

**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 .....	18
1.3)	Broadcast receivers .....	20
Bài 2)	Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền .....	20
2.1)	Thông báo .....	20
2.2)	Trình quản lý cảnh báo .....	20
2.3)	JobScheduler .....	20
CHƯƠNG 4. Lưu dữ liệu người dùng .....		20
Bài 1)	Tùy chọn và cài đặt.....	20
1.1)	Shared preferences .....	20
1.2)	Cài đặt ứng dụng.....	20
Bài 2)	Lưu trữ dữ liệu với Room .....	20
2.1)	Room, LiveData và ViewModel.....	20
2.2)	Room, LiveData và ViewModel.....	20
3.1)	Trình gowx loi .....	

## CHƯƠNG 1. LÀM QUEN

### Bài 1) Tạo ứng dụng đầu tiên

#### 1.1) Android Studio và Hello World

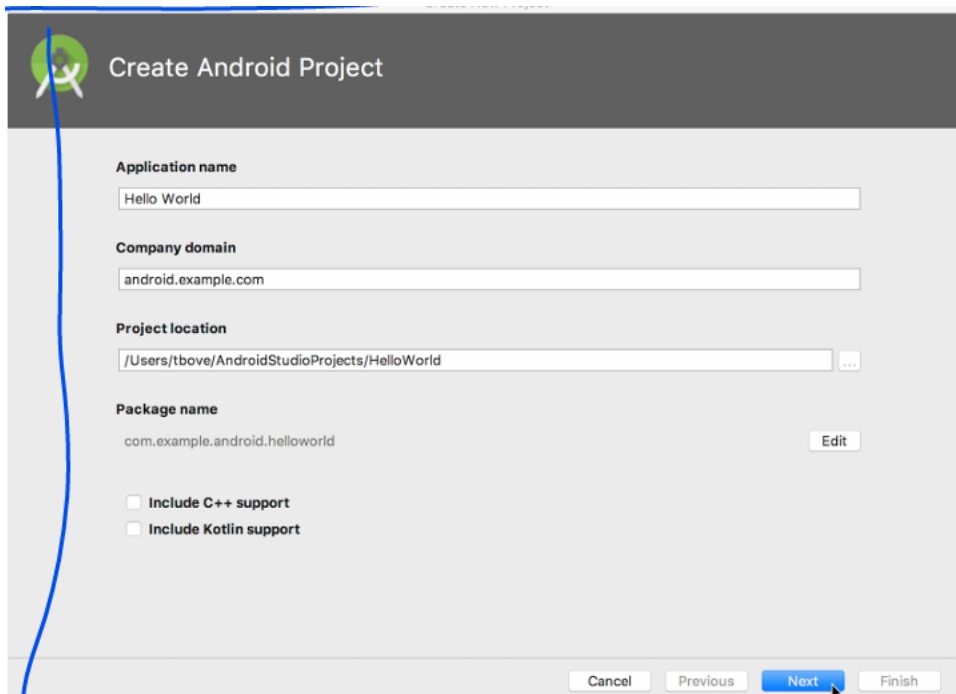
### Giới thiệu

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

## Những gì Bạn nên biết

Bạn nên có khả năng:

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



## Những gì Bạn sẽ cần:

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

## Những gì bạn sẽ học

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

## **Những gì bạn sẽ làm**

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

### **1.2) Giao diện người dùng tương tác đầu tiên**

### **1.3) Trình chỉnh sửa bố cục**

### **1.4) Văn bản và các chế độ cuộn**

### **1.5) Tài nguyên có sẵn**

## **Bài 2) Activities**

### **2.1) Activity và Intent**

### **2.2) Vòng đời của Activity và trạng thái**

### **2.3) Intent ngầm định**

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

### **3.1) Trình gỡ lỗi**

### **3.2) Kiểm thử đơn vị**

### **3.3) Thư viện hỗ trợ**

## CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

### Bài 1) Tương tác người dùng

- 1.1) Hình ảnh có thể chọn
- 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

### Giới thiệu

Luồng (thread) là một đường dẫn thực thi độc lập trong một chương trình đang chạy. Khi một chương trình Android được khởi chạy, hệ thống sẽ tạo một luồng chính, còn được gọi là luồng UI. Luồng UI này là cách ứng dụng của bạn tương tác với các thành phần từ UI Android Toolkit.

