BÁO CÁO TUẦN 5

Họ và tên: Nguyễn Kỳ Anh
MSSV: 20225793
GR1
Giáo viên hướng dẫn: Nguyễn Đức Toàn

1. Container

- Container là một Widget đơn giản, giống như một hộp (box) bao quanh một Widget con (child). Cho phép tùy chỉnh giao diện và bố cục của Widget con thông qua các thuộc tính.
  - Kích thước (width, height)
  - Màu nền (color)
  - Viền (border)
  - Đệm (padding) và lề(margin)
  - Căn chỉnh (alignment)
  - Hình dàng (shape) và hiệu ứng (làm bóng – boxShadow)

  VD:

```
Container(
 width: 100.0, // Chiều rộng
 height: 100.0, // Chiều cao
 color: Colors.blue, // Màu nền
 padding: EdgeInsets.all(16.0), // Đệm bên trong
 margin: EdgeInsets.all(16.0), // Lề bên ngoài
 alignment: Alignment.center, // Căn chỉnh widget con
 child: Text('Hello, Flutter!'), // Widget con
)
```

1.1 Các thuộc tính của Container

a. Kích thước (width , height)

- Width: Chiều rộng của Container (giá trị kiểu double).
- Height: Chiều cao của Container (giá trị kiểu double).
- Nếu không chỉ định, Container sẽ mặc đinh chiếm hết không gian có sẵn hoặc phù hợp với Widget con

VD:

```
Container(
  width: 100.0, // Chiều rộng
  height: 100.0, // Chiều cao
);
```

b. Màu sắc (color , decoration)

- Color: đặt màu nền cho Container. Chỉ sử dụng khi không sử dụng decoration.
- Decoration: Một đối tượng như BoxDecoration cho phép tùy chỉnh nâng cao như:

- color: Màu nền.
- border: Viền (Border).
- borderRadius: Bo góc (BorderRadius).
- boxShadow: Hiệu ứng bóng.
- gradient: Hiệu ứng chuyển màu.
- image: Hình nền.

- Chú ý: không được sử dụng đồng thời color và decoration.

VD:

```
Container(
  decoration: BoxDecoration(
    color: Colors.blue,
    borderRadius: BorderRadius.circular(10.0),
    boxShadow: [
      BoxShadow(
        color: Colors.black26,
        blurRadius: 10.0,
        offset: Offset(0, 5),
      ),
```

```
    ],
  ),
)
```

c. Đệm và Lề (padding, margin)

- padding: Khoảng cách bên trong giữa nội dung (child) và viền của Container.
- margin: Khoảng cách bên ngoài giữa Container và các widget khác.
- Cả hai đều sử dụng EdgeInsets để định nghĩa:

- EdgeInsets.all(value): Đặt cùng giá trị cho tất cả các cạnh.
- EdgeInsets.symmetric(horizontal: value, vertical: value): Đặt theo chiều ngang và dọc.
- EdgeInsets.only(left: value, top: value, right: value, bottom: value): Đặt riêng cho từng cạnh.

VD:

```
Container(
padding : EdgeInsets.all(20), // khoảng cách giữa các nội dung là 20px
margin : EdgeInsets.all(15), //khoảng cách giữa các widget khác là 20px
)
```

d. Căn chỉnh (alignment)

- Alignment : dùng để căn chỉnh các widget con trong container.
- Các cách sử dụng Alignment:

- Alignment.center : căn giữa
- Alignment.topLeft : Căn góc trái trên màn hình
- Alignment.bottomRight: Căn góc dưới bên phải màn hình
  - Hoặc sử dụng Alignment(x, y) với x, y từ -1.0 đến 1.0 để căn chỉnh tùy chỉnh.

VD:

```
Container (
alignment : Alignment.Center,// căn phải
alignment : Alignment(-1.0,1.0),
);
```

e. Widget con (child)

- child: Widget được chứa bên trong Container. Có thể là bất kỳ widget nào như Text, Image, Column, Row, v.v.
- Nếu không có child, Container sẽ hiển thị như một hộp rỗng với các thuộc tính được định nghĩa.

VD:

```
Container(
```

```
child: Text("abc"),
);
```

f. Hạn chế (constraints)

- constraints: Giới hạn kích thước của Container bằng BoxConstraints.

VD:

```
Container(
  constraints: BoxConstraints(
    minWidth: 100.0,
    maxWidth: 200.0,
    minHeight: 50.0,
    maxHeight: 150.0,
  ),
);
```

g. Transform

- transform: Áp dụng các biến đổi (transform) như xoay, di chuyển, hoặc co giãn Container.
- Sử dụng Matrix4 để định nghĩa biến đổi.

