

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.....	19
Bài 2)	Lưu trữ dữ liệu với Room.....	31
2.1)	Room, LiveData và ViewModel.....	31
2.2)	Room, LiveData và ViewModel.....	31

3.1) Trinfh gowx loi

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

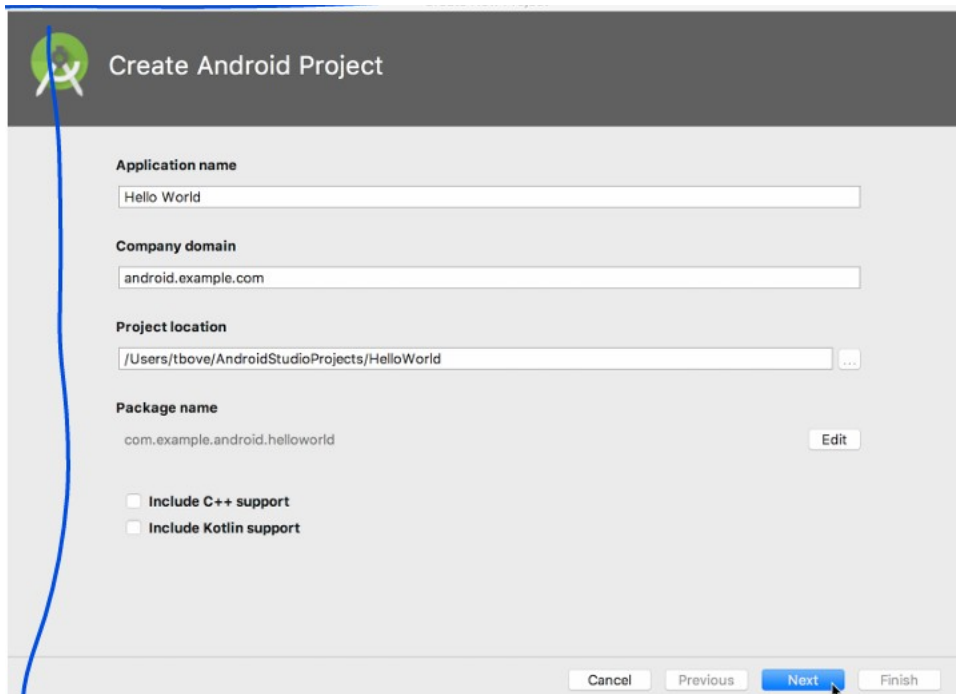
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

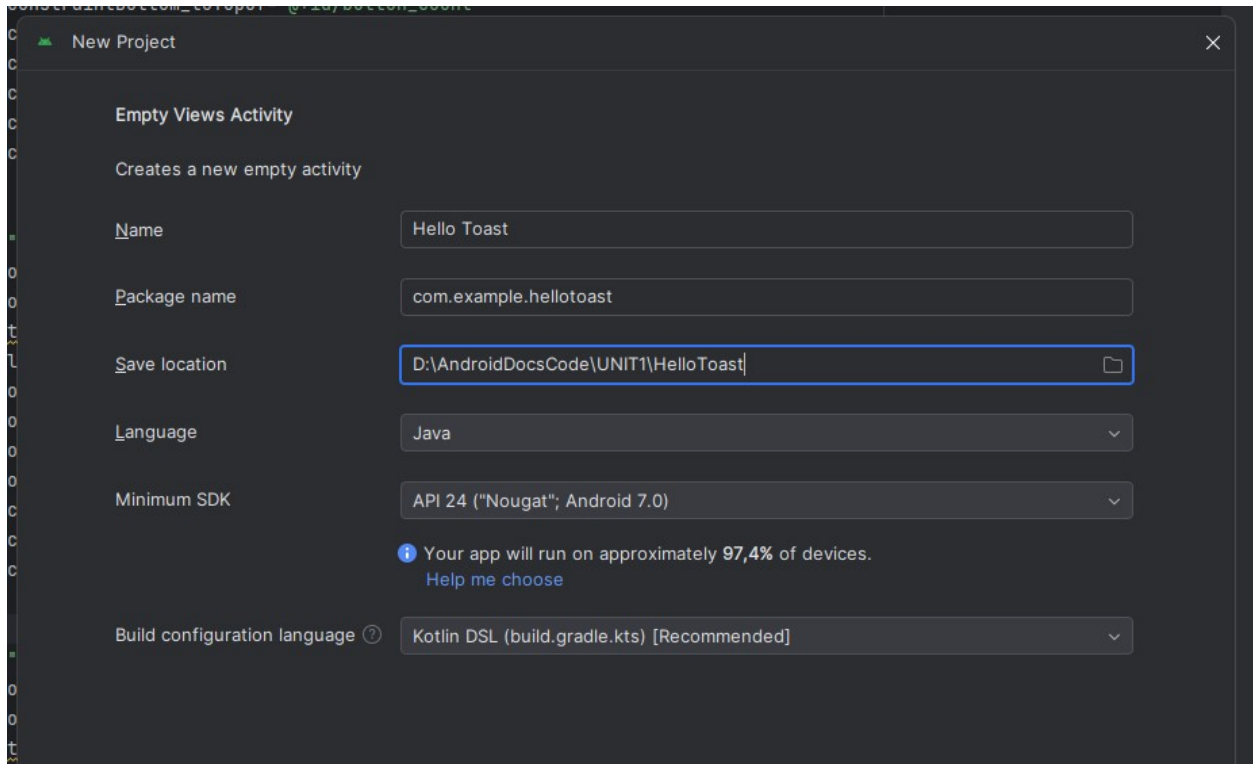
- Cách cài đặt và sử dụng IDE Android Studio.
- Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android.
- Cách tạo một dự án Android từ một mẫu.
- Cách thêm thông điệp ghi lại vào ứng dụng của bạn để phục vụ mục đích gỡ lỗi.

