

LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK



JUDUL :
FUNDAMENTAL DART

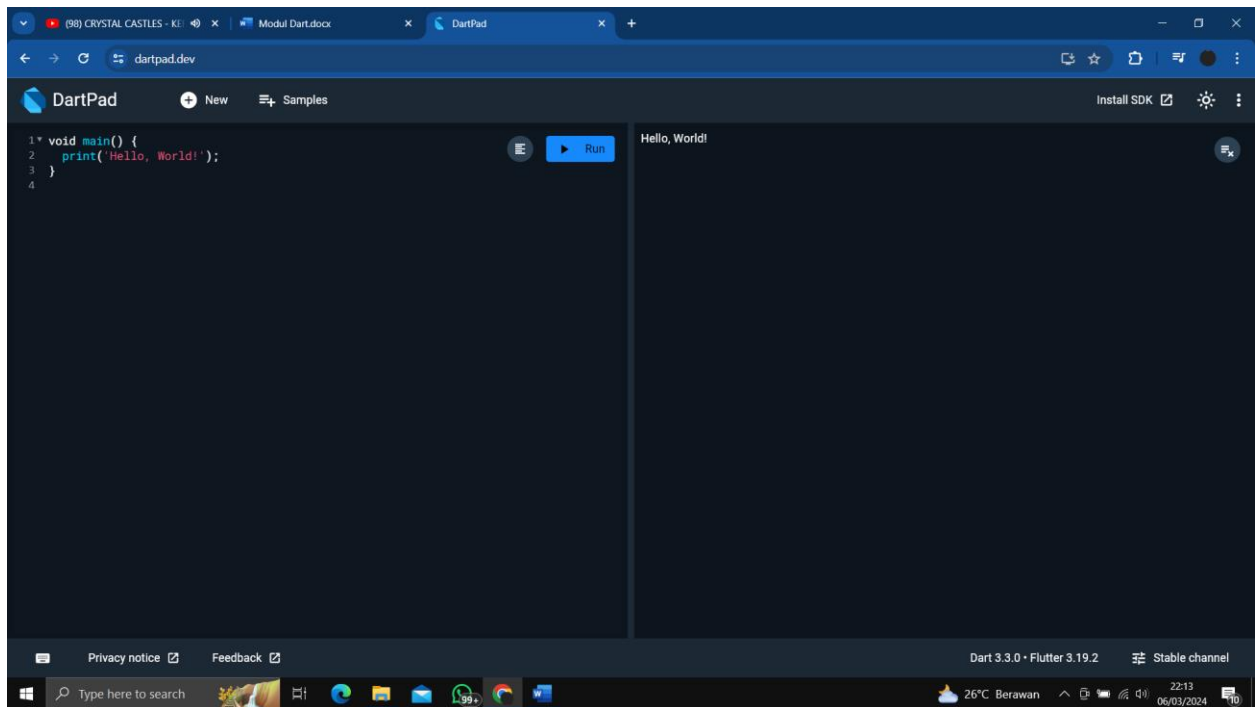
Disusun oleh :
Fajar Setiawan (21102183)

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
BANYUMAS, JAWA TENGAH
2024

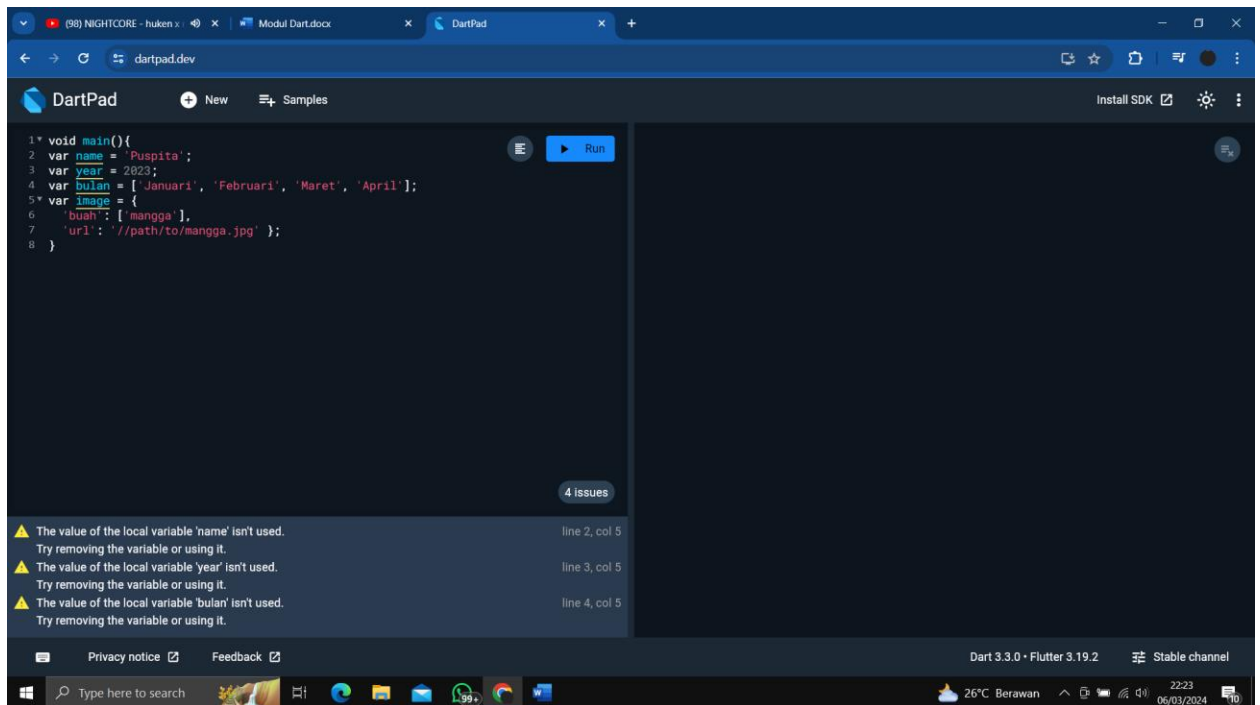
Pembahasan

Langkah – Langkah praktikum

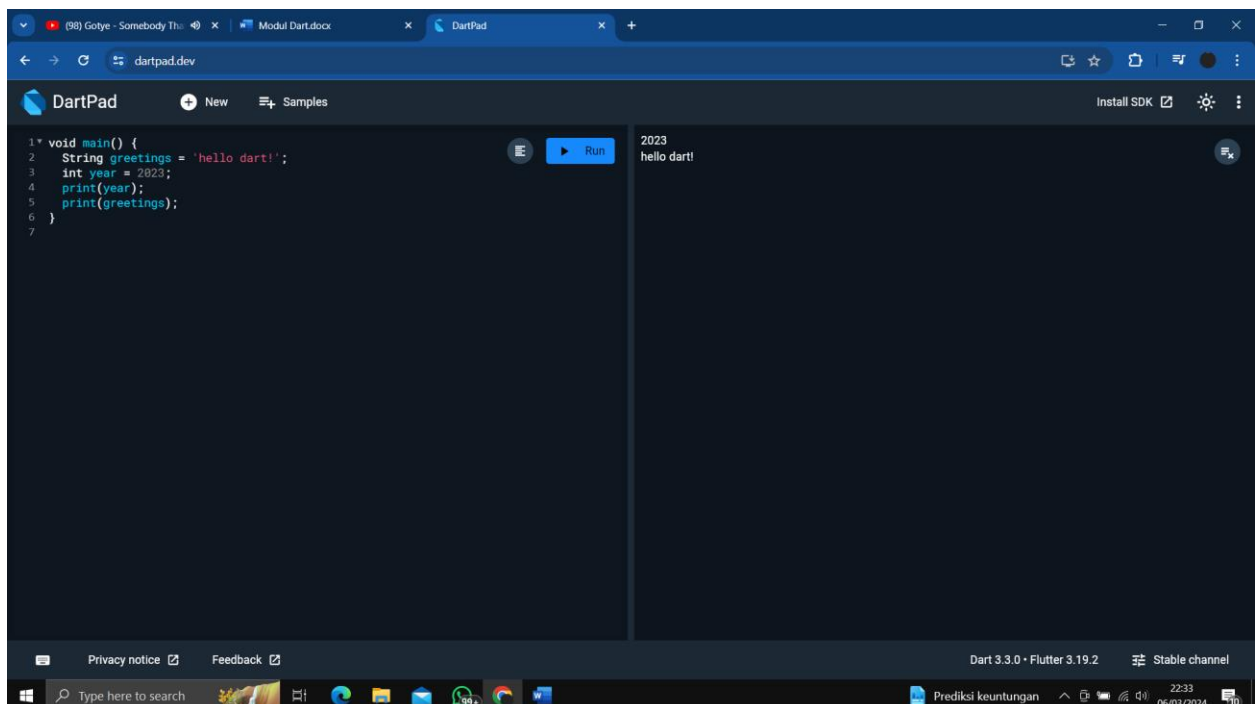
1. Fundamental Dart



Membuat program Hello world, cukup menggunakan perintah `print ()` yang mana kalimat hello world tersebut harus menggunakan tanda petik " karena secara default adalah string.

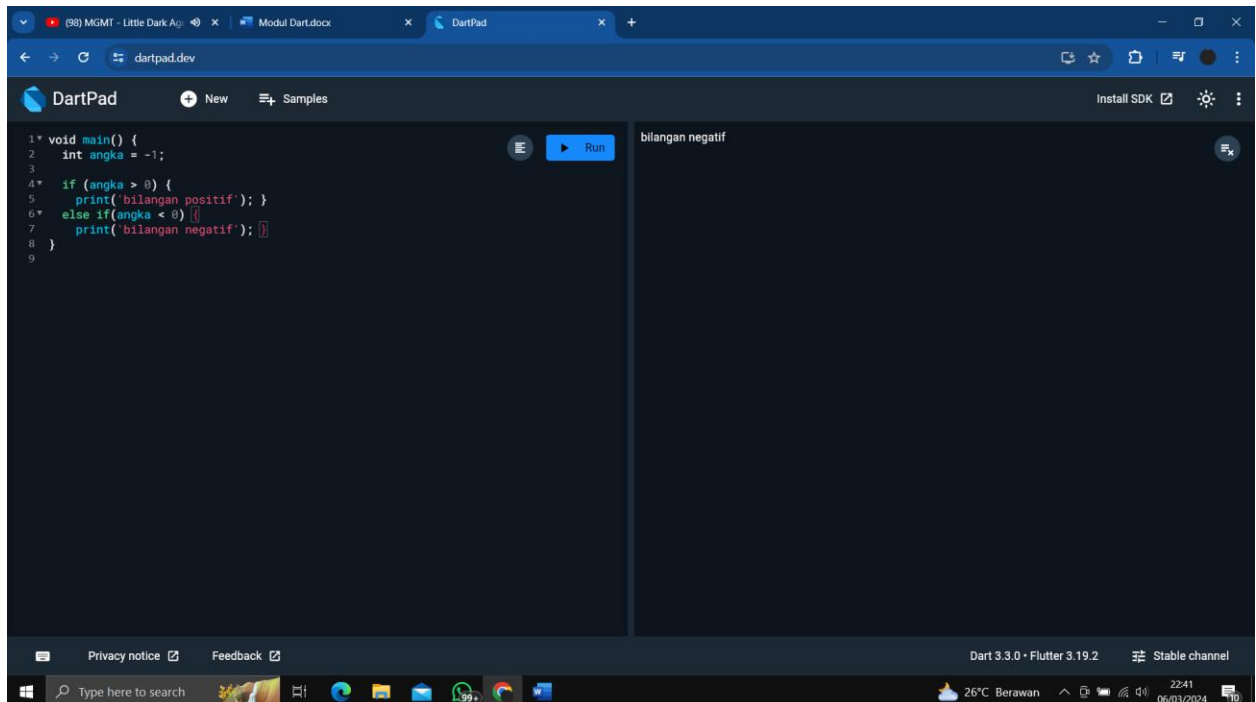


Langkah diatas merupakan cara untuk mendeklarasikan variable, namun variable diatas belum memiliki tipe data yang jelas. Perlu diketahui bahwa setiap program yang pertama kali dieksekusi adalah program yang terdapat didalam function main.



Langkah diatas merupakan deklarasi variable dengan tipe data String dan int. String sendiri merupakan tipe data untuk kalimat, kata bahkan karakter. Sedangkan int merupakan tipe data untuk bilangan bulat. Perintah print tersebut untuk membuktikan bahwa variable tidak eror.

2. Control Flow

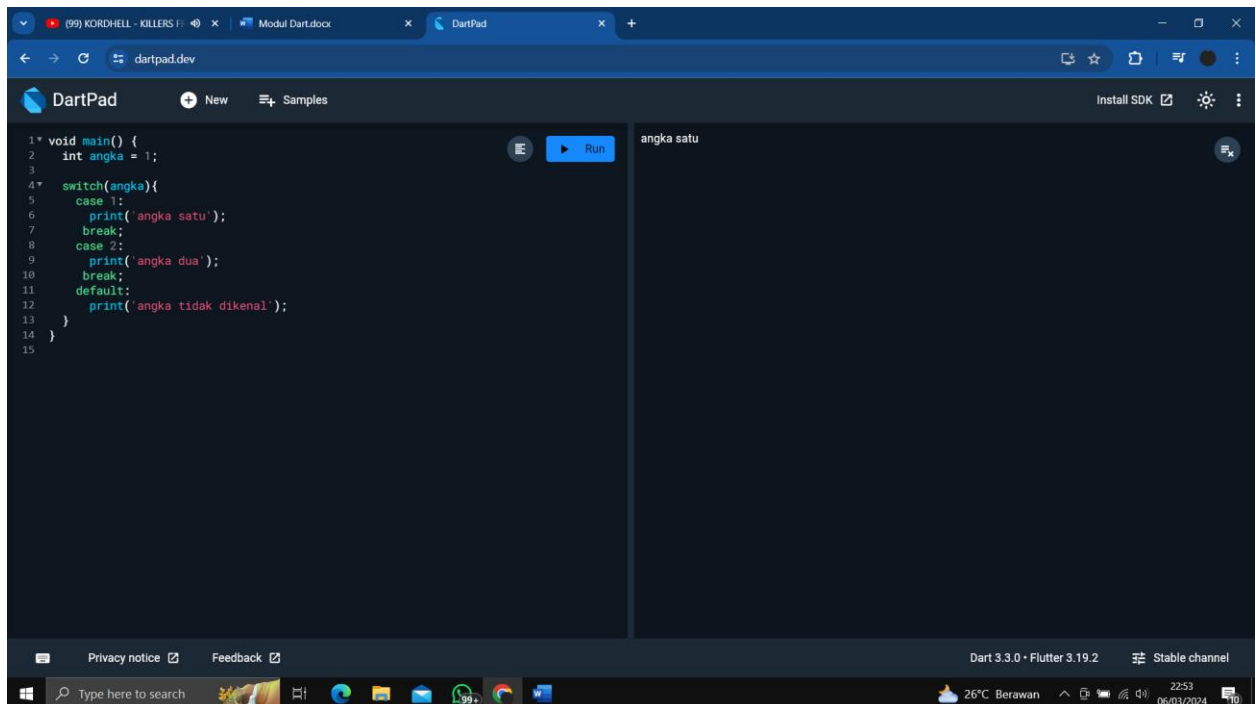


The screenshot shows the DartPad web interface. The code editor on the left contains the following Dart code:

```
1* void main() {  
2   int angka = -1;  
3  
4*   if (angka > 0) {  
5     print('bilangan positif');  
6*   } else if (angka < 0) {  
7     print('bilangan negatif');  
8   }  
9 }
```

The output console on the right displays the result: "bilangan negatif". The status bar at the bottom indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

if & else merupakan percabangan yang mana jika suatu kondisi memenuhi maka kondisi itulah yang akan dieksekusi. Pada program diatas membuat variable bertipe int dengan nilai -1, kemudian terdapat percabangan jika angka lebih besar dari 0 maka program akan mengeksekusi print bilangan bulat. Namun pada program diatas, variable nilai diberi nilai -1, sehingga hasil yang didapat adalah bilangan negative. Percabangan diatas merupakan percabangan 2 kondisi.

