

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	13
1.3)	Trình chỉnh sửa bố cục	27
1.4)	Văn bản và các chế độ cuộn	27
1.5)	Tài nguyên có sẵn	27
Bài 2)	Activities	27
2.1)	Activity và Intent	27
2.2)	Vòng đời của Activity và trạng thái	27
2.3)	Intent ngầm định	27
Bài 3)	Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	27
3.1)	Trình gỡ lỗi	27
3.2)	Kiểm thử đơn vị	27
3.3)	Thư viện hỗ trợ	27
CHƯƠNG 2.	Trải nghiệm người dùng	28
Bài 1)	Tương tác người dùng	28
1.1)	Hình ảnh có thể chọn	28
1.2)	Các điều khiển nhập liệu	28
1.3)	Menu và bộ chọn	28
1.4)	Điều hướng người dùng	28
1.5)	RecyclerView	28
Bài 2)	Trải nghiệm người dùng thú vị	28
2.1)	Hình vẽ, định kiểu và chủ đề	28
2.2)	Thẻ và màu sắc	28
2.3)	Bố cục thích ứng	28
Bài 3)	Kiểm thử giao diện người dùng	28

3.1) Espresso cho việc kiểm tra UI	28
CHƯƠNG 3. Làm việc trong nền	28
Bài 1) Các tác vụ nền	28
1.1) AsyncTask	28
1.2) AsyncTask và AsyncTaskLoader	28
1.3) Broadcast receivers	28
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền	28
2.1) Thông báo	28
2.2) Trình quản lý cảnh báo	28
2.3) JobScheduler	28
CHƯƠNG 4. Lưu dữ liệu người dùng	29
Bài 1) Tùy chọn và cài đặt	29
1.1) Shared preferences	29
1.2) Cài đặt ứng dụng	29
Bài 2) Lưu trữ dữ liệu với Room	29
2.1) Room, LiveData và ViewModel	29
2.2) Room, LiveData và ViewModel	29
3.1) Trình gỡ lỗi	

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

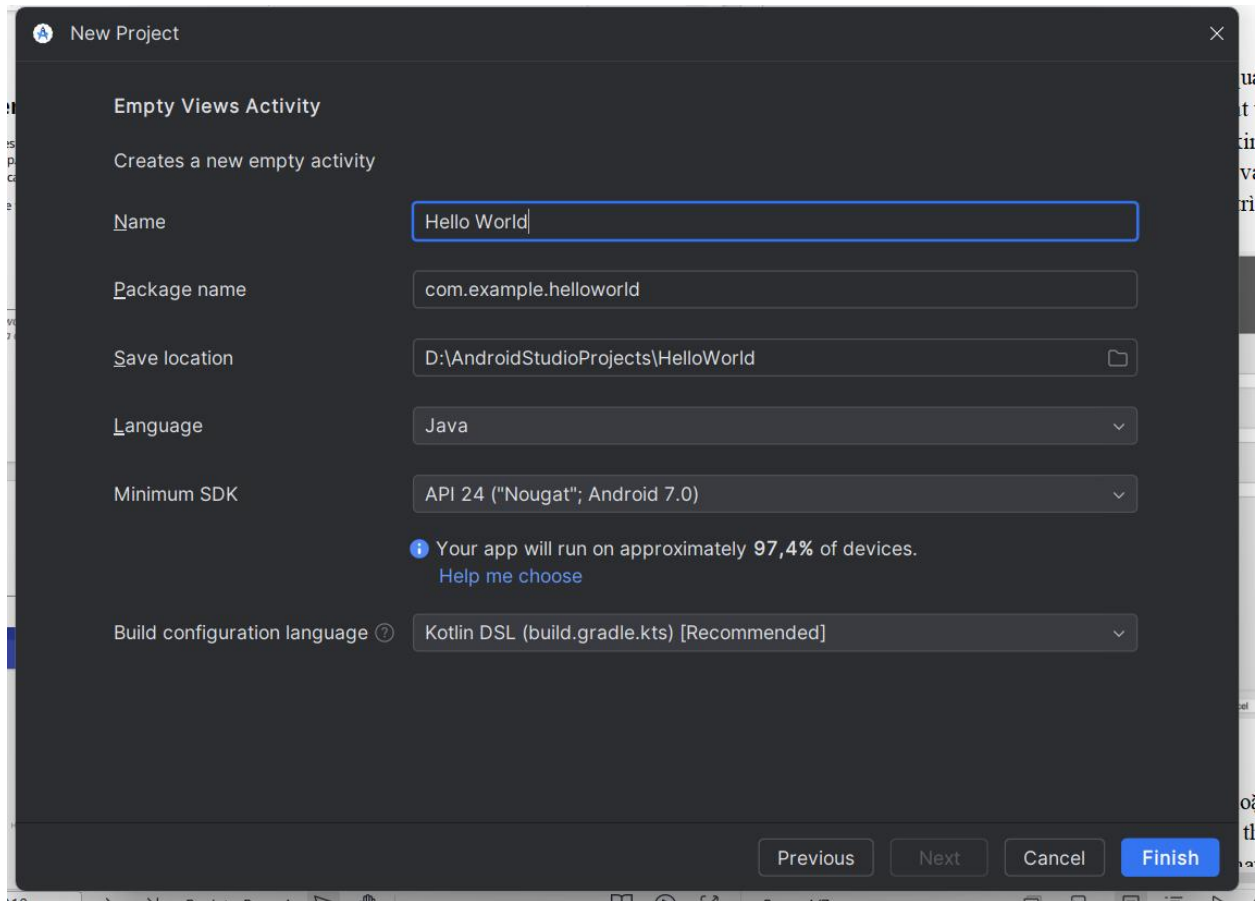
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

- Cách cài đặt và sử dụng IDE Android Studio.

