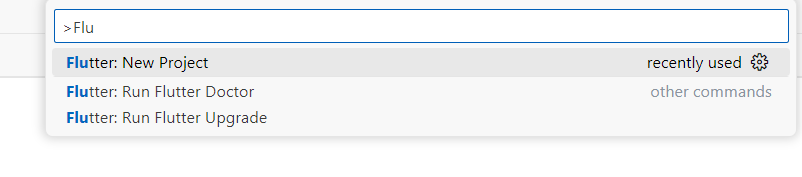
**ใบงานที่ 3**

**ชื่อ - นามสกุล..........นายศรสิวะพงษ์....โพธิวงศ์.......รหัสนักศึกษา...........................2661031741113............................**

1. เปิด VsCode ขึ้นมาสร้าง Flutter Project ใหม่โดยการกดคีย์ Ctrl+Shift+P เพื่อเปิด Command Palette จากนั้นพิมพ์ Flutter และเลือก Flutter : New Project

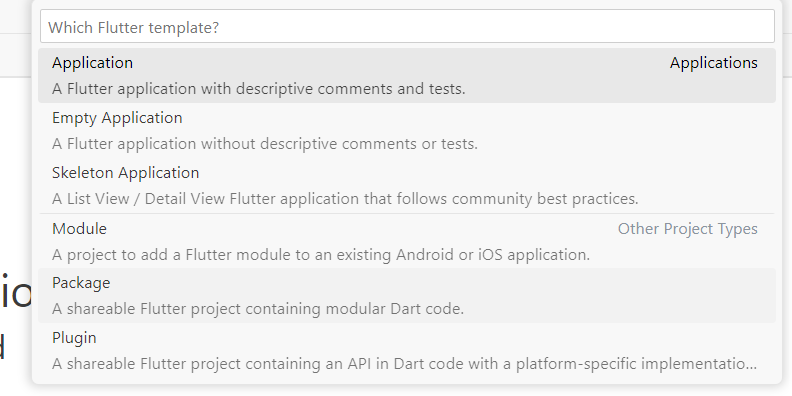


Flutter

1

2

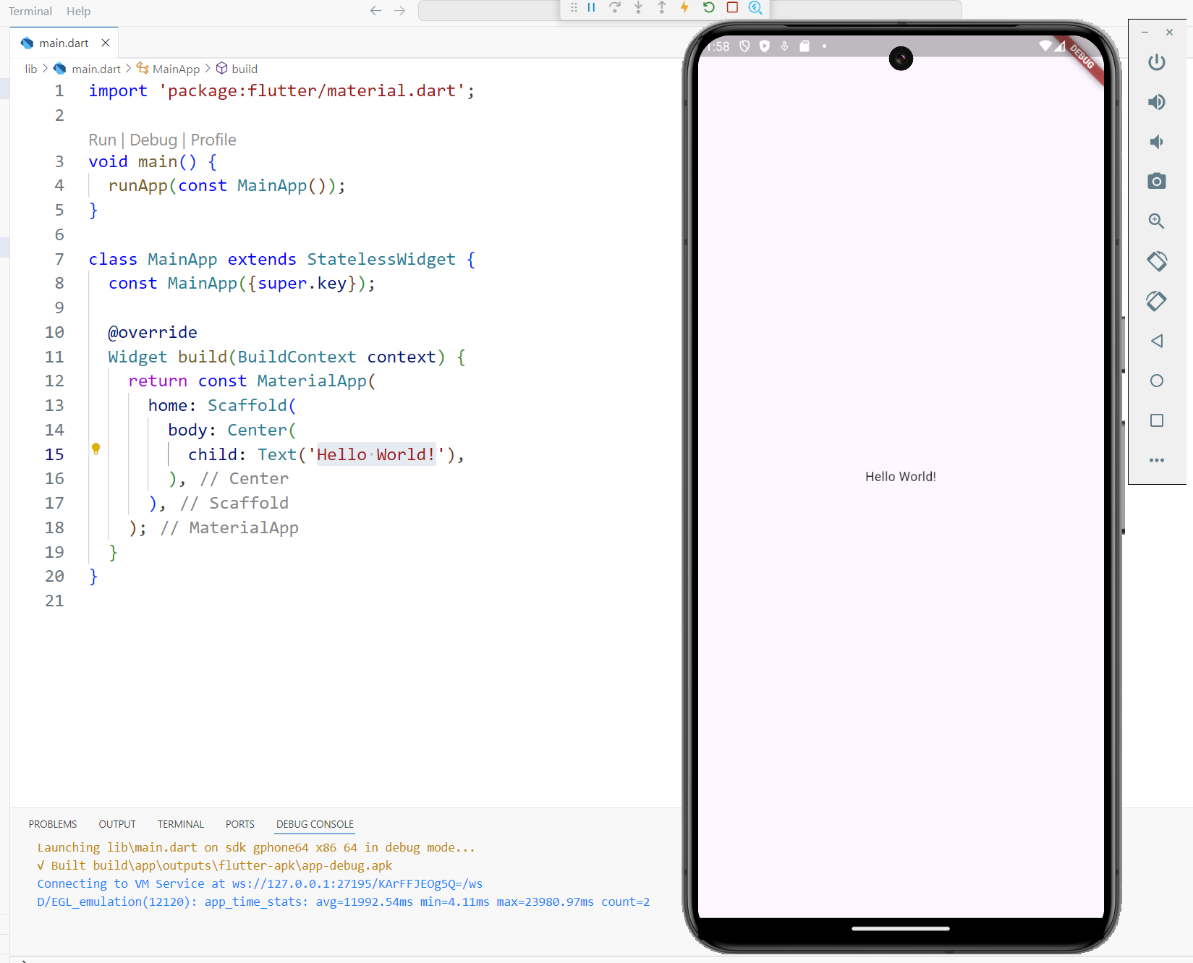
1. จากนั้นเลือก Flutter Template เป็น Empty Application



1. เลือกที่เก็บไฟล์ในขั้นตอนนี้ Flutter จะให้เราเลือกว่าจะให้เก็บ Folder Project ที่ไหนเมื่อเลือกแล้ว VsCode จะให้เราตั้งชื่อโปรเจค ซึ่งจะเป็นชื่อของ Folder ของโปรเจคนั้นๆ ด้วย ใช้ใบงานนี้ให้ตั้งชื่อโปรเจคเป็น mobilelab1cetรหัสนักศึกษา 3 หลักหลัง



1. เมื่อสร้างโปรเจคเสร็จให้นักศึกษาทดลองสั่งรันโปรแกรมโดยการเลือก Device ก่อนที่มุมขวาล่าง เมื่อ Device บูตเสร็จให้กลับมาที่ไฟล์ main.dart และกดคีย์ F5 รอจนมีข้อความ “Hello World!” แสดงที่จอของ Device สิ่งนี้เป็นการยืนยันว่าโปรเจคเราสร้างสำเร็จและสามารถรันโค๊ดได้ปกติ



ถ้าสามารถรันได้ตามปกติจะพบข้อความ Hello World!

**import:**

import 'package:flutter/material.dart';

บรรทัดนี้โหลดไลบรารี Flutter Material Design ซึ่งจำเป็นสำหรับการสร้างส่วนติดต่อผู้ใช้ (UI) ของแอปพลิเคชั่น

**main():**

void main() {

  runApp(const MainApp());

}

ฟังก์ชัน main() เป็นจุดเริ่มต้นของโปรแกรม Flutter ฟังก์ชันนี้จะเรียกใช้ runApp() ซึ่งรับวิดเจ็ต MainApp ของเราเป็นอาร์กิวเมนต์

**MainApp:**

class MainApp extends StatelessWidget {

  const MainApp({super.key});

  @override

  Widget build(BuildContext context) {

~~return const MaterialApp(~~

~~home: Scaffold(~~

~~body: Center(~~

~~child: Text('Hello World!'),~~

        ),

      ),

    );

  }

}

คลาส MainApp สืบทอดมาจาก StatelessWidget ซึ่งหมายความว่า UI ของวิดเจ็ตนี้จะไม่เปลี่ยนแปลงตามสถานะ

* const MainApp({super.key});: บรรทัดนี้สร้างตัวสร้างวิดเจ็ตคงที่ ซึ่งหมายความว่าวัตถุ MainApp ใหม่จะไม่ถูกสร้างทุกครั้งที่มีการเรียกใช้ build()
* @override Widget build(BuildContext context);: เมธอด build() เป็นเมธอดที่สำคัญที่สุดของ StatelessWidget   
  เมธอดนี้รับอ็อบเจ็กต์ BuildContext เป็นอาร์กิวเมนต์ ซึ่งให้ข้อมูลเกี่ยวกับบริบท(context)ของวิดเจ็ตใน  
  แอปพลิเคชั่น และต้องส่งคืนวิดเจ็ตที่เป็นตัวแทนของ UI

**หมายเหตุสำคัญ :**

**Stateless Widget จะไม่สามารถเปลี่ยนแปลงค่าภายในเองได้ เหมาะกับ UI ที่คงที่**

**Stateful Widget มีสถานะ (state) ที่เปลี่ยนได้ ใช้สำหรับ UI ที่โต้ตอบกับผู้ใช้**

**MaterialApp:**

return const MaterialApp(

  home: Scaffold(

    body: Center(

      child: Text('Hello World!'),

      ),

    ),

  ),

);