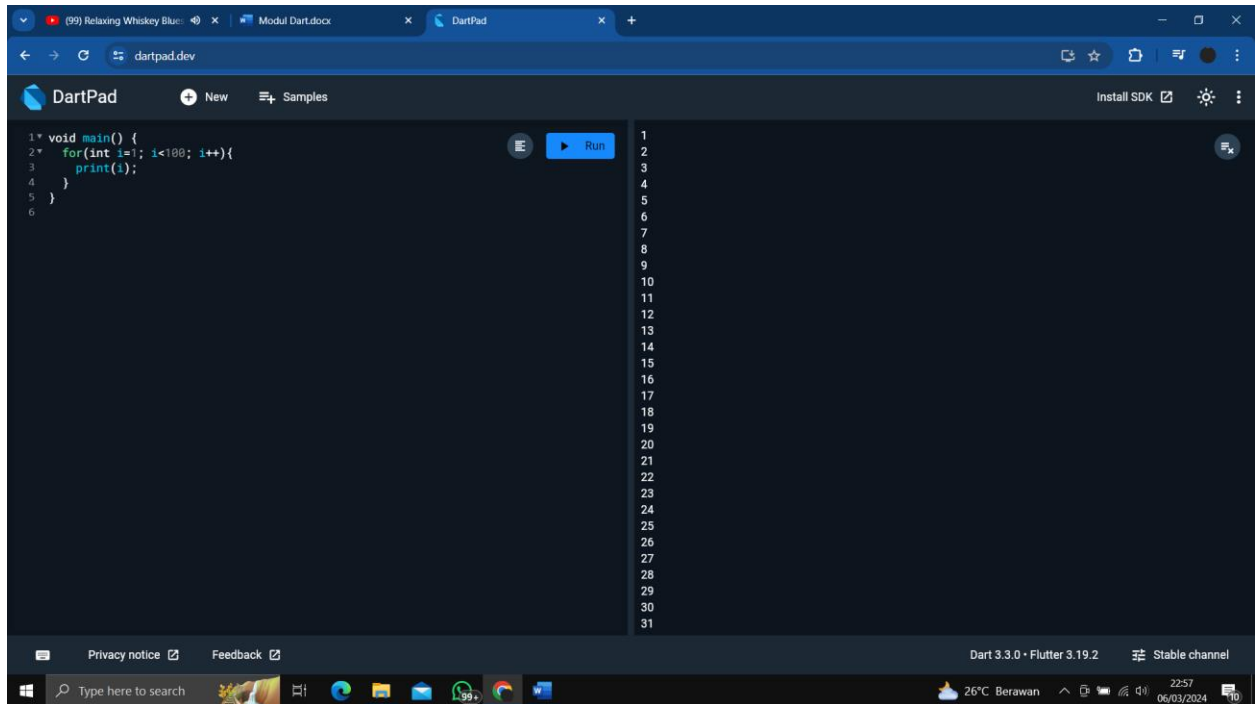


The screenshot shows the DartPad web interface. The code editor on the left contains the following Dart code:

```
1* void main() {  
2   int angka = 1;  
3  
4*   switch(angka){  
5     case 1:  
6       print('angka satu');  
7       break;  
8     case 2:  
9       print('angka dua');  
10      break;  
11     default:  
12       print('angka tidak dikenal');  
13   }  
14 }  
15 }
```

The output console on the right displays the result: "angka satu". The status bar at the bottom indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

Switch case merupakan percabangan. Sama halnya dengan if else namun switch case digunakan untuk percabangan yang memiliki lebih dari 2 kondisi. Pada tiap masing masing kondisi pada switch case selalu diakhiri dengan break & default seagai penanda bahwa akhirnya sebuah percabangan.

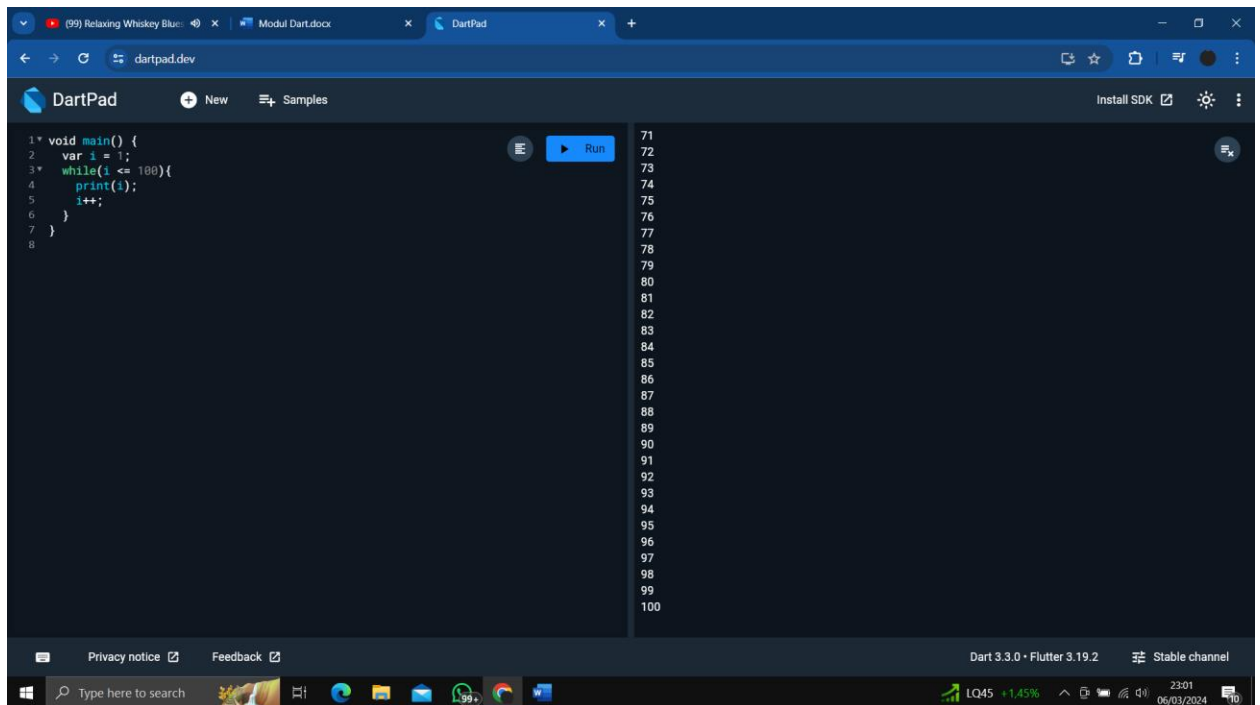


The screenshot shows the DartPad web interface in a browser. The address bar shows 'dartpad.dev'. The interface has a dark theme. On the left, there is a code editor with the following Dart code:

```
1* void main() {  
2*   for(int i=1; i<100; i++){  
3*     print(i);  
4*   }  
5* }  
6
```

On the right, there is a console output area showing a list of numbers from 1 to 31. The status bar at the bottom indicates 'Dart 3.3.0 • Flutter 3.19.2' and 'Stable channel'. The Windows taskbar is visible at the very bottom.

Pada Langkah ini merupakan perulangan menggunakan for. Program diatas mencetak angka dari 1 hingga 99. Perulangan akan diawali dari angka 1, kemudian angka 1 akan discounter 1. Hal ini akan terus berulang hingga i berjumlah kurang dari 100 yaitu 99.

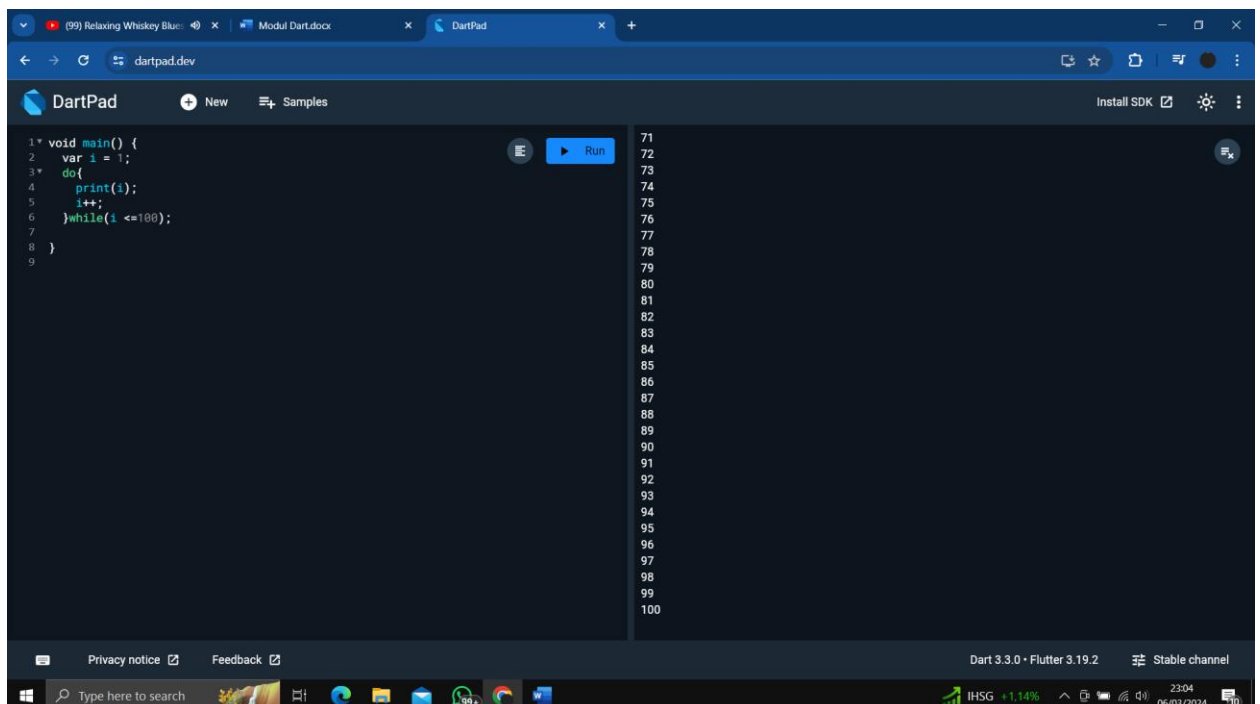


The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1* void main() {  
2  var i = 1;  
3* while(i <= 100){  
4    print(i);  
5    i++;  
6  }  
7 }  
8
```

The code is executed, and the output on the right shows a vertical list of numbers from 71 to 100, indicating the loop has completed its execution.

Perulangan menggunakan do sebenarnya hampir sama dengan for, hanya saja perulangan ini akan mengambil logika terlebih dahulu kemudian I nya brau dicounter.



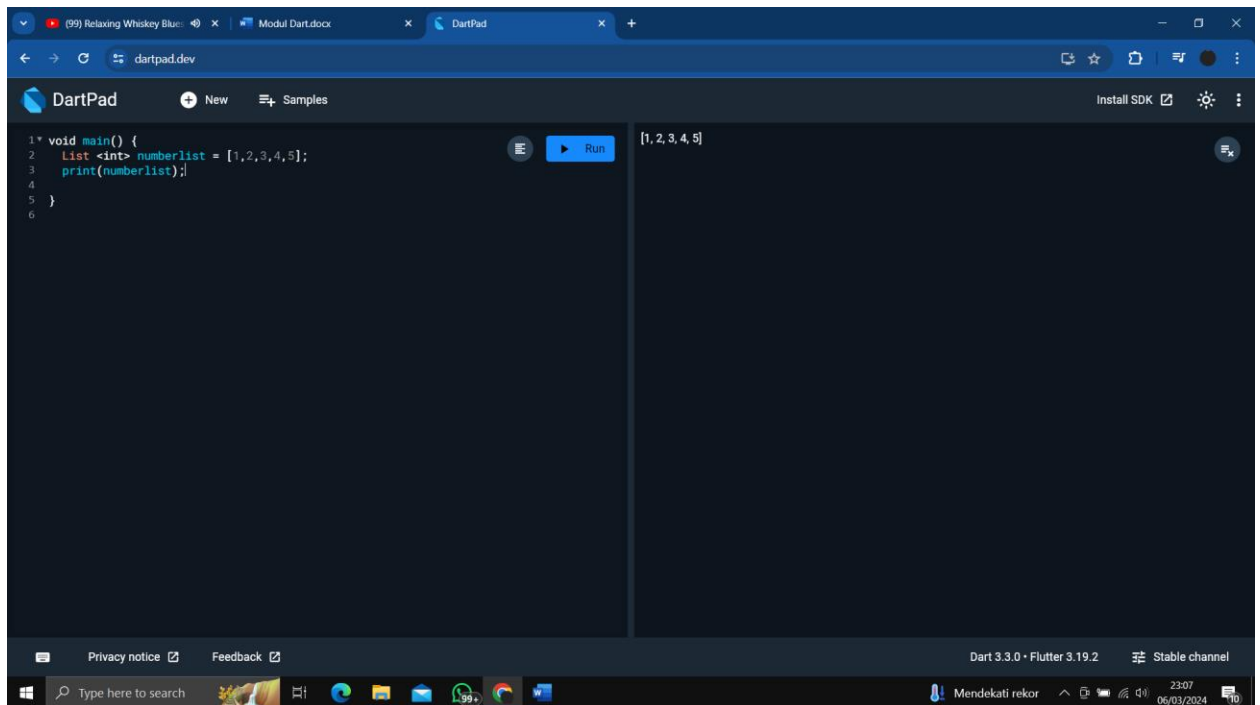
The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1* void main() {  
2  var i = 1;  
3* do{  
4    print(i);  
5    i++;  
6  }while(i <=100);  
7 }  
8  
9
```

The code is executed, and the output on the right shows a vertical list of numbers from 71 to 100, indicating the loop has completed its execution.

Perulangan do while hampir mirip dengan perulangan do. Namun pada perulangan ini program akan mengeksekusi statement terlebih dahulu kemudian baru memilih logika.