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

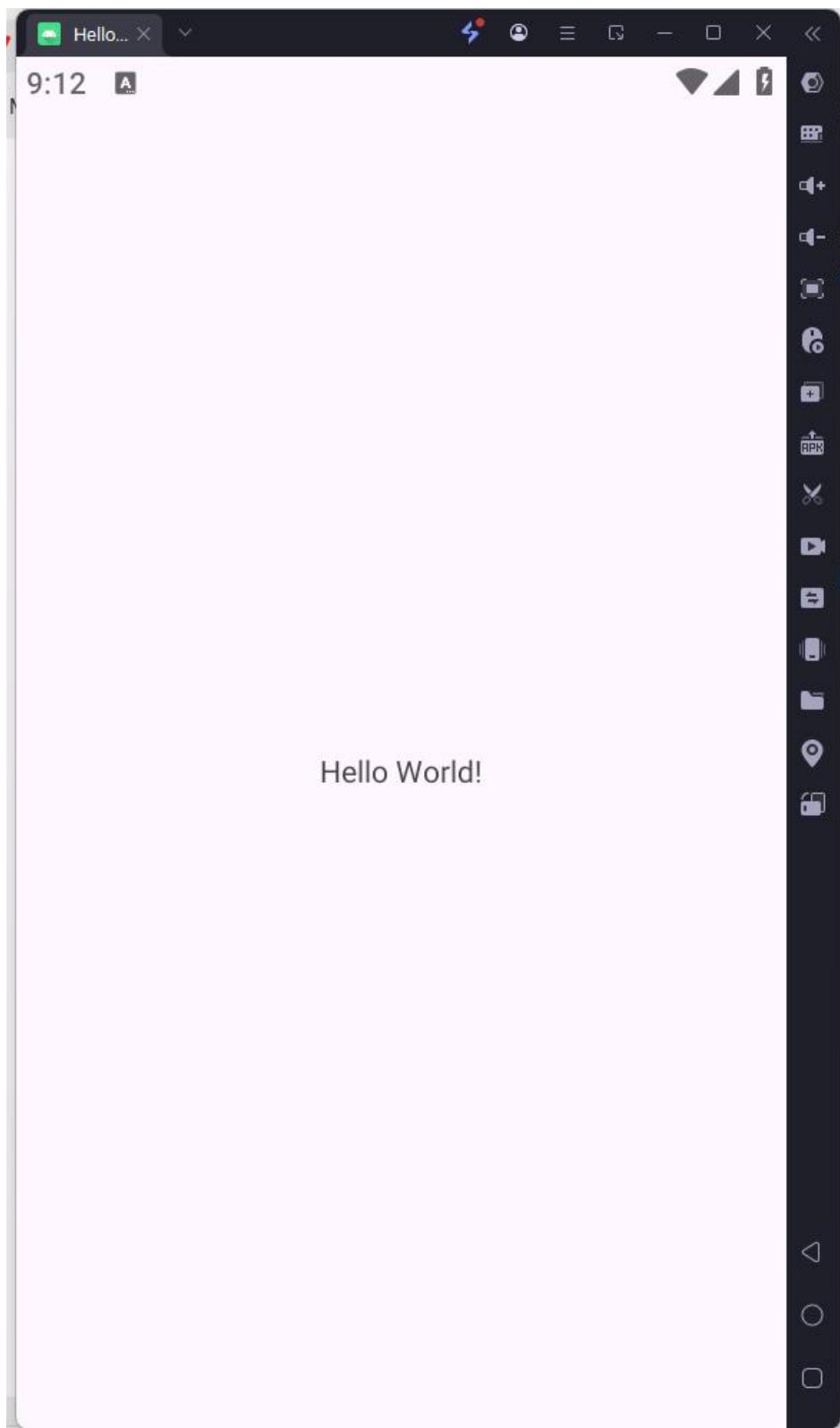
Những gì bạn sẽ làm

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

Tổng quan ứng dụng

Sau khi cài đặt thành công Android Studio, bạn sẽ tạo 1 dự án mới từ mẫu cho ứng dụng Hello World. Ứng dụng đơn giản này hiện chữ “Hello Word” lên màn hình của thiết bị Android ảo hoặc thực.

Đây là giao diện của ứng dụng sau khi hoàn thành:



Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp (IDE) hoàn chỉnh, bao gồm một trình soạn thảo mã nâng cao và một tập hợp các mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ hỗ trợ phát triển, gỡ lỗi, kiểm thử và tối ưu hiệu suất, giúp việc phát triển ứng dụng trở nên nhanh chóng và dễ dàng hơn.

Bạn có thể kiểm thử ứng dụng của mình trên nhiều trình giả lập được cấu hình sẵn hoặc trực tiếp trên thiết bị di động của mình, xây dựng ứng dụng hoàn chỉnh và phát hành trên cửa hàng Google Play.

Android Studio có sẵn cho máy tính chạy Windows hoặc Linux, cũng như cho Mac chạy macOS. Phiên bản mới nhất của **OpenJDK (Java Development Kit)** được tích hợp sẵn trong Android Studio.

Để bắt đầu sử dụng Android Studio, trước tiên hãy kiểm tra **yêu cầu hệ thống** để đảm bảo máy tính của bạn đáp ứng đủ điều kiện. Quá trình cài đặt tương tự trên tất cả các nền tảng, mọi điểm khác biệt sẽ được ghi chú bên dưới.

1. Truy cập **trang web dành cho nhà phát triển Android** và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
2. Chấp nhận các **cấu hình mặc định** trong tất cả các bước và đảm bảo chọn đầy đủ các thành phần cần thiết để cài đặt.
3. Sau khi cài đặt xong, **Trình hướng dẫn thiết lập (Setup Wizard)** sẽ tải xuống và cài đặt thêm một số thành phần bổ sung, bao gồm **Android SDK**. Hãy kiên nhẫn, quá trình này có thể mất một chút thời gian tùy thuộc vào tốc độ Internet, và có thể một số bước sẽ trông giống nhau.
4. Khi quá trình tải xuống hoàn tất, **Android Studio sẽ khởi động**, và bạn đã sẵn sàng tạo dự án đầu tiên!

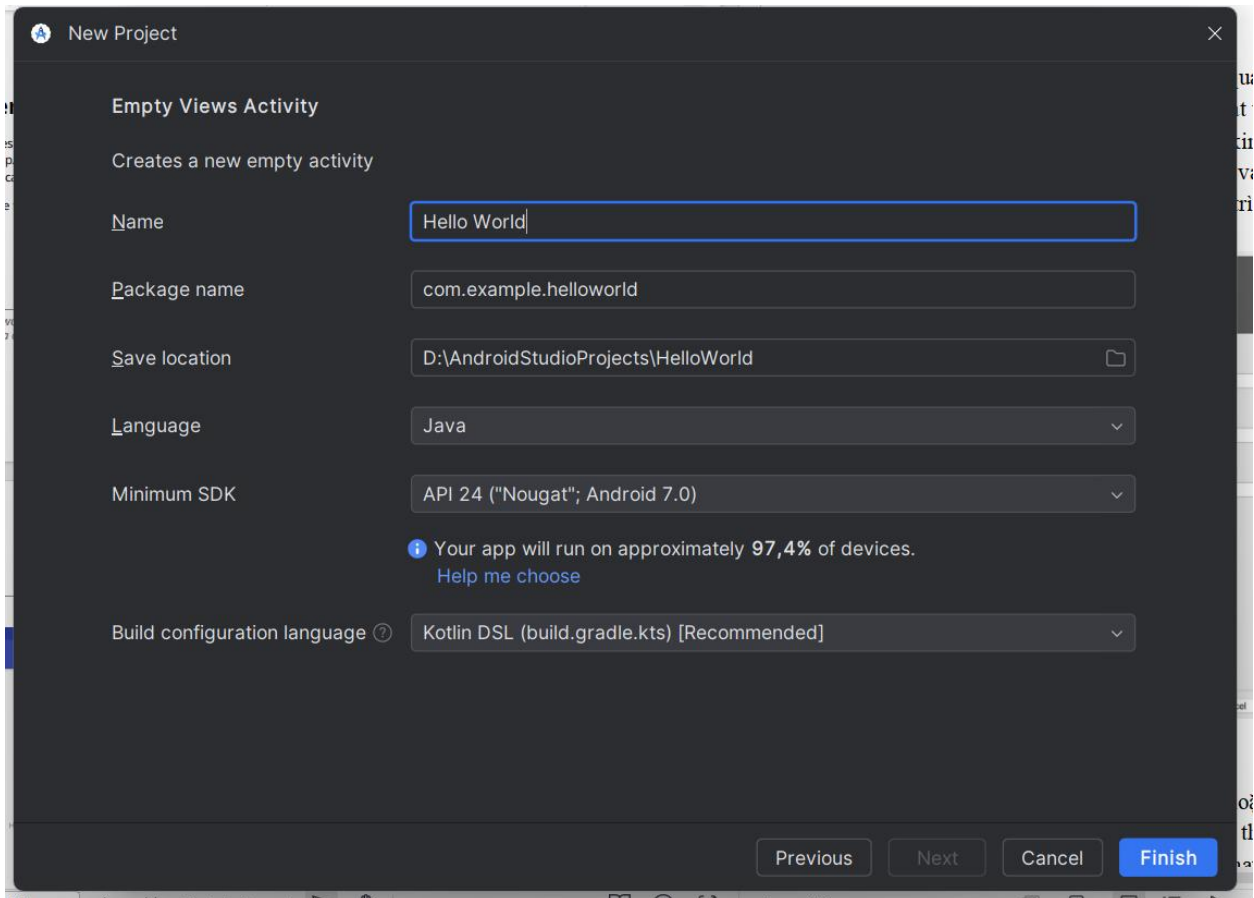
Nhiệm vụ 2: Tạo ứng dụng Hello World

Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị dòng chữ "Hello World" để xác minh rằng Android Studio đã được cài đặt đúng cách và tìm hiểu những kiến thức cơ bản về phát triển ứng dụng với Android Studio.

2.1 Tạo dự án ứng dụng

1. Mở **Android Studio** nếu nó chưa được mở.

2. Trong cửa sổ **Welcome to Android Studio**, nhấp vào **Start a new Android Studio project** (Bắt đầu một dự án Android Studio mới).
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** vào trường **Application name** (Tên ứng dụng).

