**Implicit animations**

* Đây là thư viện hỗ trợ animation ngầm hiểu (tức là mặc định có sẵn trong flutter cơ bản: flutter/animation.dart)
* Gọi chung là animation ngầm hoặc widget animation ngầm. Tên gọi được lấy từ class ImplicitlyAnimatedWidget
* Gồm các animation ứng với các class:
  + TweenAnimationBuilder thay đổi một Tween từ 1 giá trị ban đầu đến 1 giá trị cố định:
    - Thay đổi được các giá trị animated property như color, rect, double…
    - Giá trị tween thay đổi thì tạo ra animation thay đổi property của đối tượng
  + AnimatedAlign - ứng với Align class.
    - Thay đổi alignment của đối tượng có animation (di chuyển đối tượng, fade in – fade out….)
  + AnimatedContainer - ứng với  Container.
    - Thay đổi các property của container bằng hiệu ứng.
  + AnimatedDefaultTextStyle - ứng với DefaultTextStyle.
    - Thay đổi style của Text.
  + AnimatedScale - ứng với  Transform.scale.
    - Thay đổi kích thước của đối tượng trong thời gian cho trước
  + AnimatedRotation - ứng với  Transform.rotate.
    - Xoay đối tượng trong thời gian cho trước
  + AnimatedSlide ứng với vị trí của widget relative so với vị trí bình thường
    - Di chuyển đối tượng theo dạng trượt
  + AnimatedOpacity - ứng với  Opacity.
    - Thay đổi độ trong suốt của đối tượng
  + AnimatedPadding - ứng với  Padding.
    - Thay đổi padding của Padding
  + AnimatedPhysicalModel - ứng với  PhysicalModel.
    - Thay đổi các property: borderRadius, Elevation, Color, Curves…
  + AnimatedPositioned - ứng với  Positioned.
    - Thay đổi vị trí của đối tượng
  + AnimatedPositionedDirectional - ứng với  PositionedDirectional.
    - Tương tự positioned nhưng thực hiện với class PositionedDirectional
    - Chỉ dùng được cho child của stack
  + AnimatedTheme - ứng với  Theme class.
    - Thay đổi property của theme như màu, …
  + AnimatedCrossFade
    - Dùng để làm mờ dần 2 child nhất định và tạo animation cho kích thước của child
  + AnimatedSize
    - Dùng trong thay đổi kích thước trong thời gian cho trước
  + AnimatedSwitcher
    - Chuyển từ widget này sang widget khác.

**Hero Animation:**

* Hero Animation là một widget giúp chúng ta tạo các hiệu ứng khi chuyển màn.
* Hero Animation sử dụng một biểu tượng hiện được gọi là “Hero” và khi quá trình chuyển đổi trang được kích hoạt, thường bằng cách nhấp vào biểu tượng, trên màn hình sẽ xuất hiện hoạt ảnh “bay” đến trang tiếp theo.
* Khi người dùng điều hướng trở lại trang trước đó, hoạt ảnh sẽ đi theo hướng khác và biểu tượng quay trở lại vị trí ban đầu của nó.
* Thường có 2 loại hero animation hay được sử dụng:
  + Standard hero animations:
    - Làm 1 hero (hình ảnh, icon..) bay từ 1route tới 1 route khác (có thể hiểu đơn giản là từ một trang này tới 1 trang khác).
    - Thường hero nằm ở vị trí khac 1nhau và khác kích thước
  + Radial hero aniimations:
    - Tương tự như standard nhưng trong quá trình “bay” thì shape của hero sẽ thay đổi từ tròn sang chữ nhật
* Cấu trúc cơ bản:
  + Có 2 Hero widget:
    - Widget ở route ban đầu, route gốc (tạm gọi là source route)
    - Widget ở route sau khi diễn ra animation, route đích ( des route)
* Cấu trúc code:

1. Khai báo source hero widget, các tính chất của widget, khai báo một “identifying tag”. Widget này được khai báo và hiển thị ở source route
2. Khai báo ending hero Widget. Nó phải được khai báo tag giống như source hero widget (thường là đối tượng đại diện cho dữ liệu cơ bản). Để kết quả đẹp nhất thì nên khai báo widget trees tương tự nhau
3. Tạo des route.
4. Trigger animation bằng cách dùng navigator stack. Navigator push và pop sẽ trigger hero animation với từng cặp hero widget matching tags với nhau ở source và des.

* Cơ chế hoạt động:
  + Graphical user interface, application

    Description automatically generatedTrước khi animation diễn ra:
  + Trong lúc diễn ra animation:

Graphical user interface, application, Word

Description automatically generated

* + Sau khi animation kết thúc

Graphical user interface, application, Word

Description automatically generated

* Class cần thiết:
  + Hero: widget sẽ “bay” từ source tới des. Phải chung tag
  + Inkwell: quản lí onTap() để push và pop.
  + Navigator: dùng chuyển giữa các route
  + Route: Quản lí screen or page.
* Tạo một standard hero animation:
  + Bước 1: tạo widget trong đó có dùng Hero widget
  + Bước 2: tạo trang source route, trang này có sử dụng widget vừa tạo ở trên
  + Bước 3: tạo trang des route, có sử dụng widget vừa tạo ở trên, lưu ý tag phải trùng với widget của source route
  + Bước 4: Dùng navigator push và pop để chuyển qua lại giữa 2 route, hero animation sẽ tự match với nhau
* Tạo một radial hero animation:
  + Bước 1: Tạo radial expansion class để quy định cách flip hero trong lúc diễn ra animation.
  + Bước 2: Tạo hero animation được bọc trong radial expansion vừa tạo.
  + Bước 3: Tạo trang source route, trang này có sử dụng widget vừa tạo ở trên
  + Bước 4: Tạo trang des route, có sử dụng widget vừa tạo ở trên, lưu ý tag phải trùng với widget của source route
  + Bước 5: Dùng navigator push và pop để chuyển qua lại giữa 2 route, hero animation sẽ tự match với nhau