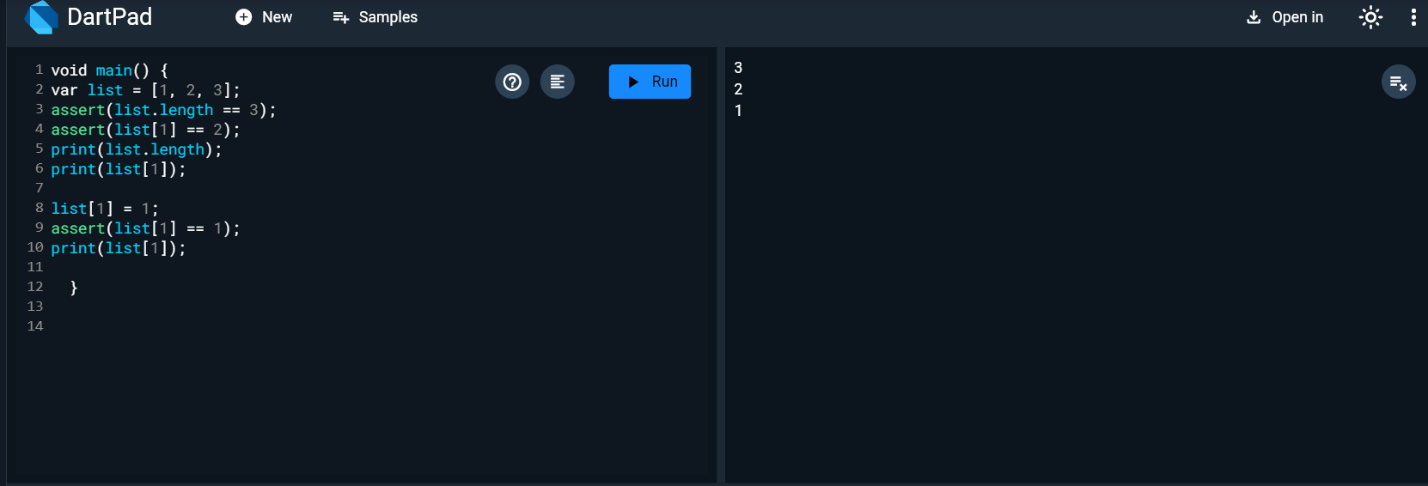
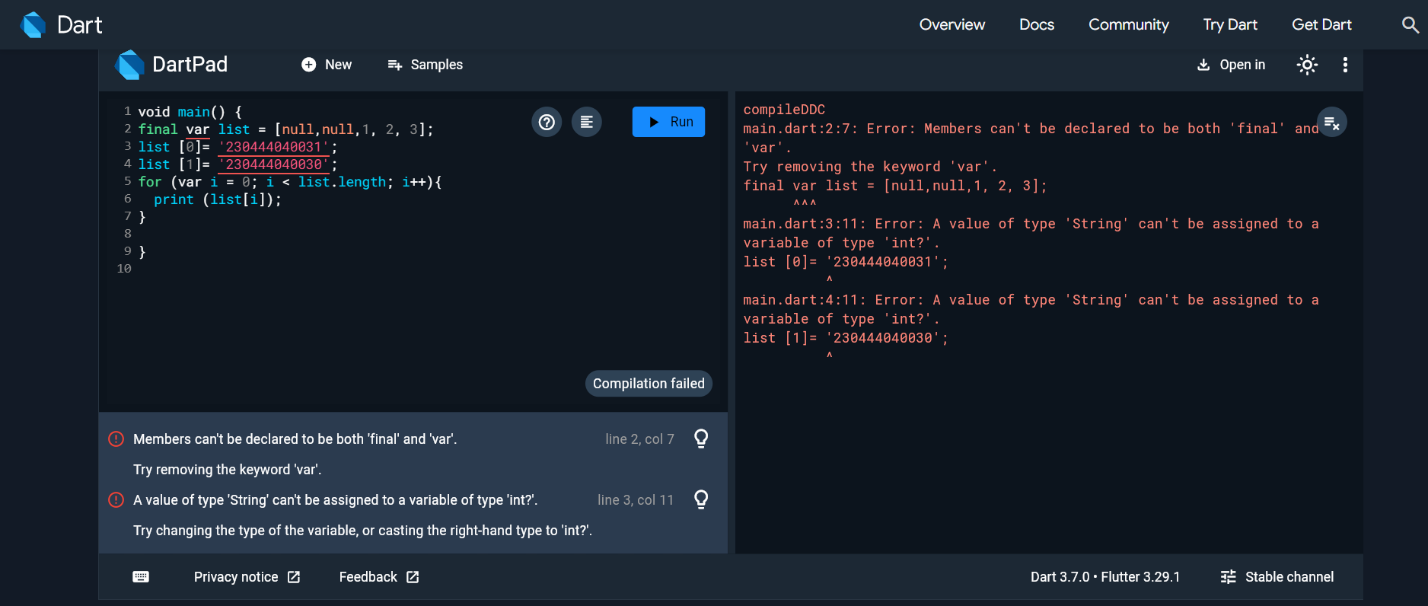
JAWABAN Tugas Praktikum 3

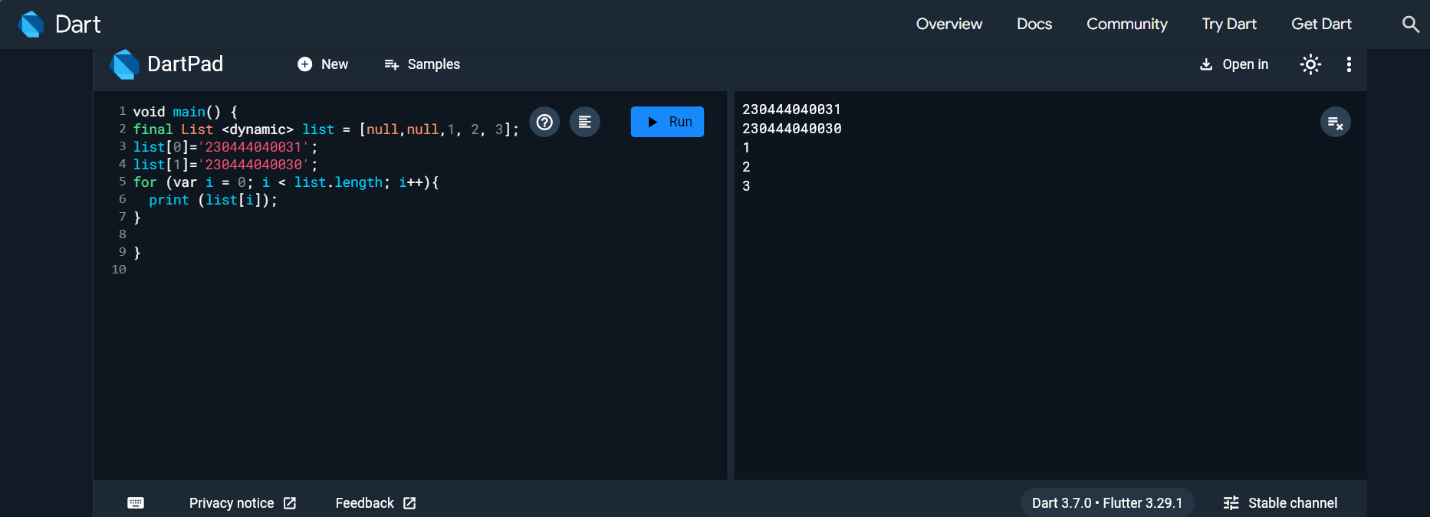
1. Praktikum

Praktikum 1

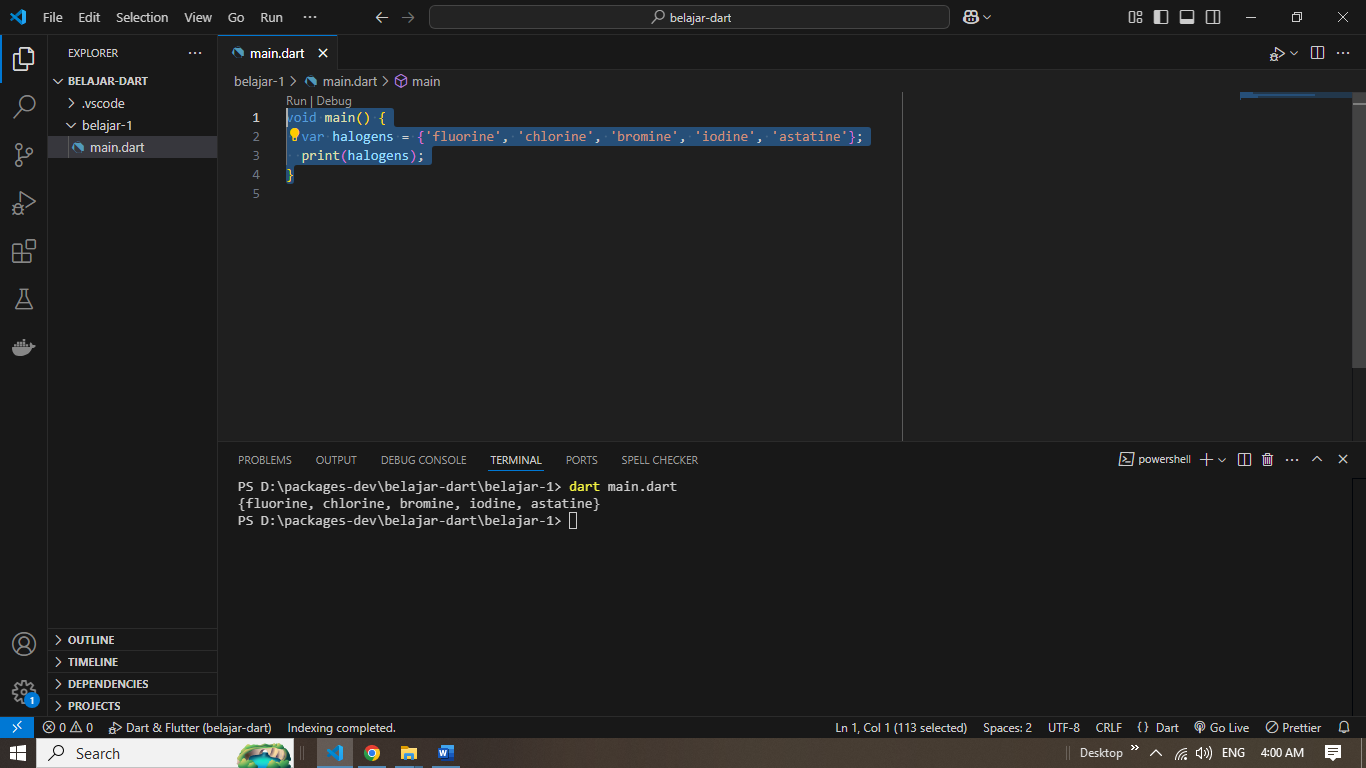


Mencetak angka 3,2,1 karena program tersebut mencetak Panjang dari list pada baris ke 5 kemudian mencetak list pada index ke 1 lalu pada baris 10 mencetak list index ke 1 karna pada baris ke 6 list index ke 1 diganti dengan angka 1 sehingga yang tercetak angka 3,2,1.