VD:

```
Container(
  transform: Matrix4.rotationZ(0.1), // Xoay 0.1 radian
  child: Text('Rotated Container'),
)
```

h. Hình dạng (shape)

Dùng trong decoration để định nghĩa hình dạng khác ngoài hình chữ nhật mặc định, ví dụ:

- BoxShape.circle: Container hình tròn.
- BoxShape.rectangle: Hình chữ nhật (mặc định).

VD:

```
Container(
shape: BoxShape.circle,// hình tròn
);
```

1.2 Ứng dụng của Container.

- Tạo khối giao diện: Container thường được sử dụng để tạo các khối giao diện như nút, thẻ, hoặc vùng nội dung.
- Bố cục linh hoạt: Kết hợp với padding, margin, và alignment để căn chỉnh giao diện.

- Tùy chỉnh giao diện: Dùng decoration để tạo các hiệu ứng như bo góc, bóng, hoặc gradient.
- Chứa các widget khác: Container là một widget bao bọc phổ biến cho Text, Image, Row, Column, v.v.

2. SizedBox.

- SizedBox là một Widget dung để:
- Đặt khích thước cụ thể (chiều rông width và chiều cao height) cho Widget con (child).
- Tạo khoảng cách (spacer) giữa các Widget mà không cần nội dung.

VD:

```
SizedBox(
  width: 100.0, // Chiều rộng
  height: 50.0, // Chiều cao
  child: Text('Hello, Flutter!'),
)
```

2.1 Các thuộc tính của SizedBox.

a. Kích thước (width, height)

- width: Chiều rộng của SizedBox (giá trị kiểu double).
- height: Chiều cao của SizedBox (giá trị kiểu double).
- Nếu không đặt width hoặc height, SizedBox sẽ lấy kích thước mặc định từ widget con hoặc không chiếm không gian (nếu không có child).

b. Widget con (child)

- child: Widget được chứa bên trong SizedBox. Nếu có child, SizedBox sẽ ép widget con tuân theo kích thước được chỉ định bởi width và height.
- Nếu không có child, SizedBox hoạt động như một khoảng trống (spacer) với kích thước được định nghĩa.

VD:

```
SizedBox(
child : Text(“data”),
```

);

c. SizedBox mặc định .

- Nếu chỉ đặt width hoặc height mà không có child, SizedBox tạo ra một khoảng cách tương ứng.
- Nếu không đặt kích thước và không có child, SizedBox không chiếm không gian

2.2. Cách sử dụng SizedBox.

a. Tạo khoảng cách giữa các Widget.

- SizedBox thường được sử dụng để tạo các khoảng cách giữa các Widget trong một Column , Row hoặc các bố cục khác.

VD:

```
Column(
  children: [
    Text('First Text'),
    SizedBox(height: 20.0), // Khoảng cách 20 pixel
    Text('Second Text'),
  ],
)
```

b. Đặt một kích thước cố định cho Widget con.

- SizedBox có thể được sử dụng để ép một Widget con có kích thước cụ thể.

VD:

```
SizedBox(
  width: 200.0,
  height: 50.0,
  child: ElevatedButton(
```

```
  onPressed: () {},
  child: Text('Click Me'),
 ),
)
```

c. Tạo vùng trống ổn định

- Nếu không cần Widget con, SizedBox có thể được tạo một vùng trống với kích thước cụ thể.

VD:

```
Row(
 children: [
  Icon(Icons.star),
  SizedBox(width: 10.0), // Khoảng cách ngang 10 pixel
  Text('Rating'),
 ],
)
```

d. Các phương thức tiện ích

- SizeBox.expan: Tạo một SizedBox chiếm toàn bộ không gian có sẵn của Widget cha.

VD:

```
SizedBox.expand(
 child: Container(color: Colors.blue),
)
```

- SizedBox.shrin : Tạo một SizedBox không chiếm không gian (kích thước bằng 0).

VD:

SizedBox.shrink(),

- SizedBox.fromSize: Tạo SizedBox với kích thước được chỉ định bằng một đối tượng Size.

VD:

```
SizedBox.fromSize(
  size: Size(100.0, 50.0),
  child: Text('Hello'),
)
```

3. Row

- Row được sử dụng để sắp xếp các Widget theo chiều ngang.