3. List

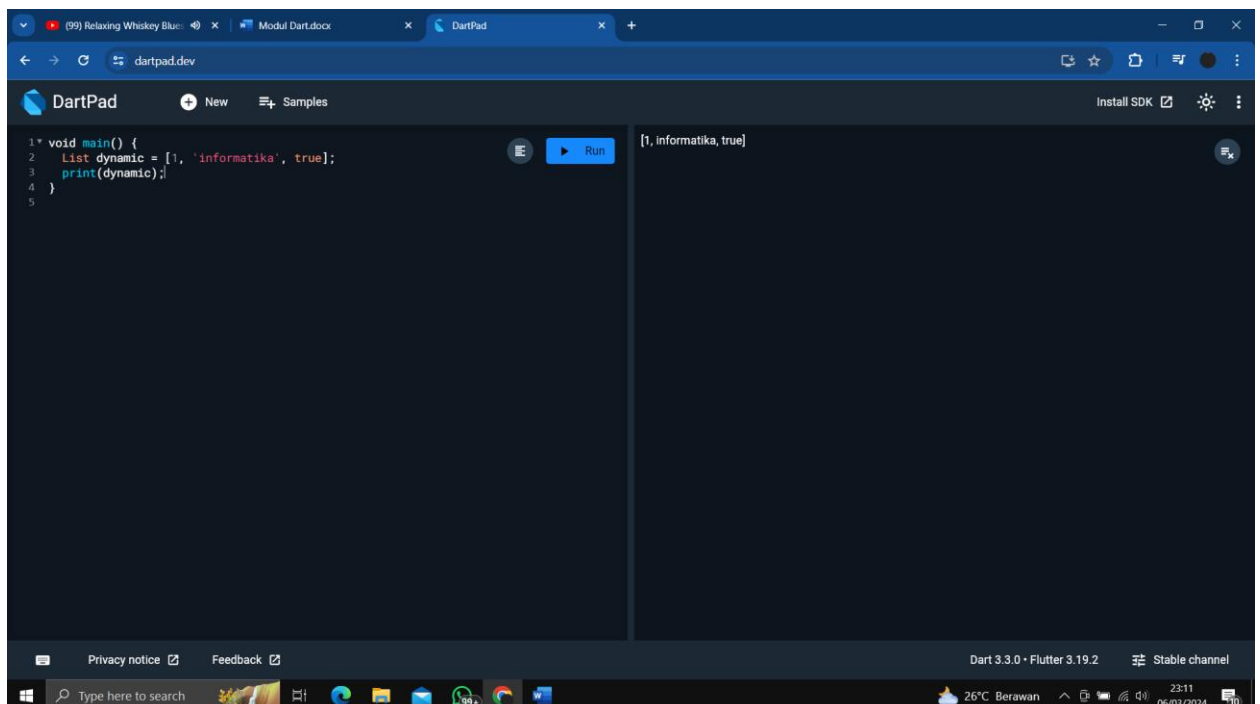


The screenshot shows the DartPad web interface. The code editor on the left contains the following Dart code:

```
1* void main() {  
2   List<int> numberlist = [1,2,3,4,5];  
3   print(numberlist);  
4  
5 }  
6
```

The output console on the right displays the result of the program execution: `[1, 2, 3, 4, 5]`. The interface includes a top navigation bar with 'New' and 'Samples' buttons, and a bottom status bar showing 'Dart 3.3.0 • Flutter 3.19.2' and 'Stable channel'.

Program diatas merupakan list. List sendiri sebenarnya hampir mirip dengan array, hanya saja jika array tipe datanya sama. Pada program diatas membuat list dengan tipe data int. setiap kali membuat list harus diawali dengan List.



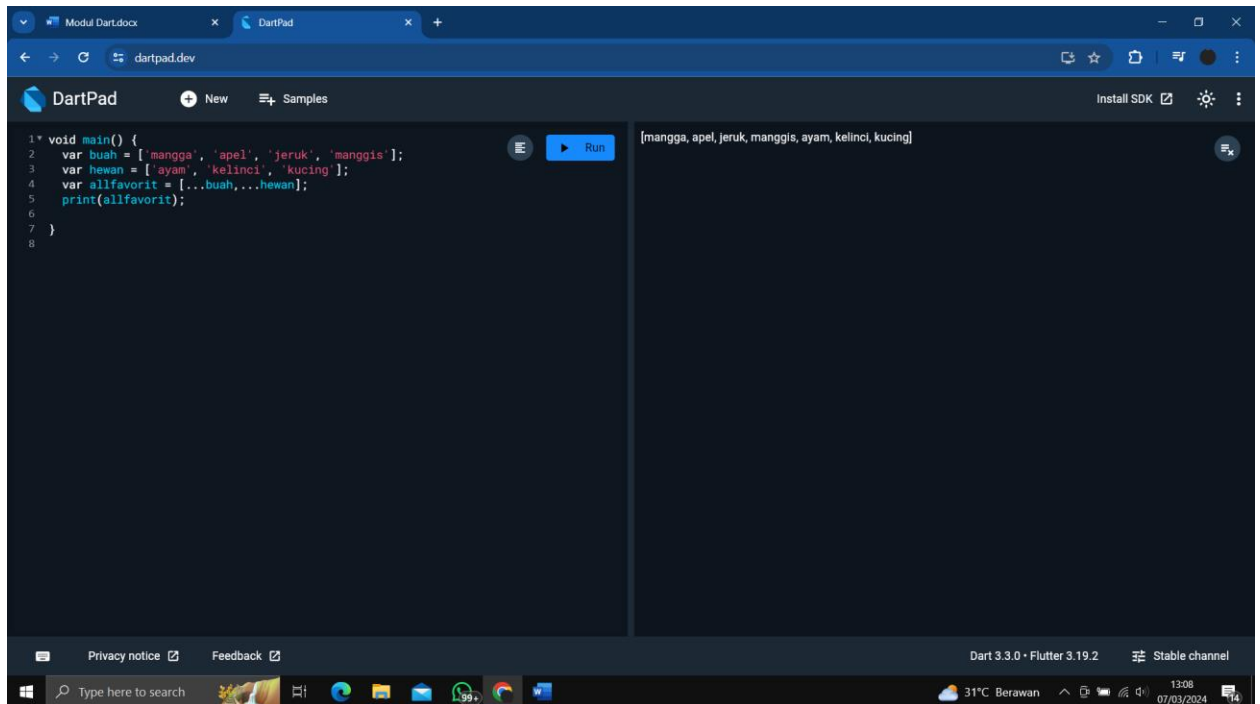
The screenshot shows the DartPad web interface. The code editor on the left contains the following Dart code:

```
1* void main() {  
2   List dynamic = [1, 'informatika', true];  
3   print(dynamic);  
4  
5 }
```

The output console on the right displays the result of the program execution: `[1, informatika, true]`. The interface includes a top navigation bar with 'New' and 'Samples' buttons, and a bottom status bar showing 'Dart 3.3.0 • Flutter 3.19.2' and 'Stable channel'.

List dynamic merupakan list yang tidak memiliki tipe data yang sama pada tiap elemennya. Pada contoh diatas merupakan list dynamic. Yang mana list dynamic dapat memiliki tipe data elemen int, string maupun Boolean.

4. Spread Operator



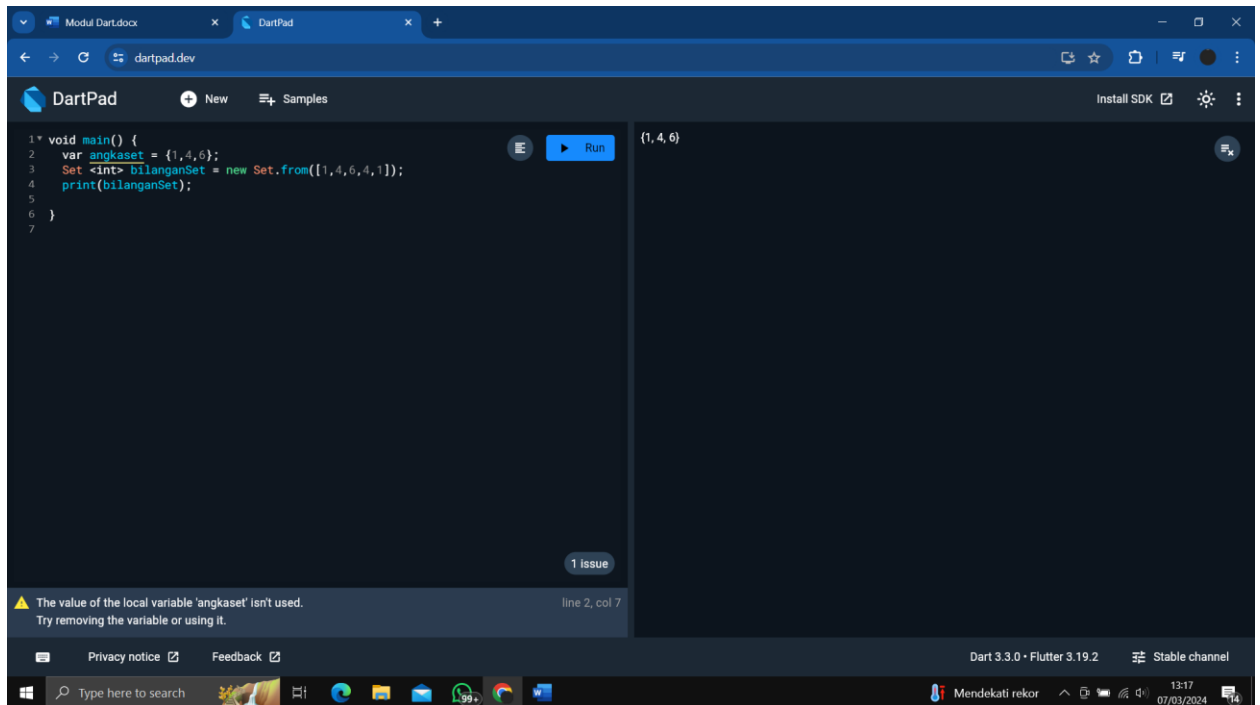
The screenshot shows the DartPad web interface. The code editor on the left contains the following Dart code:

```
1* void main() {  
2  var buah = ['mangga', 'apel', 'jeruk', 'manggis'];  
3  var hewan = ['ayam', 'kelinci', 'kucing'];  
4  var allfavorit = [...buah, ...hewan];  
5  print(allfavorit);  
6  
7 }  
8
```

The output console on the right displays the result of the print statement: `[mangga, apel, jeruk, manggis, ayam, kelinci, kucing]`. The bottom status bar indicates the environment is Dart 3.3.0 with Flutter 3.19.2 on the Stable channel.

Operator spread merupakan fitur yang ditandai dengan tanda titik 3 (...) yang digunakan untuk menggabungkan beberapa nilai dari beberapa list menjadi 1 list.

5. Set



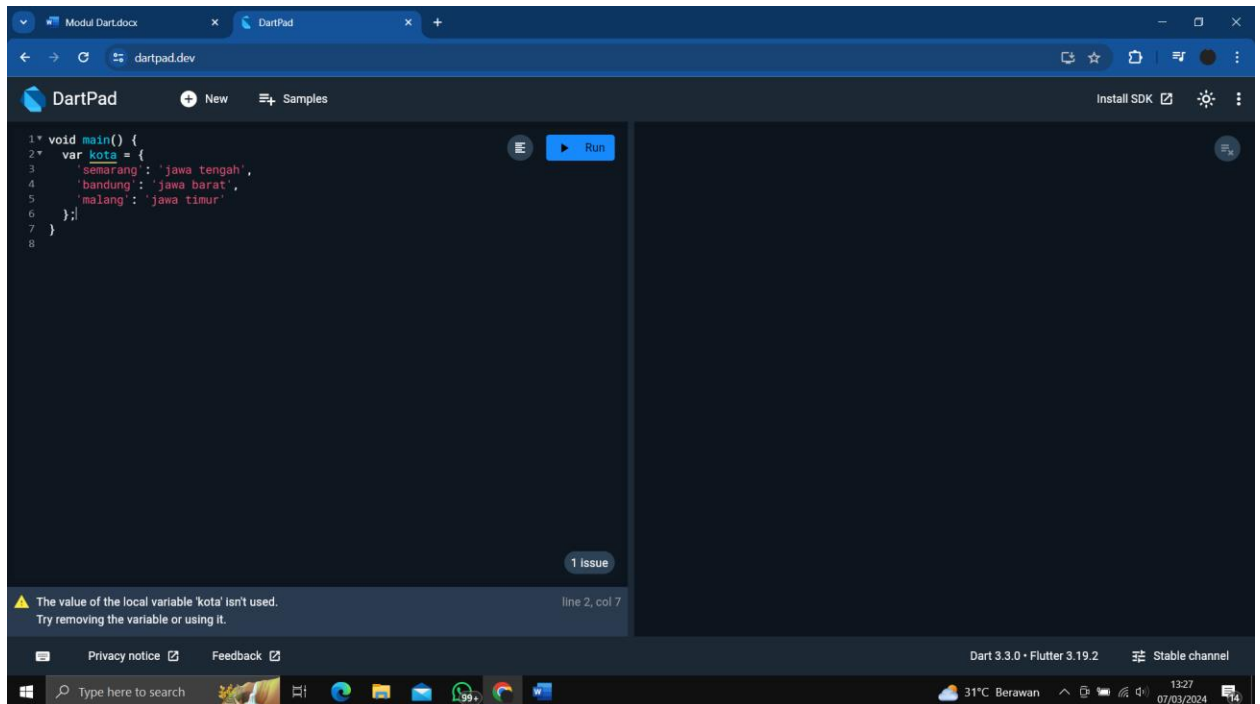
The screenshot shows the DartPad web interface. The code editor on the left contains the following Dart code:

```
1* void main() {  
2  var angkaset = {1,4,6};  
3  Set<int> bilanganSet = new Set.from([1,4,6,4,1]);  
4  print(bilanganSet);  
5  
6 }  
7
```

The output console on the right displays the result of the print statement: `{1, 4, 6}`. A warning message is visible at the bottom: "The value of the local variable 'angkaset' isn't used. Try removing the variable or using it." The bottom status bar indicates the environment is Dart 3.3.0 with Flutter 3.19.2 on the Stable channel.

Definisi set adalah koleksi yang menyimpan data unik saja. Pada program diatas dapat dilihat bahwa set bilangan bulat hanya menyimpan 1,4 dan 6 saja.