วิดเจ็ต MaterialApp เป็นวิดเจ็ตพื้นฐานสำหรับแอปพลิเคชั่น Flutter ทั้งหมด เป็นการตั้งค่าธีมเริ่มต้นของแอปพลิเคชั่น และกำหนดวิดเจ็ต home ซึ่งจะเป็นวิดเจ็ตหลักที่แสดงบนหน้าจอ

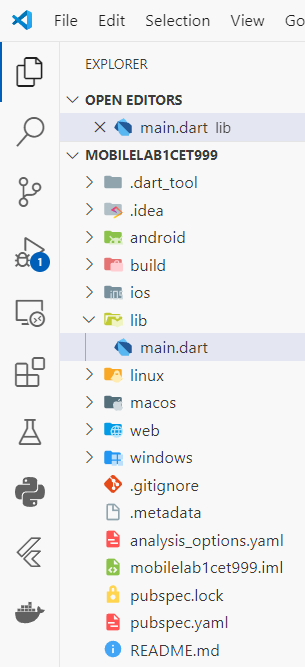
* home: Scaffold(): วิดเจ็ต Scaffold เป็นโครงร่างพื้นฐานสำหรับหน้าจอแอปพลิเคชั่น Flutter ทั่วไป โดยจะมีแถบแอปพลิเคชั่น (AppBar) แถบบาร์ด้านล่าง (BottomNavigationBar) และพื้นที่เนื้อหา (body) เป็นส่วนประกอบ
* body: Center(): วิดเจ็ต Center เป็นการวางวิดเจ็ตลูกไว้ตรงกลางของพื้นที่ว่างภายใน Scaffold
* child: Text('Hello World!'): สร้างวิดเจ็ต Text ภายใต้ วิดเจ็ต Center ที่อยู่ภายใต้ Scaffold อีกทีเพื่อแสดงข้อความ "Hello World!" บนหน้าจอ

**MaterialApp**

Scaffold

child

**โครงสร้างของโปรเจคและไฟล์สำคัญ**



ชื่อ Folder ของโปรเจค

lib จะเป็นที่เก็บไฟล์การทำงานของแอพพลิเคชั่นที่เราสร้าง เช่น ส่วนของหน้าจอแอพ, ข้อมูลในแอพ หน้าอื่นๆ ที่เป็นส่วนประกอบของแอพเรา โดยไฟล์จะต้องมีนามสกุล .dart เช่น main.dart ซึ่งเป็นไฟล์ที่ถูกประกาศให้เป็นจุดเริ่มต้นการทำงานของแอพ

pubspec.yaml ใช้ตั้งค่าต่างๆ และการเรียกใช้งานไลบราลี่ที่ใช้งานในแอพ

เป็น Folder ที่เตรียมไว้สำหรับแยกจัดเก็บโปรเจคในแต่ละ Platform

**Widget**

คือส่วนติดต่อผู้ใช้(UI) ที่ถูกแยกออกมาเป็นส่วนๆ และมีให้เรียกใช้งานอยู่แล้วใน Flutter ในการสร้างแอพจึงเหมือนการนำ Widget ที่เป็น UI ที่ต้องการมารวมกันเพื่อให้เกิดการทำงานตามที่ผู้พัฒนาต้องการ

**คุณสมบัติของ Widget:**

สร้าง: Widget แต่ละตัวมีหน้าที่สร้างส่วนต่างๆ ของหน้าจอ เช่น ปุ่ม ข้อความ รูปภาพ ไอคอน เมนู ฯลฯ

ปรับแต่ง: Widget สามารถปรับแต่งรูปลักษณ์และพฤติกรรมได้ตามต้องการ

จัดเรียง: Widget สามารถจัดเรียงซ้อนทับกันเพื่อสร้างโครงสร้าง UI ที่ซับซ้อน

โต้ตอบ: Widget บางตัวรองรับการโต้ตอบกับผู้ใช้ เช่น ปุ่ม ข้อความ ฯลฯ

\*\*\* Stateful/Stateless: Widget แบ่งเป็น 2 ประเภทหลักๆ คือ Stateful Widget เก็บสถานะข้อมูลไว้ภายในตัว และ Stateless Widget ไม่มีสถานะข้อมูล

**ประเภทของ Widget:**

พื้นฐาน: Widget พื้นฐาน เช่น Text, Image, Container, Row, Column ฯลฯ ใช้สร้างองค์ประกอบพื้นฐานของ UI

Material Design: Widget Material Design ช่วยให้สร้าง UI ที่สอดคล้องกับ Material Design

Cupertino: Widget Cupertino ช่วยให้สร้าง UI ที่สอดคล้องกับ Human Interface Guidelines ของ Apple

Custom: Widget Custom ช่วยให้สร้าง Widget เองตามต้องการ

**Widget Tree ใน Flutter**

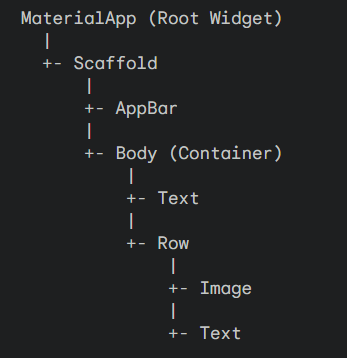
Widget Tree คือ โครงสร้างแบบลำดับชั้น (hierarchy) ที่ใช้ในการสร้างและจัดการ UI ของแอปพลิเคชัน Flutter ทุกๆ ส่วนของ UI ถูกสร้างขึ้นมาจาก Widget ซึ่ง Widget เหล่านี้จะถูกจัดเรียงและเชื่อมโยงกันเป็นต้นไม้ (Tree) โดยมี Widget ราก (Root Widget) อยู่ด้านบนสุด

**ความสำคัญของ Widget Tree**

1. **กำหนดโครงสร้างของ UI:** Widget Tree ช่วยกำหนดว่า Widget ตัวไหนอยู่บน Widget ตัวไหน และ Widget แต่ละตัวมีความสัมพันธ์กันอย่างไร
2. **แบ่งส่วนการทำงาน:** Widget Tree แบ่ง UI ออกเป็นส่วนๆ ทำให้การพัฒนาและบำรุงรักษาแอปพลิเคชันง่ายขึ้น
3. **เพิ่มประสิทธิภาพ:** Widget Tree ช่วยให้ Flutter ประเมินความเปลี่ยนแปลงของ UI และปรับปรุงเฉพาะส่วนที่จำเป็น

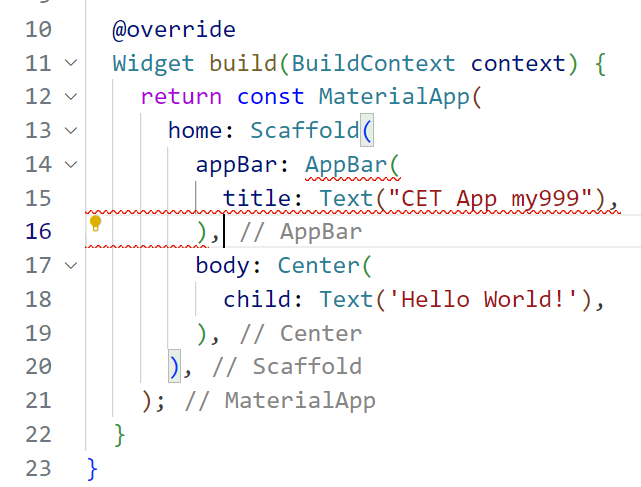
**องค์ประกอบของ Widget Tree**

1. **Widget:** เป็นบล็อคตัวต่อพื้นฐานที่ใช้สร้าง UI แต่ละ Widget มีหน้าที่และคุณสมบัติเฉพาะตัว
2. **Parent Widget:** Widget ที่อยู่ด้านบนในลำดับชั้น มีหน้าที่ในการจัดการและส่งต่อข้อมูลไปยัง Child Widget
3. **Child Widget:** Widget ที่อยู่ด้านล่างในลำดับชั้น ขึ้นอยู่กับ Parent Widget

