LAPORAN PRAKTIKUM

NIM	**:**	2003073
Nama	**:**	Ica Natasya
Kelas	**:**	D3TI.2C
Mata Kuliah	**:**	**Pemrograman Perangkat Bergerak (TIU3403)**
Praktikum ke / Judul	**:**	3/ Dasar Pemrograman Dart Lanjut
Tanggal Praktikum	**:**	24 Februari 2022
Dosen Pengampu	**:**	Fachrul Pralienka Bani Muhamad, S.ST., M.Kom

PROGRAM STUDI D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
POLITEKNIK NEGERI INDRAMAYU
2022

PRAKTIKUM 3
DASAR PEMROGRAMAN DART LANJUT

A. TUJUAN PRAKTIKUM

Tujuan Umum

Mahasiswa dapat memahami tingkatan lebih lanjut dari bahasa pemrograman Dart sehingga mahasiswa mampu mengikuti alur pembelajaran dengan lebih baik.

Tujuan Khusus

Mahasiswa dapat

1. Membuat percabangan dengan bahasa pemrograman Dart
2. Membuat perulangan dengan bahasa pemrograman Dart
3. Membuat *method* dengan bahasa pemrograman Dart
4. Melakukan pengujian unit pada bahasa pemrograman Dart

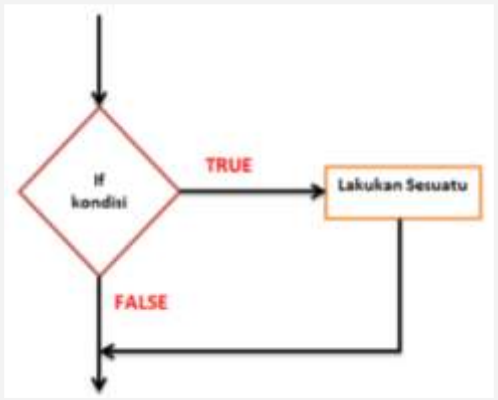
B. RANGKUMAN TEORI SINGKAT

Pada modul Dart bagian sebelumnya, Anda sudah mengetahui tipe-tipe data dalam bahasa Dart serta mampu membuat variabel dan konstantanya. Memahami dasar pemrograman Dart sangatlah penting agar dapat meningkatkan kemampuan berpikir secara logis, mengembangkan cara berpikir dengan sistematis dan melatih ketelitian terhadap detail saat membuat aplikasi menggunakan bahasa Dart. Beberapa dasar yang perlu dipahami dari bahasa Dart meliputi percabangan, perulangan, method, dan pengujian unit untuk membuat sebuah aplikasi menggunakan bahasa pemrograman Dart.

Percabangan

Percabangan adalah suatu pilihan atau opsi dimana terdapat kondisi tertentu yang harus dipenuhi oleh program untuk menjalankan perintah. Jika pilihan yang menjadi syarat terpenuhi, maka pilihan dijalankan. Sebaliknya, jika tidak maka program tidak akan menjalankan perintah atau melewatinya serta melihat kondisi lainnya untuk dijalankan atau berhenti sama sekali.
Percabangan juga dikenal dengan istilah "*Control Flow*", "*Struktur Kondisi*", "*Struktur IF*", "*Decision*", dsb. Semuanya itu pada dasarnya adalah sama. Struktur percabangan pada bahasa pemrograman Dart, sama seperti pada C++, C#, Java, dan Javascript. Ada empat macam bentuk percabangan pada bahasa pemrograman Dart meliputi percabangan `If`; percabangan `If..Else`; percabangan `If..Else If..Else`; dan percabangan `Switch..Case`.

1. **Percabangan If**
   Digunakan untuk percabangan tunggal maksudnya digunakan untuk menyeleksi logika yang menghasilkan nilai benar atau salah (*true* dan *false*). Jadi percabangan `if` hanya memiliki satu opsi atau pilihan, artinya hanya dikerjakan jika kondisinya benar, tapi jika pilihan dengan syarat salah maka tidak terjadi apa-apa. Dalam *flowchart* dijelaskan bahwa jika variabel kondisi yang nilainya *true* dengan syarat pilihan benar, maka proses dijalankan. Sebaliknya, jika tidak maka tidak terjadi apa-apa.



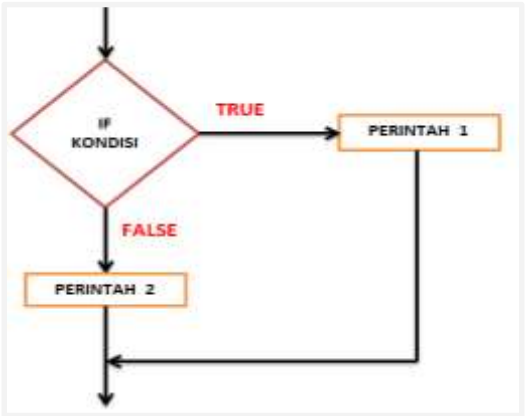
Gambar 1. Kondisi If

Format kondisi `if` sebagai berikut:

```
if(suatu kondisi) {
      // maka kerjakan ini
}
```

2. Percabangan If..Else

Digunakan bila terdapat 2 pilihan, maksudnya suatu perintah akan dijalankan apabila suatu kondisi terpenuhi dan sebaliknya jika kondisi tidak terpenuhi maka perintah lain yang akan dijalankan. Berdasarkan Gambar 2, jika kondisi variabel bernilai `true` dengan syarat pilihan benar maka perintah 1 dilakukan, jika salah maka perintah 2 yang dilakukan.



Gambar 2. Kondisi If..Else

Format kondisi `If.. Else` sebagai berikut :

```
if(suatu kondisi) {
      //maka kerjakan ini
} else {
      //kerjakan ini
}
```

3. Percabangan If..Else If..Else

Digunakan bila terdapat lebih dari 2 pilihan. Pemeriksaan kondisi dilakukan secara berurutan. Apabila suatu kondisi awal tidak terpenuhi, maka kondisi berikutnya yang akan diperiksa. Jika kondisi berikutnya tidak terpenuhi lagi, maka dilanjutkan hingga berakhir pada suatu kondisi akhir (`else`).

Format kondisi `If..Else If..Else` sebagai berikut:

```
if(suatu kondisi) {
      //maka kerjakan ini
} else if (kondisi lain) {
      //kerjakan ini
} else if (kondisi lain lagi) {
     //kerjakan ini
} else {
      //kerjakan ini kalau tidak ada yang memenuhi diatas
}
```

4. Percabangan Switch .. Case

Percabangan `SWITCH..CASE` sebenarnya adalah bentuk lain dari `IF..ELSE IF..ELSE`. Bedanya, percabangan ini menggunakan kata kunci `switch, case`, dan `default`. Formatnya juga berbeda, tapi cara kerjanya sama.

Format kondisi `Switch .. Case` sebagai berikut :

```
switch(variabel){
case 1:
      // kerjakan kode ini
      break;
case 2:
      // kerjakan kode ini
      break;
case 3:
      // kerjakan kode ini
      break;
default:
      // kerjakan kode ini
}
```

Perhatikan: `case` 1 artinya nilai variabel yang akan dibandingkan, apakah nilainya sama dengan 1 atau tidak. Kalau iya, maka kerjakan kode yang ada di dalam `case` 1.

Perlu diperhatikan juga: di sana ada kata kunci `break` dan `default`.

- ***break*** artinya berhenti. Ini untuk memerintahkan komputer untuk berhenti memeriksa *case* yang lainnya.

- ***default*** artinya jika nilai variabel tidak ada yang sama dengan pilihan `case` di atas, maka kerjakan kode yang ada di dalam `default`. Pilihan `default` bisa juga tidak memiliki `break`, karena dia adalah pilihan terakhir. Artinya pemeriksaan akan berakhir di situ.

Fitur menarik lain dari Dart adalah ***conditional expressions***. Dengan ini, kita bisa menuliskan struktur `if-else` statement hanya dalam satu baris:

```
// condition ? true expression : false expression
var shopStatus = now > openHours ? "Hello, we're open" : "Sorry,
we've closed";
```

Selain itu Dart juga mendukung ***conditional expressions*** seperti berikut:

```
expression1 ?? expression2
   var buyer = name ?? 'user';
```

Pada kode di atas jika variabel name tidak bernilai ***`null`***, maka `buyer` akan menyimpan nilai dari `name`. Namun jika tidak, `buyer` akan berisi `'user'`.

