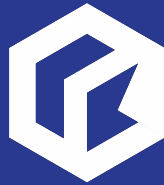


Dart Object Oriented Programming

By Saiful Bahri

<https://linkedin.com/in/bahrie>



CODE with BAHRI

Profil Pemateri

Experiences:

- 2012 : Software Engineer at Telkomsel (Tcash, LinkAja)
- 2013: Senior Software Engineer at PT Hitachi Ebworx
- 2015: Tech Lead at Go-Jek, GOTIX
- 2017 Principal Engineer at PT Atmatech Global Teknologi
- 2019 Tech Lead at Sampingan.id
- 2021-2023 Tech Leat at Woo-hoo.org
- 2023 - Now Flutter Trainer & Founder Jagoflutter.com

Social Media:

- Instagram: <https://instagram.com/codewithbahri>
- Youtube : <https://youtube.com/@codewithbahri>
- Linkedin: <https://linkedin.com/in/bahrie>
- Github: <https://github.com/bahrie127>



<https://instagram.com/codewithbahri>

OOP (Object-oriented programming)

OOP (Object-oriented programming) adalah metode pemrograman yang menggunakan Object dan interaksinya untuk merancang dan membuat aplikasi. OOP digunakan pada banyak bahasa pemrograman, seperti Dart, Java, C++, Python, dan lain-lain.

OOP (Object-oriented programming)

Keuntungan dari OOP antara lain mudah dimengerti dan digunakan, meningkatkan daya guna ulang dan mengurangi kompleksitas, meningkatkan produktivitas programmer, membuat kode lebih mudah dipelihara, dimodifikasi, dan didebug, serta mendorong kerjasama dan kolaborasi.

OOP (Object-oriented programming)

Fitur-fitur OOP antara lain class, object, enkapsulasi, pewarisan(inheritance), polimorfisme, dan abstraksi. OOP bertujuan untuk memecahkan masalah yang kompleks menjadi Object yang lebih kecil.

OOP (Object-oriented programming)

OOP dapat membuat kode menjadi lebih modular, fleksibel, dan mudah diperluas. OOP dapat membantu Anda memahami dan menyelesaikan masalah dengan lebih baik.

Class In Dart

Class dalam bahasa pemrograman Dart merupakan salah satu konsep dasar dalam paradigma pemrograman berorientasi Object (OOP). Konsep ini memungkinkan programmer untuk membuat Object yang memiliki properti dan method tertentu yang dapat digunakan untuk melakukan tugas tertentu.

Class In Dart

Sintaks dasar class dalam Dart dimulai dengan kata kunci "class" diikuti dengan nama class. Struktur class dalam Dart terdiri dari bagian header, body, dan constructor. Header class berisi nama class dan deklarasi pewarisan (inheritance), sedangkan body class berisi properti dan method yang didefinisikan untuk class tersebut. Constructor adalah method khusus yang digunakan untuk membuat Object class tersebut.

Class In Dart

Pembuatan Object dalam Dart dilakukan dengan menggunakan operator "=" diikuti dengan nama class dan parameter Constructor. Constructor memiliki dua jenis, yaitu Default Constructor dan Constructor dengan parameter. Default Constructor merupakan Constructor yang otomatis dibuat jika tidak didefinisikan, sedangkan Constructor dengan parameter digunakan untuk memasukkan nilai ke dalam Object.

Class In Dart

Setiap Object yang dibuat dari class memiliki properti yang unik. Properti class dibagi menjadi dua jenis, yaitu variabel instance dan variabel statis. Variabel instance adalah variabel yang nilainya berbeda untuk setiap Object, sedangkan variabel statis nilainya sama untuk semua Object.

Class In Dart


Method adalah fungsi khusus yang dapat digunakan untuk melakukan operasi pada Object. method juga dibagi menjadi dua jenis, yaitu method instance dan method statis. method instance dipanggil pada Object yang spesifik, sedangkan method statis dipanggil pada class itu sendiri.

Class In Dart

Dalam pewarisan(Inheritance), class dapat mewarisi properti dan method dari class lain. Dalam Dart, konsep pewarisan ini dapat diimplementasikan dengan menggunakan kata kunci "extends". Dalam implementasinya, class turunan dapat mengakses properti dan method dari class induknya.

