#13 | Kelas Dart dan Konstruktor - Bagian 3

| **Outcome** | Mahasiswa mampu menggunakan enum type dan tipe data generic |
| --- | --- |
| **URL** | 13-kelas-dart-dan-construct-bagian-3 |
| **Category** | Mobile Flutter |
| **Environment** | Flutter, mobile, android, iOS, web, desktop |
| **Status** | Published |
| **Feedback Link** | https://github.com/jti-polinema/flutter-codelab/issues |
| **Author** | Tim Pengajar Mobile TSA 2022 |
| **Author LDAP** | hbb-polinema |
| **Analytics Account** | G-H7507PFEJB |

[Kelas Dart dan Konstruktor - Bagian 3](#_ok7k5uux6)

[Tujuan Praktikum](#_21yzqg98x7h6)

[Tujuan Visual](#_hs6hf8vnv2d6)

[Sumber Daya yang Dibutuhkan](#_dusjnkkj26ob)

[Daftar Perintah](#_6mgk7cf2soh8)

[Bantuan](#_f4c7kiz5py9x)

[Pengetahuan yang Anda harus dimiliki](#_319h1e1sjzdz)

[Tipe Data Enum](#_t0tqoph2ohlh)

[Tipe Data Generics](#_g7nutupwvu6v)

[Kapan dan Mengapa menggunakan Generics](#_f6859gp3jkil)

[Generics dan Dart Literals](#_3l9l0tbcvfsp)

[Nullability di dalam Generics](#_y1s0r9od88td)

[Praktikum 1:](#_86j27ymyw6sc)

[Praktikum 2:](#_gz7tbatl2uh8)

[Praktikum 3:](#_5n6rxl9gxlgw)

[Tugas Praktikum](#_dz35nqtk0wjb)

[Selamat!](#_s22qmcj4zu61)

[Apa selanjutnya?](#_9mqifffit2ew)

[Referensi](#_an95hoqxrlhg)

# Kelas Dart dan Konstruktor - Bagian 3



**Terakhir diperbarui:** 23 Agustus 2022

**Penulis:** Tim Pengajar Mobile Flutter

Pada codelab ini, Anda akan mempelajari konsep dan praktik untuk tipe data enum dan generics pada bahasa pemrograman Dart.

## Tujuan Praktikum

Setelah menyelesaikan codelab ini Anda akan mampu untuk:

* Menerapkan enum type pada bahasa pemrograman Dart
* Menerapkan tipe data generic pada bahasa pemrograman Dart

| **Catatan:** Materi ini diadaptasi dari Buku Flutter for Beginners - second edition dan Buku Ajar Pemrograman Mobile Flutter JTI Polinema serta sumber referensi lainnya. |
| --- |

## Tujuan Visual

Tidak ada, hanya menggunakan Console di DartPad.

## Sumber Daya yang Dibutuhkan

Berikut merupakan sumber daya yang diperlukan untuk menyelesaikan praktikum ini:

* PC atau Laptop dengan spesifikasi minimum RAM 4GB Processor Core i3
* Koneksi internet
* Chrome Browser

## Daftar Perintah

Pada praktikum ini, Anda akan menggunakan [DartPad](https://dartpad.dev/) sebagai kakas untuk bereksperimen atau praktikum.

## Bantuan

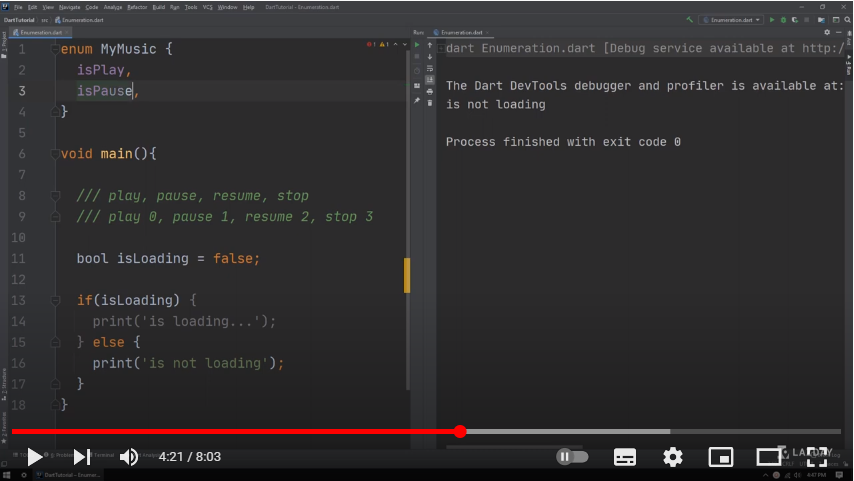
Tidak ada.

