

MINGGU KE III

PEMROGAMAN BERORIENTASI OBJEK

Deskripsi

Materi pemrograman berorientasi objek memadukan teori dan praktik melalui *case study* dalam kehidupan nyata, mulai dari konsep dasar pemrograman OOP meliputi class, abstraction, inheritance, encapsulasi, polimorphism.

Tujuan Pembelajaran

Setelah mahasiswa mempelajari dan mempraktikkan materi OOP dengan metode pembelajaran *case study*, mahasiswa mampu memahami, menerapkan dan implementasi OOP dalam bahasa pemrograman dart sehingga mahasiswa memiliki kemudahan pemahaman saat menggunakan bahasa pemrograman pada skala besar (framework flutter) yang berorientasi pada OOP. Lebih spesifik, mahasiswa memiliki kemampuan

C. Konsep Dasar Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan cara bagaimana sistem perangkat lunak yang dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktifitas:

1. Analisis berorientasi objek
2. Perancangan berorientasi objek
3. Pemrograman berorientasi objek
4. Pengujian berorientasi objek

Pendekatan berorientasi objek akan memandang sistem yang dikembangkan sebagai suatu kumpulan objek yang berkoresponden dengan objek-objek dunia nyata. Ada banyak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dari abstraksi objek, kelas, hubungan antar kelas sampai dengan abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dipunyai oleh objek akan dienkapsulasi (dibungkus) menjadi satu kesatuan.

Pemrograman berorientasi objek sebuah tata cara pembuatan program (*programming paradigm*) dengan menggunakan konsep “objek”, dimana objek ini bisa memiliki data (dikenal dengan istilah

atribut) dan kode program dalam bentuk prosedur (dikenal dengan istilah method). Dalam teori pemrograman, terdapat 3 prinsip dasar yang melandasi pemrograman berorientasi objek, yakni *encapsulation*, *inheritance* dan *polymorphism*. Pada modul ini akan fokus pada pemrograman berorientasi objek

B. Best Practice OOP Bahasa Pemrograman Dart

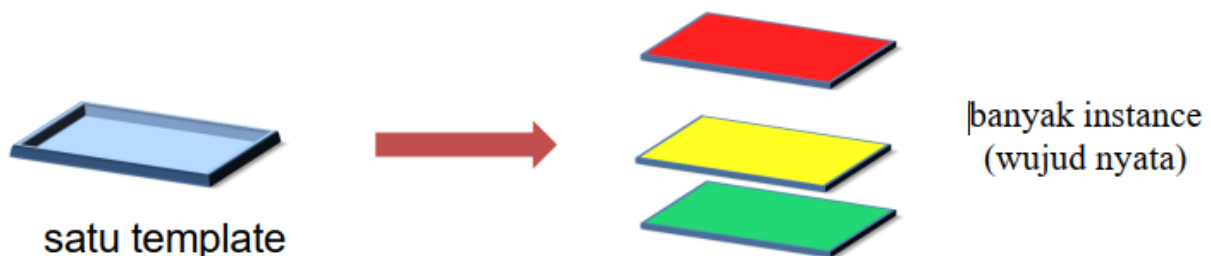
Review konsep OOP dari mata kuliah Metode Perancangan Program

1. Objek (*Object*)

- Objek adalah representasi sebuah entitas yang memiliki makna tertentu yang menjadi perhatian sipemandang.
- Segala sesuatu yang ada di dunia adalah objek. Cth : Manusia, Bunga, Hewan, Mobil, Meja, Kursi, Sepeda, Kereta, Pesawat terbang, dll.
- Setiap sistem terdiri dari objek-objek (sistem juga termasuk objek).
- Evaluasi & pengembangan sistem disebabkan oleh interaksi antara objek-objek di dalam atau di luar sistem.

2. Kelas (*Class*)

- Merupakan template untuk membuat obyek. merupakan prototipe/blue prints yang mendefinisikan variable-variabel dan method –method secara umum
- Objek (instances) merupakan hasil instansiasi dari suatu kelas, proses pembentukan obyek dari suatu class disebut dengan instantiation.
- Analogi hubungan antara class dan objek seperti gambar dibawah



