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

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.

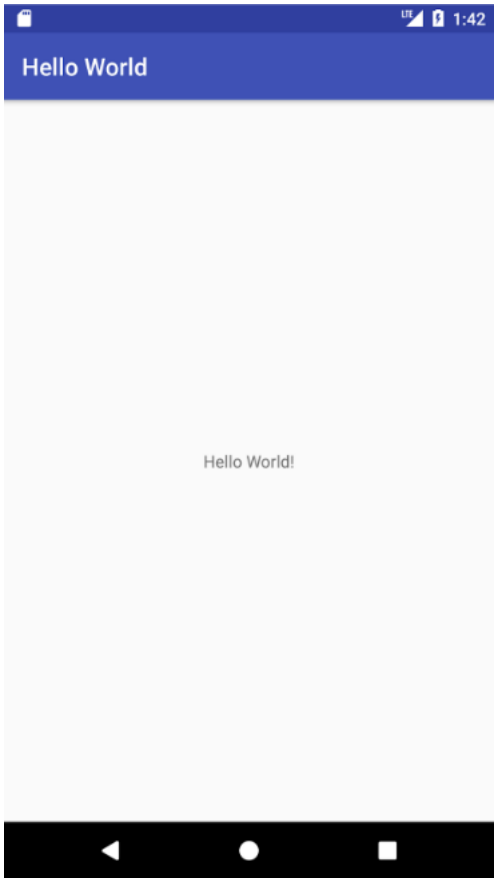
Những gì bạn sẽ làm

- 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**

Tổng quan về ứng dụng

Sau khi cài đặt thành công **Android Studio**, bạn sẽ tạo một dự án mới từ mẫu để phát triển ứng dụng **Hello World**. Đây là một ứng dụng đơn giản hiển thị chuỗi **"Hello World"** trên màn hình của thiết bị Android, có thể là thiết bị ảo (**Android Virtual Device - AVD**) hoặc thiết bị vật lý.

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



Task 1: Cài đặt Android Studio

Android Studio cung cấp một **môi trường phát triển tích hợp (IDE - Integrated Development Environment)** hoàn chỉnh, bao gồm một **trình soạn thảo mã nâng cao** và một tập hợp các **mẫu ứng dụng**. Ngoài ra, nó còn đi kèm với các công cụ hỗ trợ phát triển, **gỡ lỗi (debugging)**, **kiểm thử (testing)** và **tối ưu hóa hiệu suất**, giúp quá trình phát triển ứng dụng nhanh chóng và dễ dàng hơn.

Bạn có thể kiểm thử ứng dụng của mình trên nhiều **trình giả lập (emulator)** được cấu hình sẵn hoặc trên thiết bị di động của chính mình, xây dựng ứng dụng hoàn chỉnh và xuất bản trên **Google Play Store**.

Lưu ý: Android Studio liên tục được cập nhật và cải tiến. Để có thông tin mới nhất về **yêu cầu hệ thống** và **hướng dẫn cài đặt**, hãy truy cập trang web chính thức của Android Studio.

Yêu cầu hệ thống

Android Studio có thể chạy trên các hệ điều hành sau: **Windows**, **Linux**, **macOS**. Phiên bản mới nhất của **OpenJDK (Java Development Kit)** được tích hợp sẵn trong **Android Studio**.

- **Hướng dẫn cài đặt**

1. **Truy cập trang web chính thức của Android Studio** và làm theo hướng dẫn để tải về và cài đặt.
2. **Chấp nhận các cấu hình mặc định** trong tất cả các Task và đảm bảo rằng tất cả các thành phần cần thiết được chọn để cài đặt.
3. **Hoàn tất cài đặt**: Trình hướng dẫn thiết lập (**Setup Wizard**) sẽ tải xuống và cài đặt thêm một số thành phần bổ sung, bao gồm **Android SDK**.
    - Quá trình này có thể mất một khoảng thời gian tùy thuộc vào tốc độ mạng của bạn.
    - Một số Task có thể trông giống nhau, nhưng hãy kiên nhẫn và hoàn thành toàn bộ quá trình.
4. **Sau khi tải xuống hoàn tất**, **Android Studio** sẽ khởi động và bạn đã sẵn sàng tạo dự án đầu tiên của mình.