## Pengetahuan yang Anda harus Miliki

* Algoritma dan Pemrograman Dasar
* Pemrograman Berorientasi Objek

# Tipe Data Enum

Duration: 30:00



Tipe data enum adalah tipe data umum yang digunakan oleh sebagian besar bahasa pemrograman untuk mewakili satu set terbatas pada suatu nilai konstan. Di Dart, itu tidak ada perbedaan. Dengan menggunakan kata kunci enum, diikuti oleh nilai konstanta, Anda dapat menentukan tipe data enum, seperti yang diilustrasikan dalam kode program berikut ini:

| enum PersonType { student, employee } |
| --- |

Perhatikan kode tersebut bahwa Anda hanya menentukan nama dari suatu nilai. Tipe data enum adalah tipe data khusus yang memiliki satu set terbatas nilai dengan properti indeks yang mewakili nilainya. Sekarang, mari kita lihat bagaimana cara kerjanya.

Pertama, kita tambahkan suatu field ke kelas Person yang telah ditentukan sebelumnya untuk menyimpan tipe datanya, seperti berikut:

| class Person {  ...  PersonType? type;  ...  } |
| --- |

Kemudian, kita dapat menggunakannya seperti pada field lainnya, yang diilustrasikan dalam kode program berikut:

| main() {  print(PersonType.values);  Person somePerson = Person();  somePerson.type = PersonType.employee;  print(somePerson.type);  print(somePerson.type.index);  print(describeEnum(PersonType.employee));  } |
| --- |

Pernyataan print pertama akan mencetak [PersonType.student,PersonType.employee]. Anda dapat melihat bahwa dapat memanggil values pada enum PersonType secara langsung. Ini adalah variabel statis dari tipe enum yang dapat mengembalikan semua nilai-nilainya.

Pernyataan print kedua akan mencetak PersonType.employee.

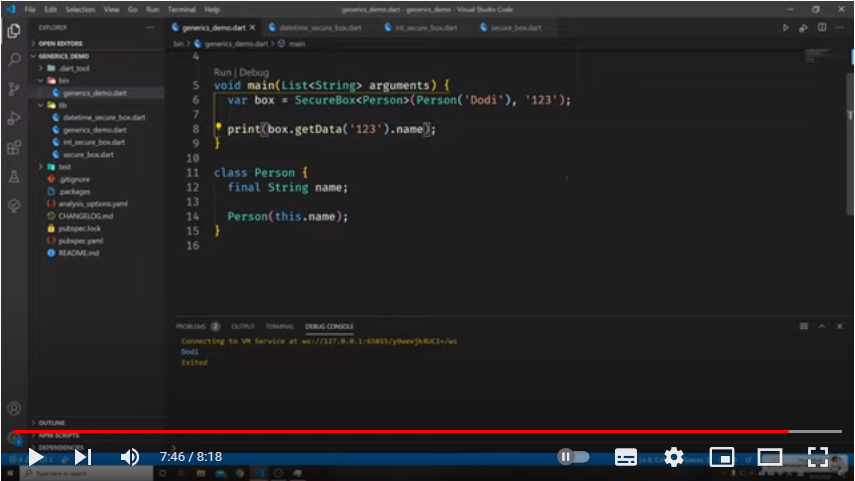
Pernyataan print berikutnya mencetak 1. Anda dapat melihat bahwa properti index bernilai nol, berdasarkan posisi deklarasi dari nilai tersebut. Secara umum, Anda tidak dianjurkan untuk mengambil nilai index karena dapat berubah jika nilai enum disusun ulang atau ditambahkan nilai baru.

Selain itu, Flutter menyediakan metode describeEnum, yang hanya mengembalikan nilai dari enum. Pada metode print terakhir, nilai yang akan dicetak adalah employee.

Semakin banyak Anda menambahkan *type safety* pada kode Anda, maka semakin aman kode tersebut. Namun, terkadang Anda juga ingin menentukan suatu tipe data yang dapat ditampung oleh suatu class, solusinya Anda dapat melakukannya dengan *generics*.