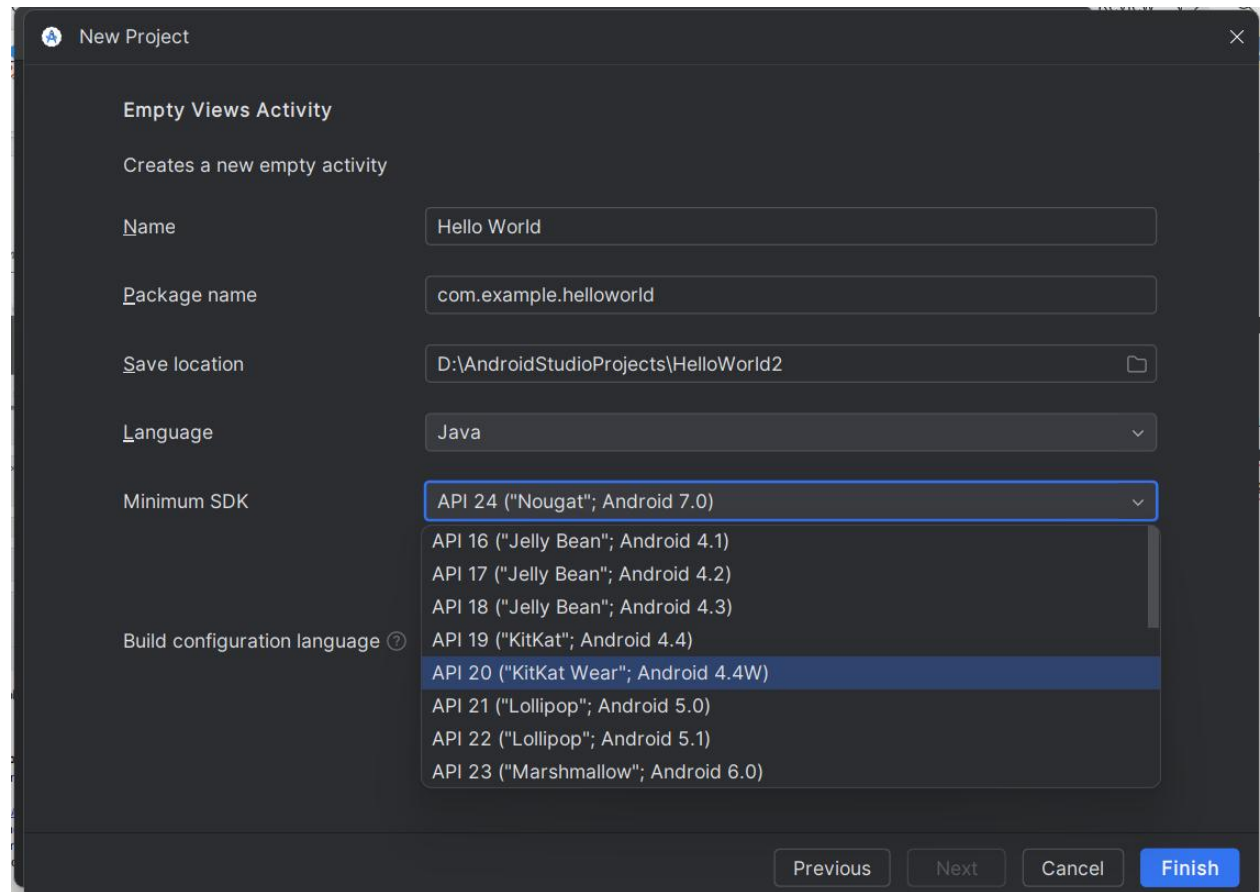


4. Xác minh rằng **Project location** (vị trí dự án) mặc định là nơi bạn muốn lưu trữ ứng dụng **Hello World** và các dự án khác trong Android Studio, hoặc thay đổi nó thành thư mục bạn muốn.
5. Chấp nhận mặc định **android.example.com** cho **Company Domain** (tên miền công ty), hoặc tạo một tên miền riêng.

Nếu bạn không có ý định xuất bản ứng dụng, bạn có thể giữ nguyên mặc định. Lưu ý rằng thay đổi **package name** (tên gói) sau này sẽ cần thêm công việc chỉnh sửa.

6. Bỏ chọn các tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấp **Next**.

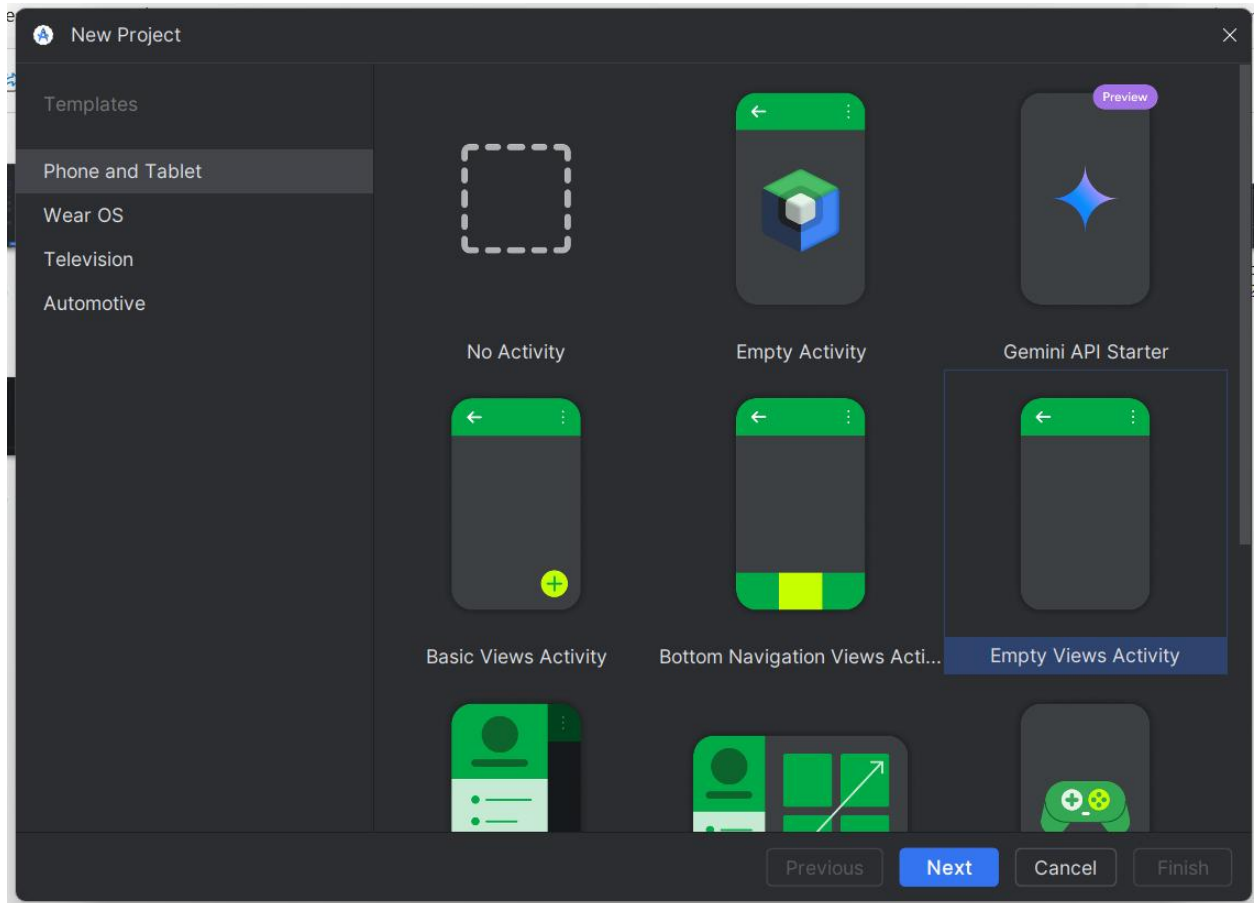
7. Trong màn hình **Target Android Devices**, đảm bảo **Phone and Tablet** đã được chọn. Đặt **API 24: Android 7.0 Nougat** làm **Minimum SDK** (SDK tối thiểu); nếu chưa đúng, hãy sử dụng menu thả xuống để chọn.



8. Bỏ chọn tùy chọn **Include Instant App support** và tất cả các tùy chọn khác, sau đó nhấp **Next**. Nếu dự án của bạn yêu cầu cài đặt thêm các thành phần cho **Target SDK**, Android Studio sẽ tự động tải và cài đặt chúng.

9. Cửa sổ **Add an Activity** sẽ xuất hiện. **Activity** là một thành phần quan trọng trong ứng dụng Android, đại diện cho một màn hình với giao diện mà người dùng có thể tương tác. Một **Activity** thường có một **layout** đi kèm để xác định cách các phần tử UI hiển thị trên màn hình. Android Studio cung

cấp nhiều mẫu **Activity template** để giúp bạn bắt đầu nhanh hơn. Đối với dự án **Hello World**, chọn **Empty Activity**, sau đó nhấp **Next**.



10. Màn hình **Cấu hình Activity (Configure Activity)** sẽ xuất hiện (giao diện có thể khác nhau tùy vào mẫu Activity bạn đã chọn ở bước trước). Mặc định, Activity trống được cung cấp bởi mẫu sẽ có tên là **MainActivity**. Bạn có thể thay đổi tên nếu muốn, nhưng trong bài học này, chúng ta sẽ sử dụng **MainActivity**.

