**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**



**GIÁO TRÌNH**

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

Hà Nội, 2.2025

MỤC LỤC

[CHƯƠNG 1. Làm quen 4](#_Toc190855147)

[Bài 1) Tạo ứng dụng đầu tiên 4](#_Toc190855148)

[1.1) Android Studio và Hello World 4](#_Toc190855149)

[1.2) Giao diện người dùng tương tác đầu tiên 5](#_Toc190855150)

[1.3) Trình chỉnh sửa bố cục 5](#_Toc190855151)

[1.4) Văn bản và các chế độ cuộn 5](#_Toc190855152)

[1.5) Tài nguyên có sẵn 5](#_Toc190855153)

[Bài 2) Activities 5](#_Toc190855154)

[2.1) Activity và Intent 5](#_Toc190855155)

[2.2) Vòng đời của Activity và trạng thái 5](#_Toc190855156)

[2.3) Intent ngầm định 5](#_Toc190855157)

[Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ 5](#_Toc190855158)

[3.1) Trình gỡ lỗi 5](#_Toc190855159)

[3.2) Kiểm thử đơn vị 5](#_Toc190855160)

[3.3) Thư viện hỗ trợ 5](#_Toc190855161)

[CHƯƠNG 2. Trải nghiệm người dùng 6](#_Toc190855162)

[Bài 1) Tương tác người dùng 6](#_Toc190855163)

[1.1) Hình ảnh có thể chọn 6](#_Toc190855164)

[1.2) Các điều khiển nhập liệu 6](#_Toc190855165)

[1.3) Menu và bộ chọn 6](#_Toc190855166)

[1.4) Điều hướng người dùng 6](#_Toc190855167)

[1.5) RecycleView 6](#_Toc190855168)

[Bài 2) Trải nghiệm người dùng thú vị 6](#_Toc190855169)

[2.1) Hình vẽ, định kiểu và chủ đề 6](#_Toc190855170)

[2.2) Thẻ và màu sắc 6](#_Toc190855171)

[2.3) Bố cục thích ứng 6](#_Toc190855172)

[Bài 3) Kiểm thử giao diện người dùng 6](#_Toc190855173)

[3.1) Espresso cho việc kiểm tra UI 6](#_Toc190855174)

[CHƯƠNG 3. Làm việc trong nền 6](#_Toc190855175)

[Bài 1) Các tác vụ nền 6](#_Toc190855176)

[1.1) AsyncTask 6](#_Toc190855177)

[1.2) AsyncTask và AsyncTaskLoader 6](#_Toc190855178)

[1.3) Broadcast receivers 6](#_Toc190855179)

[Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền 6](#_Toc190855180)

[2.1) Thông báo 6](#_Toc190855181)

[2.2) Trình quản lý cảnh báo 6](#_Toc190855182)

[2.3) JobScheduler 6](#_Toc190855183)

[CHƯƠNG 4. Lưu dữ liệu người dùng 7](#_Toc190855184)

[Bài 1) Tùy chọn và cài đặt 7](#_Toc190855185)

[1.1) Shared preferences 7](#_Toc190855186)

[1.2) Cài đặt ứng dụng 7](#_Toc190855187)

[Bài 2) Lưu trữ dữ liệu với Room 7](#_Toc190855188)

[2.1) Room, LiveData và ViewModel 7](#_Toc190855189)

[2.2) Room, LiveData và ViewModel 7](#_Toc190855190)

3.1) Trinfh gowx loi ……………………………………………………………………...

# LÀM QUEN

## Tạo ứng dụng đầu tiên

### 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.

A screenshot of a computer

AI-generated content may be incorrect.

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**

### Giao diện người dùng tương tác đầu tiên

### Trình chỉnh sửa bố cục

### Văn bản và các chế độ cuộn

### Tài nguyên có sẵn

## Activities

### Activity và Intent

### Vòng đời của Activity và trạng thái

### Intent ngầm định

## Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ

### Trình gỡ lỗi

### Kiểm thử đơn vị

### Thư viện hỗ trợ

# TRẢI NGHIỆM NGƯỜI DÙNG

## Tương tác người dùng

### Hình ảnh có thể chọn

### Các điều khiển nhập liệu

### Menu và bộ chọn

### Điều hướng người dùng

### RecycleView

## Trải nghiệm người dùng thú vị

### Hình vẽ, định kiểu và chủ đề

### Thẻ và màu sắc

### Bố cục thích ứng

## Kiểm thử giao diện người dùng

### Espresso cho việc kiểm tra UI

# LÀM VIỆC TRONG NỀN

## Các tác vụ nền

### AsyncTask

### AsyncTask và AsyncTaskLoader

### Broadcast receivers

## Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền

### Thông báo

### Trình quản lý cảnh báo

### JobScheduler

# LƯU DỮ LIỆU NGƯỜI DÙNG

## Tùy chọn và cài đặt

### Shared preferences

### Cài đặt ứng dụng

## Lưu trữ dữ liệu với Room

### Room, LiveData và ViewModel

### Room, LiveData và ViewModel

**Bài 9.1: Share Preferences**

**Giới thiệu**

Shared preferences cho phép bạn lưu trữ một lượng nhỏ dữ liệu nguyên thủy làm cặp khóa/giá trị trong một tệp trên thiết bị. Để có được xử lý tệp ưu tiên và để đọc, ghi và quản lý dữ liệu ưu tiên, sử dụng lớp SharedPreferences . Android framework quản lý chính tệp tùy chọn được chia sẻ. Tệp có thể truy cập được cho tất cả các thành phần của ứng dụng của bạn, nhưng không thể truy cập được cho các ứng dụng khác.

Dữ liệu bạn lưu vào các tùy chọn được chia sẻ khác với dữ liệu trong trạng thái hoạt động đã lưu, mà bạn đã học về một chương trước:

* Dữ liệu trong trạng thái thể hiện hoạt động đã lưu được giữ lại trong các trường hợp hoạt động trong cùng một người dùng phiên họp.
* Shared preferences vẫn tồn tại trong các phiên người dùng. Shared preferences vẫn tồn tại ngay cả khi ứng dụng của bạn dừng và khởi động lại, hoặc nếu thiết bị khởi động lại.

Chỉ sử dụng các Shared preferences khi bạn cần lưu một dữ liệu số lượng nhỏ dưới dạng các cặp khóa/giá trị đơn giản. Để quản lý số lượng lớn hơn của dữ liệu ứng dụng liên tục, hãy sử dụng phương pháp lưu trữ như thư viện phòng hoặc cơ sở dữ liệu SQL.

Những điều bạn nên biết

Bạn nên quen thuộc với:

* Tạo, xây dựng và chạy ứng dụng trong Android Studio.
* Thiết kế bố cục với các nút và chế độ xem văn bản.
* Sử dụng phong cách và chủ đề.
* Lưu và khôi phục trạng thái thể hiện hoạt động.

Những gì bạn sẽ học

Bạn sẽ học cách:

* Xác định sở thích chia sẻ là gì.
* Tạo một tệp tùy chọn được chia sẻ cho ứng dụng của bạn.
* Xóa dữ liệu trong các tùy chọn được chia sẻ.
* Cập nhật một ứng dụng để nó có thể lưu, truy xuất và đặt lại các tùy chọn được chia sẻ.

Tổng quan về ứng dụng

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

Bạn bắt đầu với ứng dụng khởi động và thêm các tùy chọn được chia sẻ vào mã hoạt động chính. Bạn cũng thêm một nút đặt lại đặt cả số đếm và màu nền cho mặc định và xóa tệp tùy chọn.

**Nhiệm vụ 1: Khám phá Hellosharedprefs**

Dự án ứng dụng khởi động hoàn chỉnh cho thực tế này có sẵn tại Hellosharedprefs Starter. Trong này

Nhiệm vụ bạn tải dự án vào Android Studio và khám phá một số tính năng chính của ứng dụng.

* 1. **Mở và chạy dự án HelloSharedPrefs**

Tải ứng dụng HelloSharedPrefs-Starter và giải nén tệp.

Mở dự án trong Android Studio, xây dựng và chạy ứng dụng.

* Nhấp vào **Count** đếm để tăng số trong chế độ xem văn bản chính.
* Nhấp 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 bảo quản.
* Nhấp vào nút **Reset** để đặt màu và đếm lại mặc định.