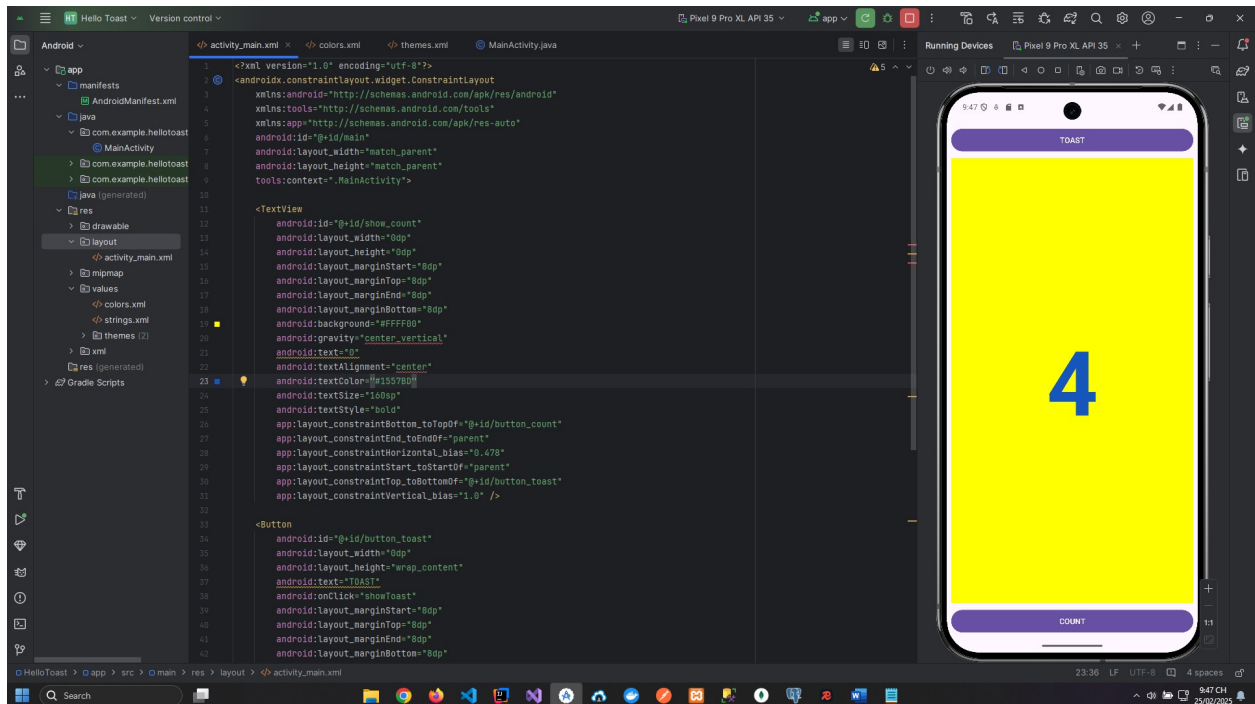
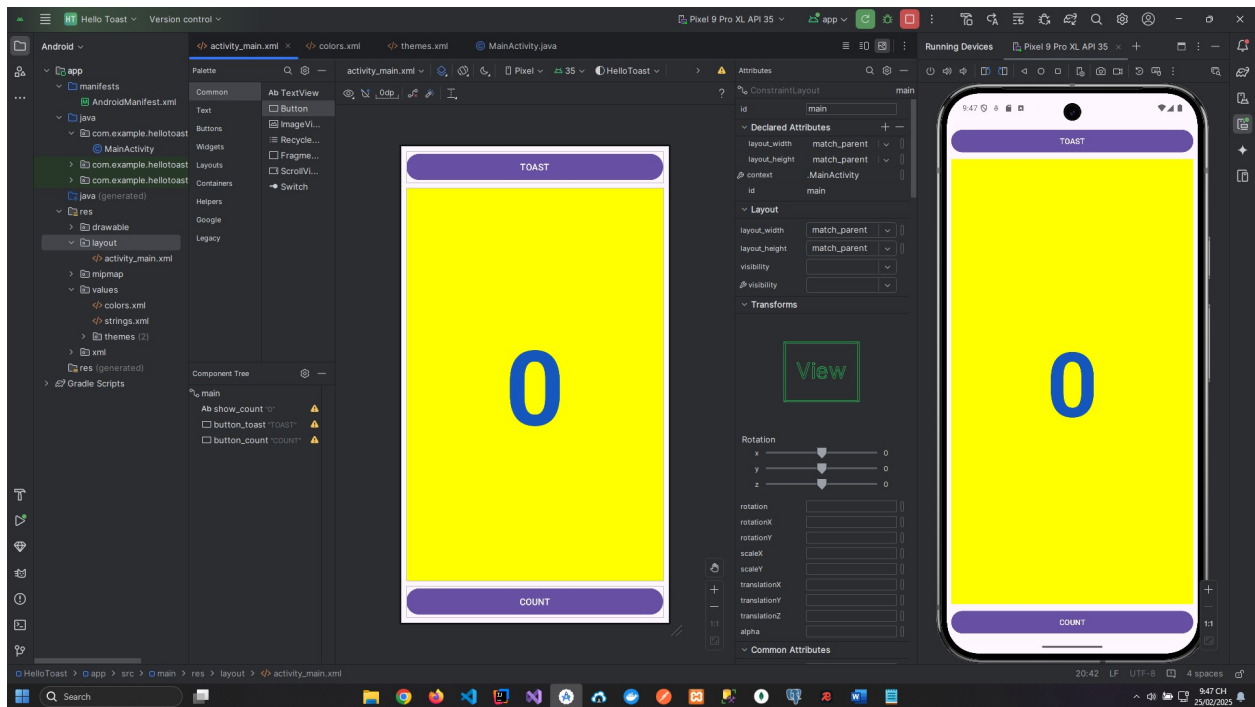
Những gì bạn sẽ làm

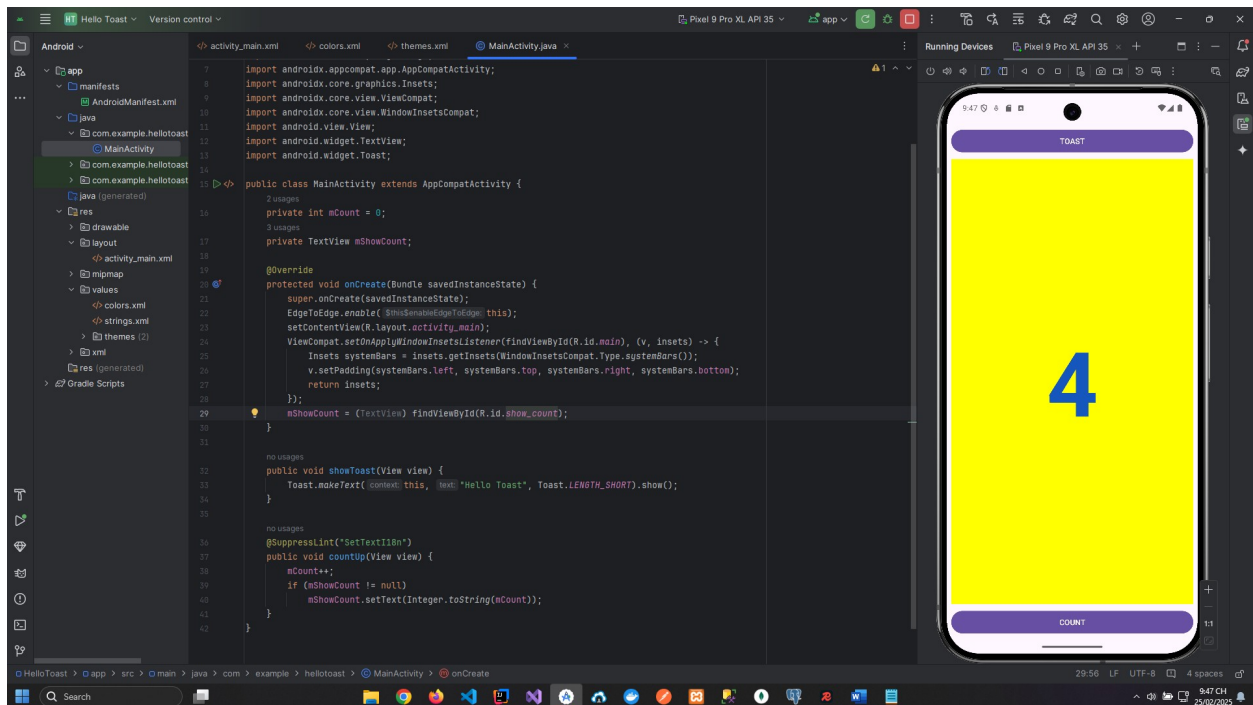
- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.

- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.
- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

1.2) Giao diện người dùng tương tác đầu tiên







1.3) Trình chỉnh sửa bố cục

1.4) Văn bản và các chế độ cuộn

1.5) Tài nguyên có sẵn

Bài 2) Activities

2.1) Activity và Intent

2.2) Vòng đời của Activity và trạng thái

2.3) Intent ngầm định

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

3.1) Trình gỡ lỗi

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

- 1.1) Hình ảnh có thể chọn
- 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

Shared preferences cho phép bạn lưu trữ một lượng nhỏ dữ liệu nguyên thủy dưới dạng các cặp key/value trong một file trên thiết bị. Để lấy một đối tượng xử lý file preference, và để đọc, ghi và quản lý dữ liệu preference, hãy sử dụng lớp SharedPreferences. Android framework tự quản lý file shared preferences. File này có thể truy cập được đối với tất cả các thành phần của ứng dụng của bạn, nhưng không thể truy cập được đối với các ứng dụng khác.