11. Đảm bảo rằng tùy chọn **Generate Layout file** được chọn. Tên mặc định của tệp bố cục là **activity_main**. Bạn có thể thay đổi nếu muốn, nhưng trong bài học này, chúng ta sẽ sử dụng **activity_main**.

12. Đảm bảo rằng tùy chọn **Backwards Compatibility (App Compat)** được chọn. Điều này giúp ứng dụng của bạn tương thích ngược với các phiên bản Android cũ hơn.

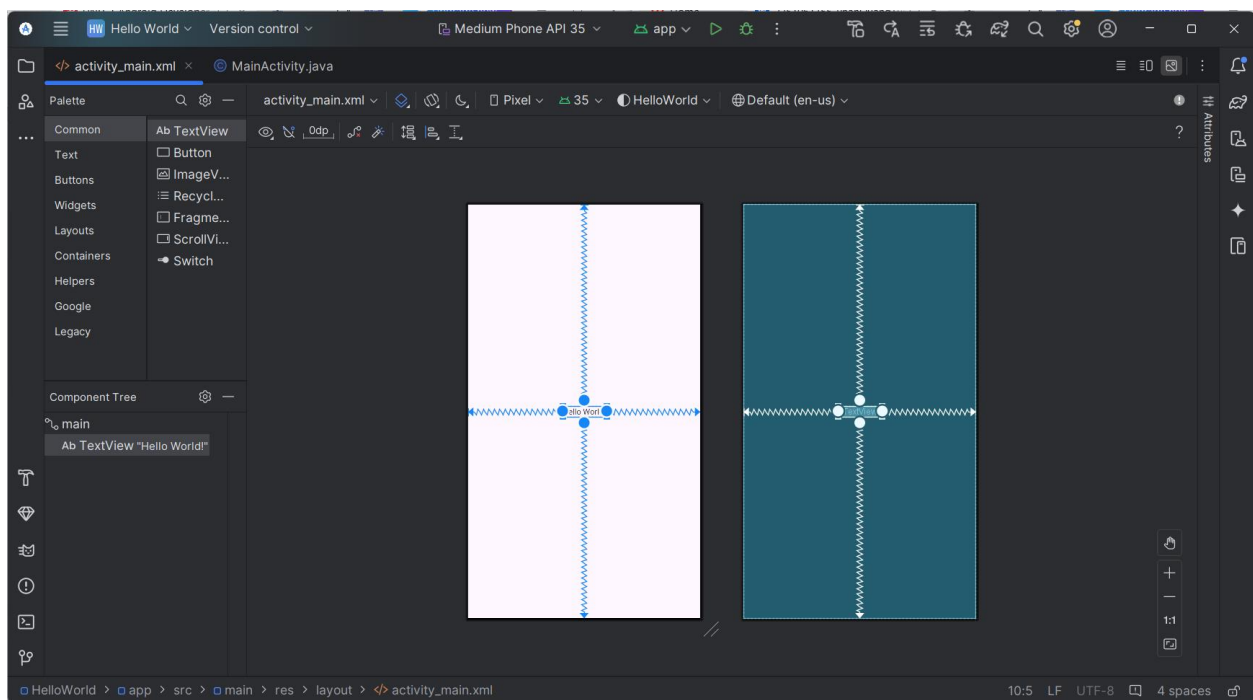
13. Nhấp vào **Finish** để hoàn tất quá trình tạo dự án.

Android Studio sẽ tạo một thư mục cho các dự án của bạn và tiến hành xây dựng (build) dự án bằng **Gradle** (quá trình này có thể mất vài phút).

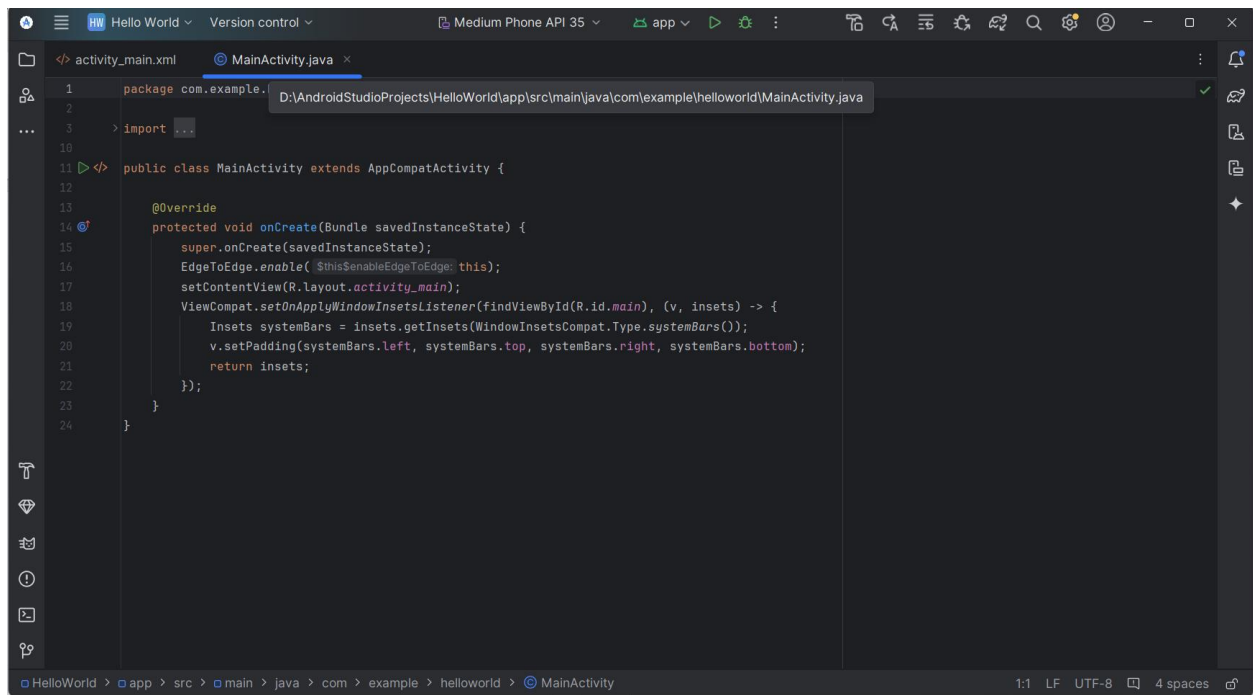
Bạn cũng có thể thấy thông báo "**Tip of the day**", hiển thị các phím tắt và mẹo hữu ích khác. Nhấp vào **Close** để đóng thông báo này.

Sau đó, trình chỉnh sửa **Android Studio** sẽ xuất hiện. Hãy làm theo các bước sau:

1. Nhấp vào tab **activity_main.xml** để mở trình chỉnh sửa bố cục (layout editor).
2. Nhấp vào tab **Design** trong trình chỉnh sửa bố cục, nếu chưa được chọn, để hiển thị bản xem trước giao diện đồ họa của bố cục như hình bên dưới.



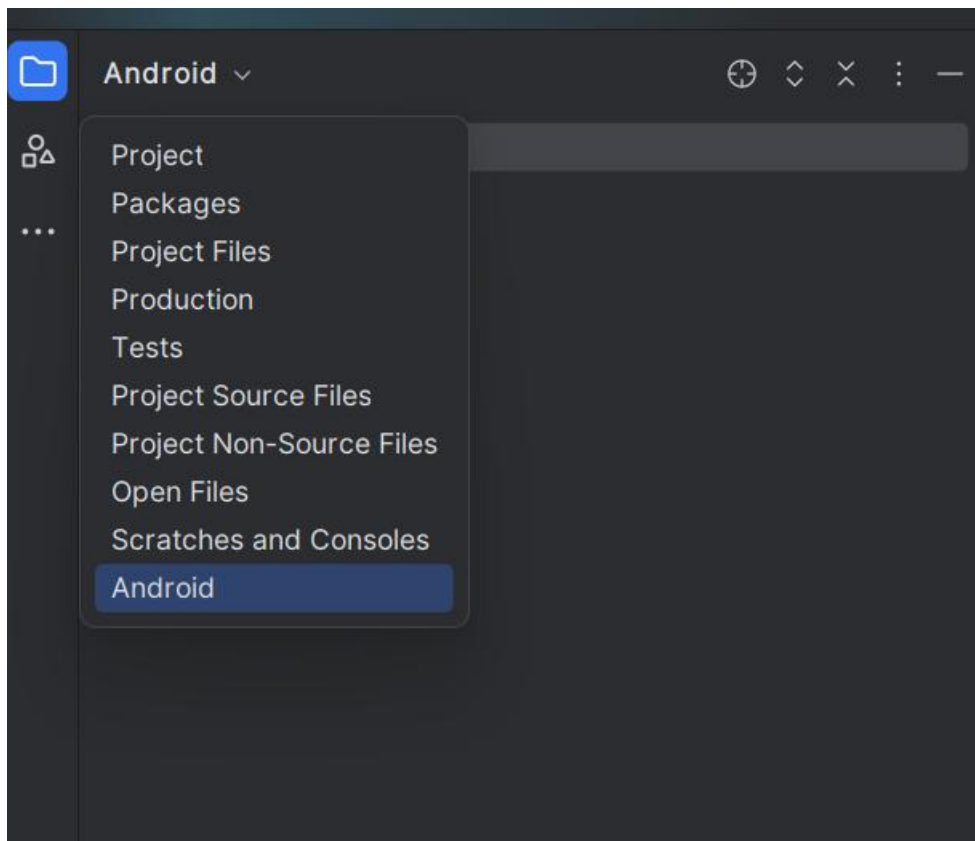
3. Nhấp vào tab **MainActivity.java** để mở trình chỉnh sửa mã (code editor) như hình bên dưới.