Tuy nhiên, đôi khi một ứng dụng cần thực hiện các tác vụ tốn nhiều tài nguyên chẳng hạn như tải xuống tệp, thực hiện truy vấn cơ sở dữ liệu, phát phương tiện hoặc tính toán phân tích phức tạp. Những công việc tiêu tốn nhiều tài nguyên này có thể làm

tắc nghẽn luồng UI khiến ứng dụng không phản hồi với thao tác của người dùng hoặc không thể cập nhật giao diện. Người dùng có thể cảm thấy khó chịu và gỡ cài đặt ứng dụng của bạn.

Để đảm bảo cho trải nghiệm người dùng (UX) mượt mà, Android framework cung cấp một lớp hỗ trợ có tên AsyncTask, giúp xử lý các tác vụ nặng ngoài luồng UI. Bằng cách chuyển các công việc này sang một luồng riêng biệt, ứng dụng vẫn có thể duy trì khả năng phản hồi, tránh tình trạng “đơ” hay chậm trễ khi người dùng tương tác.

Vì luồng riêng biệt không được đồng bộ hóa với luồng gọi, nên nó được gọi là luồng không đồng bộ (asynchronous thread). AsyncTask cũng cung cấp các phương thức gọi lại (callback), cho phép bạn trả kết quả xử lý về luồng UI.

Trong phần thực hành này, bạn sẽ tìm hiểu cách tích hợp tác vụ nền vào ứng dụng Android bằng AsyncTask.

## **Những kiến thức bạn cần biết trước**

Bạn cần có thể:

- Tạo một Activity.
- Thêm một TextView vào bố cục của Activity.
- Lấy id của TextView bằng mã lập trình và thiết lập nội dung cho nó.
- Sử dụng Button và xử lý sự kiện onClick của nó.

## **Những gì bạn sẽ tìm hiểu:**

- Cách thêm AsyncTask vào ứng dụng để chạy một tác vụ trong nền.
- Những hạn chế của AsyncTask khi sử dụng cho các tác vụ nền