Class In Dart

Untuk mendefinisikan class dalam Dart, kita menggunakan sintaks berikut:



```
class ClassName {  
    // variabel  
    // metode  
}
```

Class In Dart

Contoh sederhana class dalam Dart:

Dalam contoh di samping, kita mendefinisikan class Mobil dengan tiga variabel (merk, model, dan tahun) dan satu method (klakson).



```
class Mobil {  
  String? merk;  
  String? model;  
  int? tahun;  
  
  void klakson() {  
    print("Beep!  
Beep!");  
  }  
}
```

Membuat Object dari Class

Untuk membuat Object dari class, kita menggunakan sintaks berikut:



```
ClassName objek = ClassName();
```

Membuat Object dari Class

Untuk membuat Object dari class, kita menggunakan sintaks berikut:

```
ClassName objek = ClassName();
```

Contoh membuat Object dari class Mobil:

```
void main() {  
    Mobil mobil1 = Mobil();  
}
```


Membuat Object dari Class

Setelah Object dibuat, kita dapat mengakses variabel dan method dalam Object dengan menggunakan tanda titik (.):

```
void main() {  
    Mobil mobil1 = Mobil();  
  
    mobil1.merk = "Toyota";  
    mobil1.model = "Corolla";  
    mobil1.tahun = 2020;  
  
    print(mobil1.merk); // Output: Toyota  
    print(mobil1.model); // Output: Corolla  
    print(mobil1.tahun); // Output: 2020  
  
    mobil1.klakson(); // Output: Beep! Beep!  
}
```

Class In Dart

Dalam Dart, kita bisa menggunakan Constructor untuk membuat Object dengan nilai yang telah ditentukan. Constructor adalah method khusus yang dipanggil saat membuat Object. Berikut adalah sintaks untuk membuat Constructor dalam class:



```
ClassName(parameter) {  
  // kode konstruktor  
}
```

Class In Dart

Contoh penggunaan Constructor dalam class Mobil:

```
class Mobil {  
  String merk;  
  String model;  
  int tahun;  
  
  Mobil(String merk, String model, int tahun) {  
    this.merk = merk;  
    this.model = model;  
    this.tahun = tahun;  
  }  
  
  void klakson() {  
    print("Beep! Beep!");  
  }  
}  
  
void main() {  
  Mobil mobil2 = Mobil("Honda", "Civic", 2019);  
  
  print(mobil2.merk); // Output: Honda  
  print(mobil2.model); // Output: Civic  
  print(mobil2.tahun); // Output: 2019  
}
```

Class In Dart

Kesimpulan

Class adalah elemen dasar dalam pemrograman berorientasi Object yang membantu kita mengorganisir kode dengan lebih baik. Dalam pemrograman Dart, kita dapat mendefinisikan class, membuat Object dari class, dan mengakses variabel serta method dalam Object.

Object adalah instansi dari class yang digunakan untuk mengakses variabel dan method dalam class. Dengan memahami cara kerja Object, kita dapat lebih mudah mengorganisir kode dan mengembangkan aplikasi secara efisien.

Contoh Class & Object dalam Pemrograman Dart

Contoh 1: class Hewan

Dalam contoh disamping ini, terdapat class Hewan dengan tiga properti: nama, jumlahKaki, dan umur. class ini juga memiliki method display() yang mencetak nilai dari ketiga properti tersebut.

```
class Hewan {  
  String? nama;  
  int? jumlahKaki;  
  int? umur;  
  
  void display() {  
    print("Nama Hewan: $nama.");  
    print("Jumlah Kaki: $jumlahKaki.");  
    print("Umur: $umur.");  
  }  
}  
  
void main() {  
  Hewan hewan = Hewan();  
  hewan.nama = "Singa";  
  hewan.jumlahKaki = 4;  
  hewan.umur = 10;  
  hewan.display();  
}
```

Contoh Class & Object dalam Pemrograman Dart

Contoh 2: Menghitung Luas Persegi Panjang Menggunakan Class dan Object

Dalam contoh di bawah ini, terdapat class PersegiPanjang dengan dua properti: panjang dan lebar. class ini juga memiliki method area() yang menghitung luas dari persegi panjang.

Catatan: Tanda seru (!) digunakan untuk memberitahu kompiler bahwa variabel tersebut tidak null. Jika Anda tidak menggunakan tanda seru, maka Anda akan mendapatkan error. Anda akan mempelajari lebih lanjut tentang null safety nanti.

```
class PersegiPanjang {  
  double? panjang;  
  double? lebar;  
  
  double area() {  
    return panjang! * lebar!;  
  }  
}  
  
void main() {  
  PersegiPanjang persegiPanjang = PersegiPanjang();  
  persegiPanjang.panjang = 10;  
  persegiPanjang.lebar = 5;  
  print("Luas persegi panjang adalah ${persegiPanjang.area()}.");  
}
```