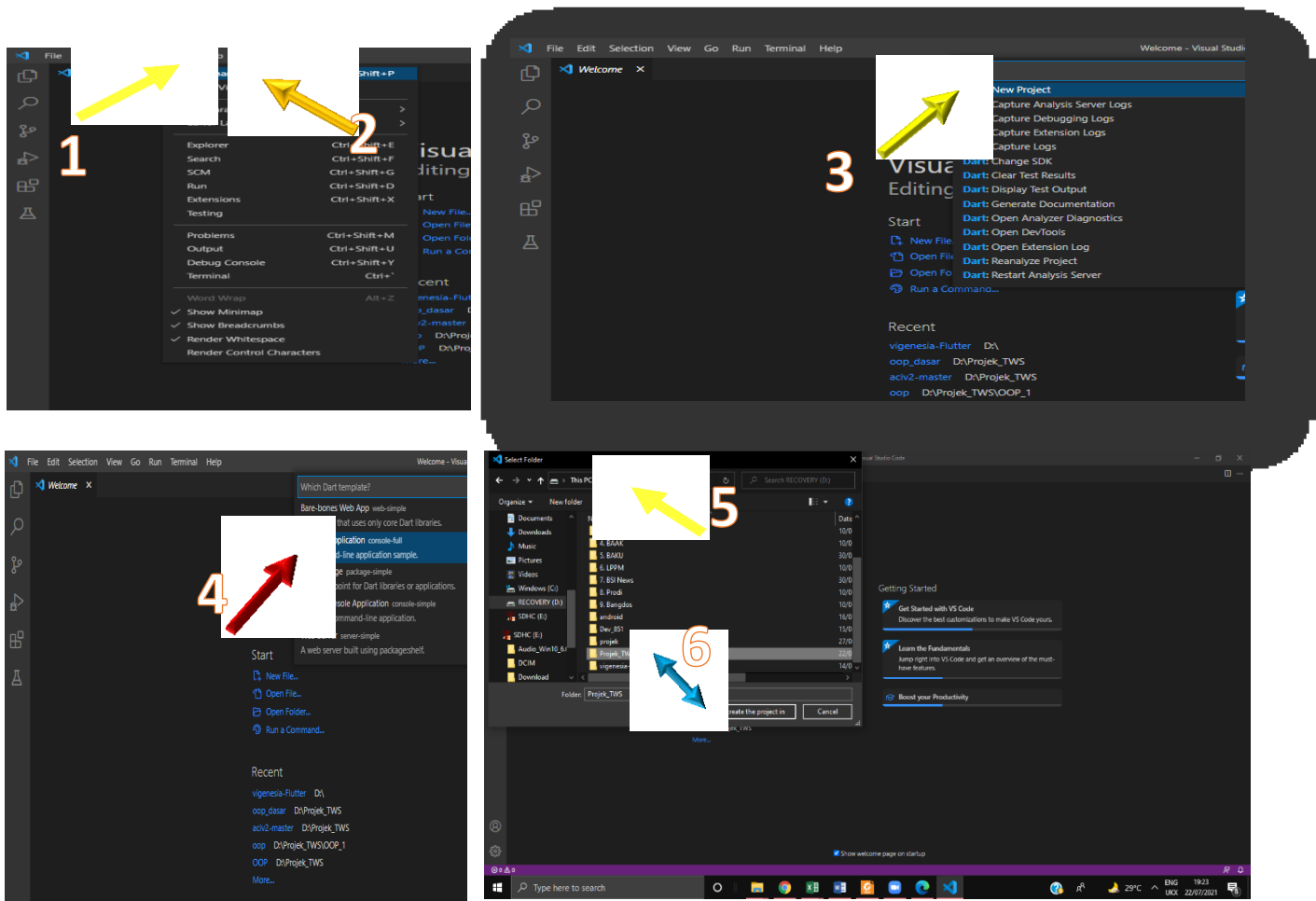
3. Atribut dan metod

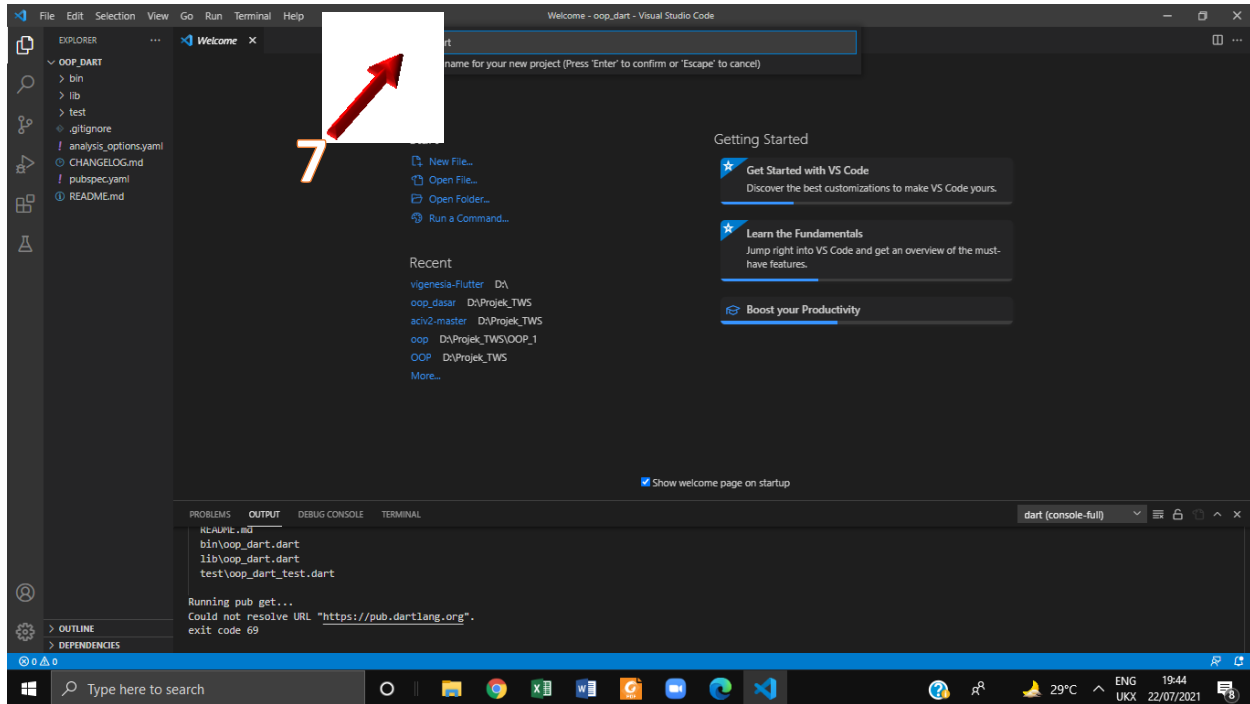
- Atribut adalah data yang membedakan antara objek satu dengan yang lain. Contoh atribut mobil : manufaktur, model, warna, jumlah pintu, ukuran engine, kecepatan dll
- Dalam class, atribut disebut sebagai variabel.
- Instance variable
 - adalah atribut untuk tiap obyek dari class yang sama.
 - Tiap obyek mempunyai dan menyimpan nilai atributnya sendiri.
 - Jadi tiap obyek dari class yang sama boleh mempunyai nilai yang sama atau beda
- Class variable
 - adalah atribut untuk semua obyek yang dibuat dari class yang sama.
 - Semua obyek mempunyai nilai atribut yang sama.
 - Jadi semua obyek dari class yang sama mempunyai hanya satu nilai yang value nya sama.
- Tingkah laku/behavior adalah hal-hal yang bisa dilakukan oleh objek dari suatu class.

- f. Behavior dapat digunakan untuk mengubah nilai atribut suatu objek, menerima informasi dari objek lain, dan mengirim informasi ke obyek lain untuk melakukan suatu task.
- g. Dalam class, behavior disebut juga sebagai method
- h. Method adalah serangkaian statements dalamsuatu class yang handle suatu task tertentu.
- i. Cara objek berkomunikasi dengan objek lain adalah dengan menggunakan method.

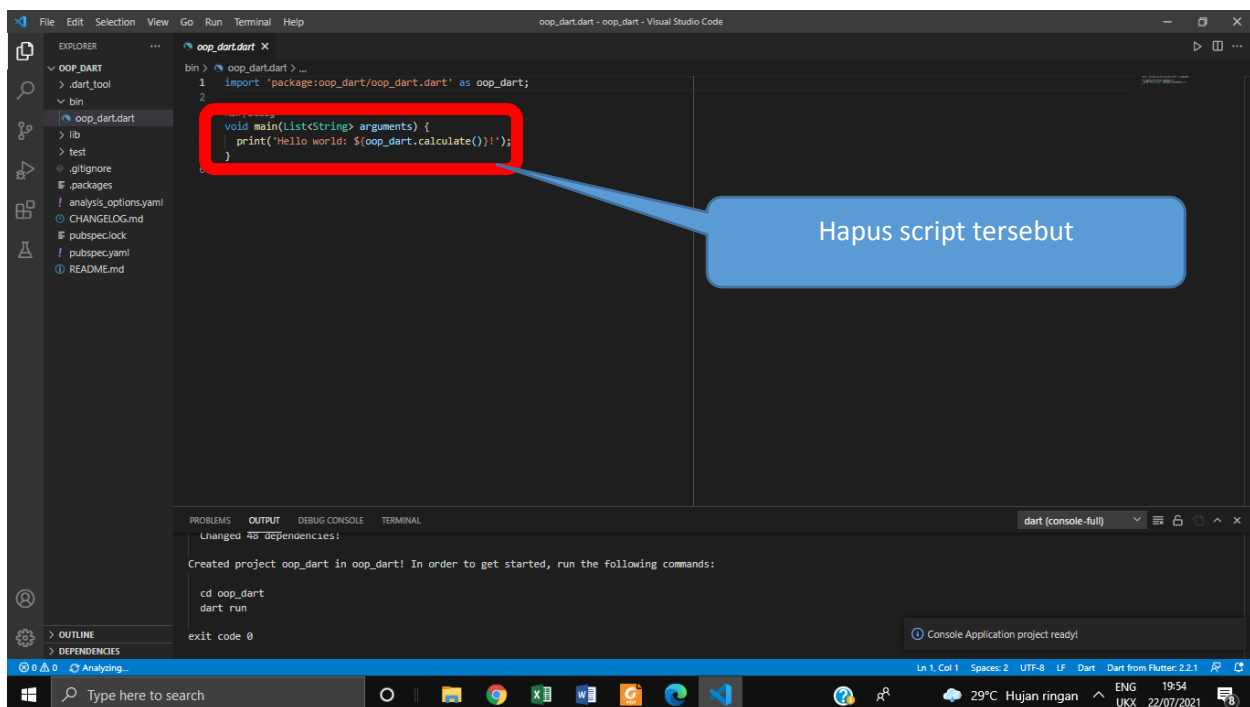
C. Best Practice OOP Bahasa Pemrograman Dart