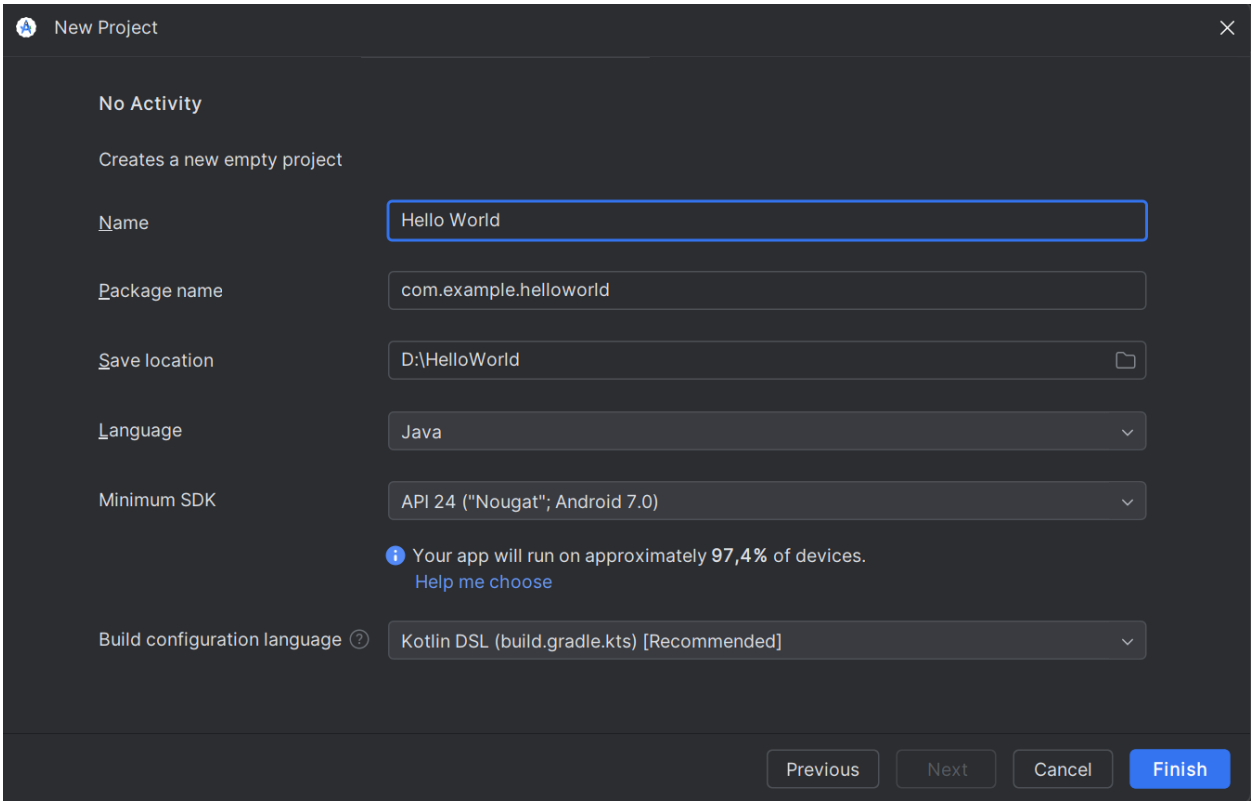
Xử lý sự cố : Nếu gặp sự cố khi cài đặt, bạn có thể kiểm tra **Ghi chú phát hành (Release Notes)** của Android Studio để xem các vấn đề đã biết; nhờ trợ giúp từ **giảng viên** hoặc **cộng đồng lập trình viên**.

Task 2: Tạo ứng dụng Hello World

Trong Task này, bạn sẽ tạo một ứng dụng hiển thị dòng chữ "Hello World" để kiểm tra xem Android Studio đã được cài đặt đúng cách hay chưa, đồng thời làm quen với các Task cơ bản khi phát triển ứng dụng Android.

2.1 Tạo dự án ứng dụng

1. **Mở Android Studio (nếu chưa mở)**.
2. Trong cửa sổ chào mừng **Welcome to Android Studio**, chọn "**Start a new Android Studio project**".
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** vào ô **Application name.**



4. Kiểm tra **Project location** của dự án **Hello World** và các dự án khác trong **Android Studio** hoặc muốn thay đổi, chọn thư mục lưu trữ theo ý bạn.
5. Giữ mặc định **android.example.com** cho **Company Domain** hoặc nhập một tên miền riêng.

   Nếu không có kế hoạch xuất bản ứng dụng, có thể giữ nguyên mặc định. Thay đổi **package name** sau này sẽ tốn thêm công sức.

6. **Bỏ chọn** tùy chọn **Include C++ support** và **Include Kotlin support** => **Next**.
7. Ở màn hình **Target Android Devices**, đảm bảo **Phone and Tablet** được chọn. Đặt **API 15: Android 4.0.3 (Ice Cream Sandwich)** làm **Minimum SDK** nếu chưa được chọn. Cấu hình này giúp ứng dụng tương thích với **97% thiết bị Android** trên **Google Play Store**.
8. Bỏ chọn **Include Instant App support** và các tùy chọn khác. Nhấn **Next**. Nếu cần cài đặt thêm thành phần cho **SDK**, **Android Studio** sẽ tự động tải và cài đặt.
9. Ở màn hình **Add an Activity**, chọn **Empty Activity** => **Next**. An **Activity** là thành phần quan trọng trong mọi ứng dụng Android, thường đi kèm với **layout** để hiển thị giao diện người dùng.
10. Ở màn hình **Configure Activity**, giữ tên mặc định **MainActivity**. Bạn có thể đổi tên, nhưng bài học này sử dụng **MainActivity**.