Contoh Class & Object dalam Pemrograman Dart

Contoh 3: Menghitung Bunga Sederhana Menggunakan class dan Object

Dalam contoh di bawah ini, terdapat class BungaSederhana dengan tiga properti: pokok, sukuBunga, dan waktu. class ini juga memiliki method bunga() yang menghitung bunga sederhana.

```
class BungaSederhana {  
  double? pokok;  
  double? sukuBunga;  
  double? waktu;  
  
  double bunga() {  
    return (pokok! * sukuBunga! * waktu!) / 100;  
  }  
}  
  
void main() {  
  BungaSederhana bungaSederhana = BungaSederhana();  
  bungaSederhana.pokok = 1000;  
  bungaSederhana.sukuBunga = 10;  
  bungaSederhana.waktu = 2;  
  print("Bunga Sederhana adalah ${bungaSederhana.bunga()}");  
}
```

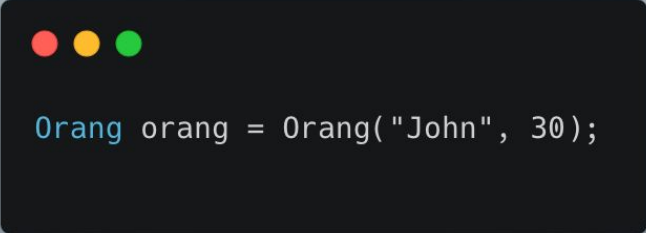
Contoh Class & Object dalam Pemrograman Dart

Challenge:

Buatlah class Rumah dengan properti nama, alamat, jumlahKamar. Buatlah method display() yang mencetak nilai dari ketiga properti tersebut. Buat Object dari class Rumah dan atur nilai-nilai propertinya. Panggil method display() untuk mencetak nilai dari ketiga properti tersebut.

Constructor dalam Pemrograman Dart

Constructor adalah method khusus yang digunakan untuk menginisialisasi Object. Constructor dipanggil secara otomatis saat Object dibuat dan dapat digunakan untuk mengatur nilai awal properti Object. Sebagai contoh, kode berikut membuat Object class Orang dan mengatur nilai awal untuk properti nama dan umur.

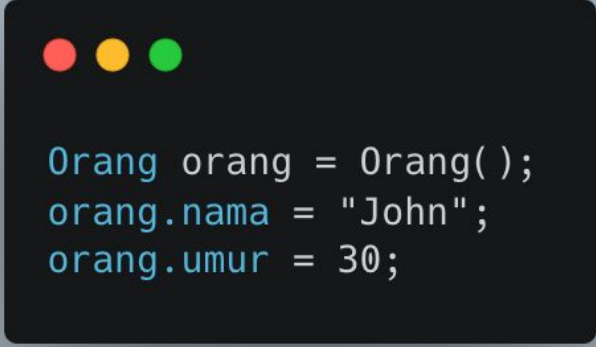


```
Orang orang = Orang("John", 30);
```

Constructor dalam Pemrograman Dart

Tanpa Constructor

Jika Anda tidak mendefinisikan Constructor untuk class, maka Anda perlu mengatur nilai properti secara manual. Sebagai contoh, kode berikut membuat Object class Orang dan mengatur nilai untuk properti nama dan umur.



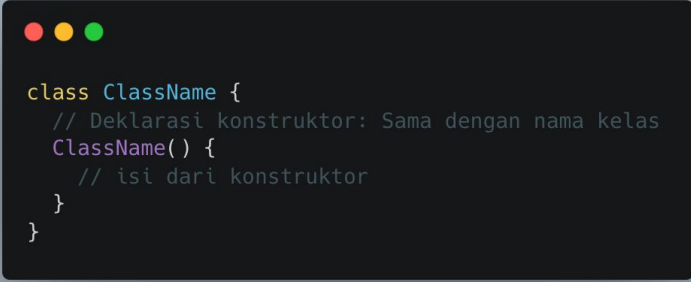
```
Orang orang = Orang();  
orang.nama = "John";  
orang.umur = 30;
```

Constructor dalam Pemrograman Dart

Hal-hal yang Perlu Diperhatikan

- Nama Constructor harus sama dengan nama class.
- Constructor tidak memiliki tipe pengembalian.

Sintaks:



```
class ClassName {  
    // Deklarasi konstruktor: Sama dengan nama kelas  
    ClassName() {  
        // isi dari konstruktor  
    }  
}
```

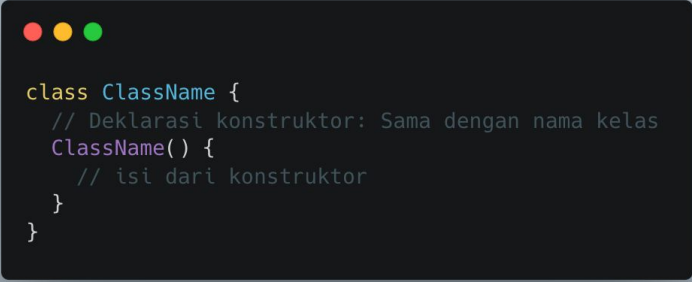
Constructor dalam Pemrograman Dart

Hal-hal yang Perlu Diperhatikan

- Nama Constructor harus sama dengan nama class.
- Constructor tidak memiliki tipe pengembalian.