6. Map



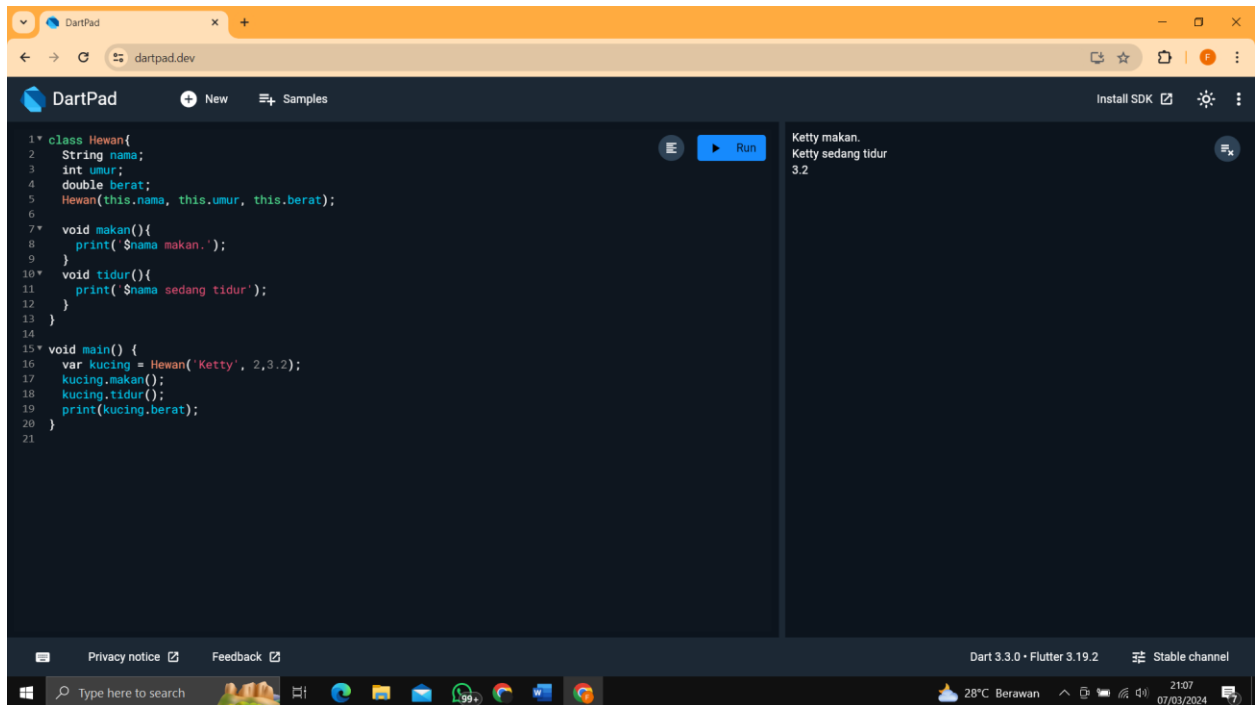
The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1* void main() {  
2*   var kota = {  
3*     'semarang': 'jawa tengah',  
4*     'bandung': 'jawa barat',  
5*     'malang': 'jawa timur'  
6*   };  
7* }  
8
```

A red error message is displayed at the bottom: "The value of the local variable 'kota' isn't used. Try removing the variable or using it." The status bar at the bottom indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

Map merupakan koleksi yang menyimpan data dengan format key. Pada program diatas, key yang dimaksud adalah semarang, bandung dan malang. Sedangkan jawa Tengah, jawa barat dan jawa timur merupakan value dari masing masing key.

7. Class



The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1* class Hewan{  
2*   String nama;  
3*   int umur;  
4*   double berat;  
5*   Hewan(this.nama, this.umur, this.berat);  
6*  
7*   void makan(){  
8*     print('$nama makan.');9*   }  
10*   void tidur(){  
11*     print('$nama sedang tidur');12*   }  
13* }  
14  
15* void main() {  
16*   var kucing = Hewan('Ketty', 2,3.2);  
17*   kucing.makan();  
18*   kucing.tidur();  
19*   print(kucing.berat);  
20* }  
21
```

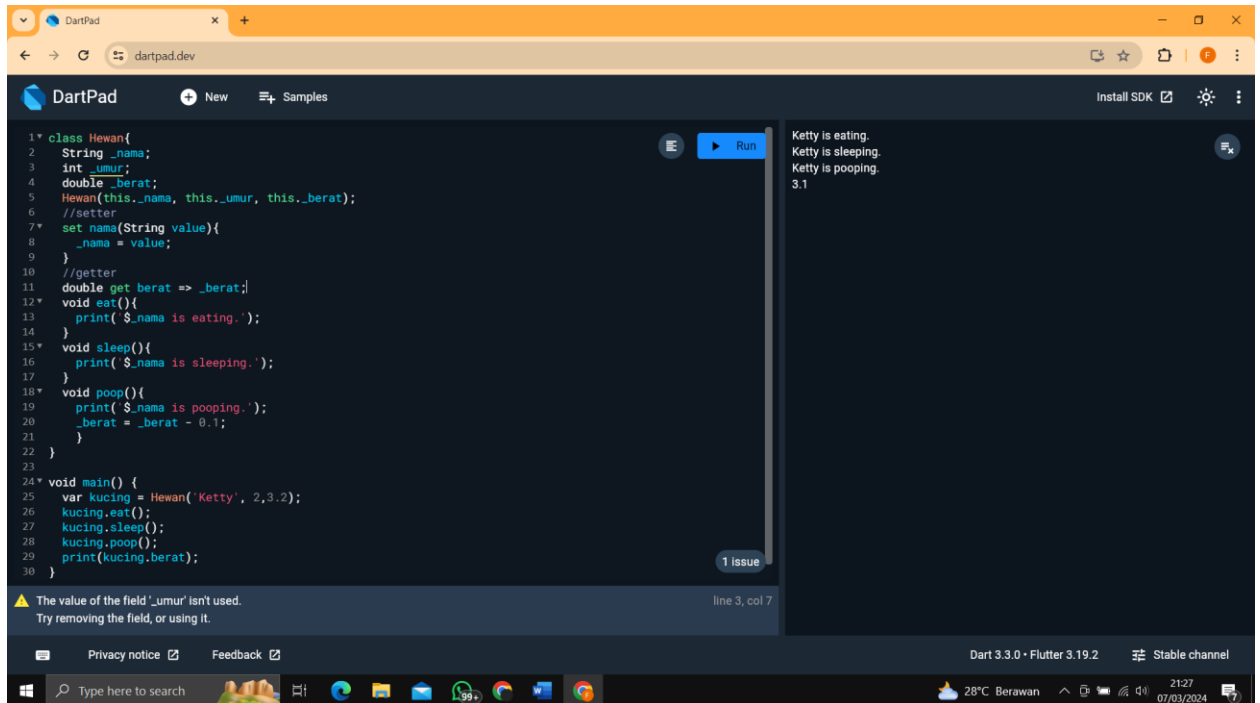
The output console on the right shows the following text:

```
Ketty makan.  
Ketty sedang tidur  
3.2
```

The status bar at the bottom indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

Pada program diatas terdapat kelas yaitu dengan nama Hewan. Didalam kelas Hewan terdapat beberapa properti seperti makan dan tidur. Lalu pada program utama, harus membuat objek, yaitu kucing. Lalu kita dapat memanggil masing masing metode melalui objek.

8. Properties & metode



```
1* class Hewan{
2  String _nama;
3  int _umur;
4  double _berat;
5  Hewan(this._nama, this._umur, this._berat);
6  //setter
7* set nama(String value){
8    _nama = value;
9  }
10 //getter
11 double get berat => _berat;
12* void eat(){
13   print('$nama is eating.');
```

```
24* void main() {
25   var kucing = Hewan('Ketty', 2,3.2);
26   kucing.eat();
27   kucing.sleep();
28   kucing.poop();
29   print(kucing.berat);
30 }
```

Ketty is eating.
Ketty is sleeping.
Ketty is pooping.
3.1

1 issue

The value of the field '_umur' isn't used.
Try removing the field, or using it.

line 3, col 7

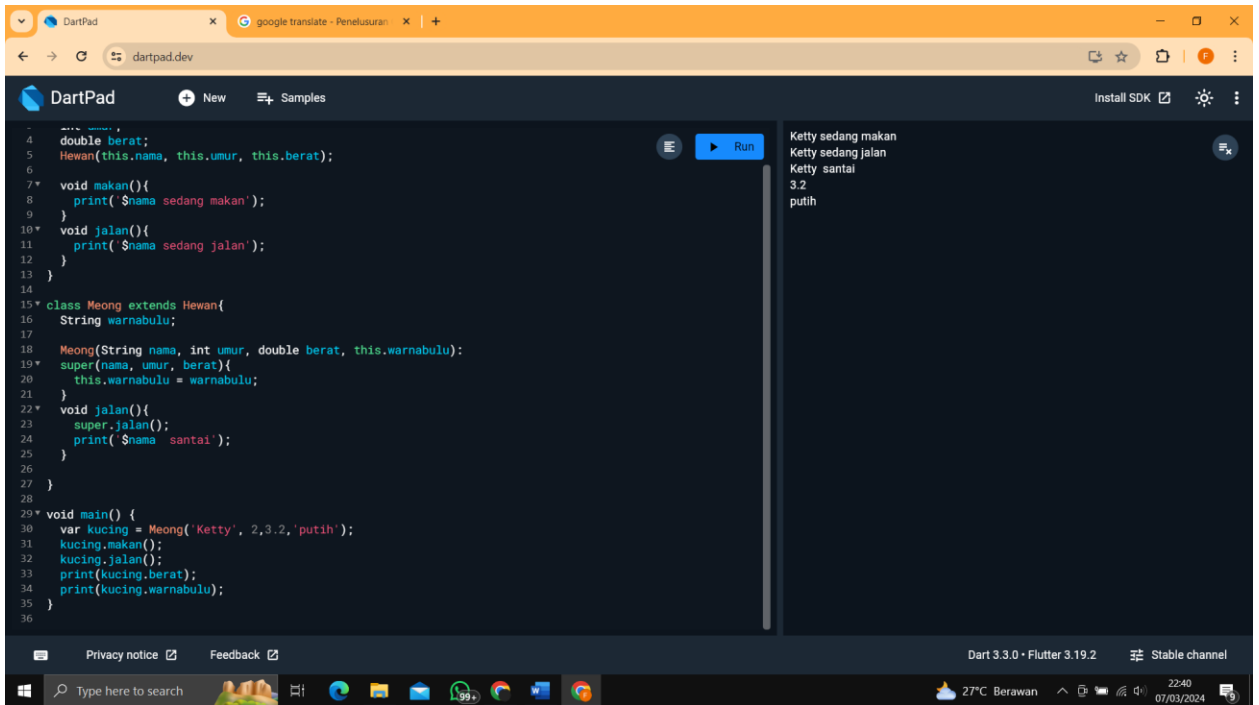
Privacy notice Feedback

Dart 3.3.0 • Flutter 3.19.2 Stable channel

28°C Berawan 21:27 07/03/2024

Program diatas sebenarnya hampir mirip dengan class hanya saja pada program ini ditambahkan underscore agar variable bersifat public dan dapat diakses dari mana saja.

9. Inheritance



```
1 // main()
2
3
4 double berat;
5 Hewan(this.nama, this.umur, this.berat);
6
7
8 void makan(){
9   print('$nama sedang makan');
10 }
11
12 void jalan(){
13   print('$nama sedang jalan');
14 }
15
16
17 class Meong extends Hewan{
18   String warnabulu;
19
20   Meong(String nama, int umur, double berat, this.warnabulu):
21     super(nama, umur, berat){
22     this.warnabulu = warnabulu;
23   }
24   void jalan(){
25     super.jalan();
26     print('$nama santai');
27   }
28 }
29
30 void main() {
31   var kucing = Meong('Ketty', 2,3.2,'putih');
32   kucing.makan();
33   kucing.jalan();
34   print(kucing.berat);
35   print(kucing.warnabulu);
36 }
```

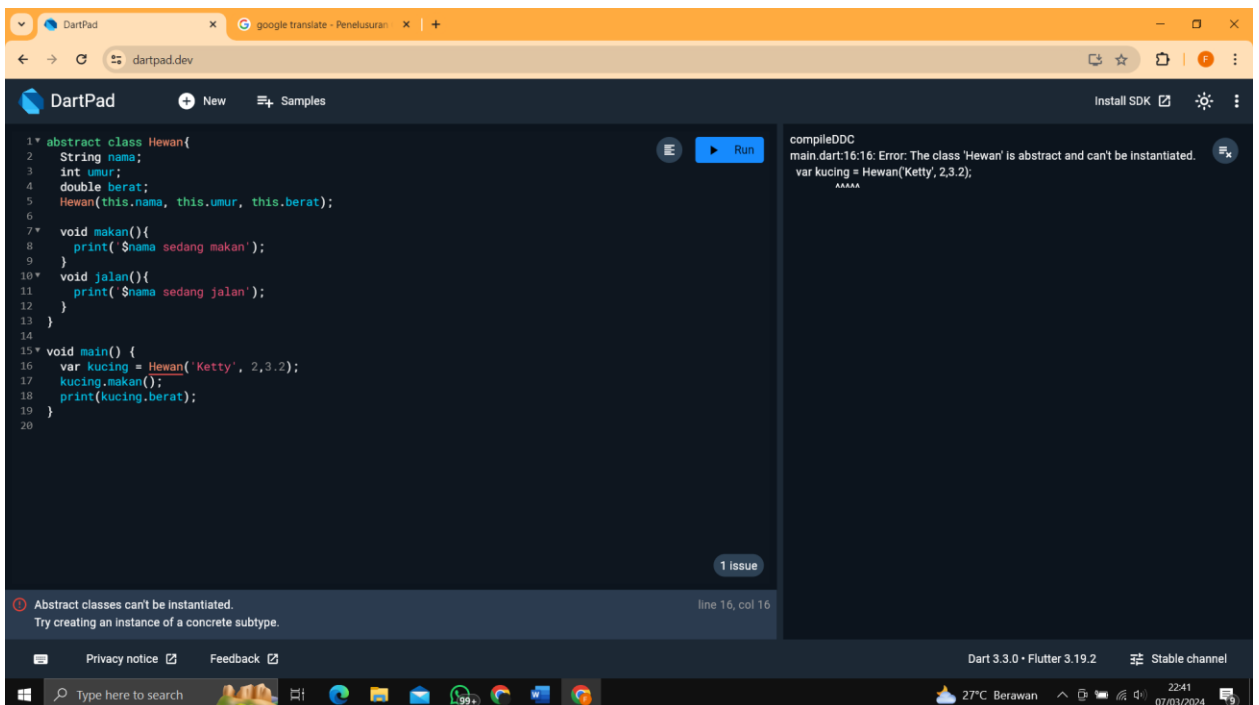
Ketty sedang makan
Ketty sedang jalan
Ketty santai
3.2
putih

Inheritance adalah pewarisan, pada program diatas kelas meong adalah pewarisan dari hewan.

Artinya setiap properti maupun variable yang dimiliki oleh meong dimiliki juga oleh hewan.

Namun sifat dari meong tidak dimiliki oleh hewan. Contohnya adalah string warna bulu.