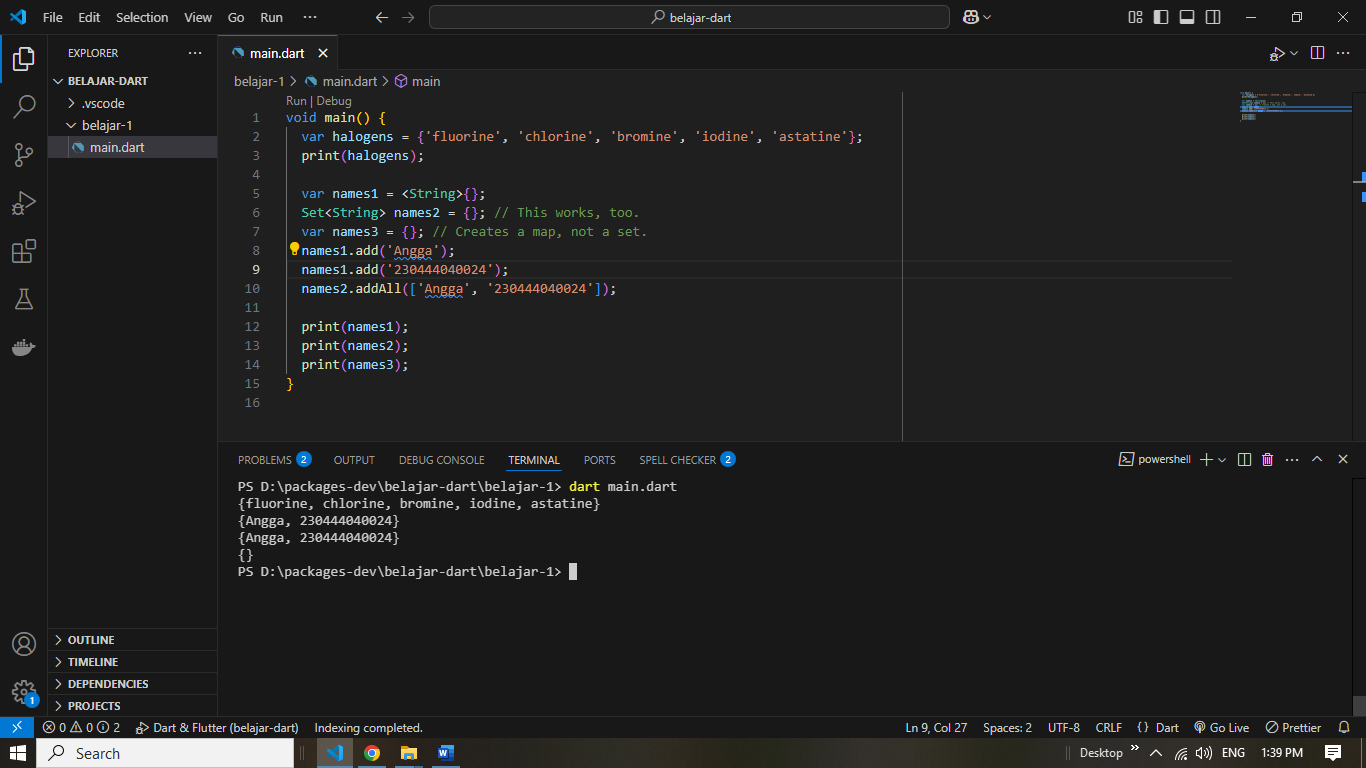


Setelah dilakukan percobaan hasilnya error karena pertama var tidak bisa di gabung dengan final lalu deklarasi var tidak bisa menyimpan list / array dengan tipe data campuran sehingga harus diganti dengan List <dynamic> agar bisa menyimpan tipe campuran . Hasil yang di perbaiki :

Praktikum 2

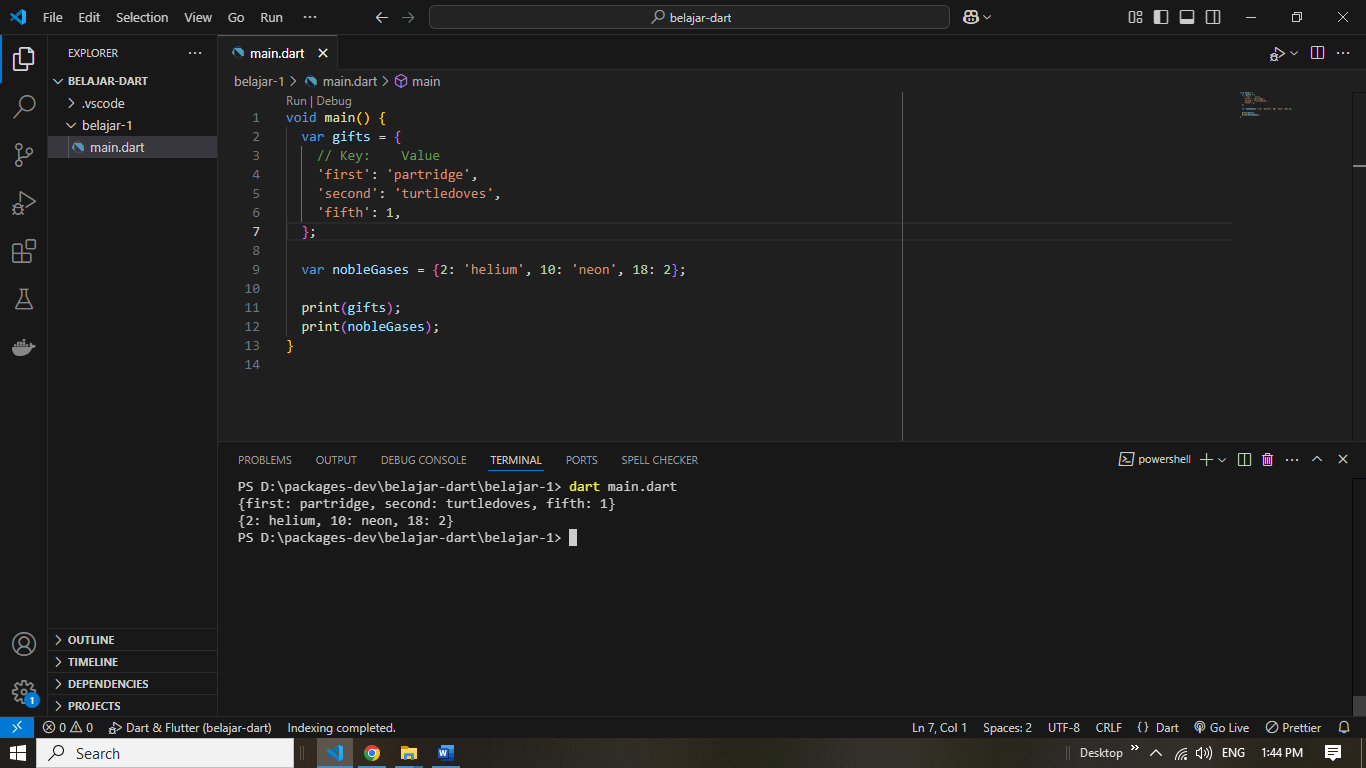


Yang dicetak dilayar adalah isi Set dari variable halogens yang mana Set itu adalah Kumpulan data yang tidak terurut dan tidak bisa ada duplikat data, hal ini berbeda dengan prinsip List yang merupakan Kumpulan data terurut memiliki index dan bisa terdapat duplikat.

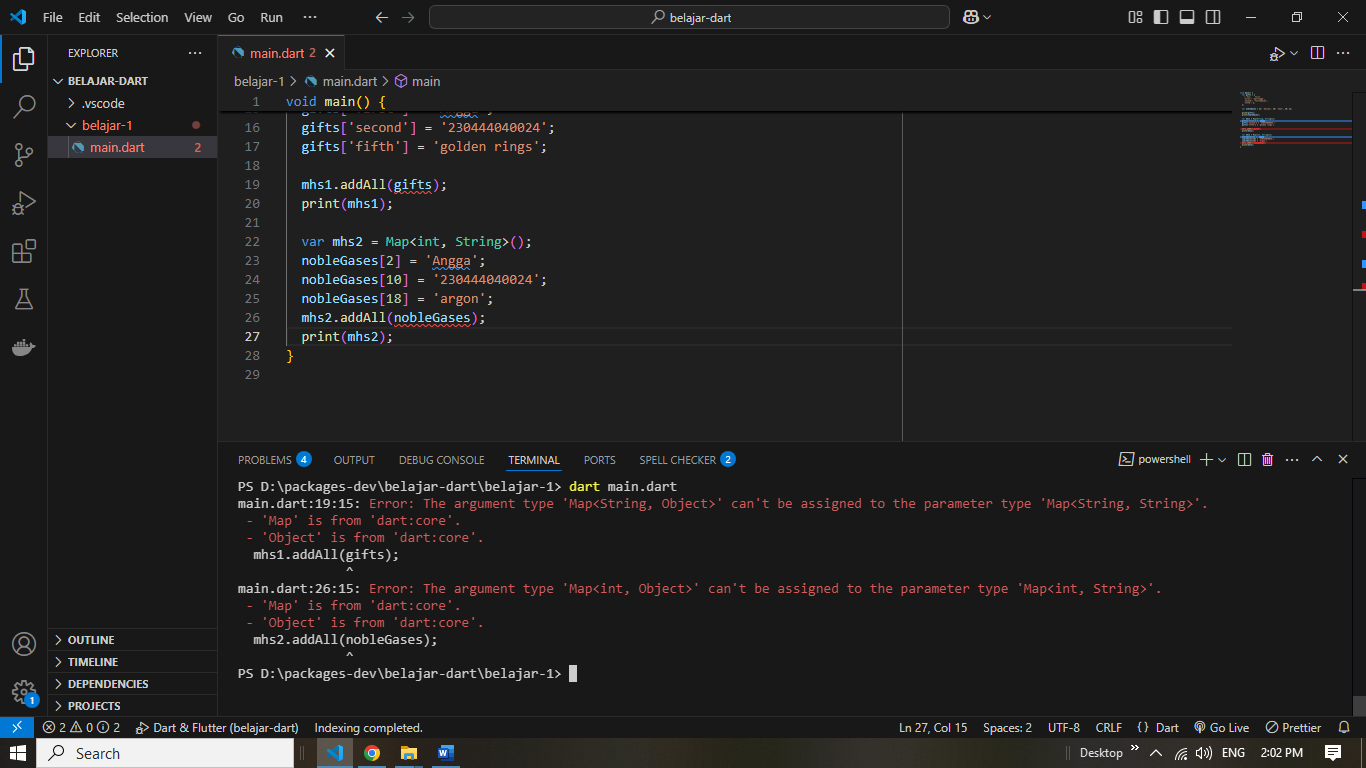


Yang terjadi adalah variable names1 ditambahkan 2kali nama lalu kemudian Nim dan variable names2 ditambahkan sekaligus dua data nama dan nim kemudian setnya dicetak ke layer.