Dữ liệu bạn lưu vào shared preferences khác với dữ liệu trong saved activity state, mà bạn đã học trong một chương trước:

- Dữ liệu trong saved activity instance state được giữ lại giữa các activity instance trong cùng một phiên người dùng.
- Shared preferences tồn tại dai dẳng giữa 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 shared preferences khi bạn cần lưu một lượng nhỏ dữ liệu dưới dạng các cặp key/value đơn giản. Để quản lý lượng lớn dữ liệu ứng dụng liên tục hơn, hãy sử dụng một phương pháp lưu trữ như thư viện Room 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 text view
- Sử dụng style và theme
- Lưu và khôi phục activity instance state

Những điều bạn sẽ học

Bạn sẽ học cách:

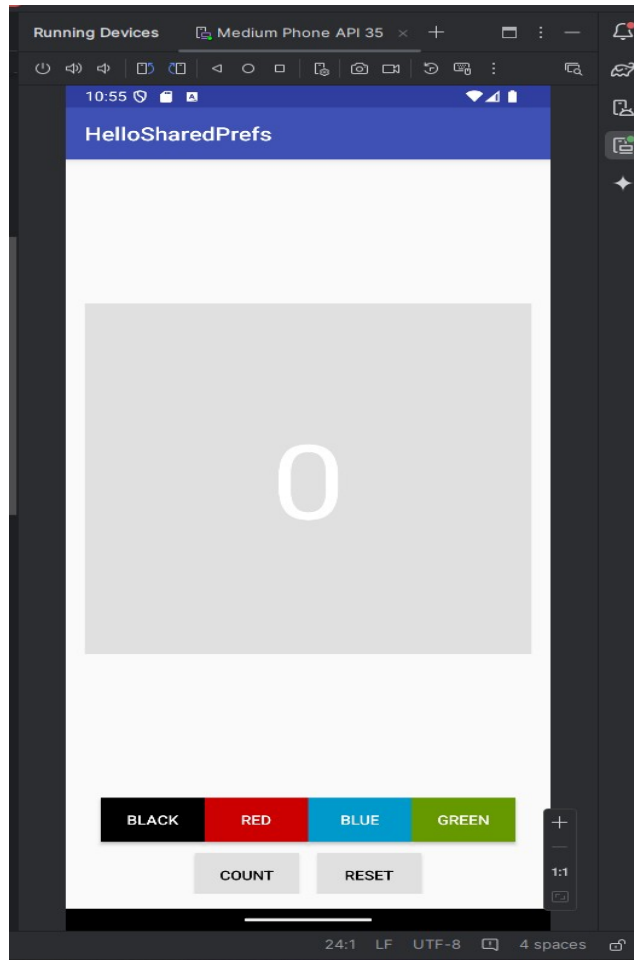
- Xác định shared preferences là gì
- Tạo một file shared preferences cho ứng dụng của bạn
- Lưu dữ liệu vào shared preferences và đọc lại các preference đó
- Xóa dữ liệu trong shared preferences

Những điều bạn sẽ làm

- Cập nhật một ứng dụng để nó có thể lưu, truy xuất và đặt lại shared preferences

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, thay đổi màu nền và đặt lại cả số và màu về mặc định của chúng. Ứng dụng này cũng sử dụng theme và style để xác định các nút.



Bạn bắt đầu với ứng dụng starter và thêm shared preferences vào mã activity chính. Bạn cũng thêm một nút reset để đặt cả số đếm và màu nền về mặc định, và xóa file preferences.

Task 1: Khám phá HelloSharedPrefs

Dự án ứng dụng starter hoàn chỉnh cho bài thực hành này có sẵn tại HelloSharedPrefs-Starter. Trong task này, 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 xuống ứng dụng HelloSharedPrefs-Starter và giải nén file
2. Mở dự án trong Android Studio
3. Xây dựng và chạy ứng dụng. Hãy thử những điều sau:
 - Nhấp vào nút **Count** để tăng số trong text view chính

- Nhấp vào bất kỳ nút màu nào để thay đổi màu nền của text view chính
 - Xoay thiết bị và lưu ý rằng cả màu nền và số đếm đều được giữ lại
 - Nhấp vào nút **Reset** để đặt màu và số đếm trở lại mặc định
4. Force-quit ứng dụng bằng một trong các phương pháp sau:
- Trong Android Studio, chọn **Run > Stop 'app'** hoặc nhấp vào biểu tượng Stop
 - Trên thiết bị, nhấn nút Recents (nút hình vuông ở góc dưới bên phải). Vuốt thẻ ứng dụng HelloSharedPreferences để thoát ứng dụng, hoặc nhấp vào X ở góc bên phải của thẻ. Nếu bạn thoát ứng dụng theo cách này, hãy đợi một vài giây trước khi khởi động lại để hệ thống có thể dọn dẹp
5. Chạy lại ứng dụng

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

1.2 Khám phá mã Activity

1. Mở MainActivity
2. Kiểm tra mã và lưu ý những điều sau:
 - Số đếm (mCount) được định nghĩa là một số nguyên. Phương thức `onClick countUp()` tăng giá trị này và cập nhật TextView chính
 - Màu (mColor) cũng là một số nguyên ban đầu được định nghĩa là màu xám trong file tài nguyên `colors.xml` dưới dạng `default_background`
 - Phương thức `onClick changeBackground()` lấy màu nền của nút đã được nhấp và sau đó đặt text view chính thành màu đó
 - Cả số nguyên mCount và mColor đều được lưu vào bundle instance state trong `onSaveInstanceState()` và được khôi phục trong `onCreate()`. Các key bundle cho số đếm và màu được định nghĩa bởi các biến private (`COUNT_KEY`) và (`COLOR_KEY`)

Task 2: Lưu và khôi phục dữ liệu vào một file shared preferences

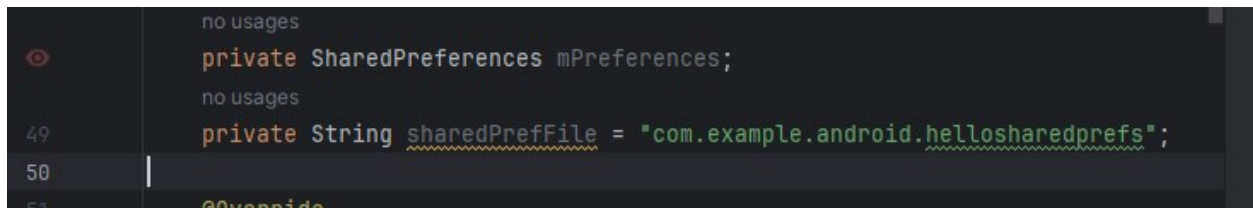
Trong task này, bạn lưu trạng thái của ứng dụng vào một file shared preferences và đọc lại dữ liệu đó khi ứng dụng được khởi động lại. Vì dữ liệu trạng thái mà bạn lưu vào shared preferences (số đếm và màu hiện tại) là **cùng** một dữ liệu mà bạn giữ lại trong instance state, bạn không phải thực hiện nó hai lần. Bạn có thể thay thế hoàn toàn instance state bằng shared preference state.

2.1 Khởi tạo preferences

1. Thêm các biến thành viên vào lớp MainActivity để giữ tên của file shared preferences và một tham chiếu đến đối tượng SharedPreferences.

```
private SharedPreferences mPreferences;
```

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



```
no usages
private SharedPreferences mPreferences;
no usages
49 private String sharedPrefFile = "com.example.android.hellosharedprefs";
50
51 @Override
```

Bạn có thể đặt tên file shared preferences của bạn bất cứ điều gì bạn muốn, nhưng theo quy ước, nó có cùng tên với tên package của 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 activity Context) mở file tại tên file 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 file shared preferences có thể đọc hoặc ghi trên toàn thế giới. Các chế độ này đã

bị deprecated trong API 17 và hiện không được khuyến khích 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.

```
private final String COLOR_KEY = "color";
1 usage
private SharedPreferences mPreferences;
1 usage
private String sharedPrefFile = "com.example.android.hellosharedprefs";

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

    // Initialize views, color
    mShowCountTextView = findViewById(R.id.count_textview);
    mColor = ContextCompat.getColor(context: this,
        R.color.default_background);

    mPreferences = getSharedPreferences(sharedPrefFile, MODE_PRIVATE);

    // Restore the saved instance state.
    if (savedInstanceState != null) {

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

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

2.2 Lưu preferences trong onPause()

Lưu preferences rất giống với việc lưu instance state -- cả hai thao tác đều đặt dữ liệu sang một đối tượng Bundle dưới dạng một cặp key/value. Tuy nhiên, đối với shared preferences, bạn lưu dữ liệu đó trong callback lifecycle

onPause(), và bạn cần một đối tượng shared editor (SharedPreferences.Editor) để ghi vào đối tượng shared preferences.

1. Thêm phương thức lifecycle onPause() vào MainActivity.

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

2. Trong onPause(), lấy một editor cho đối tượng SharedPreferences:

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

Cần có trình soạn thảo tùy chọn chia sẻ để ghi vào đối tượng tùy chọn chia sẻ. Thêm dòng này vào onPause() sau khi gọi super.onPause().

3. Sử dụng phương thức putInt() để đặt cả số nguyên mCount và mColor vào shared preferences với các key thích 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 preferences:

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

Phương thức apply() lưu preferences không đồng bộ, ngoài UI thread. Shared preferences editor cũng có một phương thức commit() để lưu preferences đồng bộ. Phương thức commit() không được khuyến khích vì nó có thể chặn các thao tác khác.

1. Xóa toàn bộ phương thức `onSaveInstanceState()`. Vì activity instance state chứa cùng dữ liệu như shared preferences, bạn có thể thay thế hoàn toàn instance state.