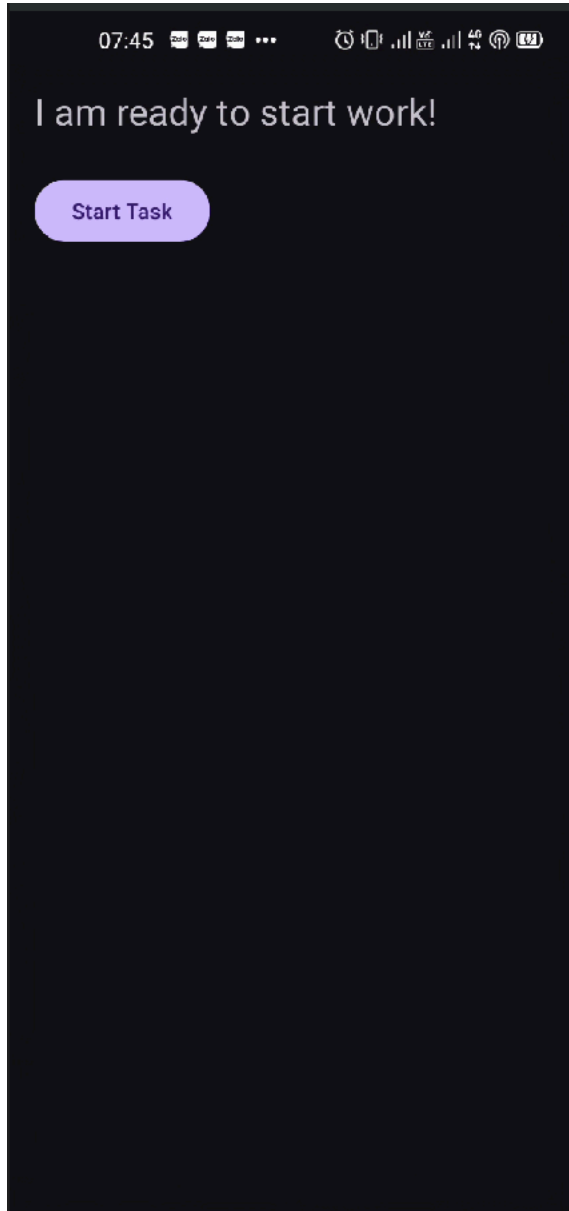
## **Những gì bạn sẽ làm:**

- Tạo một ứng dụng đơn giản thực hiện tác vụ nền bằng AsyncTask.
- Chạy ứng dụng và quan sát điều gì xảy ra khi xoay màn hình thiết bị.
- Triển khai lưu trạng thái Activity để bảo toàn nội dung của TextView khi thay đổi cấu hình

## **Tổng quan về ứng dụng**

Bạn sẽ xây dựng một ứng dụng với một TextView và một Button. Khi người dùng nhấn vào Button, ứng dụng sẽ ngủ trong một khoảng thời gian ngẫu nhiên, sau đó hiển thị một thông báo trong TextView khi hoạt động trở lại.

Dưới đây là giao diện của ứng dụng sau khi hoàn thành:



## Nhiệm vụ 1: Thiết lập dự án SimpleAsyncTask

Giao diện của SimpleAsyncTask bao gồm một Button để kích hoạt AsyncTask và một TextView để hiển thị trạng thái của ứng dụng.

### 1.1 Tạo dự án và bố cục giao diện

1. Tạo một dự án mới có tên SimpleAsyncTask sử dụng mẫu Empty Activity. Giữ nguyên các tùy chọn mặc định.
2. Mở tệp bố cục activity\_main.xml, sau đó chuyển sang tab Text.



3. Thêm thuộc tính layout\_margin vào ConstraintLayout cấp cao nhất



```
android:layout_margin="16dp"
```

4. Thêm hoặc chỉnh sửa các thuộc tính sau của TextView chứa dòng chữ “Hello World!” để có giá trị sau (trích xuất chuỗi văn bản vào resource):

Thuộc tính	Giá trị
android:id	“@+id/textView1
android:text	“I am ready to start work!”
android:textSize	“24sp”

5. Xóa các thuộc tính app:layout\_constraintRight\_toRightOf và app:layout\_constraintTop\_toTopOf.

6. Thêm một Button ngay bên dưới TextView, đồng thời thiết lập các thuộc tính sau (trích xuất văn bản của Button vào resource):

Thuộc tính	Giá trị
android:id	“@+id/button”
android:layout_width	“wrap_content”
android:layout_height	“wrap_content”
android:text	“Start Task”
android:layout_marginTop	“24dp”
android:onClick	“startTask”
app:layout_constraintStart_toStartOf	“parent”
app:layout_constraintTop_toBottomOf	“@+id/textView1”

7. Thuộc tính onClick của Button sẽ được tô vàng vì phương thức startTask() chưa được triển khai trong MainActivity. Để tạo phương thức này, hãy đặt con trỏ vào đoạn văn bản được đánh dấu, nhấn Alt + Enter (hoặc Option + Enter trên Mac), sau đó chọn Create ‘startTask(View)’ in ‘MainActivity’. Điều này sẽ tự động tạo một phương thức khung (stub) trong MainActivity.

Mã giải pháp cho activity\_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ready_to_start"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:onClick="startTask"
        android:text="@string/start_task"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView1"/>