Catatan: Ketika Anda membuat Object dari class, Constructor dipanggil secara otomatis. Constructor digunakan untuk menginisialisasi nilai saat Object dibuat.

Sintaks:



```
class ClassName {  
    // Deklarasi konstruktor: Sama dengan nama kelas  
    ClassName() {  
        // isi dari konstruktor  
    }  
}
```

Constructor dalam Pemrograman Dart

Contoh 1: Cara Mendeklarasikan Constructor dalam Dart

Dalam contoh di bawah ini, terdapat class Siswa dengan tiga properti: nama, umur, dan nomorInduk. class ini memiliki satu Constructor. Constructor digunakan untuk menginisialisasi nilai dari ketiga properti tersebut. Kami juga membuat Object dari class Siswa yang disebut siswa.

```
class Siswa {  
  String? nama;  
  int? umur;  
  int? nomorInduk;  
  
  // Konstruktor  
  Siswa(String nama, int umur, int nomorInduk) {  
    print(  
      "Konstruktor dipanggil"); // ini digunakan untuk memeriksa apakah konstruktor dipanggil atau  
      tidak.  
    this.nama = nama;  
    this.umur = umur;  
    this.nomorInduk = nomorInduk;  
  }  
}  
  
void main() {  
  // Di sini siswa adalah objek dari kelas Siswa.  
  Siswa siswa = Siswa("John", 20, 1);  
  print("Nama: ${siswa.nama}");  
  print("Umur: ${siswa.umur}");  
  print("Nomor Induk: ${siswa.nomorInduk}");  
}
```

Constructor dalam Pemrograman Dart

Catatan: Kata kunci `this` digunakan untuk merujuk ke instance saat ini dari class. Kata kunci ini digunakan untuk mengakses properti class saat ini. Dalam contoh di atas, nama parameter dan properti class dari Constructor Siswa adalah sama. Oleh karena itu, untuk menghindari kebingungan, kita menggunakan kata kunci `this`.

```
class Siswa {  
  String? nama;  
  int? umur;  
  int? nomorInduk;  
  
  // Konstruktor  
  Siswa(String nama, int umur, int nomorInduk) {  
    print(  
      "Konstruktor dipanggil"); // ini digunakan untuk memeriksa apakah konstruktor dipanggil atau  
      tidak.  
    this.nama = nama;  
    this.umur = umur;  
    this.nomorInduk = nomorInduk;  
  }  
}  
  
void main() {  
  // Di sini siswa adalah objek dari kelas Siswa.  
  Siswa siswa = Siswa("John", 20, 1);  
  print("Nama: ${siswa.nama}");  
  print("Umur: ${siswa.umur}");  
  print("Nomor Induk: ${siswa.nomorInduk}");  
}
```

Constructor dalam Pemrograman Dart

Contoh 2: Constructor dalam Dart

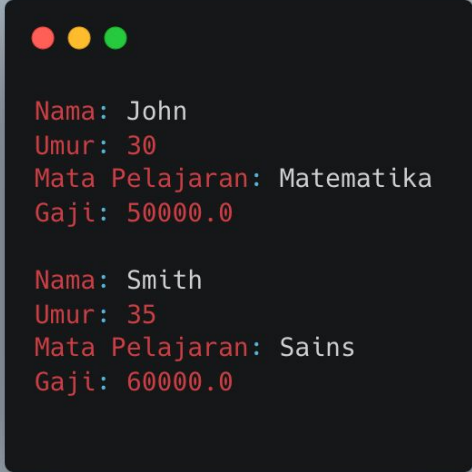
Dalam contoh berikut, ada class Guru dengan empat properti: nama, umur, mataPelajaran, dan gaji. class ini memiliki satu Constructor untuk menginisialisasi nilai dari properti. class ini juga memiliki method display() yang digunakan untuk menampilkan nilai properti. Kami juga membuat 2 Object dari class Guru yang disebut guru1 dan guru2.

```
class Guru {  
  String? nama;  
  int? umur;  
  String? mataPelajaran;  
  double? gaji;  
  
  // Konstruktor  
  Guru(String nama, int umur, String mataPelajaran, double gaji) {  
    this.nama = nama;  
    this.umur = umur;  
    this.mataPelajaran = mataPelajaran;  
    this.gaji = gaji;  
  }  
  
  // Metode  
  void tampilkan() {  
    print("Nama: ${this.nama}");  
    print("Umur: ${this.umur}");  
    print("Mata Pelajaran: ${this.mataPelajaran}");  
    print("Gaji: ${this.gaji}\n"); // \n digunakan untuk baris baru  
  }  
}  
  
void main() {  
  // Membuat objek guru1 dari kelas Guru  
  Guru guru1 = Guru("John", 30, "Matematika", 50000.0);  
  guru1.tampilkan();  
  
  // Membuat objek guru2 dari kelas Guru  
  Guru guru2 = Guru("Smith", 35, "Sains", 60000.0);  
  guru2.tampilkan();  
}
```