2.2 Khám phá Project > Android pane

Trong bài thực hành này, bạn sẽ khám phá cách tổ chức dự án trong **Android Studio**.

1. Nếu chưa được chọn, hãy nhấp vào tab **Project** trong cột tab dọc ở bên trái cửa sổ **Android Studio**. Khi đó, **Project pane** sẽ xuất hiện.
2. Để xem dự án theo hệ thống phân cấp tiêu chuẩn của **Android**, hãy chọn **Android** từ menu thả xuống ở đầu **Project pane**, như hình bên dưới.



1.2) Giao diện người dùng tương tác đầu tiên

Nhiệm vụ 3: Thay đổi UI element attributes

Ngăn Thuộc tính (Attributes pane) cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng (UI element). Bạn có thể tìm thấy các thuộc tính (còn được gọi là thuộc tính property) chung cho tất cả các View trong tài liệu của lớp **View**.

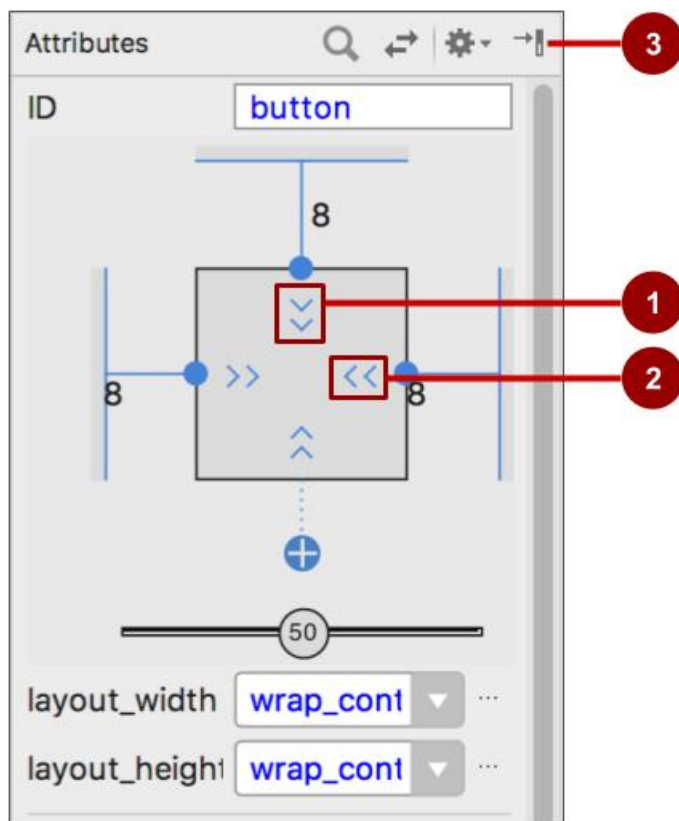
Trong nhiệm vụ này, bạn sẽ nhập các giá trị mới và thay đổi các giá trị của các thuộc tính quan trọng của **Button**, những thuộc tính này có thể áp dụng cho hầu hết các loại **View**.

3.1 Thay đổi kích thước Button

Trình chỉnh sửa bố cục (layout editor) cung cấp các tay cầm thay đổi kích thước ở bốn góc của một **View**, giúp bạn có thể thay đổi kích thước của **View** một cách nhanh chóng.

Bạn có thể kéo các tay cầm ở từng góc của **View** để thay đổi kích thước, nhưng làm như vậy sẽ **gán cứng (hardcode) giá trị chiều rộng và chiều cao**. Nên tránh gán cứng kích thước cho hầu hết các phần tử **View**, vì các kích thước cố định không thể thích ứng với nội dung và kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn **Attributes** ở bên phải của trình chỉnh sửa bố cục để chọn **chế độ kích thước (sizing mode)** không sử dụng các kích thước cố định. Ngăn **Attributes** có một bảng điều chỉnh kích thước hình vuông gọi là **view inspector** ở trên cùng. Các ký hiệu bên trong hình vuông này đại diện cho các cài đặt **chiều cao và chiều rộng** như sau:



Trong hình trên:

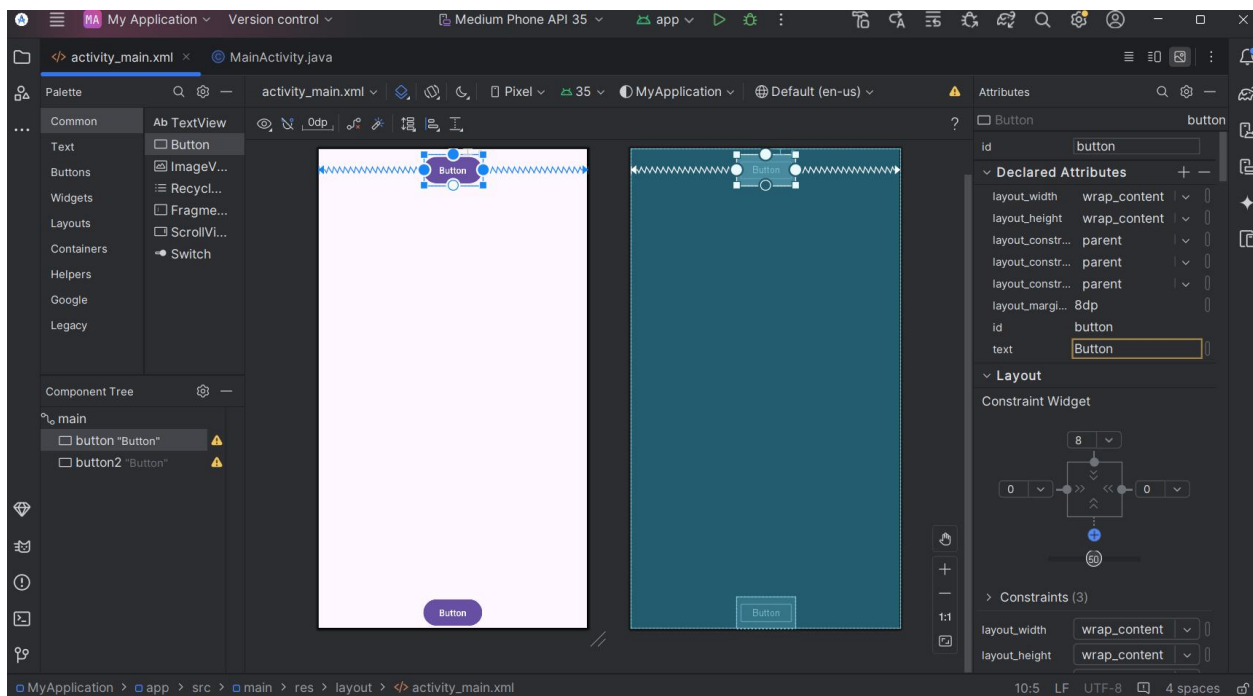
1.Điều khiển chiều cao. Điều khiển này xác định thuộc tính **layout_height** và xuất hiện trong hai đoạn ở các cạnh trên và dưới của hình vuông. Các góc cho thấy rằng điều khiển này được đặt thành **wrap_content**, có nghĩa là View sẽ mở rộng theo chiều dọc tùy theo nội dung bên trong. Số "8" chỉ ra rằng có một khoảng lề tiêu chuẩn được đặt thành **8dp**.