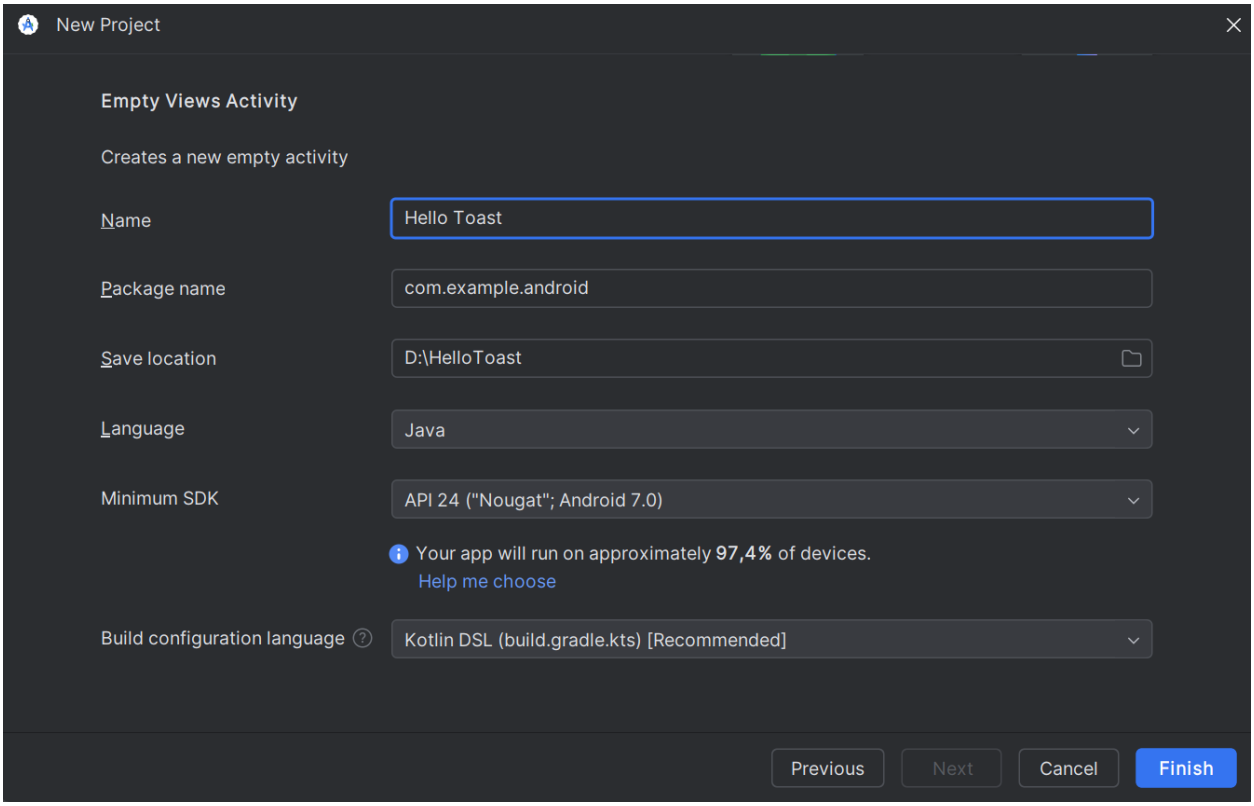
11. Đảm bảo tùy chọn **Generate Layout file** được chọn. Tên mặc định: **activity_main.xml.** Bạn có thể thay đổi, nhưng bài học này sử dụng **activity_main**.

12. Đảm bảo tùy chọn **Backwards Compatibility (AppCompat)** được chọn. Điều này giúp ứng dụng chạy tốt trên các phiên bản Android cũ hơn.

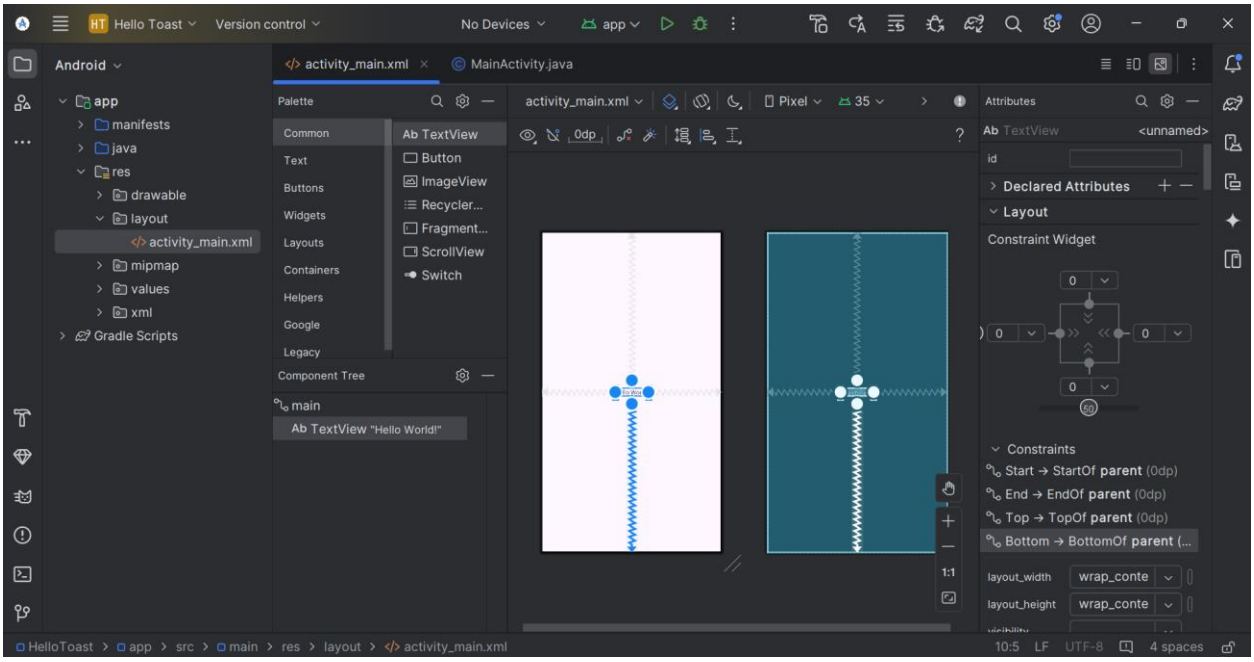
13. Nhấn **Finish** để **Android Studio** khởi tạo dự án.