Constructor dalam Pemrograman Dart

Contoh 2: Constructor dalam Dart

Hasil Keluaran:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It displays two sets of output from Dart constructors, each on a new line.

```
Nama: John  
Umur: 30  
Mata Pelajaran: Matematika  
Gaji: 50000.0  
  
Nama: Smith  
Umur: 35  
Mata Pelajaran: Sains  
Gaji: 60000.0
```


Default Constructor dalam Bahasa Pemrograman Dart

Default Constructor adalah Constructor yang dibuat secara otomatis oleh kompiler Dart jika Anda tidak membuat Constructor. Default Constructor tidak memiliki parameter. Default Constructor dideklarasikan dengan menggunakan nama class diikuti dengan tanda kurung ().

Default Constructor dalam Bahasa Pemrograman Dart

Contoh 1: Default Constructor dalam Dart

Dalam contoh berikut ini, terdapat class Laptop dengan dua properti: merek, dan harga. Mari kita buat Constructor tanpa parameter dan mencetak sesuatu dari Constructor tersebut. Kami juga memiliki Object dari class Laptop yang disebut laptop.

```
class Laptop {  
  String? merek;  
  int? harga;  
  
  // Konstruktor  
  Laptop() {  
    print("Ini adalah konstruktor default");  
  }  
}  
  
void main() {  
  // Di sini laptop adalah objek dari kelas Laptop.  
  Laptop laptop = Laptop();  
}
```

Parameterized Constructor dalam Bahasa Pemrograman Dart

Parameterized Constructor digunakan untuk menginisialisasi variabel instan dari class.

Parameterized Constructor adalah Constructor yang menggunakan parameter. Constructor ini digunakan untuk mengirimkan nilai ke Constructor pada saat pembuatan Object.



```
class NamaKelas {  
    // Variabel Instan  
    int? angka;  
    String? nama;  
    // Konstruktor Parameter  
    NamaKelas(this.angka, this.nama);  
}
```

Parameterized Constructor dalam Bahasa Pemrograman Dart


Contoh 1: Parameterized Constructor dalam Dart
Dalam contoh di bawah ini, terdapat class Student dengan tiga properti: name, age, dan rollNumber. class ini memiliki satu Constructor. Constructor ini digunakan untuk menginisialisasi nilai dari ketiga properti tersebut. Kami juga memiliki Object dari class Student yang disebut student.

```
class Student {  
  String? name;  
  int? age;  
  int? rollNumber;  
  // Konstruktor  
  Student(this.name, this.age, this.rollNumber);  
}  
  
void main(){  
  // Di sini student adalah objek dari kelas Student.  
  Student student = Student("John", 20, 1);  
  print("Name: ${student.name}");  
  print("Age: ${student.age}");  
  print("Roll Number: ${student.rollNumber}");  
}
```

Parameterized Constructor dalam Bahasa Pemrograman Dart

Contoh 1: Parameterized Constructor dalam Dart

Hasil Keluaran:



```
Name: John  
Age: 20  
Roll Number: 1
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. The text inside the terminal is displayed in a red monospace font.

Parameterized Constructor dalam Bahasa Pemrograman Dart

Contoh 2: Parameterized Constructor dengan Parameter Bernama dalam Dart


Dalam contoh di bawah ini, terdapat class Student dengan tiga properti: name, age, dan rollNumber. class ini memiliki satu Constructor. Constructor ini digunakan untuk menginisialisasi nilai dari ketiga properti tersebut. Kami juga memiliki Object dari class Student yang disebut student.