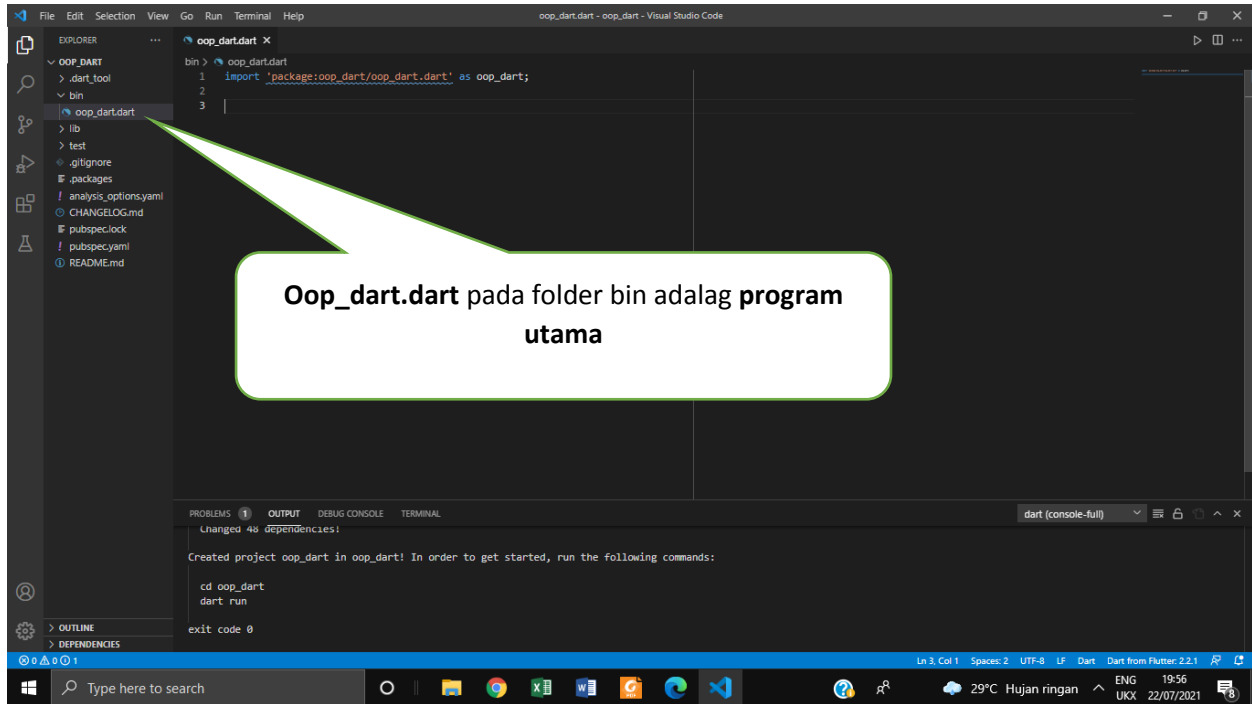
1. Persiapan:
 - a. Pastikan perangkat komputer mahasiswa terinstall editor Visual Studio Code
 - b. Siapkan folder misal -> **D:\Projek_TWS\OOP_Dart**
2. Aktifkan editor Visual Studio Code
3. Langkah awal persiapan pemrograman OOP