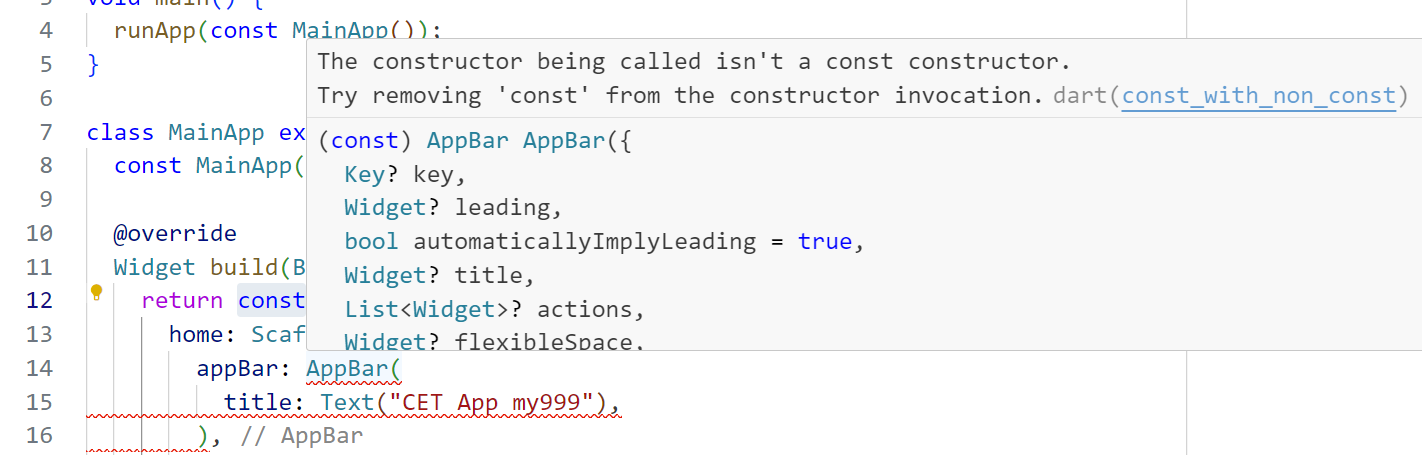


ตัวอย่างของ Widget Tree

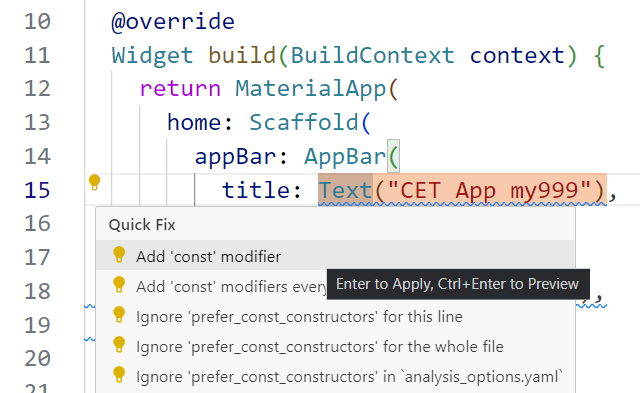
1. ให้เพิ่ม AppBar เข้าไปในแอพที่สร้างโดยจะทำการเพิ่มเข้าไปในส่วนของ home : scaffold ดังภาพ



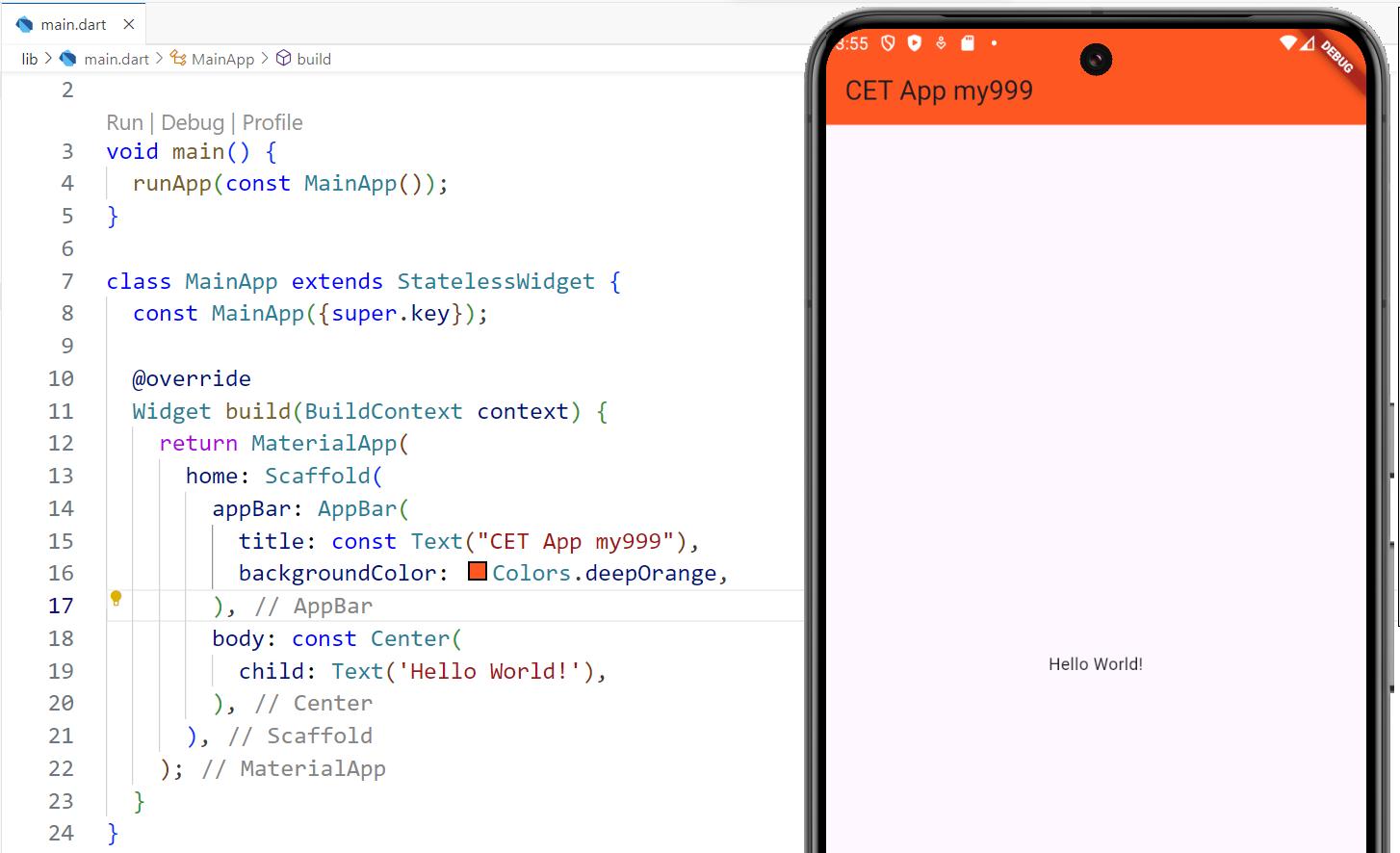
เส้นสีแดงที่แดงอยู่หมายถึงมีข้อผิดพลาดให้เอาเมาส์ไปชี้ที่ข้อความที่มีเส้นใต้สีแดงตรงไหนก็ได้



จะเห็นว่าการแจ้งเตือนให้ทำการ remove const ดังนั้นเราต้องเอา const ออกจากโค๊ดที่บางจุดโดยให้ลองไล่ขึ้นไปในทีนี้คือ บรรทัดที่ 12 จาก const MaterialApp ให้ลบ const ออกแล้วกดบันทึก(save) หรือ Ctrl+s สังเกตุผลที่ Device จะพบว่ามีการเพิ่ม AppBar และข้อความที่ด้านบนของแอพ แต่ในส่วนของโค๊ดโปรแกรมจะเห็นว่ายังมีเส้นสีฟ้าด้านใต้ให้ทดลองเอาเมาส์ไปชี้เพื่ออ่านคำแนะนำ จะพบว่าคำแนะนำให้เพิ่ม const เข้าไป ให้เราคลิ๊กที่บรรทัดที่มีเส้นสีฟ้าจะมีรูปหลอดไฟแสดงด้านหน้าให้คลิ๊กที่หลอดไฟและเลือก Add ‘const’ modifier ตามรูปจะเห็นว่ามีการเพิ่ม const หน้า Text Widget ให้และเส้นสีฟ้าก็จะหายไป จากนั้นทำกับบรรทัดที่เหลือ



เมื่อแก้ไขครบแล้วจะพบว่าโค๊ดไม่มีการแจ้งเตื่อนใดๆ แล้ว ต่อไปจะเป็นการเปลี่ยนสีให้ AppBar โดยเพิ่มคำสั่ง ไปที่ด้านล่างของ title ดังภาพ