```
@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 preferences trong `onCreate()`

Như với instance state, ứng dụng của bạn đọc bất kỳ shared preferences đã lưu nào trong phương thức `onCreate()`. Một lần nữa, vì shared preferences chứa cùng dữ liệu như instance state, chúng ta có thể thay thế state bằng preferences ở đây. Mỗi khi `onCreate()` được gọi -- khi ứng dụng khởi động, trên các thay đổi cấu hình -- shared preferences được sử dụng để khôi phục state của view.

1. Xác định vị trí phần của phương thức `onCreate()` kiểm tra xem đối số `savedInstanceState` có null hay không và khôi phục instance state:

```
if (savedInstanceState != null) {
    mCount = savedInstanceState.getInt(COUNT_KEY);
    if (mCount != 0) {
        mShowCountTextView.setText(String.format("%s", mCount));
    }
    mColor = savedInstanceState.getInt(COLOR_KEY);
}
```

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

```
}
```

2. Xóa toàn bộ khối đó.
3. Trong phương thức onCreate(), ở cùng vị trí nơi có mã instance state, lấy số đếm từ preferences bằng key COUNT_KEY và gán nó cho biến mCount.

```
mCount = mPreferences.getInt(COUNT_KEY, 0);
```

Khi bạn đọc dữ liệu từ preferences, bạn không cần phải lấy một shared preferences editor. Sử dụng bất kỳ phương thức "get" nào trên một đối tượng shared preferences (chẳng hạn như getInt() hoặc getString()) để truy xuất dữ liệu preference.

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

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

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

5. Lấy màu từ preferences bằng key COLOR_KEY và gán nó cho biến mColor.

```
mColor = mPreferences.getInt(COLOR_KEY, mColor);
```

Như trước, đối số thứ hai cho getInt() là giá trị mặc định để sử dụng trong trường hợp key không tồn tại trong shared preferences. Trong trường hợp này, bạn có thể chỉ cần sử dụng lại giá trị của mColor, vừa được khởi tạo thành màu nền mặc định ở phía trên phương thức.

6. Cập nhật màu nền của text view chính.

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

7. Chạy ứng dụng. Nhấp vào nút Count và thay đổi màu nền để cập nhật instance state và preferences.
8. Xoay thiết bị hoặc trình giả lập để xác minh rằng số đếm và màu được lưu trên các thay đổi cấu hình.
9. Force-quit ứng dụng bằng một trong các phương pháp sau:
 - Trong Android Studio, chọn Run > Stop 'app'.
 - Trên thiết bị, nhấn nút Recents (nút hình vuông ở góc dưới bên phải). Vuốt thẻ ứng dụng HelloSharedPreferences để thoát ứng dụng, hoặc nhấp vào X ở góc bên phải của thẻ.
10. Chạy lại ứng dụng. Ứng dụng khởi động lại và tải preferences, duy trì trạng thái.

Code cho phương thức onCreate() của MainActivity:

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

    // Initialize views, color
    mShowCountTextView = findViewById(R.id.count_textview);
    mColor = ContextCompat.getColor(context: this,
        R.color.default_background);

    mPreferences = getSharedPreferences(sharedPrefFile, MODE_PRIVATE);

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

2.4 Đặt lại preferences trong trình xử lý nhấp reset()

Nút reset trong ứng dụng starter đặt lại cả số đếm và màu cho activity về các giá trị mặc định của chúng. Vì preferences giữ trạng thái của activity, điều quan trọng là phải xóa preferences cùng lúc.

1. Trong phương thức onClick reset(), sau khi màu và số đếm được đặt lại, lấy một editor cho đối tượng SharedPreferences:

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

2. Xóa tất cả shared preferences:

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

1. Áp dụng các thay đổi:

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

Code cho phương thức reset():

```
1 usage
public void reset(View view) {
    // Reset count
    mCount = 0;
    mShowCountTextView.setText(String.format("%s", mCount));

    // Reset color
    mColor = ContextCompat.getColor(context: this,
        R.color.default_background);
    mShowCountTextView.setBackgroundColor(mColor);
    // Clear preferences
    SharedPreferences.Editor preferencesEditor = mPreferences.edit();
    preferencesEditor.clear();
    preferencesEditor.apply();
}
```

Tóm tắt

- Lớp SharedPreferences cho phép một ứng dụng lưu trữ một lượng nhỏ dữ liệu nguyên thủy dưới dạng các cặp key-value.
- Shared preferences tồn tại dai dẳng giữa các phiên người dùng khác nhau của cùng một ứng dụng.

- Để ghi vào shared preferences, hãy lấy một đối tượng SharedPreferences.Editor.
- Sử dụng các phương thức "put" khác nhau trong một đối tượng SharedPreferences.Editor, chẳng hạn như putInt() hoặc putString(), để đặt dữ liệu vào shared preferences với một key và một value.
- Sử dụng các phương thức "get" khác nhau trong một đối tượng SharedPreferences, chẳng hạn như getInt() hoặc getString(), để lấy dữ liệu ra khỏi shared preferences bằng một key.
- Sử dụng phương thức clear() trong một đối tượng SharedPreferences.Editor để xóa tất cả dữ liệu được lưu trữ trong preferences.
- Sử dụng phương thức apply() trong một đối tượng SharedPreferences.Editor để lưu các thay đổi vào file preferences.

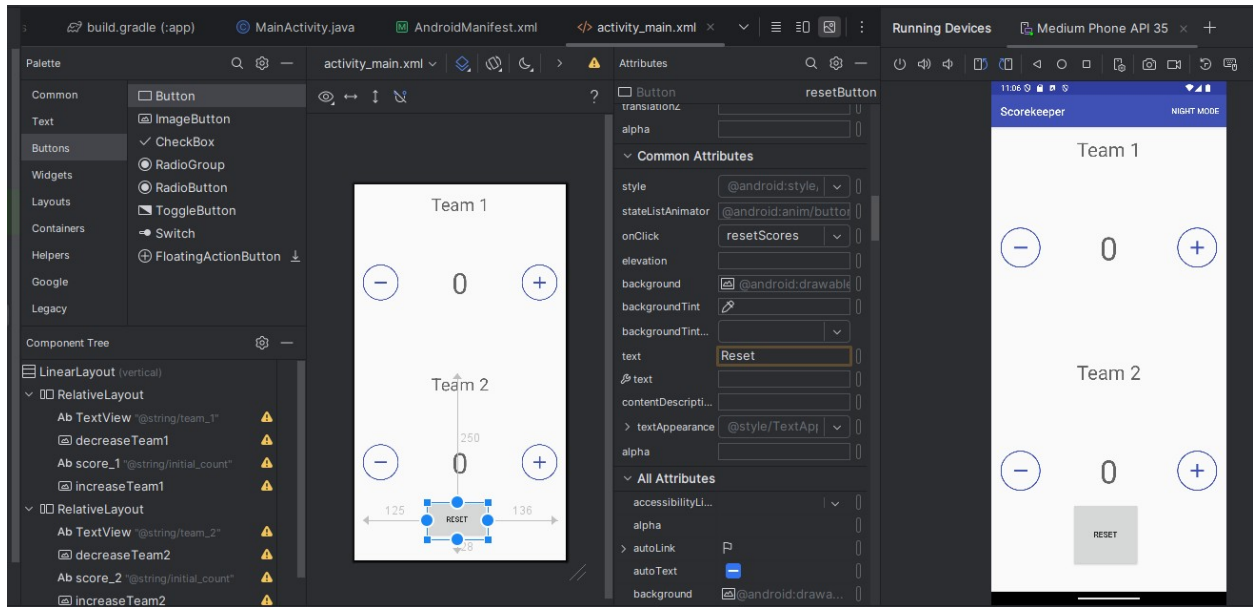
Bài tập về nhà

Xây dựng và chạy một ứng dụng

Mở ứng dụng ScoreKeeper mà bạn đã tạo trong **bài Android fundamentals**

5.1: Drawables, styles, and themes.