1.2) Giao diện người dùng tương tác đầu tiên

1.1 Tạo một dự án Android Studio



1.2 Khám phá trình chỉnh sửa giao diện



2: Thêm các phần tử view trong Layout editor

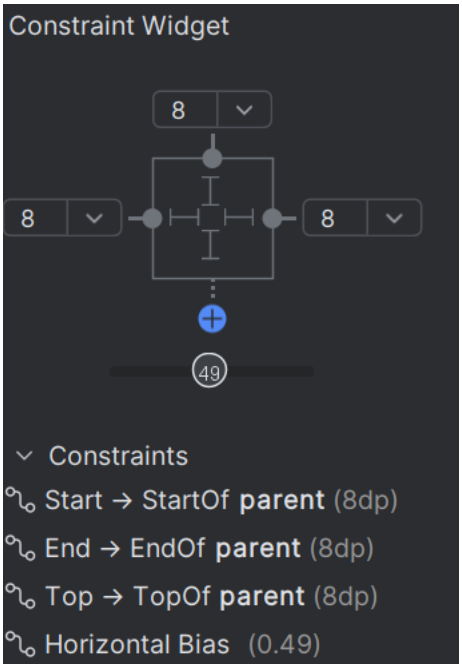
2.2 Thêm một button vào layout

2.3 Thêm button thứ 2 vào layout

Task 3: Thay đổi thuộc tính của phần tử giao diện

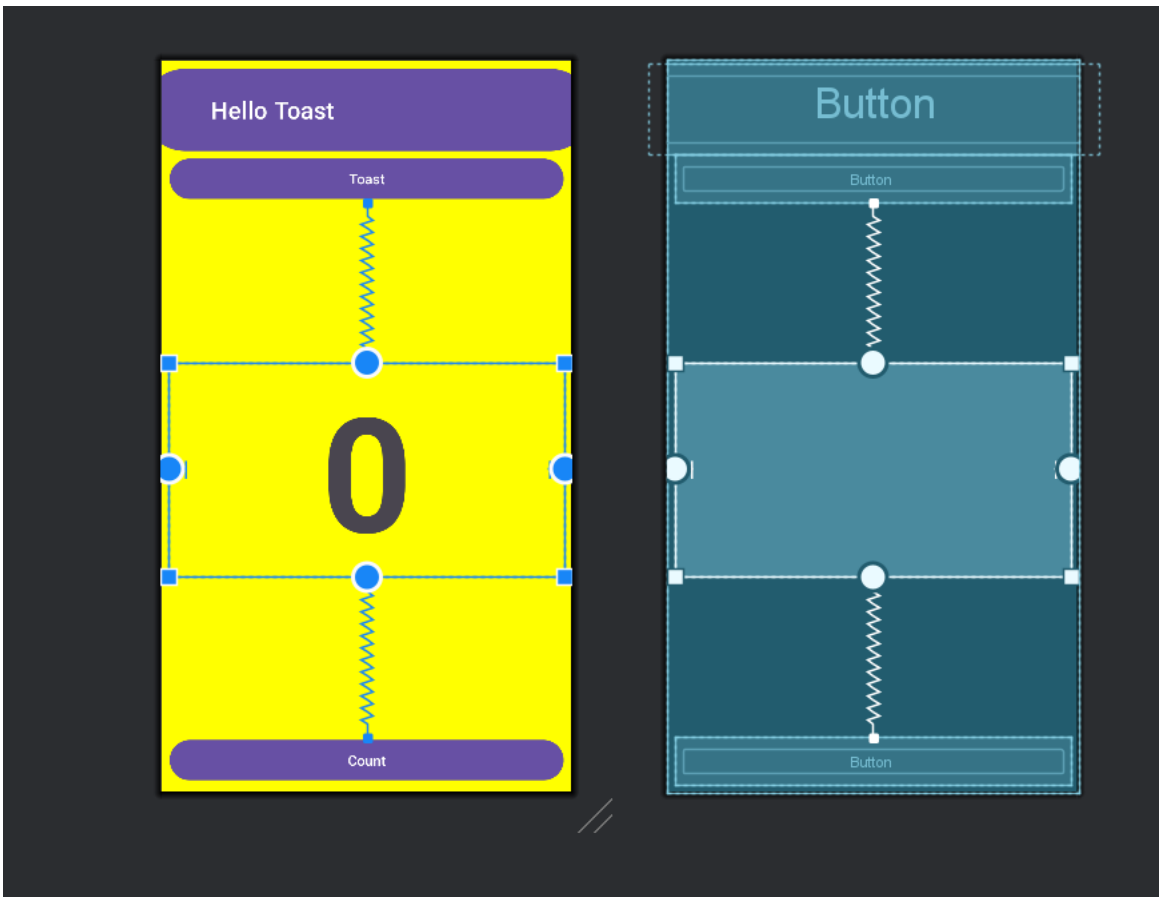
Pane Thuộc tính cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng. Bạn có thể tìm thấy các thuộc tính (được gọi là thuộc tính) chung cho tất cả các view trong tài liệu của lớp View. Trong nhiệm vụ này, bạn sẽ nhập giá trị mới và thay đổi giá trị cho các thuộc tính quan trọng của Button, các thuộc tính này có thể áp dụng cho hầu hết các loại View.