```
class Student {  
  String? name;  
  int? age;  
  int? rollNumber;  
  
  // Konstruktor  
  Student({String? name, int? age, int? rollNumber}) {  
    this.name = name;  
    this.age = age;  
    this.rollNumber = rollNumber;  
  }  
}  
  
void main(){  
  // Di sini student adalah objek dari kelas Student.  
  Student student = Student(name: "John", age: 20, rollNumber: 1);  
  print("Name: ${student.name}");  
  print("Age: ${student.age}");  
  print("Roll Number: ${student.rollNumber}");  
}
```

Parameterized Constructor dalam Bahasa Pemrograman Dart

Contoh 2: Parameterized Constructor dengan Parameter Bernama dalam Dart

Hasil Keluaran:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It displays the output of a Dart program: 'Name: John', 'Age: 20', and 'Roll Number: 1'.

```
Name: John  
Age: 20  
Roll Number: 1
```

Parameterized Constructor dalam Bahasa Pemrograman Dart

Contoh 3: Parameterized Constructor dengan Nilai Default dalam Dart

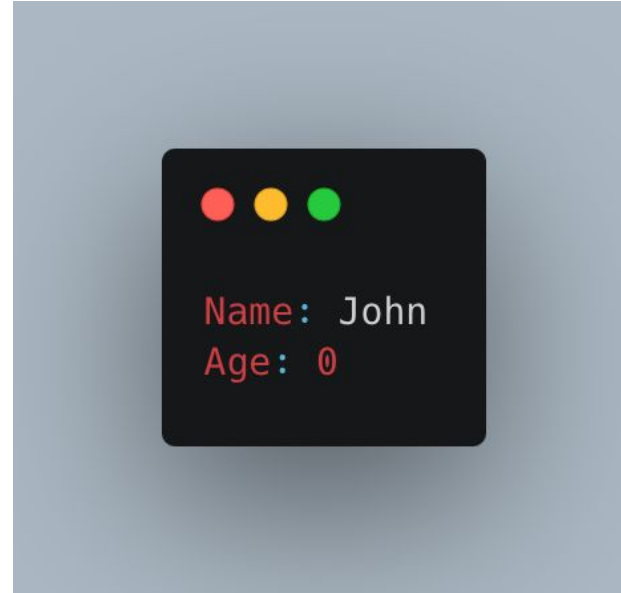
Dalam contoh di bawah ini, terdapat class Student dengan dua properti: name, dan age. class ini memiliki Parameterized Constructor dengan nilai default. Constructor ini digunakan untuk menginisialisasi nilai dari kedua properti tersebut. Kami juga memiliki Object dari class Student yang disebut student.

```
class Student {  
  String? name;  
  int? age;  
  
  // Konstruktor  
  Student({String? name = "John", int? age = 0}) {  
    this.name = name;  
    this.age = age;  
  }  
}  
  
void main(){  
  // Di sini student adalah objek dari kelas Student.  
  Student student = Student();  
  print("Name:${student.name}");  
  print("Age: ${student.age}");  
}
```


Parameterized Constructor dalam Bahasa Pemrograman Dart

Contoh 3: Parameterized Constructor dengan Nilai Default dalam Dart

Hasil Keluaran:



Named Constructor dalam Dart

Dalam kebanyakan bahasa pemrograman seperti Java, C++, C#, dll., kita bisa membuat beberapa Constructor dengan nama yang sama. Namun dalam Dart, hal ini tidak mungkin. Tetapi ada cara lain, kita bisa membuat beberapa Constructor dengan nama yang sama menggunakan Named Constructor.

Catatan: Named Constructor meningkatkan keterbacaan kode. Ini berguna ketika Anda ingin membuat beberapa Constructor dengan nama yang sama.

Named Constructor dalam Dart

Contoh 1: Named Constructor dalam Dart

Dalam contoh di bawah ini, ada class Student dengan tiga properti: name, age, dan rollNumber. class ini memiliki dua Constructor. Constructor pertama adalah Default Constructor. Constructor kedua adalah Named Constructor. Named Constructor digunakan untuk menginisialisasi nilai ketiga properti. Kami juga memiliki Object dari class Student yang disebut student.

```
class Student {  
  String? name;  
  int? age;  
  int? rollNumber;  
  
  // Default Constructor  
  Student() {  
    print("This is a default constructor");  
  }  
  
  // Named Constructor  
  Student.namedConstructor(String name, int age, int rollNumber) {  
    this.name = name;  
    this.age = age;  
    this.rollNumber = rollNumber;  
  }  
}  
  
void main() {  
  // Here student is object of class Student.  
  Student student = Student.namedConstructor("John", 20, 1);  
  print("Name: ${student.name}");  
  print("Age: ${student.age}");  
  print("Roll Number: ${student.rollNumber}");  
}
```

Named Constructor dalam Dart

Contoh 2: Named Constructor dalam Dart

Dalam contoh di bawah ini, ada class Mobile dengan tiga properti name, color, dan prize. Class ini memiliki satu method display yang mencetak nilai ketiga properti. Kami juga memiliki Object dari class Mobile yang disebut mobile. Ada juga Constructor Mobile yang mengambil ketiga properti sebagai parameter. Named Constructor Mobile.namedConstructor digunakan untuk membuat Object dari class Mobile dengan name, color dan prize opsional. Nilai default dari prize adalah 0. Jika prize tidak dilewatkan, maka nilai default akan digunakan.

