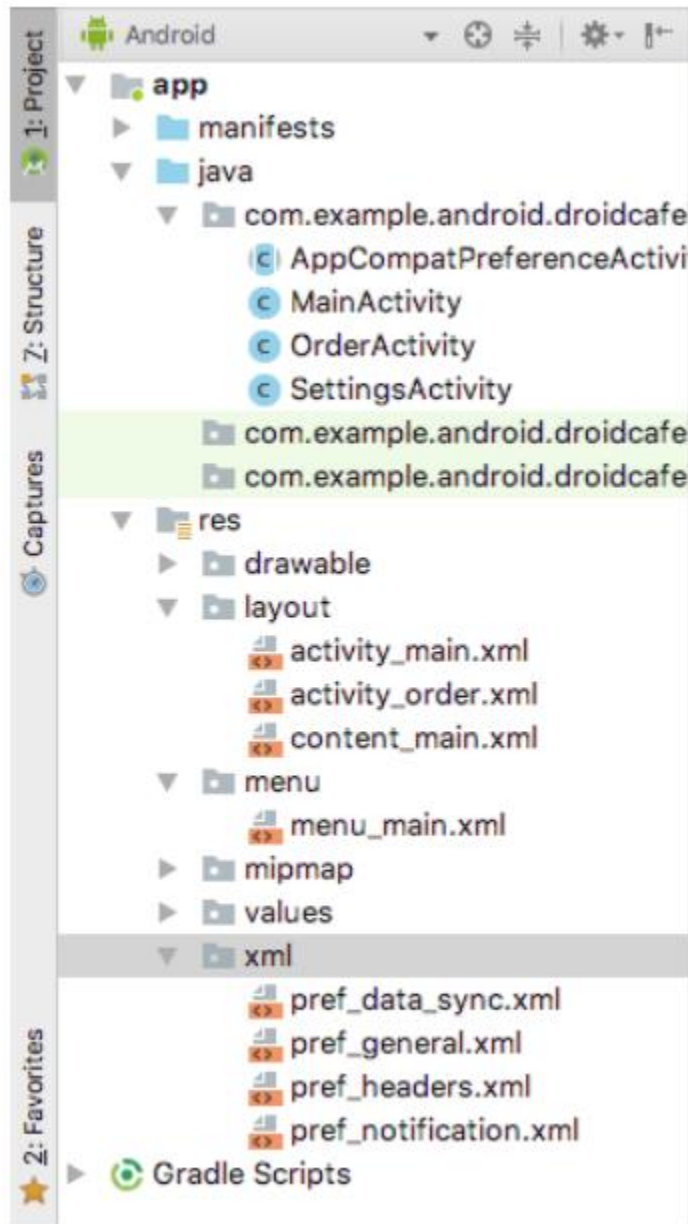
Để thêm mẫu **Settings Activity** vào một dự án ứng dụng trong Android Studio, hãy làm theo các bước sau:

1. Sao chép thư mục dự án **DroidCafeOptionsUp** và đổi tên thành **DroidCafeWithSettings**. Chạy ứng dụng để đảm bảo nó hoạt động đúng.
2. Trong **Project > Android**, chọn **app**, sau đó chọn **New > Activity > Settings Activity**.
3. Trong hộp thoại xuất hiện, chấp nhận tên Activity mặc định (**SettingsActivity** là tên được đề xuất) và tiêu đề (**Settings**).
4. Nhấp vào **dấu ba chấm** ở cuối menu **Hierarchical Parent**, sau đó trong hộp thoại **Select Activity**, chọn tab **Project** (xem hình minh họa bên dưới).
5. Mở rộng thư mục **DroidCafeWithSettings > app > src > main > java > com.example.android.droidcafeinput**, chọn **MainActivity** làm Activity cha, như minh họa trong hình. Nhấn **OK**.



Bạn chọn **MainActivity** làm **Activity cha** để nút **Up** trên thanh ứng dụng trong **SettingsActivity** đưa người dùng quay lại **MainActivity**. Việc chọn **Activity cha** sẽ tự động cập nhật tệp **AndroidManifest.xml** để hỗ trợ điều hướng bằng nút **Up**.

6. Nhấp vào **Finish**.
7. Trong **Project > Android**, mở rộng thư mục **app > res > xml** để xem các tệp XML được tạo bởi mẫu **Settings Activity**.



Bạn có thể mở và tùy chỉnh các tệp XML để thêm hoặc chỉnh sửa các cài đặt theo ý muốn:

- **pref\_data\_sync.xml**: Bố cục **PreferenceScreen** cho cài đặt "Data & sync".
- **pref\_general.xml**: Bố cục **PreferenceScreen** cho cài đặt "General".
- **pref\_headers.xml**: Bố cục tiêu đề cho màn hình chính của **Settings**.

- **pref\_notification.xml**: Bố cục **PreferenceScreen** cho cài đặt "Notifications".

Các tệp XML trên sử dụng nhiều **lớp con của Preference** thay vì **View**, trong đó một số lớp con cung cấp **container** cho bố cục chứa nhiều cài đặt khác nhau. Ví dụ, **PreferenceScreen** đại diện cho một **Preference** cấp cao nhất, đóng vai trò là **gốc của một hệ thống Preference**. Các tệp XML trên sử dụng **PreferenceScreen** ở đầu mỗi màn hình cài đặt. Các lớp con khác của **Preference** cung cấp giao diện người dùng phù hợp để người dùng có thể thay đổi cài đặt theo ý muốn. Ví dụ:

- **CheckBoxPreference**: Một hộp kiểm cho phép bật hoặc tắt một cài đặt.
- **ListPreference**: Một hộp thoại chứa danh sách các nút radio để chọn một tùy chọn.
- **SwitchPreference**: Một tùy chọn hai trạng thái có thể chuyển đổi (chẳng hạn như bật/tắt hoặc true/false).
- **EditTextPreference**: Một hộp thoại chứa **EditText** để nhập văn bản.
- **RingtonePreference**: Một hộp thoại hiển thị danh sách nhạc chuông có sẵn trên thiết bị.

Mẹo: Bạn có thể chỉnh sửa các tệp XML để thay đổi cài đặt mặc định và tùy chỉnh văn bản cài đặt. Tất cả các chuỗi được sử dụng trong hoạt động cài đặt, chẳng hạn như tiêu đề cài đặt, mảng chuỗi cho danh sách và mô tả cài đặt, được định nghĩa là tài nguyên chuỗi trong tệp strings.xml. Chúng được đánh dấu bằng các chú thích như sau:

```
<!-- Strings related to Settings -->
```

```
<!-- Example General Settings -->
```

**Tip:** You can edit the XML files to change the default settings and customize the setting text. All strings used in the settings activity, such as setting titles, string arrays for lists, and setting descriptions, are defined as string resources in a `strings.xml` file. They are marked by comments such as the following:

```
<!-- Strings related to Settings -->
<!-- Example General settings -->
```

Mẫu **Settings Activity** cũng tạo ra các tệp sau:

- **SettingsActivity** trong thư mục `java/com.example.android.projectname`, bạn có thể sử dụng trực tiếp. Đây là **Activity** hiển thị cài đặt. **SettingsActivity** kế

thừa từ **AppCompatActivity** để duy trì khả năng tương thích với các phiên bản Android cũ hơn.

- **AppCompatActivity** trong thư mục `java/com.example.android.projectname`, cũng có thể được sử dụng nguyên vẹn. Đây là một **Activity trợ giúp**, hỗ trợ **SettingsActivity** duy trì tính tương thích ngược với các phiên bản Android trước đó.

---

## 2.2 Thêm mục menu Cài đặt và kết nối với Activity

Như bạn đã học trong một bài thực hành khác, bạn có thể chỉnh sửa tệp **menu\_main.xml** để thêm hoặc xóa các mục menu trong **Options Menu**.

1. Trong **Project > Android**, mở rộng thư mục **res**, sau đó mở tệp **menu\_main.xml**. Nhấp vào tab **Text** để hiển thị mã XML.
2. Thêm một mục menu mới có tên **Settings** với resource id là **action\_settings**:

```
<item
    android:id="@+id/action_settings"
    android:orderInCategory="50"
    android:title="Settings"
    app:showAsAction="never" />
```

Bạn đặt giá trị "never" cho thuộc tính `app:showAsAction` để mục **Settings** chỉ xuất hiện trong menu tràn (overflow options menu) mà không hiển thị trực tiếp trên thanh ứng dụng (app bar), vì nó không cần được sử dụng thường xuyên. Bạn đặt giá trị "50" cho thuộc tính `android:orderInCategory` để **Settings** xuất hiện bên dưới **Favorites** (được đặt là "30") nhưng trên **Contact** (được đặt là "100").

3. Trích xuất chuỗi tài nguyên cho "Settings" trong thuộc tính `android:title` và đặt tên tài nguyên là **settings**.
4. Mở **MainActivity** và tìm khối switch case trong phương thức `onOptionsItemSelected()`, phương thức này xử lý thao tác nhấn vào các mục trong menu tùy chọn. Dưới đây là một đoạn mã mẫu của phương thức, hiển thị trường hợp đầu tiên (case cho `action_order`):

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_order:
            Intent intent = new Intent(MainActivity.this,
                OrderActivity.class);
            intent.putExtra(EXTRA_MESSAGE, mOrderMessage);
            startActivity(intent);
            return true;
        case R.id.action_status:
            // Code for action_status and other cases...
            return true;
    }
    return super.onOptionsItemSelected(item);
}

```

5. Lưu ý rằng trong đoạn mã trên, case đầu tiên sử dụng Intent để khởi chạy OrderActivity. Hãy thêm một case mới cho action\_settings vào khối switch case với mã Intent tương tự để khởi chạy SettingsActivity (nhưng không có intent.putExtra).

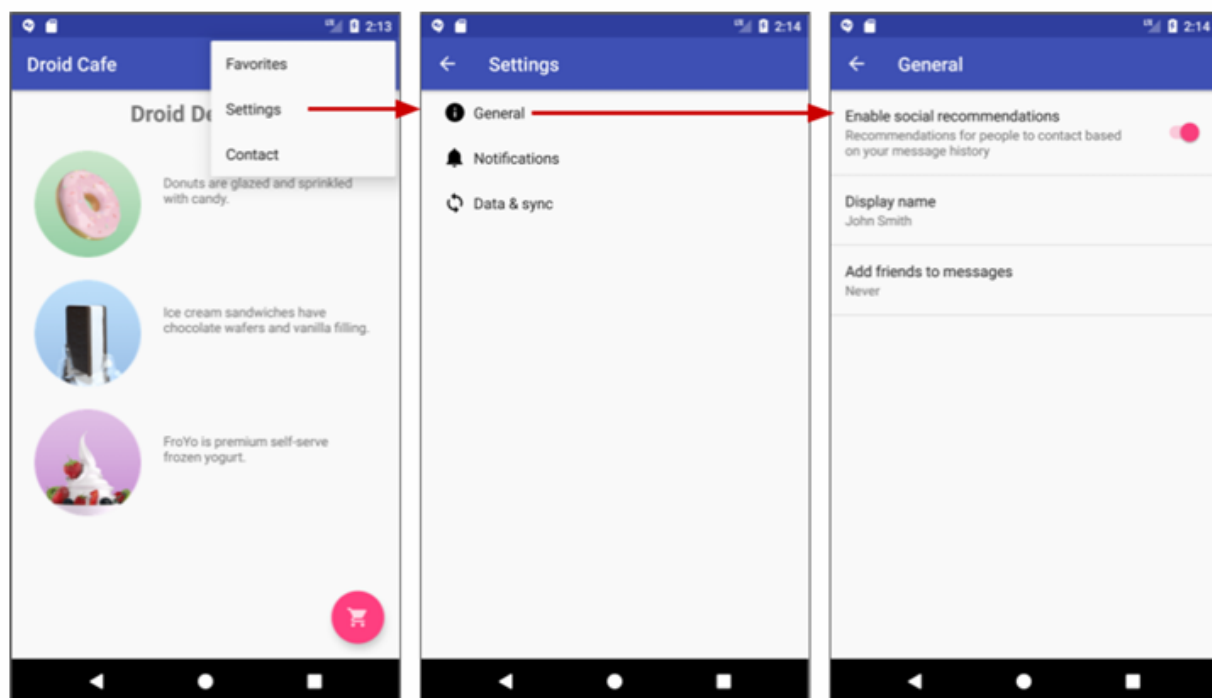
```

case R.id.action_settings:
    Intent settingsIntent = new Intent(this,
        SettingsActivity.class);