Buộc đóng ứng dụng bằng một trong những cách sau.

* Trong Android Studio, chọn **Run > Stop 'app'** hoặc nhấp vào **Biểu tượng Dừng** trên thanh công cụ.
* Trên thiết bị, nhấn nút **Recents** (nút hình vuông ở góc dưới bên phải). Vuốt thẻ ứng dụng **HelloSharedPrefs** để thoát ứng dụng, hoặc nhấn vào dấu **X** ở góc phải của thẻ. Nếu bạn thoát ứng dụng theo cách này, hãy đợi vài giây trước khi khởi động lại để hệ thống có thời gian dọn dẹp.

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

* 1. **Khám phá code của Activity.**

1. Mở MainActivity
2. Kiểm tra code và lưu ý những điều này:

* Số đếm (**mCount**) được định nghĩa là một số nguyên. Phương thức **countUp()** trong sự kiện **onClick** tăng giá trị này lên và cập nhật **TextView** chính.
* Màu sắc (**mColor**) cũng là một số nguyên, ban đầu được định nghĩa là màu xám trong tệp tài nguyên **colors.xml** với tên **default\_background**.
* Phương thức **changeBackground()** trong sự kiện **onClick** lấy màu nền của nút được nhấn, sau đó đặt màu đó cho **TextView** chính.
* Cả hai số nguyên **mCount** và **mColor** đều được lưu vào gói trạng thái phiên bản trong phương thức **onSaveInstanceState()**, và được khôi phục trong **onCreate()**. Các khóa của gói cho số đếm và màu sắc được định nghĩa bởi các biến riêng (**COUNT\_KEY**) và (**COLOR\_KEY**).

**Nhiệm vụ 2: Lưu và khôi phục dữ liệu vào tệp tùy chọn chia sẻ.**

Trong nhiệm vụ này, bạn lưu trạng thái của ứng dụng vào một tệp tùy chọn chia sẻ và đọc lại dữ liệu đó khi ứng dụng được khởi động lại. Vì dữ liệu trạng thái mà bạn lưu vào tùy chọn chia sẻ (số đếm hiện tại và màu sắc) là cùng dữ liệu mà bạn bảo toàn trong trạng thái phiên bản, bạn không cần phải thực hiện việc này hai lần. Bạn có thể thay thế hoàn toàn trạng thái phiên bản bằng trạng thái từ shared preference.

* 1. **Khởi tạo preferences**

1. Để thêm các biến thành viên (member variables) vào lớp MainActivity nhằm lưu tên của tệp Shared Preferences và một tham chiếu đến đối tượng SharedPreferences,

A screenshot of a computer

AI-generated content may be incorrect.

Bạn có thể đặt tên tệp Shared Preferences của mình bất kỳ như thế nào bạn muốn, nhưng theo thông lệ, nó thường có cùng tên với tên gói của ứng dụng của bạn.

1. Trong phương thức onCreate(), hãy khởi tạo Shared Preferences. Chèn đoạn mã này trước câu lệnh if:



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

Ghi chú: Các phiên bản cũ của Android có các chế độ khác cho phép bạn tạo tệp Shared Preferences có thể đọc được hoặc ghi được bởi toàn bộ hệ thống. Những chế độ này đã bị deprecated (không khuyến khích sử dụng) từ API 17 và hiện nay bị phản đối mạnh mẽ vì lý do bảo mật. Nếu bạn cần chia sẻ dữ liệu với các ứng dụng khác, hãy cân nhắc sử dụng content URI được cung cấp bởi FileProvider.

Code giải pháp cho lớp MainActivity, một phần:

A screen shot of a computer program

AI-generated content may be incorrect.

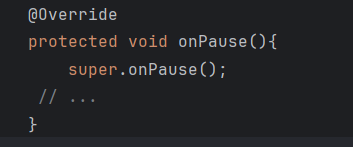
A computer screen shot of a program code

AI-generated content may be incorrect.

* 1. **Lưu** **Lưu các tùy chọn trong onPause()**

Việc lưu các tùy chọn rất giống với việc lưu trạng thái phiên bản -- cả hai thao tác đều lưu dữ liệu vào một đối tượng Bundle dưới dạng cặp khóa/giá trị. Tuy nhiên, đối với Shared Preferences, bạn lưu dữ liệu đó trong hàm gọi lại vòng đời onPause(), và bạn cần một đối tượng biên tập chia sẻ (SharedPreferences.Editor) để ghi vào đối tượng Shared Preferences.

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



1. Trong onPause(), lấy một đối tượng biên tập (Editor) cho đối tượng SharedPreferences:



Một trình biên tập Shared Preferences là cần thiết để ghi vào đối tượng Shared Preferences. Thêm dòng này vào `onPause()` sau lời gọi đến `super.onPause()`.

1. Sử dụng phương thức `putInt()` để lưu cả hai số nguyên `mCount` và `mColor` vào Shared Preferences với các khóa phù hợp:

A screenshot of a computer screen

AI-generated content may be incorrect.

1. Gọi apply() để lưu các tùy chọn:



Phương thức `apply()` lưu các tùy chọn một cách bất đồng bộ, ngoài luồng giao diện người dùng (UI thread). Trình biên tập Shared Preferences cũng có phương thức `commit()` để lưu các tùy chọn một cách đồng bộ. Phương thức `commit()` không được khuyến khích vì nó có thể chặn các hoạt động khác.

1. Xóa toàn bộ phương thức `onSaveInstanceState()`. Vì trạng thái phiên bản của hoạt động chứa cùng dữ liệu với Shared Preferences, bạn có thể thay thế hoàn toàn trạng thái phiên bản.

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

A screen shot of a computer program

AI-generated content may be incorrect.

* 1. **Khôi phục tùy chỉnh trong onCreate()**

Cũng giống như trạng thái phiên làm việc, ứng dụng của bạn sẽ đọc bất kỳ tùy chỉnh đã lưu nào trong phương thức onCreate(). Một lần nữa, vì các tùy chỉnh dùng chung chứa cùng dữ liệu với trạng thái phiên làm việc, chúng ta có thể thay thế trạng thái bằng các tùy chỉnh ở đây. Mỗi khi onCreate() được gọi – khi ứng dụng khởi động hoặc khi có thay đổi cấu hình – các tùy chỉnh dùng chung sẽ được sử dụng để khôi phục trạng thái của giao diện.

1. Xác định phần của phương thức onCreate() kiểm tra xem đối số savedInstanceState có bằng null hay không và khôi phục trạng thái phiên làm việc.

A computer screen shot of a code

AI-generated content may be incorrect.

1. Xóa toàn bộ khối
2. Trong phương thức onCreate(), tại cùng vị trí nơi mã trạng thái phiên (instance state) đã được sử dụng, hãy lấy giá trị đếm từ SharedPreferences bằng khóa COUNT\_KEY và gán nó cho biến mCount.



Khi bạn đọc dữ liệu từ SharedPreferences, bạn không cần lấy một SharedPreferences.Editor. Hãy sử dụng bất kỳ phương thức "get" nào trên đối tượng SharedPreferences (chẳng hạn như getInt() hoặc getString()) để truy xuất dữ liệu từ SharedPreferences.

Lưu ý rằng phương thức getInt() nhận hai đối số: một là khóa để truy xuất giá trị, và một là giá trị mặc định nếu không tìm thấy khóa đó. Trong trường hợp này, giá trị mặc định là 0, giống với giá trị khởi tạo của mCount.

1. Cập nhật giá trị của TextView chính với số đếm mới.



1. Lấy màu từ SharedPreferences bằng khóa COLOR\_KEY và gán nó cho biến mColor.



Như trước đây, đối số thứ hai của getInt() là giá trị mặc định được sử dụng trong trường hợp khóa không tồn tại trong SharedPreferences. Trong trường hợp này, bạn có thể chỉ cần sử dụng lại giá trị của mColor, vì nó vừa được khởi tạo với màu nền mặc định ở phần trên của phương thức.