2. Thay thế saved instance state bằng shared preferences cho mỗi điểm số.
3. Để kiểm tra ứng dụng, hãy xoay thiết bị để đảm bảo rằng các thay đổi cấu hình đọc các shared preferences đã lưu và cập nhật UI.
4. Dừng ứng dụng và khởi động lại để đảm bảo rằng các preferences đã được lưu.
5. Thêm một nút **Reset** để đặt lại các giá trị điểm về 0 và xóa shared preferences.



```
private SharedPreferences sharedPreferences;
```

```
1 usage
```

```
private static final String PREFS_NAME = "ScoreKeeperPrefs";
```

```
2 usages
```

```
private static final String SCORE_TEAM_1 = "score1";
```

```
2 usages
```

```
private static final String SCORE_TEAM_2 = "score2";
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
    mScoreText1 = findViewById(R.id.score_1);
```



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

```
    sharedPreferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
```

```
    mScore1 = sharedPreferences.getInt(SCORE_TEAM_1, 0);
```

```
    mScore2 = sharedPreferences.getInt(SCORE_TEAM_2, 0);
```

```
    updateScoreUI();
```

```
}
```

```
4 usages
```

```
@Override
```

```
protected void onPause() {
```

```
    super.onPause();
```

```
    SharedPreferences.Editor editor = sharedPreferences.edit();
```

```
    editor.putInt(SCORE_TEAM_1, mScore1);
```

```
    editor.putInt(SCORE_TEAM_2, mScore2);
```

```
    editor.apply();
```

```
}
```



```

4 usages
private void updateScoreUI() {
    mScoreText1.setText(String.valueOf(mScore1));
    mScoreText2.setText(String.valueOf(mScore2));
}

2 usages
public void decreaseScore(View view) {
    if (view.getId() == R.id.decreaseTeam1) {
        mScore1--;
    } else if (view.getId() == R.id.decreaseTeam2) {...}
    updateScoreUI();
}

2 usages
public void increaseScore(View view) {
    if (view.getId() == R.id.increaseTeam1) {
        mScore1++;
    } else if (view.getId() == R.id.increaseTeam2) {
        mScore2++;
    }
    updateScoreUI();
}

1 usage
public void resetScores(View view) {
    mScore1 = 0;
    mScore2 = 0;
    sharedPreferences.edit().clear().apply();
    updateScoreUI();
}

```

1.2) Cài đặt ứng dụng

Giới thiệu

Các 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 của họ, các đơn vị mặc định cho các phép đo và các cài

đặt khác áp dụng cho toàn bộ ứng dụng. Người dùng không truy cập cài đặt thường xuyên, bởi vì một khi người dùng thay đổi một cài đặt, chẳng hạn như vị trí nhà, họ hiếm khi cần quay lại và thay đổi lại.

Người dùng mong đợi điều hướng đến cài đặt ứng dụng bằng cách nhấn vào **Settings** trong điều hướng bên, chẳng hạn như navigation drawer như được hiển thị ở phía bên trái của hình bên dưới, hoặc trong menu tùy chọn trong app bar, được hiển thị ở phía bên phải của hình bên dưới.

Trong hình trên:

1. Cài đặt trong điều hướng bên (navigation drawer)
2. Cài đặt trong menu tùy chọn của app bar

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

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo một dự án Android Studio từ một template và tạo bố cục chính.
- Chạy ứng dụng trên trình giả lập hoặc thiết bị đã kết nối.
- Tạo và chỉnh sửa các phần tử UI bằng trình chỉnh sửa bố cục và mã XML.
- Trích xuất tài nguyên chuỗi và chỉnh sửa các giá trị chuỗi.
- Truy cập các phần tử UI từ mã của bạn bằng findViewById().
- Xử lý một nhấp chuột Button.
- Hiển thị một tin nhắn Toast.
- Thêm một Activity vào một ứng dụng.
- Tạo một menu tùy chọn trong app bar.
- Thêm và chỉnh sửa các mục menu trong menu tùy chọn.
- Sử dụng style và theme trong một dự án.
- Sử dụng SharedPreferences.

Những điều bạn sẽ học

Bạn sẽ học cách:

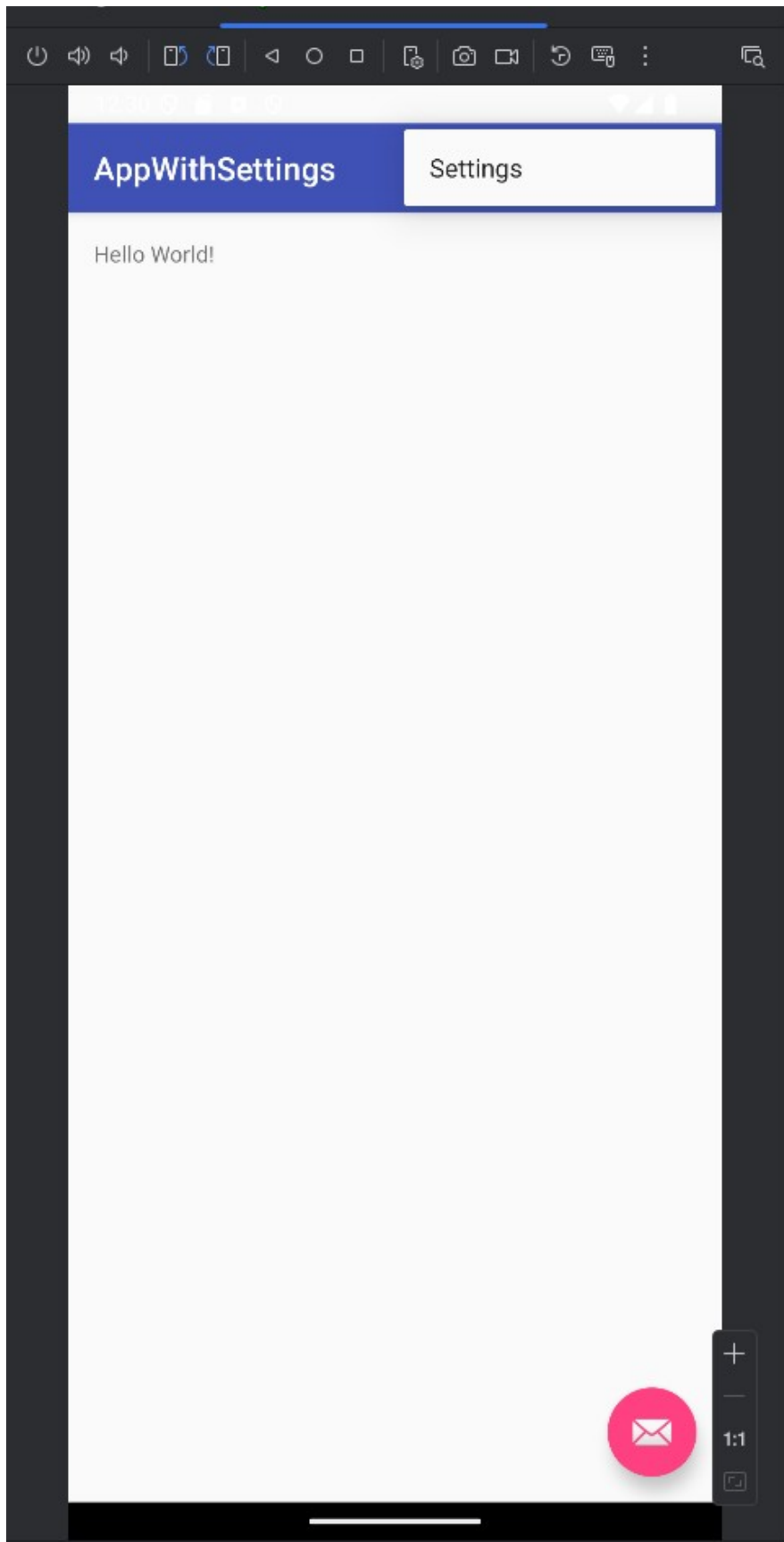
1. Thêm một Fragment để quản lý cài đặt.
2. Tạo một file tài nguyên XML của các cài đặt với các thuộc tính của chúng.
3. Tạo điều hướng đến Activity cài đặt.
4. Đặt các giá trị mặc định của cài đặt.
5. Đọc các giá trị cài đặt được thay đổi bởi người dùng.
6. Tùy chỉnh template Activity cài đặt.