Perulangan

Perulangan adalah proses mengulang-ulang eksekusi satu statement atau lebih blok *statement* tanpa henti, selama kondisi yang dijadikan acuan terpenuhi. Biasanya disiapkan variabel untuk iterasi atau variabel penanda kapan perulangan akan diberhentikan. Perulangan ini dibagi menjadi dua kelompok, yaitu: *counted loop* dan *uncounted loop*.

- ***Counted Loop*** merupakan perulangan yang jelas dan sudah ditentukan berapa banyak perulangannya. Perulangan yang termasuk dalam *counted loop* meliputi. perulangan `for` dan perulangan `for-in`
- ***Uncounted Loop***, merupakan perulangan yang tidak jelas berapa kali ia harus mengulang. Perulangan yang termasuk dalam *uncounted loop* meliputi perulangan `while` dan perulangan `Do..While`

1. **Perulangan for**

   Struktur perulangan `for` bisa digunakan untuk mengulang proses yang sudah diketahui jumlah perulangannya. Struktur penulisan `for` lebih efisien susunannya lebih sederhana karena sudah diketahui batas awal dan merupakan salah satu bagian dari *counted loop* yang saya jelaskan sebelumnya.

   Format penulisan *for*

```
for(inisialisasi nilai awal; kondisi/syarat; statement
control increment/decrement){
      //perintah yang diulang
}
```

Yang perlu diperhatikan adalah kondisi yang ada di dalam tanda kurung setelah kata `for`. Kondisi ini akan menentukan:

a. Nilai awal suatu perulangan atau **inisialisasi nilai awal** perulangan
b. Batas perulangan yang bergantung pada suatu **kondisi/syarat**
c. Perubahan nilai awal hingga mencapai suatu batas perulangan melalui penambahan atau pengurangan nilai yang bergantung pada ***statement control increment/decrement***

2. Perulangan for-in

Perulangan `for-in` biasanya digunakan untuk mencetak item di dalam suatu *collection*. Perulangan ini termasuk dalam perulangan *counted loop*, karena jumlah perulangannya ditentukan oleh panjang dari array.
contoh :

```
dynamic data= ['buku', '25','tas','3.14'];
for(var contoh in data){
      print(contoh);
}
```

3. Perulangan while

Sama seperti statement `for`, cara kerja perulangan ini seperti percabangan, ia akan melakukan perulangan selama kondisinya benar atau `true`. Jadi perbedaan perulangan `while` dengan `for` adalah diharuskannya pembuatan nilai awal variabel yang kemudian dibuat batas akhir pada suatu kondisi melalui operasi *increment* atau *decrement*.

Format penulisan `while`

```
while(kondisi){
      statement yang dijalankan;
      statement control;
}
```

Pada kode diatas, kondisi sebagai memiliki nilai `true` atau `false`. Perulangan `while` akan berhenti jika kondisinya bernilai benar (`true`).

4. Perulangan do.. while

Statement `do..while` hampir sama dengan `while`. Perbedaannya, jika `do while` dilakukan minimal satu kali perulangan terlebih dahulu kemudian memeriksa kondisinya. Sedangkan `while` kondisi diperiksa dulu baru kemudian statement perulangan dijalankan. Jadi akibat dari itu `do..while` minimal terdapat 1x perulangan, sedangkan `while` dimungkinkan perulangan tidak pernah terjadi, ketika kondisi bernilai `false`.

Format penulisan `do..while`

```
do(
     Blog pernyataan;
}while(kondisi)
```

Method

Method adalah suatu operasi atau serangkaian perintah yang dapat dipanggil pada suatu program. *Method* didefinisikan untuk menyederhanakan suatu proses ke dalam satu bentuk perintah. Terdapat dua jenis *method* yaitu:

1. *Function* : *Method* yang mengembalikan suatu nilai.
2. *Procedure* : *Method* yang tidak mengembalikan suatu nilai.

Functions pada Dart digunakan untuk menghasilkan *output* berdasarkan *input* tertentu yang diberikan. Selain itu juga digunakan sebagai blok kode atau prosedur yang dapat digunakan kembali. Sadar atau tidak, sebenarnya kita telah mengimplementasikan beberapa *functions* pada kode kita. Semua program Dart dimulai dari fungsi `main()`. `main()` adalah contoh fungsi utama yang selalu kita gunakan.

Selain itu, `print()` juga termasuk fungsi.

```
print('Hello Polindra!');
```

Fungsi `print()` akan mengambil nilai *String* atau objek lainnya dan menampilkannya ke konsol. Untuk mencetak sesuatu ke konsol sebenarnya dibutuhkan beberapa instruksi yang lebih *low-level*, namun kita menjadi sangat terbantu dengan adanya fungsi `print()` ini dan dapat menggunakannya secara berulang.
Untuk mendeklarasikan fungsi, caranya sama dengan penulisan fungsi `main()` yaitu dengan menentukan tipe nilai balik atau *return value* lalu nama fungsi dan parameter inputnya.

```
returnType functionName(type param1, type param2, ...) {
return result;
}
```

Setiap fungsi Dart selalu mengembalikan nilai. Namun ada satu tipe data khusus yang bisa kita lihat pada fungsi main yaitu `return type` ***void***. *Keyword* `void` berarti fungsi tersebut tidak menghasilkan *output* atau nilai kembali. Biasanya fungsi seperti ini digunakan untuk kumpulan instruksi atau prosedur yang berulang dan sering digunakan.
Setelah fungsi dibuat, selanjutnya kita bisa memanggilnya pada fungsi `main()` atau pada bagian program lain yang Anda inginkan.

```
void main() {
     greet();      //output : Hello!
}
void greet() {
```

```
print('Hello’);
}
```

Pada contoh sederhana di atas fungsi `greet()` memang belum menghemat banyak kode yang Anda tulis. Namun, apabila Anda memiliki 30 instruksi `greet()` dan ternyata versi terbaru aplikasi Anda memerlukan perubahan teks yang ditampilkan, Anda cukup ubah satu baris kode saja, tak perlu 30 baris kode yang berbeda. Selain itu, jika Anda memiliki kode yang cukup panjang akan lebih baik jika kode tersebut dimasukkan ke dalam fungsi supaya lebih mudah dibaca.

Function parameters

Pada beberapa kasus fungsi bisa memerlukan *input* data untuk diproses. *Input* data ini kita kenal sebagai *parameter*. Untuk menambahkan parameter ke dalam fungsi, kita bisa memasukkannya ke dalam tanda kurung. Sebuah fungsi bisa menerima nol, satu, atau beberapa parameter.

Contoh penggunaan parameter pada fungsi yang pernah kita lihat adalah pada fungsi *`print()`*.

```
print('Hello Polindra!');
```

Berikut ini adalah contoh fungsi dengan dua parameter:

```
void main() {
    greet('Dicoding', 2015); // output : Halo Dicoding! Tahun ini
Anda berusia 5 tahun
}
void greet(String name, bornYear) {
var age = 2020 - bornYear;
     print('Halo $name! Tahun ini Anda berusia $age tahun');
}
```

Sebuah fungsi juga bisa menghasilkan *output* atau mengembalikan nilai. Fungsi yang mengembalikan nilai ditandai dengan definisi *return type* selain *`void`* dan memiliki *keyword* *`return`*. Contohnya seperti berikut:

```
void main() {
  var firstNumber = 7;
  var secondNumber = 10;
 print('Rata-rata   dari   $firstNumber   &   $secondNumber   adalah
${average(firstNumber, secondNumber)}');
}
double average(num num1, num num2) {
  return (num1 + num2) / 2;
```

```
}
```

Jika fungsi hanya memiliki satu baris kode atau instruksi di dalamnya, maka bisa disingkat dengan anotasi `=>`. Ini juga dikenal dengan nama `arrow syntax`.

```
double average(num num1, num num2) => (num1 + num2) / 2;
void greeting() => print('Hello');
```

Optional parameters

Anda memiliki fungsi seperti berikut:

```
void greetNewUser(String name, int age, bool isVerified)
```

Satu-satunya cara untuk bisa memanggil fungsi di atas adalah dengan cara berikut:

```
greetNewUser('Widy', 20, true);
```

Namun, Dart mendukung *`optional parameter`*, di mana kita tidak wajib mengisi parameter yang diminta oleh fungsi. Untuk bisa membuat parameter menjadi opsional, kita perlu memasukkannya ke dalam kurung siku seperti contoh berikut:

```
void greetNewUser([String name, int age, bool isVerified])
```