2.Điều khiển chiều rộng. Điều khiển này xác định thuộc tính **layout_width** và xuất hiện trong hai đoạn ở các cạnh trái và phải của hình vuông. Các góc cho thấy rằng điều khiển này được đặt thành **wrap_content**, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần để phù hợp với nội dung bên trong, lên đến một lề tối đa là 8dp.

3.Nút đóng bảng thuộc tính. Nhấp để đóng bảng.

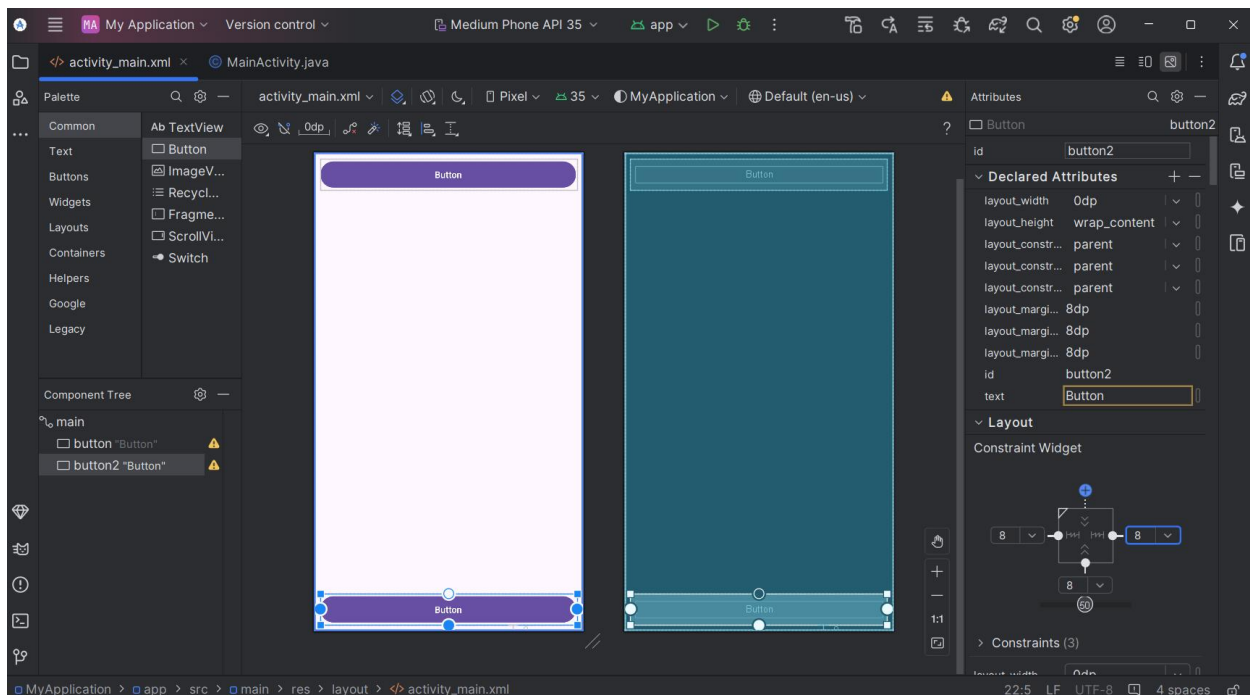
Làm theo các bước sau:

1. Chọn Button đầu tiên trong Component Tree.
2. Nhấn vào tab Attributes ở phía bên phải của cửa sổ trình chỉnh sửa layout.
3. Nhấp vào điều khiển chiều rộng hai lần—Lần nhấp đầu tiên: Thay đổi thành Fixed với các đường thẳng. Lần nhấp thứ hai: Thay đổi thành Match Constraints với các lò xo, như hiển thị trong hình động bên dưới.



Kết quả của việc thay đổi điều khiển chiều rộng, thuộc tính **layout_width** trong Attributes pane hiển thị giá trị **match_constraint** và phần tử Button kéo dẫn theo chiều ngang để lấp đầy khoảng trống giữa cạnh trái và cạnh phải của bố cục.

4. Chọn Button thứ hai và thực hiện các thay đổi tương tự đối với **layout_width** như ở bước trước, như minh họa trong hình bên dưới.



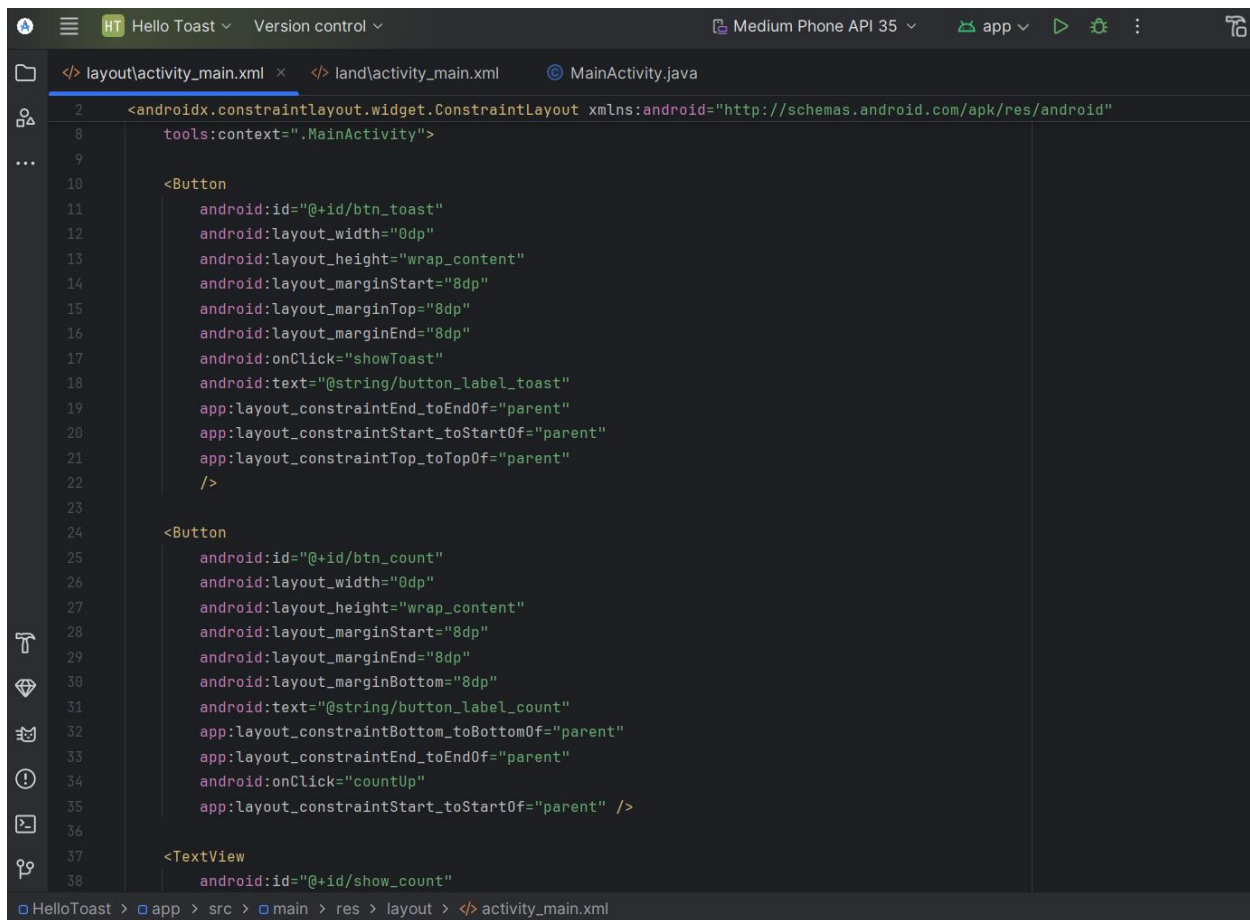
Như đã thể hiện trong các bước trước, các thuộc tính `layout_width` và `layout_height` trong bảng Attributes sẽ thay đổi khi bạn điều chỉnh các điều khiển chiều cao và chiều rộng trong trình kiểm tra. Các thuộc tính này có thể nhận một trong ba giá trị cho bố cục, đó là `ConstraintLayout`:

Nhiệm vụ 4: Thêm `TextEdit` và sửa thuộc tính cho nó


```
<TextView
    android:id="@+id/show_count"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:text="@string/count_initial_value"
    android:textAlignment="center"
    android:textColor="@color/design_default_color_primary"
    android:textSize="160sp"
    android:background="#FFFF00"
    android:gravity="center_vertical"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/btn_count"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/btn_toast"
    tools:ignore="RtlCompat" />
```