Những điều bạn sẽ làm

- Tạo một ứng dụng bao gồm Settings trong menu tùy chọn.
- Thêm một switch toggle 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 template Activity cài đặt tiêu chuẩn được cung cấp bởi Android Studio.

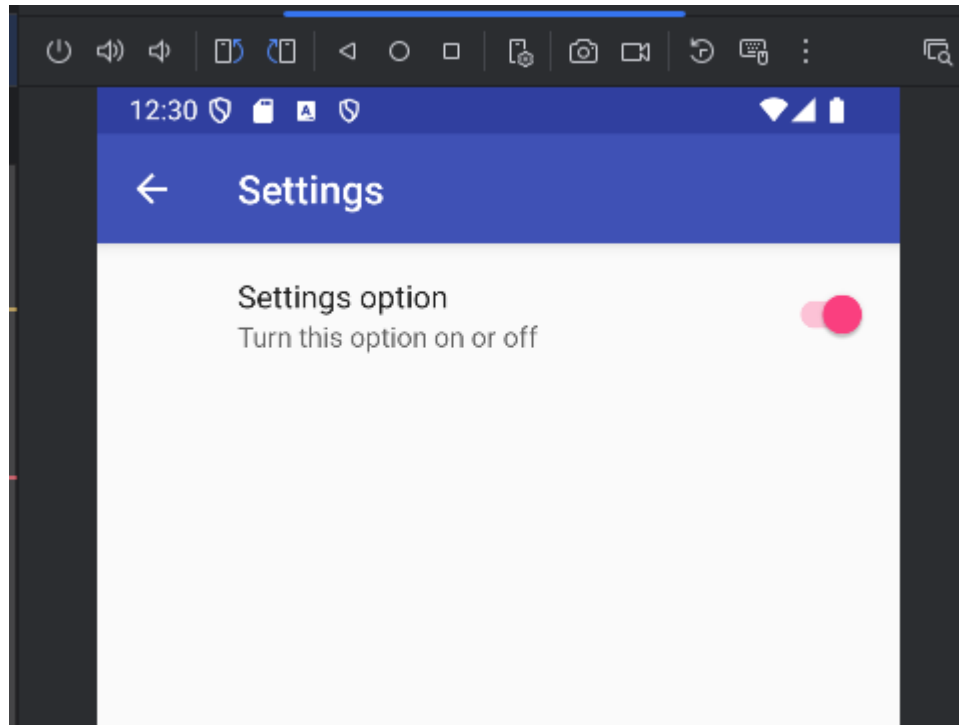
Tổng quan về ứng dụng

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



Template cũng bao gồm một floating action button ở góc dưới bên phải của màn hình với một biểu tượng phong bì. Bạn có thể bỏ qua nút này cho 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 template Basic Activity, và bạn sẽ thêm một Activity cài đặt cung cấp một cài đặt switch toggle mà người dùng có thể bật hoặc tắt:

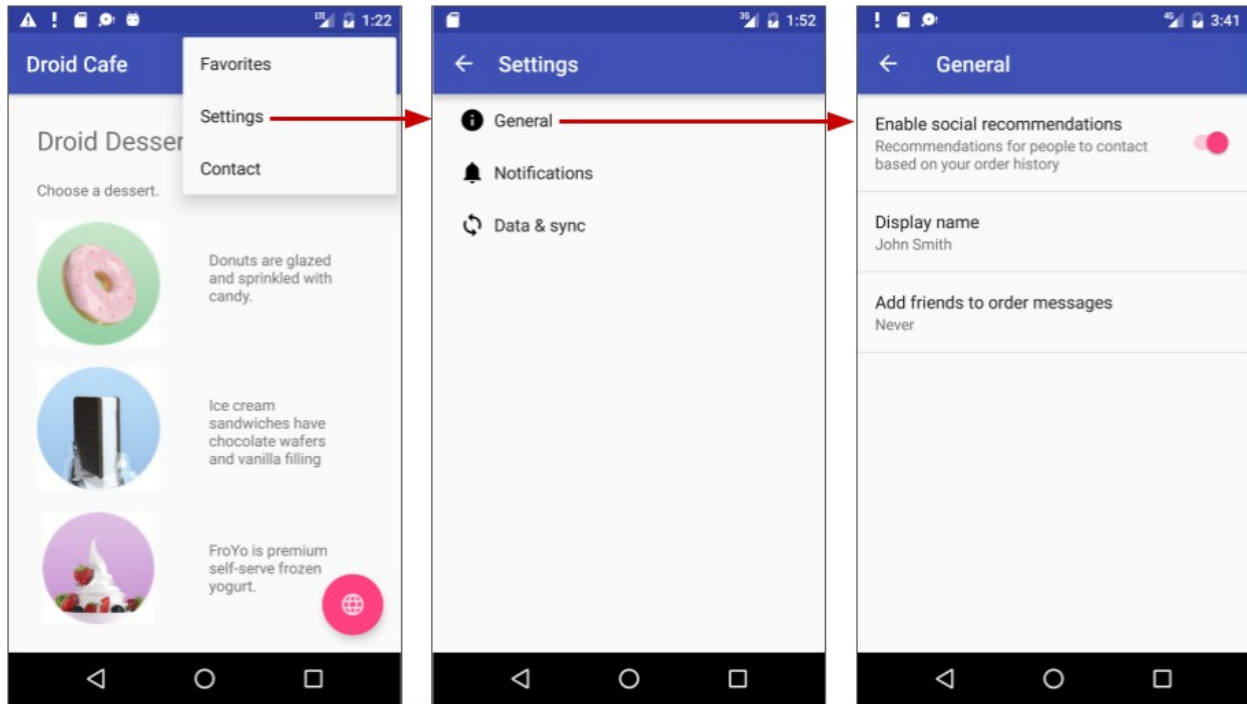


Bạn sẽ thêm mã để đọc cài đặt và thực hiện một hành động dựa trên giá trị của nó. Vì sự đơn giản, hành động sẽ là hiển thị một tin nhắn Toast với giá trị của cài đặt.