1. Cập nhật màu nền của TextView chính.



1. Chạy ứng dụng. Nhấn nút **Count** và thay đổi màu nền để cập nhật trạng thái phiên (instance state) và SharedPreferences.
2. Xoay thiết bị hoặc trình giả lập để kiểm tra rằng số đếm và màu sắc được lưu lại sau khi thay đổi cấu hình.
3. Buộc đóng 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 **HelloSharedPrefs** để đóng ứng dụng hoặc nhấn **X** ở góc phải của thẻ.
4. Chạy lại ứng dụng. Ứng dụng sẽ khởi động lại và tải SharedPreferences, duy trì trạng thái.

Code giải pháp cho phương thức onCreate() trong MainActivity:

**A computer screen shot of a program code

AI-generated content may be incorrect.**

* 1. **Đặt lại SharedPreferences trong trình xử lý sự kiện nhấp reset().**

Nút đặt lại trong ứng dụng ban đầu sẽ đặt lại cả số đếm và màu sắc của hoạt động về giá trị mặc định. Vì SharedPreferences lưu trạng thái của hoạt động, nên cũng cần xóa SharedPreferences cùng lúc.

1. Trong phương thức reset() của sự kiện onClick, sau khi màu sắc và số đếm được đặt lại, hãy lấy một Editor cho đối tượng SharedPreferences: 
2. Xóa tất cả dữ liệu trong SharedPreferences. 
3. Áp dụng các thay đổi.

Code giải pháp cho phương thức reset():A screen shot of a computer program

AI-generated content may be incorrect.

**CODE GIẢI PHÁP**  
Dự án Android Studio: **HelloSharedPrefs**

**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:** Hãy chỉnh sửa ứng dụng **HelloSharedPrefs** sao cho thay vì tự động lưu trạng thái vào tệp SharedPreferences, bạn sẽ thêm một **Activity** thứ hai để thay đổi, đặt lại và lưu các tùy chọn. Thêm một nút có tên **Settings** để mở activity này. Trong activity mới, sử dụng các **ToggleButton** và **Spinner** để chỉnh sửa tùy chọn. Thêm hai nút **Save** và **Reset** để lưu hoặc xóa SharedPreferences.

**Tóm tắt**

* Lớp **SharedPreferences** cho phép ứng dụng lưu trữ một lượng nhỏ dữ liệu kiểu nguyên thủy dưới dạng cặp **key-value**.
* **SharedPreferences** được duy trì qua các phiên làm việc khác nhau của cùng một ứng dụng.
* Để ghi dữ liệu vào **SharedPreferences**, cần lấy một đối tượng **SharedPreferences.Editor**.
* Sử dụng các phương thức **"put"** trong **SharedPreferences.Editor**, chẳng hạn như putInt() hoặc putString(), để lưu dữ liệu vào **SharedPreferences** với một **key** và một **value**.
* Sử dụng các phương thức **"get"** trong đối tượng **SharedPreferences**, chẳng hạn như getInt() hoặc getString(), để lấy dữ liệu từ **SharedPreferences** bằng một **key**.
* Sử dụng phương thức clear() trong đối tượng **SharedPreferences.Editor** để xóa toàn bộ dữ liệu được lưu trong **SharedPreferences**.
* Sử dụng phương thức apply() trong đối tượng **SharedPreferences.Editor** để lưu các thay đổi vào tệp **SharedPreferences**.

**Khái niệm liên quan**

Tài liệu về khái niệm liên quan có trong:

* **9.0: Lưu trữ dữ liệu**
* **9.1: SharedPreferences**
* **Tìm hiểu thêm**

**Tài liệu dành cho nhà phát triển Android:**

* **Tổng quan về lưu trữ dữ liệu và tệp**
* **Lưu dữ liệu dưới dạng cặp khóa-giá trị**
* **SharedPreferences**
* **SharedPreferences.Editor**

**Stack Overflow:**

* **Cách sử dụng SharedPreferences trong Android để lưu trữ, truy xuất và chỉnh sửa giá trị**
* **onSavedInstanceState vs. SharedPreferences**

**Bài tập về nhà**

**Xây dựng và chạy ứng dụng**  
Mở ứng dụng **ScoreKeeper** mà bạn đã tạo trong bài học **Android Fundamentals 5.1: Drawables, styles, and themes**.

1. Thay thế trạng thái phiên đã lưu bằng **SharedPreferences** cho từng điểm số.
2. Để kiểm tra ứng dụng, xoay thiết bị để đảm bảo rằng khi cấu hình thay đổi, ứng dụng đọc dữ liệu từ **SharedPreferences** và cập nhật giao diện người dùng.
3. Dừng ứng dụng và khởi động lại để kiểm tra xem dữ liệu có được lưu trong **SharedPreferences** hay không.
4. Thêm nút **Reset** để đặt lại điểm số về **0** và xóa dữ liệu trong **SharedPreferences**.

* **Trả lời các câu hỏi**

**Câu hỏi 1:**  
Trong phương thức vòng đời nào bạn lưu trạng thái ứng dụng vào **SharedPreferences**?

**Câu hỏi 2:**  
Trong phương thức vòng đời nào bạn khôi phục trạng thái ứng dụng?

**Câu hỏi 3:**  
Bạn có thể nghĩ ra trường hợp nào mà việc sử dụng cả **SharedPreferences** và **Instance State** là hợp lý không?

* **Nộp ứng dụng để chấm điểm**

**Hướng dẫn cho người chấm điểm**  
Kiểm tra xem ứng dụng có các tính năng sau không:

* Ứng dụng giữ nguyên điểm số khi thiết bị xoay.
* Ứng dụng giữ nguyên điểm số hiện tại sau khi bị dừng và khởi động lại.
* Ứng dụng lưu điểm số vào **SharedPreferences** trong phương thức onPause().
* Ứng dụng khôi phục **SharedPreferences** trong phương thức onCreate().
* Ứng dụng có một nút **Reset** để đặt lại điểm số về **0**.

**Đảm bảo rằng phương thức xử lý sự kiện onClick của nút Reset thực hiện các tác vụ sau:**

* Đặt cả hai biến điểm số về **0**.
* Cập nhật cả hai **TextView**.
* Xóa dữ liệu trong **SharedPreferences**.

**Bài học 9.2: Cài đặt ứng dụng**

**Giới thiệu**

Các ứng dụng thường bao gồm **cài đặt** để cho phép người dùng tùy chỉnh tính năng và hành vi của ứng dụng. Ví dụ:

Một số ứng dụng cho phép người dùng đặt **vị trí nhà**, Chọn **đơn vị đo lường mặc định**, Và các tùy chọn khác ảnh hưởng đến toàn bộ ứng dụng. Người dùng không thường xuyên truy cập cài đặt, vì sau khi thiết lập (chẳng hạn như vị trí nhà), họ hiếm khi cần thay đổi lại.

Người dùng mong đợi có thể truy cập **cài đặt ứng dụng** bằng cách: Nhấn vào mục **Settings** trong **thanh điều hướng bên** (navigation drawer) – như hình bên trái. Hoặc trong **menu tùy chọn** trên **thanh ứng dụng** (app bar) – như hình bên phải.

Trong hình trên:

1. **Settings** trong **thanh điều hướng bên** (navigation drawer).
2. **Settings** trong **menu tùy chọn** của **thanh ứng dụng** (app bar).

**Trong bài thực hành này, bạn sẽ thêm một Settings Activity vào ứng dụng.** Người dùng có thể điều hướng đến **cài đặt ứng dụng** bằng cách nhấn vào **Settings**, được đặt trong **menu tùy chọn** trên **thanh ứng dụng** (app bar).

**Những kiến thức bạn cần biết trước**

Bạn nên có khả năng:

* Tạo một **dự án Android Studio** từ 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 **layout editor** và mã XML.
* Trích xuất tài nguyên chuỗi và chỉnh sửa giá trị chuỗi.
* Truy cập các phần tử giao diện từ mã bằng **findViewById()**.
* Xử lý sự kiện **nhấn nút (Button click)**.
* Hiển thị **Toast message**.
* Thêm một **Activity** vào ứng dụng.
* Tạo **menu tùy chọn** trên **thanh ứng dụng** (app bar).
* Thêm và chỉnh sửa **các mục menu** trong **menu tùy chọn**.
* Sử dụng **styles và themes** trong dự án.
* Sử dụng **SharedPreferences**.

Bạn sẽ học được gì?