Cara ini disebut dengan *`positional optional parameters`*. Dengan *`optional parameter`* seperti di atas kita bisa memanggil fungsi seperti berikut:

```
greetNewUser('Widy', 20, true);
greetNewUser('Widy', 20);
greetNewUser('Widy');
greetNewUser();
```

Setiap parameter yang tidak dimasukkan akan memiliki nilai *`null`*. Dengan cara ini, urutan parameter masih perlu diperhatikan sehingga jika kita hanya ingin mengisi parameter terakhir, kita perlu mengisi parameter sebelumnya dengan *`null`*.

```
greetNewUser(null, null, true);
```

Untuk mengatasi masalah di atas kita bisa memanfaatkan *`named optional parameters`*. Pada opsi ini kita menggunakan kurung kurawal pada parameter.

```
void greetNewUser({String name, int age, bool isVerified})
```

Dengan cara ini Anda bisa memasukkan parameter tanpa mempedulikan urutan parameter dengan menyebutkan nama parameternya.

```
greetNewUser(name: 'Widy', age: 20, isVerified: true);
greetNewUser(name: 'Widy', age: 20);
greetNewUser(age: 20);
greetNewUser(isVerified: true);
```

Terkadang `null` bukanlah pilihan yang kita inginkan ketika menggunakan *`optional parameter`*. Sebagai solusi, kita bisa menggunakan *`default value parameter`*. Kita akan memberikan nilai *default* pada parameter lalu nilai ini akan digunakan jika kita tidak memasukkan parameternya.

```
void greetNewUser({String name = "Dicoding", int age = 5, bool
isVerified = false})
```

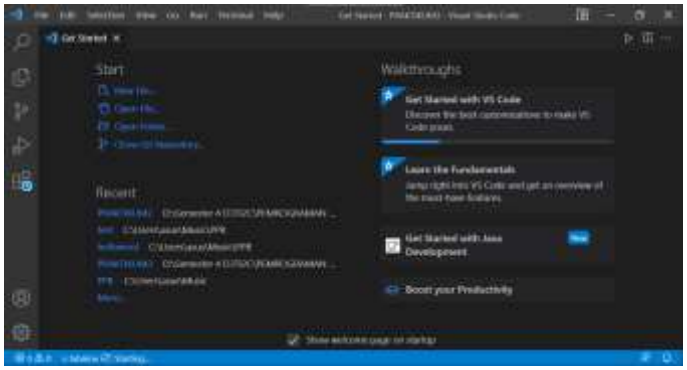
Named Parameters

Secara default, posisi parameter ketika kita memanggil *function* harus sesuai dengan posisi parameter di *function* tersebut. Dart memiliki fitur dengan *named parameter*, dimana saat memanggil parameter kita bisa menyebutkan nama parameter nya, sehingga posisinya tidak perlu harus sesuai dengan posisi parameternya. Namun ketika membuat *function* nya, kita perlu melakukan perubahan ketika membuat parameter nya, yaitu dengan menggunakan kurung kurawal `{}`. Secara *default*, *named parameter* adalah *nullable*, sehingga kita perlu tambahkan karakter.

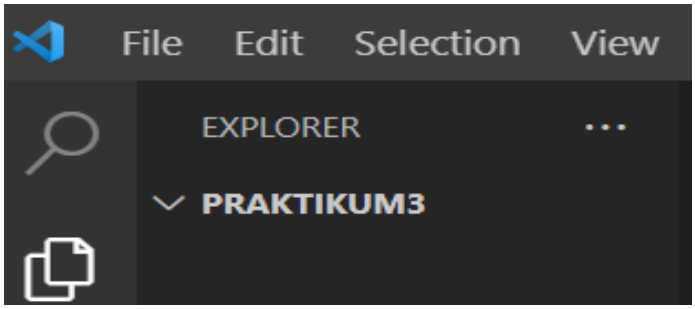
C. LANGKAH DAN HASIL PELAKSANAAN PRAKTIKUM

Langkah-langkah praktikum Percabangan If

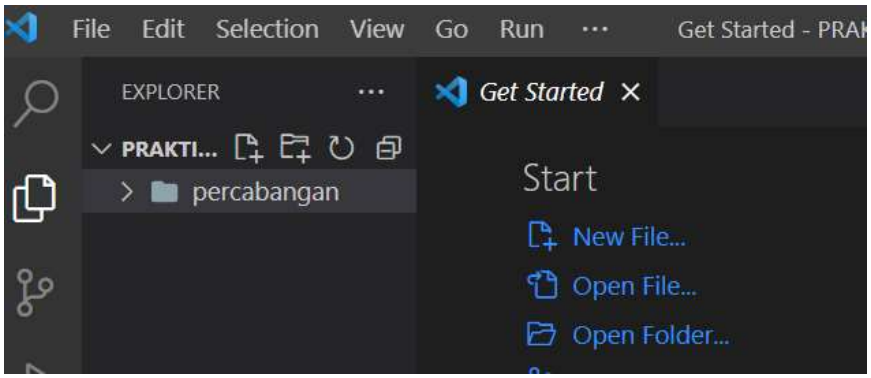
1. Buka aplikasi text editor *visual studio code*



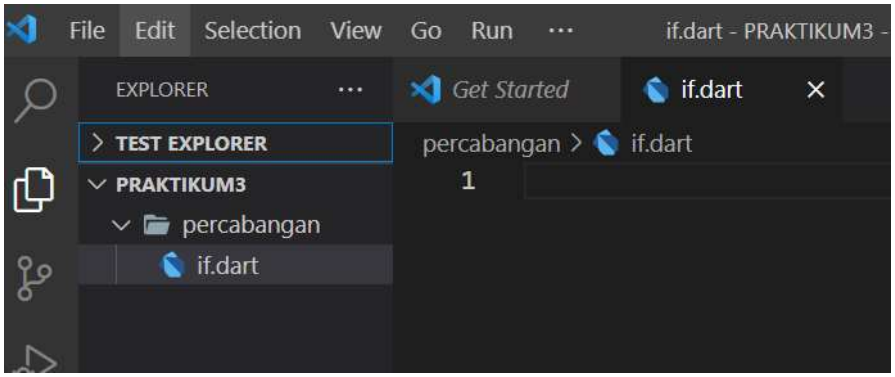
2. Buatlah sebuah folder yang akan digunakan untuk penyimpanan dengan nama "**PRAKTIKUM3**"



3. Buat sebuah *folder* pada folder **PRAKTIKUM3** dengan cara mengklik icon *New Folder* pada folder tersebut. Buat nama foldernya dengan nama **Percabangan**

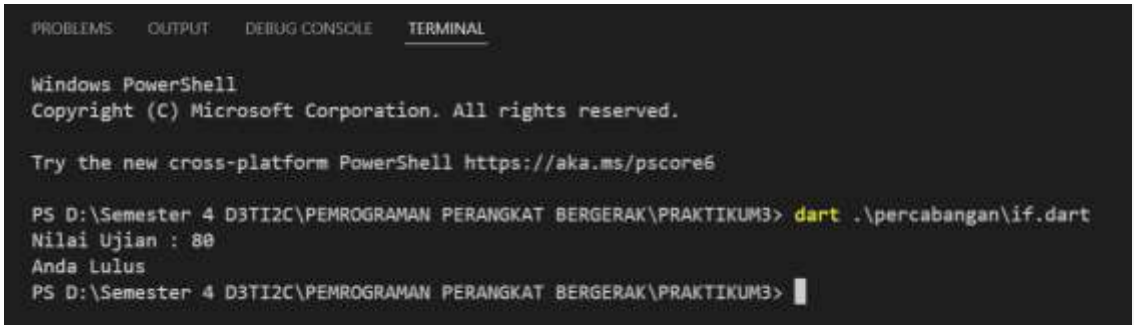


4. Buat sebuah *File* pada folder **Percabangan** dengan cara mengklik icon *New File* pada folder percabangan atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `If.dart`