</android.support.constraint.ConstraintLayout>
```

## Nhiệm vụ 2: Tạo lớp con của AsyncTask

AsyncTask là một lớp trừu tượng, có nghĩa là bạn cần tạo một lớp con kế thừa từ nó để sử dụng. Trong ví dụ này, AsyncTask sẽ thực hiện một tác vụ nền rất đơn giản: ngủ trong một khoảng thời gian ngẫu nhiên. Trong một ứng dụng thực tế, tác vụ nền có thể bao gồm nhiều công việc phức tạp hơn, từ truy vấn cơ sở dữ liệu, kết nối internet cho đến tính toán nước đi tiếp theo để đánh bại nhà vô địch cờ vây hiện tại.

Một lớp con của AsyncTask có các phương thức sau để thực hiện công việc ngoài luồng chính:

- `onPreExecute()`: Chạy trên luồng UI, được sử dụng để chuẩn bị cho tác vụ (chẳng hạn như hiển thị thanh tiến trình).
- `doInBackground()`: Chứa mã thực thi tác vụ chính trên một luồng riêng biệt.
- `onProgressUpdate()`: Được gọi trên luồng UI để cập nhật tiến trình (ví dụ: hiển thị mức độ hoàn thành trên thanh tiến trình).
- `onPostExecute()`: Cũng chạy trên luồng UI, dùng để cập nhật kết quả lên giao diện sau khi `AsyncTask` hoàn tất.

Khi tạo một lớp con của `AsyncTask`, bạn có thể cần cung cấp thông tin về công việc cần thực hiện, cách báo cáo tiến trình (nếu có), cũng như định dạng của kết quả trả về.

Bạn có thể cấu hình lớp con `AsyncTask` bằng các tham số sau:

- **Params**: Kiểu dữ liệu của tham số được truyền vào `doInBackground()` khi tác vụ được thực thi.
- **Progress**: Kiểu dữ liệu của đơn vị tiến trình được cập nhật thông qua `onProgressUpdate()`.
- **Result**: Kiểu dữ liệu của kết quả trả về từ `onPostExecute()`.

Ví dụ, nếu bạn có một lớp con `AsyncTask` có tên `MyAsyncTask`, nó có thể sử dụng các tham số như sau”

- `doInBackground()` nhận một `String` làm tham số (chẳng hạn để thực hiện truy vấn).
- `onProgressUpdate()` sử dụng một `Integer` để biểu thị phần trăm công việc đã hoàn thành.
- `onPostExecute()` trả về một `Bitmap` làm kết quả của truy vấn.

```
public class MyAsyncTask
    extends AsyncTask <String, Integer, Bitmap>{}
```

Trong phần thực hành này, bạn sẽ sử dụng một lớp con AsyncTask để định nghĩa một công việc sẽ chạy trên một luồng riêng biệt, tách biệt với luồng UI.

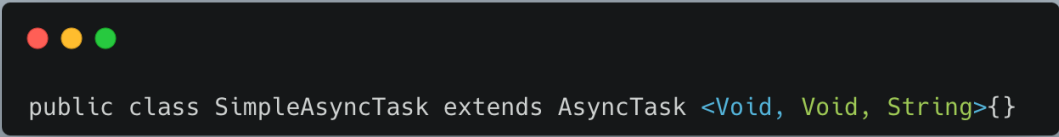
## 2.1 Tạo lớp con từ AsyncTask

Trong ứng dụng này, lớp con AsyncTask mà bạn tạo sẽ không cần tham số truy vấn hay cập nhật tiến trình. Bạn chỉ sử dụng hai phương thức: `doInBackground()` và `onPostExecute()`.

Các bước thực hiện:

1. Tạo một lớp Java mới có tên SimpleAsyncTask, kế thừa từ AsyncTask và sử dụng ba tham số kiểu dữ liệu tổng quát (generic type parameters). Đặt các tham số như sau:

Sử dụng Void cho Params, vì tác vụ này không cần dữ liệu đầu vào. Sử dụng Void cho Progress, vì tiến trình không được cập nhật. Sử dụng String cho Result, vì sau khi AsyncTask hoàn thành, bạn sẽ cập nhật TextView bằng một chuỗi.



```
public class SimpleAsyncTask extends AsyncTask <Void, Void, String>{}
```

2. Ở đầu lớp, khai báo một biến thành viên mTextView với kiểu WeakReference<TextView>:



```
private WeakReference<TextView> mTextView;
```

3. Triển khai constructor cho AsyncTask, nhận một TextView làm tham số và tạo một tham chiếu yếu (weak reference) đến TextView đó:

```
SimpleAsyncTask(TextView tv) {  
    mTextView = new WeakReference<>(tv);  
}
```

Lớp AsyncTask cần cập nhật TextView trong Activity sau khi hoàn tất tác vụ ngủ (sleeping) trong phương thức onPostExecute(). Vì vậy, constructor của lớp sẽ cần một tham chiếu đến TextView để có thể cập nhật nội dung sau khi tác vụ hoàn tất.

Tại sao cần sử dụng WeakReference? Nếu bạn truyền trực tiếp một TextView vào constructor của AsyncTask và lưu nó vào một biến thành viên, thì tham chiếu đó sẽ khiến Activity không thể được garbage collected (thu hồi bộ nhớ), ngay cả khi Activity bị hủy và tạo lại (chẳng hạn như khi xoay màn hình. Điều này được gọi là leaky context (rò rỉ bộ nhớ), và Android Studio sẽ cảnh báo nếu bạn cố làm như vậy.

Việc sử dụng WeakReference giúp ngăn chặn rò rỉ bộ nhớ bằng cách cho phép đối tượng được tham chiếu có thể bị thu hồi bộ nhớ nếu cần thiết.

## 2.2 Triển khai phương thức doInBackground()

Phương thức doInBackground() là bắt buộc trong lớp con của AsyncTask.

1. Đặt con trỏ chuột vào phần khai báo lớp bị tô sáng, nhấn Alt + Enter (Option + Enter trên Mac), sau đó chọn Implement methods. Chọn doInBackground() và nhấn OK. Một mẫu phương thức sau sẽ được thêm vào lớp của bạn:

```
@Override  
protected String doInBackground(Void... voids) {  
    return null;  
}
```

2. Thêm đoạn mã để tạo một số nguyên ngẫu nhiên trong khoảng từ 0 đến 10. Đây sẽ là số mili giây mà tác vụ sẽ tạm dừng. Vì khoảng thời gian này khá ngắn, hãy nhân số đó với 200 để kéo dài thời gian tạm dừng:

```
Random r = new Random();  
int n = r.nextInt(11);  
int s = n * 200;
```

3. Thêm một khối try / catch để cho luồng ngủ (sleep) trong khoảng thời gian đã tính toán:

```
try {  
    Thread.sleep(s);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}
```

4. Thay thế dòng return hiện hữu để trả về chuỗi “Thức dậy rồi! Sau khi ngủ trong xx mili-giây”. Trong đó xx là số mili-giây mà ứng dụng đã tạm dừng.

```
return "Awake at last after sleeping for " + s + " milliseconds!";
```

Phương thức `doInBackground()` hoàn chỉnh sẽ trông như sau:

```

@Override
protected String doInBackground(Void... voids) {
    Random r = new Random();
    int n = r.nextInt(bound: 11);
    int s = n*200;

    try {
        Thread.sleep(s);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return "Awake at last after sleeping for " + s + " milliseconds!";
}

```

## 2.3 Triển khai phương thức onPostExecute()

Sau khi phương thức doInBackground() hoàn thành, giá trị trả về sẽ tự động được truyền vào phương thức onPostExecute() để xử lý kết quả.

1. Triển khai phương thức onPostExecute() nhận một tham số kiểu String và hiển thị nó lên TextView:

```

@Override
protected void onPostExecute(String result) { mTextView.get().setText(result); }

```

Tham số String cho phương thức này là tham số bạn đã định nghĩa trong tham số thứ ba của định nghĩa lớp AsyncTask và là kết quả phương thức doInBackground() trả về.

Bởi vì mTextView là một weak reference, nên bạn phải tham chiếu nó bằng phương thức get() để lấy đối tượng TextView cơ bản và gọi setText() trên đó.

## Nhiệm vụ 3: Triển khai những bước cuối cùng

### 3.1 Triển khai phương thức để khởi động AsyncTask

Hiện tại, ứng dụng của bạn đã có một lớp AsyncTask thực hiện tác vụ trong nền (hoặc sẽ thực hiện nếu không gọi sleep() là công việc mô phỏng). Bây giờ, bạn có thể triển khai phương thức onClick để nút “Start Task” kích hoạt được tác vụ nền.

1. Trong file MainActivity.java, thêm một biến thành viên để lưu trữ TextView.