Bạn sẽ học cách:

* Thêm một **Fragment** để quản lý **cài đặt**.
* Tạo một **tệp tài nguyên XML** chứa các cài đặt và thuộc tính của chúng.
* Tạo điều hướng đến **Settings Activity**.
* Đặt **giá trị mặc định** cho cài đặt.
* Đọc **giá trị cài đặt** do người dùng thay đổi.
* Tùy chỉnh **mẫu Settings Activity**.

**Những gì bạn sẽ làm**

* Tạo một ứng dụng có mục **Settings** trong **menu tùy chọn**.
* Thêm một **công tắc bật/tắt (toggle switch)** cho tùy chọn **Settings**.
* Thêm mã để đặt giá trị mặc định cho cài đặt và truy cập giá trị cài đặt sau khi nó đã thay đổi.
* Sử dụng và tùy chỉnh mẫu **Settings Activity** của Android Studio.

**Tổng quan về ứng dụng**

Android Studio cung cấp một phím tắt để thiết lập menu tùy chọn với **Cài đặt**. Nếu bạn bắt đầu một dự án Android Studio cho điện thoại hoặc máy tính bảng bằng mẫu **Basic Activity**, ứng dụng mới sẽ bao gồm **Cài đặt**, như được hiển thị bên dưới:

ảnh

Mẫu này cũng bao gồm một nút hành động nổi (**Floating Action Button**) ở góc dưới bên phải của màn hình với biểu tượng phong bì. Bạn có thể bỏ qua nút này trong bài thực hành này, vì bạn sẽ không sử dụng nó.

Bạn sẽ bắt đầu bằng cách tạo một ứng dụng có tên **AppWithSettings** bằng mẫu **Basic Activity**, sau đó thêm một **Settings Activity**, cung cấp một cài đặt công tắc bật/tắt mà người dùng có thể chuyển đổi giữa bật hoặc tắt.

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

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

Mẫu **Settings Activity** được điền sẵn các cài đặt mà bạn có thể tùy chỉnh cho ứng dụng và cung cấp một bố cục khác nhau cho điện thoại và máy tính bảng.

* Điện thoại: Một màn hình **Cài đặt chính** với liên kết tiêu đề cho từng nhóm cài đặt, chẳng hạn như **General** cho cài đặt chung, như hiển thị bên dưới.

ảnh

* Máy tính bảng: Một bố cục màn hình **chính/chi tiết**, với liên kết tiêu đề cho từng nhóm cài đặt ở bên trái (màn hình chính) và nhóm cài đặt tương ứng ở bên phải (màn hình chi tiết), như hiển thị trong hình bên dưới.

ảnh

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

Ứng dụng **DroidCafeOptionsUp** đã được tạo trong bài học trước từ mẫu **Basic Activity**, mẫu này cung cấp một menu tùy chọn trên thanh ứng dụng để đặt tùy chọn **Cài đặt**.

Bạn sẽ tùy chỉnh mẫu **Settings Activity** được cung cấp bằng cách thay đổi tiêu đề, mô tả, giá trị và giá trị mặc định của một cài đặt. Bạn cũng sẽ thêm mã để đọc giá trị cài đặt sau khi người dùng thay đổi nó và hiển thị giá trị đó.

**Nhiệm vụ 1: Thêm một cài đặt công tắc (Switch Setting) vào ứng dụng**

Trong nhiệm vụ này, bạn sẽ thực hiện các bước sau:

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

**1.1 Tạo dự án và thêm thư mục XML cùng tệp tài nguyên**

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

|  |  |  |  |
| --- | --- | --- | --- |
| Thuộc tính | |  | | --- | |  |  |  | | --- | | **Giá trị** | |
| Tên ứng dụng | AppWithSettings |
| Tên công ty | android.example.com (hoặc tên miền của bạn) |
| |  | | --- | | Vị trí dự án (**Project location**) |  |  | | --- | |  | | Đường dẫn đến thư mục chứa dự án của bạn |
| SDK tối thiểu cho điện thoại và máy tính bảng | API 15: Android 4.0.3 IceCreamSandwich |
| Mẫu | Basic Activity |
| Tên Activity | MainActivity |
| **Tên Layout** | activity\_main |
| Tiêu đề | MainActivity |
|  |  |
|  |  |

1. Chạy ứng dụng và nhấn vào biểu tượng **overflow** trên thanh ứng dụng để xem menu tùy chọn, như hiển thị trong hình bên dưới. Mục duy nhất trong menu tùy chọn là **Settings**.

ảnh

1. Bạn cần tạo một thư mục tài nguyên mới để chứa tệp XML cài đặt.Chọn thư mục **res** trong bảng **Project > Android**, sau đó chọn **File > New > Android Resource Directory**. Hộp thoại **New Resource Directory** sẽ xuất hiện.
2. Trong menu thả xuống **Resource type**, chọn **xml**. **Tên thư mục** sẽ tự động thay đổi thành **xml**. Nhấn **OK**.
3. Thư mục **xml** sẽ xuất hiện trong bảng **Project > Android** bên trong thư mục **res**. Chọn **xml** và chọn **File > New > XML resource file** (hoặc nhấp chuột phải vào **xml** và chọn **New > XML resource file**).
4. Nhập tên tệp XML là **preferences** vào trường **File name**, sau đó nhấn **OK**. Tệp **preferences.xml** sẽ xuất hiện trong thư mục **xml**, và trình chỉnh sửa giao diện (**layout editor**) sẽ hiển thị, như trong hình bên dưới.

ảnh trang 23

Trong hình trên:

1. Tệp **preferences.xml** bên trong thư mục **xml**.
2. Trình chỉnh sửa giao diện (**layout editor**) hiển thị nội dung của **preferences.xml**.

**1.2. Thêm Preference XML và các thuộc tính cho cài đặt.**

1. Kéo một SwitchPreference từ bảng Palette ở bên trái vào đầu bố cục, như hiển thị trong hình bên dưới.

ảnh

1. Thay đổi các giá trị trong bảng **Attributes** ở bên phải của trình chỉnh sửa giao diện như sau, và như hiển thị trong hình bên dưới:

○ **defaultValue**: true  
○ **key**: example\_switch

○ **title**: Settings option  
○ **summary**: Turn this option on or off

ảnh

1. Nhấp vào tab **Text** ở dưới cùng của trình chỉnh sửa giao diện để xem mã XML.

ảnh

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

Các thuộc tính XML cho một **Preference** bao gồm:

● **android:defaultValue**: Giá trị mặc định của cài đặt khi ứng dụng khởi chạy lần đầu tiên.  
● **android:title**: Tiêu đề của cài đặt. Đối với **SwitchPreference**, tiêu đề xuất hiện bên trái của công tắc bật/tắt.  
● **android:key**: Khóa 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 trữ cài đặt trong tệp **SharedPreferences** mặc định của ứng dụng.  
● **android:summary**: Văn bản mô tả xuất hiện bên dưới cài đặt.

**1.3 Sử dụng SwitchPreferenceCompat**

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

* 1. Trong bảng **Project > Android**, mở tệp **build.gradle (Module: app)** trong thư mục **Gradle Scripts**, và thêm dòng sau vào phần **dependencies**:

ảnh

Câu lệnh trên thêm thư viện **android.support.v7.preference** để sử dụng **PreferenceFragmentCompat** thay cho **PreferenceFragment**.

* 1. Trong tệp **preferences.xml** trong thư mục **xml**, thay đổi <SwitchPreference trong mã thành <android.support.v7.preference.SwitchPreferenceCompat.

ảnh

Dòng **SwitchPreferenceCompat** ở trên có thể hiển thị biểu tượng bóng đèn màu vàng với cảnh báo, nhưng bạn có thể bỏ qua nó ngay bây giờ.

* 1. Mở tệp **styles.xml** trong thư mục **values**, và thêm mục **preferenceTheme** sau vào phần khai báo **AppTheme**:

ảnh

Để sử dụng **PreferenceFragmentCompat**, bạn cũng phải khai báo **preferenceTheme** với kiểu **PreferenceThemeOverlay** trong chủ đề của ứng dụng.