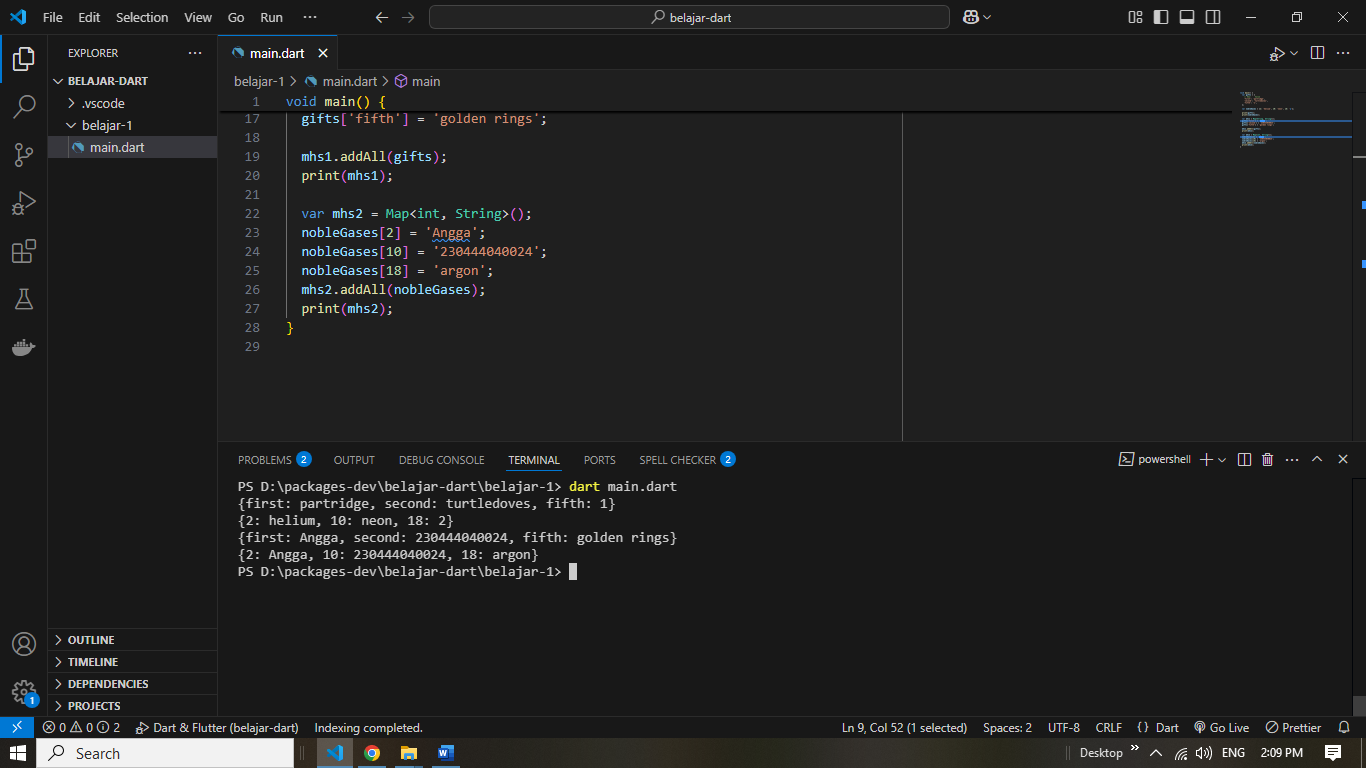
Praktikum 3



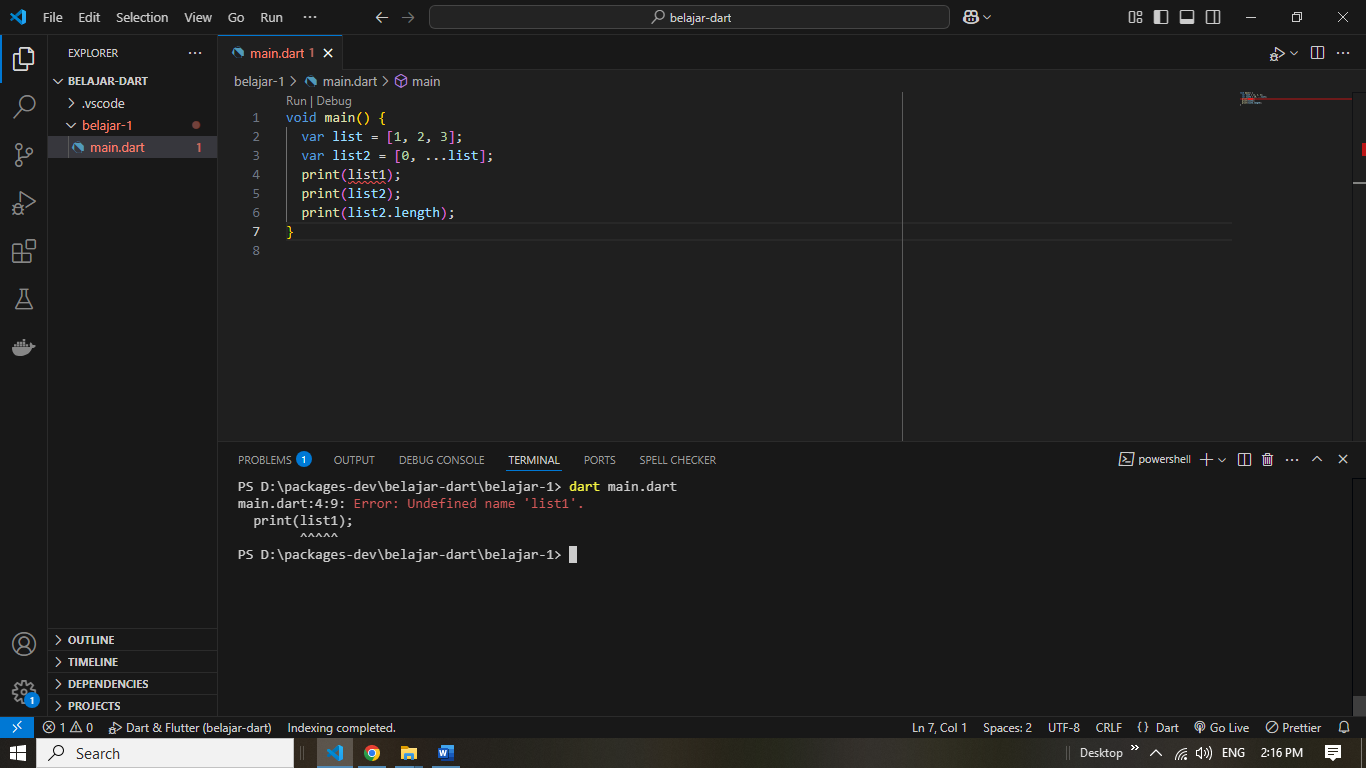
Yang terjadi adalah sebuah map tercetak di console map merupakan Kumpulan data yang memiliki pasangan key dan value mirip dengan set hanya saja tiap value memiliki key.



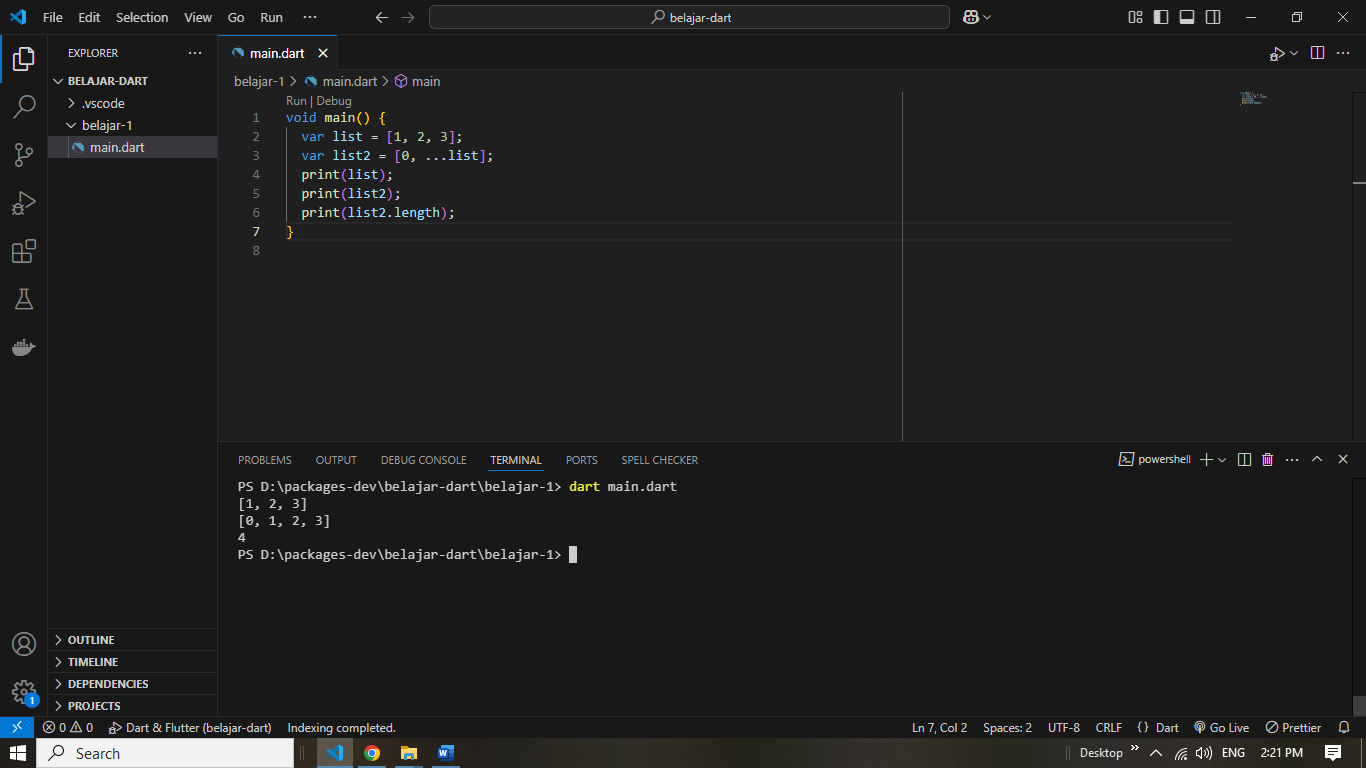
Setelah coba di tambahkan hasilnya error karna variable gifts dan nobleGases sebelumnya terisi value ccampuran maka keduanya dianggap Map<String, Object> untuk gifts dan Map <Int, Object> untuk nobleGases, hal ini tentu tidak cocok jika ingin memasukan kedua variable ke variable mhs1 dan mhs2 karna keduanya bertipe Map<String, String>. Saya memperbaiki dengan mengganti semua value default var gifts dan nobleGases ke tipe string semua agar sesuai :