```
private TextView mTextView;
```

2. Trong phương thức onCreate(), khởi tạo mTextView cho TextView trong bố cục

```
setContentView(R.layout.activity_main);  
mTextView = findViewById(R.id.textView1);
```

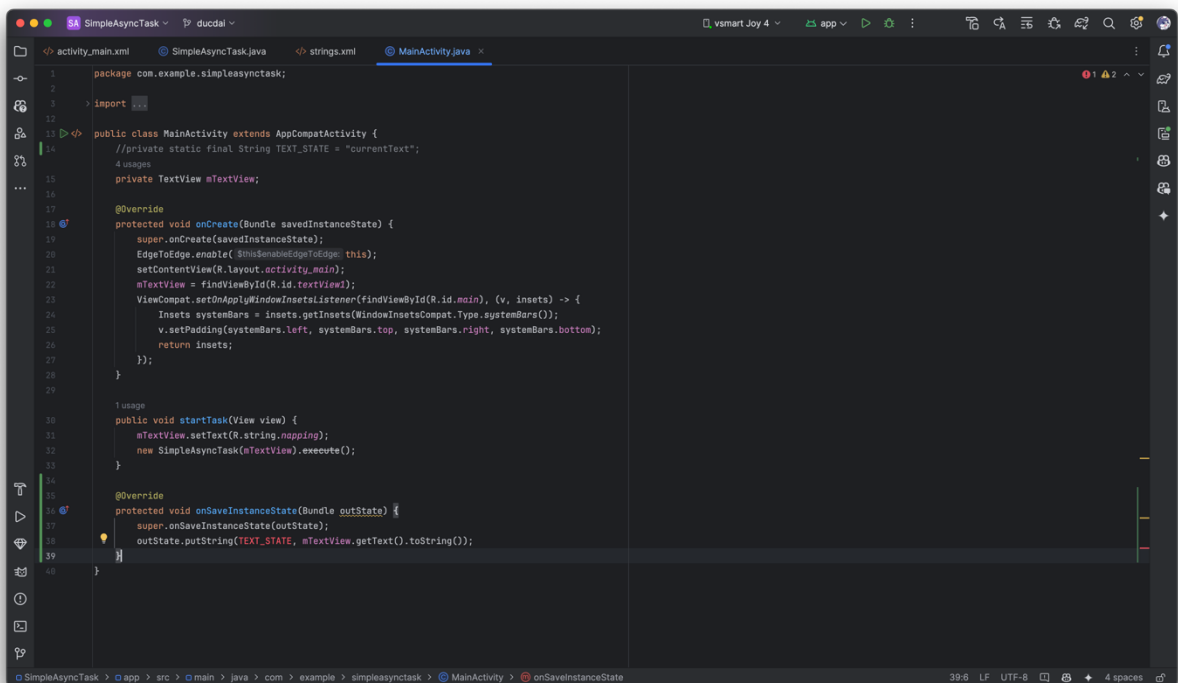
3. Trong phương thức startTask(), cập nhật TextView để hiển thị dòng chữ “Napping...”. Trích xuất tin nhắn đó thành một chuỗi tài nguyên

```
mTextView.setText(R.string.napping);
```

4. Tạo một phiên bản của SimpleAsyncTask, truyền TextView mTextView cho hàm tạo. Gọi execute() trên phiên bản SimpleAsyncTask.