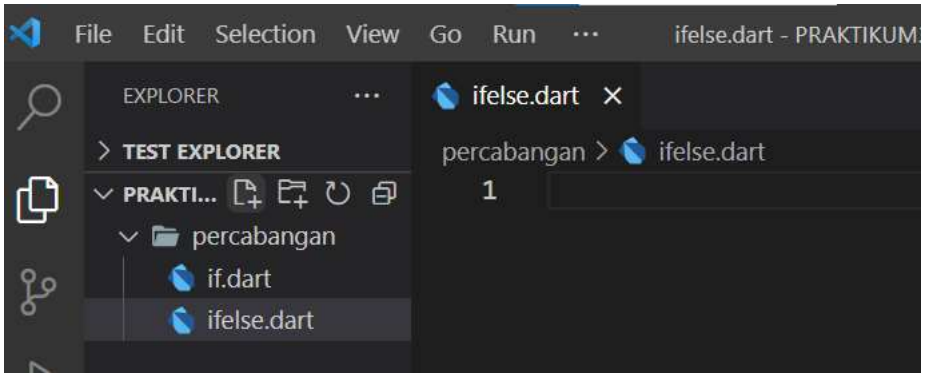
5. Tulis dan simpan kode script berikut ini

6. Kemudian jalankan kode script `if.dart` melalui menu *Terminal* > *New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Percabangan` dan tekan ENTER. Kemudian ketik perintah `dart if.dart`



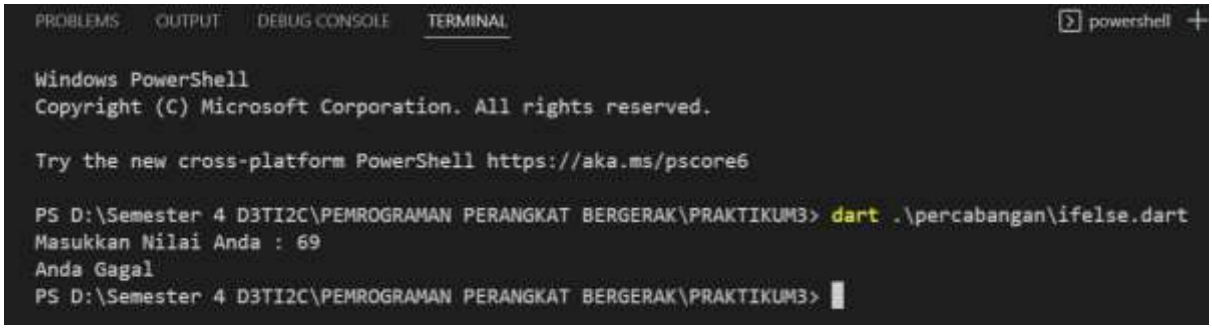
Langkah-langkah praktikum Percabangan If..Else

1. Diasumsikan masih berada pada folder **Percabangan**
2. Buat sebuah file pada folder **Percabangan** dengan cara mengklik icon *New File* pada folder tersebut atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `ifelse.dart`



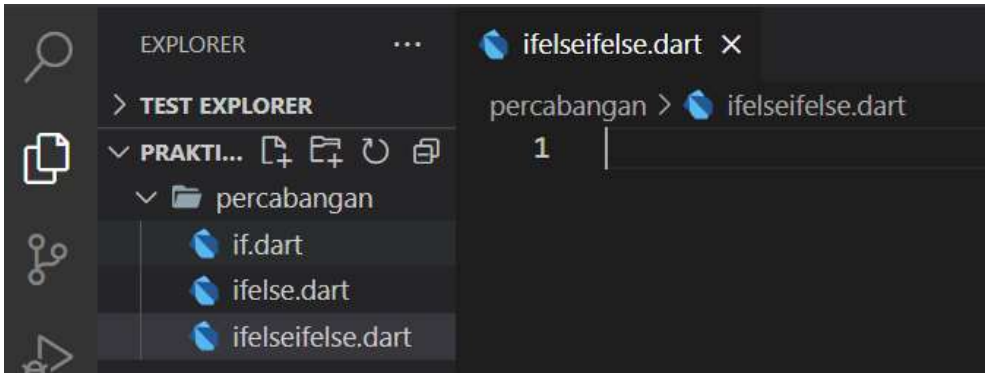
3. Tulis dan simpan kode script berikut ini

4. Kemudian jalankan kode script `ifelse.dart` melalui menu *Terminal* > *New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Percabangan` dan tekan ENTER. Kemudian ketik perintah `dart ifelse.dart`

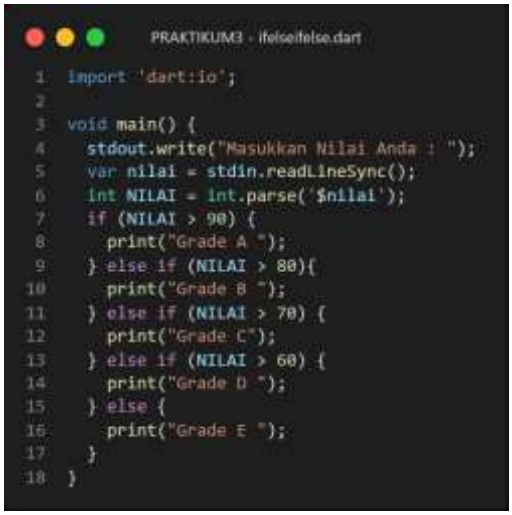


Langkah-langkah praktikum Percabangan if..elseif..else

1. Diasumsikan masih berada pada folder **percabangan**
2. Buat sebuah file pada folder **percabangan** dengan cara mengklik icon *New File* pada folder tersebut atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `ifelseifelse.dart`



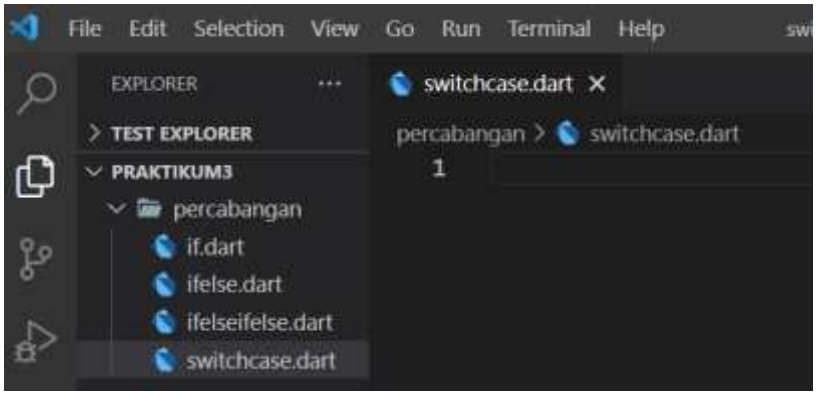
3. Tulis dan simpan kode script berikut ini



4. Kemudian jalankan kode script `ifelseifelse.dart` melalui menu *Terminal* > *New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Percabangan` dan tekan ENTER. Kemudian ketik perintah `dart ifelseifelse.dart`

Langkah-langkah praktikum Percabangan Switch..Case

1. Diasumsikan masih berada pada folder **Percabangan**
2. Buat sebuah file pada folder **Percabangan** dengan cara mengklik icon *New File* pada folder tersebut atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `switchcase.dart`



3. Tulis dan simpan kode script berikut ini

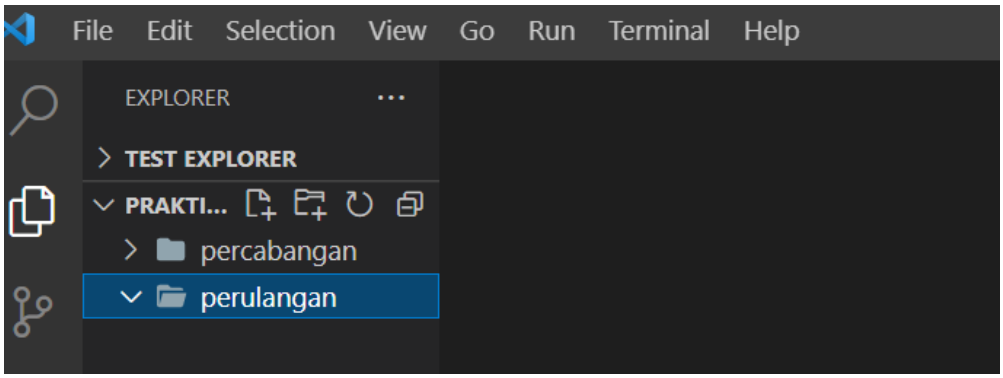
PRAKTIKUM3 - switchcase.dart

```dart
import 'dart:io';

main() {
  print("LAMPU LALU LINTAS");
  stdout.write("Warna : ");
  var warna = stdin.readLineSync();
  String arti;
  switch (warna) {
    case "merah":
    {
      arti = "berhenti";
      break;
    }
    case "kuning":
    {
      arti = "hati-hati";
      break;
    }
    case "hijau":
    {
      arti = "Maju";
      break;
    }
    default:
    {
      arti = "Gak ada Warna itu...!";
    }
  }
  print(arti);
}
```