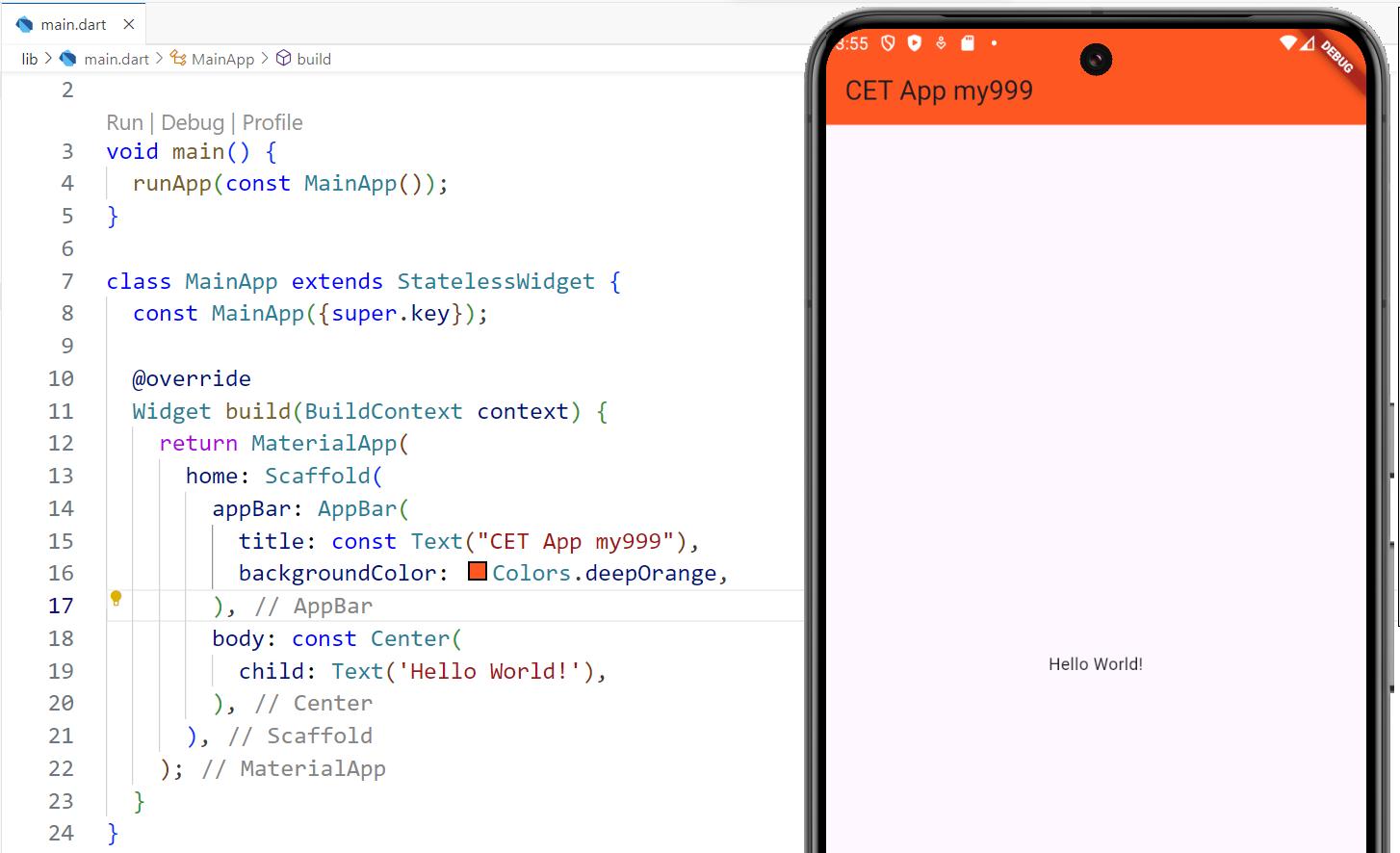
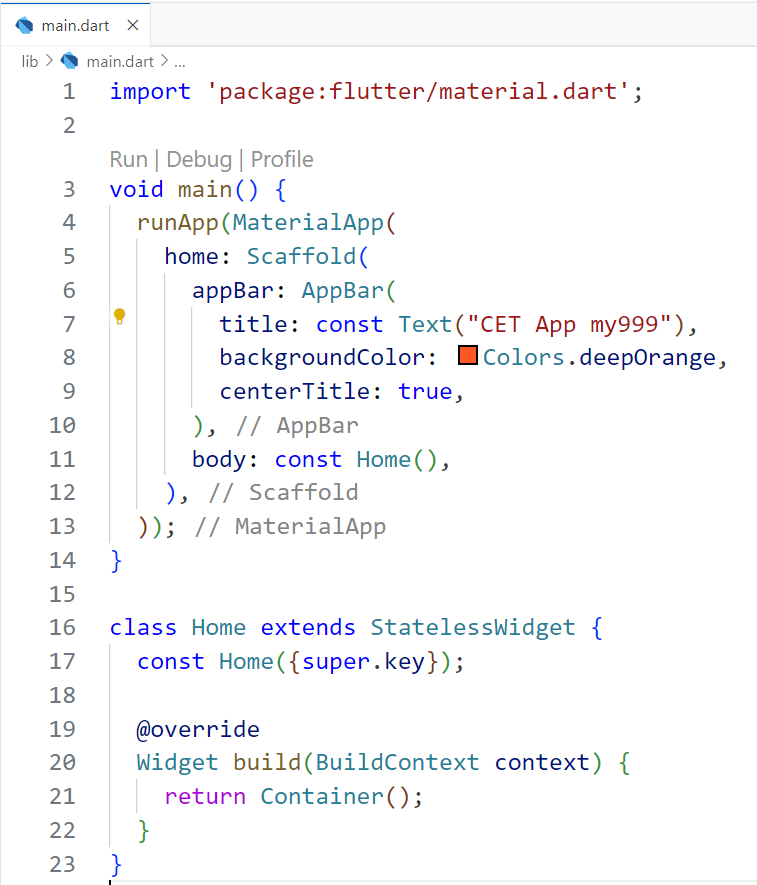
\*\*\*หากต้องการให้ข้อความบน AppBar อยู่ตรงกลางสามารถทำได้โดยเพิ่มคำสั่ง centerTitle: true, ใต้ backgroundColor

**Container Widget**

Container Widget เปรียบเสมือนกล่องที่ใช้บรรจุ Widget อื่นๆ บนหน้าจอ Container ช่วยให้เราสามารถกำหนดเค้าโครง รูปลักษณ์ และพฤติกรรมของ Widget ภายในกล่องได้ โดยมีคุณสมบัติดังนี้

* 1. กำหนดขนาด: สามารถกำหนดความกว้าง ความสูง และขนาดอื่นๆ ของ Container ได้
  2. กำหนด Padding: ช่องว่างระหว่างขอบ Container กับ Widget ภายใน
  3. กำหนด Margin: ช่องว่างระหว่าง Container กับ Widget อื่นๆ
  4. กำหนด Border: เส้นขอบของ Container
  5. กำหนดสีพื้นหลัง: สีพื้นหลังของ Container
  6. กำหนด Decoration: รูปแบบเพิ่มเติม เช่น เงา มุมโค้งมน ฯลฯ
  7. จัดเรียง Widget: รองรับการจัดเรียง Widget ภายใน Container

1. การใช้งาน Container เพื่อจัดการกับ Widget ต่างๆ ให้อยู่ในขอบเขตที่เราต้องการ โดยก่อนอื่นต้องปรับโค๊ดให้เป็นดังภาพใหม่



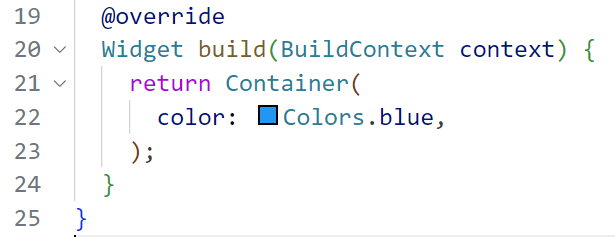
เดิม

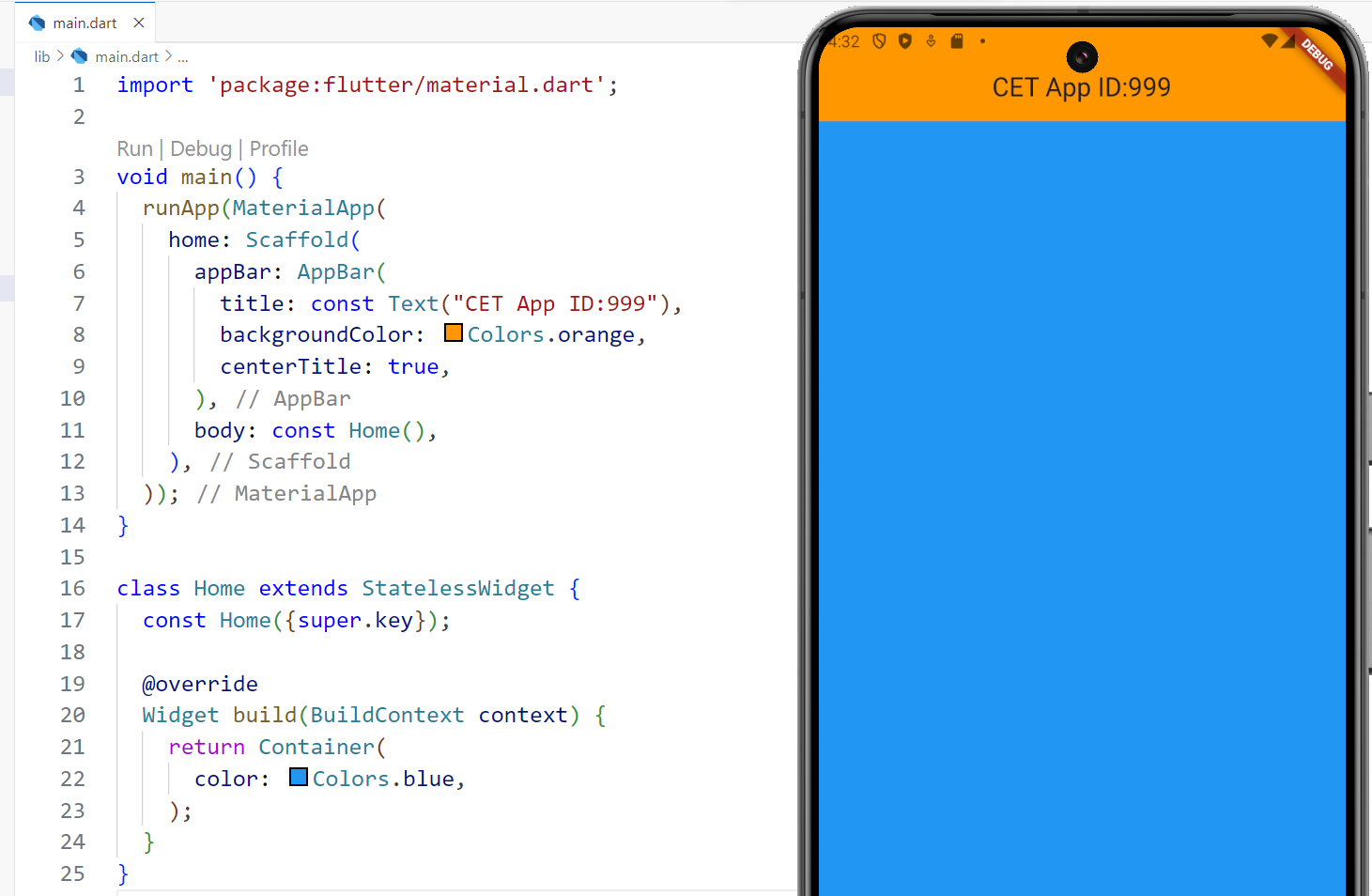
ใหม่

เมื่อปรับโค๊ดเสร็ตแล้วให้ทดลองดูผลที่ Device ตรงส่วนนี้จะพบปัญหาผลไม่อัพเดทนะครับ ต้องทำการกด Stop และ F5 ใหม่