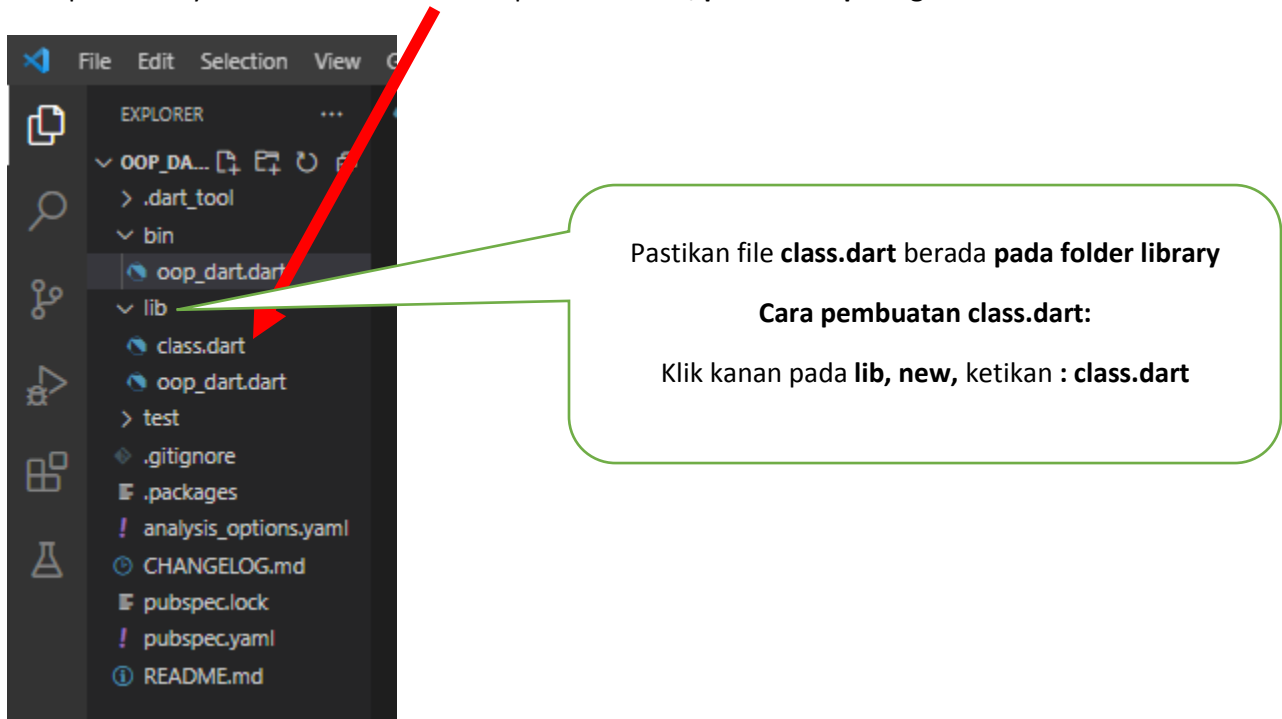


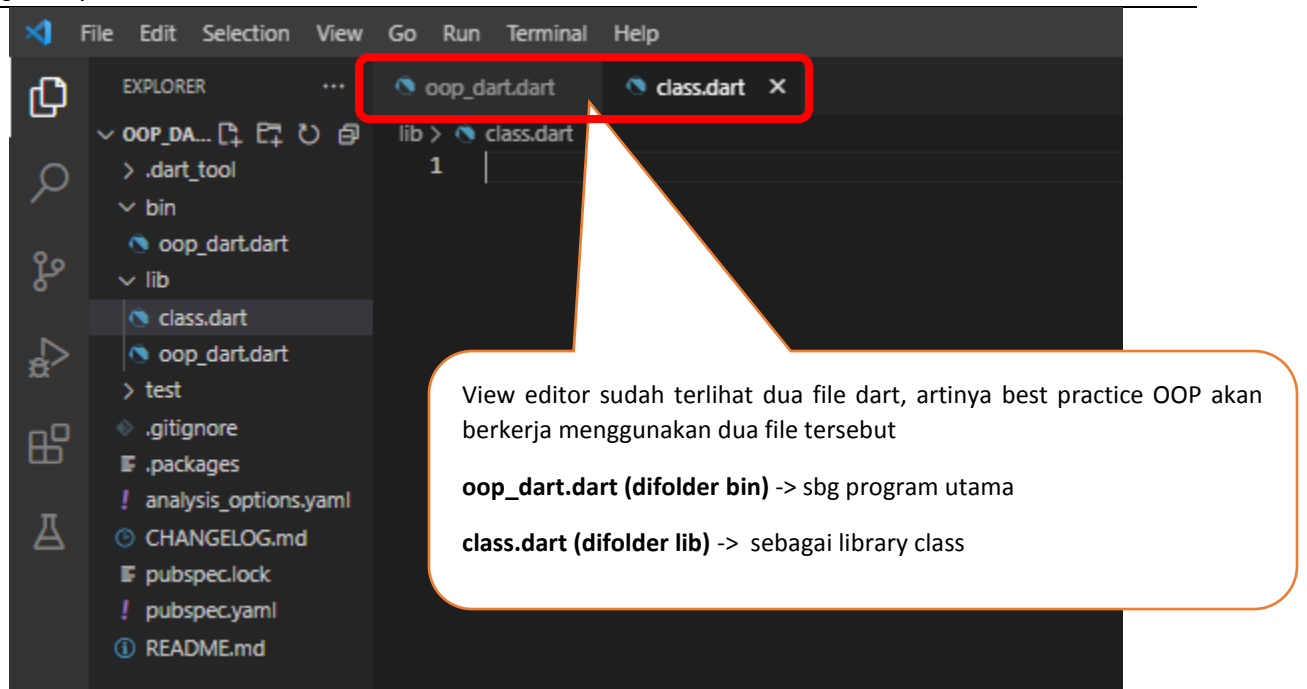
View akhir screen dari langkah 1-7





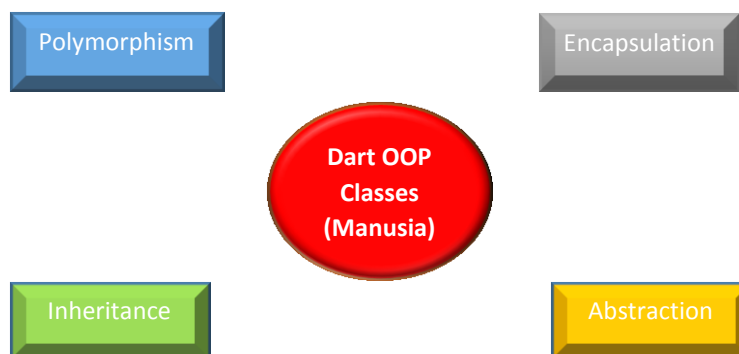
Setup berikutnya membuat file **class.dart** pada folder **lib**, perhatikan pada gambar





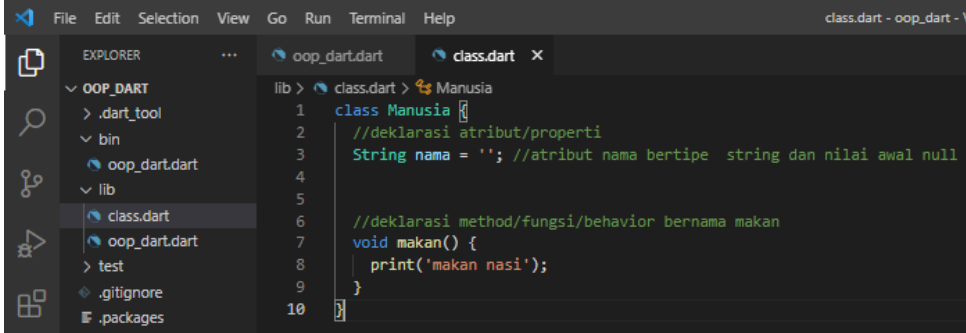
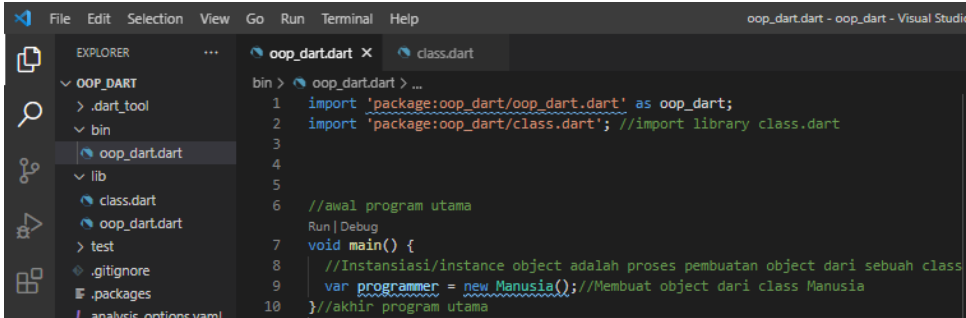
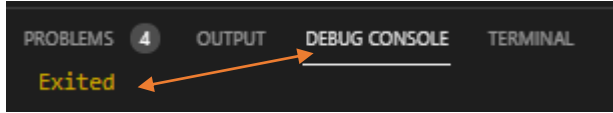
4. Tahapan Praktikum Pemograman berorientasi objek

Case study u praktikum OOP adalah



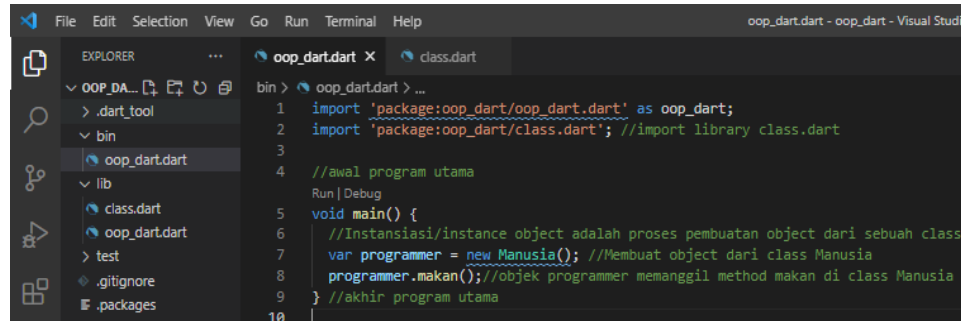
Sumber

<https://subscription.packtpub.com/book/mobile/9781788996082/2/ch02lvl1sec06/introduction-to-oop-in-dart>

Source	output
<p>1. Menciptakan class Manusia di file class.dart</p>  <pre> lib > class.dart > Manusia 1 class Manusia 2 //deklarasi atribut/properti 3 String nama = ''; //atribut nama bertipe string dan nilai awal null 4 5 6 //deklarasi method/fungsi/behavior bernama makan 7 void makan() { 8 print('makan nasi'); 9 } 10 </pre> <p>2. Mengakses class Manusia dilakukan pada program utama (oop_dart.dart), aktifkan tab oop_dart.dart dan tambahkan script sesuai gambar.</p>  <pre> bin > oop_dart.dart > ... 1 import 'package:oop_dart/oop_dart.dart' as oop_dart; 2 import 'package:oop_dart/class.dart'; //import library class.dart 3 4 5 6 //awal program utama 7 void main() { 8 //Instansiasi/instance object adalah proses pembuatan object dari sebuah class 9 var programmer = new Manusia(); //Membuat object dari class Manusia 10 } //akhir program utama </pre> <p>Sejauh ini tidak ada error, run menggunakan fitur Run Debug</p>	 <p>Penjelasan var programmer = new Manusia()</p> <p>Membuat object dari class Manusia yang disimpan dalam variabel programmer, dengan kata lain, variabel programmer adalah sebuah object dari class Manusia</p>