**1.4 Thêm một Activity cho cài đặt**

Để tạo một **Settings Activity** cung cấp giao diện người dùng cho cài đặt, hãy thêm một **Empty Activity** vào ứng dụng. Thực hiện các bước sau:

1. Chọn **app** ở đầu bảng **Project > Android**, sau đó chọn **New > Activity > Empty Activity**.
2. Đặt tên **Activity** là **SettingsActivity**. Bỏ chọn tùy chọn **Generate Layout File** (vì bạn không cần tệp giao diện), và giữ nguyên tùy chọn **Launcher Activity** ở trạng thái bỏ chọn.
3. Giữ nguyên tùy chọn **Backwards Compatibility (AppCompat)** được chọn. **Package name** sẽ tự động được đặt thành **com.example.android.projectname**.
4. Chọn kết thúc.

**1.5 Thêm một Fragment cho một cài đặt cụ thể**

Một **Fragment** giống như một phần độc lập của **Activity** — nó có vòng đời riêng, nhận các sự kiện đầu vào riêng và có thể được thêm hoặc xóa trong khi **Activity** đang chạy.

Bạn sẽ sử dụng một lớp con chuyên biệt của **Fragment** để hiển thị danh sách cài đặt. Cách tốt nhất là sử dụng một **Activity** thông thường để chứa một **PreferenceFragment**, hiển thị cài đặt của ứng dụng. **PreferenceFragment** cung cấp một kiến trúc linh hoạt hơn so với việc sử dụng **Activity** để quản lý cài đặt.

Bạn sẽ sử dụng **PreferenceFragmentCompat** thay vì **PreferenceFragment** để đảm bảo khả năng tương thích với **AppCompatActivity**.

Trong bước này, bạn sẽ thêm một **Blank Fragment** để nhóm các cài đặt tương tự (không có layout, phương thức factory hoặc interface callbacks) vào ứng dụng và mở rộng từ **PreferenceFragmentCompat**.

Thực hiện các bước sau:

1. Chọn **app**, sau đó chọn **New > Fragment > Fragment (Blank)**.
2. Đặt tên **Fragment** là **SettingsFragment**. Bỏ chọn tùy chọn **Create layout XML?** (vì bạn không cần tệp giao diện).
3. Bỏ chọn các tùy chọn để thêm **fragment factory methods** và **interface callbacks**.
4. **Target Source Set** nên được đặt thành **main**.
5. Nhấn **Finish**. Kết quả là lớp **SettingsFragment** sau sẽ được tạo:

ảnh

1. Chỉnh sửa phần định nghĩa lớp **SettingsFragment** để mở rộng từ **PreferenceFragmentCompat**.

ảnh

Khi bạn chỉnh sửa định nghĩa lớp để khớp với định nghĩa ở trên, một biểu tượng bóng đèn đỏ sẽ xuất hiện ở lề bên trái. Nhấp vào bóng đèn đỏ và chọn **Implement methods**, sau đó chọn **onCreatePreferences**. Android Studio sẽ tạo ra một phương thức **onCreatePreferences()** mẫu như sau:

ảnh

Để mở rộng **Fragment**, Android Studio sẽ thêm câu lệnh **import** sau:

ảnh

1. Xóa toàn bộ phương thức **onCreateView()** trong **Fragment**.

Lý do bạn thay thế **onCreateView()** bằng **onCreatePreferences()** là vì bạn sẽ thêm **SettingsFragment** vào **SettingsActivity** hiện có để hiển thị cài đặt, thay vì hiển thị một màn hình **Fragment** riêng biệt. Việc thêm **Fragment** vào **Activity** giúp bạn dễ dàng thêm hoặc xóa **Fragment** trong khi **Activity** đang chạy. **PreferenceFragment** được gắn vào **PreferenceScreen** bằng **rootKey**.

Bạn cũng có thể xóa an toàn constructor rỗng của **SettingsFragment**, vì **Fragment** này không được hiển thị độc lập.

ảnh

1. Bạn cần liên kết **Fragment** này với tệp tài nguyên cài đặt **preferences.xml** mà bạn đã tạo ở bước trước. Thêm vào phương thức onCreatePreferences() vừa tạo một lệnh gọi đến setPreferencesFromResource(), truyền vào **ID của tệp XML** (R.xml.preferences) và rootKey để xác định **gốc của PreferenceScreen**.

ảnh

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

ảnh

**1.6 Hiển thị Fragment trong SettingsActivity**

Để hiển thị **Fragment** trong **SettingsActivity**, hãy làm theo các bước sau:

1. Mở **SettingsActivity**.
2. Thêm đoạn mã sau vào cuối phương thức onCreate() để hiển thị **Fragment** làm nội dung chính:

ảnh

Đoạn mã trên sử dụng mẫu thông thường để thêm **Fragment** vào **Activity**, giúp Fragment hiển thị làm nội dung chính của Activity:

* Sử dụng getFragmentManager() nếu lớp kế thừa từ **Activity** và **Fragment** kế thừa từ **PreferenceFragment**.
* Sử dụng getSupportFragmentManager() nếu lớp kế thừa từ **AppCompatActivity** và **Fragment** kế thừa từ **PreferenceFragmentCompat**.

Phương thức onCreate() hoàn chỉnh trong **SettingsActivity** bây giờ sẽ trông như sau:

ảnh

**1.7 Kết nối mục Settings trong menu với SettingsActivity**

Sử dụng **Intent** để khởi chạy **SettingsActivity** từ **MainActivity** khi người dùng chọn **Settings** trong menu tùy chọn.

1. Mở **MainActivity** và tìm khối if trong phương thức onOptionsItemSelected(), nơi xử lý sự kiện nhấn vào **Settings** trong menu tùy chọn:

ảnh

1. Thêm một **Intent** vào khối if để khởi chạy **SettingsActivity**.

ảnh

1. Để thêm nút **Up** trên **thanh ứng dụng** (app bar) vào **SettingsActivity**, bạn cần chỉnh sửa khai báo của nó trong tệp **AndroidManifest.xml** để xác định **MainActivity** là **Activity cha** của **SettingsActivity**. Mở **AndroidManifest.xml** và tìm khai báo của **SettingsActivity**:

ảnh

Thay đổi khai báo thành như sau:

ảnh

1. Chạy ứng dụng. Nhấn vào **biểu tượng dấu ba chấm** để mở **menu tùy chọn**, như trong hình bên trái. Nhấn vào **Settings** để mở **SettingsActivity**, như trong hình ở giữa. Nhấn vào **nút Up** trên **thanh ứng dụng** của **SettingsActivity**, như trong hình bên phải, để quay lại **MainActivity**.

ảnh

**1.8 Lưu giá trị mặc định vào SharedPreferences**

Mặc dù giá trị mặc định cho **công tắc bật/tắt (toggle switch)** đã được đặt trong thuộc tính android:defaultValue (ở **Bước 1.2** của nhiệm vụ này), ứng dụng vẫn cần lưu giá trị mặc định vào tệp **SharedPreferences** cho mỗi cài đặt khi người dùng mở ứng dụng lần đầu tiên.

Thực hiện các bước sau để đặt giá trị mặc định cho **toggle switch**:

1. Mở **MainActivity**.
2. Thêm đoạn mã sau vào cuối phương thức onCreate(), sau phần mã của **FloatingActionButton**.

ảnh

Đoạn mã trên đảm bảo rằng các cài đặt được khởi tạo đúng với giá trị mặc định của chúng.

Phương thức PreferenceManager.setDefaultValues() nhận **ba đối số**:

* **Context** của ứng dụng, chẳng hạn như this.
* **ID tài nguyên** (preferences) của tệp XML chứa một hoặc nhiều cài đặt.
* Một **giá trị boolean** xác định liệu các giá trị mặc định có nên được thiết lập nhiều lần hay không: Nếu false, hệ thống chỉ thiết lập các giá trị mặc định nếu phương thức này chưa từng được gọi trước đó. Điều này có nghĩa là bạn có thể gọi phương thức này mỗi khi **MainActivity** khởi động mà không ghi đè lên các giá trị đã lưu của người dùng. Nếu true, phương thức sẽ **ghi đè** bất kỳ giá trị nào trước đó bằng các giá trị mặc định.

**1.9 Đọc giá trị cài đặt đã thay đổi từ SharedPreferences**

Khi ứng dụng khởi động, phương thức onCreate() của **MainActivity** có thể đọc các giá trị cài đặt đã thay đổi và sử dụng chúng thay vì các 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** 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 dùng **khóa (key)** để lấy cài đặt từ tệp **SharedPreferences** bằng phương thức PreferenceManager.getDefaultSharedPreferences().