Nhiệm vụ 5: Sửa bố cục trong XML

5.1 Mở mã nguồn XML của bố cục



```
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
8     tools:context=".MainActivity">
9
10     <Button
11         android:id="@+id/btn_toast"
12         android:layout_width="0dp"
13         android:layout_height="wrap_content"
14         android:layout_marginStart="8dp"
15         android:layout_marginTop="8dp"
16         android:layout_marginEnd="8dp"
17         android:onClick="showToast"
18         android:text="@string/button_label_toast"
19         app:layout_constraintEnd_toEndOf="parent"
20         app:layout_constraintStart_toStartOf="parent"
21         app:layout_constraintTop_toTopOf="parent"
22     />
23
24     <Button
25         android:id="@+id/btn_count"
26         android:layout_width="0dp"
27         android:layout_height="wrap_content"
28         android:layout_marginStart="8dp"
29         android:layout_marginEnd="8dp"
30         android:layout_marginBottom="8dp"
31         android:text="@string/button_label_count"
32         app:layout_constraintBottom_toBottomOf="parent"
33         app:layout_constraintEnd_toEndOf="parent"
34         android:onClick="countUp"
35         app:layout_constraintStart_toStartOf="parent" />
36
37     <TextView
38         android:id="@+id/show_count"
```

5.2 Trích xuất tài nguyên chuỗi

Thay vì mã hóa cứng các chuỗi, một thực tiễn tốt nhất là sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Việc đặt các chuỗi trong một tệp riêng biệt giúp dễ dàng quản lý chúng hơn, đặc biệt nếu bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và bản địa hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhấp một lần vào từ "Toast" (cảnh báo được đánh dấu đầu tiên).
2. Nhấn **Alt-Enter** trên Windows hoặc **Option-Enter** trên macOS và chọn **Extract string resource** từ menu bật lên.
3. Nhập **button_label_toast** cho **Resource name**.
4. Nhấp vào **OK**. Một tài nguyên chuỗi được tạo trong tệp **values/res/strings.xml**, và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: `@string/button_label_toast`

5. Trích xuất các chuỗi còn lại:

- **button_label_count** cho "Count"
- **count_initial_value** cho "0"

6. Trong **Project > Android** pane, mở rộng **values** trong **res**, và sau đó nhấp đúp vào **strings.xml** để xem tài nguyên chuỗi của bạn trong tệp **strings.xml**.

```
<resources>
  <string name="app_name">Hello Toast</string>
  <string name="button_label_toast">TOAST</string>
  <string name="button_label_count">COUNT</string>
  <string name="count_initial_value">0</string>
</resources>
```

7. Bạn cần một chuỗi khác để sử dụng trong một nhiệm vụ tiếp theo hiển thị thông báo. Thêm vào tệp **strings.xml** một tài nguyên chuỗi khác có tên **toast_message** cho cụm từ "Hello Toast!":

```
</> activity_main.xml  </> strings.xml  x  Translations Editor  © MainActivity
1  <resources>
2      <string name="app_name">Hello Toast</string>
3      <string name="button_label_toast">TOAST</string>
4      <string name="button_label_count">COUNT</string>
5      <string name="count_initial_value">0</string>
6      <string name="toast_message">Hello Toast!</string>
7  </resources>
```

Nhiệm vụ 6: Thêm onClick handlers cho các nút

6.1 Thêm thuộc tính onClick và trình xử lý sự kiện cho từng Button.

Trình xử lý sự kiện nhấp chuột (click handler) là một phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử UI có thể nhấp.

Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường **onClick** trong **tab Design** của **Attributes pane**. Bạn cũng có thể chỉ định tên của phương thức xử lý trong **trình chỉnh sửa XML** bằng cách thêm thuộc tính `android:onClick` vào **Button**.

Bạn sẽ sử dụng phương pháp thứ hai vì bạn chưa tạo các phương thức xử lý, và trình chỉnh sửa XML cung cấp một cách tự động để tạo các phương thức đó.

1. Với trình chỉnh sửa XML đang mở (tab **Text**), tìm thẻ `<Button>` có thuộc tính `android:id` được đặt thành `@+id/button_toast`.
2. Thêm thuộc tính `android:onClick` vào phần tử `button_toast`.
3. Nhấp vào biểu tượng bóng đèn màu đỏ xuất hiện bên cạnh thuộc tính. Chọn **Create click handler**, chọn **MainActivity**, và nhấp vào **OK**.
4. Lặp lại 2 bước trên với Button `button_count`.

```
<Button
    android:id="@+id/btn_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="@string/button_label_toast"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="showToast"
/>

<Button
    android:id="@+id/btn_count"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:text="@string/button_label_count"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:onClick="countUp"/>
```

5. Mở file MainActivity.java

```
package com.example.hellotoast;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        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);
            return insets;
        });
        mShowCount = (TextView) findViewById(R.id.show_count);
    }

    2 usages
    public void showToast(View view) {

    }

    2 usages
    public void countUp(View view) {

    }
}
```

6.2 Sửa trình xử lý sự kiện của nút Toast

Bây giờ bạn sẽ chỉnh sửa phương thức `showToast()`—trình xử lý sự kiện khi nhấn nút **Toast** trong `MainActivity`—để hiển thị một thông báo. Một **Toast** cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ popup nhỏ. Nó chỉ chiếm không gian cần thiết để hiển thị thông báo. Hoạt động hiện tại vẫn hiển thị và có thể tương tác. Một **Toast** có thể hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn—hãy thêm một thông báo **Toast** để hiển thị kết quả khi nhấn một **Button** hoặc thực hiện một hành động.

Làm theo các bước sau:

1. Tìm phương thức `showToast` mới tạo.

```
public void showToast(View view) {  
}
```

2. Để tạo 1 thực thể Toast, gọi đến thuộc tính `makeText` của lớp `Toast`

```
public void showToast(View view) {  
    Toast toast = Toast.makeText();  
}
```

3. Cung cấp **context** của **Activity** cho **Toast**. Vì **Toast** hiển thị trên giao diện **Activity**, hệ thống cần thông tin về **Activity** hiện tại. Khi bạn đang ở trong **Activity** cần truyền context, có thể sử dụng `this` làm một cách viết tắt.

```
public void showToast(View view) {  
    Toast toast = Toast.makeText(context: this,);  
}
```

4. Cung cấp thông điệp để hiển thị, chẳng hạn như một **string resource** (chuỗi tài nguyên) mà bạn đã tạo trước đó, chẳng hạn như `toast_message`. Chuỗi tài nguyên `toast_message` được xác định bằng `R.string.toast_message`.

```
public void showToast(View view) {  
    Toast toast = Toast.makeText(context: this, R.string.toast_message,);  
}
```

5. Cung cấp thời gian hiển thị cho **Toast**. Ví dụ, `Toast.LENGTH_SHORT` sẽ hiển thị **Toast** trong một khoảng thời gian ngắn.

```
public void showToast(View view) {  
    Toast toast = Toast.makeText(context: this, R.string.toast_message, Toast.LENGTH_SHORT);  
}
```