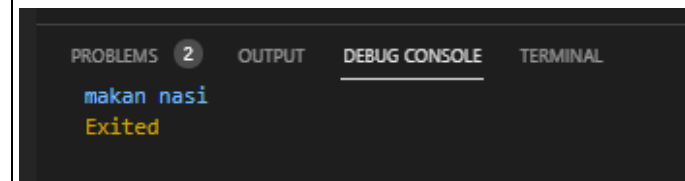
Case study: **programmer** itu manusia dan bisa melakukan **makan nasi** sehingga output yang diharapkan sebagai berikut

Tambahkan script pada program utama (**oop_dart**) seperti pada gambar



```
bin > oop_dart.dart > ...
1 import 'package:oop_dart/oop_dart.dart' as oop_dart;
2 import 'package:oop_dart/class.dart'; //import library class.dart
3
4 //awal program utama
5 void main() {
6   //Instansiasi/instance object adalah proses pembuatan object dari sebuah class
7   var programmer = new Manusia(); //Membuat object dari class Manusia
8   programmer.makan(); //objek programmer memanggil method makan di class Manusia
9 } //akhir program utama
10
```

programmer.makan(); objek programmer memanggil method **makan()**

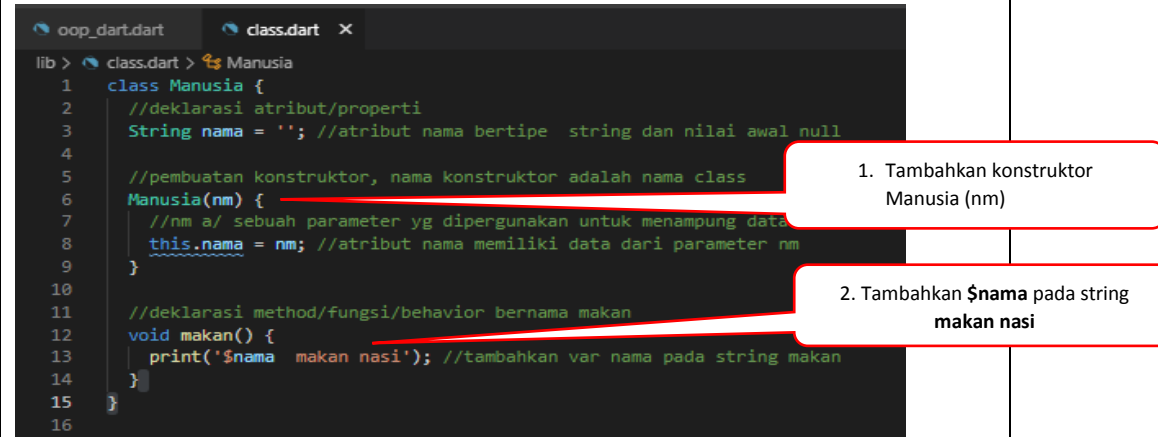


```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL
makan nasi
Exited
```

Output diatas belum sempurna, karena tidak diketahui programer nya siapa. Saatnya kita kembangkan misalnya programmernya **Fauko Misalam**. Output yang diharapkan

Fauko Misalam makan nasi

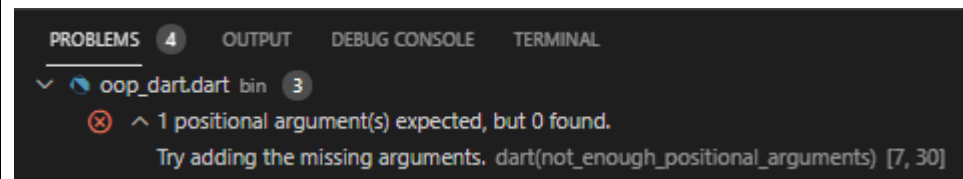
Case study: Fauko Misalam sebagai programmer dan dapat melakukan **maka nasi**, maka pengembangan scriptnya pada **class.dart** seperti gambar



```
lib > class.dart > Manusia
1 class Manusia {
2   //deklarasi atribut/properti
3   String nama = ''; //atribut nama bertipe string dan nilai awal null
4
5   //pembuatan konstruktor, nama konstruktor adalah nama class
6   Manusia(nm) {
7     //nm a/ sebuah parameter yg dipergunakan untuk menampung data
8     this.nama = nm; //atribut nama memiliki data dari parameter nm
9   }
10
11  //deklarasi method/fungsi/behavior bernama makan
12  void makan() {
13    print('$nama makan nasi'); //tambahkan var nama pada string makan
14  }
15 }
16
```

1. Tambahkan konstruktor Manusia (nm)

2. Tambahkan \$nama pada string makan nasi



PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL

oop_dart.dart bin 3

⊗ ^ 1 positional argument(s) expected, but 0 found.
Try adding the missing arguments. dart(not_enough_positional_arguments) [7, 30]

Akan ditemukan problem seperti gambar diatas dan untuk mengatasi tambahkan script pada **oop_dart.class**. yang ditambahkan **Fauko Misalam** pada baris **var programmer** seperti pada gambar dibawah

Output

```
bin > oop_dart.dart > ...
1 import 'package:oop_dart/oop_dart.dart' as oop_dart;
2 import 'package:oop_dart/class.dart'; //import library class.dart
3
4 //awal program utama
Run | Debug
5 void main() {
6   //Instansiasi/instance object adalah proses pembuatan object dari sebuah class
7   var programmer = new Manusia('Fauko Misalam'); //Membuat object dari class Manusia
8   programmer.makan(); //objek programmer memanggil method makan di class Manusia
9 } //akhir program utama
```

Setelah ditambahkan sudah tidak error, save dan silahkan **Run**

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
Fauko Misalam makan nasi
Exited
```

Case study yang akan dikembangkan melalui konsep bahwa manusia ada yang menjadi programmer, pedang, dokter, guru, dosen dan lain-lain. mereka memiliki pola yang sama dapat melakukan **makan nasi**, dan masing juga memiliki **nama**. Output yang diharapkan seperti pada kolom sebelah kanan.

Keywordnya:

1. Dengan memahami script yang sudah dipraktika, tentu mahasiswa bisa mengembangkan untuk mendapatkan output yang diinginkan

```
//Instansiasi/instance object adalah proses pembuatan object dari sebuah class
var programmer = new Manusia('Fauko Misalam'); //Membuat object dari class Manusia
programmer.makan(); //objek programmer memanggil method makan di class Manusia
```

2. Mahasiswa cukup mengembangkan coding sebelumnya pada **oop_dart.dart**
3. Yang perlu diingat adalah Ketiganya adalah Manusia (Fauko sebagai programmer, Intan sebagai dosen, dan Dio sebagai hacker) semua melakukan makan nasi

```
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
Fauko Misalam makan nasi
Intan makan nasi
Dio makan nasi
Exited
```

5. Inheritance

Mereview mata kuliah Metode Perancangan Program, Inheritance adalah class dapat menurunkan metode-metode dan properti-properti yang dimilikinya pada kelas lain. Kelas yang mewarisi metode dan properti dari objek lain dinamakan kelas turunan. Kelas turunan ini mampu mengembangkan metode sendiri. Berikut case study implementasi inheritance OOP pada bahasa Dart.

Source	Output
--------	--------