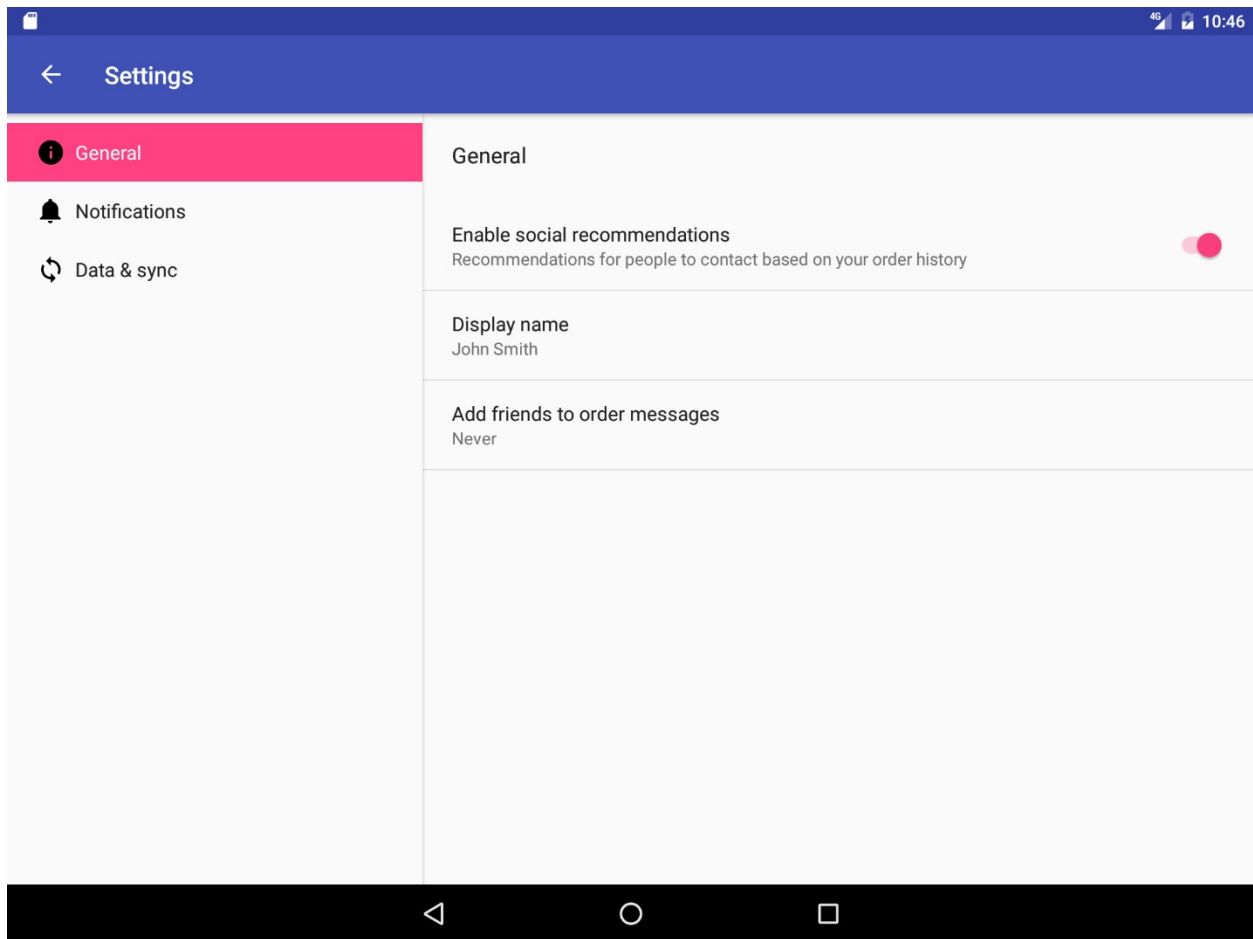
Trong task thứ hai, bạn sẽ thêm template Activity cài đặt tiêu chuẩn được cung cấp bởi Android Studio vào ứng dụng DroidCafeOptionsUp mà bạn đã tạo trong một bài học trước.

Template Activity cài đặt được điền sẵn với các cài đặt bạn có thể tùy chỉnh cho một ứng dụng và cung cấp một bố cục khác nhau cho điện thoại và máy tính bảng:

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



- **Máy tính bảng:** Một bố cục màn hình master/detail với một liên kết tiêu đề cho mỗi nhóm ở phía bên trái (master) và nhóm cài đặt ở phía bên phải (detail), như được hiển thị trong hình bên dưới.



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

Ứng dụng DroidCafeOptionsUp đã được tạo trong một bài học trước từ template Basic Activity, cung cấp một menu tùy chọn trong app bar để đặt tùy chọn Settings. Bạn sẽ tùy chỉnh template Activity cài đặt được cung cấp bằng cách thay đổi tiêu đề, mô tả, giá trị và giá trị mặc định của một cài đặt duy nhất. Bạn sẽ thêm mã để đọc giá trị của cài đặt sau khi người dùng thay đổi nó và hiển thị giá trị đó.

Task 1: Thêm một cài đặt switch vào một ứng dụng

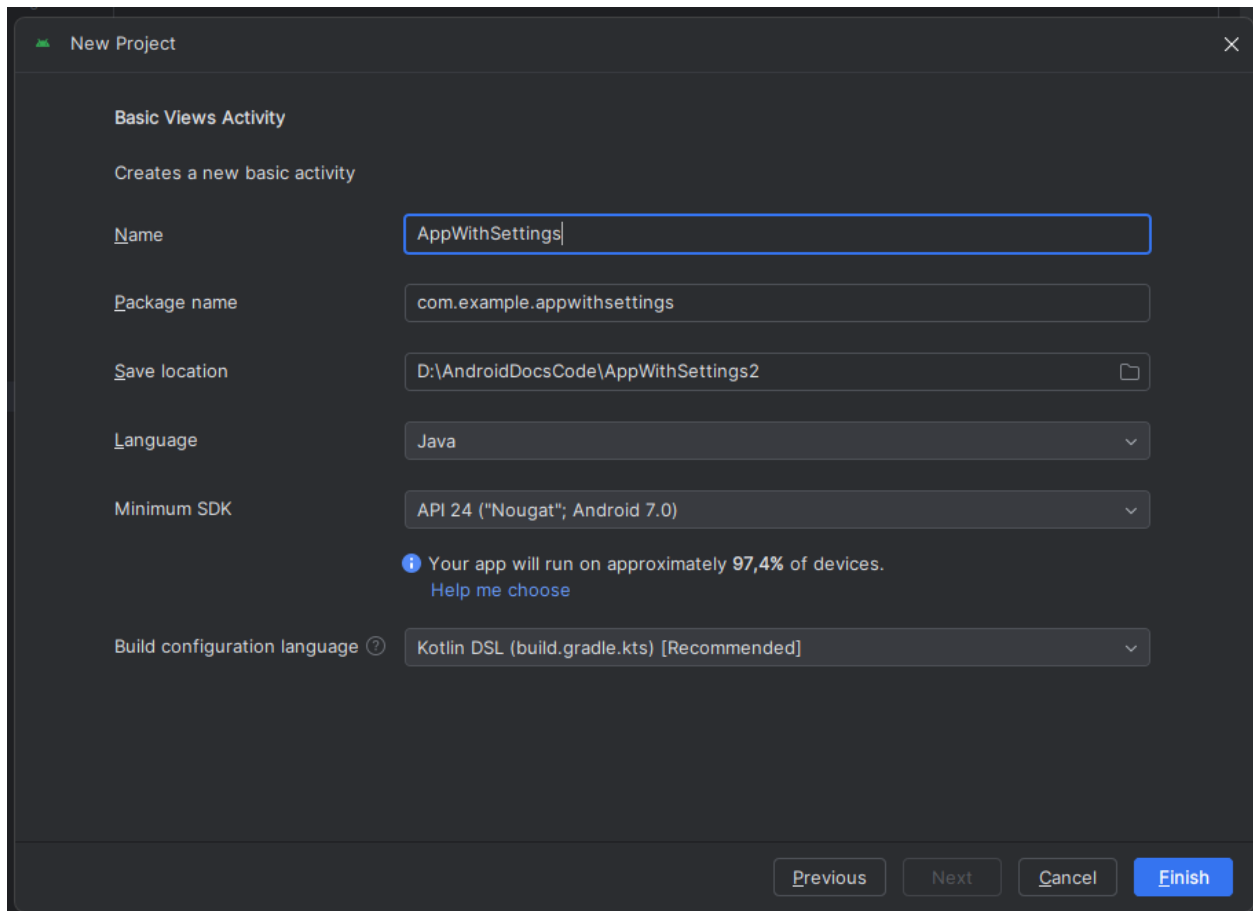
Trong task này, bạn thực hiện những điều sau:

- Tạo một dự án mới dựa trên template Basic Activity, cung cấp một menu tùy chọn.

