Nama : Mochamad Gilang Fadil Hakim

NIM : H1D022082

SHIFT : B

1.**Import Library**:

```
import 'dart:async';

import 'dart:math';
```

- **dart:async**: Digunakan untuk mendukung fitur asynchronous seperti Future, async, dan await.
- **dart:math**: Untuk fungsi matematika, dalam hal ini sin dari Dart.

2**. Kelas LimitCalculator**:

```
class LimitCalculator {
  double Function(double) function;
  LimitCalculator(this.function);
```

- **double Function(double) function**: Tipe data untuk menyimpan fungsi matematika yang akan dihitung limitnya.
- **Constructor LimitCalculator**: Menginisialisasi objek dengan fungsi matematika yang akan digunakan

3**. Fungsi calculateLimit**:

```
double calculateLimit(double c, double epsilon) {
  try {
    if (epsilon <= 0) {
      throw Exception("Epsilon harus lebih besar dari 0.");
    }
    double f1 = function(c - epsilon);
    double f2 = function(c + epsilon);
    return (f1 + f2) / 2;
  } catch (e) {
    print("Terjadi kesalahan: ${e.toString()}");
    return double.nan;
  }
}
```

- **Parameter c dan epsilon**: c adalah titik limit yang ingin dihitung, dan epsilon adalah perbedaan kecil untuk mendekati limit dari kiri dan kanan.
- **Validasi Input**: Memeriksa apakah epsilon lebih besar dari 0. Jika tidak, maka akan memunculkan pengecualian (Exception).
- **Perhitungan Limit**: Menggunakan pendekatan numerik dengan f(c - epsilon) dan f(c + epsilon) untuk menghitung nilai limit mendekati titik c.
- **Penanganan Eksepsi**: Jika terjadi kesalahan selama perhitungan (seperti pembagian oleh nol), eksepsi ditangani, dan fungsi mengembalikan nilai NaN (Not a Number).

4**. Fungsi Asynchronous calculateLimitAsync**:

```
Future<double> calculateLimitAsync(double c, double epsilon) async {
  return await Future.delayed(Duration(seconds: 2), () {
    return calculateLimit(c, epsilon);
  });
}
```

- **Future<double>**: Mengembalikan Future yang akan menyimpan hasil perhitungan limit setelah proses asynchronous selesai.
- **Future.delayed**: Mensimulasikan penundaan selama 2 detik untuk menunjukkan bagaimana perhitungan limit dapat berjalan secara asynchronous
- **await dan async**: Digunakan untuk menjalankan perhitungan secara asynchronous dan menunggu hasil.

5. **Fungsi main (Program Utama)**

```
void main() async {
  double function(double x) {
    if (x == 0) {
      return 1.0;
    }
    return sin(x) / x;
  }
```

- **function**: Mendefinisikan fungsi f(x)=sin⁡(x)xf(x) = \frac{\sin(x)}{x}f(x)=xsin(x), yang memiliki limit spesial saat x→0x \to 0x→0, yaitu 1. Ini ditangani dengan penanganan khusus saat x == 0 untuk menghindari pembagian dengan nol.

```
  var calculator = LimitCalculator(function);
  double c = 0;
  double epsilon = 0.0001;
```

- **LimitCalculator**: Membuat objek calculator dari kelas LimitCalculator yang akan digunakan untuk menghitung limit.

- **c dan epsilon**: Menetapkan titik limit c = 0 dan nilai kecil epsilon = 0.0001 untuk mendekati nilai limit.

```
print("Menghitung limit secara asynchronous...");
double result = await calculator.calculateLimitAsync(c, epsilon);
print("Hasil limit: $result");
```

- **Asynchronous Call**: Memanggil fungsi calculateLimitAsync secara asynchronous dan menunggu hasilnya dengan await.

```
if (result.isFinite) {
  print("Perhitungan limit berhasil!");
} else {
  print("Perhitungan limit gagal.");
}
}
```

Struktur Kontrol: Mengecek apakah hasil perhitungan adalah nilai yang terdefinisi dengan baik (isFinite). Jika ya, perhitungan berhasil, jika tidak, perhitungan dianggap gagal.