10. Abstrac class



```
1 abstract class Hewan{
2   String nama;
3   int umur;
4   double berat;
5   Hewan(this.nama, this.umur, this.berat);
6
7   void makan(){
8     print('$nama sedang makan');
9   }
10  void jalan(){
11    print('$nama sedang jalan');
12  }
13 }
14
15 void main() {
16   var kucing = Hewan('Ketty', 2,3.2);
17   kucing.makan();
18   print(kucing.berat);
19 }
20
```

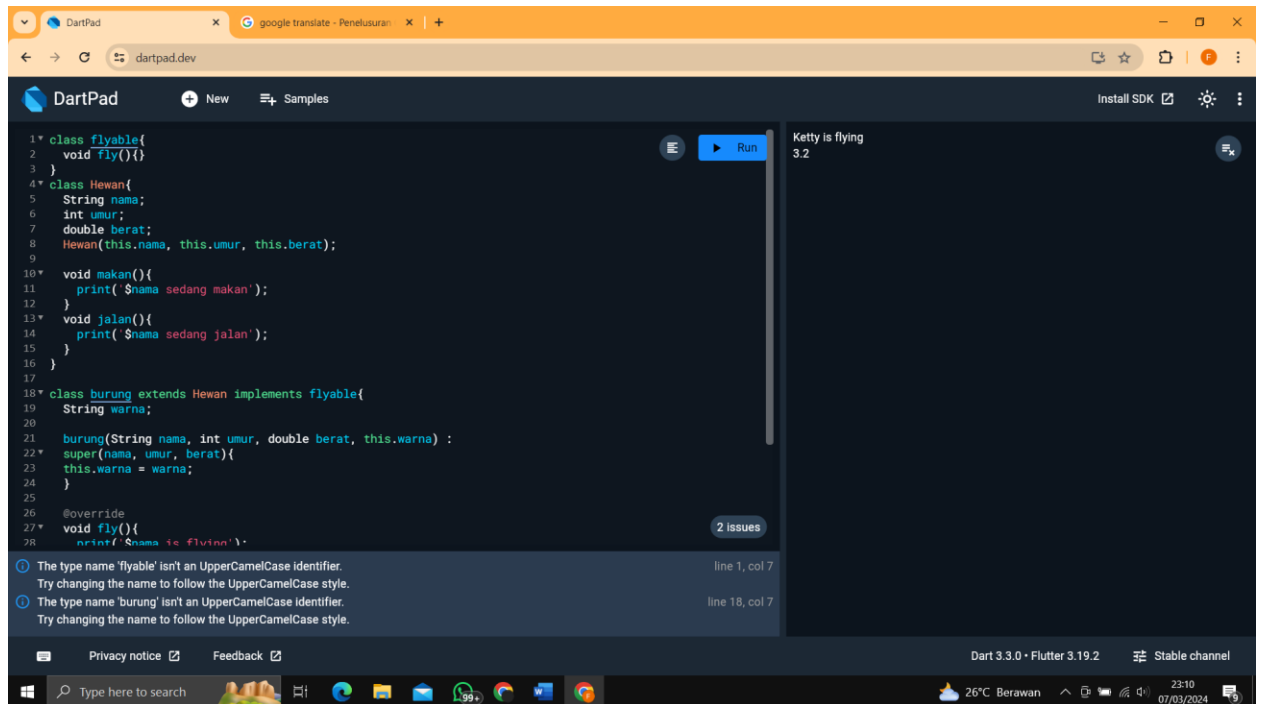
compileDDC
main.dart:16:16: Error: The class 'Hewan' is abstract and can't be instantiated.
var kucing = Hewan('Ketty', 2,3.2);
AAAAA

1 issue

Abstract classes can't be instantiated.
Try creating an instance of a concrete subtype.
line 16, col 16

Program diatas merupakan abstract kelas, yang mana tidak akan berjalan walaupun sudah direalisasikan dengan objek. Hasilnya akan tetap eror

11. Implicit interface



The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

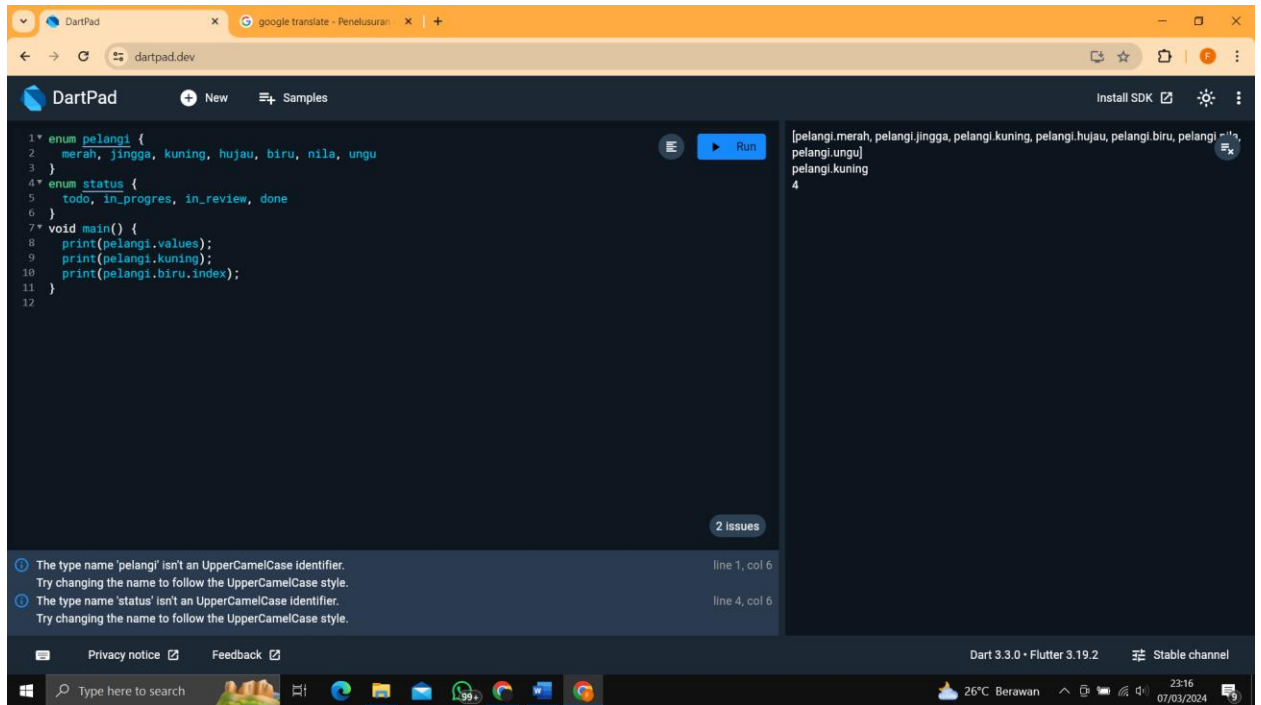
```
1 class flyable{
2   void fly(){
3   }
4 class Hewan{
5   String nama;
6   int umur;
7   double berat;
8   Hewan(this.nama, this.umur, this.berat);
9
10  void makan(){
11    print('$nama sedang makan');
12  }
13  void jalan(){
14    print('$nama sedang jalan');
15  }
16 }
17
18 class burung extends Hewan implements flyable{
19   String warna;
20
21   burung(String nama, int umur, double berat, this.warna) :
22     super(nama, umur, berat){
23     this.warna = warna;
24   }
25
26   @override
27   void fly(){
28     print('$nama is flying');
29   }
30 }
```

The console on the right shows the output: "Ketty is flying" and "3.2". At the bottom, there are two error messages:

- The type name 'flyable' isn't an UpperCamelCase identifier. Try changing the name to follow the UpperCamelCase style. (line 1, col 7)
- The type name 'burung' isn't an UpperCamelCase identifier. Try changing the name to follow the UpperCamelCase style. (line 18, col 7)

Sama halnya dengan kelas extends atau pewarisan, bahwa kelas extends mewarisi sifat dari induknya namun sifat dari kelas anak dapat dimodifikasi. Pada program diatas dapat menggunakan override.

12. Enumerated Types



```
1* enum pelangi {
2  merah, jingga, kuning, hijau, biru, nila, ungu
3 }
4* enum status {
5  todo, in_progres, in_review, done
6 }
7* void main() {
8  print(pelangi.values);
9  print(pelangi.kuning);
10 print(pelangi.biru.index);
11 }
12 }
```

[pelangi.merah, pelangi.jingga, pelangi.kuning, pelangi.hijau, pelangi.biru, pelangi.nila, pelangi.ungu]
pelangi.kuning
4

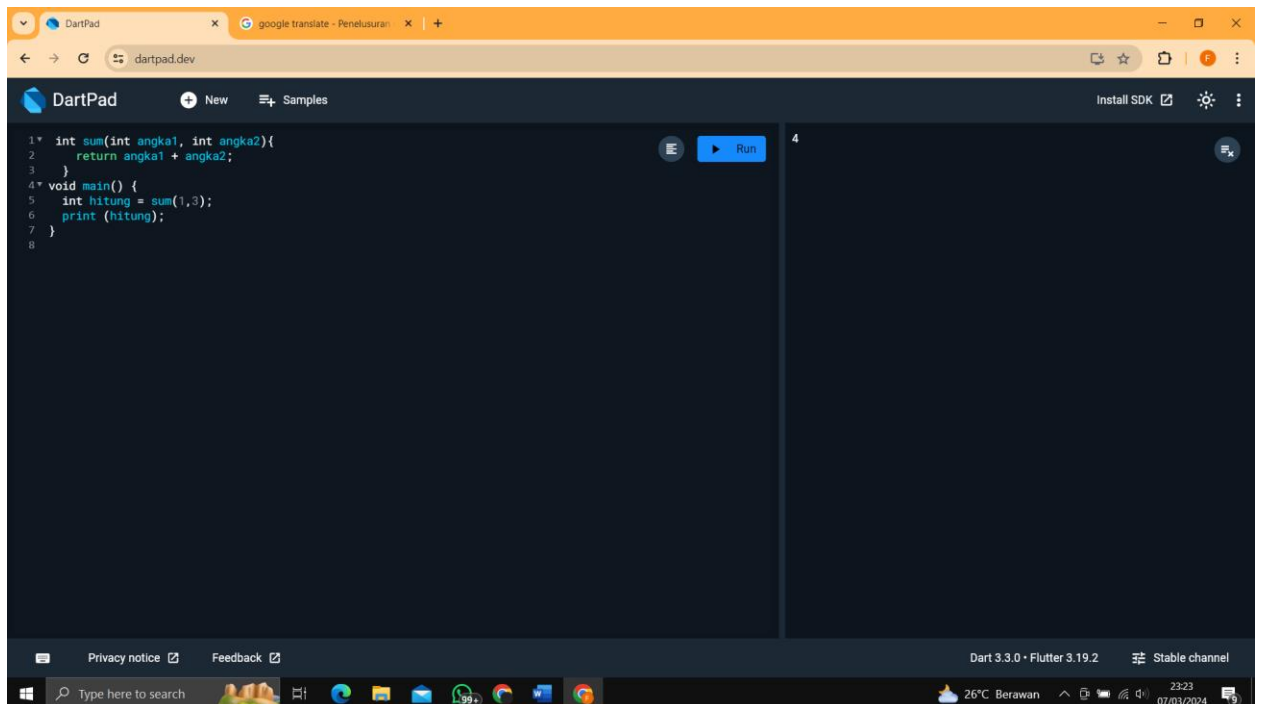
2 issues

- The type name 'pelangi' isn't an UpperCamelCase identifier. Try changing the name to follow the UpperCamelCase style. line 1, col 6
- The type name 'status' isn't an UpperCamelCase identifier. Try changing the name to follow the UpperCamelCase style. line 4, col 6

Privacy notice Feedback Dart 3.3.0 • Flutter 3.19.2 Stable channel

Enumerate types sebenarnya hampir mirip dengan list namun dengan enumerate types dapat melihat index dan item secara menyeluruh.

13. Paradigma Function Program



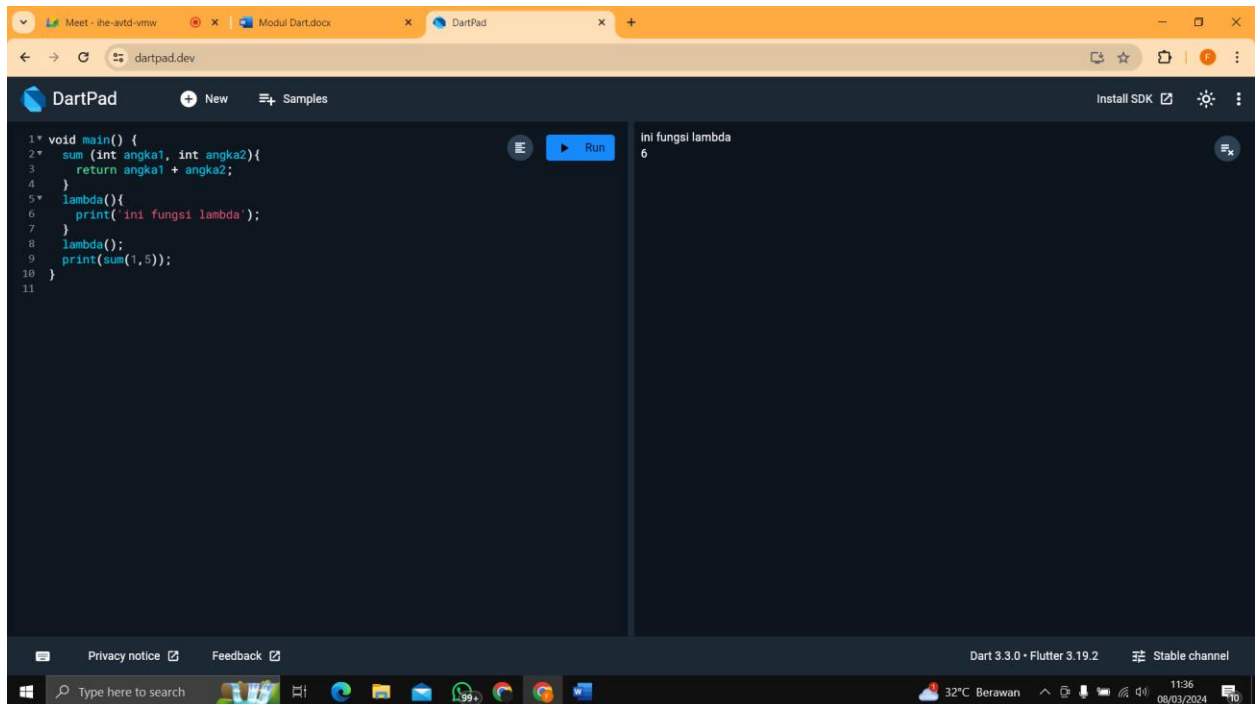
```
1* int sum(int angka1, int angka2){
2  return angka1 + angka2;
3 }
4* void main() {
5  int hitung = sum(1,3);
6  print (hitung);
7 }
8 }
```

4

Privacy notice Feedback Dart 3.3.0 • Flutter 3.19.2 Stable channel

Pada program diatas pure fungsi hanya akan menjalankan tugas sesuai dengan apa yang dimasukan pada parameter. Pada program tersebut akan menjalankan perintah angka1 + angka2.