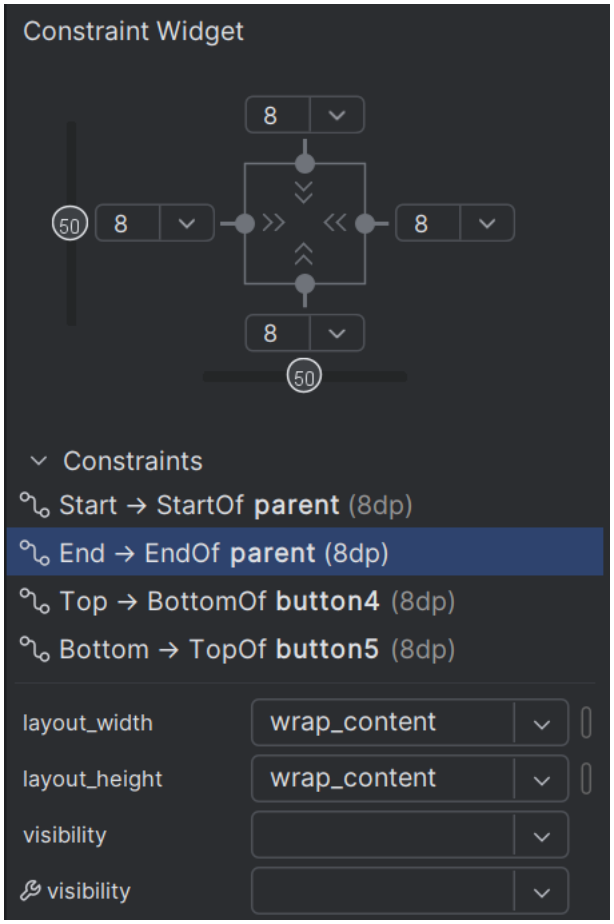
3.1 Thay đổi kích thước button



3.2 Thay đổi thuộc tính của button

Task 4: Thêm TextEdit và set lấy thuộc tính của nó





Task 5: Chỉnh sửa layout trong XML

Bố cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử giao diện người dùng trong Component Tree. Di chuột qua các dấu chấm than này để xem thông báo cảnh báo như hình bên dưới. Cảnh báo giống nhau xuất hiện cho cả ba phần tử: chuỗi được mã hóa cứng nên sử dụng tài nguyên.

Task 6: Thêm onClick (phương thức sự kiện) cho button

6.1 Thêm sự kiện onlick vào 2 button

```
<Button
    android:id="@+id/button5"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:text="Count"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.51"
    app:layout_constraintStart_toStartOf="parent"
    android:onClick="countUp"/>
```

```
<Button
    android:id="@+id/button4"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:text="Toast"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button3"
    android:onClick="showToast"/>
```

2. Tự động tạo Phương thức onClick trong MainActivity.java: Khi nhập **android:onClick="showToast"** trong XML, sẽ xuất hiện biểu tượng bóng đèn đỏ bên cạnh. Nhấp vào biểu tượng bóng đèn đỏ hoặc click vào **"showToast",** sau đó: Chọn **Create click handler**, **Chọn MainActivity**, **Nhấn OK**

=> Làm tương tự cho **countUp:**

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

    1 usage
    public void countUp(View view) {
    }
}
```

6.2 Chỉnh sửa sự kiện cho nút Toast

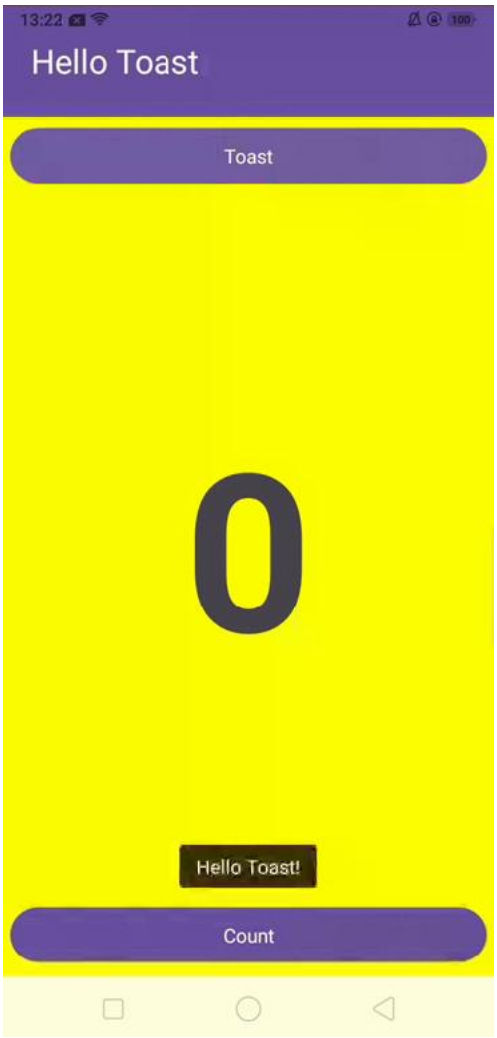
Bây giờ bạn sẽ chỉnh sửa phương thức showToast()—trình xử lý khi nhấn nút Toast trong MainActivity—để hiển thị một thông báo. Toast cung cấp một cách để hiển thị thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ chiếm không gian vừa đủ cho nội dung thông báo. Hoạt động hiện tại vẫn hiển thị và có thể tương tác. Toast có thể hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn - thêm một thông báo Toast để hiển thị kết quả khi nhấn một nút hoặc thực hiện một hành động.

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

Các bước:

1. Xác định vị trí phương thức `showToast()` mới được tạo.
2. Để tạo một instance của `Toast`, gọi phương thức `makeText()` của lớp `Toast`.
3. Cung cấp ngữ cảnh của Activity trong ứng dụng. Vì `Toast` hiển thị trên giao diện của Activity, hệ thống cần biết thông tin về Activity hiện tại. Khi đang ở trong Activity đó, có thể sử dụng `this` làm tham chiếu ngắn gọn.
4. Cung cấp thông điệp cần hiển thị, chẳng hạn như một tài nguyên chuỗi (`string resource`) đã tạo trước đó (`toast_message`). Tài nguyên chuỗi `toast_message` được xác định bằng `R.string.toast_message`.
5. Cung cấp thời gian hiển thị. Ví dụ, `Toast.LENGTH_SHORT` sẽ hiển thị `Toast` trong một khoảng thời gian ngắn.
   Thời gian hiển thị của `Toast` có thể là `Toast.LENGTH_LONG` hoặc `Toast.LENGTH_SHORT`, với thời gian thực tế khoảng 3.5 giây cho `LENGTH_LONG` và 2 giây cho `LENGTH_SHORT`.
6. Hiển thị `Toast` bằng cách gọi phương thức `show()`. Dưới đây là toàn bộ phương thức `showToast()`:

- Chạy app và xuất hiện thông điệp “ Hello Toast! “



6.3 Chỉnh sửa cho nút sự kiện Count

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

Đoạn code phải:
- Theo dõi số đếm khi nó thay đổi.
- Gửi số đếm đã cập nhật đến **`TextView`** để hiển thị.

Các bước:

1. Xác định vị trí phương thức `countUp()` mới được tạo.
2. Để theo dõi số lần nhấn, cần một biến thành viên (`private member variable`). Mỗi lần nhấn nút `Count`, giá trị của biến này sẽ tăng lên. Nhập đoạn mã sau, dòng này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn đỏ:

Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy chọn biểu thức `mCount++`. Biểu tượng bóng đèn sẽ xuất hiện sau đó.

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'mCount'** từ menu bật lên. Thao tác này sẽ tạo một biến thành viên `private` ở đầu `MainActivity`, và Android Studio sẽ tự động giả định kiểu dữ liệu của biến này là **int**.
4. Thay đổi câu lệnh khai báo biến thành viên `private` để khởi tạo giá trị của biến thành **0**.