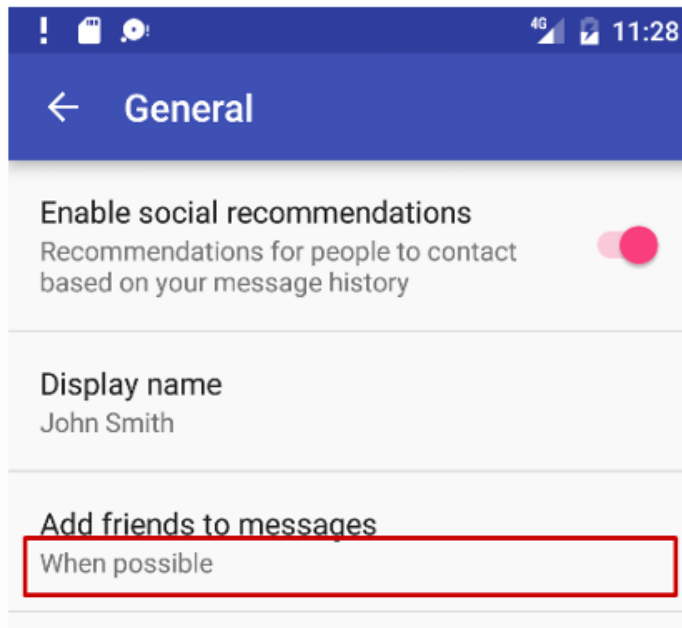
```

5. Chạy ứng dụng trên điện thoại hoặc trình giả lập để xem cách **Settings Activity** xử lý kích thước màn hình điện thoại.
6. Nhấn vào biểu tượng menu tràn (overflow icon) trong menu tùy chọn, sau đó nhấn **Settings** để mở **Settings Activity**, như hình bên trái trong hình minh họa dưới đây.
7. Nhấn vào từng tiêu đề cài đặt (**General**, **Notifications**, và **Data & sync**), như hình trung tâm trong hình minh họa, để xem nhóm cài đặt trên từng màn hình con của màn hình **Settings**, được hiển thị ở phía bên phải của hình minh họa.
8. Nhấn vào nút **Up** trong **Settings Activity** để quay lại **MainActivity**.





Bạn sử dụng mã mẫu của **Settings Activity** như nguyên bản. Nó không chỉ cung cấp bố cục phù hợp cho cả màn hình điện thoại và máy tính bảng, mà còn có chức năng lắng nghe sự thay đổi trong cài đặt và cập nhật phần **summary** để phản ánh thay đổi đó. Ví dụ, nếu bạn thay đổi cài đặt "**Add friends to messages**" (với các tùy chọn **Always**, **When possible**, hoặc **Never**), thì lựa chọn của bạn sẽ xuất hiện trong phần tóm tắt bên dưới cài đặt đó.



Nói chung, bạn không cần thay đổi mã mẫu của **Settings Activity** để tùy chỉnh **Activity** cho các cài đặt trong ứng dụng của mình. Bạn có thể tùy chỉnh **tiêu đề cài đặt, tóm tắt, các giá trị có thể chọn, và giá trị mặc định** mà không cần chỉnh sửa mã mẫu. Bạn cũng có thể thêm nhiều cài đặt hơn vào các nhóm đã được cung cấp sẵn.

## 2.3 Tùy chỉnh các cài đặt được cung cấp bởi mẫu

Để tùy chỉnh các cài đặt được cung cấp bởi **Settings Activity template**, bạn cần chỉnh sửa **chuỗi ký tự (string)** và **mảng chuỗi (string array)** trong tệp `strings.xml`, cũng như các thuộc tính bố cục trong các tệp trong thư mục `xml`.

Trong bước này, bạn sẽ thay đổi cài đặt **"Data & sync"**.

1. Mở rộng thư mục `res > values` và mở tệp `strings.xml`. Cuộn xuống phần nội dung có chứa dòng bình luận:

```
<!--Strings related to Settings -->
```

```

<string name="pref_header_data_sync">Data & sync</string>

<string name="pref_title_sync_frequency">Sync frequency</string>
<string-array name="pref_sync_frequency_titles">
    <item>15 minutes</item>
    <item>30 minutes</item>
    <item>1 hour</item>
    <item>3 hours</item>
    <item>6 hours</item>
    <item>Never</item>
</string-array>
<string-array name="pref_sync_frequency_values">
    <item>15</item>
    <item>30</item>
    <item>60</item>
    <item>180</item>
    <item>360</item>
    <item>-1</item>
</string-array>
<string-array name="list_preference_entries">
    <item>Entry 1</item>
    <item>Entry 2</item>
    <item>Entry 3</item>
</string-array>
<string-array name="list_preference_entry_values">
    <item>1</item>
    <item>2</item>
    <item>3</item>
</string-array>
<string-array name="multi_select_list_preference_default_value" />
<string name="pref_title_system_sync_settings">System sync settings</string>

```

2. Chỉnh sửa tài nguyên chuỗi `pref_header_data_sync`, hiện đang được đặt là `Data & sync` (& là mã HTML cho dấu &). Thay đổi giá trị thành **Account** (không có dấu ngoặc kép).
3. Bạn nên đổi tên tài nguyên (ứng dụng vẫn hoạt động nếu không đổi tên, nhưng làm vậy giúp mã dễ hiểu hơn).
  - **Nhấp chuột phải** (hoặc **Control + Click** trên macOS) vào tài nguyên `pref_header_data_sync`, chọn **Refactor > Rename**.

- Đổi tên thành **pref\_header\_account**.
  - Nhấp vào tùy chọn **Search in comments and strings** để tìm kiếm trong nhận xét và chuỗi.
  - Nhấp vào **Refactor**.
4. Bạn cũng nên đổi tên tệp XML (ứng dụng vẫn hoạt động nếu không đổi tên, nhưng làm vậy giúp mã dễ hiểu hơn).
- **Nhấp chuột phải** (hoặc **Control + Click** trên macOS) vào tài nguyên `pref_data_sync` trong **Project > Android**, chọn **Refactor > Rename**.
  - Đổi tên thành **pref\_account**.
  - Nhấp vào tùy chọn **Search in comments and strings** để tìm kiếm trong nhận xét và chuỗi.
  - Nhấp vào **Refactor**.
5. Chỉnh sửa tài nguyên chuỗi `pref_title_sync_frequency`, hiện đang được đặt là **Sync frequency**, đổi thành **Market**.
6. **Refactor > Rename** tài nguyên `pref_title_sync_frequency` thành **pref\_title\_account**, như đã làm trước đó.
7. **Refactor > Rename** tài nguyên mảng chuỗi `pref_sync_frequency_titles` thành **pref\_market\_titles**.
8. Thay đổi từng giá trị trong mảng chuỗi `pref_market_titles` (hiện tại là **15 minutes, 30 minutes, 1 hour**, v.v.) thành tên của các thị trường, chẳng hạn như **United States, Canada**, v.v., thay vì các khoảng thời gian.

```
<string-array name="pref_market_titles">
    <item>United States</item>
    <item>Canada</item>
    <item>United Kingdom</item>
    <item>India</item>
    <item>Japan</item>
    <item>Other</item>
</string-array>
```

9. **Refactor > Rename** tài nguyên mảng chuỗi `pref_sync_frequency_values` thành **pref\_market\_values**.

10. Thay đổi từng giá trị trong mảng chuỗi `pref_market_values` (hiện tại là **15, 30, 60**, v.v.) thành các giá trị đại diện cho các thị trường—các viết tắt tương ứng với các quốc gia ở trên, chẳng hạn như **US, CA**, v.v.

```
<string-array name="pref_market_values">
    <item>US</item>
    <item>CA</item>
    <item>UK</item>
    <item>IN</item>
    <item>JA0</item>
    <item>-1</item>
</string-array>
```

11. Cuộn xuống tài nguyên chuỗi `pref_title_system_sync_settings` và chỉnh sửa giá trị (hiện đang được đặt là **System sync settings**) thành **Account settings**.
12. **Refactor > Rename** tài nguyên chuỗi `pref_title_system_sync_settings` thành **pref\_title\_account\_settings**.
13. Mở tệp `pref_account.xml`. Trong bố cục này, phần **ListPreference** định nghĩa cài đặt mà bạn vừa thay đổi. Lưu ý rằng các tài nguyên chuỗi cho các thuộc tính `android:entries`, `android:entryValues`, và `android:title` giờ đã được cập nhật theo các giá trị bạn đã chỉnh sửa trong các bước trước.

```
<ListPreference
    android:defaultValue="180"
    android:entries="@array/pref_market_titles"
    android:entryValues="@array/pref_market_values"
    android:key="sync_frequency"
    android:negativeButtonText="@null"
    android:positiveButtonText="@null"
    android:title="@string/pref_title_account" />
```

14. Thay đổi thuộc tính `android:defaultValue` thành "US".