```
class Mobile {  
  String? name;  
  String? color;  
  int? prize;  
  
  Mobile(this.name, this.color, this.prize);  
  // here Mobile() is a named constructor  
  Mobile.namedConstructor(this.name, this.color, [this.prize = 0]);  
  
  void displayMobileDetails() {  
    print("Mobile name: $name.");  
    print("Mobile color: $color.");  
    print("Mobile prize: $prize");  
  }  
}  
  
void main() {  
  var mobile1 = Mobile("Samsung", "Black", 20000);  
  mobile1.displayMobileDetails();  
  var mobile2 = Mobile.namedConstructor("Apple", "White");  
  mobile2.displayMobileDetails();  
}
```

Named Constructor dalam Dart

Contoh 3: Named Constructor dalam Dart

Dalam contoh di bawah ini, ada class Animal dengan dua properti name dan age. class ini memiliki tiga Constructor. Constructor pertama adalah Default Constructor. Constructor kedua dan ketiga adalah Named Constructor. Constructor kedua digunakan untuk menginisialisasi nilai name dan age, dan Constructor ketiga digunakan untuk menginisialisasi nilai name saja. Kami juga memiliki Object dari class Animal yang disebut animal

```
class Animal {  
  String? name;  
  int? age;  
  
  // Default Constructor  
  Animal() {  
    print("This is a default constructor");  
  }  
  
  // Named Constructor  
  Animal.namedConstructor(String name, int age) {  
    this.name = name;  
    this.age = age;  
  }  
  
  // Named Constructor  
  Animal.namedConstructor2(String name) {  
    this.name = name;  
  }  
}  
  
void main() {  
  // Here animal is object of class Animal.  
  Animal animal = Animal.namedConstructor("Dog", 5);  
  print("Name: ${animal.name}");  
  print("Age: ${animal.age}");  
  
  Animal animal2 = Animal.namedConstructor2("Cat");  
  print("Name: ${animal2.name}");  
}
```

Named Constructor dalam Dart

Contoh 4: Contoh Nyata Named Constructor dalam Dart

Dalam contoh di bawah ini, ada class Person dengan dua properti name dan age. class ini memiliki tiga Constructor. Constructor pertama adalah Parameterized Constructorisasi yang mengambil dua parameter name dan age. Constructor kedua dan ketiga adalah Named Constructor. Constructor kedua fromJson digunakan untuk membuat Object dari class Person dari JSON. Constructor ketiga fromJsonString digunakan untuk membuat Object dari class Person dari string JSON. Kami juga memiliki Object dari class Person yang disebut person.

```
import 'dart:convert';

class Person {
  String? name;
  int? age;

  Person(this.name, this.age);

  Person.fromJson(Map<String, dynamic> json) {
    name = json['name'];
    age = json['age'];
  }

  Person.fromJsonString(String jsonString) {
    Map<String, dynamic> json = jsonDecode(jsonString);
    name = json['name'];
    age = json['age'];
  }
}

void main() {
  // Here person is object of class Person.
  String jsonString1 = '{"name": "Bishworaj", "age": 25}';
  String jsonString2 = '{"name": "John", "age": 30}';

  Person p1 = Person.fromJsonString(jsonString1);
  print("Person 1 name: ${p1.name}");
  print("Person 1 age: ${p1.age}");

  Person p2 = Person.fromJsonString(jsonString2);
  print("Person 2 name: ${p2.name}");
  print("Person 2 age: ${p2.age}");
}
```

Constant Constructor dalam Dart

Constant Constructor adalah Constructor yang menciptakan Object konstan. Object konstan adalah Object yang nilai tidak dapat diubah. Constant Constructor dideklarasikan dengan menggunakan kata kunci `const`.

Info

Catatan: Constant Constructor digunakan untuk membuat Object yang nilainya tidak dapat diubah. Ini meningkatkan kinerja program.

Constant Constructor dalam Dart

Aturan untuk Mendeklarasikan Constant Constructor dalam Dart:

- Semua properti class harus final.
- Tidak memiliki badan.
- Hanya class yang mengandung Constructor const diinisialisasi menggunakan kata kunci const.

Constant Constructor dalam Dart

Contoh 1: Constant Constructor dalam Dart

Dalam contoh di bawah ini, ada class Point dengan dua properti final: x dan y. class ini juga memiliki Constant Constructor yang menginisialisasi dua properti tersebut. class ini juga memiliki method display yang mencetak nilai dari dua properti tersebut.