VD:

```
Row(
  mainAxisAlignment: ..., // canh chỉnh theo chiều ngang (main axis)
  crossAxisAlignment: ..., // canh chỉnh theo chiều dọc (cross axis)
  children: [
    Widget1(),
    Widget2(),
    ...
  ],
)
```

❖ **Các thuộc tính quan trọng của Row**

- mainAxisAlignment - Căn chỉnh theo chiều ngang.

VD:

```
Row (
MainAxisAlignment.start       // Canh trái (mặc định)
MainAxisAlignment.center      // Canh giữa
MainAxisAlignment.end         // Canh phải
MainAxisAlignment.spaceBetween // Khoảng cách đều, không có khoảng ở đầu/cuối
MainAxisAlignment.spaceAround  // Khoảng cách đều, có khoảng đầu/cuối
MainAxisAlignment.spaceEvenly  // Khoảng cách đều nhau giữa tất cả các phần tử
Children: [],
)
```

- crossAxíAlignment – căn chỉnh theo chiều dọc

VD:

```
Row(
CrossAxisAlignment.start    // Canh trên
CrossAxisAlignment.center   // Canh giữa (theo chiều dọc)
CrossAxisAlignment.end      // Canh dưới
Children: []
)
```

- mainAxisSize – Kích thước hàng theo chiều ngang.

VD:

```
Row(
MainAxisSize.max  // Chiếm hết chiều ngang
MainAxisSize.min  // Vừa đủ để chứa các widget con
```

```
Children: [],
)
```

4. Column

- Column dung để sắp xếp các Widget theo chiều dọc.

VD:

```
Column(
 mainAxisAlignment: ..., // canh chỉnh theo chiều dọc (main axis)
 crossAxisAlignment: ..., // canh chỉnh theo chiều ngang (cross axis)
 children: [
  Widget1(),
  Widget2(),
  ...
 ],
)
```

❖ **Các thuộc tính của Column**

- mainAxisAlignment - Căn chỉnh theo chiều ngang.

VD:

```
Column(
MainAxisAlignment.start       // Canh trái (mặc định)
MainAxisAlignment.center      // Canh giữa
MainAxisAlignment.end         // Canh phải
MainAxisAlignment.spaceBetween // Khoảng cách đều, không có khoảng ở đầu/cuối
```

MainAxisAlignment.spaceAround // Khoảng cách đều, có khoảng đầu/cuối

MainAxisAlignment.spaceEvenly // Khoảng cách đều nhau giữa tất cả các phần tử

Children: [],

)

- crossAxíAlignment – căn chỉnh theo chiều dọc

VD:

Column(

CrossAxisAlignment.start   // Canh trên

CrossAxisAlignment.center   // Canh giữa (theo chiều dọc)

CrossAxisAlignment.end   // Canh dưới

Children: []

)

- mainAxisSize – Kích thước hàng theo chiều ngang.

VD:

Column (

MainAxisSize.max // Chiếm hết chiều ngang

MainAxisSize.min // Vừa đủ để chứa các widget con

Children: [],

)

5. Expanded

- Expended là một Widget thuộc nhóm layout widgets trong Flutter, được sử dụng để kiểm soát các widget con phân bổ không gian trong bố cục cha (Row, Column , hoặc Flex).

- Nó chỉ hoạt động trong các widget bố cục Flex (như Row, Column), vù chúng có khái niệm main axis sau khi các widget khác không sử dụng Expanded) đã được phân bổ kích thước cố định

VD:

```
Column(
 children: [
  Expanded(
   child: Container(color: Colors.blue),
  ),
  Expanded(
   child: Container(color: Colors.red),
  ),
 ],
)
```

❖ **Cách hoạt động.**

- Expanded làm cho wedget con mở rộng để lấp đầy không gian trống trong Row hoặc Column.
- Nếu có nhiều Expanded trong cùng một bố cục, không gian trống sẽ được chia đều hoặc chia theo tỷ lệ dựa trên thuộc tính flex.

❖ **Thuộc tính chính.**

- child: Widget con mà Expanded sẽ mở rộng. Đây là widget sẽ lấp đầy không gian trống.
- flex: Một số nguyên (mặc định là 1) xác định tỷ lệ phân chia không gian trống giữa các Expanded trong cùng một bố cục.

Vd: nếu có hai Expanded với flex:1 và flex:2, thì widget thứ 2 sẽ gấp đôi không gian so với widget thứ nhất.