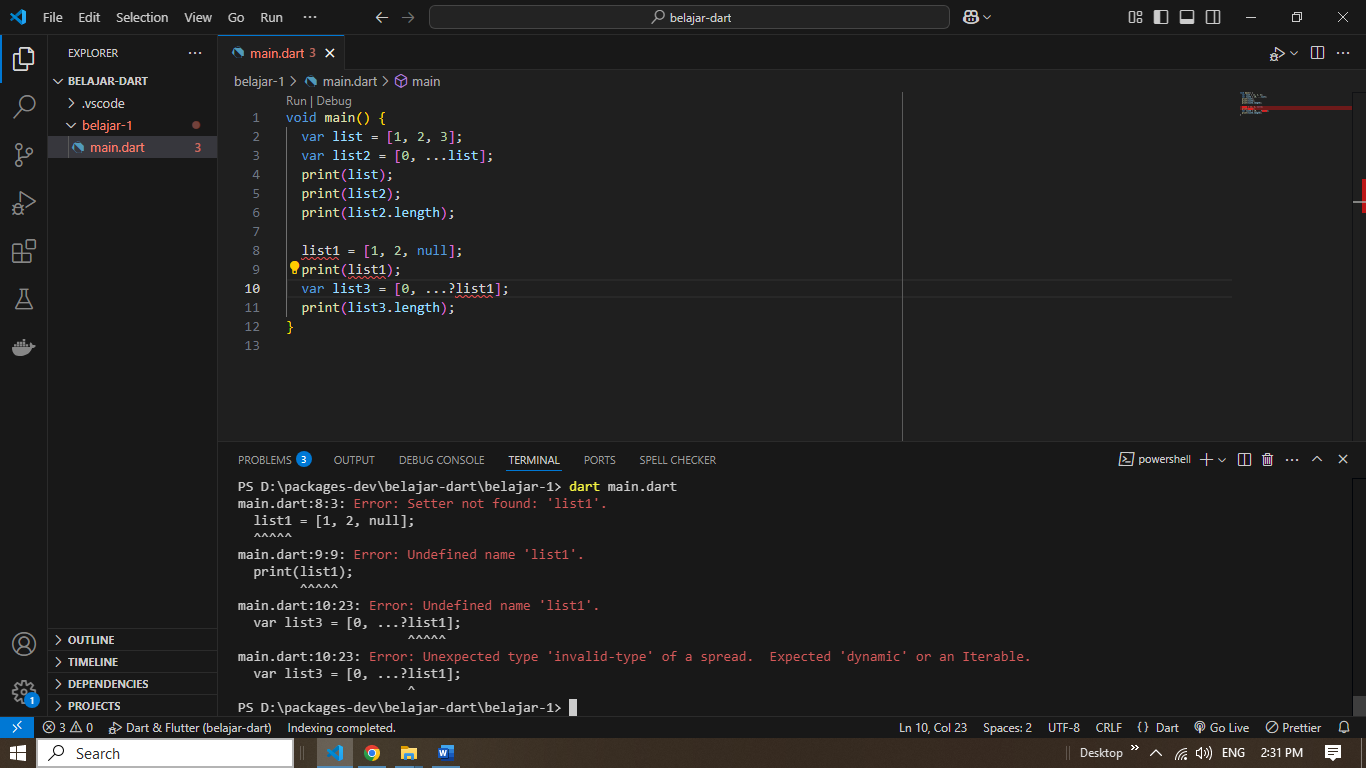


Praktikum 4

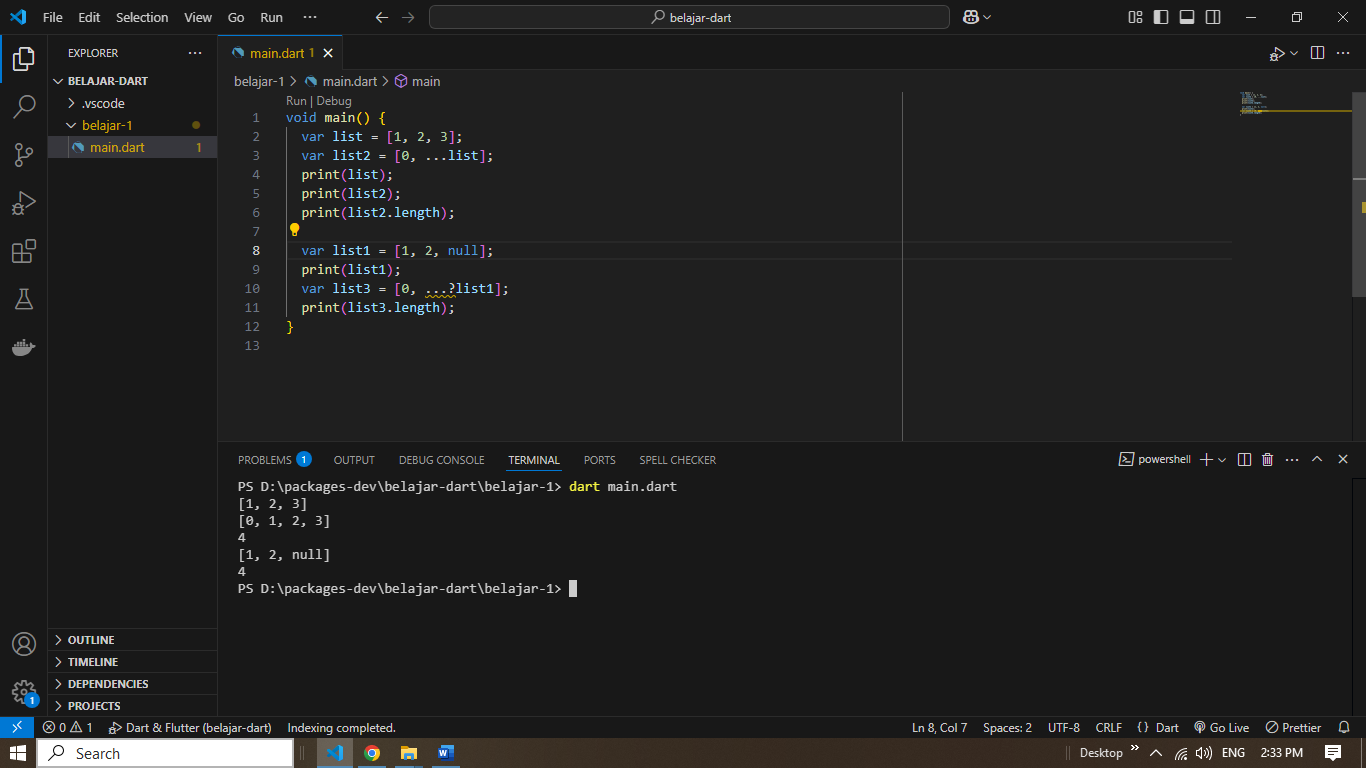


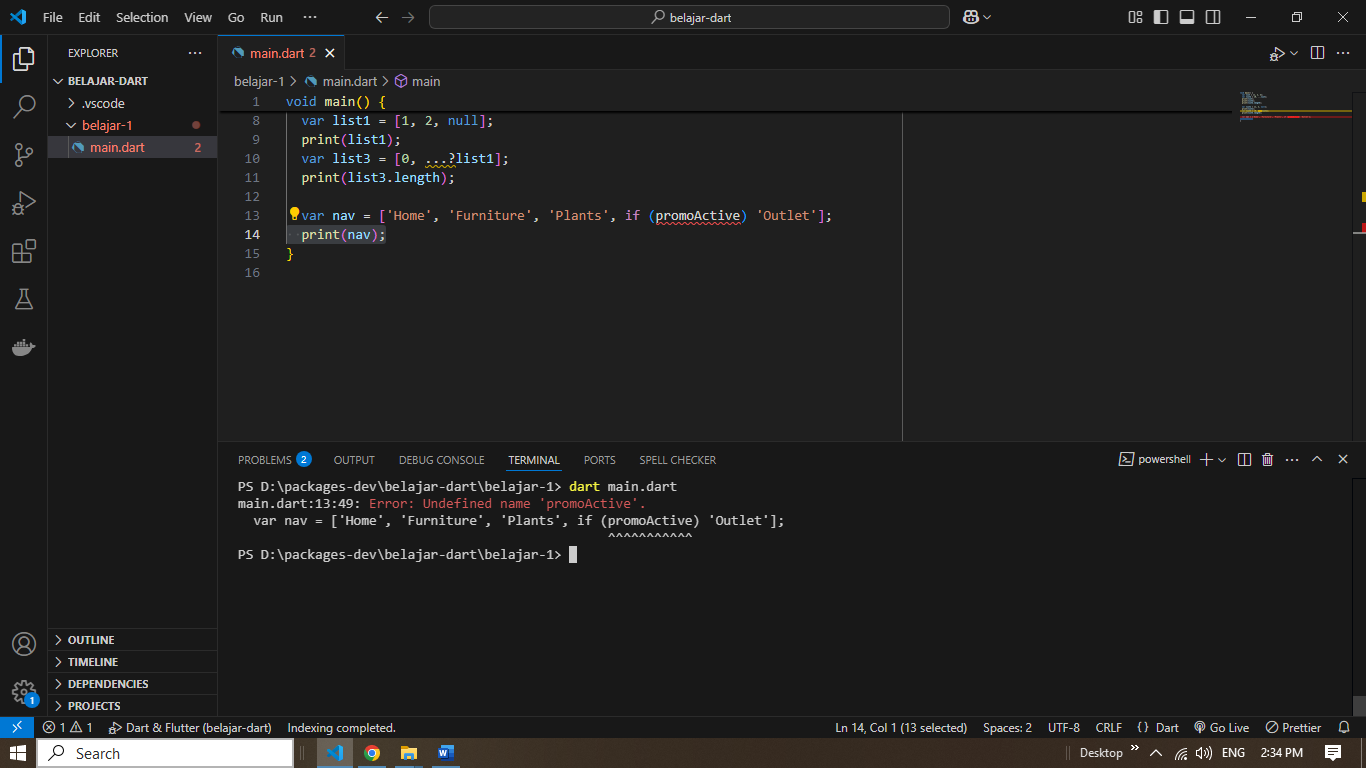
Yang terjadi adalah error karna variable list1 belum dideklarasikan selain itu pada deklarasi list2 disi mulai dari angka 0 kemudian diikuti spread operator … kepada vriable list yang mana isi dari variable list akan terisi di list2. Kode yang diperbaiki :