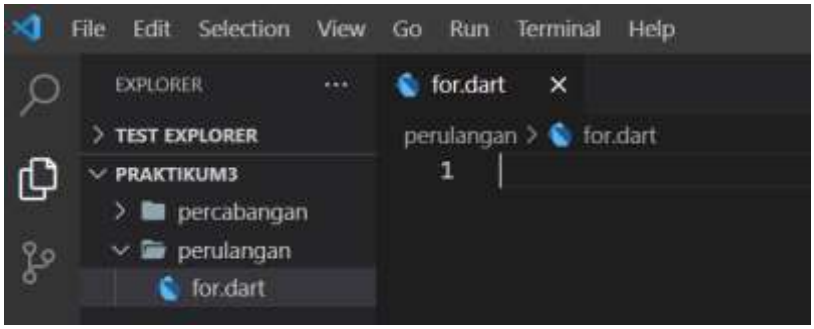
4. Kemudian jalankan kode script `switchcase.dart` melalui menu *Terminal* > *New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Percabangan` dan tekan ENTER. Kemudian ketik perintah `dart switchcase.dart`

Langkah-langkah praktikum Perulangan For

1. Diasumsikan masih berada pada folder **PRAKTIKUM3**
2. Buat sebuah *folder* pada folder **PRAKTIKUM3** dengan cara mengklik icon *New Folder* pada folder tersebut. Buat nama foldernya dengan nama **Perulangan**



3. Buat sebuah file pada folder **Perulangan** dengan cara mengklik icon *New File* pada folder tersebut atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `for.dart`

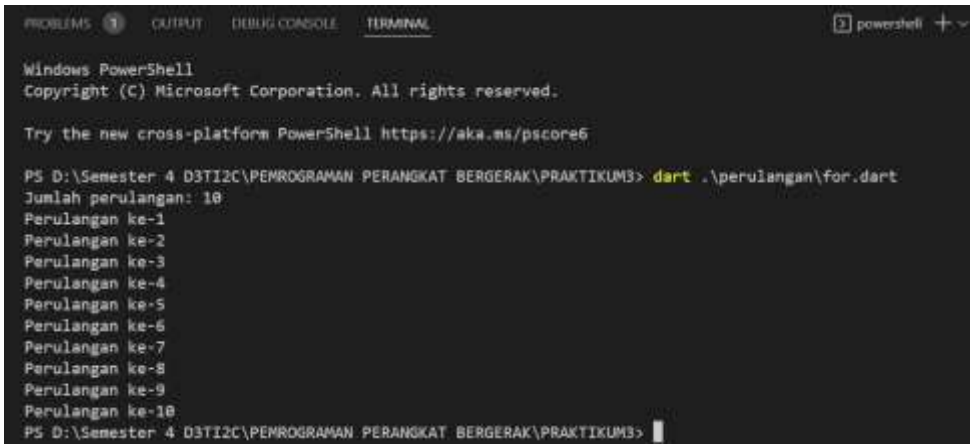


4. Tulis dan simpan kode script berikut ini

PRAKTIKUM3 - for.dart

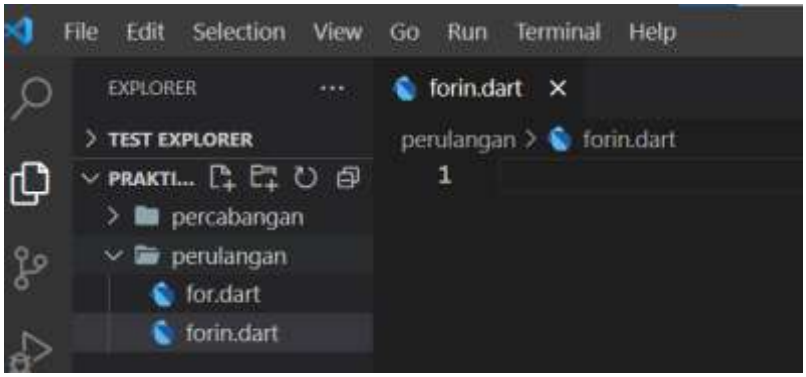
```dart
import 'dart:io';

void main() {
  int n = 10;
  print("Jumlah perulangan: $n");
  for (int i = 1; i <= n; i++) {
    print("Perulangan ke-$i");
  }
}
```

5. Kemudian jalankan kode script `for.dart` melalui menu *Terminal* > *New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Perulangan` dan tekan ENTER. Kemudian ketik perintah `dart for.dart`



Langkah-langkah praktikum Perulangan for-in

1. Diasumsikan masih berada pada folder **Perulangan**
2. Buat sebuah file pada folder **Perulangan** dengan cara mengklik icon *New File* pada folder tersebut atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `forin.dart`



3. Tulis dan simpan kode script berikut ini

PRAKTIKUM3 - forin.dart

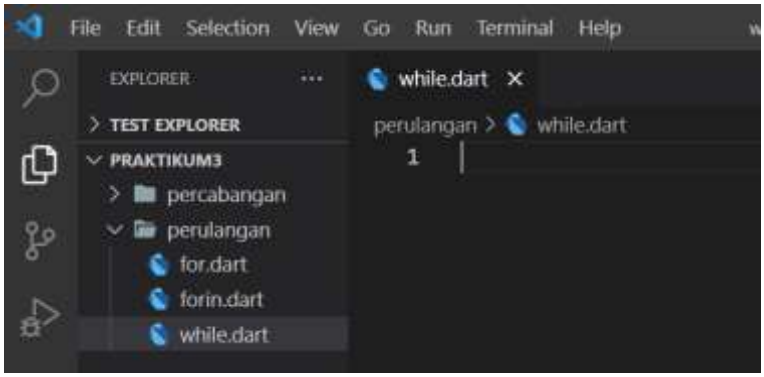
```dart
import 'dart:io';

void main() {
  var cemilan = ["Comro", "Gehu", "Cireng", "Cilok", "Javascript"];
  print("Daftar Cemilan ");
  print("-------------");
  for (var makanan in cemilan) {
    print(makanan);
  }
  print("------------");
  print("Total Cemilan : ${cemilan.length}");
}
```

4. Kemudian jalankan kode script `forin.dart` melalui menu *Terminal > New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Perulangan` dan tekan ENTER. Kemudian ketik perintah `dart forin.dart`

```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL                    powershell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Semester 4 D3TI2C\PEMROGRAMAN PERANGKAT BERGERAK\PRAKTIKUM3> dart .\perulangan\forin.dart
Daftar Cemilan
-------------
Comro
Gehu
Cireng
Cilok
Javascript
------------
Total Cemilan : 5
PS D:\Semester 4 D3TI2C\PEMROGRAMAN PERANGKAT BERGERAK\PRAKTIKUM3>
```

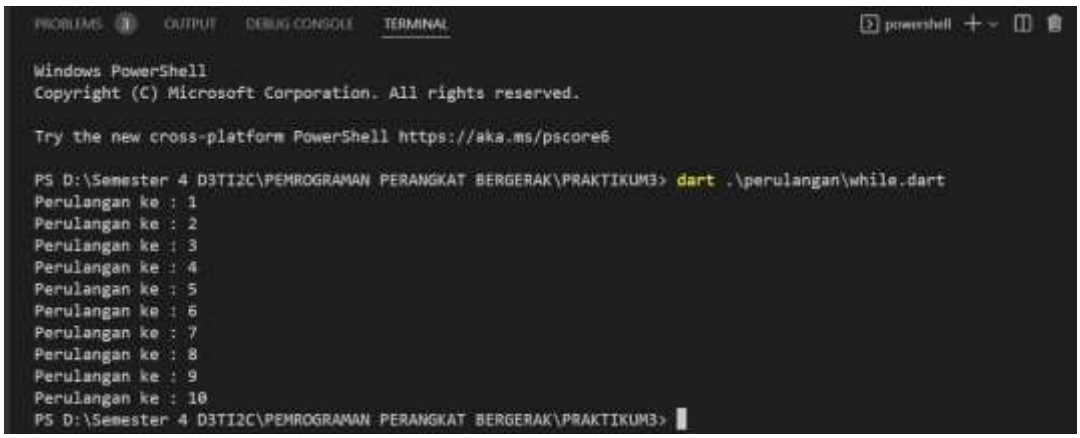
Langkah-langkah praktikum Perulangan While