```
public class MainActivity extends AppCompatActivity {
    2 usages
    private int mCount = 0;
```

5. Bên cạnh biến `mCount`, cần thêm một biến thành viên `private` khác để tham chiếu đến `TextView` có ID `show_count`. Đặt tên biến này là `mShowCount`.

```
public class MainActivity extends AppCompatActivity {
    2 usages
    private int mCount = 0;
    3 usages
    private TextView mShowCount;
```

6. Khi đã có biến `mShowCount`, có thể lấy tham chiếu đến `TextView` bằng cách sử dụng ID đã thiết lập trong tệp layout. Để chỉ lấy tham chiếu này một lần, hãy đặt nó trong phương thức `onCreate()`. Trong một bài học khác, bạn sẽ tìm hiểu rằng phương thức `onCreate()` được sử dụng để **inflate** layout, nghĩa là thiết lập nội dung hiển thị của màn hình dựa trên tệp XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện khác trong layout, chẳng hạn như `TextView`. Xác định vị trí phương thức `onCreate()` trong `MainActivity`.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: 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;
    });
    setContentView(R.layout.activity_main);
    mShowCount = (TextView) findViewById(R.id.show_Count);
}
```

7. Thêm câu lệnh `findViewById` vào cuối phương thức `onCreate()`. Một `View`, giống như một chuỗi (`String`), là một tài nguyên có thể có một **ID**. Lệnh `findViewById` nhận ID của `View` làm tham số và trả về đối tượng `View`. Vì phương thức này trả về một `View`, cần ép kiểu kết quả thành kiểu `TextView`:
8. Bây giờ, khi đã gán `TextView` cho `mShowCount`, có thể sử dụng biến này để **cập nhật nội dung** trong `TextView` với giá trị của biến `mCount`. Thêm đoạn mã sau vào phương thức `countUp()`:

```
public void countUp(View view) {
    mCount++;
    if(mShowCount != null) {
        mShowCount.setText(Integer.toString(mCount));
    }
}
```

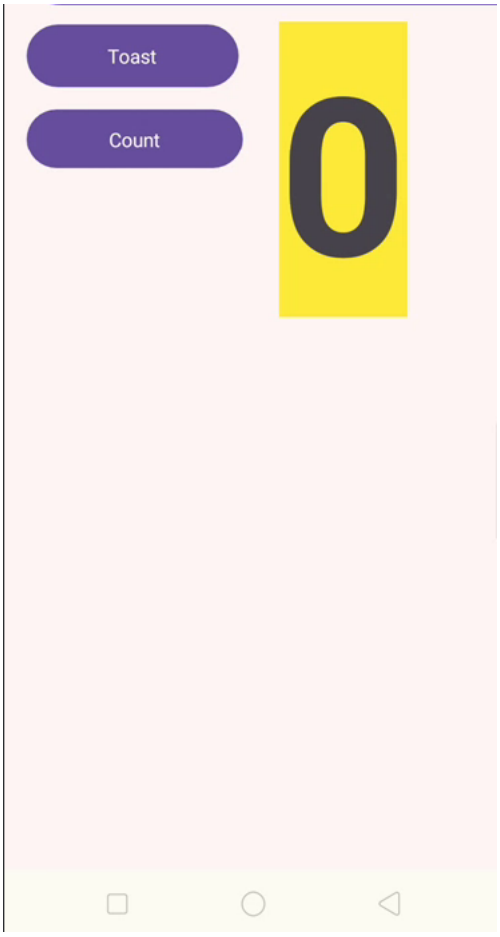
9.Chạy app => nhấn count

Thử thách code

Ứng dụng HelloToast trông ổn khi thiết bị hoặc trình giả lập được đặt theo hướng dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang hướng ngang, nút Count có thể chồng lên TextView ở phía dưới, như minh họa trong hình dưới đây.

Thử thách: Thay đổi bố cục sao cho hiển thị đẹp trong cả hai hướng ngang và dọc:

1. Trên máy tính của bạn, sao chép thư mục dự án **HelloToast** và đổi tên thành **HelloToastChallenge.**
2. Mở **HelloToastChallenge** trong Android Studio và thực hiện refactor. (Xem Phụ lục: Tiện ích để biết hướng dẫn sao chép và refactor một dự án.)
3. Thay đổi bố cục sao cho nút Toast và nút Count xuất hiện ở phía bên trái, như minh họa trong hình dưới đây. TextView xuất hiện bên cạnh chúng, nhưng chỉ rộng đủ để hiển thị nội dung của nó. (Gợi ý: Sử dụng wrap_content.)
4. Chạy ứng dụng trong cả hai hướng ngang và dọc.