```
class Point {  
    final int x;  
    final int y;  
  
    const Point(this.x, this.y);  
}  
  
void main() {  
    // p1 dan p2 memiliki kode hash yang sama.  
    Point p1 = const Point(1, 2);  
    print("Kode hash p1 adalah: ${p1.hashCode}");  
  
    Point p2 = const Point(1, 2);  
    print("Kode hash p2 adalah: ${p2.hashCode}");  
  
    // tanpa menggunakan const  
    // ini memiliki kode hash yang berbeda.  
    Point p3 = Point(2, 2);  
    print("Kode hash p3 adalah: ${p3.hashCode}");  
  
    Point p4 = Point(2, 2);  
    print("Kode hash p4 adalah: ${p4.hashCode}");  
}
```

Constant Constructor dalam Dart

Contoh 1: Constant Constructor dalam Dart

Dalam contoh di bawah ini, ada class Point dengan dua properti final: x dan y. class ini juga memiliki Constant Constructor yang menginisialisasi dua properti tersebut. class ini juga memiliki method display yang mencetak nilai dari dua properti tersebut.

Info

Catatan: Di sini p1 dan p2 memiliki kode hash yang sama. Ini karena p1 dan p2 adalah Object konstan. Kode hash dari Object konstan sama. Ini karena kode hash dari Object konstan dihitung saat waktu kompilasi. Kode hash dari Object non-konstan dihitung saat waktu runtime. Inilah mengapa p3 dan p4 memiliki kode hash yang berbeda.

```
class Point {  
    final int x;  
    final int y;  
  
    const Point(this.x, this.y);  
}  
  
void main() {  
    // p1 dan p2 memiliki kode hash yang sama.  
    Point p1 = const Point(1, 2);  
    print("Kode hash p1 adalah: ${p1.hashCode}");  
  
    Point p2 = const Point(1, 2);  
    print("Kode hash p2 adalah: ${p2.hashCode}");  
  
    // tanpa menggunakan const  
    // ini memiliki kode hash yang berbeda.  
    Point p3 = Point(2, 2);  
    print("Kode hash p3 adalah: ${p3.hashCode}");  
  
    Point p4 = Point(2, 2);  
    print("Kode hash p4 adalah: ${p4.hashCode}");  
}
```

Constant Constructor dalam Dart

Contoh 2: Constant Constructor dalam Dart

Dalam contoh di bawah ini, ada class Student dengan tiga properti: name, age, dan rollNumber. class ini memiliki satu Constant Constructor. Constructor ini digunakan untuk menginisialisasi nilai dari ketiga properti tersebut. Kami juga memiliki Object dari class Student yang disebut student.

```
class Student {  
  final String? name;  
  final int? age;  
  final int? rollNumber;  
  
  // Constant Constructor  
  const Student({this.name, this.age, this.rollNumber});  
}  
  
void main() {  
  // Here student is object of Student.  
  const Student student = Student(name: "John", age: 20, rollNumber: 1);  
  print("Name: ${student.name}");  
  print("Age: ${student.age}");  
  print("Roll Number: ${student.rollNumber}");  
}
```

Constant Constructor dalam Dart

Contoh 3: Constant Constructor dengan Parameter Bernama dalam Dart

Dalam contoh di bawah ini, ada class Car dengan tiga properti: name, model, dan prize. class ini memiliki satu Constructor. Constructor ini digunakan untuk menginisialisasi nilai dari ketiga properti tersebut. Kami juga memiliki Object dari class Car yang disebut car.

```
class Car {  
  final String? name;  
  final String? model;  
  final int? prize;  
  
  // Constant Constructor  
  const Car({this.name, this.model, this.prize});  
}  
  
void main() {  
  // Here car is object of class Car.  
  const Car car = Car(name: "BMW", model: "X5", prize: 50000);  
  print("Name: ${car.name}");  
  print("Model: ${car.model}");  
  print("Prize: ${car.prize}");  
}
```

Support with Follow/Subscribe

- Youtube : <https://youtube.com/@codewithbahri>
- Instagram: <https://instagram.com/codewithbahri>
- Linkedin: <https://linkedin.com/in/bahrie>
- Github: <https://github.com/bahrie127>
- Tiktok: <https://tiktok.com/@codewithbahri>
- WhatsApp: <https://wa.me/6285640899224>