VD:

```
Row(
 children: [
```

```
  Expanded(
    flex: 1,
    child: Container(color: Colors.blue),
  ),
  Expanded(
    flex: 2,
    child: Container(color: Colors.red),
  ),
 ],
)
```

6. Stack

- Stack là một widget bố cục cho phép xếp chồng các widget con theo thứ tự (Z – index).
- Các widget con trong Stack có thể được đặt ở vị trí tương đối hoặc tuyệt đối trong không gian của Stack.
- Stack thường được sử dụng khi bạn cần các widget chồng lên nhau, ví dụ:

- Văn bản hiển thị trên hình ảnh.
- Các nút hoặc biểu tượng nổi trên giao diện.
- Hiệu ứng lớp phủ hoặc giao diện tùy chỉnh.

VD:

```
Stack(
 alignment: Alignment.center,
 children: [
  Container(
    width: 200,
    height: 200,
    color: Colors.blue,
  ),
```

```
      Container(
        width: 100,
        height: 100,
        color: Colors.red,
      ),
    ],
  )
```

6.1. Cách hoạt động.

- Stack quản lý một danh sách các widget con (children) và hiển thị chúng theo thứ tự thêm vào:
- Widget đầu tiên trong danh sách nằm ở dưới cùng (z-index thấp nhất).
- Widget cuối cùng trong danh sách nằm ở trên cùng (z-index cao nhất).
- Stack có thể điều chỉnh kích thước dựa trên widget con lớn nhất hoặc dựa trên kích thước của widget cha.
- Các widget con có thể được định vị tự do bằng cách sử dụng các widget như Positioned hoặc để Stack tự động sắp xếp.

6.2. Thuộc tính chính.

- Children: Danh sách các widget con được xếp chồng lên nhau.
- Alignment: Căn chỉnh các widget con không được định vị (non – positioned) trong Stack. Mặc định là AlignmentDirectional.topStart.

VD: Alignment.center , Alignment.bottomRight.

- fit: Xác định cách các widget con không được vị mở rộng trong Stack. Các giá trị:

- StackFit.loose: Widget con giữ kích thước tự nhiên.
- StackFit.expand: Widget con mở rộng để lấp đầy Stack.
- StackFit.passthrough: Widget con giữ kích thước tự nhiên và không bị ảnh hưởng bởi Stack.

- clipBehavior: Xác định cách xử lý khi widget con vượt ra ngoài ranh giới của Stack. Các giá trị:

- Clip.hardEdge: Cắt bỏ phần vượt ra ngoài.
- Clip.antiAlias: Cắt với hiệu ứng mịn.
- Clip.none: Không cắt (mặc định).
    - textDirection: Hướng văn bản, ảnh hưởng đến căn chỉnh của alignment khi sử dụng AlignmentDirectional.

6.3. Positioned.

- Positioned là một widget đặc biệt được sử dụng trong Stack để định vị chính xác widget con tại các vị trí cụ thể.
- Thuộc tính của Positioned:

- top, bottom, left, right: Khoảng cách từ các cạnh tương ứng của Stack.
- width, height: Kích thước cụ thể của widget con.

- Positioned chỉ hoạt động bên trong Stack. Nếu sử dụng ở nơi khác, Flutter sẽ báo lỗi.

VD:

```
Stack(
  children: [
    Container(color: Colors.blue, width: 200, height: 200),
    Positioned(
      top: 50,
      left: 50,
      child: Container(color: Colors.red, width: 100, height: 100),
    ),
  ],
)
```

6.4 Các trường hợp sử dụng.

- Hiển thị văn bản trên hình ảnh

VD:

```
Stack(
  children: [
    Image.network('https://example.com/image.jpg'),
    Positioned(
      bottom: 10,
      left: 10,
      child: Text(
        'Caption',
        style: TextStyle(color: Colors.white, fontSize: 20),
      ),
    ),
  ],
)
```

- Tạo nút nổi (FAB)

VD:

```
Stack(
  children: [
    Container(color: Colors.grey, width: 300, height: 300),
    Positioned(
      bottom: 20,
```

```
    right: 20,
    child: FloatingActionButton(
     onPressed: () {},
     child: Icon(Icons.add),
    ),
   ),
  ],
)
```

- Tạo hiệu ứng lớp phủ (overlay)

VD:

```
Stack(
  children: [
   Container(color: Colors.black),
   Container(
    color: Colors.black54,
    child: Center(child: Text('Overlay', style: TextStyle(color: Colors.white))),
   ),
  ],
)
```