```
lib > class.dart > ...  
1 class Manusia {  
2   //deklarasi atribut/properti  
3   String nama = ''; //atribut nama bertipe string dan nilai awal null  
4  
5   //pembuatan konstruktor, nama konstruktor adalah nama class  
6   Manusia(nm) {  
7     //nm a/ sebuah parameter yg dipergunakan untuk menampung data  
8     this.nama = nm; //atribut nama memiliki data dari parameter nm  
9   }  
10  
11   //deklarasi method/fungsi/behavior bernama makan  
12   void makan() {  
13     print('$nama makan nasi'); //tambahkan var nama pada string makan  
14   }  
15 }  
16  
17 //awal inheritance  
18 class ManusalMilenial extends Manusia {  
19   String email = '';  
20  
21  
22   ManusalMilenial(String email) : super(email);  
23   void info() {  
24     print('nama: $nama, Email:$email ');  
25   }  
26 } //akhir inheritance
```

1. Tambahkan inheritance pada **class.dart**

```
32 //program utama
Run | Debug
33 void main() {
34     //Instansiasi/instance object adalah proses pembuatan object dari sebuah class
35     var programmer = new ManusalMilenial('Fauko Misalam');
36     programmer.email = 'fauko@bsi.ac.id';
37     programmer.info();
38     programmer.makan();
39
40     print('\n'); //pindah baris
41
42     var dosen = new ManusalMilenial('Intan');
43     dosen.email = 'intan@bsi.ac.id';
44     dosen.info();
45     dosen.makan();
46
47     print('\n'); //pindah baris
48
49     var hacker = new ManusalMilenial('Dio');
50     hacker.email = 'dio@bsi.ac.id';
51     hacker.info();
52     hacker.makan();
53 }
54
```

2. Lakukan modifikasi seperti pada gambar pada `oop_dart.dart`

Lakukan save dan Run

Output

```
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
nama: Fauko Misalam, Email:fauko@bsi.ac.id
Fauko Misalam makan nasi

nama: Intan, Email:intan@bsi.ac.id
Intan makan nasi

nama: Dio, Email:dio@bsi.ac.id
Dio makan nasi
Exited
```

Kesimpulan:

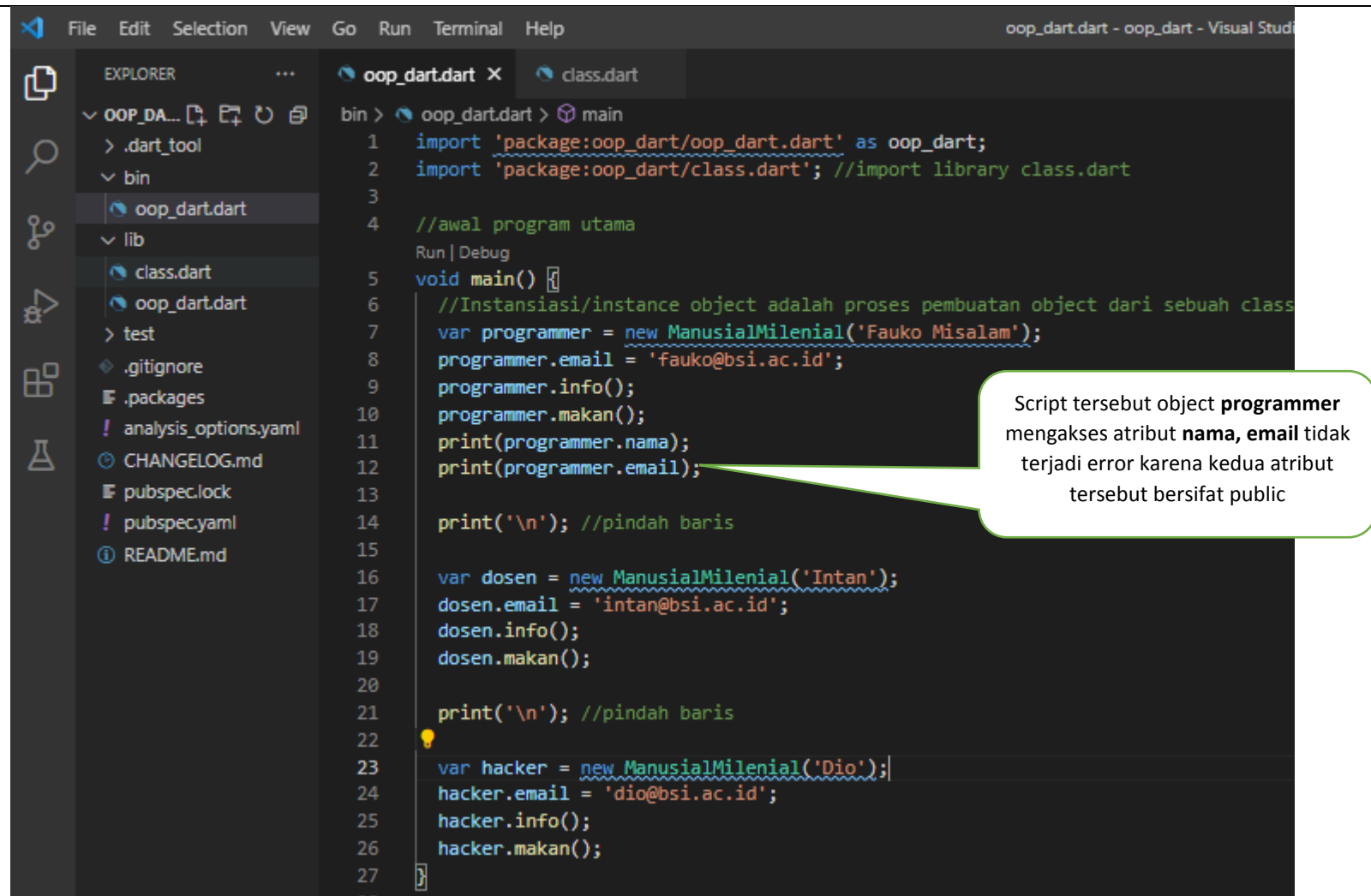
Inheritance pembuatannya seperti membuat class pada umumnya, namun yang perlu diperhatikan menggunakan *reserved word* **extends**. class **ManusiaMilenial** **extends** **Manusia**, bahwa class **ManusiaMilenial** akan mewarisi karakteristik dari class **Manusia**