Toast
Count
0


Hello Toast
Toast
Count
0

Tóm tắt

View, ViewGroup and layouts:

● Tất cả các phần tử UI đều là các lớp con của lớp View, do đó kế thừa nhiều thuộc tính từ lớp View cha.
● Các phần tử View có thể được nhóm bên trong một ViewGroup, hoạt động như một vùng chứa. Mối quan hệ này là cha-con, trong đó cha là một ViewGroup và con là một View hoặc một ViewGroup khác.

Phương thức onCreate() được sử dụng để inflate layout, nghĩa là thiết lập nội dung hiển thị của màn hình theo tệp XML layout. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các thành phần giao diện người dùng khác trong layout.
● Một View, giống như một chuỗi (string), là một tài nguyên có thể có một id. Lệnh findViewById nhận ID của một View làm tham số và trả về View đó.

Sử dụng trình chỉnh sửa layout:
● Nhấp vào tab Design để thao tác với các thành phần và bố cục, và tab Text để chỉnh sửa mã XML của layout.
● Trong tab Design, ngăn Palettes hiển thị các thành phần UI mà bạn có thể sử dụng trong bố cục ứng dụng của mình, và ngăn Component Tree hiển thị cấu trúc phân cấp của các thành phần UI.
● Các ngăn thiết kế và bản thiết kế trong trình chỉnh sửa layout hiển thị các thành phần UI trong layout.
● Tab Attributes hiển thị ngăn Thuộc tính (Attributes) để thiết lập các thuộc tính cho một thành phần UI.
● Tay cầm ràng buộc (Constraint handle): Nhấp vào tay cầm ràng buộc, được hiển thị dưới dạng vòng tròn ở mỗi bên của một phần tử, sau đó kéo đến một tay cầm ràng buộc khác hoặc đến ranh giới của phần tử cha để tạo ràng buộc. Ràng buộc được biểu thị bằng đường gấp khúc.
● Tay cầm thay đổi kích thước (Resizing handle): Bạn có thể kéo tay cầm thay đổi kích thước hình vuông để thay đổi kích thước phần tử. Khi kéo, tay cầm sẽ thay đổi thành một góc xiên.
● Khi được bật, công cụ Autoconnect tự động tạo hai hoặc nhiều ràng buộc cho một thành phần UI vào bố cục cha. Sau khi kéo phần tử vào bố cục, công cụ này sẽ tạo ràng buộc dựa trên vị trí của phần tử.
● Bạn có thể xóa ràng buộc của một phần tử bằng cách chọn phần tử đó và di chuột qua để hiển thị nút "Clear Constraints". Nhấp vào nút này để xóa tất cả các ràng buộc của phần tử đã chọn. Để xóa một ràng buộc cụ thể, nhấp vào tay cầm ràng buộc tương ứng.
● Ngăn Thuộc tính (Attributes) cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một thành phần UI. Nó cũng bao gồm một bảng định cỡ hình vuông gọi là "view inspector" ở phía trên. Các biểu tượng bên trong hình vuông đại diện cho cài đặt chiều cao và chiều rộng.

.....

1.3) Trình chỉnh sửa bố cục

Giới thiệu

Như bạn đã học trong Phần 1.2: Giao diện người dùng tương tác đầu tiên của bạn, bạn có thể xây dựng giao diện người dùng (UI) bằng cách sử dụng **ConstraintLayout** trong trình chỉnh sửa bố cục (layout editor). ConstraintLayout sắp xếp các thành phần UI trong bố cục bằng cách sử dụng các kết nối ràng buộc (constraint connections) với các phần tử khác và với các cạnh của bố cục. ConstraintLayout được thiết kế để giúp bạn dễ dàng kéo các thành phần UI vào trình chỉnh sửa bố cục.

ConstraintLayout là một ViewGroup, tức là một View đặc biệt có thể chứa các đối tượng View khác (được gọi là con hoặc child views). Bài thực hành này giới thiệu thêm nhiều tính năng của ConstraintLayout và trình chỉnh sửa bố cục.

Bài thực hành này cũng giới thiệu hai lớp con khác của ViewGroup:
● LinearLayout: Một nhóm sắp xếp các thành phần View con theo chiều ngang hoặc chiều dọc.
● RelativeLayout: Một nhóm các thành phần View con mà trong đó mỗi thành phần được định vị và căn chỉnh tương đối với các thành phần View khác trong cùng ViewGroup. Vị trí của các thành phần con được mô tả dựa trên mối quan hệ giữa chúng với nhau hoặc với ViewGroup cha.

Những gì bạn cần biết trước

Bạn cần có khả năng:

● Tạo một ứng dụng "Hello World" bằng Android Studio.
● Chạy ứng dụng trên trình giả lập (emulator) hoặc thiết bị thực tế.
● Tạo một bố cục đơn giản cho ứng dụng bằng ConstraintLayout.
● Trích xuất và sử dụng tài nguyên chuỗi (string resources).

Những gì bạn sẽ học

● Cách tạo một biến thể bố cục cho hướng ngang (landscape).
● Cách tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
● Cách sử dụng ràng buộc đường cơ sở (baseline constraint) để căn chỉnh các thành phần UI với văn bản.
● Cách sử dụng các nút pack và align để căn chỉnh các phần tử trong bố cục.
● Cách định vị các thành phần trong LinearLayout.
● Cách định vị các thành phần trong RelativeLayout.

Những gì bạn sẽ làm

● Tạo một biến thể bố cục cho màn hình có hướng ngang.
● Tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.