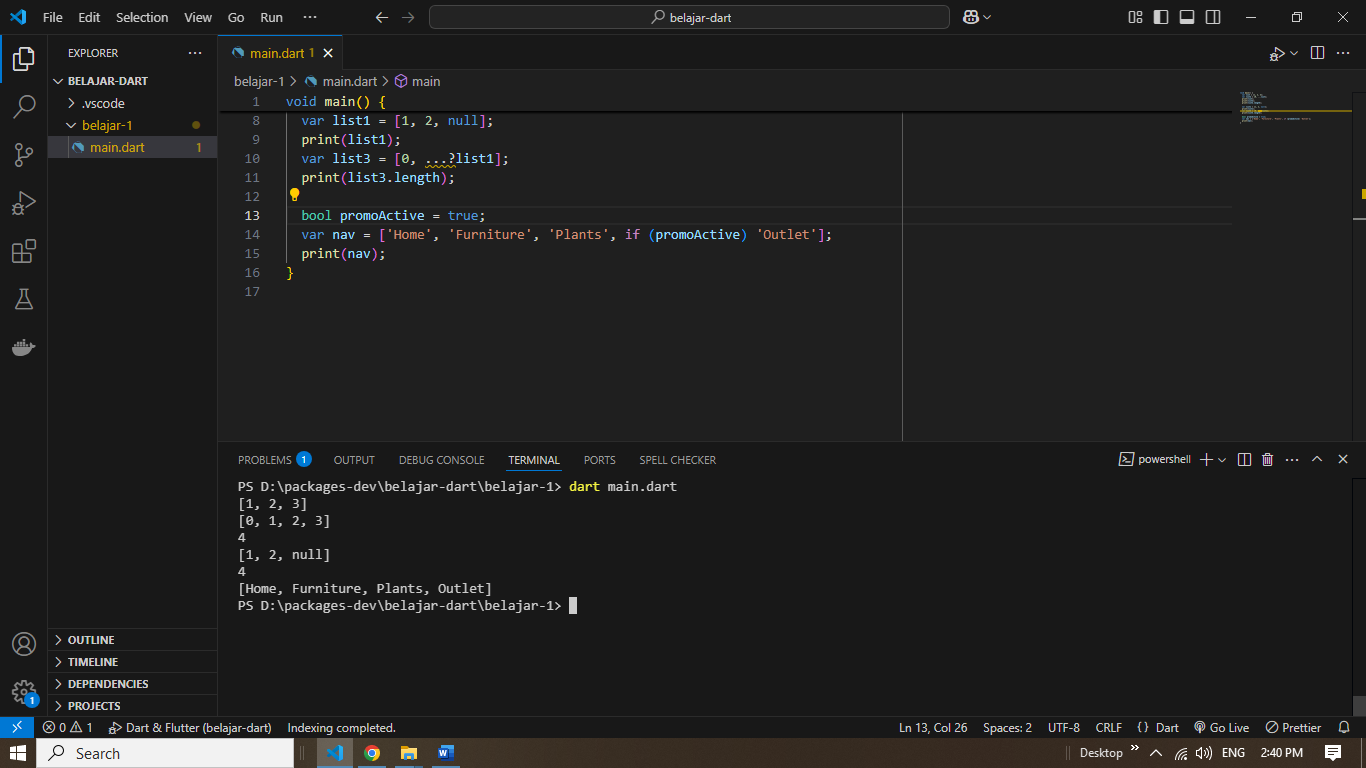


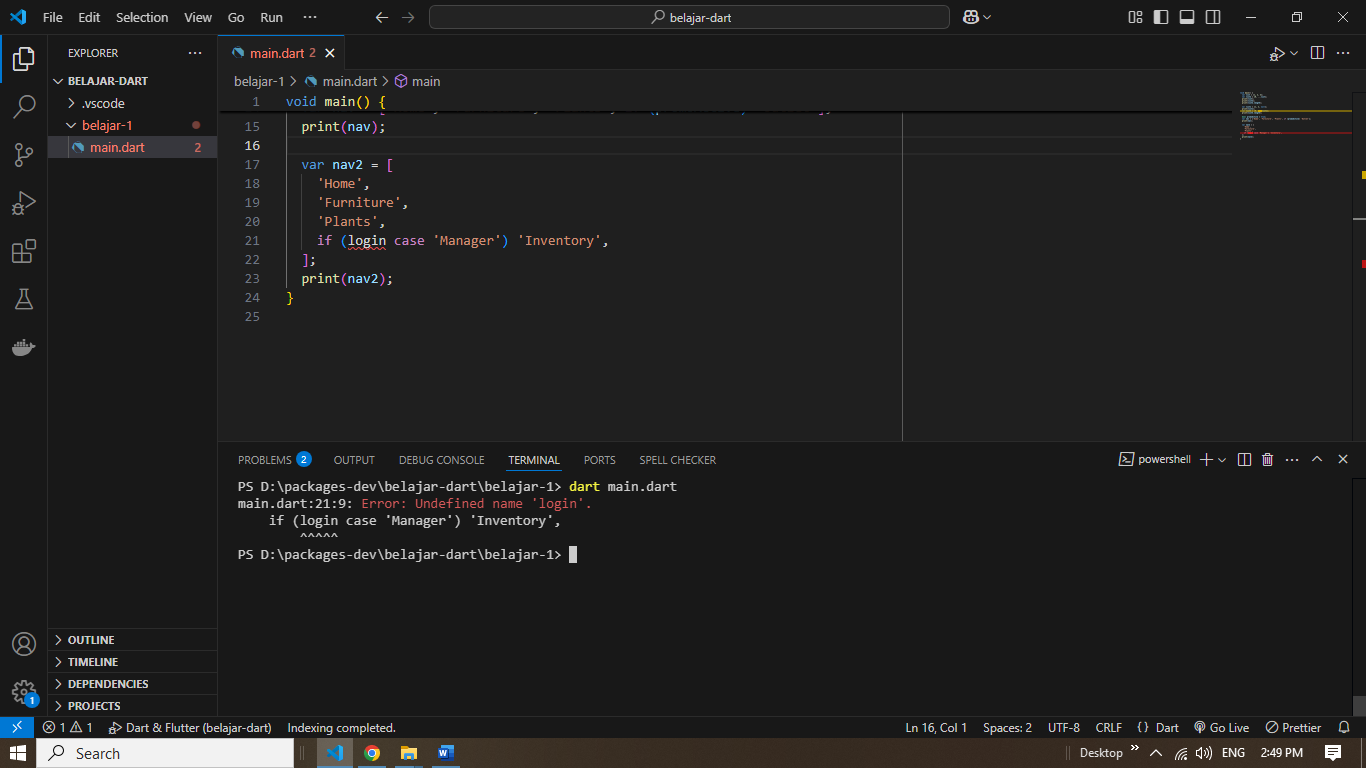
Yang terjadi error ini karna deklarasi list1 tidak valid kurang menambahkan setter. Kode yang di perbaiki :



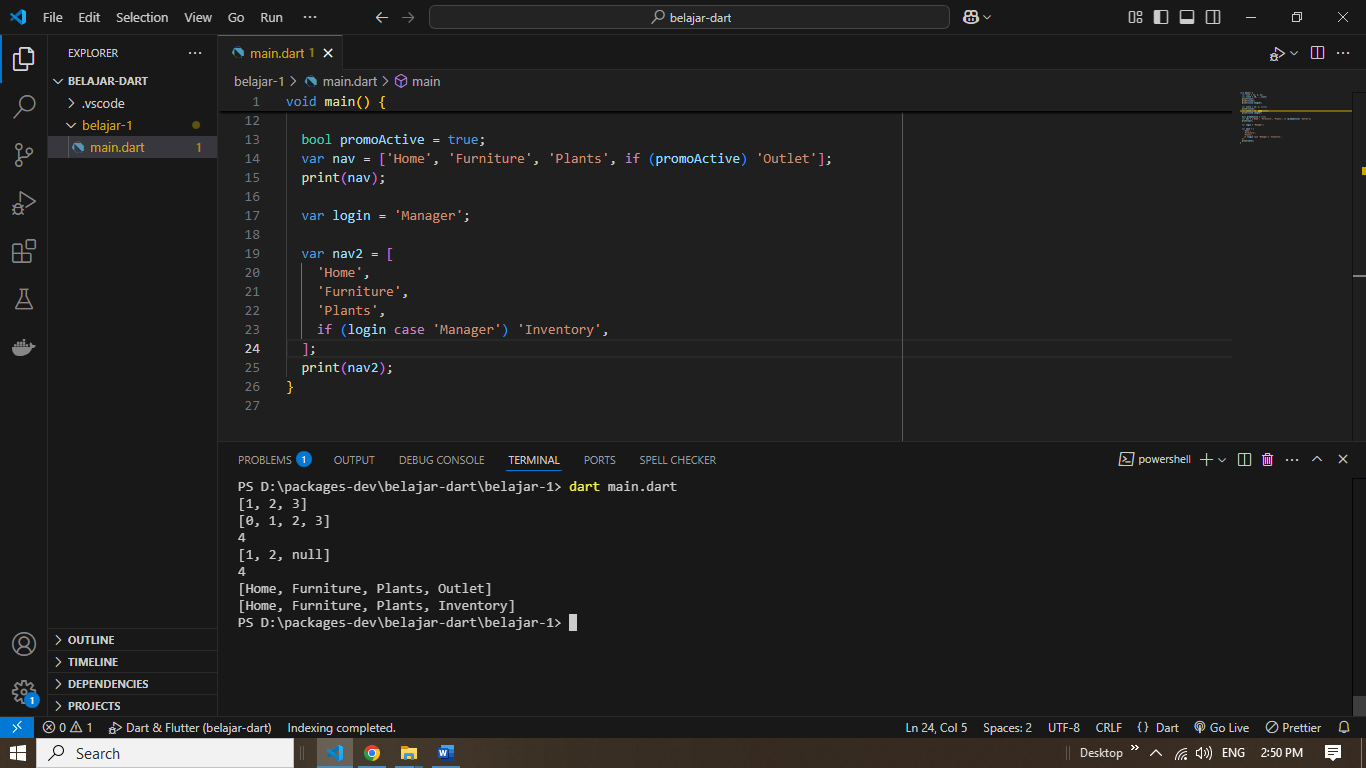


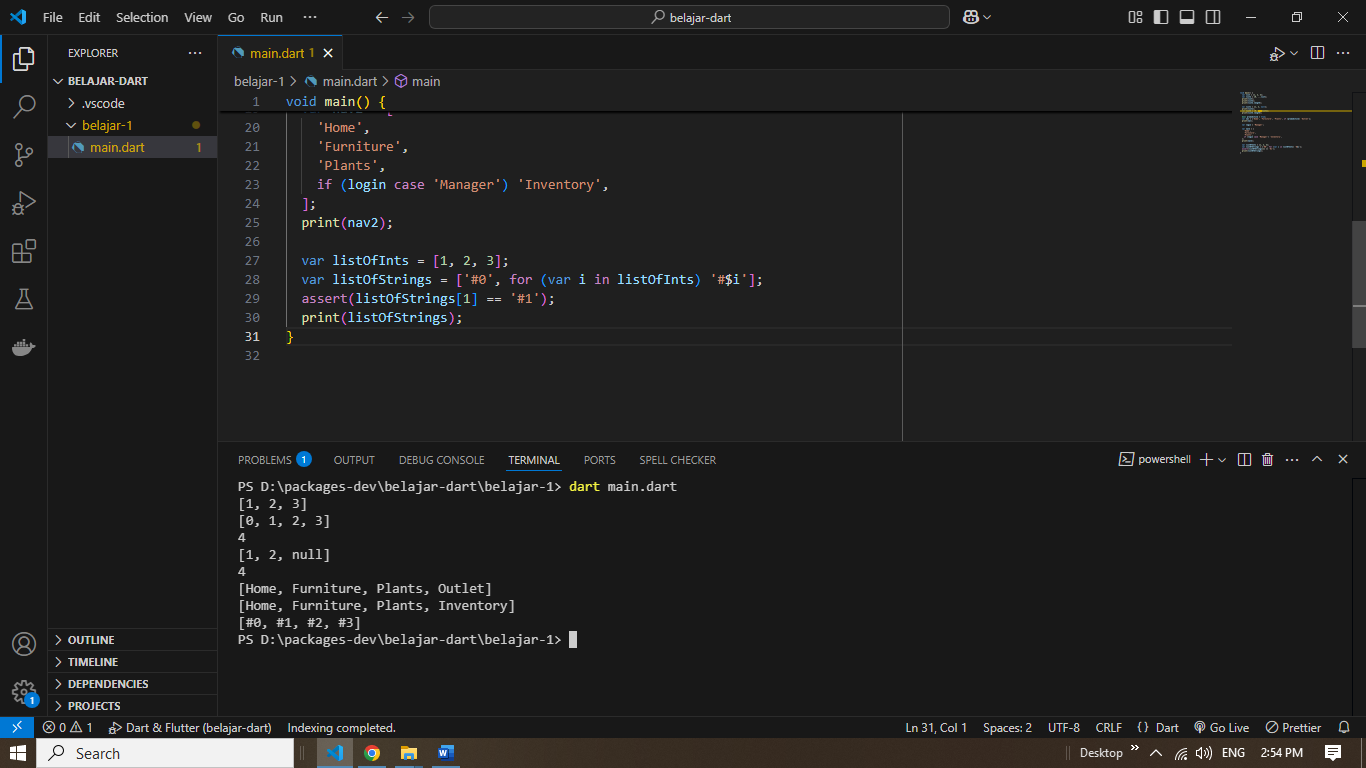
Error terjadi karna promoActive tidak di ketahui maksudnya. Memperbaiki dengan mendeklasasikan bool promoActive, dan jika true outlet akan di tambah ke var nav jika false tidak ditambahkan outlet :