```
new SimpleAsyncTask(mTextView).execute();
```

Mã giải pháp cho MainActivity:



## 3.2 Triển khai phương thức onSaveInstanceState()



1. Chạy ứng dụng và nhấp vào nút Bắt đầu tác vụ. Ứng dụng ngủ trong bao lâu?
2. Nhấp vào nút Bắt đầu tác vụ một lần nữa và trong khi ứng dụng đang ngủ, hãy xoay thiết bị. Nếu tác vụ nền hoàn tất trước khi bạn có thể xoay điện thoại, hãy thử lại.
  - Khi bạn xoay thiết bị, hệ thống sẽ khởi động lại ứng dụng, gọi `onDestroy()` và sau đó `onCreate()`. `AsyncTask` sẽ tiếp tục chạy ngay cả khi hoạt động bị hủy, nhưng nó sẽ mất khả năng báo cáo lại cho UI của hoạt động. Nó sẽ không bao giờ có thể cập nhật `TextView` đã được truyền cho nó, vì `TextView` cụ thể đó cũng đã bị hủy.
  - Sau khi hoạt động bị hủy, `AsyncTask` sẽ tiếp tục chạy cho đến khi hoàn tất trong nền, tiêu tốn tài nguyên hệ thống. Cuối cùng, hệ thống sẽ hết tài nguyên và `AsyncTask` sẽ không hoạt động.
  - Ngay cả khi không có `AsyncTask`, việc xoay thiết bị sẽ đặt lại tất cả các thành phần UI về trạng thái ban đầu của chúng. Trạng thái mặc định, đối với `TextView` là chuỗi mặc định mà bạn đặt trong tệp bố cục.

Vì những lý do này, `AsyncTask` không phù hợp với các tác vụ có thể bị gián đoạn do Activity bị hủy. Trong các trường hợp sử dụng mà điều này là quan trọng, bạn có thể sử dụng một loại lớp nền khác gọi là `AsyncTaskLoader` mà bạn sẽ tìm hiểu trong phần thực hành sau.

Để ngăn `TextView` khỏi việc thiết lập lại chuỗi ban đầu, bạn cần lưu trạng thái của nó. Bây giờ bạn sẽ triển khai `onSaveInstanceState()` để bảo toàn nội dung của `TextView` khi hoạt động bị hủy để phản hồi thay đổi cấu hình như xoay thiết bị.

3. Ở đầu lớp, thêm hằng số cho khóa của văn bản hiện tại trong gói trạng thái:

```
private static final String TEXT_STATE = "currentText";
```

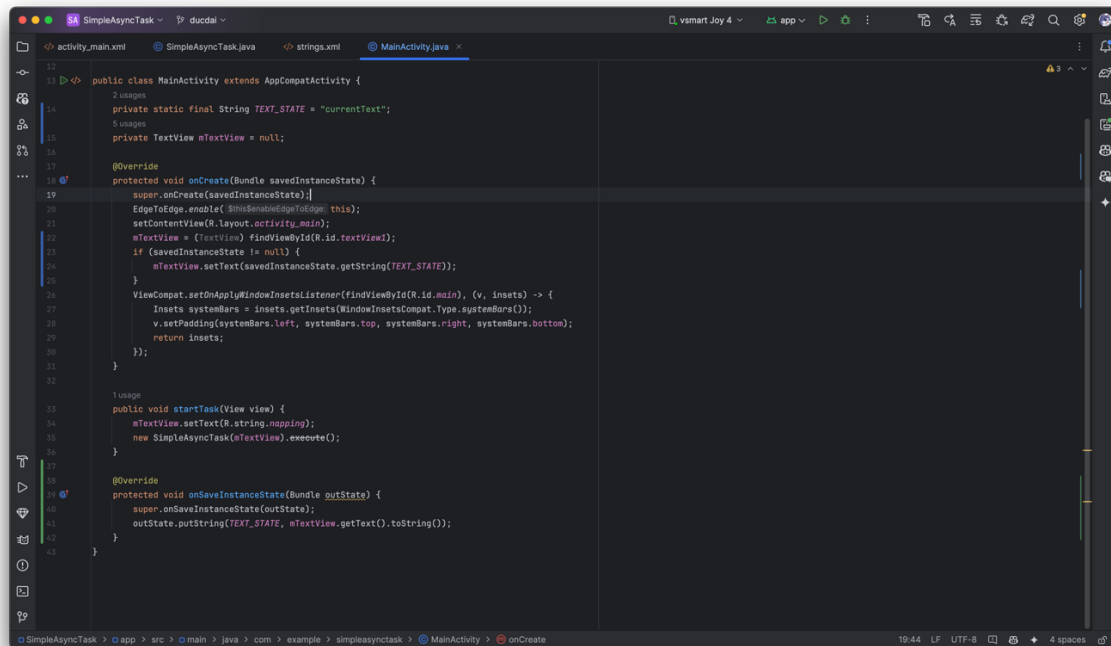
4. Override phương thức `onSaveInstanceState()` trong `MainActivity` để giữ lại văn bản bên trong `TextView` khi activity bị hủy:

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
}
```