จากนั้นจะทดสอบแสดงพื้นที่ทำงานของ Container โดยจะทดสอบใส่สีเข้าไปเพื่อให้แสดงผลออกมาโดยคลิ๊กไปที่ในวงเล็บของ Container กด Enter เพื่อขึ้นบรรทัดใหม่ และพิมพ์คำสั่งดังภาพ จากนั้นกดบันทึก





1. กำหนดการเว้นระยะห่างจากขอบให้กับ Container โดยใช้คำสั่ง margin: const EdgeInsets.symmetric(vertical: 3, horizontal: 3), โดย EdgeInsets สามารถเลือกลักษณะได้อีกหลายรูปแบบ

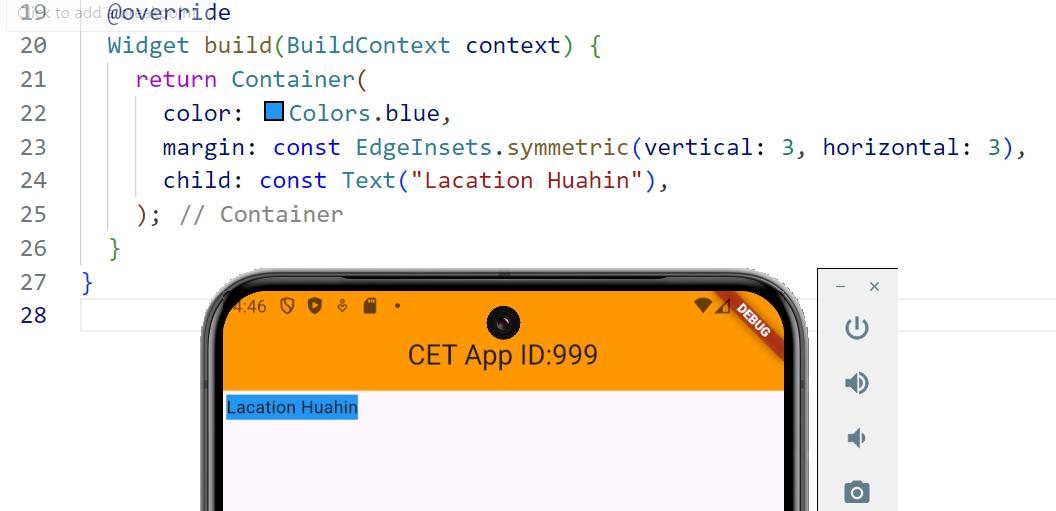
**คำถาม** ทดลองเลือกแต่ละแบบ กำหนดขนาด และบันทึกผลจากจอของ Device

..........................................................................................ภาพ.....................................................................................

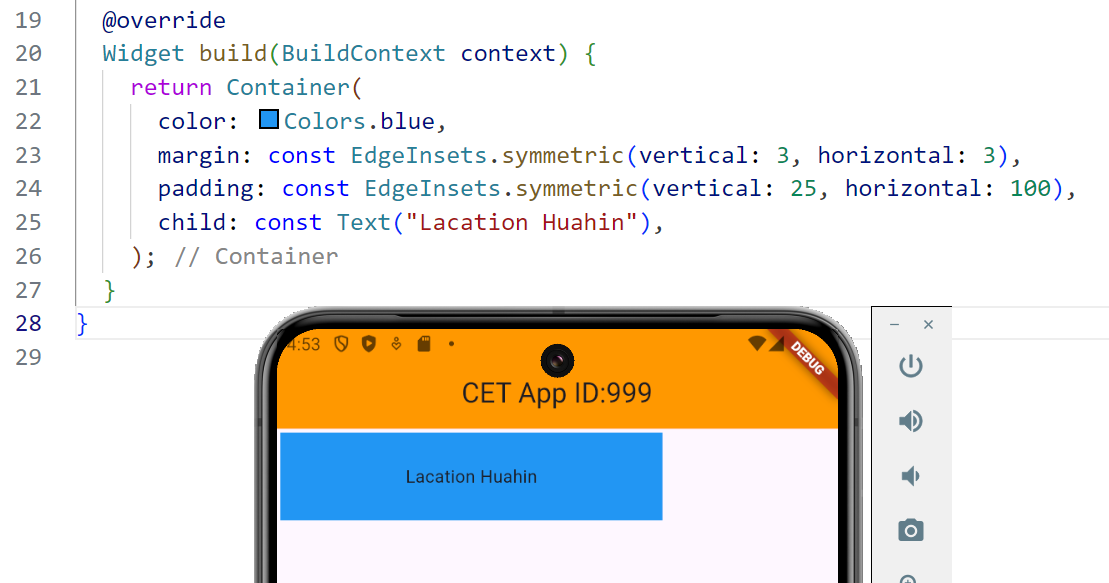
A screen shot of a cell phone

AI-generated content may be incorrect.

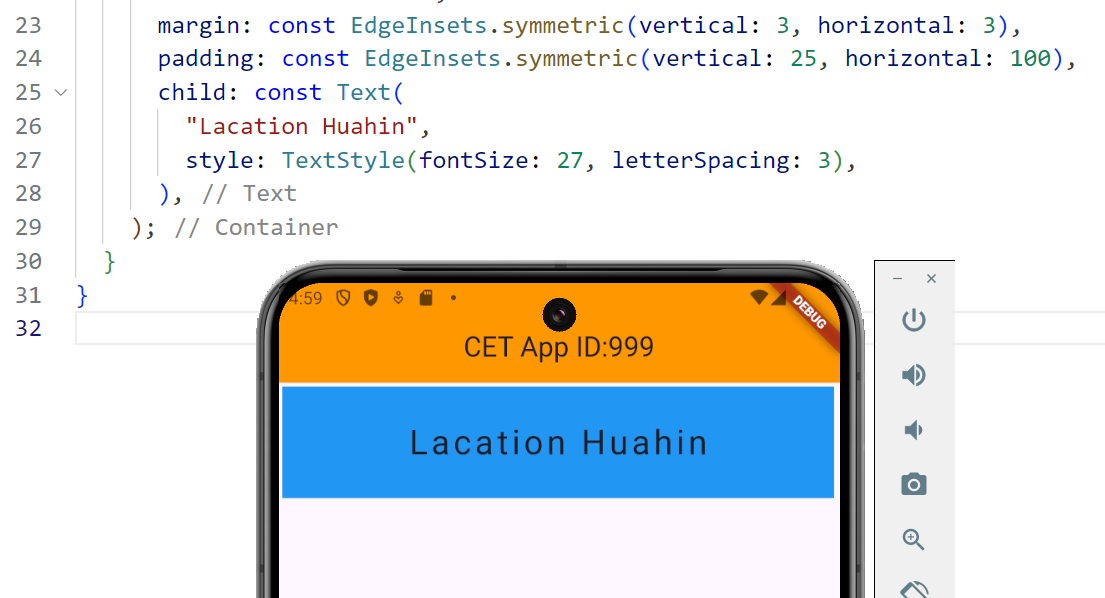
1. ในการจะเพิ่ม Widget อื่นๆ เข้าไปใน Container จะต้องใส่คำสั่ง child : ก่อนเสมอเช่นต้องการเพิ่ม Text widget เพื่อให้แสดงข้อความสามารถทำได้โดยใช้คำสั่งดังนี้ (ข้อความ Location Huahin)



จะพบว่าสีของ Container ที่เคยเต็มจอตอนนี้เหมือนเท่าขนาดของ Text Widget ซึ่งเราสามารถกำหนดให้ความห่างของ Container กับ Text Widget เพิ่มขึ้นได้โดยใช้ Padding โดยจะเพิ่มไปที่ใต้คำสั่ง Margin

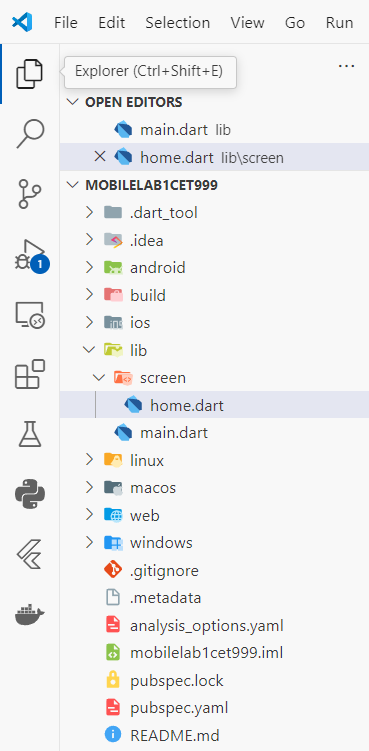


การปรับขนาดของข้อความสามารถทำได้โดยใช้ style เพื่อเรียกใช้ TextStyle() Widget โดยตัวอย่างต่อไปนี้จะเป็นการปรับขนาดของข้อความด้วยคำสั่ง fontSize และระยะห่างของตัวอักษรแต่ละตัวด้วยคำสั่ง letterspacing