Thực hiện các bước sau để thêm đoạn mã này:

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

ảnh

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

ảnh

1. Chạy ứng dụng và nhấn vào **Settings** để mở **SettingsActivity**.
2. Nhấn vào tùy chọn cài đặt để chuyển **toggle** từ **bật** sang **tắt**, như trong hình bên trái dưới đây.
3. Nhấn nút **Up** trong **SettingsActivity** để quay lại **MainActivity**. Một thông báo **Toast** sẽ xuất hiện trong **MainActivity** với giá trị của cài đặt, như trong hình bên phải dưới đây.
4. Lặp lại các bước này để xem 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 một 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 cho biến switchPref. Nếu không có giá trị nào cho khóa này, phương thức getBoolean() sẽ đặt giá trị của switchPref thành **false**. Đối với các kiểu dữ liệu khác như **chuỗi (String), số nguyên (Integer), hoặc số thực (Floating point)**, bạn có thể sử dụng tương ứng các phương thức getString(), getInt(), hoặc getFloat().
* 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 các giá trị cài đặt để sử dụng trong ứng dụng. Phương thức Toast.makeText() sau này có thể được thay thế bằng một phương thức khởi tạo cài đặt.

Bây giờ, ứng dụng của bạn đã có một **SettingsActivity** hoạt động hoàn chỉnh.

**Mã nguồn giải pháp Task 1**  
Dự án Android Studio: AppWithSettings

**Nhiệm vụ 2: Sử dụng mẫu Settings Activity**

Nếu bạn cần xây dựng nhiều màn hình con cho cài đặt và muốn tận dụng kích thước màn hình máy tính bảng, đồng thời 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 Settings Activity trống và một Fragment trống để thêm 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 cài đặt và giá trị của chúng.
* Hiển thị màn hình Settings chính, trong đó mỗi nhóm cài đặt sẽ có một tiêu đề liên kết, chẳng hạn như General cho cài đặt chung, như trong hình minh họa dưới đây.

ảnh

* Hiển thị bố cục màn hình **master/detail**, trong đó: **Phía bên trái (master):** Chứa các liên kết tiêu đề cho từng nhóm cài đặt. **Phía bên phải (detail):** Hiển thị nhóm cài đặt tương ứng khi được chọn, như trong hình minh họa dưới đây.

ảnh

* 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 trên thanh ứng dụng, như minh họa bên dưới.

Chú thích cho hình minh họa ở trên:

ảnh

1. Thanh ứng dụng (App bar)
2. Biểu tượng hành động trong 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 bình thường.
2. Trong **Project > Android**, chọn **app** ở đầu danh sách, sau đó nhấp chuột phải và chọn **New > Activity > Settings Activity**.
3. Trong hộp thoại xuất hiện, chấp nhận tên mặc định của **Activity** (**SettingsActivity** là tên được đề xuất) và tiêu đề (**Settings**).
4. Nhấp vào biểu tượng ba chấm ở cuối menu **Hierarchical Parent**, sau đó chọn tab **Project** trong hộp thoại **Select Activity**.
5. Mở rộng thư mục **DroidCafeWithSettings > app > src > main > java > com.example.android.droidcafeinput** và chọn **MainActivity** làm **Parent Activity**, như minh họa trong hình bên dưới. Nhấn **OK**.

ảnh

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

1. Nhấn **Finish**.
2. Trong **Project > Android**, mở rộng thư mục **app > res > xml** để xem các tệp XML được tạo bởi mẫu **Settings Activity**.

ảnh

Bạn có thể mở và tùy chỉnh các tệp XML để thêm hoặc chỉnh sửa 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 bố cục XML trên sử dụng nhiều lớp con khác nhau của lớp **Preference** thay vì **View**. Một số lớp con trực tiếp của **Preference** đóng vai trò làm container cho bố cục chứa nhiều cài đặt khác nhau. Ví dụ: **PreferenceScreen** đại diện cho một **Preference** cấp cao nhất, 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 phù hợp để người dùng thay đổi cài đặt. Ví dụ:

* **CheckBoxPreference**: Một hộp kiểm cho cài đặt có thể được bật hoặc tắt.
* **ListPreference**: Một hộp thoại chứa danh sách các nút radio.
* **SwitchPreference**: Một tùy chọn hai trạng thái có thể bật/tắt (ví dụ: On/Off 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 hiển thị trong phần cài đặt. Tất cả chuỗi văn bản được sử dụng trong **Settings Activity**, bao gồm tiêu đề cài đặt, mảng chuỗi cho danh sách và mô tả cài đặt, đều được định nghĩa trong tệp **strings.xml**. Các chuỗi này được đánh dấu bằng các nhận xét như sau:

<!‑‑ Strings liên quan đến Settings ‑‑>

<!‑‑ Ví dụ: Cài đặt chung ‑‑>

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

* **SettingsActivity** trong thư mục **java/com.example.android.projectname**, có thể sử dụng ngay mà không cần chỉnh sửa. Đây là **Activity** hiển thị phần cài đặt. **SettingsActivity** kế thừa từ **AppCompatPreferenceActivity** để duy trì khả năng tương thích với các phiên bản Android cũ hơn.
* **AppCompatPreferenceActivity** trong thư mục **java/com.example.android.projectname**, có thể sử dụng ngay mà không cần chỉnh sửa. Đây là một **Activity** trợ giúp mà **SettingsActivity** sử dụng để duy trì khả năng tương thích ngược với các phiên bản Android trước đó.

**2.2 Thêm mục menu "Settings" và kết nối nó 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 trong menu tùy chọn.

1. Mở rộng thư mục **res** trong **Project > Android**, 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**:

ảnh

Bạn đặt giá trị **"never"** cho thuộc tính **app:showAsAction** để mục **Settings** chỉ xuất hiện trong menu tùy chọn tràn (overflow menu) và không hiển thị trực tiếp trên thanh ứng dụng, vì mục này không 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 **bên trên** **Contact** (được đặt là **"100"**).

1. Trích xuất chuỗi tài nguyên cho **"Settings"** trong thuộc tính **android:title** và đặt tên tài nguyên là **settings**.
2. Mở **MainActivity**, tìm khối **switch-case** trong phương thức **onOptionsItemSelected()**, nơi xử lý sự kiện khi người dùng nhấn vào các mục trong menu tùy chọn. Dưới đây là một đoạn mã minh họa cách xử lý **case** đầu tiên (cho **action\_order**):ảnh

ảnh

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

ảnh

1. Chạy ứng dụng trên điện thoại hoặc trình giả lập để xem cách **Settings Activity template** xử lý kích thước màn hình điện thoại.
2. Nhấn vào biểu tượng **overflow** trong menu tùy chọn, sau đó chọn **Settings** để mở **Settings Activity**, như hiển thị ở phía bên trái của hình minh họa bên dưới.
3. Nhấn vào từng tiêu đề cài đặt (**General**, **Notifications**, và **Data & Sync**), như hiển thị ở giữa hình minh họa bên dưới, để xem nhóm cài đặt tương ứng trên từng màn hình con của **Settings screen**, được hiển thị ở phía bên phải của hình minh họa.
4. Nhấn nút **Up** trong **Settings Activity** để quay lại **MainActivity**.

ảnh

Bạn sử dụng mã của **Settings Activity template** mà không cần chỉnh sửa. Mẫu này không chỉ cung cấp bố cục cho màn hình điện thoại và máy tính bảng, mà còn có chức năng **lắng nghe thay đổi cài đặt** và **cập nhật phần tóm tắt** để phản ánh thay đổi đó.

Ví dụ: Nếu bạn thay đổi cài đặt **"Add friends to messages"** (với các lựa chọn **Always**, **When possible**, hoặc **Never**), thì lựa chọn bạn đã chọn sẽ hiển thị trong phần tóm tắt bên dưới cài đặt.

ảnh

Nhìn chung, bạn không cần thay đổi mã của **Settings Activity template** để tùy chỉnh **Activity** theo các cài đặt mong muốn trong ứng dụng của mình.

Bạn có thể **tùy chỉnh** tiêu đề cài đặt, phần tóm tắt, các giá trị có thể chọn, giá trị mặc định mà không cần sửa đổi mã của mẫu, và thậm chí có thể **thêm nhiều cài đặt hơn** vào các nhóm có sẵn.