5. Trong phương thức `onCreate()`, lấy giá trị của `TextView` từ gói trạng thái khi activity được khôi phục

```
outState.putString(TEXT_STATE, mTextView.getText().toString());
```

Mã giải pháp cho MainActivity:



## 1.2) AsyncTask và AsyncTaskLoader

### Giới thiệu

Trong phần này bạn sử dụng AsyncTask để chạy một tác vụ nền thực hiện lấy dữ liệu từ internet sử dụng một REST API đơn giản. Bạn sẽ dùng Google APIs Explorer để truy vấn Books API, triển khai truy vấn này trong luồng công việc bằng cách sử dụng AsyncTask và hiển thị kết quả trong UI của bạn

Sau đó, bạn triển khai lại tác vụ nền đó bằng AsyncTaskLoader, đây là cách hiệu quả hơn để cập nhật UI của bạn

### Những điều bạn nên biết

Bạn cần có thể:

- Tạo một activity
- Thêm một TextView vào bố cục của activity
- Triển chức năng onClick cho một button trong bố cục của bạn
- Triển khai một AsyncTask và hiển thị kết quả trên giao diện của bạn
- Truyền thông tin giữa các activity dưới dạng các thông tin bổ sung

Những gì bạn sẽ tìm hiểu

- Cách sử dụng Google APIs Explorer để tìm hiểu về Google API và xem phản hồi JSON đối với các HTTP request
- Cách sử dụng Google Books API để truy xuất dữ liệu qua internet và giữ cho giao diện người dùng nhanh và phản hồi. Bạn sẽ không tìm hiểu chi tiết về Book API, ứng dụng của bạn sẽ chỉ sử dụng chức năng tìm kiếm sách đơn giản
- Cách phân tích kết quả JSON từ truy vấn API của bạn
- Cách triển khai AsyncTaskLoader để lưu trữ dữ liệu về các thay đổi cấu hình
- Cách cập nhật UI của bạn bằng cách sử dụng lệnh gọi lại trình tải

Bạn sẽ làm những gì

- Sử dụng Google APIs Explorer để tìm hiểu về Book API
- Tạo ứng dụng “Who Wrote It?”, ứng dụng sử dụng luồng worker truy vấn Book API và hiển thị kết quả lên UI
- Điều chỉnh ứng dụng “Who Wrote it?” để sử dụng AsyncTaskLoader thay cho AsyncTask

Tổng quan về ứng dụng

Bạn sẽ xây dựng một ứng dụng chứa một EditText và một Button.

- Người dùng nhập tên của cuốn sách vào EditText và nhấn vào Button
- Button thực thi một AsyncTask thực hiện truy vấn Google Books API để tìm tác giả và tiêu đề của cuốn sách mà người dùng đang tìm kiếm
- Kết quả được lấy và hiển thị trong TextView bên dưới Button

Một khi ứng dụng hoạt động, bạn chỉnh sửa app để sử dụng AsyncTaskLoader thay cho lớp AsyncTask.

Nhiệm vụ 1: Tìm hiểu Google Books API

Trong bài thực hành này, bạn sử dụng Google Books API để tìm kiếm thông tin về một cuốn sách, chẳng hạn như tác giả và tiêu đề của cuốn sách. Book API cung cấp quyền truy cập theo chương trình vào dịch vụ Google Book Search bằng cách sử dụng REST API. Đây là dịch vụ tương tự được sử dụng ở chế độ nền khi bạn thực hiện tìm kiếm thủ công trên Google Books. Bạn có thể sử dụng Google APIs

Explorer và Google Book Search trong trình duyệt của mình để xác minh rằng ứng dụng Android của bạn đang nhận được kết quả mong đợi.

### **1.3) Broadcast receivers**

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

### **2.1) Thông báo**

### **2.2) Trình quản lý cảnh báo**

### **2.3) JobScheduler**

## **CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG**

### **Bài 1) Tùy chọn và cài đặt**

#### **1.1) Shared preferences**

#### **1.2) Cài đặt ứng dụng**

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

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

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