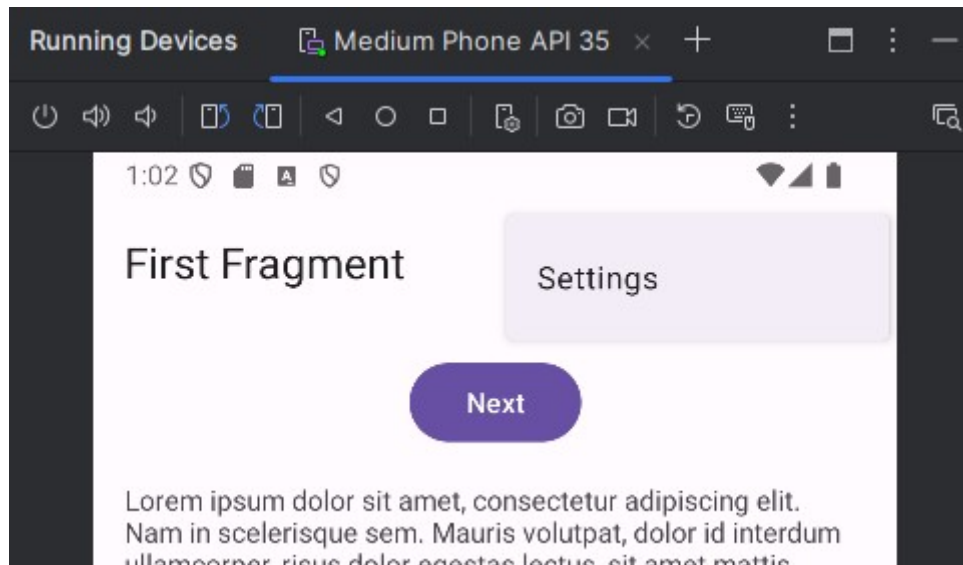
- Thêm một switch toggle (SwitchPreference) với các thuộc tính trong một file XML preference.
- Thêm một activity cho cài đặt và một fragment cho một cài đặt cụ thể. Để duy trì khả năng tương thích với AppCompatActivity, bạn sử dụng PreferenceFragmentCompat thay vì PreferenceFragment. Bạn cũng thêm thư viện android.support.v7.preference.
- Kết nối mục Settings trong menu tùy chọn với activity cài đặt.

1.1 Tạo dự án và thêm thư mục xml và file tài nguyên

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



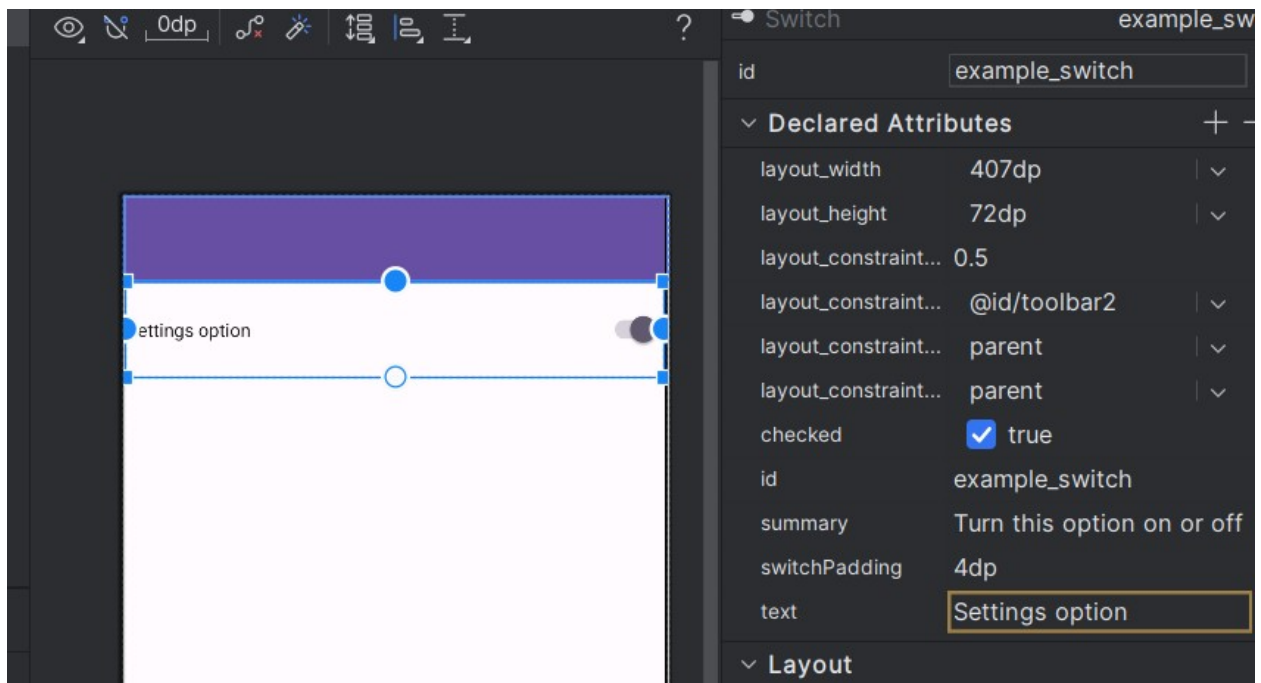
2. Chạy ứng dụng và nhấn vào biểu tượng overflow trong app bar để xem menu tùy chọn, như được hiển thị trong hình bên dưới. Mục duy nhất trong menu tùy chọn là **Settings**.



3. Bạn cần tạo một thư mục tài nguyên mới để giữ file XML chứa các cài đặt. Chọn thư mục res trong ngăn Project > Android, và chọn File > New > Android Resource Directory. Hộp thoại New Resource Directory xuất hiện.
4. Trong menu thả xuống Resource type, chọn xml. Directory name tự động thay đổi thành xml. Nhấp vào OK.
5. Thư mục xml xuất hiện trong ngăn 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).
6. Nhập tên của file XML, preferences, vào trường File name và nhấp vào OK. File preferences.xml xuất hiện bên trong thư mục xml và trình chỉnh sửa bố cục xuất hiện, như được hiển thị trong hình bên dưới.

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

Kéo một SwitchPreference từ bảng Palette ở phía bên trái lên phía trên cùng của bố cục, như được hiển thị trong hình bên dưới.



1.3 Thêm Activity cho cài đặt

Lưu và tải dữ liệu trong SettingsActivity

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_settings);

    // Tìm Toolbar trong layout và đặt làm ActionBar
    Toolbar toolbar = findViewById(R.id.toolbar2);
    setSupportActionBar(toolbar);

    // Hiển thị nút quay lại (Up button)
    Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);

    // Lấy Switch từ layout
    switchNotifications = findViewById(R.id.example_switch);

    // Lấy SharedPreferences
    sharedPreferences = getSharedPreferences("AppSettings", MODE_PRIVATE);

    // Tải giá trị đã lưu
    loadSettings();

    // Xử lý khi người dùng thay đổi trạng thái Switch
    switchNotifications.setOnCheckedChangeListener((buttonView, isChecked) -> saveSettings());

    // Xử lý padding để hỗ trợ EdgeToEdge
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
    });
}

```

1.4 Cập nhật MainActivity cho cài đặt

```

@Override
protected void onResume() {
    super.onResume();
    updateSettings();
}

1 usage
private void updateSettings() {
    // Lấy SharedPreferences
    SharedPreferences sharedPreferences = getSharedPreferences( name: "AppSettings", MODE_
    boolean isNotificationsEnabled = sharedPreferences.getBoolean( key: "notifications", d

    // Hiển thị trạng thái mới nhất của Switch trong Snackbar
    Snackbar.make(binding.getRoot(),
        text: "Notifications: " + (isNotificationsEnabled ? "Enabled" : "Disabled"),
        Snackbar.LENGTH_LONG).show();
}

```

Task 2: Sử dụng mẫu Activity Settings

Nếu bạn cần xây dựng một số màn hình con của cài đặt và bạn muốn tận dụng lợi thế của màn hình kích thước máy tính bảng cũng như duy trì khả năng tương thích với các phiên bản Android cũ hơn cho máy tính bảng, Android Studio cung cấp một lối tắt: mẫu Activity Settings.

Trong task trước đó, bạn đã học cách sử dụng một settings Activity trống và một Fragment trống để thêm một cài đặt vào một ứng dụng. Task 2 bây giờ sẽ cho bạn thấy cách sử dụng mẫu Settings Activity được cung cấp bởi Android Studio để:

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

2.1 Khám phá mẫu Activity Settings

Bài 2) Lưu trữ dữ liệu với Room

2.1) Room, LiveData và ViewModel

2.2) Room, LiveData và ViewModel