1. Diasumsikan masih berada pada folder **Perulangan**
2. Buat sebuah file pada folder **Perulangan** dengan cara mengklik icon *New File* pada folder tersebut atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `while.dart`



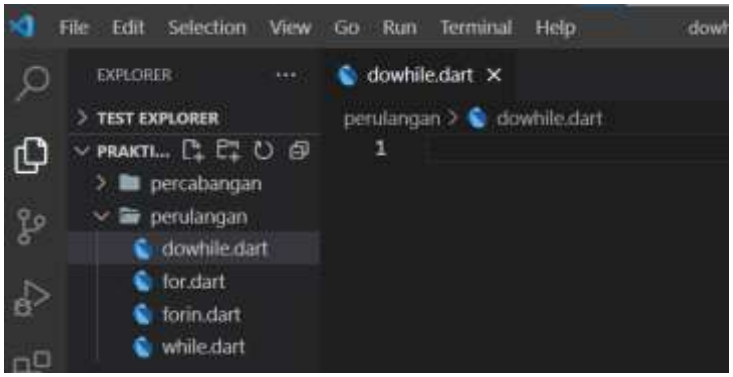
3. Tulis dan simpan kode script berikut ini

4. Kemudian jalankan kode script `while.dart` melalui menu *Terminal* > *New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Perulangan` dan tekan ENTER. Kemudian ketik perintah `dart while.dart`



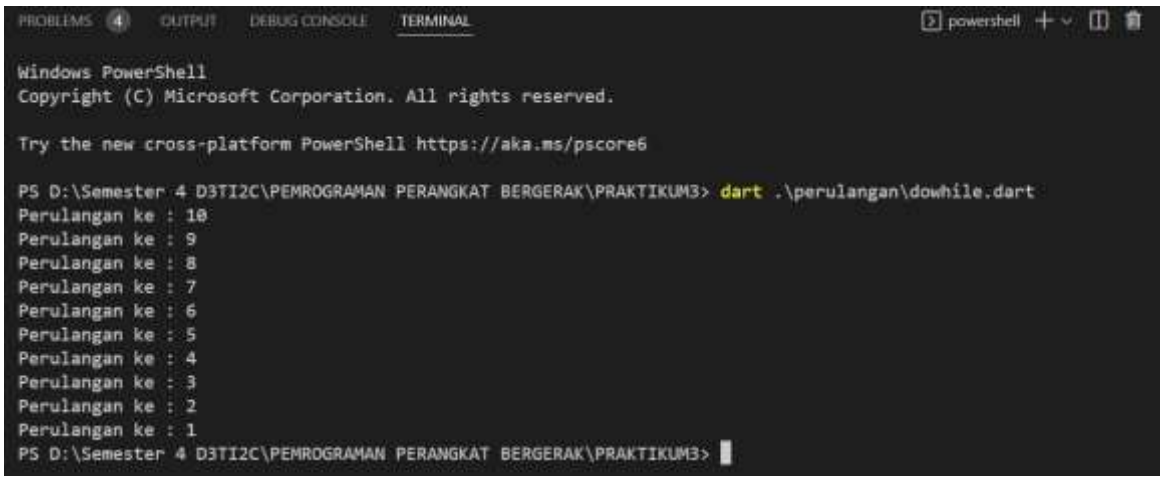
Langkah-langkah praktikum Perulangan Do..While

1. Diasumsikan masih berada pada folder **Perulangan**
2. Buat sebuah file pada folder **Perulangan** dengan cara mengklik icon *New File* pada folder tersebut atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `dowhile.dart`



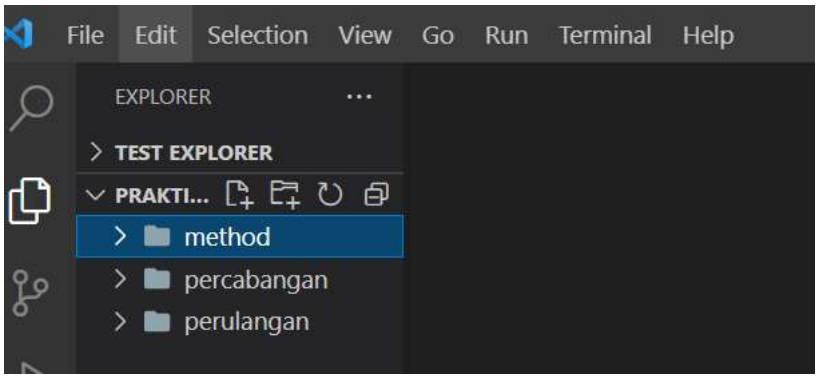
3. Tulis dan simpan kode script berikut ini

4. Kemudian jalankan kode script `dowhile.dart` melalui menu *Terminal* > *New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Perulangan` dan tekan ENTER. Kemudian ketik perintah `dart dowhile.dart`

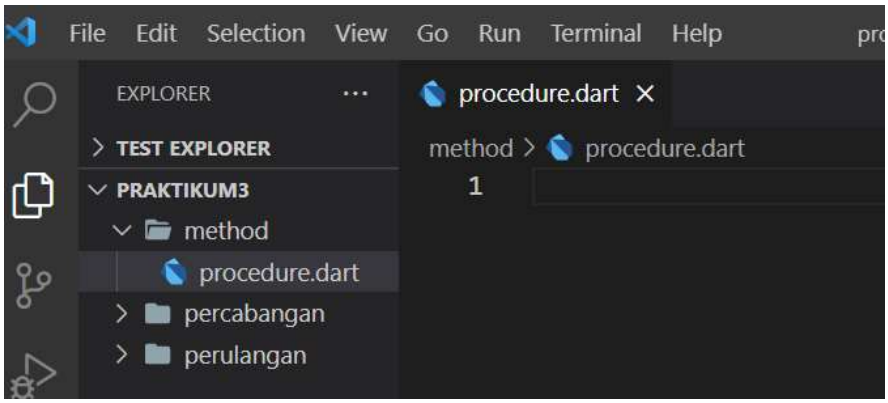


Langkah-langkah praktikum Method Procedure

1. Diasumsikan masih berada pada folder **PRAKTIKUM3**
2. Buat sebuah *folder* pada folder **PRAKTIKUM3** dengan cara mengklik icon *New Folder* pada folder tersebut. Buat nama foldernya dengan nama **Method**



3. Buat sebuah file pada folder **Method** dengan cara mengklik icon *New File* pada folder tersebut atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `procedure.dart`



4. Tulis dan simpan kode script berikut ini

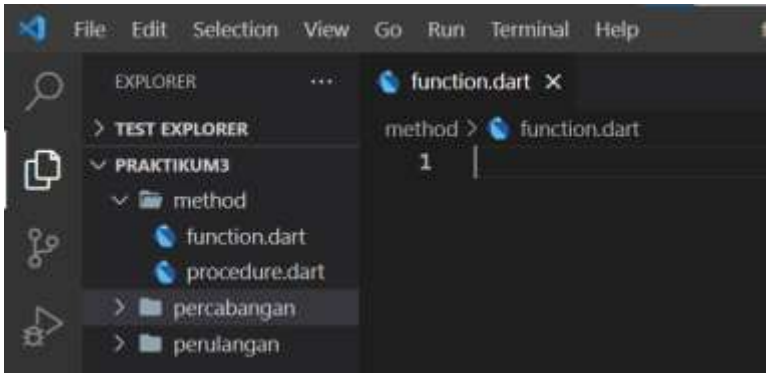
PRAKTIKUM3 - procedure.dart

```dart
void main() {
  var waktusekarang = 17;
  var dia = 'Nur Budi Nugraha';
  if (waktusekarang == 17) {
    eat(dia);
  } else {
    play(dia);
  }
}

void eat(var nama) {
  print('$nama sedang  makan.');
}

void play(var nama) {
  print('$nama sedang main.');
}
```