1. การสร้าง class และการ import class เพื่อนำมาใช้งาน ในหัวข้อนี้จะทำการแยกโปรแกรมออกเป็น 2 ส่วนโดยจะแยกในส่วน

ของ Class Home ออกเป็นอีกไฟล์โดยจะเก็บที่ Folder ชื่อว่า screen และไฟล์ชื่อว่า home.dart



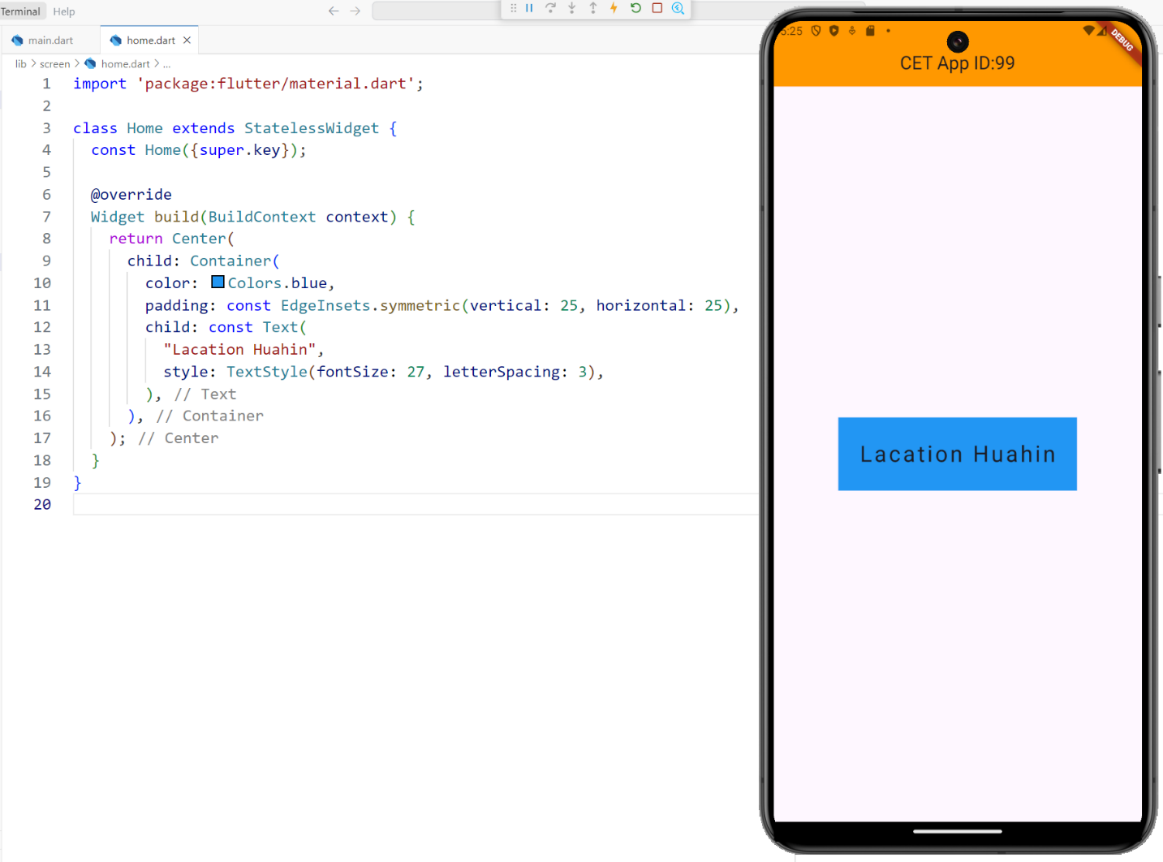
เมื่อสร้างไฟล์ home.dart แล้วให้ตัดโค๊ดส่วนของ class Home จาก main.dart มาไว้ที่ home.dart ซึ่งจะเกิด Error เนื่องจากหา material.dart ไม่เจอให้ทำการคลิ๊กที่ StatelessWidget เลือกที่หลอดไฟเลือก import 'package:flutter/material.dart';

จากนั้นที่ไฟล์ main.dart ก็จะพบว่ามี Error เช่นกันให้นักศึกษาคลิ๊กที่ Home() เลือกที่หลอดไฟและเลือก import 'screen/home.dart'; ซึ่งเป็น class ที่เราเพิ่งแยกออกไปเข้ามาทำงานร่วมด้วย

**การแยกส่วนของโค๊ดออกเป็น class ย่อยๆ จะช่วยให้การบำรุงรักษา ปรับแก้ไขโค๊ดของโปรเจคทำได้ง่ายขึ้นและเป็นสัดส่วนมากขึ้น**

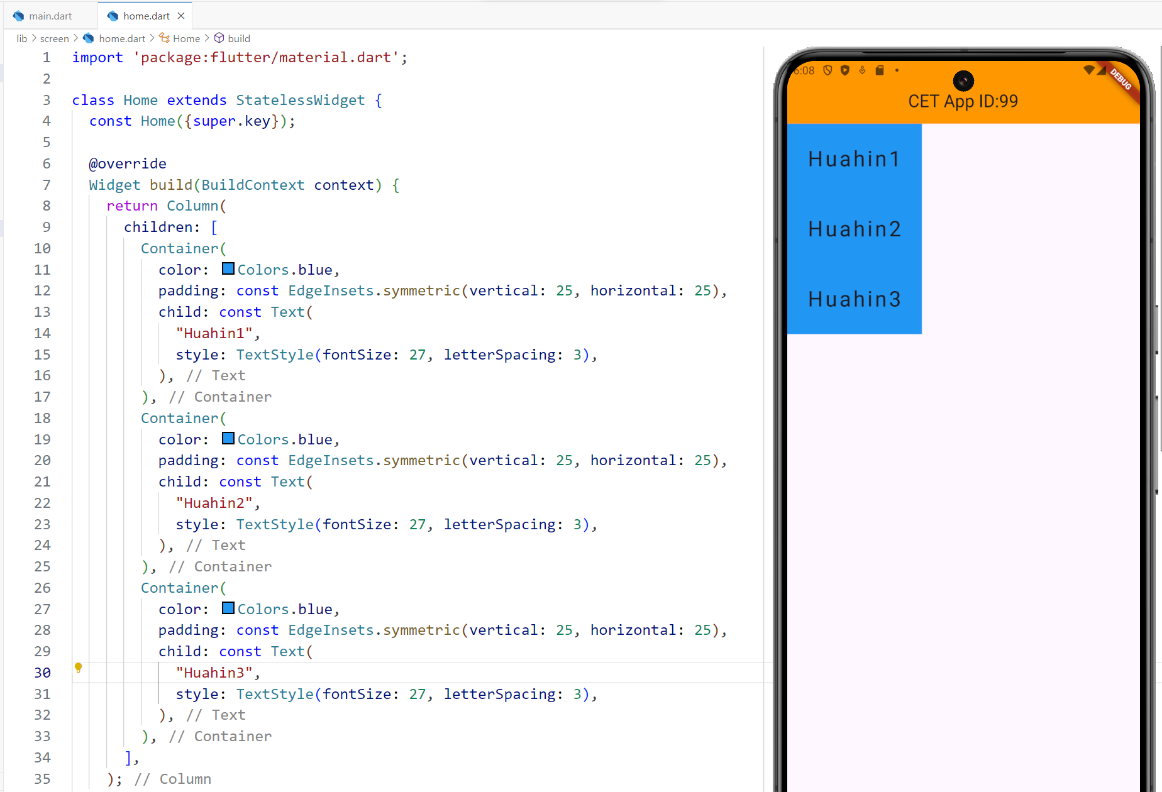
1. การจัดการตำแหน่งของ Widget จากตัวอย่างต่อไปนี้คือการย้ายตำแหน่ง Container Widget ให้อยู่กึ่งกลางจอ โดยมีการปรับ

โค๊ดโดยลบคำสั่ง margin ออกและปรับระยะห่างจากขอบให้เท่ากันทั้งสองด้าน สามารถทำได้โดยคลิ๊กไปที่ Container แล้วคลิ๊กที่หลอไฟ เลือกที่ More Action เลือก Wrap with Center