**2.3 Tùy chỉnh cài đặt do mẫu cung cấp**

Để tùy chỉnh các cài đặt được cung cấp bởi **Settings Activity template**, hãy chỉnh sửa **chuỗi văn bản** và **mảng chuỗi** trong tệp **strings.xml**, cũng như các thuộc tính bố cục cho từng cài đặt 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 nhận xét **<!‑‑ Example settings for Data & Sync ‑‑>**.

ảnh

1. Chỉnh sửa **pref\_header\_data\_sync**, chuỗi tài nguyên hiện đang được đặt là **Data & sync** (**&** là mã HTML cho dấu "&"). Thay đổi giá trị thành **Account** (không có dấu ngoặc kép).
2. Bạn nên **đổi tên tài nguyên** để giúp mã dễ hiểu hơn (ứng dụng vẫn hoạt động nếu không đổi tên, nhưng việc này giúp mã rõ ràng hơn). Nhấp chuột phải (**hoặc Control + click**) vào tài nguyên **pref\_header\_data\_sync**, chọn **Refactor > Rename**. Đổi tên thành **pref\_header\_account**, chọn tùy chọn **search in comments and strings**, sau đó nhấn **Refactor**.
3. Bạn cũng nên **đổi tên tệp XML** để giúp mã dễ hiểu hơn. Nhấp chuột phải (**hoặc Control + click**) vào tài nguyên **pref\_data\_sync** trong **Project > Android**, chọn **Refactor > Rename**. Đổi tên thành **pref\_account**, chọn tùy chọn **search in comments and strings**, sau đó nhấn **Refactor**.
4. Chỉnh sửa chuỗi tài nguyên **pref\_title\_sync\_frequency** (hiện đang đặt là **Sync frequency**) và thay đổi thành **Market**.
5. Thực hiện **Refactor > Rename** cho tài nguyên **pref\_title\_sync\_frequency**, đổi tên thành **pref\_title\_account**, giống như các bước trước.
6. Thực hiện **Refactor > Rename** cho mảng chuỗi **pref\_sync\_frequency\_titles**, đổi tên thành **pref\_market\_titles**.
7. Thay đổi từng giá trị trong mảng **pref\_market\_titles** (ví dụ: **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.

ảnh

1. Thực hiện **Refactor > Rename** cho tên tài nguyên mảng chuỗi **pref\_sync\_frequency\_values**, đổi tên thành **pref\_market\_values**.
2. Thay đổi từng giá trị trong mảng **pref\_market\_values** (ví dụ: **15**, **30**, **60**, v.v.) thành các giá trị tương ứng với thị trường—sử dụng các chữ viết tắt của quốc gia tương ứng, chẳng hạn như **US**, **CA**, v.v.

ảnh

1. Cuộn xuống tài nguyên chuỗi **pref\_title\_system\_sync\_settings** và chỉnh sửa giá trị (hiện đang đặt là **System sync settings**) thành **Account settings**.
2. Thực hiện **Refactor > Rename** cho tên tài nguyên chuỗi **pref\_title\_system\_sync\_settings**, đổi tên thành **pref\_title\_account\_settings**.
3. Mở tệp **pref\_account.xml**. Thành phần **ListPreference** trong bố cục này xác định cài đặt mà bạn vừa thay đổi. Lưu ý rằng các tài nguyên chuỗi cho thuộc tính **android:entries**, **android:entryValues**, và **android:title** hiện đã được thay đổi thành các giá trị bạn đã cập nhật trong các bước trước.

ảnh

1. Thay đổi thuộc tính **android:defaultValue** thành **"US"**.

ảnh

Vì khóa của tùy chọn cài đặt này (**"sync\_frequency"**) đã được mã hóa cứng ở những nơi khác trong mã Java, **đừng thay đổi** thuộc tính **android:key**. Thay vào đó, hãy tiếp tục sử dụng **"sync\_frequency"** làm khóa cho cài đặt này trong ví dụ này.

Nếu bạn đang **tùy chỉnh toàn diện** cài đặt cho một ứng dụng thực tế, bạn nên dành thời gian để thay đổi tất cả các khóa đã mã hóa cứng trong toàn bộ mã.

**Lưu ý:** Tại sao không sử dụng một tài nguyên chuỗi cho khóa?

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 chuỗi khóa vô tình bị dịch cùng với các chuỗi khác, điều đó có thể khiến ứng dụng **bị lỗi**.

**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()**.

ảnh

Các giá trị mặc định đã được chỉ định trong tệp XML bằng thuộc tính **android:defaultValue**, nhưng 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 của ứng dụng** (*context*), chẳng hạn như **this**.
* **ID tài nguyên** của tệp bố cục XML cài đặt, trong đó bao gồm các giá trị mặc định được đặt bằng thuộc tính **android:defaultValue**.
* **Một giá trị boolean** chỉ định liệu có nên đặt các giá trị mặc định nhiều lần hay không. Khi đặt là **false**, hệ thống sẽ chỉ đặt các giá trị mặc định **một lần duy nhất**, khi phương thức này được gọi lần đầu tiên. Miễn là bạn đặt đối số thứ ba này thành **false**, bạn có thể gọi phương thức này an toàn mỗi khi ứng dụng khởi động.

**Hoạt động** sẽ bắt đầu mà không ghi đè các giá trị cài đặt đã lưu của người dùng bằng cách đặt lại chúng về giá trị mặc định. Tuy nhiên, nếu bạn đặt giá trị này thành **true**, phương thức sẽ **ghi đè** mọi giá trị 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 các giá trị mặc định cho cài đặt.

ảnh

Như bạn đã học ở nhiệm vụ trước, bạn sử dụng **PreferenceManager.getDefaultSharedPreferences(this)** để lấy cài đặt dưới dạng 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 **-1** cho *marketPref*, đây là giá trị của **Other** trong mảng **pref\_market\_values**.

1. Chạy ứng dụng. Khi màn hình chính của ứng dụng xuất hiện lần đầu tiên, bạn sẽ thấy một thông báo **Toast** hiển thị ở cuối màn hình.
   * Lần đầu chạy ứng dụng, bạn sẽ thấy **"-1"** xuất hiện trong thông báo **Toast**, vì bạn chưa thay đổi cài đặt nào.
2. Nhấn vào **Settings** trong menu tùy chọn, sau đó nhấn **Account** trong màn hình **Settings**.
   * Nhấn vào **Market** và chọn **Canada**, như minh họa bên dưới.
3. Nhấn vào nút **Up** trên thanh ứng dụng để quay lại màn hình **Settings**, sau đó nhấn lần nữa để quay lại màn hình chính.
4. Chạy lại ứng dụng từ **Android Studio**. Bạn sẽ thấy một thông báo **Toast** hiển thị **"CA"** (đại diện cho Canada), và cài đặt **Market** bây giờ đã được đặt thành **Canada**.

ảnh

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

ảnh

Mã giải pháp cho **Nhiệm vụ 2**  
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 phần cài đặt đúng cách trên màn hình có kích thước máy tính bảng, nhưng nút **Up** trên thanh ứng dụng không đưa người dùng quay lại **MainActivity** như trên màn hình có kích thước điện thoại. Nguyên nhân là do có ba phương thức **onOptionsItemSelected()**—mỗi phương thức dành cho một **Fragment**—trong **SettingsActivity**. Nó sử dụng đoạn mã sau để khởi động lại **SettingsActivity** khi người dùng nhấn vào nút **Up**:

Hành động trên phù hợp với 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 cài đặt, khi người dùng nhấn vào nút **Up**, họ sẽ quay lại màn hình tiêu đề cài đặt trong **SettingsActivity**. Nếu nhấn **Up** lần nữa, họ sẽ trở về **MainActivity**.

Tuy nhiên, trên máy tính bảng, các tiêu đề luôn hiển thị ở ngăn bên trái (trong khi phần cài đặt hiển thị ở ngăn bên phải). Do đó, nhấn vào nút **Up** không đưa người dùng trở 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:** Mặc dù có nhiều cách để khắc phục vấn đề này, bạn có thể tham khảo các bước sau:

1. **Thêm một tệp dimens.xml mới** để hỗ trợ các màn hình có kích thước lớn hơn 600dp.