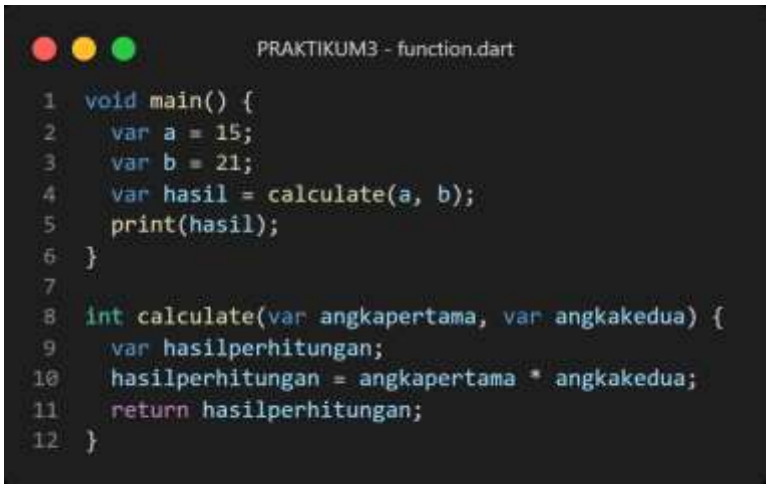
5. Kemudian jalankan kode script `procedure.dart` melalui menu *Terminal* > *New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Method` dan tekan ENTER. Kemudian ketik perintah `dart procedure.dart`

Langkah-langkah praktikum Method Function

1. Diasumsikan masih berada pada folder **Method**
2. Buat sebuah file pada folder **Method** dengan cara mengklik icon *New File* pada folder tersebut atau dengan cara klik menu *File* > pilih *New File.* Buat nama file nya dengan nama `function.dart`



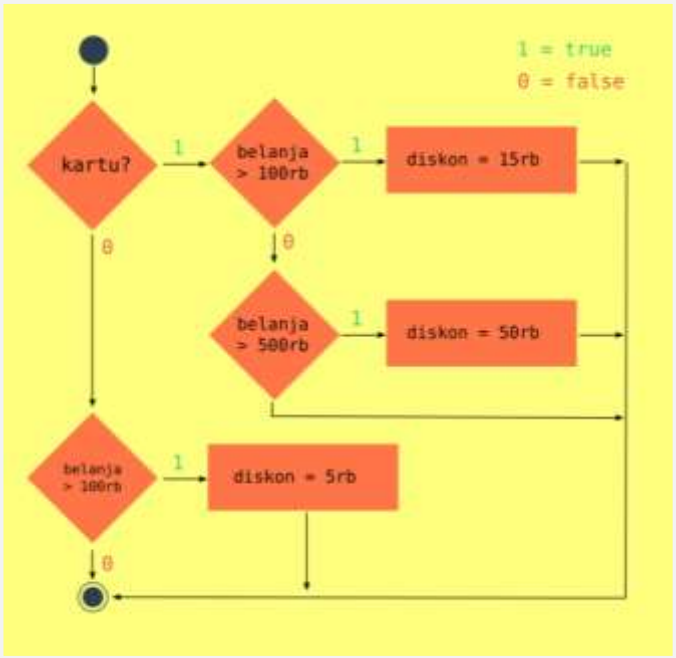
3. Tulis dan simpan kode script berikut ini



4. Kemudian jalankan kode script `function.dart` melalui menu *Terminal* > *New Terminal* (Ctrl+Shift+`), lalu ketik perintah `cd Method` dan tekan ENTER. Kemudian ketik perintah `dart function.dart`

D. HASIL DAN PENJELASAN TUGAS

1. Buatlah program belanja dengan kondisi seperti *flowchart* di bawah ini



Kode Program :

PRAKTIKUM3 - 1.dart

```dart
import 'dart:io';

void main() {
  print("Aplikasi Pembayaran");
  stdout.write("Mempunyai kartu member : yes/no : ");

  var member = stdin.readLineSync();
  stdout.write("Total Pembayaran : ");
  var belanja = stdin.readLineSync();

  int BELANJA = int.parse('$belanja');
  String? diskon;

  if (member == 'yes') {
    if (BELANJA > 500000) {
      diskon = "50000";
    } else if (BELANJA > 100000) {
      diskon = "15000";
    } else {
      diskon = "Tidak Ada Diskon";
    }
  } else {
    if (BELANJA > 100000) {
      diskon = "5000";
    } else {
      diskon = "Tidak Ada Diskon";
    }
  }
  print("Diskon : $diskon");
}
```

Hasil Kode Program

2. Tentu Anda mengenal syair lagu anak-anak berikut ini:

Anak ayam turun 10
Anak ayam turun 10, mati satu tinggal 9
Anak ayam turun 9, mati satu tinggal 8
Anak ayam turun 8, mati satu tinggal 7
Anak ayam turun 7, mati satu tinggal 6
Anak ayam turun 6, mati satu tinggal 5
Anak ayam turun 5, mati satu tinggal 4
Anak ayam turun 4, mati satu tinggal 3
Anak ayam turun 3, mati satu tinggal 2
Anak ayam turun 2, mati satu tinggal 1
Anak ayam turun 1, mati satu tinggal induknya

Gunakan fungsi perulangan FOR untuk mengenerate syair lagu anak ayam tersebut untuk jumlah anak ayam mula-mula adalah N. Buatlah input untuk memasukkan sembarang nilai N ini, setelah itu tekan ENTER. Selanjutnya akan muncul baris syair seperti di atas.

Kode Program

PRAKTIKUM3 - 2.dart

```dart
import 'dart:io';

void main() {
  stdout.write("Masukan jumlah ayam : ");
  int n = int.parse(stdin.readLineSync()!);

  print('anak ayam turun $n');
  for (int i = n; i > 1; i--) {
    int s = i - 1;
    print('Anak ayam turun $i, mati satu tinggal $s');
  }
  print('Anak ayam turun 1, mati 1 tinggal induknya ');
}
```

Hasil Kode Program

```
PS D:\Semester 4 D3TI2C\PEMROGRAMAN PERANGKAT BERGERAK\Tugas Praktikum\PRAKTIKUM3> dart .\tugass\2.dart
Masukan jumlah ayam : 10
anak ayam turun 10
Anak ayam turun 10, mati satu tinggal 9
Anak ayam turun 9, mati satu tinggal 8
Anak ayam turun 8, mati satu tinggal 7
Anak ayam turun 7, mati satu tinggal 6
Anak ayam turun 6, mati satu tinggal 5
Anak ayam turun 5, mati satu tinggal 4
Anak ayam turun 4, mati satu tinggal 3
Anak ayam turun 3, mati satu tinggal 2
Anak ayam turun 2, mati satu tinggal 1
Anak ayam turun 1, mati 1 tinggal induknya
PS D:\Semester 4 D3TI2C\PEMROGRAMAN PERANGKAT BERGERAK\Tugas Praktikum\PRAKTIKUM3>
```

3. Diketahui terdapat suatu perpustakaan yang mengalami kendala dalam pengelolaan data buku, data mahasiswa, dan data peminjaman buku. Kendalanya antara lain seringkali ditemukan mahasiswa yang meminjam buku hingga stok pada perpustakaan habis, mengingat petugas perpustakaan minim informasi stok buku terkini. Menindaklanjuti permasalahan tersebut tim pengurus perpustakaan memberlakukan peraturan yang hanya mengizinkan mahasiswa meminjam buku tidak lebih dari 2 buku. Agar kegiatan peminjaman buku lebih terorganisasi berdasarkan sistem yang telah dibuat, maka tim pengurus perpustakaan sepakat untuk menggunakan bantuan suatu aplikasi yang ringan *(command line)* dan bisa berjalan pada perangkat *desktop*.