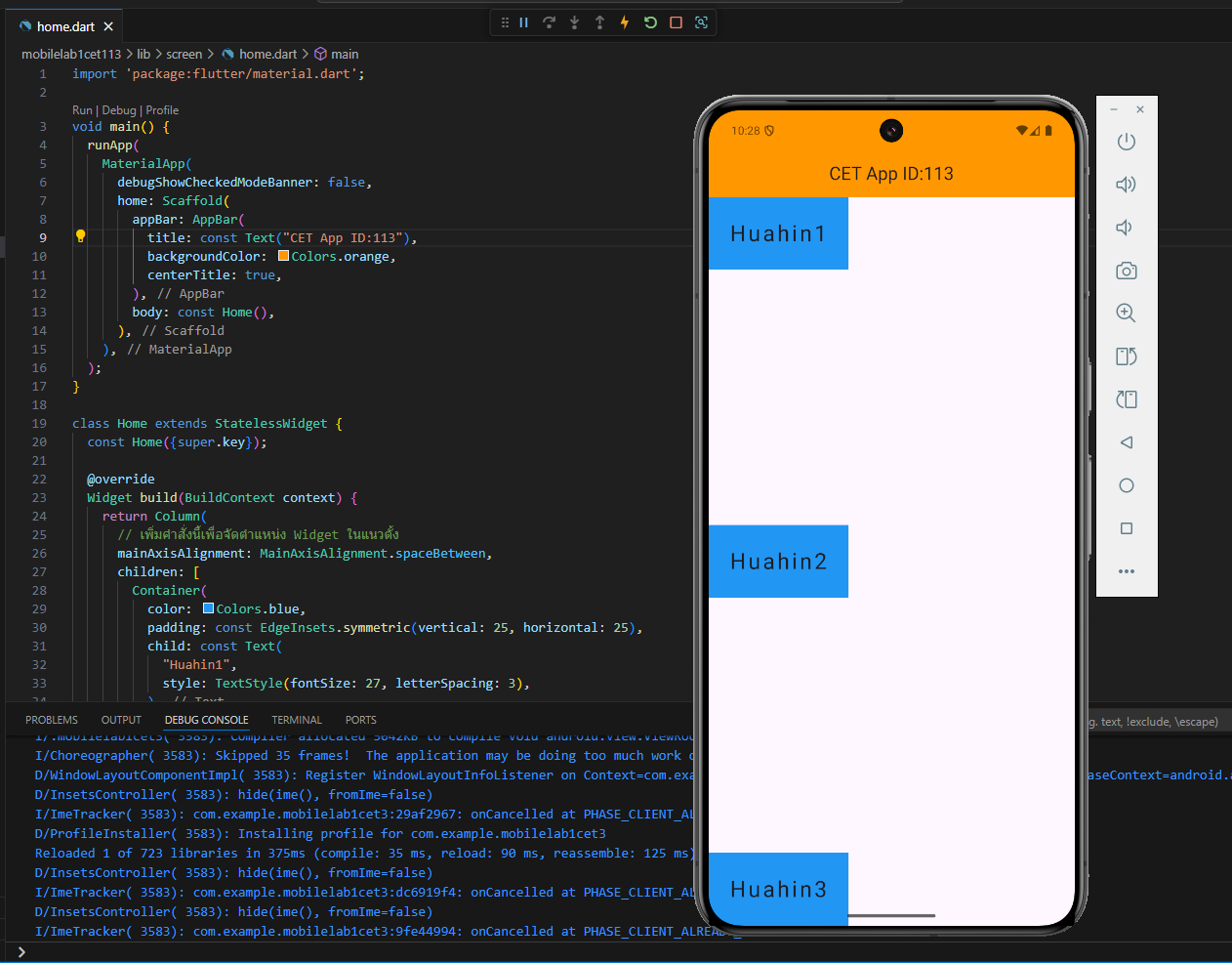
1. การจัดตำแหน่ง Container ด้วย Column Widget สามารถทำได้โดยตัดโค๊ดส่วนของ Container หรือคัดลอกไปไว้ที่ Notepad

แล้วลบโค๊ดใน home.dart ออก จากนั้นลบ child: ออก และเปลี่ยน Center เป็น Column จากนั้นสร้าง children: widget และนำโค๊ดชุดที่ตัดไป มาวางใน []

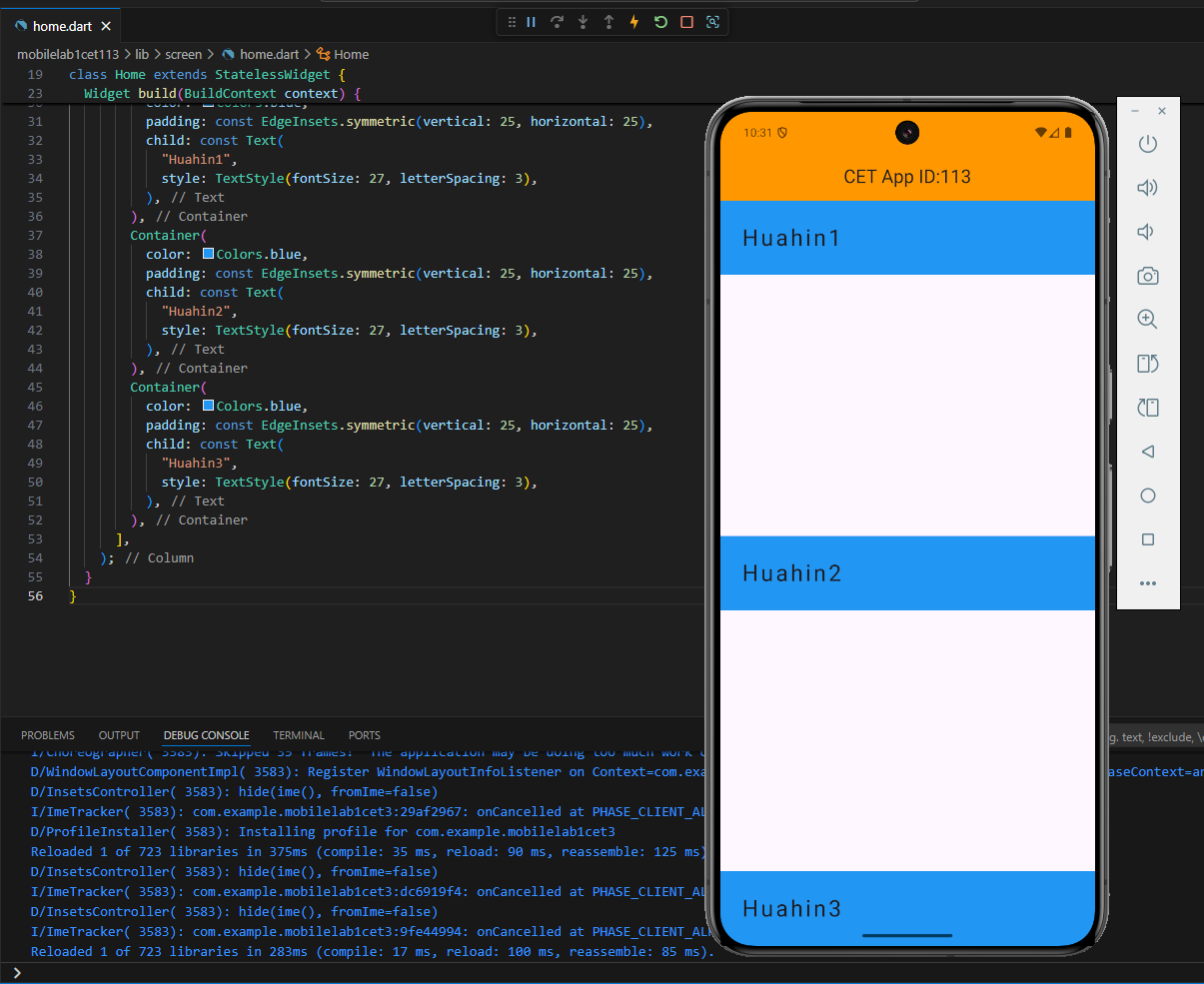
จากนั้นคัดลอกส่วนของ Container เพิ่มให้ครบ 3 Container ซึ่งการใช้ Column ร่วมกับ children จะทำให้สามารถใส่ Widget มากกว่า 1 เข้าไปได้

**คำถาม** ลองเพิ่มคำสั่ง mainAxisAlignment: MainAxisAlignment.spaceBetween, ใต้คำสั่ง return Column บันทึกผลการรัน

....................................................................................................................................................................................................



**คำถาม** ลองเพิ่มคำสั่ง crossAxisAlignment: CrossAxisAlignment.stretch,ใต้คำสั่ง mainAxisAlignment บันทึกผลการรัน

.................................................................................................................................................................................................. 

1. การจัดตำแหน่ง Container ด้วย Row Widget สามารถเปลี่ยนโค๊ดจาก Column เป็น Row เริ่มต้นให้ทำการลบ

mainAxisAlignment และ crossAxisAlignment เดิมออกก่อน และเปลี่ยนขนาด fontSize = 15 ให้ครบทั้ง 3 Container

**คำถาม** .............................................................บันทึกรูปผลการรัน.............................................................................................

**คำถาม** เมื่อเพิ่ม mainAxisAlignment = spaceEvenly

.............................................................บันทึกรูปผลการรัน.............................................................................................