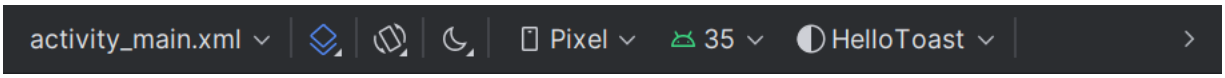
- Chỉnh sửa bố cục để thêm các ràng buộc cho các thành phần UI.
- Sử dụng ràng buộc đường cơ sở của ConstraintLayout để căn chỉnh các phần tử với văn bản.
- Sử dụng các nút pack và align của ConstraintLayout để căn chỉnh các phần tử.
- Thay đổi bố cục để sử dụng LinearLayout.
- Định vị các thành phần trong LinearLayout.
- Thay đổi bố cục để sử dụng RelativeLayout.
- Sắp xếp lại các thành phần trong bố cục chính để chúng tương quan với nhau.

Tổng quan về ứng dụng

Ứng dụng Hello Toast trong bài học trước sử dụng ConstraintLayout để sắp xếp các thành phần UI trong bố cục của Activity, như minh họa trong 1.1

Task 1: Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu thay đổi bố cục của ứng dụng Hello Toast để nó phù hợp trong cả chế độ ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo các biến thể của bố cục cho chế độ ngang (còn gọi là landscape) và chế độ dọc (còn gọi là portrait) trên điện thoại, cũng như cho các màn hình lớn hơn như máy tính bảng.
Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trên hai thanh công cụ trên cùng của trình chỉnh sửa bố cục. Thanh công cụ trên cùng cho phép bạn cấu hình giao diện xem trước của bố cục trong trình chỉnh sửa bố cục.



Trong hình trên:

1. **Chọn Bề mặt Thiết kế**: Chọn *Design* để hiển thị bản xem trước có màu của bố cục hoặc *Blueprint* để chỉ hiển thị các đường viền của từng phần tử UI. Để xem cả hai khung bên cạnh nhau, chọn *Design + Blueprint*.
2. **Hướng trong Trình chỉnh sửa**: Chọn *Portrait* hoặc *Landscape* để hiển thị bản xem trước ở chế độ dọc hoặc ngang. Điều này hữu ích để xem trước bố cục mà không cần chạy ứng dụng trên trình giả lập hoặc thiết bị. Để tạo bố cục thay thế, chọn *Create Landscape Variation* hoặc các biến thể khác.
3. **Thiết bị trong Trình chỉnh sửa**: Chọn loại thiết bị (*điện thoại/máy tính bảng, Android TV hoặc Android Wear*).
4. **Phiên bản API trong Trình chỉnh sửa**: Chọn phiên bản Android để hiển thị bản xem trước.
5. **Chủ đề trong Trình chỉnh sửa**: Chọn một chủ đề (*ví dụ: AppTheme*) để áp dụng cho bản xem trước.
6. **Ngôn ngữ trong Trình chỉnh sửa**: Chọn ngôn ngữ và khu vực cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có sẵn trong tài nguyên chuỗi (*string resources*). Xem bài học về **địa phương hóa** (*localization*) để biết cách thêm ngôn ngữ. Bạn cũng có thể chọn *Preview as Right To Left* để xem bố cục như thể một ngôn ngữ RTL đã được chọn.

Thanh công cụ thứ hai cho phép bạn cấu hình giao diện của các phần tử UI trong *ConstraintLayout*, cũng như phóng to và di chuyển bản xem trước.

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

Giới thiệu

Giao diện người dùng (**UI - User Interface**) hiển thị trên màn hình của thiết bị chạy **Android** được tạo thành từ một **cây phân cấp** gồm các đối tượng gọi là **View**. Mỗi phần tử trên màn hình đều là một **View**.

Lớp **View** là khối xây dựng cơ bản của tất cả các thành phần UI. Đây là lớp cha của các thành phần UI có tính tương tác, chẳng hạn như **Button**. Một **Button** là một thành phần UI mà người dùng có thể nhấn hoặc bấm để thực hiện một hành động nào đó.

Bạn có thể biến bất kỳ **View** nào, chẳng hạn như **ImageView**, thành một phần tử UI có thể được nhấn hoặc bấm. Để làm điều này, bạn cần lưu trữ hình ảnh cho **ImageView** trong thư mục **drawables** của dự án.

Trong bài thực hành này, bạn sẽ học cách sử dụng hình ảnh làm phần tử UI mà người dùng có thể tương tác bằng cách nhấn hoặc bấm.

Kiến thức cần có trước

Bạn nên có khả năng:

- Tạo một dự án **Android Studio** từ mẫu và tạo bố cục (**layout**) chính.
- Chạy ứng dụng trên trình giả lập (**emulator**) hoặc thiết bị kết nối.
- Tạo và chỉnh sửa các thành phần UI bằng **layout editor** và **mã XML**.
- Truy cập các phần tử UI từ mã nguồn bằng **findViewById()**.
- Xử lý sự kiện **click** của **Button**.
- Hiển thị thông báo **Toast**.
- Thêm hình ảnh vào thư mục **drawable** của dự án.

Những gì bạn sẽ học

- Cách sử dụng hình ảnh như một phần tử tương tác để thực hiện một hành động.
- Cách thiết lập thuộc tính cho **ImageView** trong **layout editor**.
- Cách thêm phương thức **onClick()** để hiển thị thông báo **Toast**.

Những gì bạn sẽ làm

- Tạo một dự án **Android Studio** mới cho một ứng dụng mô phỏng đặt món tráng miệng, trong đó sử dụng hình ảnh làm phần tử tương tác.
- Thiết lập trình xử lý **onClick()** cho các hình ảnh để hiển thị các thông báo **Toast** khác nhau.
- Thay đổi **Floating Action Button (FAB)** mặc định của mẫu để hiển thị một biểu tượng khác và mở một **Activity** khác.

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