14. Anonymous function



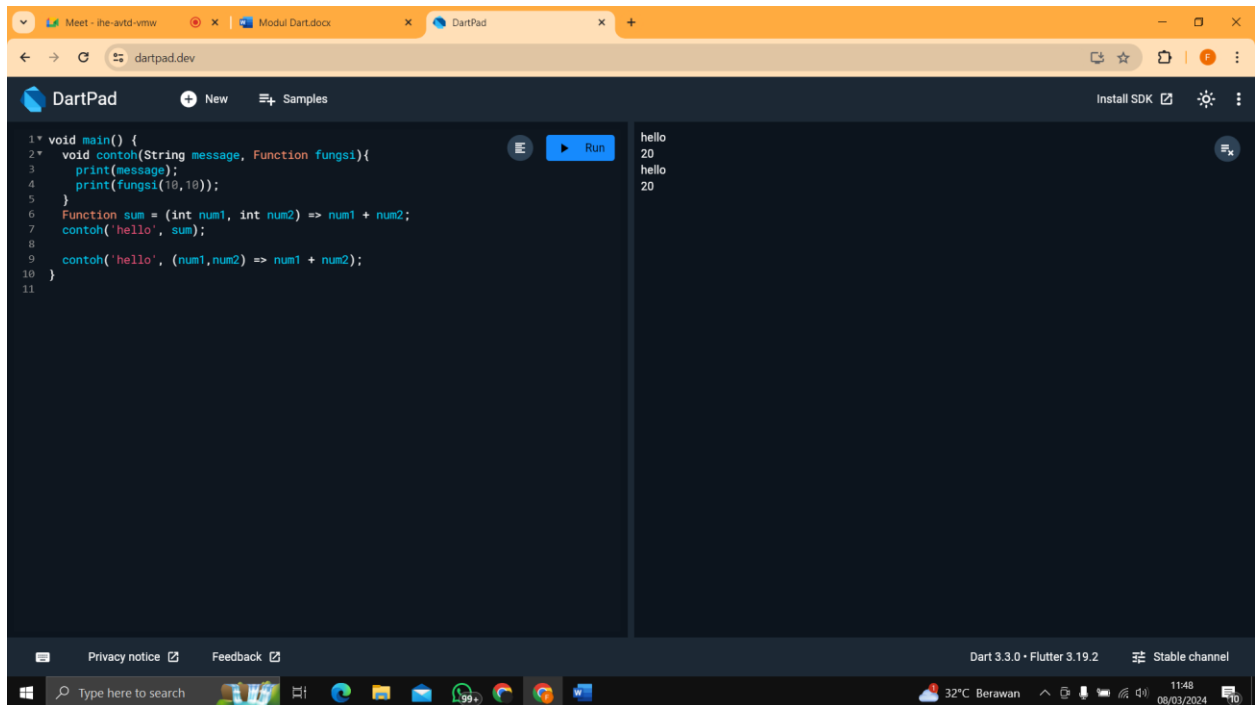
The screenshot shows the DartPad web interface. The left pane contains the following Dart code:

```
1* void main() {  
2*   sum (int angka1, int angka2){  
3     return angka1 + angka2;  
4   }  
5*   lambda(){  
6     print('ini fungsi lambda');  
7   }  
8   lambda();  
9   print(sum(1,5));  
10 }  
11 }
```

The right pane shows the output: "ini fungsi lambda" followed by "6". The status bar at the bottom indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

Berdasarkan definisi anonymous function merupakan lambda, function ini juga dapat menggunakan exspresi agar lebih singkat dengan menggunakan syntax =>(fat arrow).

15. Higer-order function



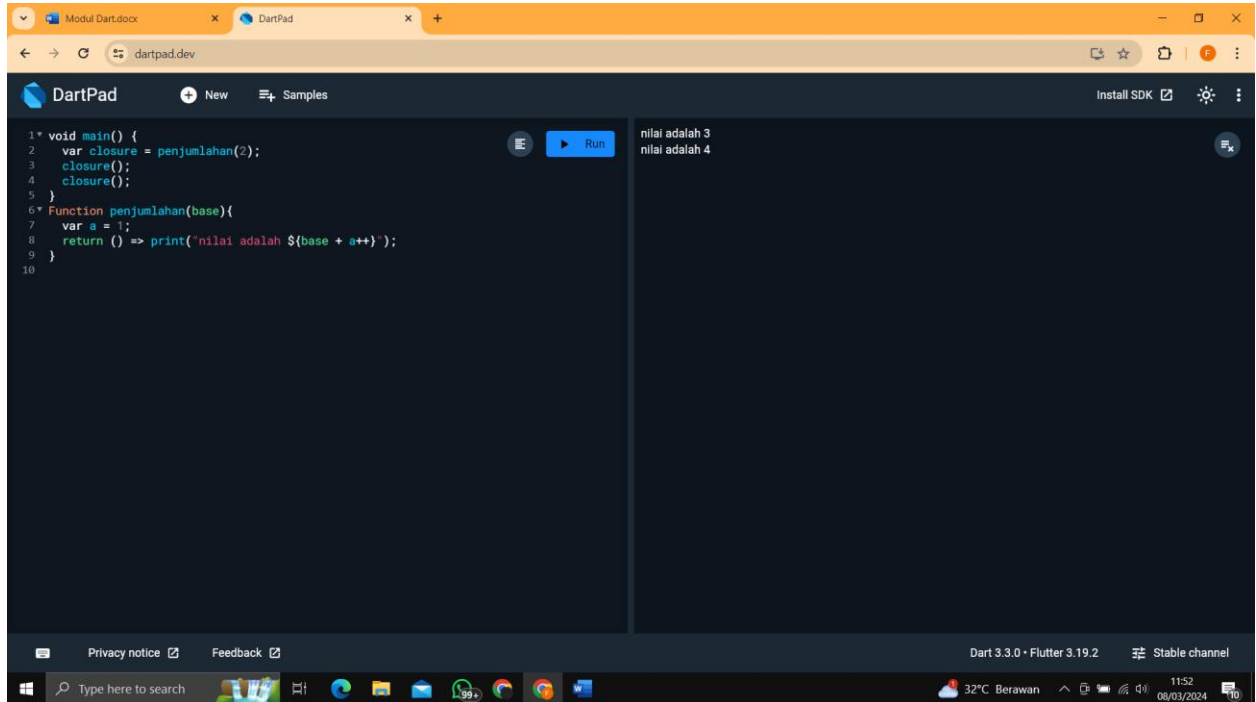
The screenshot shows the DartPad web interface. The left pane contains the following Dart code:

```
1* void main() {  
2*   void contoh(String message, Function fungsi){  
3     print(message);  
4     print(fungsi(10,10));  
5   }  
6   Function sum = (int num1, int num2) => num1 + num2;  
7   contoh('hello', sum);  
8  
9   contoh('hello', (num1,num2) => num1 + num2);  
10 }  
11 }
```

The right pane shows the output: "hello", "20", "hello", "20". The status bar at the bottom indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

Berdasarkan definisi higer-order merupakan sebuah fungsi yang menjadikan fungsi lainnya sebagai parameter. Pada program diatas dapat dilihat bahwa fungsi contoh menjadikan fungsi sum sebagai parameternya.

16. Closures

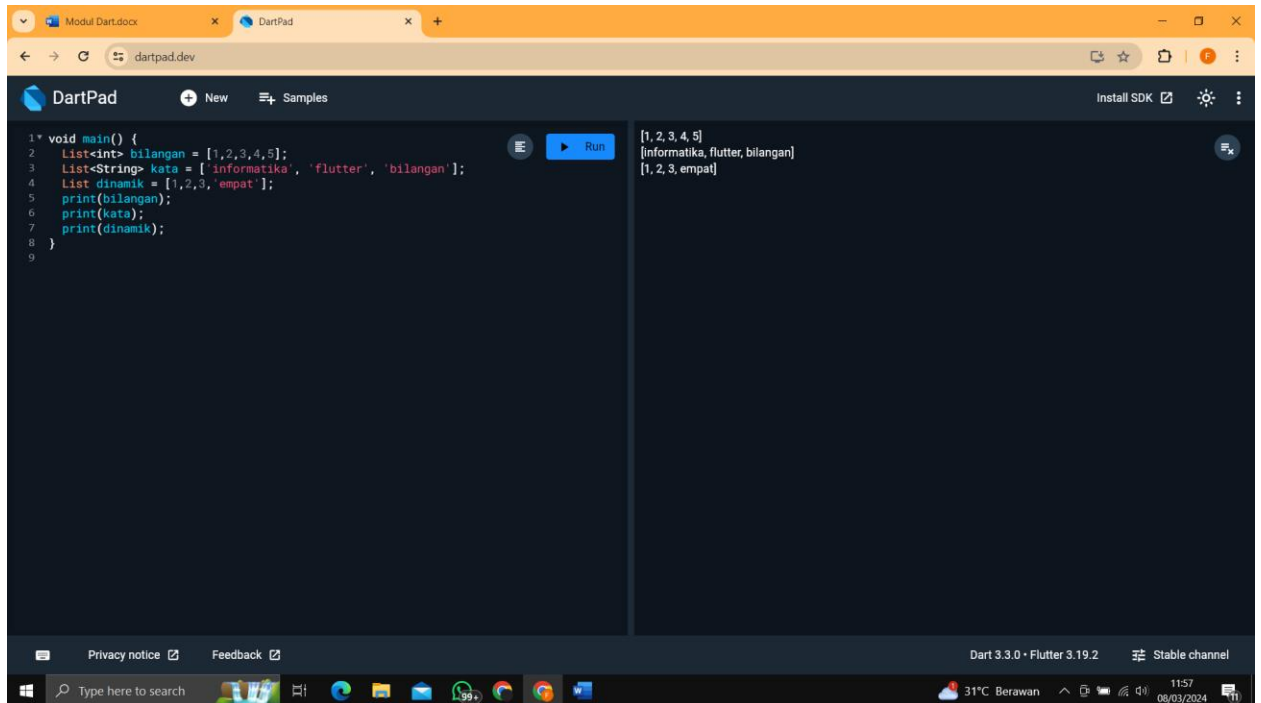


```
1* void main() {  
2  var closure = penjumlahan(2);  
3  closure();  
4  closure();  
5 }  
6* Function penjumlahan(base){  
7  var a = 1;  
8  return () => print('nilai adalah ${base + a++}');  
9 }  
10 }
```

nilai adalah 3
nilai adalah 4

fungsi yang dapat mengakses variabel di dalam lexical scope-nya. Lexical scope berarti pada sebuah fungsi bersarang, fungsi yang berada di dalam memiliki akses ke variabel di lingkupindiknya.

17. Generic

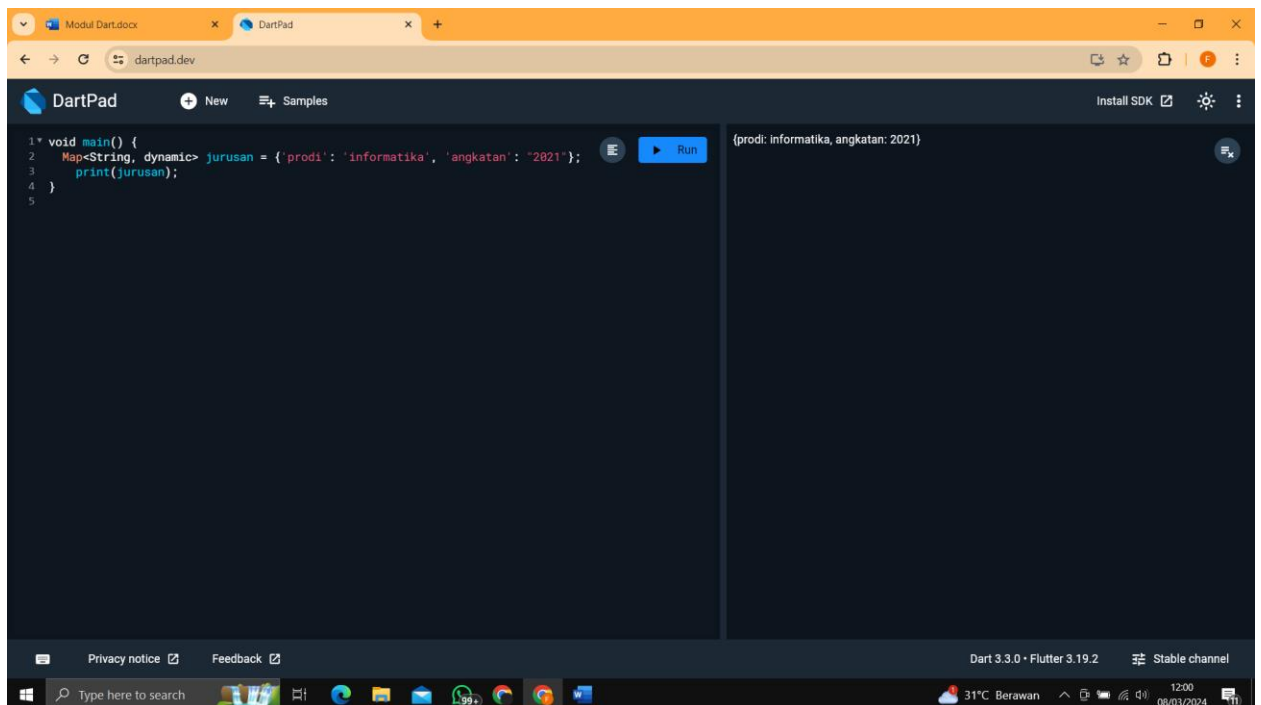


```
1* void main() {  
2   List<int> bilangan = [1,2,3,4,5];  
3   List<String> kata = ['informatika', 'flutter', 'bilangan'];  
4   List<dynamic> dinamik = [1,2,3, empat];  
5   print(bilangan);  
6   print(kata);  
7   print(dinamik);  
8 }  
9
```

[1, 2, 3, 4, 5]
[informatika, flutter, bilangan]
[1, 2, 3, empat]

Generic merupakan konsep yang digunakan untuk menentukan tipe data yang akan kita gunakan. Kita dapat mengganti tipe parameter generic pada Dart dengan lebih spesifik. Pada contoh program diatas kita dapat membuat konsep generic secara dinamis. Artinya tipe data yang digunakan pada tiap elemen-elemen list dapat berbeda beda.