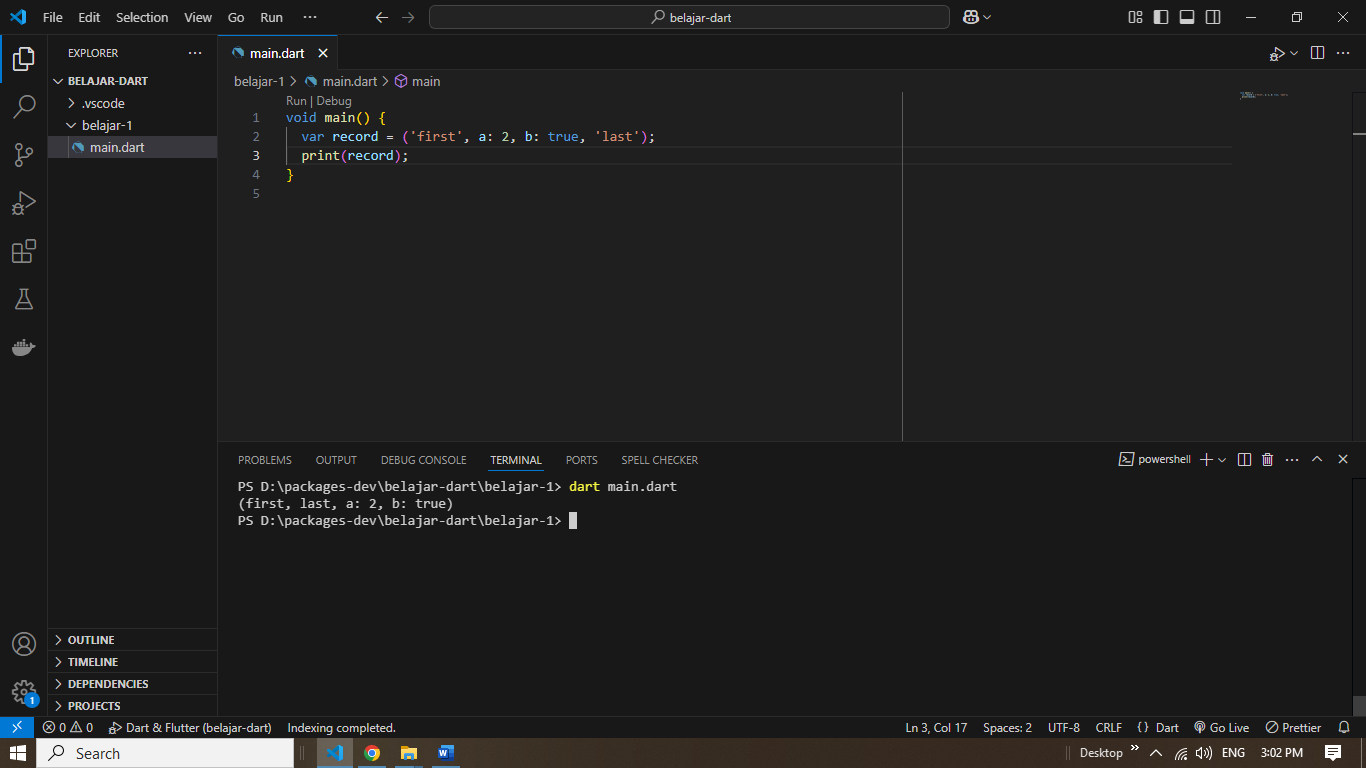
Error karna login belum didefinisikan kode yang di perbaiki :



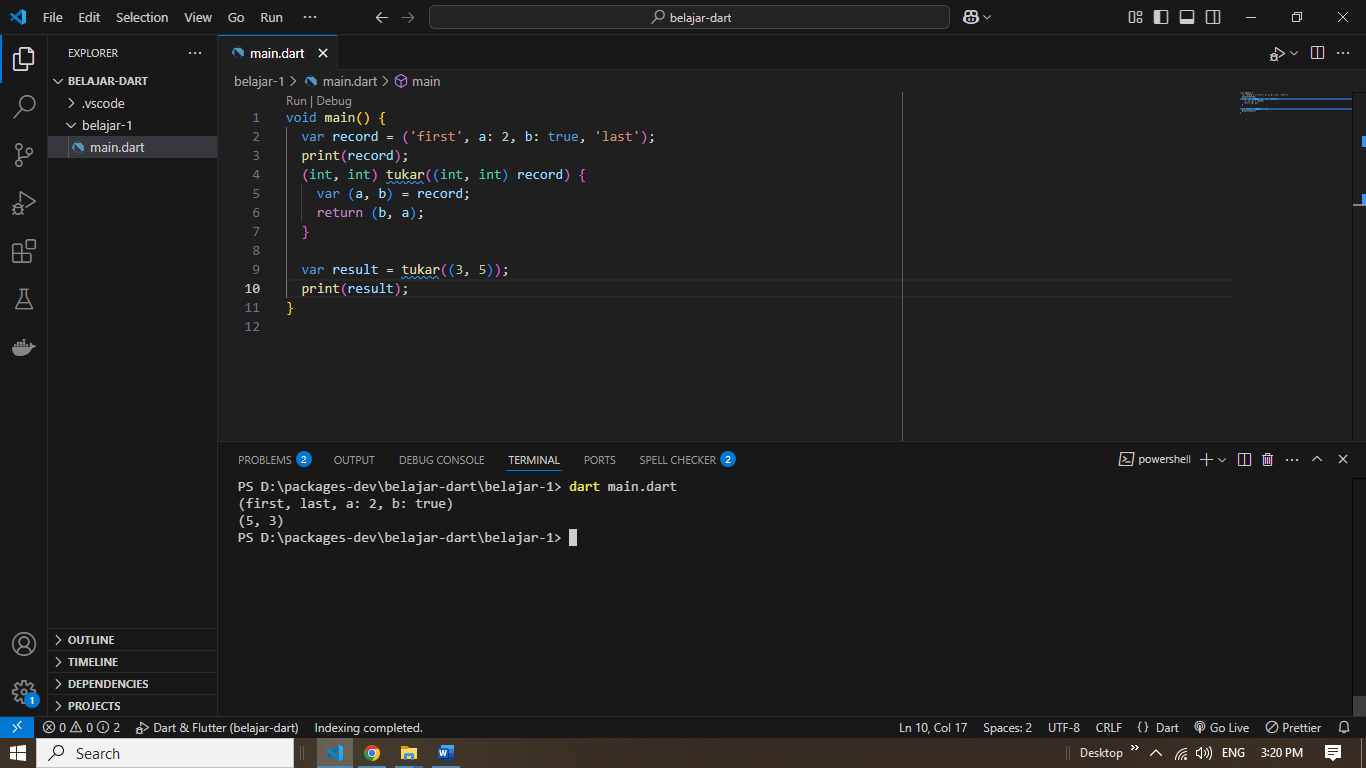


Yang terjadi mencetak [#0, #1, #2, #3] ke console hal ini terjadi loop collection for / for in saat deklarasi var listOfString memasukan 1 per 1 sari seetiap element dalam listOfInts yang mana setiap 1 kali iterasi value dalam listOfInts masuk kedalam var i .