```
android:defaultValue="US"
```

Vì khóa cho tùy chọn cài đặt này ("sync\_frequency") đã được mã hóa cứng (hard-coded) trong mã Java ở nơi khác, **không thay đổi** thuộc tính `android:key`. Thay vào đó, tiếp tục sử dụng "sync\_frequency" làm khóa cho cài đặt này trong ví dụ này. Nếu bạn đang tùy chỉnh cài đặt cho một ứng dụng thực tế, bạn nên dành thời gian để thay đổi tất cả các khóa đã được mã hóa cứng trong toàn bộ mã nguồn.

**Lưu ý:** Tại sao không sử dụng tài nguyên chuỗi (string resource) cho khóa (key)? Bởi vì tài nguyên chuỗi có thể được bản địa hóa (localized) cho các ngôn ngữ khác nhau bằng cách sử dụng các tệp XML đa ngôn ngữ. Nếu khóa chuỗi vô tình bị dịch cùng với các chuỗi khác, điều này có thể khiến ứng dụng bị lỗi (crash).

## 2.4 Thêm mã để đặt giá trị mặc định cho cài đặt

Để thêm mã đặt giá trị mặc định cho cài đặt, hãy làm theo các bước sau:

1. Mở **MainActivity** và tìm phương thức `onCreate()`.
2. Thêm các câu lệnh `PreferenceManager.setDefaultValues` sau vào cuối phương thức `onCreate()`:

```
PreferenceManager.setDefaultValues(this, R.xml.pref_general, false);  
PreferenceManager.setDefaultValues(this, R.xml.pref_notification, false);  
PreferenceManager.setDefaultValues(this, R.xml.pref_account, false);
```

Các giá trị mặc định đã được chỉ định trong tệp XML bằng thuộc tính `android:defaultValue`. Tuy nhiên, các câu lệnh trên đảm bảo rằng tệp **SharedPreferences** được khởi tạo đúng cách với các giá trị mặc định.

Phương thức `setDefaultValues()` nhận ba đối số:

- **Ngữ cảnh (context) của ứng dụng**, ví dụ: `this`.

- **ID tài nguyên của tệp XML bố cục cài đặt**, trong đó chứa các giá trị mặc định được đặt bằng thuộc tính `android:defaultValue`.
- **Một giá trị boolean** cho biết liệu các giá trị mặc định có nên được đặt nhiều lần hay không. Khi đặt **false**, hệ thống chỉ đặt các giá trị mặc định **một lần** (khi phương thức này được gọi lần đầu tiên). Do đó, bạn có thể gọi phương thức này mỗi khi **Activity** khởi động mà không làm ghi đè các giá trị đã lưu của người dùng. Nếu đặt **true**, phương thức sẽ ghi đè bất kỳ giá trị nào trước đó bằng các giá trị mặc định.

## 2.5 Thêm mã để đọc giá trị của cài đặt

1. Thêm đoạn mã sau vào cuối phương thức `onCreate()` trong **MainActivity**. Bạn có thể thêm ngay sau đoạn mã đã thêm ở bước trước để thiết lập giá trị mặc định cho cài đặt:

```
SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(this);

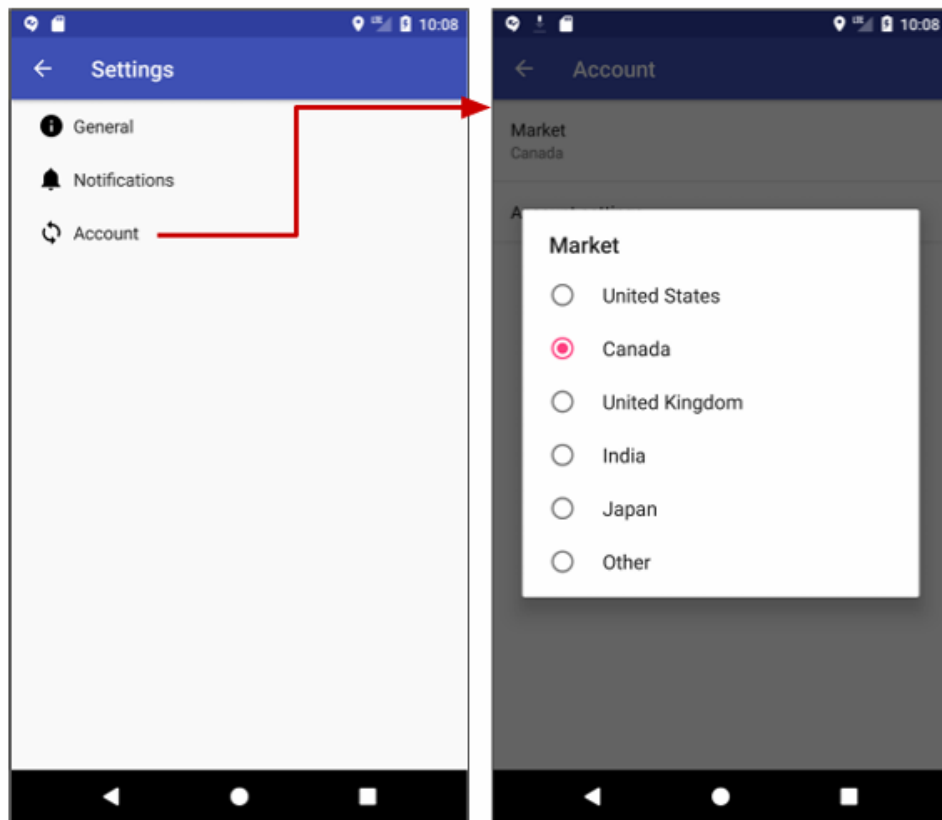
String marketPref = sharedPref.getString("sync_frequency", "-1");

displayToast(marketPref);
```

Như bạn đã học trong bước trước, bạn sử dụng `PreferenceManager.getDefaultSharedPreferences(this)` để lấy cài đặt dưới dạng một đối tượng `SharedPreferences` (`marketPref`). Sau đó, bạn sử dụng `getString()` để lấy giá trị chuỗi của cài đặt sử dụng khóa (`sync_frequency`) và gán nó cho `marketPref`.

Nếu không có giá trị nào cho khóa này, phương thức `getString()` sẽ gán giá trị của `marketPref` là `-1`, đây là giá trị của `Other` trong mảng `pref_market_values`.

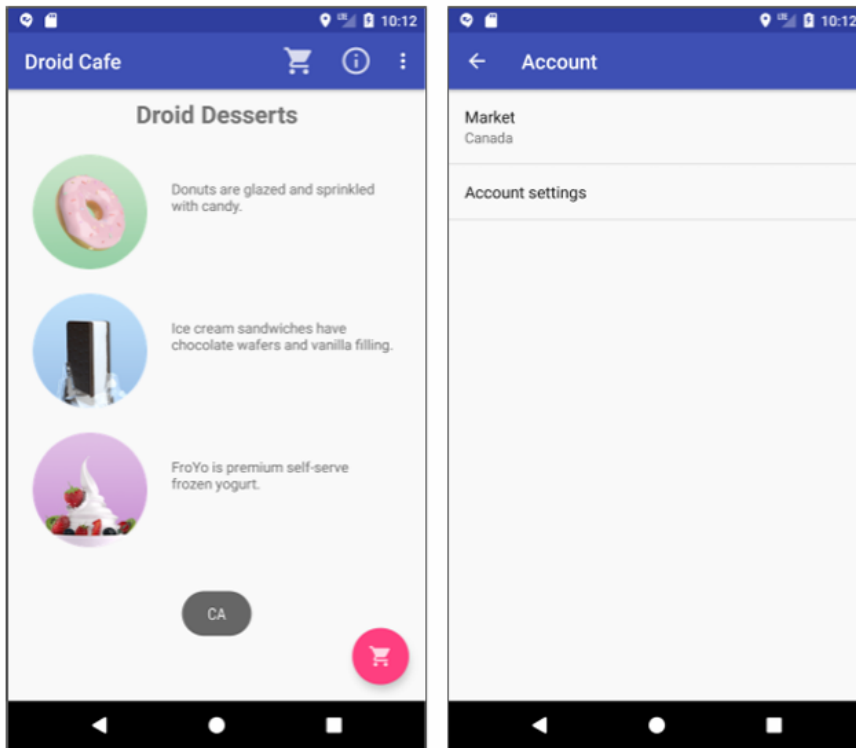
2. Chạy ứng dụng khi màn hình chính của ứng dụng xuất hiện lần đầu tiên, bạn sẽ thấy một Toast xuất hiện ở cuối màn hình. Lần đầu chạy ứng dụng, bạn sẽ thấy "-1" hiển thị trong Toast vì bạn chưa thay đổi cài đặt.
3. Thay đổi cài đặt nhấn vào Settings trong menu tùy chọn. Trong màn hình Settings, nhấn vào Account. Nhấn vào Market và chọn Canada, như minh họa bên dưới.





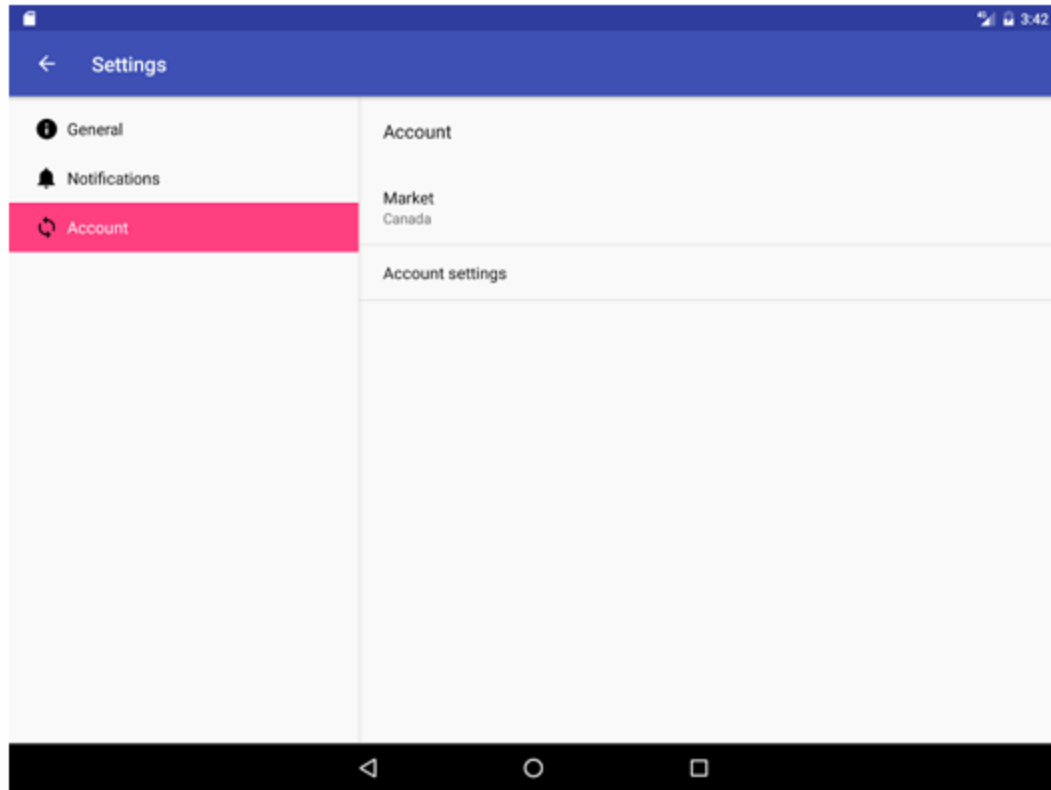
4. Nhấn nút **Up** để quay lại màn hình chính. Nhấn nút **Up** trên thanh ứng dụng để trở về màn hình **Settings**, sau đó nhấn lần nữa để quay lại màn hình chính của ứng dụng.

5. Chạy lại ứng dụng từ **Android Studio**. Khi ứng dụng khởi động lại, bạn sẽ thấy một thông báo **Toast** hiển thị "CA" (đại diện cho **Canada**), xác nhận rằng cài đặt **Market** đã được lưu thành công và vẫn được áp dụng sau khi khởi động lại ứng dụng.



Bạn đã tích hợp thành công **Settings Activity** (Hoạt động Cài đặt) với ứng dụng.

6. Bây giờ hãy chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng. Vì máy tính bảng có màn hình lớn hơn về mặt vật lý, **Android runtime** sẽ tận dụng không gian bổ sung này. Trên máy tính bảng, phần cài đặt và chi tiết sẽ được hiển thị trên cùng một màn hình, giúp người dùng dễ dàng quản lý cài đặt hơn.



## Task 2 mã giải pháp

Dự án Android Studio: **DroidCafeWithSettings**

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: Ứng dụng **DroidCafeWithSettings** hiển thị cài đặt đúng cách trên màn hình có kích thước máy tính bảng, nhưng nút **Up** trên thanh ứng dụng không đưa người dùng trở về **MainActivity** như trên màn hình điện thoại. Nguyên nhân là do có ba phương thức **onOptionsItemSelected()**—mỗi phương thức tương ứng với một **Fragment**—trong **SettingsActivity**. Ứng dụng sử dụng đoạn mã sau để khởi động lại **SettingsActivity** khi người dùng nhấn nút **Up**:

```
startActivity(new Intent(getActivity(), SettingsActivity.class));
```

Nội dung trên là hành vi phù hợp trên màn hình điện thoại, nơi các tiêu đề cài đặt (**General**, **Notifications**, và **Account**) xuất hiện trên một màn hình riêng (**SettingsActivity**). Sau khi thay đổi một cài đặt, bạn muốn khi người dùng nhấn nút **Up**, họ sẽ quay lại màn hình tiêu đề cài đặt trong **SettingsActivity**. Nếu tiếp tục nhấn **Up**, họ sẽ quay lại **MainActivity**.

Tuy nhiên, trên máy tính bảng, các tiêu đề luôn hiển thị trong ngăn bên trái (trong khi cài đặt hiển thị ở ngăn bên phải). Do đó, khi nhấn nút **Up**, người dùng không quay lại **MainActivity**.

Hãy tìm cách làm cho nút **Up** hoạt động đúng trong **SettingsActivity** trên màn hình có kích thước máy tính bảng.

**Gợi ý:** Có nhiều cách để khắc phục vấn đề này, nhưng hãy xem xét các bước sau:

1. Thêm một tệp **dimens.xml** khác để hỗ trợ các màn hình có kích thước lớn hơn **600dp**.
  - Khi ứng dụng chạy trên một thiết bị cụ thể, tệp **dimens.xml** phù hợp sẽ được chọn dựa trên bộ định tính của nó.
  - Bạn có thể thêm một tệp **dimens.xml** khác với bộ định tính **Smallest Screen Width** được đặt thành **600dp (sw600dp)**, như bạn đã học trong phần thực hành về hỗ trợ chế độ ngang và kích thước màn hình, để áp dụng cho các thiết bị có màn hình lớn như máy tính bảng.
2. Thêm tài nguyên **bool** sau vào giữa các thẻ `<resources>` và `</resources>` trong tệp **dimens.xml (sw600dp)**, tệp này sẽ được chọn tự động cho máy tính bảng:

```
<bool name="isTablet">true</bool>
```

3. Thêm tài nguyên **bool** sau vào tệp **dimens.xml** tiêu chuẩn, tệp này sẽ được chọn khi ứng dụng chạy trên bất kỳ thiết bị nào **không** có màn hình lớn:

```
<bool name="isTablet">false</bool>
```

- Trong **SettingsActivity**, bạn có thể thêm một khối **if-else** vào cả ba phương thức **onOptionsItemSelected()** (mỗi phương thức cho một **Fragment**) để kiểm tra xem **isTablet** có giá trị **true** hay không. Nếu **isTablet** là **true**, mã của bạn có thể điều hướng hành động của nút **Up** về **MainActivity**.

## Mã giải pháp thử thách

Dự án Android Studio: DroidCafeWithSettingsChallenge

### Bản tóm tắt

Người dùng mong đợi có thể điều hướng đến cài đặt ứng dụng bằng cách nhấn vào *Settings* trong thanh điều hướng bên, chẳng hạn như *navigation drawer* hoặc trong menu tùy chọn trên thanh ứng dụng.

Để cung cấp cài đặt người dùng cho ứng dụng, hãy cung cấp một *Activity* cho cài đặt:

- Sử dụng một *Activity* chứa *PreferenceFragment* để hiển thị cài đặt ứng dụng.
- Để duy trì khả năng tương thích với *AppCompatActivity* và thư viện *android.support.v7.preference*, hãy sử dụng *PreferenceFragmentCompat* thay vì *PreferenceFragment*.

Hiển thị từng fragment trong settings activity:

- Nếu lớp activity mở rộng từ *Activity* và lớp fragment mở rộng từ *PreferenceFragment*, sử dụng *getFragmentManager()*.
- Nếu lớp activity mở rộng từ *AppCompatActivity* và lớp fragment mở rộng từ *PreferenceFragmentCompat*, sử dụng *getSupportFragmentManager()*.
- Để liên kết tài nguyên cài đặt *preferences.xml* với fragment, sử dụng *setPreferencesFromResource()*.
- Để đặt giá trị mặc định cho cài đặt, sử dụng *PreferenceManager.setDefaultValues()*.
- Để kết nối mục menu Settings với settings activity, sử dụng một *Intent*.

Thêm tệp tài nguyên XML cho cài đặt:

1. Tạo một thư mục tài nguyên mới (*File > New > Android Resource Directory*).
2. Trong menu thả xuống *Resource type*, chọn *xml*. Thư mục *xml* sẽ xuất hiện bên trong thư mục *res*.
3. Nhấp vào *xml* và chọn *File > New > XML resource file*.
4. Nhập tùy chọn làm tên của tệp XML. Tệp *preferences.xml* xuất hiện bên trong thư mục *xml*.

Thêm các điều khiển giao diện người dùng như công tắc bật tắt, với các thuộc tính trong tệp XML tùy chọn:

- Để duy trì khả năng tương thích với *AppCompatActivity*, hãy sử dụng Phiên bản thư viện *android.support.v7.preference*. Ví dụ, sử dụng *SwitchPreferenceCompat* cho các công tắc bật tắt. Sử dụng các thuộc tính với từng thành phần giao diện người dùng cho cài đặt:
- *android:defaultValue* là giá trị của cài đặt khi ứng dụng khởi động lần đầu tiên.
- *android:title* là tiêu đề cài đặt hiển thị cho người dùng.
- *android:key* là khóa được sử dụng để lưu trữ giá trị cài đặt.
- *android:summary* là văn bản hiển thị cho người dùng xuất hiện trong cài đặt.

Lưu và đọc các giá trị cài đặt:

- Khi ứng dụng khởi động, phương thức *MainActivity onCreate()* có thể đọc các giá trị cài đặt đã thay đổi và sử dụng các giá trị đã thay đổi thay vì giá trị mặc định.
- Mỗi cài đặt được xác định bằng cặp khóa-giá trị. Hệ thống Android sử dụng cặp khóa-giá trị này khi lưu hoặc truy xuất cài đặt từ tệp *SharedPreferences* cho ứng dụng của bạn. Khi người dùng thay đổi cài đặt, hệ thống sẽ cập nhật giá trị tương ứng trong Tệp *SharedPreferences*.

- Để sử dụng giá trị của cài đặt, ứng dụng của bạn có thể sử dụng phím để nhận cài đặt từ Tập SharedPreferences .

- Ứng dụng của bạn đọc các giá trị cài đặt từ SharedPreferences bằng cách sử dụng

PreferenceManager.getDefaultSharedPreferences() và nhận từng giá trị cài đặt sử dụng .getString , .getBoolean , v.v.

## **Các khái niệm liên quan**

Tài liệu khái niệm liên quan có trong 9.2: App settings .

## **Tìm hiểu thêm**

Tài liệu về Android Studio: Hướng dẫn sử dụng Android Studio

Tài liệu dành cho nhà phát triển Android:

- Settings (overview)
- Preference
- PreferenceFragment
- PreferenceFragmentCompat
- Fragment ● SharedPreferences
- Save key-value data
- Support different screen sizes

Thông số thiết kế Material Design: Cài đặt Android

Tràn ngăn xếp:

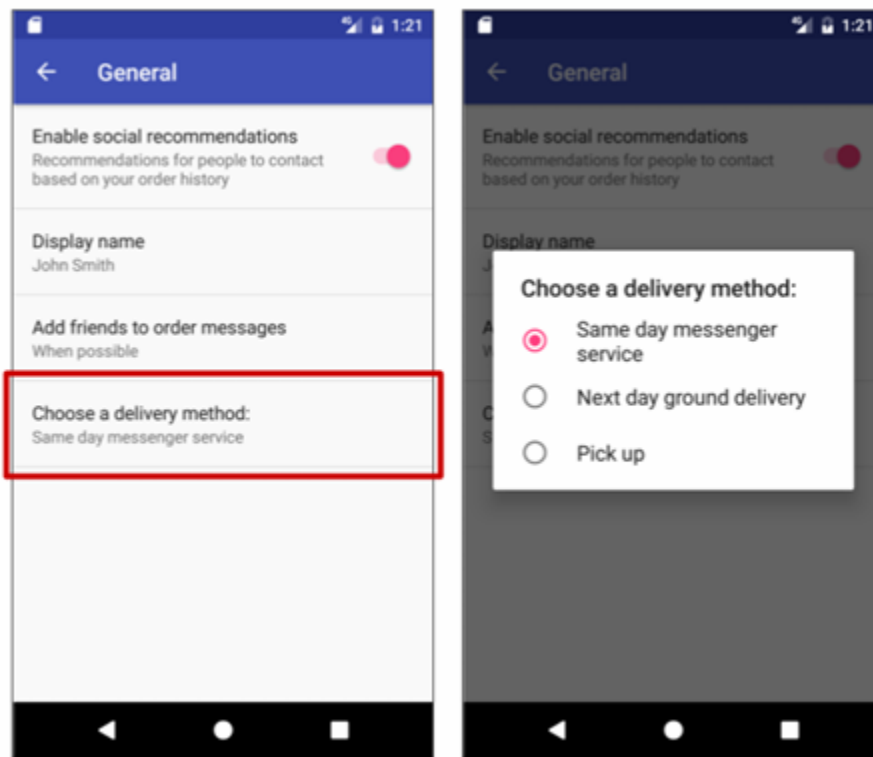
- How does one get dimens.xml into Android Studio?
- Determine if the device is a smartphone or tablet?

## **Bài tập về nhà**

## Xây dựng và chạy ứng dụng

Mở dự án ứng dụng *DroidCafeWithSettings*.

1. Thêm một *ListPreference* (hộp thoại với các nút radio) vào phần cài đặt chung. Đặt hộp thoại này trong màn hình cài đặt **General**, bên dưới *ListPreference* có tiêu đề "**Add friends to order messages**".
2. **Chỉnh sửa mảng chuỗi dùng cho *ListPreference* để bao gồm tiêu đề "*Choose a delivery method.*"** Sử dụng cùng các phương thức giao hàng giống như các tùy chọn **radio buttons** trong *OrderActivity*.
3. **Hiển thị phương thức giao hàng do người dùng chọn trong cùng một thông báo *Toast* với cài đặt *Market*.**
4. **Thử thách thêm. Hiển thị phương thức giao hàng được chọn làm văn bản mô tả (summary) bên dưới tiêu đề *ListPreference*. Cập nhật văn bản này mỗi khi người dùng thay đổi tùy chọn giao hàng.**



**Answer:**

Câu hỏi 1: Trong dự án *DroidCafeWithSettings*, các **mảng mục (entries)** và **mảng giá trị (values)** của *ListPreference* thường được định nghĩa trong **strings.xml**, vì đây là nơi lưu trữ các chuỗi văn bản sử dụng trong ứng dụng, bao gồm cả danh sách các tùy chọn cho *ListPreference*. Chọn một:

- pref\_general.xml
- strings.xml
- menu\_main.xml
- activity\_main.xml
- content\_main.xml

Câu hỏi 2: Trong tệp nào của dự án *DroidCafeWithSettings*, bạn sử dụng mảng mục nhập và mảng giá trị để thiết lập *ListPreference*, đồng thời đặt khóa *ListPreference* và giá trị mặc định? Chọn một:

- pref\_general.xml
- strings.xml
- menu\_main.xml
- content\_main.xml
- SettingsActivity.java

Câu hỏi 3: Làm cách nào để thiết lập cài đặt Hoạt động và Đoạn bằng *SwitchPreference* cho giao diện người dùng mà vẫn tương thích với thư viện *appcompat v7* để tương thích ngược với các phiên bản cũ hơn của Android?

- Sử dụng hoạt động cài đặt mở rộng Hoạt động, một đoạn mở rộng *PreferenceFragment* và *SwitchPreference* cho giao diện người dùng.
- Thay đổi *MainActivity* để mở rộng Hoạt động.
- Sử dụng hoạt động cài đặt mở rộng *AppCompatActivity*, một đoạn mở rộng *PreferenceFragmentCompat* và *SwitchPreferenceCompat* cho giao diện người dùng.
- Bạn không thể sử dụng một đoạn có *SwitchPreference* mà vẫn tương thích với thư viện ứng dụng *v7*



## **Gửi ứng dụng của bạn để chấm điểm**

### **Hướng dẫn cho học sinh chấm điểm**

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

- Phương thức `onCreate()` đọc cài đặt `deliveryPref` bằng `sharedPref.getString()` .
- Tập `pref_general.xml` bao gồm `ListPreference` sử dụng một mảng các lựa chọn giao hàng cho các mục nhập của nó.
- Điểm thưởng: Câu lệnh `bindPreferenceSummaryToValue(findPreference("delivery"))` đã được thêm vào phương thức `onCreate()` của lớp `GeneralPreferenceFragment` trong `SettingsActivity` để hiển thị lựa chọn giao hàng trong bản tóm tắt tùy chọn.

## **1.2) Room, LiveData và ViewModel**

Hệ điều hành Android cung cấp nền tảng vững chắc để xây dựng các ứng dụng chạy tốt trên nhiều loại thiết bị và kiểu dáng. Tuy nhiên, các vấn đề như vòng đời phức tạp và việc thiếu kiến trúc ứng dụng được khuyến nghị khiến việc viết các ứng dụng mạnh mẽ trở nên khó khăn. Các thành phần kiến trúc Android cung cấp các thư viện cho các tác vụ chung như quản lý vòng đời và tính bền vững của dữ liệu để giúp triển khai kiến trúc được khuyến nghị dễ dàng hơn.

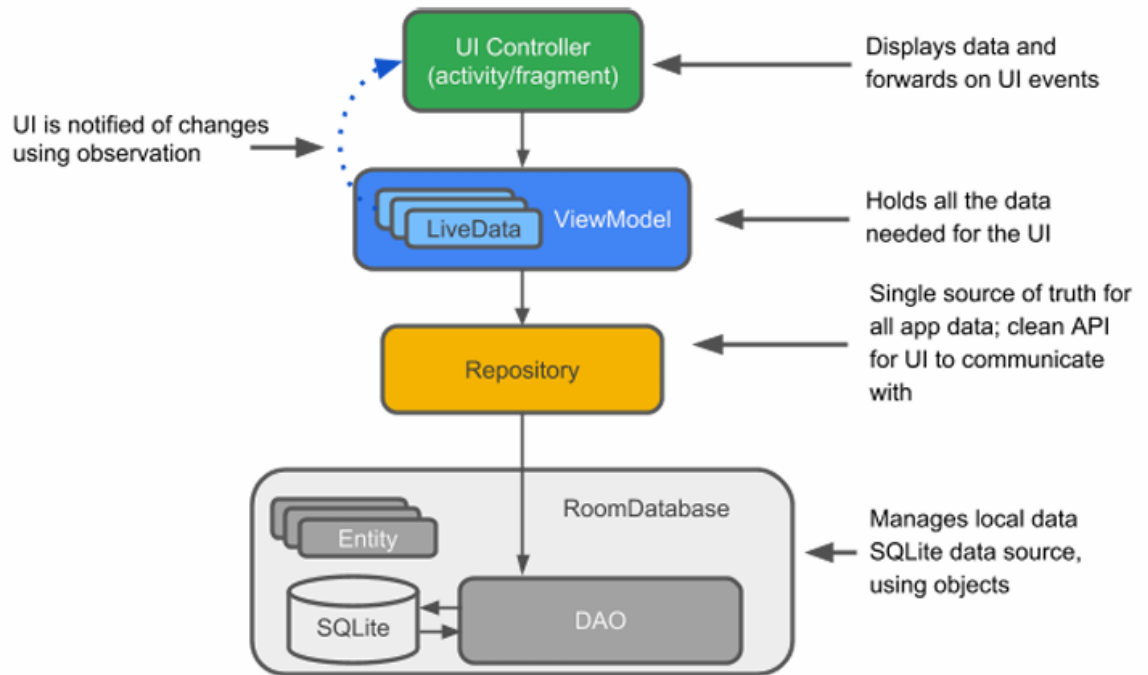