```
22 //awal inheritance
23 class ManusalMilenial extends Manusia {
24     String email = '';
25
26     ManusalMilenial(String email) : super(email);
27     void info() {
28         print('nama: $nama, Email:$email ');
29     }
30 } //akhir inheritance
31
```

6. Encapsulation

Review materi mata kuliah Metode Perancangan Program, Enkapsulasi atau pembungkusan berfungsi untuk melindungi suatu objek dari dunia luar, sehingga seseorang tidak akan mampu merusak objek yang terbungkus. Objek yang terbungkus dalam suatu kelas baik data maupun fungsinya tidak bisa terlihat apalagi dirubah pada saat objek digunakan. Struktur class yang dimaksud adalah property dan method. Dengan enkapsulasi, kita bisa membuat pembatasan akses kepada property dan method, sehingga hanya property dan method tertentu saja yang bisa diakses dari luar class. Proses enkapsulasi diterapkan dengan menggunakan 3 jenis hak akses: **Public, Protected dan Private**. **Case study** dari inheritance akan dikembangkan implementasi enkapsulasi dengan bahasa pemrograman Dart.

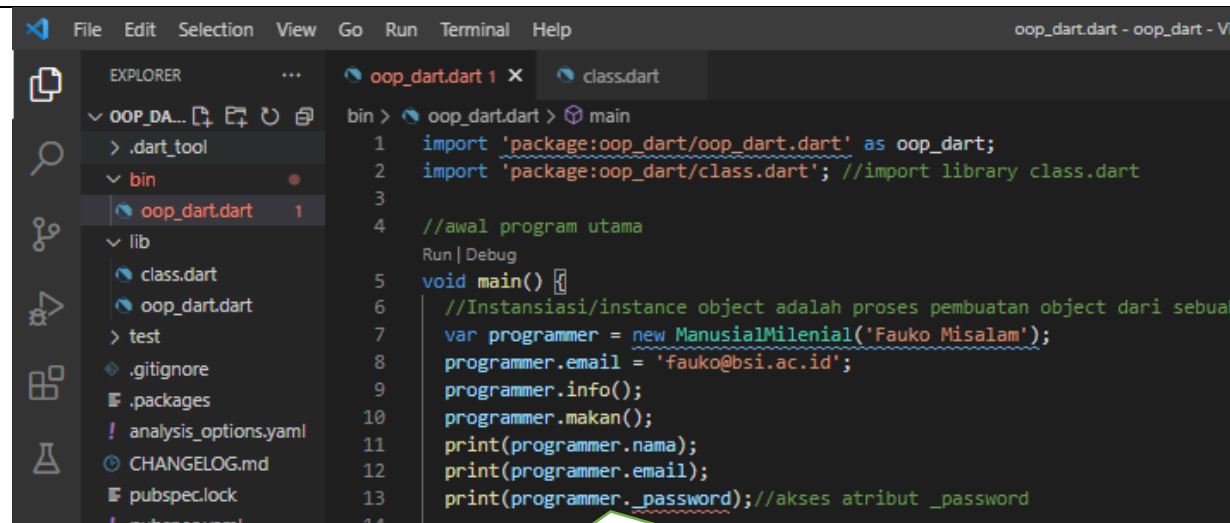
Source	Output
Setiap class memiliki berbagai atribut atau properti sesuai dengan kebutuhan dalam membangun suatu sistem. Ada kalanya atribut tersebut tidak diperbolehkan untuk diakses oleh siapapun karena bersifat private. Dari contoh yang sudah dipraktikan atribut nama pada class Manusia bersifat public, begitu juga atribut email dari class ManusiaMilenial bersifat public, artinya baik nama dan email dapat diakses oleh objek lainnya. Case study : akan kita tambahkan atribut password di kelas class ManusiaMilenial . Dalam kehidupan sehari-hari, tentunya password bersifat rahasia atau private. Berikut implementasi enkapsulasi .	
<pre>17 //awal inheritance 18 class ManusiaMilenial extends Manusia { 19 String email = ''; 20 String <u>password</u> = ''; 21 22 ManusiaMilenial(String email) : super(email); 23 void info() { 24 print('nama: \$nama, Email:\$email'); 25 } 26 } //akhir inheritance</pre>	u/ menambahkan atribut bersifat private dengan menuliskan <u>password</u> . Diawali dengan underscore
Untuk melakukan pengujian, akan dilakukan dua percobaan dengan memodifikasi sebagai berikut:	
1. pada oop_dart.dart lakukan penambahan scriptnya mengakses atribut nama dan emai di class.dart seperti pada gambar	



```
bin > oop_dart.dart > main
1  import 'package:oop_dart/oop_dart.dart' as oop_dart;
2  import 'package:oop_dart/class.dart'; //import library class.dart
3
4  //awal program utama
Run | Debug
5  void main() {
6      //Instansiasi/instance object adalah proses pembuatan object dari sebuah class
7      var programmer = new ManusalMilenial('Fauko Misalam');
8      programmer.email = 'fauko@bsi.ac.id';
9      programmer.info();
10     programmer.makan();
11     print(programmer.nama);
12     print(programmer.email);
13
14     print('\n'); //pindah baris
15
16     var dosen = new ManusalMilenial('Intan');
17     dosen.email = 'intan@bsi.ac.id';
18     dosen.info();
19     dosen.makan();
20
21     print('\n'); //pindah baris
22
23     var hacker = new ManusalMilenial('Dio');
24     hacker.email = 'dio@bsi.ac.id';
25     hacker.info();
26     hacker.makan();
27 }
```

Script tersebut object **programmer** mengakses atribut **nama**, **email** tidak terjadi error karena kedua atribut tersebut bersifat public

2. akan ditambahkan script pada **oop_dart** mengakses atribut `_password`, seperti pada gambar

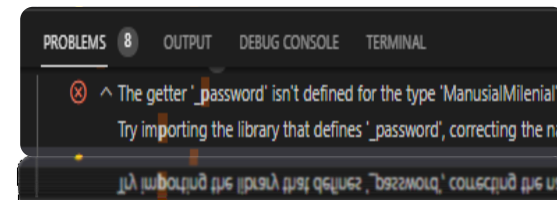


```
bin > oop_dart.dart > main
1 import 'package:oop_dart/oop_dart.dart' as oop_dart;
2 import 'package:oop_dart/class.dart'; //import library class.dart
3
4 //awal program utama
5 Run | Debug
6 void main() {
7     //Instansiasi/instance object adalah proses pembuatan object dari sebuah
8     var programmer = new ManusalMilenial('Fauko Misalam');
9     programmer.email = 'fauko@bsi.ac.id';
10    programmer.info();
11    programmer.makan();
12    print(programmer.nama);
13    print(programmer.email);
14    print(programmer._password); //akses atribut _password
```

Dengan penambahan scriptnya akses atribut `_password` terjadi error
(perhatikan gambar disamping)

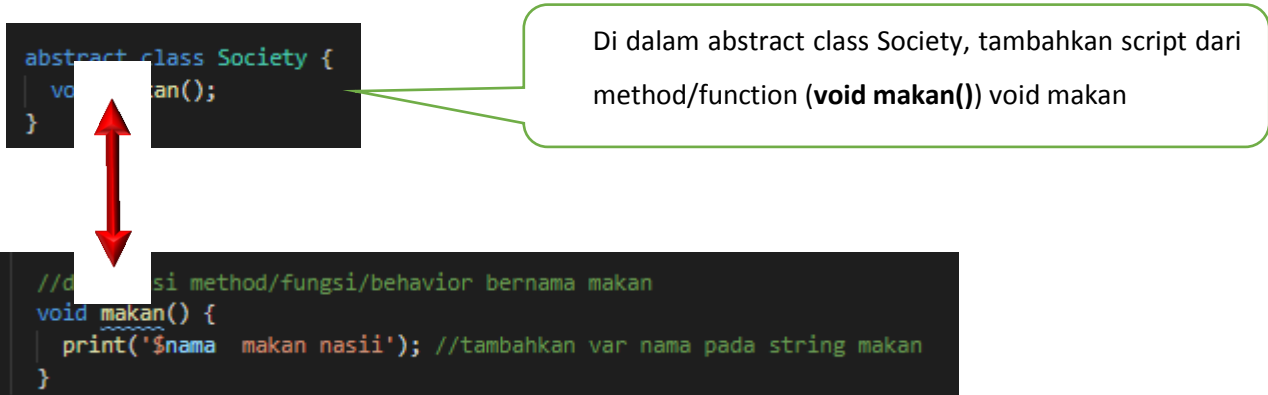
Kesimpulannya karena `_password` bersifat private (hanya bisa
diakses di dalam class **Manusia**

Dan tidak bisa diakses diluar kelas



7. abstraction/ abstraksi

Review mata kuliah Metode Perancangan Program bahwa Abstraksi adalah proses menyembunyikan kerumitan dari suatu proses untuk permasalahan yang dihadapi, atau dengan kata lain kelas abstrak adalah kelas yang tidak dapat dibuat objeknya (di instans), tujuan dari kelas abstrak adalah sebagai superclass (kelas awal) untuk kelas lainnya.

Source code	Output
<p>Untuk menerapkan abstraction pada bahasa pemrograman dart, <i>case study</i> menggunakan program sebelumnya dengan melakukan penyesuaian melalui file class/dart dengan tahapan sebagai berikut:</p> <p>1. Deklarasi abstract class</p> <div><pre>abstract class Society { void makan(); }</pre><p>Di dalam abstract class Society, tambahkan script dari method/function (void makan()) void makan</p><pre>//deklasi method/fungsi/behavior bernama makan void makan() { print('\$nama makan nasii'); //tambahkan var nama pada string makan }</pre></div>	