18. Type inference

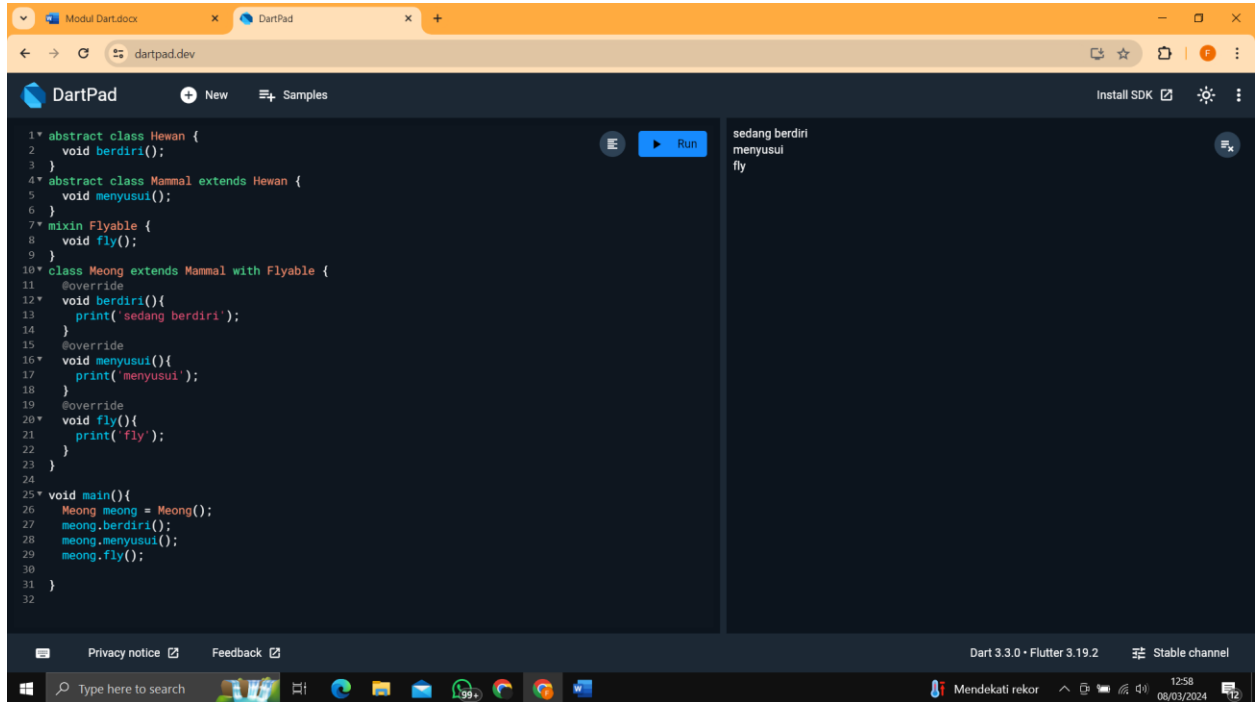


```
1* void main() {  
2   Map<String, dynamic> jurusan = {'prodi': 'informatika', 'angkatan': '2021'};  
3   print(jurusan);  
4 }  
5
```

{prodi: informatika, angkatan: 2021}

Dart memiliki analyzer yang dapat menentukan tipe untuk field, method, variabel lokal, dan beberapa tipe argumen generic. Maksudnya adalah key diatas memiliki tipe data string namun namun valuenya bersifat dinamis.

19. Effective dart



The screenshot shows the DartPad web interface. The code editor on the left contains the following Dart code:

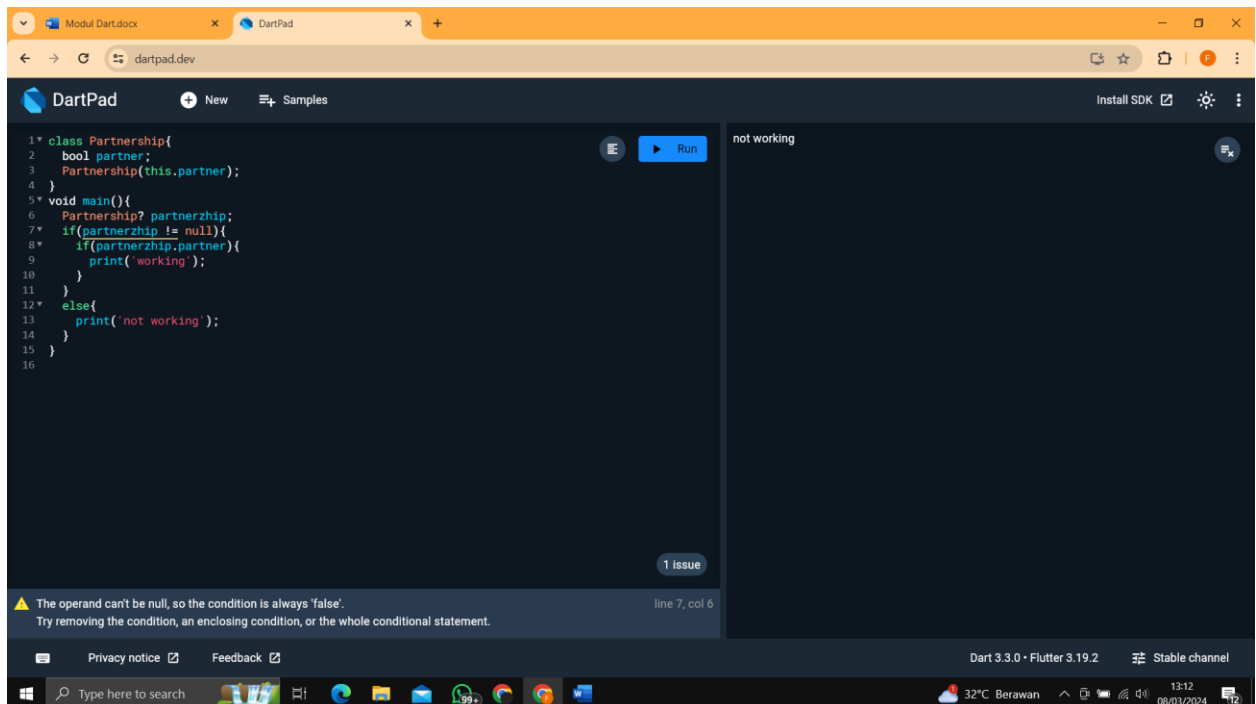
```
1* abstract class Hewan {
2  void berdiri();
3 }
4* abstract class Mammal extends Hewan {
5  void menyusui();
6 }
7* mixin Flyable {
8  void fly();
9 }
10* class Meong extends Mammal with Flyable {
11  @override
12  void berdiri(){
13    print('sedang berdiri');
14  }
15  @override
16  void menyusui(){
17    print('menyusui');
18  }
19  @override
20  void fly(){
21    print('fly');
22  }
23 }
24
25* void main(){
26  Meong meong = Meong();
27  meong.berdiri();
28  meong.menyusui();
29  meong.fly();
30 }
31
32
```

The output on the right shows the result of running the code:

```
sedang berdiri
menyusui
fly
```

The bottom status bar indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

Dengan DO kita dapat membuat sebuah kelas turunan yang seperti mengimpkementasikan suatu methode dengan syarat override.



The screenshot shows the DartPad web interface. The code editor on the left contains the following Dart code:

```
1* class Partnership{
2  bool partner;
3  Partnership(this.partner);
4 }
5* void main(){
6  Partnership? partnership;
7  if(partnership != null){
8    if(partnership.partner){
9      print('working');
10   }
11 }
12* else{
13  print('not working');
14 }
15 }
16
```

The output on the right shows the result of running the code:

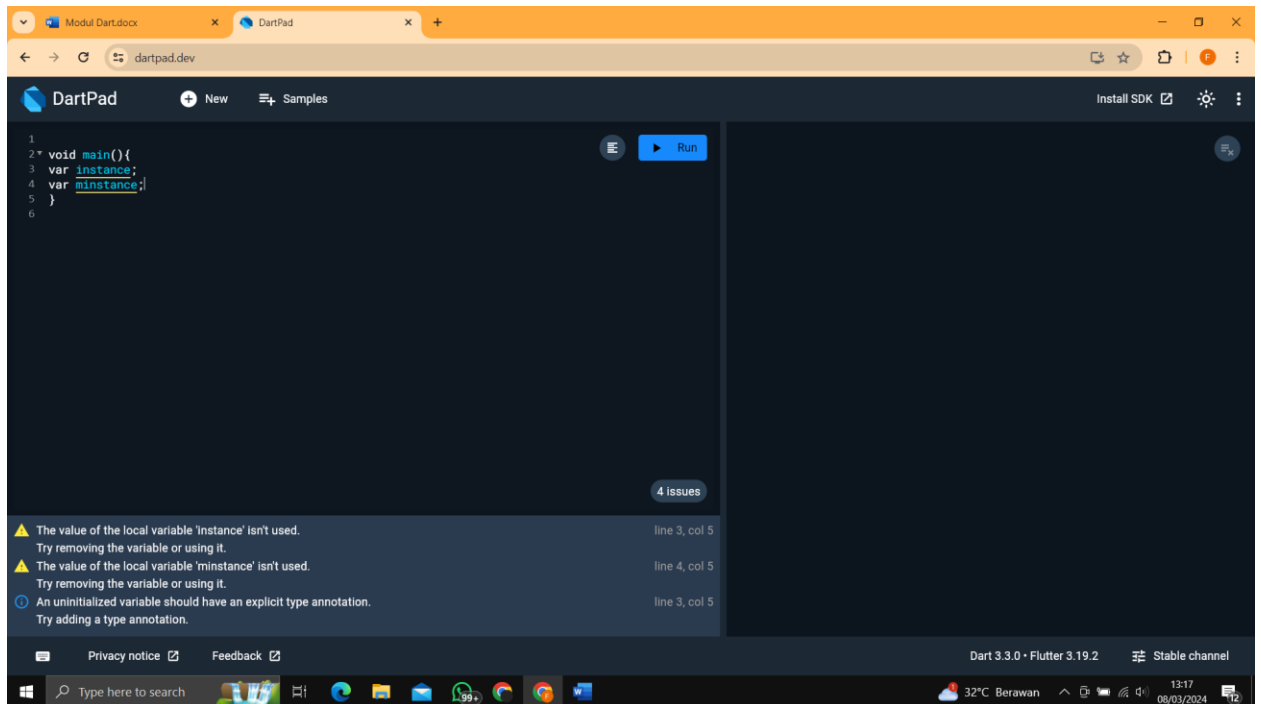
```
not working
```

A red error message is displayed at the bottom:

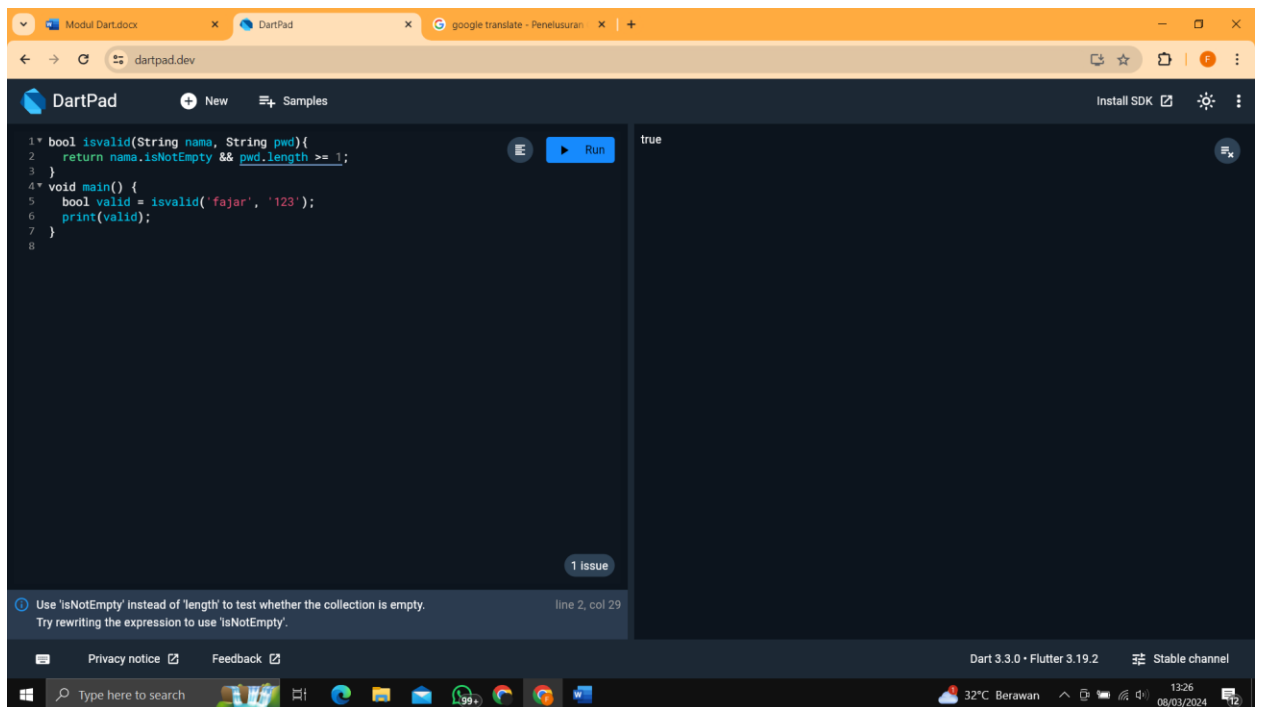
```
1 issue
The operand can't be null, so the condition is always 'false'.
Try removing the condition, an enclosing condition, or the whole conditional statement.
line 7, col 6
```

The bottom status bar indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

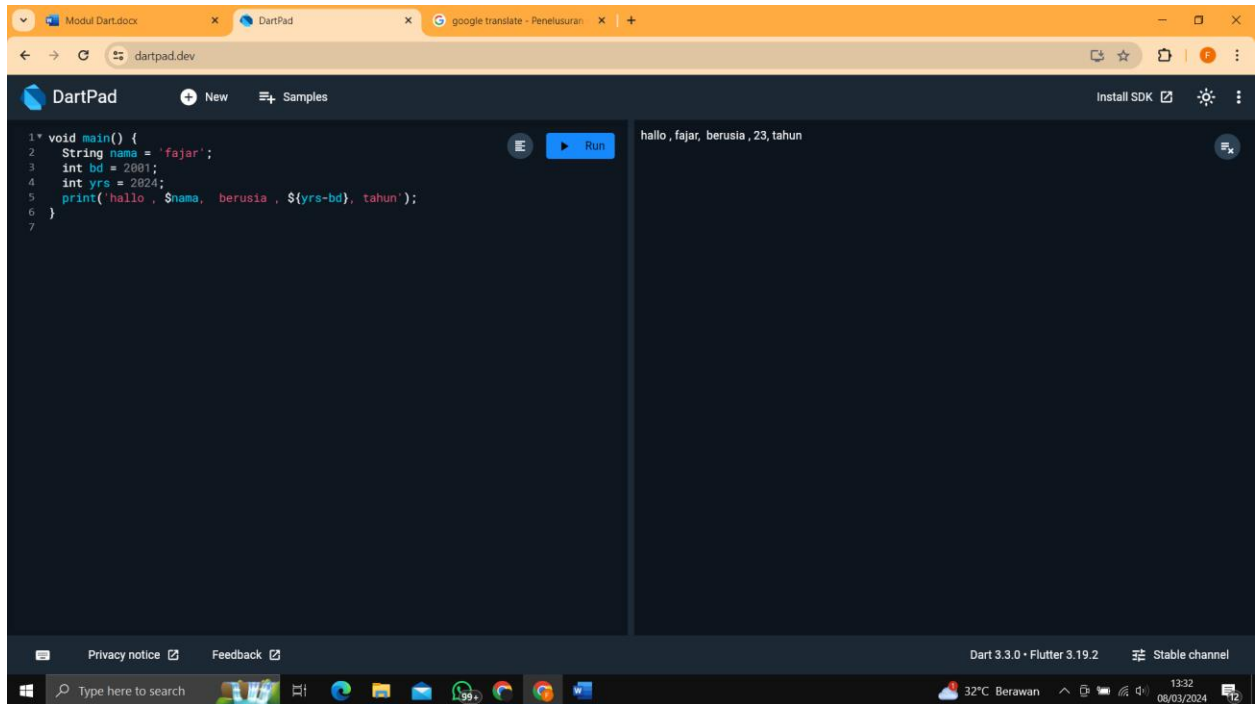
DO dengan DO USE sebenarnya hampir mirip, kita dapat membuah sebuah percabangan didalam program utama. Hanya saja kode program diatas kelasnya tidak diturunkan.