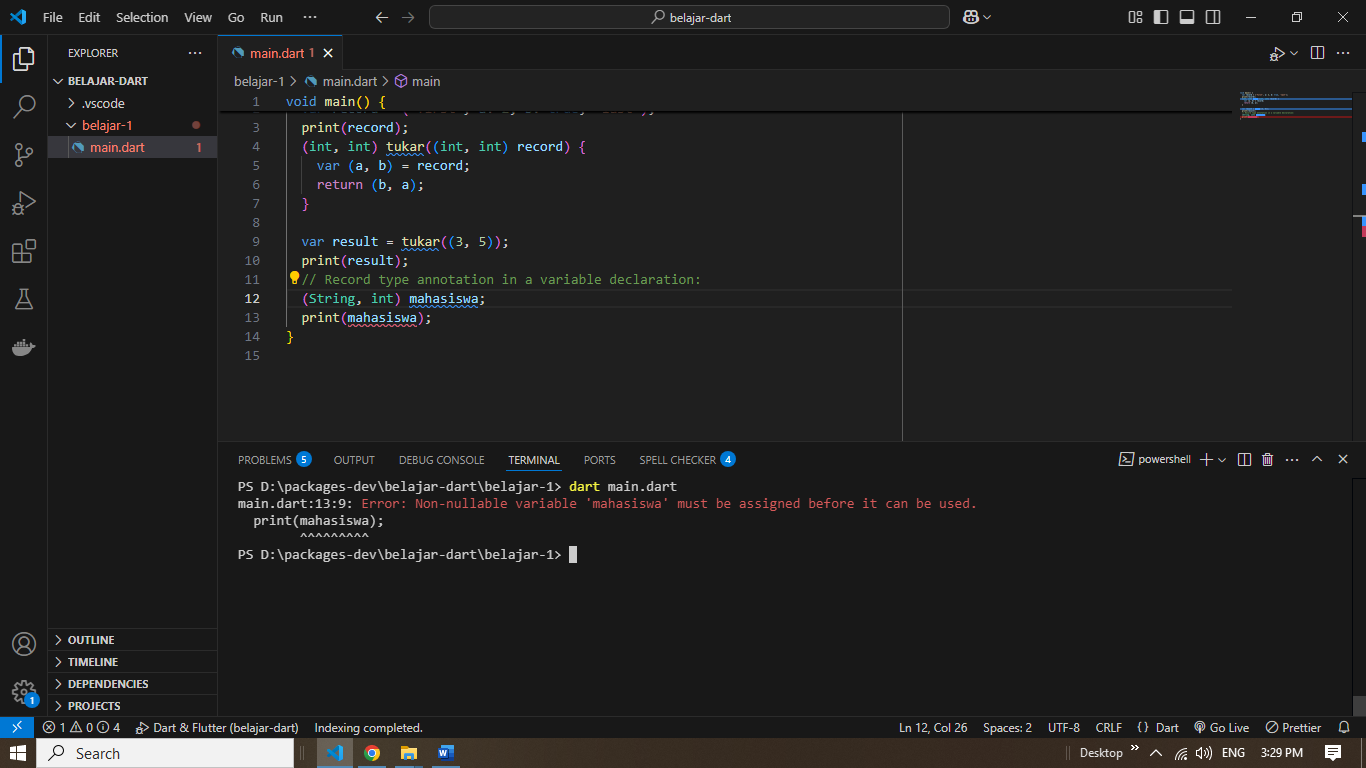
Praktikum 6



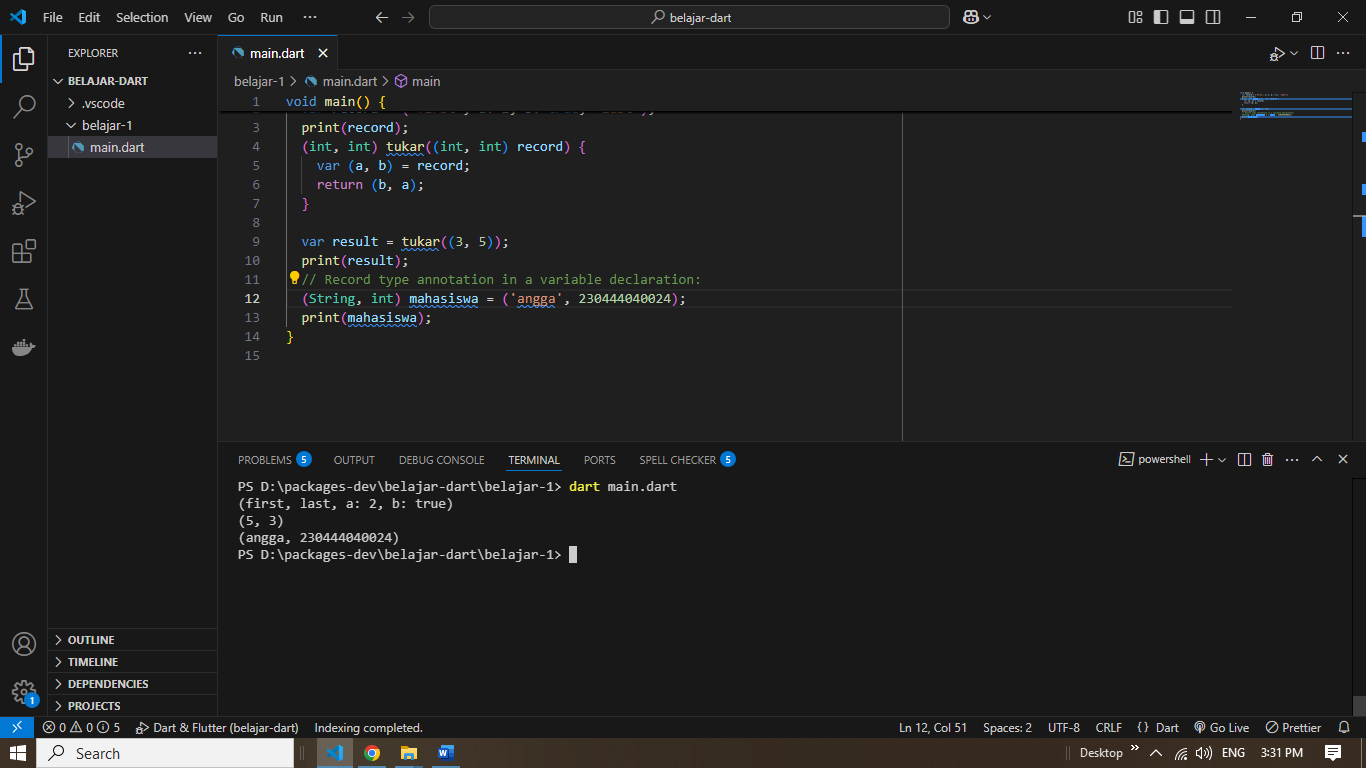
Mencetak isi variable record kelayar yang berisi tuple Kumpulan data dari berbagai jenis nilai urutannya berubah saat di cetak karna dart mendahulukan positional fields lalu named fields setelahnya.

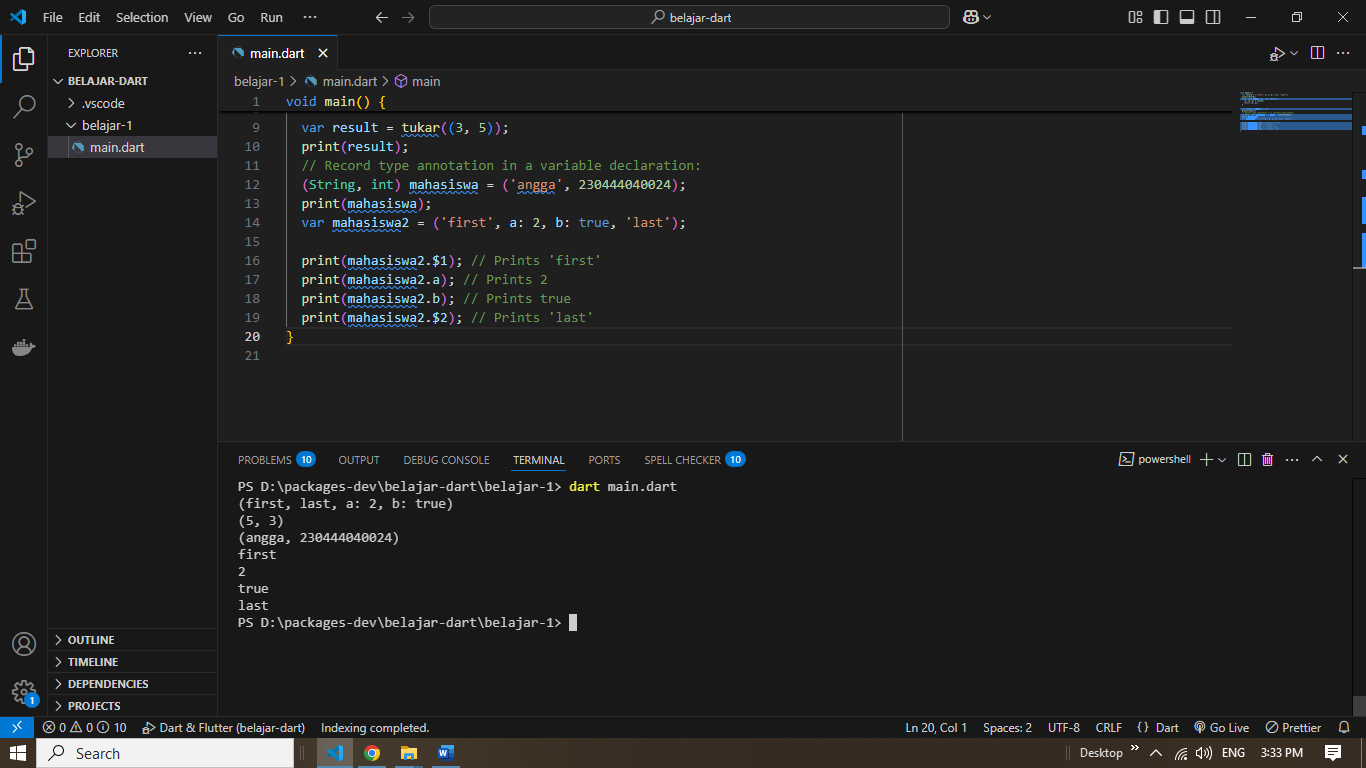


Saya menambahkan kode untuk memanggil fuhsi tukar nya dan yang terjadi adalah urutan angka di tukar dari kode pada parameter saat fungsi di panggil.

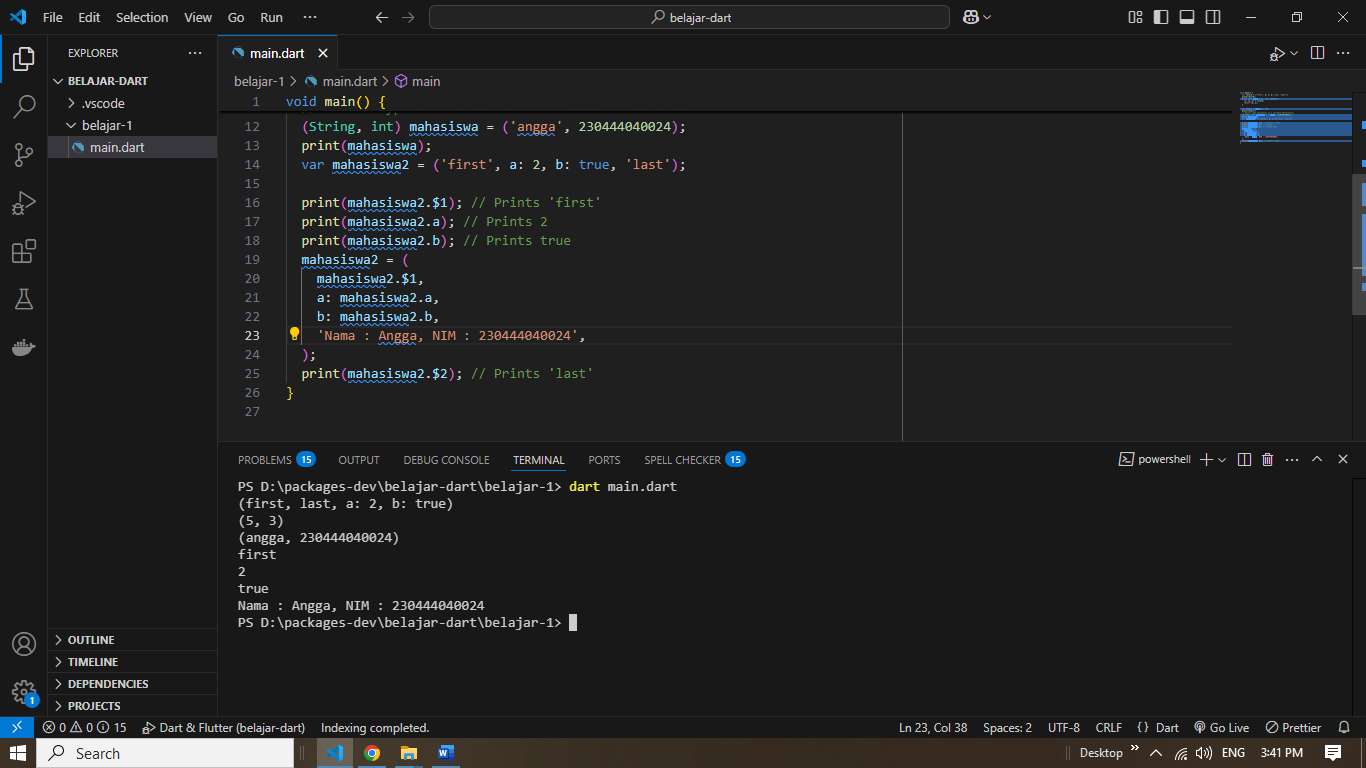


Error terjadi karna var mahasiswa belum di deklarasikan valuenya sesuai type recordnya. Kode yang di perbaiki :





Yang terjadi mencetak isi mahasiswa 2 keconsole 1 per satu sesuai uruta print(). Kode yang di ubah :



Untuk mengubahnya kita tidak bisa mengubah element dalam field Ketika sudah dibuat sehingga saya membuat record baru karna recordnya bersifat immutable.

1. Dalam **Dart**, **function (fungsi)** adalah blok kode yang dapat dipanggil untuk menjalankan tugas tertentu. Fungsi membantu **mengorganisir kode**, **menghindari duplikasi**, dan **mempermudah pemeliharaan**.
2. Fungsi Tanpa Parameter & Tanpa Return

void sapa() {

print("Halo, selamat datang di Dart!");

}

void main() {

sapa(); // Memanggil fungsi

}

1. Fungsi dengan Parameter

void sapa(String nama) {

print("Halo, $nama!");

}

void main() {

sapa("Budi"); // Output: Halo, Budi!

sapa("Siti"); // Output: Halo, Siti!