   Sebagai *problem solver*, Anda diminta untuk menangani permasalahan tersebut. Maka, buatlah program dengan menggunakan bahasa Dart yang bisa membantu menyelesaikan permasalahan mereka dengan ketentuan sebagai berikut:
   1. Penyimpanan data buku, mahasiswa, dan peminjaman buku dilakukan dengan memanfaatkan *dynamic collection*
   2. Pengisian data buku, mahasiswa, dan peminjaman dilakukan oleh pengguna *(user),* bukan melalui *hard code*
   3. Stok yang diisi pada suatu data buku minimal 2
   4. Judul buku dan nim/nama mahasiswa tidak boleh ada yang terduplikasi
   5. Jumlah total peminjaman buku hanya diperbolehkan 2 buku per mahasiswa
   6. Jika stok buku tinggal 1, maka buku tersebut tidak boleh/tidak dapat dipinjam
   7. Digunakan percabangan, perulangan, dan pembuatan *method* (*function* atau *procedure*) sebagai bentuk konkrit penyelesaian masalah melalui kode program Dart
   8. Penyajian tampilan interaksi program dibebaskan untuk setiap mahasiswa
   9. Ringkasan struktur penyajian data buku, mahasiswa, dan peminjaman mengacu pada tabel berikut:

buku:

id	judul	penerbit	stok
001	Belajar Dart	Informatika	4
002	Belajar Flutter	Andi	5

mahasiswa:

nim	nama
09030015	Joko
09030016	Udin

peminjaman:

no		nim	nama	buku	
				id	judul
1		09030015	Joko	001	Belajar Dart
				002	Belajar Flutter
2		09030016	Udin	001	Belajar Dart

Kode Program :

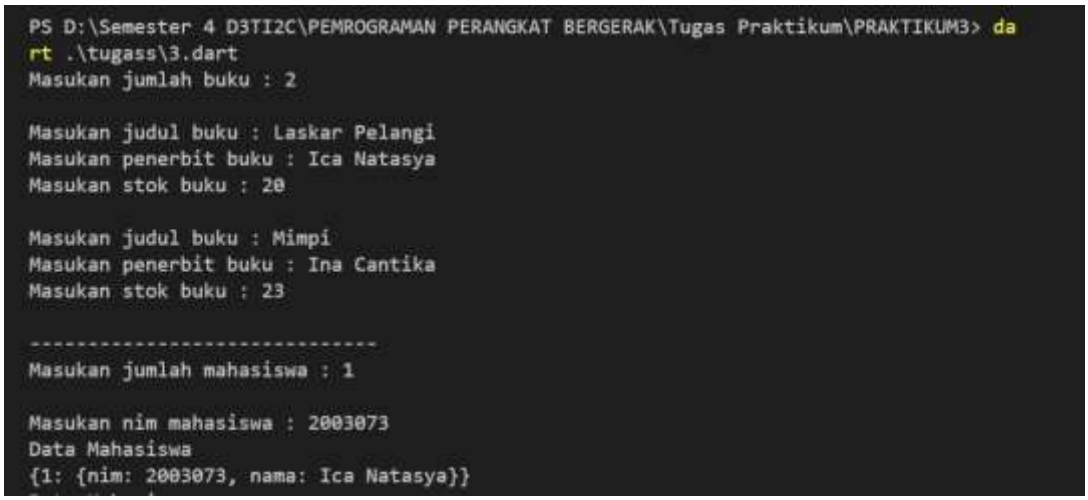
PRAKTIKUM3 - 3.dart

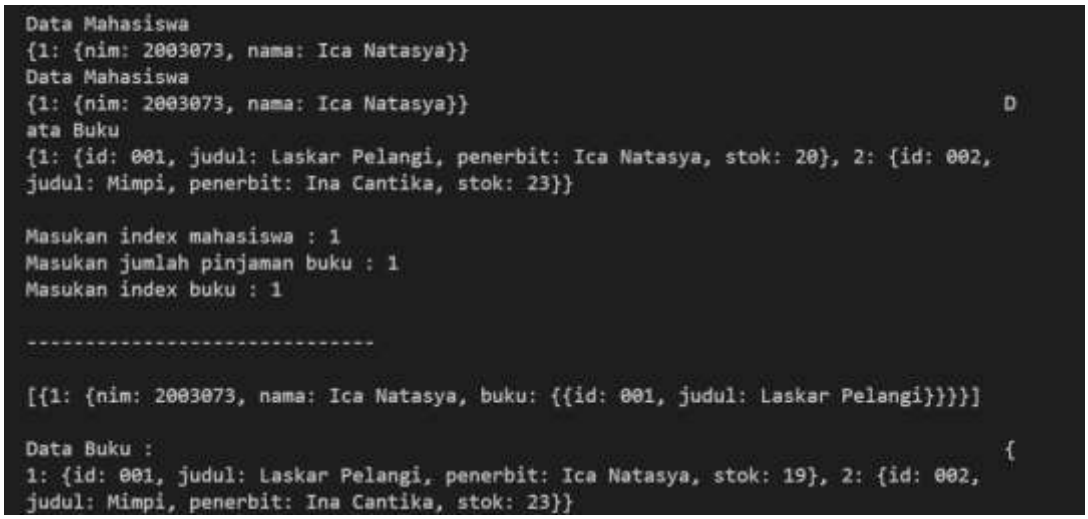
```dart
import 'dart:io';

void main(List<String> args) {
  var dataBuku = new Map();
  var dataMahasiswa = new Map();
  List<dynamic> dataPeminjam = <dynamic>[];
  List<dynamic> dataPinjaman = <dynamic>[];

  stdout.write('Masukan jumlah buku : ');
  int? jumlahBuku = int.parse(stdin.readLineSync()!);

  print("");

  for (int i = 1; i <= jumlahBuku; i++) {
    stdout.write('Masukan judul buku : ');
    String? judulBuku = stdin.readLineSync()!;
    stdout.write('Masukan penerbit buku : ');
    String? penerbitBuku = stdin.readLineSync()!;
    stdout.write('Masukan stok buku : ');
    int? stokBuku = int.parse(stdin.readLineSync()!);

    print("");

    dataBuku[i] = {
      'id': '00$i',
      'judul': judulBuku,
      'penerbit': penerbitBuku,
      'stok': stokBuku,
    };
  }

  print("-----------------------------");

  stdout.write('Masukan jumlah mahasiswa : ');
  int? jumlahMahasiswa = int.parse(stdin.readLineSync()!);

  print("");

  for (int j = 1; j <= jumlahMahasiswa; j++) {
    stdout.write('Masukan nim mahasiswa : ');
    String? nimMahasiswa = stdin.readLineSync()!;
    stdout.write('Masukan nama mahasiswa : ');
    String? namaMahasiswa = stdin.readLineSync()!;

    dataMahasiswa[j] = {'nim': nimMahasiswa, 'nama': namaMahasiswa};

    print("");
  }

  print("-----------------------------");
```

PRAKTIKUM3 - 3.dart

```dart

  stdout.write('Masukan jumlah peminjam : ');
  int? jumlahPeminjam = int.parse(stdin.readLineSync()!);

  print("");

  print("Data Mahasiswa");
  print(dataMahasiswa);
  print("Data Buku");
  print(dataBuku);

  print("");

  for (int k = 1; k <= jumlahPeminjam; k++) {
    stdout.write('Masukan index mahasiswa : ');
    int? indexMahasiswa = int.parse(stdin.readLineSync()!);
    stdout.write('Masukan jumlah pinjaman buku : ');
    int? jumlahPinjaman = int.parse(stdin.readLineSync()!);

    dataPinjaman = <dynamic>[];
    for (var l = 1; l <= jumlahPinjaman; l++) {
      stdout.write('Masukan index buku : ');
      int? indexBuku = int.parse(stdin.readLineSync()!);
      dataBuku[indexBuku]['stok'] -= 1;
      dataPinjaman.add(indexBuku);
    }

    dataPeminjam.add({
      k: {
        'nim': dataMahasiswa[indexMahasiswa]['nim'],
        'nama': dataMahasiswa[indexMahasiswa]['nama'],
        'buku': {
          for (var dp in dataPinjaman)
            {'id': dataBuku[dp]['id'], 'judul': dataBuku[dp]['judul']}
        }
      }
    });

    print("");
  }

  print("-----------------------------");

  print("");
  print(dataPeminjam);

  print("");

  print("Data Buku :");
  print(dataBuku);
}
```

Hasil Kode Program





E. REFERENSI

- Dieter Meiller. “Modern App Development with Dart and Flutter 2”. The Deutsche Nationalbibliothek. Berlin. 2021.
- Alberto Miola. ”Flutter Complete Reference Create Beautiful, Fast and Native Apps for Any Device”. Independently Published. 2020.
- Kode TR. “Belajar Flutter Mulai dari Dasar menggunakan Pemrograman Dart”. 2019. https://www.kodetr.com/belajar-flutter-mulai-dari-dasar-menggunakan-pemrograman-dart/. Diakses tanggal 19 Februari 2022.
- Omadi Jaya. “Pengenalan Syntax dan Data Type pada Dart Lang”. 2020. https://belajarflutter.com/dart-pengenalan-syntax-dan-data-type-pada-dart/. Diakses tanggal 19 Februari 2022.