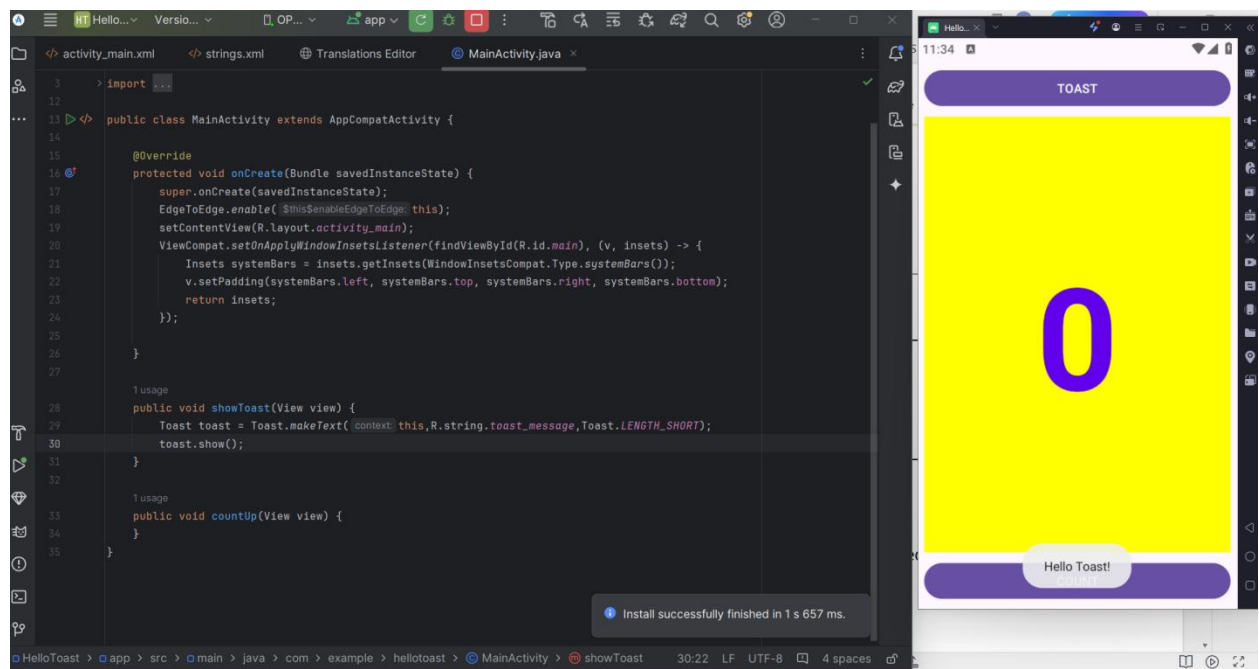
Thời gian hiển thị của **Toast** có thể là `Toast.LENGTH_LONG` hoặc `Toast.LENGTH_SHORT`.

Thời gian thực tế khoảng **3.5 giây** đối với Toast.LENGTH_LONG và **2 giây** đối với Toast.LENGTH_SHORT.

6. Gọi show() để hiển thị **Toast**. Dưới đây là toàn bộ phương thức showToast():

```
public void showToast(View view) {
    Toast toast = Toast.makeText(context, this, R.string.toast_message, Toast.LENGTH_SHORT);
    toast.show();
}
```

Chạy ứng dụng và kiểm tra rằng thông báo **Toast** xuất hiện khi nút **Toast** được nhấn.



6.3 Sửa trình xử lý sự kiện của nút Count

Bây giờ bạn sẽ chỉnh sửa phương thức `countUp()`—trình xử lý sự kiện khi nhấn nút **Count** trong `MainActivity`—để hiển thị số đếm hiện tại sau mỗi lần nhấn **Count**. Mỗi lần nhấn sẽ tăng giá trị đếm lên một.

Mã xử lý sự kiện cần:

- Ghi nhớ số đếm khi thay đổi.
- Cập nhật số đếm lên **TextView** để hiển thị.

Làm theo các bước sau:

1. Tìm phương thức countUp()

```
2 usages
public void countUp(View view) {
}
```

2. Để theo dõi số đếm, bạn cần một biến thành viên **private**. Mỗi lần nhấn nút **Count**, giá trị của biến này sẽ tăng lên. Nhập dòng sau vào **MainActivity**, dòng này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn màu đỏ:

```
2 usages
public void countUp(View view) {
    mCount++;
}
```

Nếu bóng đèn đỏ không xuất hiện, hãy chọn vào biểu thức `mCount++`. Nó sẽ xuất hiện.

3. Nhấp vào biểu tượng **bóng đèn đỏ** và chọn **Create field 'mCount'** từ menu popup. Điều này sẽ tạo một **biến thành viên private** ở đầu **MainActivity**, và **Android Studio** sẽ tự động xác định nó là kiểu **int**.

```
public class MainActivity extends AppCompatActivity {

    1 usage
    private int mCount;
}
```

4. Thay đổi câu lệnh khai báo biến thành viên **private** để khởi tạo biến với giá trị **0**:

```
public class MainActivity extends AppCompatActivity {

    1 usage
    private int mCount = 0;
}
```


5. Cùng với biến trên, bạn cũng cần một **biến thành viên private** để tham chiếu đến **TextView** show_count, biến này sẽ được sử dụng trong trình xử lý sự kiện.

```
public class MainActivity extends AppCompatActivity {  
  
    1 usage  
    private int mCount = 0;  
    1 usage  
    private TextView mShowCount;  
}
```

6. Bây giờ bạn đã có mShowCount, bạn có thể lấy tham chiếu đến TextView bằng cách sử dụng **ID** đã đặt trong tệp bố cục (**layout XML**).

Để tránh lấy tham chiếu nhiều lần, hãy thực hiện trong phương thức onCreate(). Phương thức onCreate() được sử dụng để **inflate** (khởi tạo) bố cục, nghĩa là đặt giao diện màn hình theo tệp **XML layout**. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử UI khác, chẳng hạn như **TextView**.

7. Thêm câu lệnh findViewById vào cuối phương thức:

```
1 usage  
private int mCount = 0;  
1 usage  
private TextView mShowCount;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable(this);  
    setContentView(R.layout.activity_main);  
    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);  
        return insets;  
    });  
    mShowCount = (TextView) findViewById(R.id.show_count);  
}
```

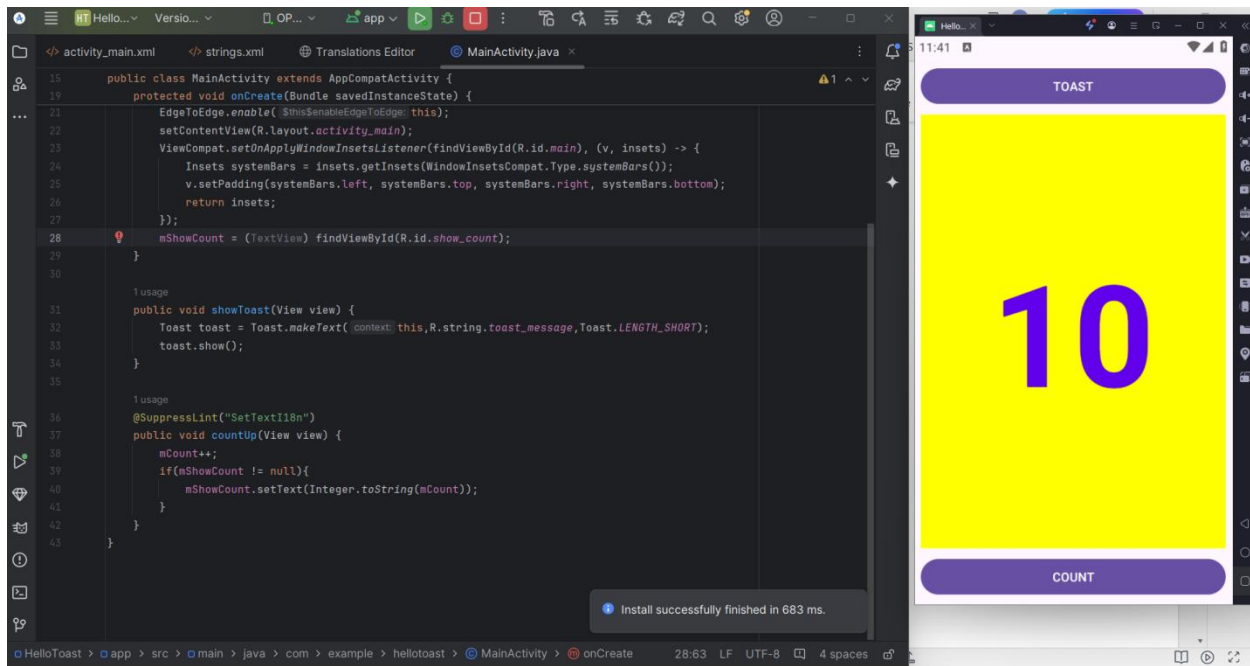
Một **View**, giống như một **chuỗi (string)**, là một **resource** có thể có **ID**.

Phương thức `findViewById()` nhận **ID của View** làm tham số và trả về **View** tương ứng. Vì phương thức này trả về một **View**, bạn cần **ép kiểu (cast)** kết quả thành loại **View** mong đợi, trong trường hợp này là `TextView`.

8. Bây giờ bạn đã gán `mShowCount` cho `TextView`, bạn có thể sử dụng biến này để **cập nhật nội dung** của `TextView` bằng giá trị của biến `mCount`. Thêm đoạn mã sau vào phương thức `countUp()`:

```
2 usages
@SuppressLint("SetTextI18n")
public void countUp(View view) {
    mCount++;
    if(mShowCount != null){
        mShowCount.setText(Integer.toString(mCount));
    }
}
```

9. Chạy ứng dụng và kiểm tra rằng số đếm tăng lên mỗi khi bấm vào nút Count.



- 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

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