Program diatas menjunkan bahwa variable instance baik digunakan sedangkan minstance tidak baik untuk digunakan.



Menurut definisi prefer adalah keadaan yang mungkin bisa masuk akal untuk melakukan sebaliknya. Pada program diatas kita dapat memasukan nama dan password selama memenuhi syarat sehingga hasilnya adalah true.

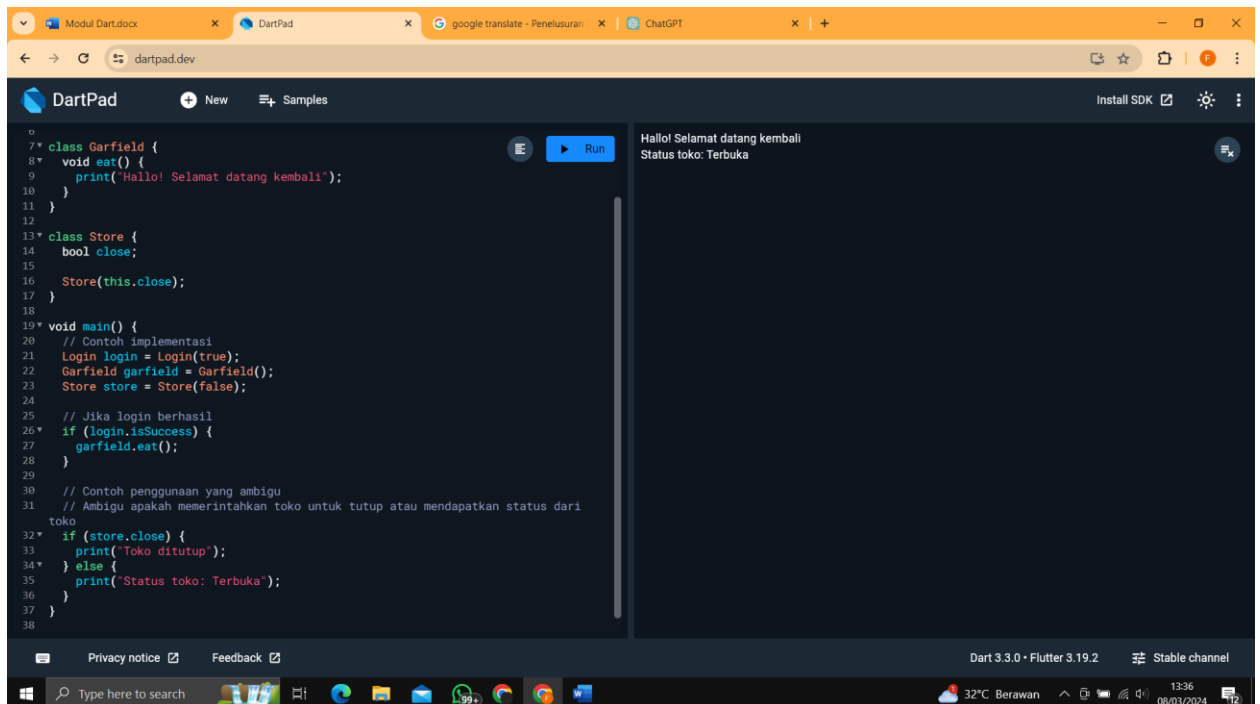


The screenshot shows the DartPad web interface in a browser. The left pane contains the following Dart code:

```
1* void main() {  
2  String nama = 'fajar';  
3  int bd = 2001;  
4  int yrs = 2024;  
5  print('hallo , $nama, berusia , ${yrs-bd}, tahun');  
6 }  
7
```

The right pane shows the output: "hallo ,fajar, berusia , 23, tahun". The bottom status bar indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

Avoid adalah kebalikan dari PREFER yang menjelaskan hal-hal yang tidak boleh dilakukan, namun kemungkinan ada alasan bagus untuk melakukannya pada beberapa kejadian.



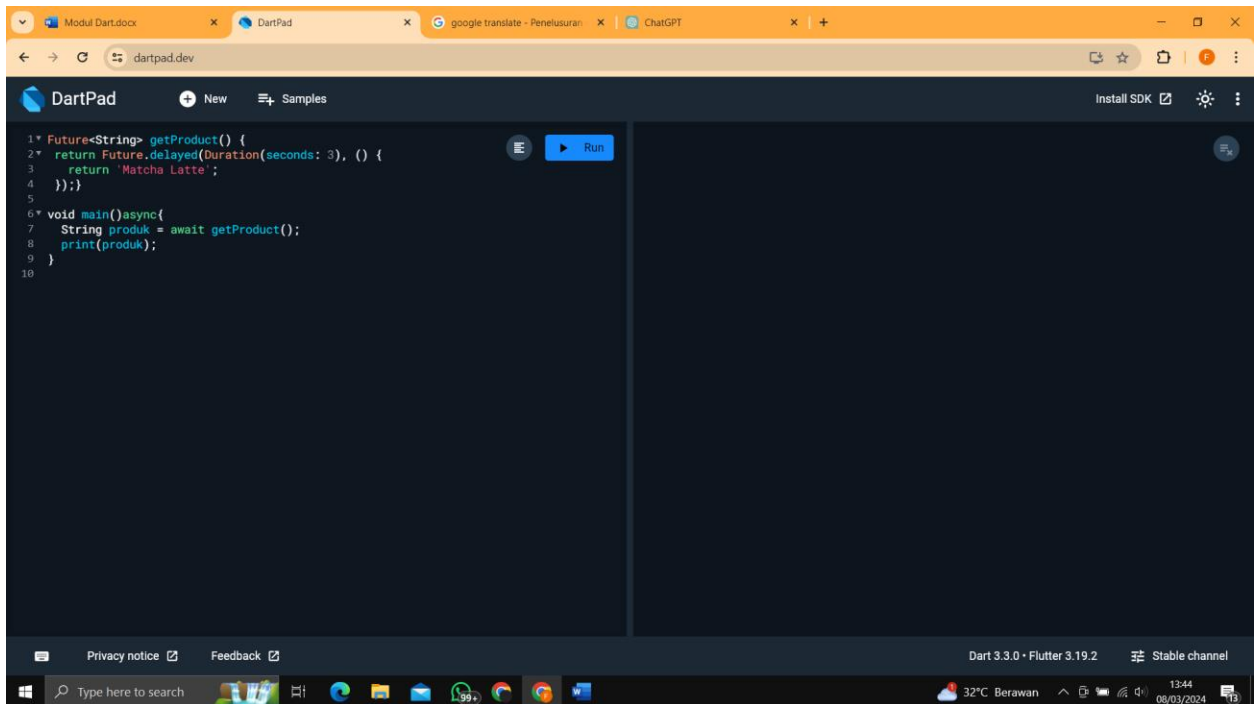
The screenshot shows the DartPad web interface with a more complex Dart program. The left pane contains the following code:

```
0  
7* class Garfield {  
8  void eat() {  
9    print("Hallo! Selamat datang kembali");  
10 }  
11 }  
12  
13* class Store {  
14  bool close;  
15  
16  Store(this.close);  
17 }  
18  
19* void main() {  
20  // Contoh implementasi  
21  Login login = Login(true);  
22  Garfield garfield = Garfield();  
23  Store store = Store(false);  
24  
25  // Jika login berhasil  
26* if (login.isSuccess) {  
27    garfield.eat();  
28  }  
29  
30  // Contoh penggunaan yang ambigu  
31  // Ambigu apakah memerintahkan toko untuk tutup atau mendapatkan status dari toko  
32* if (store.close) {  
33    print("Toko ditutup");  
34* } else {  
35    print("Status toko: Terbuka");  
36  }  
37 }  
38
```

The right pane shows the output: "Hallo! Selamat datang kembali" and "Status toko: Terbuka". The bottom status bar indicates "Dart 3.3.0 • Flutter 3.19.2" and "Stable channel".

Pada kode program diatas, output tergantung pada kondisi. Apakah hasil dapat diikuti atau tidak.

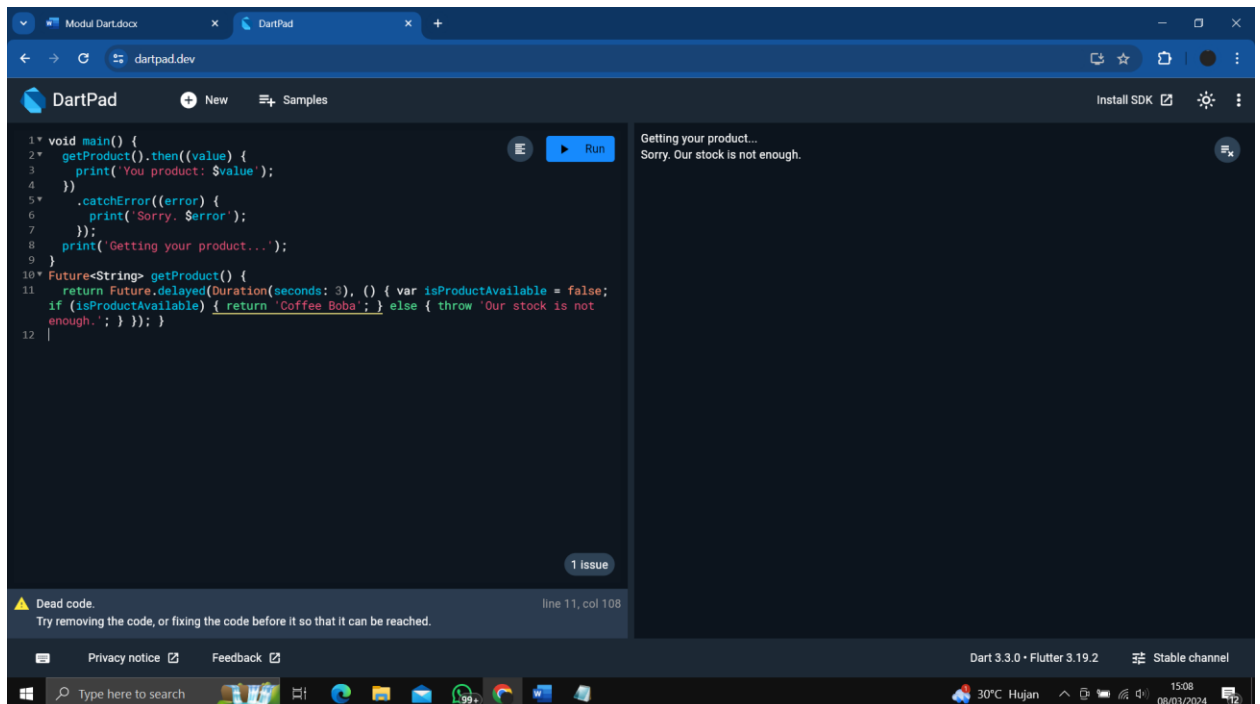
20. Future



```
1* Future<String> getProduct() {
2*   return Future.delayed(Duration(seconds: 3), () {
3*     return 'Matcha Latte';
4*   });
5* }
6* void main() async {
7*   String produk = await getProduct();
8*   print(produk);
9* }
10
```

Dengan menggunakan konsep future sebuah program dapat mewakili nilai potensial keasalahan yang akan terjadi. Maksudnya adalah kode program atas akan delay selama 3 detik untuk menampilkan hasilnya.

21. Completed with error



```
1* void main() {
2*   getProduct().then((value) {
3*     print('You product: $value');
4*   })
5*   .catchError((error) {
6*     print('Sorry. $error');
7*   });
8*   print('Getting your product...');
9* }
10* Future<String> getProduct() {
11*   return Future.delayed(Duration(seconds: 3), () { var isProductAvailable = false;
12*     if (isProductAvailable) { return 'Coffee Boba'; } else { throw 'Our stock is not enough.'; } }); }
13
```

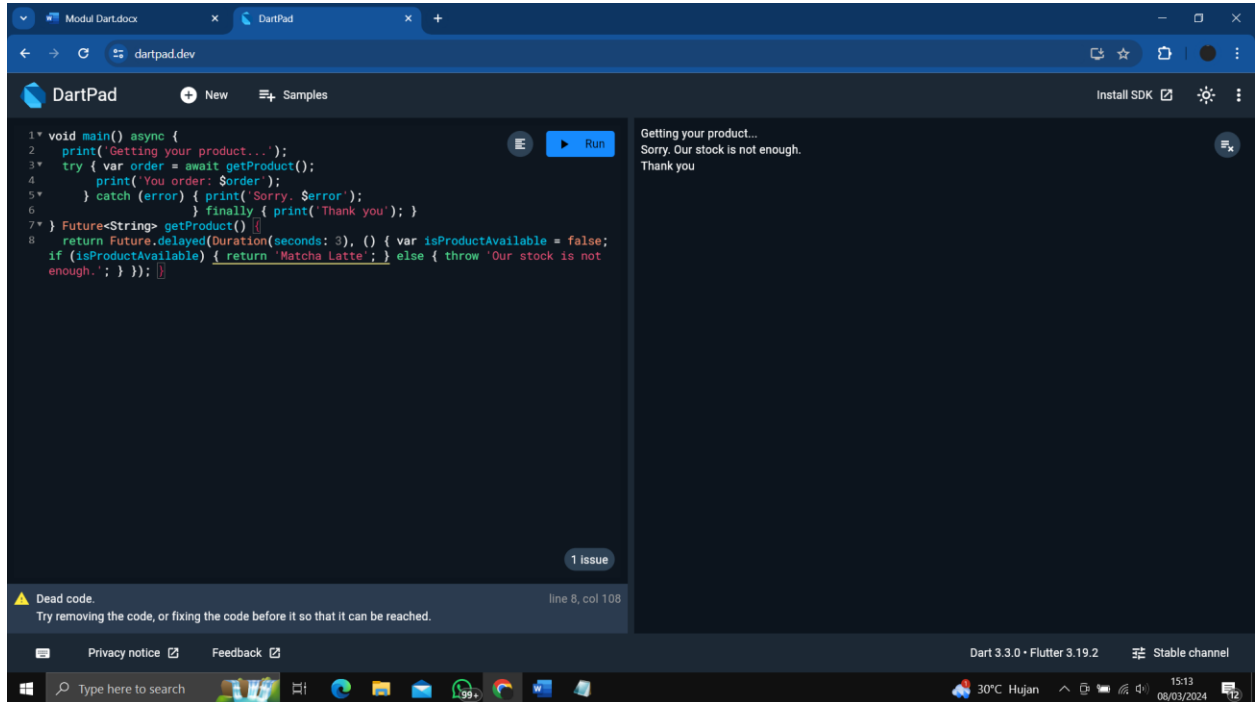
Getting your product...
Sorry. Our stock is not enough.

1 issue

Dead code.
Try removing the code, or fixing the code before it so that it can be reached. line 11, col 108

Konsep complete with data merupakan konsep exception pada java. Ketika sebuah nilai inputan salah, maka program akan menangkap kesalahan tersebut lalu akan menampilkan pesan kesalahan.

22. Complete with async-await



Konsep future dengan sync await hampir mirip sekali dengan complete with data, perbedaanya adalah program ini akan mencoba statementnya terlebih dahulu baru kemudian akan menangkap eror dan menampilkan pesan eror apabila terjadi kesalahan.