```
class Manusia {  
    //deklarasi atribut/properti  
    String nama = ''; //atribut nama bertipe string dan nilai awal null  
  
    //pembuatan konstruktor, nama konstruktor adalah nama class  
    Manusia(nm) {  
        //nm a/ sebuah parameter yg dipergunakan untuk menampung data  
        this.nama = nm; //atribut nama memiliki data dari parameter nm  
    }  
  
    //deklarasi method/fungsi/behavior bernama makan  
    void makan() {  
        print('$nama makan nasii'); //tambahkan var nama pada string makan  
    }  
}
```

```
class Manusia extends Society {  
    //deklarasi atribut/properti
```

Lakukan perubahan **class Manusia**
Menjadi
Class Manusia extends Society

2. Hasil penyesuaian keseluruhan dari **abstract class Society**

```
lib > class.dart > ManuialMilenial
1
2 abstract class Society {
3   void makan();
4 }
5
6
7
8 class Manusia extends Society {
9   //deklarasi atribut/properti
10  String nama = ''; //atribut nama bertipe string dan nilai awal null
11
12  //pembuatan konstruktor, nama konstruktor adalah nama class
13  Manusia(nm) {
14    //nm a/ sebuah parameter yg dipergunakan untuk menampung data
15    this.nama = nm; //atribut nama memiliki data dari parameter nm
16  }
17
18  //deklarasi method/fungsi/behavior bernama makan
19  void makan() {
20    print('$nama makan nasii'); //tambahkan var nama pada string makan
21  }
22 }
23
24 //awal inheritance
25 class ManuialMilenial extends Manusia {
26   String email = '';
27   String _password = '';
28
29   ManuialMilenial(String email) : super(email);
30
31   void info() {
32     print('nama: $nama, Email:$email');
33   }
34 } //akhir inheritance
```

3. Tahap akhir **Run**

8. Polymorphism/Polimorfisme

Merujuk pada materi Metode Perancangan Program, Polimorfisme dapat diartikan sebagai kemampuan suatu bahasa pemrograman untuk memiliki fungsi-fungsi atau metode yang bernama sama tetapi berbeda dalam parameter dan implementasi kodenya (*overloading*). Jadi fokus dari Polymorphism/Polimorfisme adalah membuat kelas turunan dapat menggunakan fungsi yang ada pada kelas pewarisnya dan dapat mengimplementasikan kode yang berbeda dari fungsi pewarisnya ini dinamakan **overriding**.

1. Dari source latihan sebelumnya, terdapat source code seperti pada gambar

```
//awal inheritance
class ManusiaMilenial extends Manusia {
    String email = '';
    String password = '';

    ManusiaMilenial(String email) : super(email);

    void info() {
        print('namaku: $nama, Email:$email');
    }
} //akhir inheritance
```

print('namaku: \$nama, Email:\$email');

script tersebut terdapat pada **class ManusiaMilenial**. Script tersebut yang akan dijadikan case study ke dalam **Polymorphism/Polimorfisme**

2. Konsep yang akan diimplementasikan membuat 2 class baru yang berbeda:

- a. Membuat sebuah class Programmer yang mewarisi class **ManusiaMilenial**

```
class Programmer extends ManusiaMilenial {
    Programmer(String email) : super(email);

    @override
    void info() {
        print('$email pemiliknya adalah $nama');
    }
}
```

script yang ada di class **ManusiaMilenial**

```
void info() {
    print('namaku: $nama, Email:$email');
}
```

sudah dilakukan perubahan pada Script class **Programmer** memanfaatkan fungsi **Polymorphism/Polimorfisme**

```
//awal inheritance
class ManusiaMilenial extends Manusia {
    String email = '';
    String password = '';

    ManusiaMilenial(String email) : super(email);

    void info() {
        print('namaku: $nama, Email:$email');
    }
} //akhir inheritance
```

- b. Membuat sebuah class Dokter yang mewarisi class **ManusiaMilenial**

```
class Dosen extends ManusiaMilenial {  
    Dosen(String nama) : super(nama);  
  
    @Override  
    void info() {  
        print('$nama, telah memiliki Email= $email')  
    }  
}
```

Kesimpulannya

class Dosen yang telah mewarisi (**extends**) dari class ManusiaMilenial dapat melakukan modifikasi pada method info()

```
@Override  
void info() {  
    print('$nama, telah memiliki Email= $email');  
}
```

```
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL  
Email: fauko@bsi.ac.id pemiliknya adalah Fauko Misalam  
Fauko Misalam makan nasii  
  
Intan, telah memiliki Email= intan@bsi.ac.id  
Intan makan nasii  
Exited
```

Output

Script keseluruhan pada

a. class.dart

```
class Manusia {  
    //deklarasi atribut/properti  
    String nama = ''; //atribut nama bertipe string dan nilai awal null  
  
    //pembuatan konstruktor, nama konstruktor adalah nama class  
    Manusia(nm) {  
        //nm a/ sebuah parameter yg dipergunakan untuk menampung data  
        this.nama = nm; //atribut nama memiliki data dari parameter nm  
    }  
  
    //deklarasi method/fungsi/behavior bernama makan  
    void makan() {  
        print('$nama makan nasii'); //tambahkan var nama pada string makan  
    }  
}  
  
//awal inheritance  
class ManusalMilenial extends Manusia {  
    String email = '';  
    String _password = '';  
  
    ManusalMilenial(String email) : super(email);  
  
    void info() {  
        print('namaku: $nama, Email:$email');  
    }  
} //akhir inheritance
```

```
class Programmer extends ManusalMilenial {
    Programmer(String email) : super(email);

    @override
    void info() {
        print('Email: $email pemiliknya adalah $nama');
    }
}

class Dosen extends ManusalMilenial {
    Dosen(String nama) : super(nama);

    @override
    void info() {
        print('$nama, telah memiliki Email= $email');
    }
}
```

b.oop_dart.dart

```
oop_dart import 'package:oop_dart/oop_dart.dart' as oop_dart;

import 'package:oop_dart/class.dart'; //import library class.dart

//awal program utama
void main() {
    //Instansiasi/instance object adalah proses pembuatan object dari sebuah class
    var programmer = new Programmer('Fauko Misalam');
    programmer.email = 'fauko@bsi.ac.id';
    programmer.info();
    programmer.makan();

    print('\n'); //pindah baris

    var dosen = new Dosen('Intan');
    dosen.email = 'intan@bsi.ac.id';
    dosen.info();
    dosen.makan();
}
```