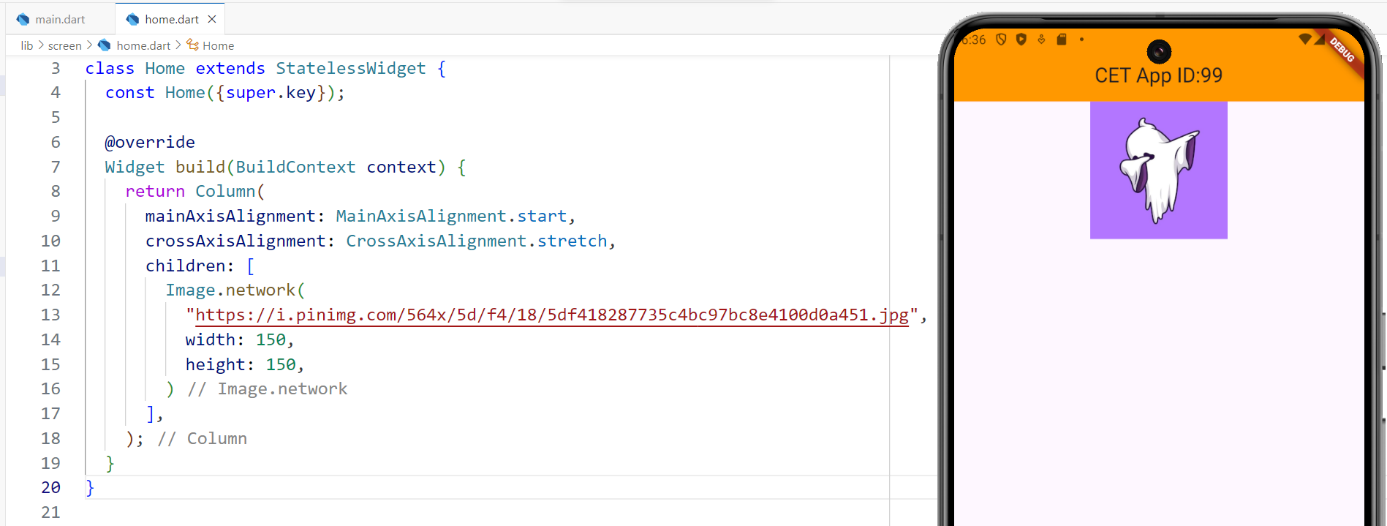
**คำถาม** เมื่อเพิ่ม crossAxisAlignment = stretch

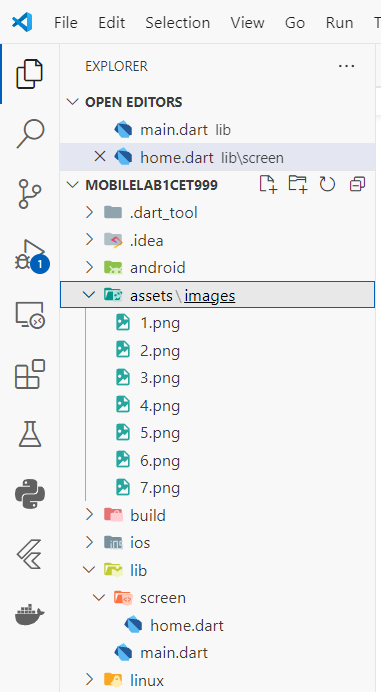
.............................................................บันทึกรูปผลการรัน.............................................................................................

1. การแสดงรูปภาพในแอพพลิเคชั่นด้วย Image Widget ซึ่งจะมีวิธีการอ้างถึงรูปภาพ 2 แบบคือ External(ภาพจากแหล่งอื่นเช่น

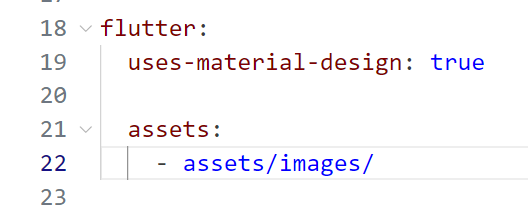
internet) โดยใช้คำสั่ง image.network() และ internal(ภาพที่ใส่ไว้ในโปรเจค) image.asset() (นักศึกษาต้องโหลดภาพจาก class room มาก่อน)

ในหัวข้อนี้เริ่มต้นให้เราลบทุก Widget ใน Row ออกให้หมดและเปลี่ยนจาก Widget Row เป็น Column จากนั้นเพิ่ม children: widget แบบแรกนี้จะใช้การดึงภาพแบบ External จาก <https://i.pinimg.com/564x/5d/f4/18/5df418287735c4bc97bc8e4100d0a451.jpg> จากนั้นกำหนดขนาดและจัดตำแหน่งภาพดังโค๊ดต่อไปนี้

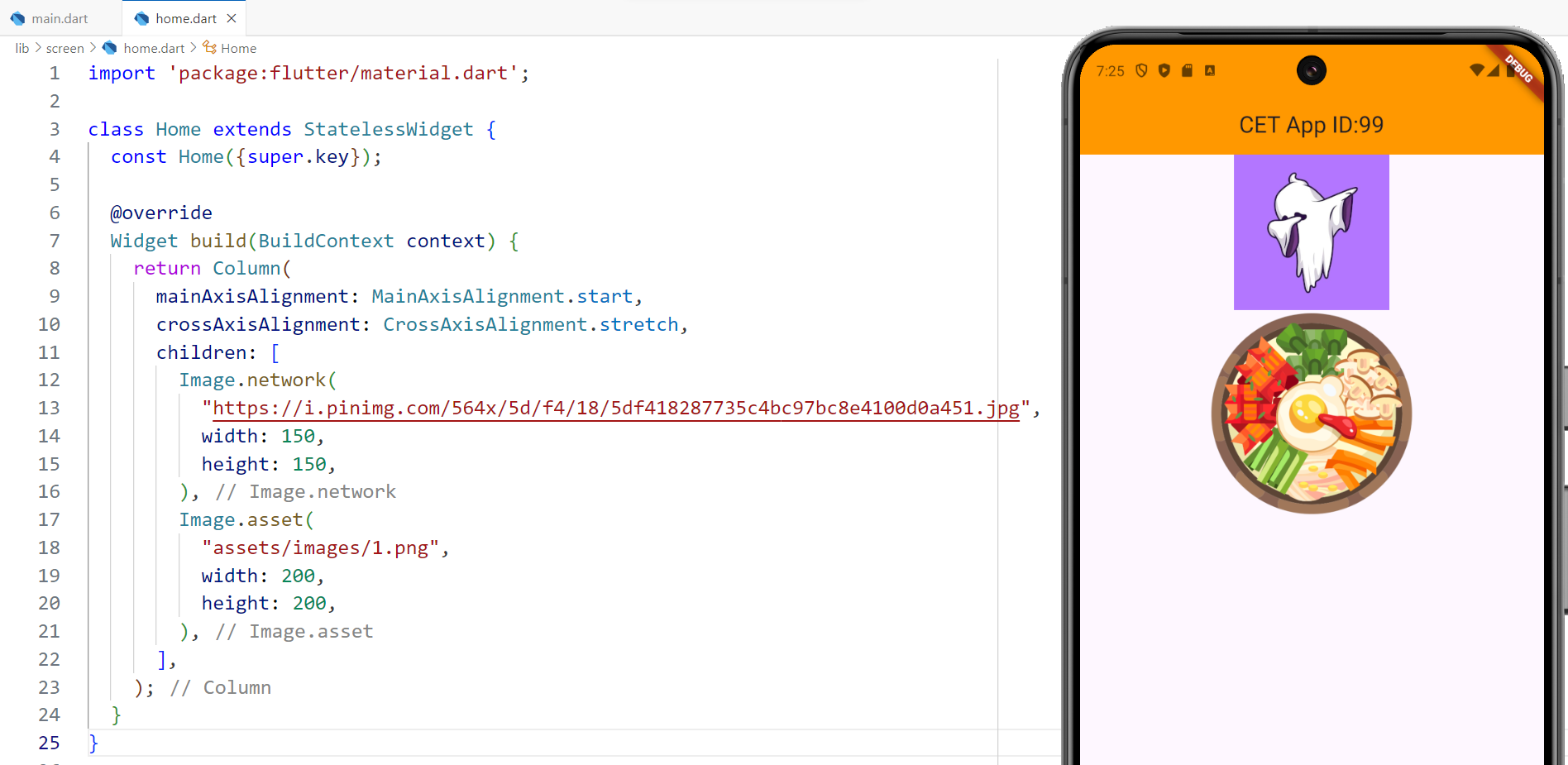


 โหลดภาพจาก Classroom และทำการแตกไฟล์ให้เรียบร้อยจากนั้นไปที่โปรเจคของเราสร้าง Folder assets เพื่อไว้เก็บภาพของเราที่โหลดมา รูปทั้งหมดจะอยู่ในใน Folder images ให้ลาก Folder images ไปใส่ใน assets

จากนั้นเราต้องระบุค่าเพื่อแจ้งให้โปรเจคทราบว่าภาพเราเก็บที่ไหนโดยเปิดไฟล์ pubspec.yaml จากนั้นพิมพ์คำสั่งต่อท้ายบรรทัดสุดท้าย



ขั้นต่อมาลองเรียกใช้งานภาพโดยเพิ่มคำสั่ง



อย่าลืมใส่คอมม่า

สามารถใช้ const SizedBox() เพื่อกำหนดความห่างของแต่ละภาพได้โดยนำไปใส่คั่นคำสั่งดึงภาพ

**คำถาม** ให้นักศึกษาลบภาพที่มาจาก Network ออกแล้วเพิ่มภาพให้ครบ 3 ภาพกำหนดขนาดทั้ง 3 ภาพให้เท่ากัน กำหนดระยะห่างแต่ละภาพเท่ากับ 15

.............................................................บันทึกรูปผลการรัน.............................................................................................

**คำถาม** ให้นักศึกษาเพิ่มข้อความใต้ภาพเป็นชื่อของอาหาร โดยจัดให้อยู่ตรงกลาง  **และจัดระยะห่างขอบบน และระหว่างภาพให้เหมาะสมสวยงาม**

.............................................................บันทึกรูปผลการรัน.............................................................................................

\*\*\*นักศึกษาต้องอัพโหลดโปรเจคทั้งหมดเข้า Git และบันทึก Git URL : ……………………………………มาในนี้เพื่อให้อาจารย์ตรวจงานได้ \*\*\*

**ตัวอย่าง**