}

1. Fungsi dengan Return Value

int tambah(int a, int b) {

return a + b;

}

void main() {

int hasil = tambah(5, 3);

print("Hasil penjumlahan: $hasil");

}

1. Fungsi dengan Parameter Opsional

void sapa(String nama, [String? ucapan]) {

if (ucapan != null) {

print("$ucapan, $nama!");

} else {

print("Halo, $nama!");

}

}

void main() {

sapa("Budi", "Selamat pagi"); // Output: Selamat pagi, Budi!

sapa("Siti"); // Output: Halo, Siti!

}

1. Fungsi dengan Default Parameter

void sapa(String nama, {String ucapan = "Halo"}) {

print("$ucapan, $nama!");

}

void main() {

sapa("Budi"); // Output: Halo, Budi!

sapa("Siti", ucapan: "Selamat malam"); // Output: Selamat malam, Siti!

}

1. Function Expression (Arrow Function)

int kali(int a, int b) => a \* b;

void main() {

print(kali(4, 5)); // Output: 20

}

1. Anonymous Function (Lambda)

void main() {

var list = [1, 2, 3];

list.forEach((angka) {

print("Angka: $angka");

});

}

KESIMPULAN

|  |  |
| --- | --- |
| Jenis Function | Contoh |
| Tanpa Parameter | void sapa() { print("Halo!"); } |
| Dengan Parameter | void sapa(String nama) { print("Halo, $nama!"); } |
| Dengan Return Value | int tambah(int a, int b) { return a + b; } |
| Parameter Opsional | void sapa(String nama, [String? ucapan]) |
| Default Parameter | void sapa(String nama, {String ucapan = "Halo"}) |
| Arrow Function | int kali(int a, int b) => a \* b; |
| Anonymous Function | list.forEach((angka) { print(angka); }); |

1. Dalam **Dart**, ada beberapa jenis parameter yang bisa digunakan dalam function. Jenis-jenis parameter ini memungkinkan fleksibilitas saat memanggil fungsi. Berikut adalah penjelasannya beserta contoh sintaksnya.
2. Positional Parameter (Parameter Biasa)

Positional parameter adalah parameter yang wajib diisi saat memanggil fungsi. Urutannya harus sesuai dengan deklarasi parameter.

void sapa(String nama, int umur) {

print("Halo, nama saya $nama dan saya berumur $umur tahun.");

}

void main() {

sapa("Budi", 25); // ✅ Output: Halo, nama saya Budi dan saya berumur 25 tahun.

// sapa("Siti"); // ❌ ERROR: Parameter kurang

}

1. Optional Positional Parameter (Parameter Opsional)

Positional parameter bisa dibuat **opsional** dengan tanda **[]**. Jika tidak diisi, nilainya akan **null** atau bisa diberikan nilai default.

void sapa(String nama, [int? umur]) {

if (umur != null) {

print("Halo, nama saya $nama dan saya berumur $umur tahun.");

} else {

print("Halo, nama saya $nama.");

}

}

void main() {

sapa("Budi", 25); // ✅ Output: Halo, nama saya Budi dan saya berumur 25 tahun.

sapa("Siti"); // ✅ Output: Halo, nama saya Siti.

}

1. Named Parameter (Parameter Bernama)

Named parameter menggunakan tanda **{}**, sehingga saat dipanggil, nama parameter harus disebutkan.

* Bisa opsional
* Bisa diberi default
* Bisa ditandai required agar wajib diisi

**Named Parameter Opsional**

void sapa({String? nama, int? umur}) {

print("Halo, nama saya ${nama ?? 'Anonim'} dan saya berumur ${umur ?? 'tidak diketahui'}.");

}

void main() {

sapa(nama: "Budi", umur: 25); // ✅ Output: Halo, nama saya Budi dan saya berumur 25 tahun.

sapa(nama: "Siti"); // ✅ Output: Halo, nama saya Siti dan saya berumur tidak diketahui.

sapa(); // ✅ Output: Halo, nama saya Anonim dan saya berumur tidak diketahui.

}

**Named Parameter dengan Nilai Default**

void sapa({String nama = "Anonim", int umur = 0}) {

print("Halo, nama saya $nama dan saya berumur $umur tahun.");

}

void main() {

sapa(nama: "Budi"); // ✅ Output: Halo, nama saya Budi dan saya berumur 0 tahun.

sapa(); // ✅ Output: Halo, nama saya Anonim dan saya berumur 0 tahun.

}

**Named Parameter Wajib (required)**

void sapa({required String nama, required int umur}) {

print("Halo, nama saya $nama dan saya berumur $umur tahun.");

}

void main() {

sapa(nama: "Budi", umur: 25); // ✅ Output: Halo, nama saya Budi dan saya berumur 25 tahun.

// sapa(nama: "Siti"); // ❌ ERROR: Parameter umur wajib diisi

}

**Parameter dengan Function sebagai Argumen**

void prosesAngka(int angka, Function(int) fungsi) {

print("Hasil proses: ${fungsi(angka)}");

}

void main() {

prosesAngka(5, (int n) => n \* 2); // ✅ Output: Hasil proses: 10

}

**KESIMPULAN :**

|  |  |  |  |
| --- | --- | --- | --- |
| Jenis Parameter | Cara Penulisan | Opsional? | Harus Berurutan? |
| Positional | void fn(int a, int b) | Tidak | Ya |
| Optional Positional | void fn(int a, [int? b]) | Ya | Ya |
| Named | void fn({int? a, int? b}) | Ya | Tidak |
| Named dgn Default | void fn({int a = 0, int b = 1}) | Ya | Tidak |
| Named Required | void fn({required int a, required int b}) | Tidak | Tidak |
| Funciton Parameter | void fn(int a, Function b) | Ya | Ya |

1. Functions sebagai First-Class Objects dalam Dart

Dalam Dart, fungsi dianggap sebagai **first-class objects**, artinya:  
✅ Bisa disimpan dalam variabel  
✅ Bisa dikirim sebagai parameter  
✅ Bisa dikembalikan dari fungsi lain

1. Fungsi Disimpan dalam Variabel Fungsi Disimpan dalam Variabel

void sapa() => print("Halo, Dart!");

void main() {

var fungsi = sapa; // Simpan fungsi dalam variabel

fungsi(); // ✅ Output: Halo, Dart!

}

1. Fungsi sebagai Parameter

void jalankanFungsi(Function fungsi) => fungsi();

void sapa() => print("Halo!");

void main() {

jalankanFungsi(sapa); // ✅ Output: Halo!

}

1. Fungsi sebagai Return Value

Function buatSapa(String nama) {

return () => print("Halo, $nama!");

}

void main() {

var sapaBudi = buatSapa("Budi");

sapaBudi(); // ✅ Output: Halo, Budi!

}

1. Fungsi dalam List atau Map

void tambah(int a, int b) => print(a + b);

void kurang(int a, int b) => print(a - b);

void main() {

var operasi = {'tambah': tambah, 'kurang': kurang};

operasi['tambah']!(10, 5); // ✅ Output: 15

}

**Kesimpulan:** Fungsi di Dart bisa diperlakukan seperti objek biasa, membuat kode lebih fleksibel.

1. Anonymous function (fungsi tanpa nama) adalah fungsi yang tidak memiliki nama dan biasanya digunakan sebagai callback atau fungsi sementara.
2. Contoh Anonymous Function dalam Variabel

void main() {

var sapa = () {

print("Halo, ini anonymous function!");

};

sapa(); // ✅ Output: Halo, ini anonymous function!

}

1. Anonymous Function sebagai Parameter

void jalankanFungsi(Function fungsi) {

fungsi();

}

void main() {

jalankanFungsi(() {

print("Fungsi tanpa nama dipanggil!");

});

}

1. Anonymous Function dalam List atau Map

void main() {

var operasi = {

'tambah': (int a, int b) => print(a + b),

'kurang': (int a, int b) => print(a - b),

};

operasi['tambah']!(10, 5); // ✅ Output: 15

}

1. Anonymous Function dalam forEach

void main() {

var list = [1, 2, 3];

list.forEach((angka) {

print("Angka: $angka");

});

}

**Kesimpulan**

✅ **Anonymous function** tidak memiliki nama.  
✅ Biasanya dipakai untuk **callback atau fungsi sekali pakai**.  
✅ Bisa disimpan dalam **variabel, list, map, atau dikirim sebagai parameter**.

🚀 **Membantu membuat kode lebih ringkas dan fleksibel!**

1. Perbedaan Lexical Scope dan Lexical Closures dalam Dart

|  |  |  |
| --- | --- | --- |
| Konsep | Penjelasan | Contoh |
| Lexical Scope | Cara **Dart menentukan variabel mana yang dapat diakses berdasarkan lokasi penulisan kode** (bukan dari mana fungsi dipanggil). | void main() { int x = 10; void tampil() { print(x); } tampil(); } |
| Lexical Closures | Fungsi yang **mengakses variabel di luar scope-nya dan mempertahankan nilainya** meskipun fungsi induknya sudah selesai dieksekusi. | Function counter() { int count = 0; return () => print(++count); } |

**CONTOH SINTAK LEXICAL SCOPE**

void main() {

int x = 10; // Variabel dalam scope `main`

void tampil() {

print(x); // ✅ Bisa mengakses x karena masih dalam lingkup `main`

}

tampil(); // ✅ Output: 10

// print(y); ❌ ERROR: y tidak dikenal di scope `main`

}

**CONTOH SINTAK LEXICAL CLOSURES**

Function buatCounter() {

int count = 0; // Variabel ini akan "diingat" oleh closure

return () {

count++;

print("Counter: $count");

};

}

void main() {

var hitung = buatCounter(); // `hitung` menyimpan closure

hitung(); // ✅ Output: Counter: 1

hitung(); // ✅ Output: Counter: 2

**}**

**KESIMPULAN**

|  |  |  |
| --- | --- | --- |
| **Perbedaan** | **Lexical Scope** | **Lexical Closure** |
| Definisi | Aturan akses variabel berdasarkan lokasi kode.Scope | Fungsi yang menyimpan keadaan variabel meskipun lingkup aslinya sudah selesai. |
| Kapan Variable bisa digunakan | Hanya dalam lingkup tempatnya dideklarasikan. | Tetap bisa diakses selama closure masih dipanggil. |
| Contoh penggunaan | Fungsi dalam fungsi, nested scope. | Counter, cache, fungsi dinamis. |

1. Dalam Dart, function **tidak bisa langsung mengembalikan lebih dari satu nilai** seperti di beberapa bahasa lain. Namun, ada beberapa cara untuk **mengembalikan multiple values**, yaitu menggunakan **List, Map, Tuple (Record), atau Class/Object**.

* Menggunakan List
* Menggunakan Map
* Menggunakan Tuple (record) – Dart 3+
* Menggunakan Class atau Object

✅ **Gunakan List atau Tuple untuk data sederhana**  
✅ **Gunakan Map atau Class jika butuh lebih banyak fleksibilitas**

🚀 **Pilihan terbaik tergantung pada kebutuhan kode!**