Các thành phần kiến trúc giúp bạn xây dựng cấu trúc ứng dụng theo cách mạnh mẽ, có thể kiểm tra và bảo trì với ít mã lệnh hơn.

### **Các thành phần kiến trúc được đề xuất là gì?**

Khi nói đến kiến trúc, trước tiên, việc nhìn thấy bức tranh toàn cảnh sẽ hữu ích. Để giới thiệu thuật ngữ, sau đây là tổng quan ngắn gọn về các Thành phần Kiến trúc và cách chúng hoạt động cùng nhau. Mỗi thành phần được giải thích rõ hơn khi bạn sử dụng trong bài thực hành này.

Sơ đồ bên dưới cho thấy một dạng cơ bản của kiến trúc được đề xuất cho các ứng dụng sử dụng `Architecture Components`. Kiến trúc bao gồm một bộ

điều khiển UI, một ViewModel phục vụ LiveData, một Repository và một cơ sở dữ liệu Room. Cơ sở dữ liệu Room được hỗ trợ bởi một cơ sở dữ liệu SQLite và có thể truy cập thông qua một đối tượng truy cập dữ liệu (DAO). Mỗi thành phần được mô tả ngắn gọn bên dưới và chi tiết trong chương khái niệm Architecture Components, 10.1: Lưu trữ dữ liệu với Room. Bạn triển khai các thành phần trong bài thực hành này.



Vì tất cả các thành phần đều tương tác với nhau nên bạn sẽ thấy các tham chiếu đến các thành phần này trong suốt bài thực hành này. Sau đây là giải thích ngắn gọn về từng thành phần.

**Thực thể:** Trong ngữ cảnh của Architecture Components, thực thể là một lớp được chú thích mô tả một bảng cơ sở dữ liệu.

**Cơ sở dữ liệu SQLite:** Trên thiết bị, dữ liệu được lưu trữ trong cơ sở dữ liệu SQLite. Thư viện lưu trữ Room tạo và duy trì cơ sở dữ liệu này cho bạn.

**DAO:** Viết tắt của **data access object**. Là một ánh xạ của các truy vấn SQL tới các hàm. Trước đây, bạn phải định nghĩa các truy vấn này trong một lớp trợ giúp. Khi bạn sử dụng DAO, mã của bạn gọi các hàm, và các thành phần sẽ xử lý phần còn lại.

**Room database:** Là lớp cơ sở dữ liệu nằm trên một cơ sở dữ liệu SQLite, xử lý các tác vụ tầm thường mà bạn trước đây phải xử lý bằng một lớp trợ giúp. Cơ sở dữ liệu Room sử dụng DAO để phát hành các truy vấn tới cơ sở dữ liệu SQLite dựa trên các hàm đã được gọi.

**Repository:** Là một lớp mà bạn tạo ra để quản lý nhiều nguồn dữ liệu. Ngoài một cơ sở dữ liệu Room, Repository có thể quản lý các nguồn dữ liệu từ xa như máy chủ web.

**ViewModel:** Cung cấp dữ liệu cho giao diện người dùng (UI) và hoạt động như một trung tâm giao tiếp giữa Repository và UI. Che giấu backend khỏi UI. Các thể hiện của ViewModel tồn tại qua các thay đổi cấu hình thiết bị.

**LiveData:** Là một lớp giữ dữ liệu theo mô hình **observer**, có nghĩa là nó có thể được quan sát. Luôn giữ/bộ nhớ phiên bản mới nhất của dữ liệu. Thông báo cho các quan sát viên của nó khi dữ liệu đã thay đổi. Thông thường, các thành phần UI quan sát dữ liệu liên quan. LiveData nhận biết vòng đời, vì vậy nó tự động quản lý việc dừng và tiếp tục quan sát dựa trên trạng thái của hoạt động hoặc fragment đang quan sát nó.

## Những điều bạn nên biết

Bạn phải có khả năng tạo và chạy ứng dụng trong Android Studio 3.0 trở lên. Đặc biệt, hãy làm quen với:

- RecyclerView và bộ điều hợp
- Cơ sở dữ liệu SQLite và ngôn ngữ truy vấn SQLite
- Luồng nói chung và AsyncTask nói riêng

Sẽ rất hữu ích nếu bạn quen thuộc với:

- Các mẫu kiến trúc phần mềm tách biệt dữ liệu khỏi UI.
- **Mô hình Observer:** Tóm lại, mô hình Observer định nghĩa một sự phụ thuộc một-nhiều giữa các đối tượng. Mỗi khi một đối tượng thay đổi trạng thái của nó, tất cả các phụ thuộc của đối tượng đó được thông báo và cập nhật tự động. Đối tượng chính được gọi là "chủ thể" và các phụ thuộc của nó được gọi là "quan sát viên". Thông thường, chủ thể thông báo cho các quan sát viên bằng cách gọi một trong các phương thức của quan sát viên. Chủ thể biết gọi phương thức nào, vì các quan sát viên đã được "đăng ký" với chủ thể và chỉ định các phương thức để gọi.

**Quan trọng:** Thực hành này triển khai kiến trúc được định nghĩa trong **Hướng dẫn về Kiến trúc Ứng dụng** và được giải thích trong chương **Các thành phần Kiến trúc**, mục 10.1: **Lưu trữ dữ liệu với Room**. Rất khuyến nghị bạn nên đọc chương về các khái niệm này.

## Những gì bạn sẽ học được

- Cách thiết kế và xây dựng một ứng dụng sử dụng một số thành phần kiến trúc Android: **Bạn sẽ sử dụng Room, ViewModel, và LiveData**.

## Bạn sẽ làm gì

- Tạo một ứng dụng với một **Activity** hiển thị các từ trong một **RecyclerView**.
- Tạo một **Entity** đại diện cho các đối tượng từ.
- Định nghĩa ánh xạ các truy vấn SQL tới các phương thức Java trong một **DAO** (data access object).
- Sử dụng **LiveData** để làm cho các thay đổi về dữ liệu có thể nhìn thấy trên UI thông qua các quan sát viên.
- Thêm một cơ sở dữ liệu **Room** vào ứng dụng để lưu trữ dữ liệu cục bộ và khởi tạo cơ sở dữ liệu.
- Trừu tượng hóa backend dữ liệu dưới dạng một lớp **Repository** với một API không phụ thuộc vào cách dữ liệu được lưu trữ hoặc thu thập.
- Sử dụng một **ViewModel** để tách biệt tất cả các thao tác dữ liệu khỏi UI.
- Thêm một **Activity** thứ hai cho phép người dùng thêm từ mới.

## Tổng quan về ứng dụng

Trong thực hành này, bạn sẽ xây dựng một ứng dụng sử dụng **Android Architecture Components**. Ứng dụng, có tên là **RoomWordsSample**, lưu trữ danh sách các từ trong một cơ sở dữ liệu Room và hiển thị danh sách đó trong một **RecyclerView**. Ứng dụng RoomWordsSample khá cơ bản, nhưng đủ hoàn chỉnh để bạn có thể sử dụng nó như một mẫu để phát triển thêm.

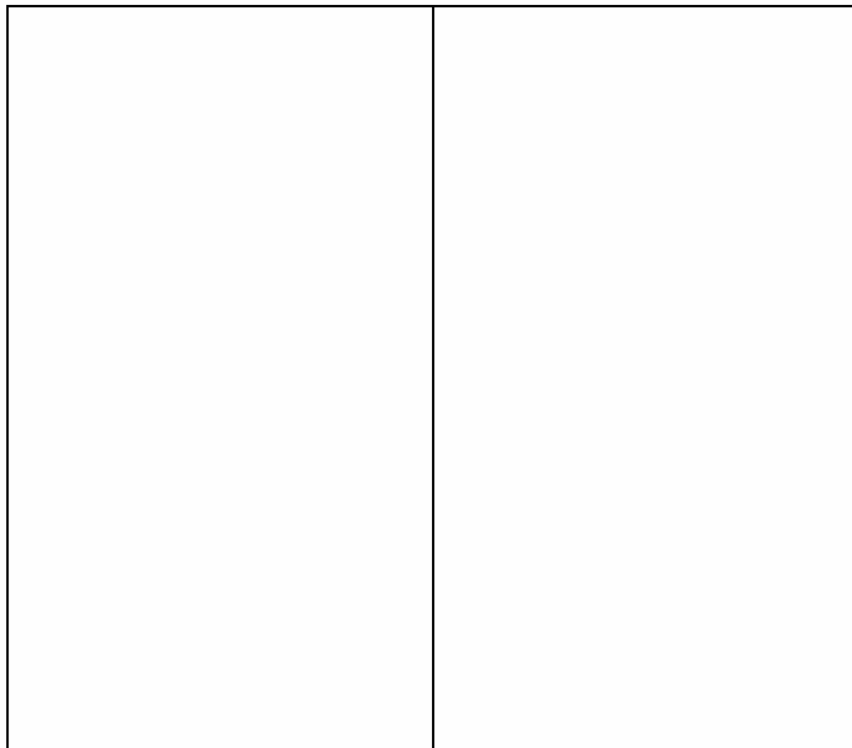
Ứng dụng RoomWordsSample thực hiện những chức năng sau:

- Làm việc với một cơ sở dữ liệu để lấy và lưu trữ các từ, đồng thời chuẩn bị sẵn cơ sở dữ liệu với một số từ.

- Hiển thị tất cả các từ trong một **RecyclerView** trong **MainActivity**.
- Mở một **Activity** thứ hai khi người dùng nhấn nút FAB (+). Khi người dùng nhập một từ, ứng dụng sẽ thêm từ đó vào cơ sở dữ liệu và sau đó danh sách sẽ tự động cập nhật.

Các ảnh chụp màn hình bên dưới cho thấy những điều sau:

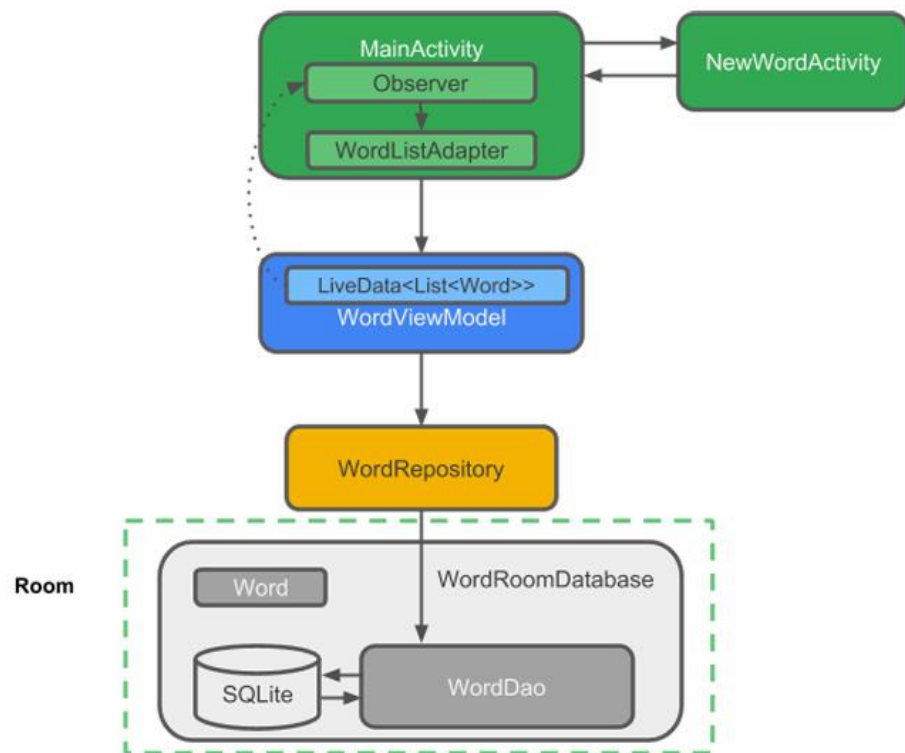
- Ứng dụng **RoomWordsSample** khi bắt đầu, với danh sách từ ban đầu.
- **Activity** để thêm một từ.



## Tổng quan về kiến trúc mẫu RoomWords

Sơ đồ sau đây phản ánh sơ đồ tổng quan từ phần giới thiệu và hiển thị tất cả các thành phần của ứng dụng **RoomWordsSample**. Mỗi hộp bao quanh (ngoại trừ cơ sở dữ liệu SQLite) đại diện cho một lớp mà bạn tạo ra.

Mẹo: In hoặc mở sơ đồ này trong một tab riêng để bạn có thể tham khảo khi xây dựng mã.



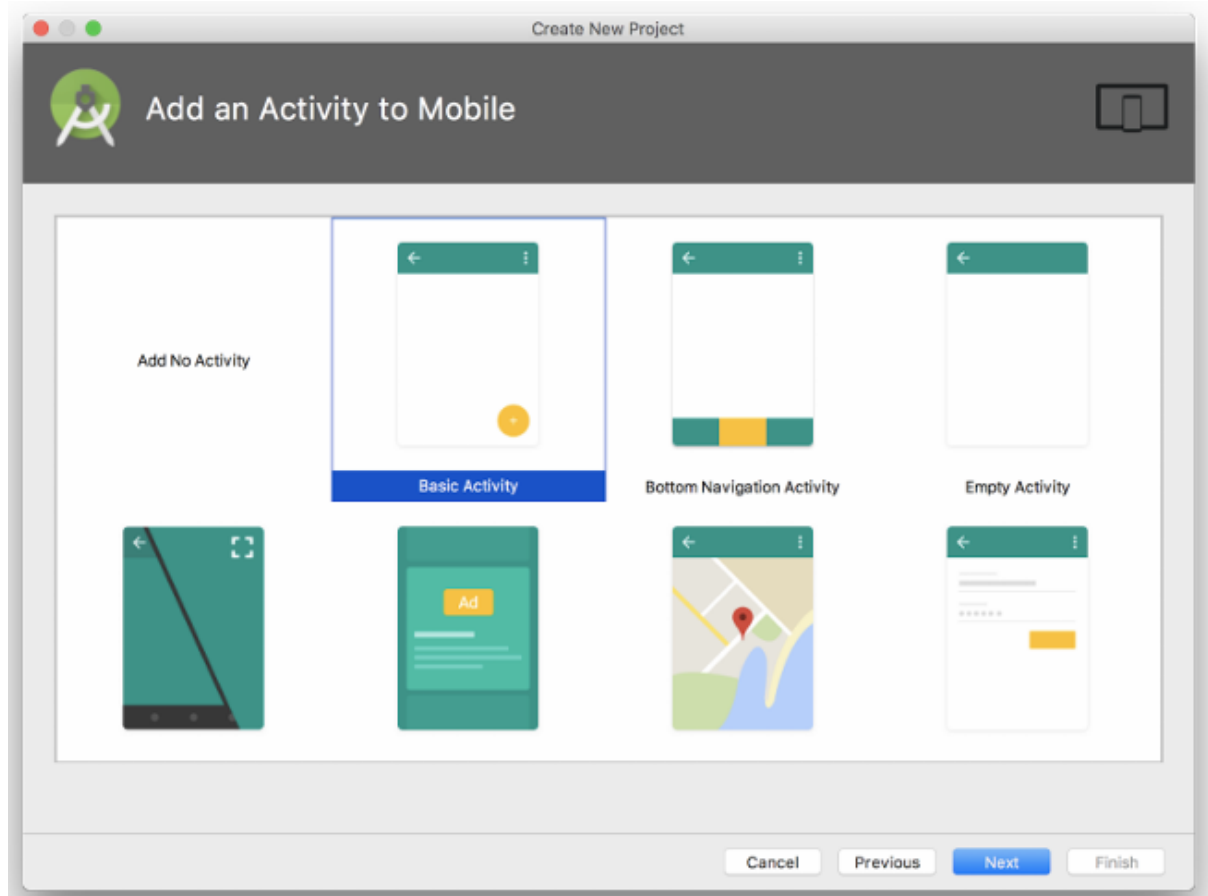
## TASK1: Tạo ứng dụng RoomWordsSample

**Lưu ý:** Trong thực hành này, bạn sẽ cần tạo các biến thành viên, nhập các lớp, và trích xuất giá trị khi cần thiết. Mặc dù bạn cần làm quen sẽ được cung cấp nhưng không được giải thích.

### 1.1 Tạo một ứng dụng với một Hoạt động

Để tạo ứng dụng **RoomWordsSample** trong Android Studio, hãy làm theo các bước sau:

1. Mở **Android Studio** và chọn **Create New Project**.
2. Trong các màn hình thiết lập:
  - Đặt tên ứng dụng là **RoomWordsSample**.
  - Nếu bạn thấy các hộp kiểm cho **Include Kotlin support** và **Include C++ support**, hãy bỏ chọn cả hai hộp.
  - Chọn chỉ **Phone & Tablet** làm định dạng và đặt **Minimum SDK** thành **API 14** hoặc cao hơn.
  - Chọn **Basic Activity**



## 1.2 Cập nhật các tệp Gradle

Trong Android Studio, hãy thêm thủ công các thư viện Architecture Component vào các tệp Gradle của bạn.

1. Thêm mã sau vào tệp build.gradle (Module: app), ở cuối khối dependencies (nhưng vẫn nằm trong đó).