Khi ứng dụng chạy trên một thiết bị cụ thể, hệ thống sẽ tự động chọn tệp dimens.xml phù hợp dựa trên các bộ định tính (**qualifiers**). Bạn có thể thêm một tệp dimens.xml mới với bộ định tính **Smallest Screen Width** được đặt thành **600dp** (sw600dp) như đã học trong bài thực hành về hỗ trợ chế độ xoay màn hình và kích thước màn hình. Điều này giúp xác định bất kỳ thiết bị nào có màn hình lớn, chẳng hạn như máy tính bảng.

1. **Thêm một tài nguyên bool** vào giữa các thẻ <resources> và </resources> trong tệp dimens.xml (sw600dp). Tệp này sẽ tự động được chọn cho các thiết bị có màn hình lớn (máy tính bảng).

ảnh

1. 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** phải màn hình lớn.

ảnh

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

**Mã giải pháp cho thử thách**  
Dự án Android Studio: **DroidCafeWithSettingsChallenge**

**Tóm tắt**

Người dùng mong muốn có thể điều hướng đến cài đặt ứng dụng bằng cách nhấn vào **Settings** trong điều hướng bên (chẳng hạn như **navigation drawer**) hoặc trong **options menu** trên thanh ứng dụng (**app bar**).

Để cung cấp cài đặt cho ứng dụng của bạn, hãy tạo một **Activity** dành cho phần cài đặt:

* Sử dụng một **Activity** chứa **PreferenceFragment** để hiển thị cài đặt của 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 **SettingsActivity**:

* Nếu lớp **Activity** kế thừa từ **Activity** và lớp **Fragment** kế thừa từ **PreferenceFragment**, sử dụng **getFragmentManager()**.
* Nếu lớp **Activity** kế thừa từ **AppCompatActivity** và lớp **Fragment** kế thừa 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 **Settings** trong menu với **SettingsActivity**, sử dụng **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**, sau đó chọn **File > New > XML resource file**.
4. Nhập **preferences** làm tên của tệp XML. Tệp **preferences.xml** sẽ 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 **preferences**:

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 phần tử 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 bên dưới cài đặt cho người dùng.

Lưu và đọc giá trị cài đặt:  
● 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 (*key-value pair*). 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 (*key*) để lấy giá trị cài đặt từ tệp **SharedPreferences**.  
● Ứng dụng đọc giá trị cài đặt từ **SharedPreferences** bằng cách sử dụng **PreferenceManager.getDefaultSharedPreferences()**, và lấy từng giá trị cài đặt bằng các phương thức **.getString**, **.getBoolean**, v.v.

**Các khái niệm liên quan**

Tài liệu về các khái niệm liên quan có trong **9.2: Cài đặt ứng dụng**.

**Tìm hiểu thêm**

Tài liệu hướng dẫn sử dụng Android Studio: [Android Studio User Guide]

Tài liệu dành cho nhà phát triển Android: [Android Developer Documentation]

* **Cài đặt (Tổng quan)**
* **Preference (Tùy chọn)**
* **PreferenceFragment (Phân mảnh tùy chọn)**
* **PreferenceFragmentCompat (Phân mảnh tùy chọn tương thích)**
* **Fragment (Phân mảnh)**
* **SharedPreferences (Bộ lưu trữ dữ liệu chia sẻ)**
* **Lưu dữ liệu dưới dạng cặp key-value**
* **Hỗ trợ các kích thước màn hình khác nhau**

**Tài liệu về Material Design:**

* **Thiết kế giao diện Android settings theo chuẩn Material Design**

**Thảo luận trên Stack Overflow:**

* **Làm thế nào để thêm tệp dimens.xml vào Android Studio?**
* **Cách xác định thiết bị là điện thoại hay máy tính bảng?**

**Bài tập về nhà**

**Xây dựng và chạy ứng dụng**

Mở dự án **DroidCafeWithSettings** và thực hiện các yêu cầu sau:

1. **Thêm một ListPreference (hộp thoại với các nút radio) vào phần cài đặt chung** Đặt hộp thoại này trong màn hình **General settings**, bên dưới ListPreference có tên **"Add friends to order messages"**.
2. **Chỉnh sửa mảng chuỗi (string-array) sử dụng cho ListPreference** Bao gồm tiêu đề **"Choose a delivery method."** Sử dụng cùng các phương thức giao hàng đã có trong các nút radio trong **OrderActivity**.
3. **Hiển thị phương thức giao hàng được chọn trong thông báo Toast** Kết hợp nó với giá trị của **Market setting** đã chọn.
4. **Bài tập nâng cao (Extra Credit)** Hiển thị phương thức giao hàng đã chọn như một phần văn bản tóm tắt (summary) bên dưới tiêu đề ListPreference. Cho phép văn bản này thay đổi khi người dùng cập nhật cài đặt.

**ảnh**

**Trả lời các câu hỏi**

**Câu hỏi 1**

Trong dự án **DroidCafeWithSettings**, bạn định nghĩa mảng các mục nhập (entries) và mảng giá trị (values) cho **ListPreference** trong tệp nào? Chọn một:  
● **pref\_general.xml**  
● **strings.xml** ✅  
● menu\_main.xml  
● activity\_main.xml  
● content\_main.xml

Câu hỏi 2  
Trong dự án **DroidCafeWithSettings**, bạn **sử dụng** mảng các mục nhập (**entries**) và mảng giá trị (**values**) để thiết lập **ListPreference**, đồng thời đặt khóa (**key**) và giá trị mặc định (**default value**) trong tệp nào?

Chọn một:  
● **pref\_general.xml** ✅  
● strings.xml  
● menu\_main.xml  
● content\_main.xml  
● SettingsActivity.java

**Câu hỏi 3**  
Làm thế nào để thiết lập một **Settings Activity** và một **Fragment** với **SwitchPreference** cho giao diện người dùng, đồng thời vẫn tương thích với **thư viện v7 appcompat** để hỗ trợ các phiên bản Android cũ hơn?

**Chọn một:**  
● Sử dụng một **Settings Activity** mở rộng từ Activity, một **Fragment** mở rộng từ PreferenceFragment, và một SwitchPreference cho giao diện người dùng.  
● Thay đổi MainActivity để mở rộng từ Activity.  
● **Sử dụng một Settings Activity mở rộng từ AppCompatActivity, một Fragment mở rộng từ PreferenceFragmentCompat, và một SwitchPreferenceCompat cho UI.** ✅  
● Không thể sử dụng một Fragment với SwitchPreference và vẫn giữ được tính tương thích với **v7 appcompat library**.

**Nộp ứng dụng của bạn để chấm điểm**

**Hướng dẫn cho người chấm điểm**  
Kiểm tra xem ứng dụng có các tính năng sau hay không:  
● Phương thức onCreate() đọc cài đặt deliveryPref bằng sharedPref.getString().  
● Tệp pref\_general.xml chứa một ListPreference sử dụng một mảng các lựa chọn phương thức giao hàng làm danh sách mục nhập.

● **Điểm cộng thêm:** Câu lệnh  
bindPreferenceSummaryToValue(findPreference("delivery"))  
đã được thêm vào phương thức onCreate() của lớp GeneralPreferenceFragment trong SettingsActivity để hiển thị lựa chọn phương thức giao hàng trong phần tóm tắt của tùy chọn.

**Bài 10.1 Phần A: Room, LiveData, và ViewMode**

**Giới thiệu**

A green and white hexagons

AI-generated content may be incorrect.

Hệ điều hành Android cung cấp một nền tảng vững chắc để xây dựng các ứng dụng hoạt động tốt trên nhiều thiết bị và hình thức khác nhau. Tuy nhiên, các vấn đề như vòng đời phức tạp và việc thiếu một kiến trúc ứng dụng được khuyến nghị khiến việc viết ứng dụng mạnh mẽ trở nên khó khăn. **Android Architecture Components** cung cấp các thư viện cho các tác vụ phổ biến như quản lý vòng đời và lưu trữ dữ liệu, giúp dễ dàng triển khai **kiến trúc ứng dụng được khuyến nghị**.

Các Architecture Components giúp bạn cấu trúc ứng dụng theo cách mạnh mẽ, dễ kiểm thử và dễ bảo trì hơn, đồng thời giảm bớt mã dư thừa (boilerplate code).