```
// Room components
implementation
"android.arch.persistence.room:runtime:$rootProject.roomVersion"
annotationProcessor
"android.arch.persistence.room:compiler:$rootProject.roomVersion"
androidTestImplementation
"android.arch.persistence.room:testing:$rootProject.roomVersion"

// Lifecycle components
implementation
"android.arch.lifecycle:extensions:$rootProject.archLifecycleVersion"
annotationProcessor
"android.arch.lifecycle:compiler:$rootProject.archLifecycleVersion"
```

2. Trong tệp build.gradle (Dự án: RoomWordsSample), hãy thêm số phiên bản vào cuối tệp.

```
ext {
    roomVersion = '1.0.0'
    archLifecycleVersion = '1.1.0'
}
```

Quan trọng: Sử dụng các phiên bản mới nhất cho các thư viện Room và lifecycle. Để tìm các số phiên bản mới nhất:

Trên trang Thêm Thành phần vào Dự án của bạn, tìm mục cho thành phần, chẳng hạn như Room.

Số phiên bản được định nghĩa ở đầu định nghĩa phụ thuộc của thành phần.

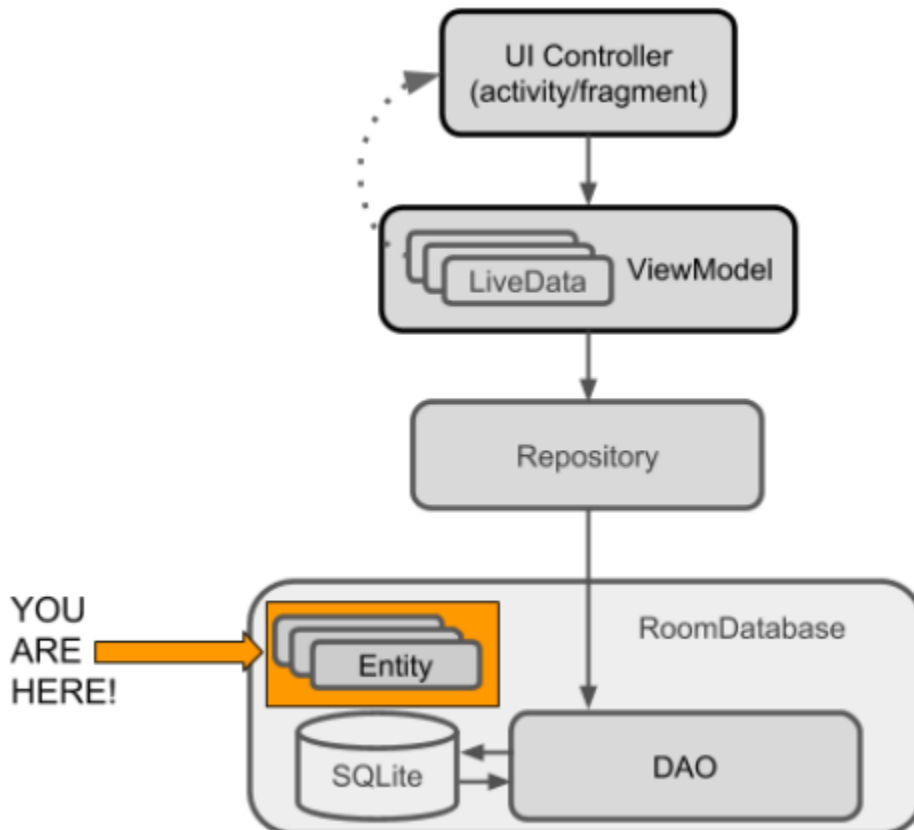
Ví dụ, số phiên bản Room trong định nghĩa dưới đây là 1.1.1:

```
def room_version = "1.1.1"
```

## Task 2: Tạo thực thể Word



Sơ đồ dưới đây là sơ đồ kiến trúc hoàn chỉnh với thành phần mà bạn sẽ triển khai trong nhiệm vụ này được làm nổi bật. Mỗi nhiệm vụ sẽ có một sơ đồ như vậy để giúp bạn hiểu thành phần hiện tại phù hợp với cấu trúc tổng thể của ứng dụng như thế nào, và để thấy cách các thành phần được kết nối với nhau.



Dữ liệu cho ứng dụng này là các từ, và mỗi từ được đại diện bởi một thực thể trong cơ sở dữ liệu. Trong nhiệm vụ này, bạn sẽ tạo lớp **Word** và chú thích nó để Room có thể tạo bảng cơ sở dữ liệu từ nó. Sơ đồ dưới đây cho thấy bảng cơ sở dữ liệu **word\_table**. Bảng có một cột từ, cũng đóng vai trò là khóa chính, và hai hàng, mỗi hàng cho "Hello" và "World."

word_table table
word (primary key, string)
"Hello"
"World"

## 2.1 Tạo lớp Word

1. Tạo một lớp có tên là `Word`.
2. Thêm một hàm khởi tạo nhận một chuỗi từ làm tham số. Thêm chú thích `@NonNull` để đảm bảo tham số này không bao giờ có giá trị null.
3. Thêm một phương thức "getter" có tên là `getWord()` để trả về từ. Room yêu cầu các phương thức "getter" trong các lớp thực thể để có thể khởi tạo các đối tượng của bạn.

```
public class Word {  
  
    private String mWord;  
  
    public Word(@NonNull String word) { this.mWord = word; }  
  
    public String getWord() { return this.mWord; }  
}
```

## 2.2 Chú thích lớp Word

Để làm cho lớp `Word` có ý nghĩa đối với cơ sở dữ liệu Room, bạn cần chú thích nó. Các chú thích xác định cách mỗi phần của lớp `Word` liên quan đến một mục trong cơ sở dữ liệu. Room sử dụng thông tin này để tạo mã.

Bạn sẽ sử dụng các chú thích sau trong các bước dưới đây:

- `@Entity(tableName = "word_table")`: Mỗi lớp `@Entity` đại diện cho một thực thể trong bảng. Chú thích phân khai báo lớp của bạn để chỉ ra rằng lớp này là một thực thể. Chỉ định tên bảng nếu bạn muốn nó khác với tên lớp.
- `@PrimaryKey`: Mỗi thực thể cần một khóa chính. Để giữ cho mọi thứ đơn giản, mỗi từ trong ứng dụng RoomWordsSample sẽ hoạt động như một khóa chính của chính nó.
- `@NonNull`: Chỉ ra rằng tham số, trường hoặc giá trị trả về của phương thức không bao giờ có thể là null. Khóa chính luôn nên sử dụng chú thích này. Sử dụng chú thích này cho bất kỳ trường bắt buộc nào trong các hàng của bạn.
- `@ColumnInfo(name = "word")`: Chỉ định tên của một cột trong bảng, nếu bạn muốn tên cột khác với tên biến thành viên.
- Mỗi trường được lưu trữ trong cơ sở dữ liệu phải là công khai hoặc có một phương thức "getter". Ứng dụng này cung cấp phương thức "getter" `getWord()` thay vì công khai các biến thành viên trực tiếp.

Để biết danh sách chú thích đầy đủ, hãy xem phần tham khảo tóm tắt gói Room.

Cập nhật lớp `Word` của bạn bằng các chú thích, như được hiển thị trong mã bên dưới

1. Thêm ký hiệu `@Entity` vào khai báo lớp và đặt `tableName` thành

`"word_table"`.

2. Chú thích biến thành viên `mWord` là `@PrimaryKey`. Yêu cầu `mWord` là `@NonNull`, và đặt tên cho cột là `"word"`.

Lưu ý: Nếu bạn nhập chú thích, Android Studio sẽ tự động nhập mọi thứ bạn cần.

Sau đây là mã đầy đủ:

```

@Entity(tableName = "word_table")
public class Word {

    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "word")
    private String mWord;

    public Word(@NonNull String word) { this.mWord = word; }

    public String getWord() { return this.mWord; }
}

```

Nếu bạn gặp lỗi cho chú thích, bạn có thể nhập chúng theo cách thủ công như sau:

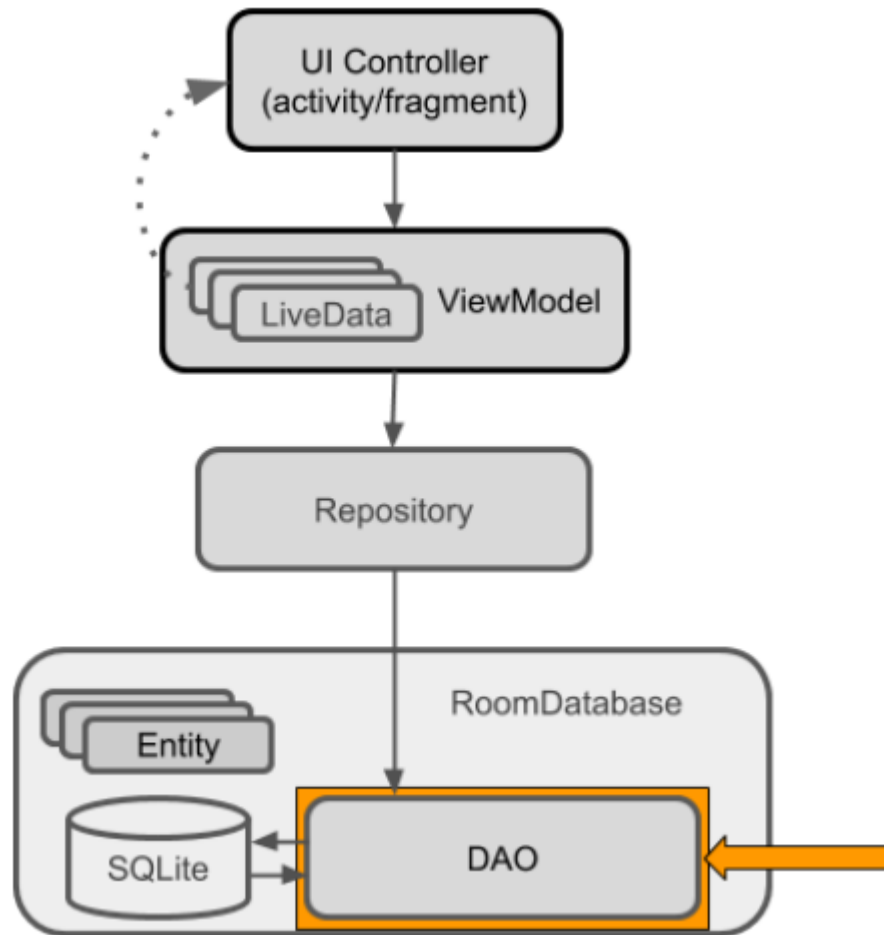
```

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.PrimaryKey;
import android.support.annotation.NonNull;

```

Mẹo về việc tự động tạo khóa: Để tự động tạo khóa duy nhất cho mỗi thực thể, bạn sẽ thêm và chú thích khóa số nguyên chính bằng `autoGenerate=true`. Xem mục Định nghĩa dữ liệu bằng thực thể Room.

### Task 3: Tạo DAO



Dao (Data Access Object) là một lớp được chú thích để chỉ định các truy vấn SQL và liên kết chúng với các phương thức gọi. Trình biên dịch kiểm tra các truy vấn SQL để phát hiện lỗi và tạo chúng từ các chú thích.

*Lưu ý:*

- DAO phải là một interface hoặc lớp trừu tượng.
- Room sử dụng DAO để tạo ra một API sạch cho mã của bạn.
- Tất cả các truy vấn (@Query) phải được thực hiện trên một luồng khác ngoài luồng chính. Đối với các thao tác như chèn hoặc xóa, Room sẽ quản lý luồng cho bạn nếu bạn sử dụng các chú thích tiện lợi.

### 3.1 Triển khai lớp DAO

DAO cho bài thực hành này rất cơ bản và chỉ cung cấp các truy vấn để lấy tất cả các từ, chèn từ và xóa tất cả các từ.

1. Tạo một interface mới và đặt tên là `WordDao`.
2. Chú thích phần khai báo lớp với `@Dao` để xác định đây là một DAO cho Room.
3. Khai báo một phương thức để chèn một từ:

```
void insert(Word word);
```

4. Chú thích phương thức `insert()` bằng `@Insert`. Bạn không cần phải cung cấp bất kỳ SQL nào! (Ngoài ra còn có các chú thích `@Delete` và `@Update` để xóa và cập nhật một hàng, nhưng bạn không sử dụng các thao tác này trong phiên bản đầu tiên của ứng dụng này.)
5. Khai báo một phương thức để xóa tất cả các từ:

```
void deleteAll();
```

6. Không có chú thích tiện lợi nào để xóa nhiều thực thể, vì vậy hãy chú thích phương thức `deleteAll()` bằng `@Query` chung. Cung cấp truy vấn SQL dưới dạng tham số chuỗi cho `@Query`. Chú thích phương thức `deleteAll()` như sau:

```
@Query("DELETE FROM word_table")
```

7. Tạo phương thức có tên là `getAllWords()` trả về Danh sách các từ:

```
List<Word> getAllWords();
```

8. Chú thích phương thức `getAllWords()` bằng truy vấn SQL để lấy tất cả các từ từ `word_table`, được sắp xếp theo thứ tự bảng chữ cái để thuận tiện:

```
@Query("SELECT * FROM word_table ORDER BY word ASC")
```

Sau đây là mã hoàn chỉnh cho lớp `WordDao`:

```
@Dao
public interface WordDao {

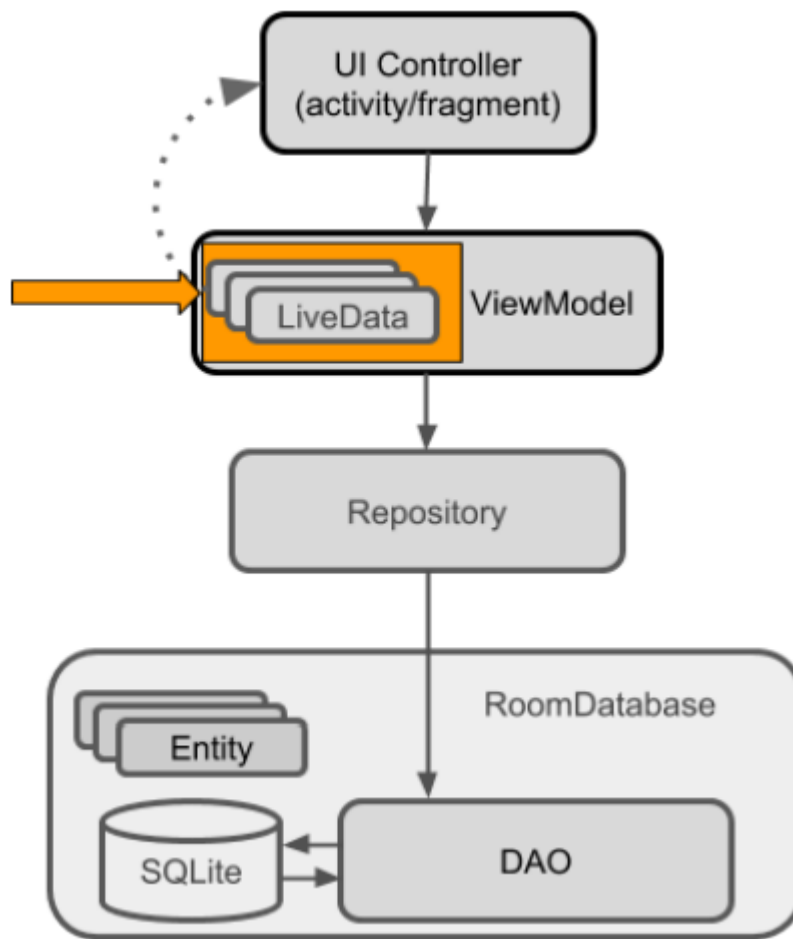
    @Insert
    void insert(Word word);

    @Query("DELETE FROM word_table")
    void deleteAll();

    @Query("SELECT * FROM word_table ORDER BY word ASC")
    List<Word> getAllWords();
}
```

Mẹo: Đối với ứng dụng này, việc sắp xếp các từ không thực sự cần thiết. Tuy nhiên, theo mặc định, thứ tự trả về không được đảm bảo và việc sắp xếp giúp việc kiểm tra trở nên dễ dàng.

#### **Task 4: Use LiveData**



Khi bạn hiển thị dữ liệu hoặc sử dụng dữ liệu theo những cách khác, bạn thường muốn thực hiện một số hành động khi dữ liệu thay đổi. Điều này có nghĩa là bạn phải quan sát dữ liệu để khi dữ liệu thay đổi, bạn có thể phản ứng.

LiveData, là một lớp thư viện vòng đời để quan sát dữ liệu, có thể giúp ứng dụng của bạn phản hồi với các thay đổi dữ liệu. Nếu bạn sử dụng giá trị trả về thuộc loại LiveData trong mô tả phương thức của mình, Room sẽ tạo tất cả mã cần thiết để cập nhật LiveData khi cơ sở dữ liệu được cập nhật.

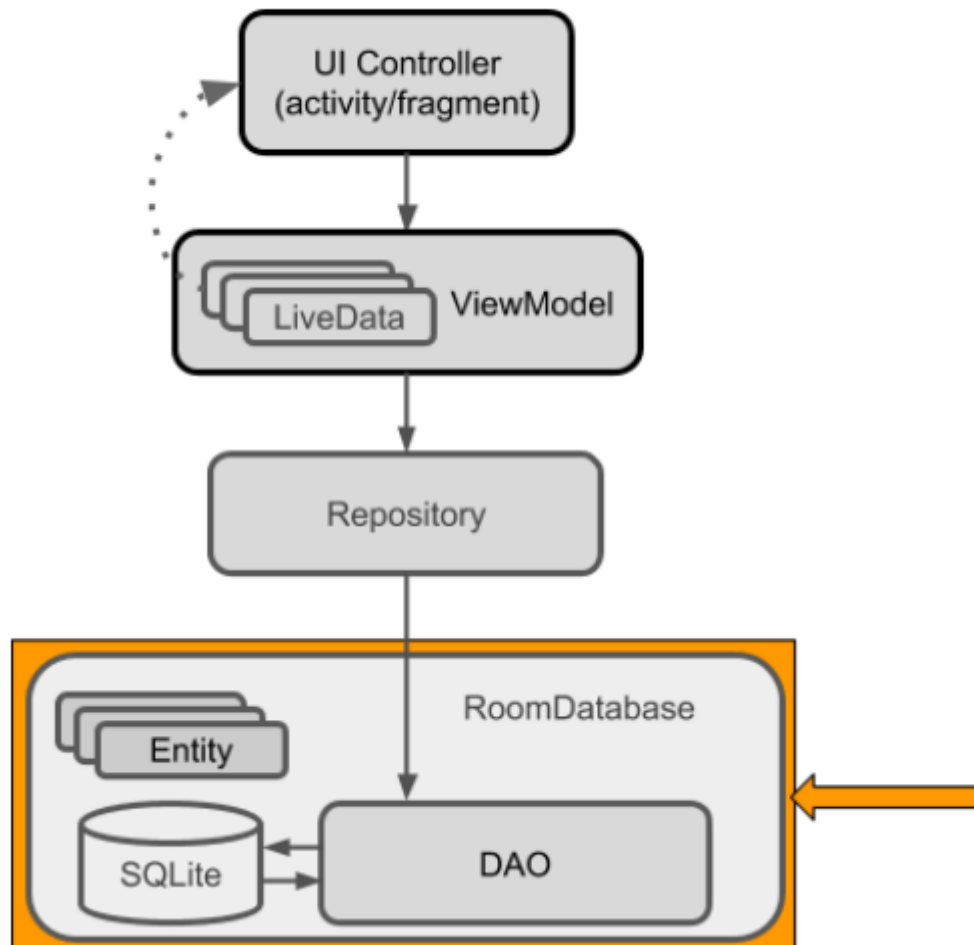
#### 4.1 Trả về LiveData trong WordDao

- Trong giao diện WordDao, hãy thay đổi chữ ký phương thức getAllWords() để List<Word> được trả về được bao bọc bằng LiveData<>



```
@Query("SELECT * FROM word_table ORDER BY word ASC")  
LiveData<List<Word>> getAllWords();
```

### Task 5: Thêm cơ sở dữ liệu Phòng



Room là một lớp cơ sở dữ liệu nằm trên SQLite. Room xử lý các tác vụ nhàm chán mà trước đây bạn phải thực hiện bằng một lớp trợ giúp cơ sở dữ liệu như SQLiteOpenHelper.

- Room sử dụng DAO để thực hiện các truy vấn đến cơ sở dữ liệu.

- Theo mặc định, để tránh hiệu suất UI kém, Room không cho phép bạn thực hiện các truy vấn cơ sở dữ liệu trên luồng chính. LiveData áp dụng quy tắc này bằng cách tự động chạy truy vấn bất đồng bộ trên một luồng nền khi cần thiết.
- Room cung cấp khả năng kiểm tra các câu lệnh SQLite ngay tại thời điểm biên dịch.
- Lớp Room của bạn phải là lớp trừu tượng và kế thừa từ `RoomDatabase`.
- Thông thường, bạn chỉ cần một instance của cơ sở dữ liệu Room cho toàn bộ ứng dụng.

## 5.1 Triển khai cơ sở dữ liệu Phòng

1. Tạo một lớp trừu tượng công khai mở rộng `RoomDatabase` và gọi nó là `WordRoomDatabase`.

```
public abstract class WordRoomDatabase extends RoomDatabase {
```

2. Chú thích lớp để trở thành một cơ sở dữ liệu Room. Khai báo các entity thuộc về cơ sở dữ liệu—trong trường hợp này, chỉ có một entity là `Word`. (Liệt kê các lớp entity sẽ tạo ra các bảng tương ứng trong cơ sở dữ liệu.) Đặt số phiên bản của cơ sở dữ liệu.

```
@Database(entities = {Word.class}, version = 1)
```

3. Xác định các DAO hoạt động với cơ sở dữ liệu. Cung cấp phương thức "getter" trừu tượng cho mỗi @Dao.

```
public abstract WordDao wordDao();
```

4. Tạo `WordRoomDatabase` dưới dạng singleton để ngăn chặn việc mở nhiều instance của cơ sở dữ liệu cùng một lúc, điều này có thể gây ra vấn đề.

```
private static WordRoomDatabase INSTANCE;

public static WordRoomDatabase getDatabase(final Context context) {
    if (INSTANCE == null) {
        synchronized (WordRoomDatabase.class) {
            if (INSTANCE == null) {
                // Create database here
            }
        }
    }
    return INSTANCE;
}
```

### 1.3) Room, LiveData và ViewModel