# Tipe Data Generics

Duration: 30:00



Sintaks < > digunakan untuk menentukan tipe data yang dapat didukung oleh suatu class. Jika Anda melihat contoh *List* dan *Maps* di *Pertemuan 7: Pengantar Bahasa Pemrograman Dart – Bagian 2*, Anda akan melihat bahwa belum ditentukan tipe data yang dapat dimuat. Hal ini karena informasi tipe data bersifat opsional, dan Dart memiliki fitur untuk menentukan sendiri tipe datanya berdasarkan isi elemen tersebut selama proses inisialisasi *collection*.

## Kapan dan Mengapa menggunakan Generics

Penggunaan tipe data *generics* dapat membantu pengembang untuk mengelola dan menjaga perilaku *collection* tetap terkontrol. Ketika kita menggunakan *collection* tanpa menentukan tipe data dari isi elemen yang dperbolehkan, maka itu perlu dilakukan secara manual untuk memeriksa elemen yang masuk ke dalam *collection* apakah sudah sesuai dengan tipe data yang diharapkan. Hal ini dapat menyebabkan *bug* ketika tipe data tidak sesuai dalam suatu *collection* atau asumsi yang salah untuk menambahkan elemen baru pada suatu *collection*.

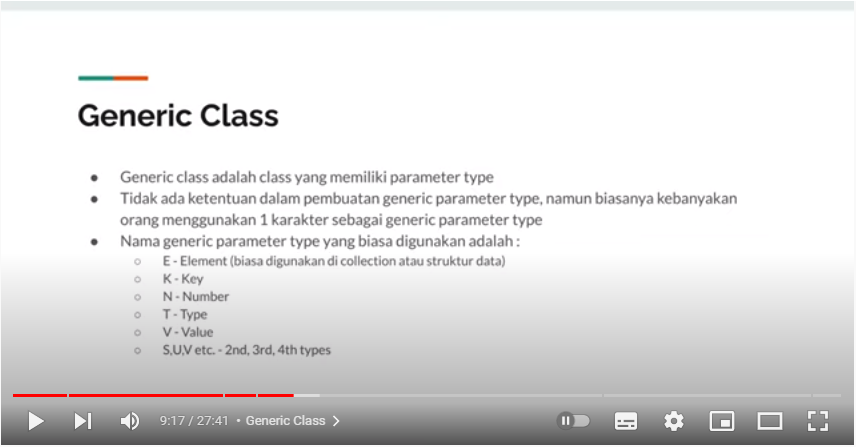
Perhatikan contoh kode berikut, di mana telah membuat sebuah List dengan nama variabel placeNames. Kita berharap variabel ini akan menjadi daftar nama dan tidak ada tipe data yang lain. Sayangnya, tanpa *generics*, kita dapat menambahkan tipe data apa saja ke dalam List tersebut, termasuk tipe data *number*. Hal ini dapat menyebabkan masalah saat mengambil nilai dari List tersebut:

| main() {  List placeNames = ["Middlesbrough", "New York"];  placeNames.add(1);  print("Place names: $placeNames");  }  // prints Place names: [Middlesbrough, New York, 1] |
| --- |

Namun, jika kita menentukan tipe data string untuk List, maka kode ini tidak akan dikompilasi, sehingga dapat meningkatkan kekuatan kode, seperti yang diilustrasikan pada kode program berikut ini:

| main() {  List<String> placeNames = ["Middlesbrough", "New York"];  placeNames.add(1);  // add() expects a String so this doesn't compile  } |
| --- |

## Generics dan Dart Literals



Pada contoh *list* dan *map* yang ada di *Pertemuan 7: Pengantar Bahasa Pemrograman Dart – Bagian 2*, Anda akan lihat itu menggunakan simbol literal yaitu [ ] dan { } untuk proses inisialisasi. Dengan *generics*, sebagai alternatif pada pendekatan sebelumnya, kita dapat menentukan tipe data selama proses inisialisasi, dengan cara menambahkan awalan <elementType>[] untuk tipe data *List* dan <keyType, valueType>{} untuk tipe data *Maps*.

Perhatikan contoh berikut:

| main() {  var placeNames = <String>["Middlesbrough", "New York"];  var landmarks = <String, String>{  "Middlesbrough": "Transporter bridge",  "New York": "Statue of Liberty",  };  } |
| --- |

Menentukan tipe data pada variabel list sepertinya berlebihan dalam kasus ini karena *Dart analyzer* akan dapat menduga bahwa itu merupakan tipe data string dari data literal yang ada. Namun, dalam beberapa kasus, hal ini penting, seperti ketika kita menginisialisasi variabel *collection* kosong, seperti pada contoh berikut:

| var emptyStringArray = <String>[]; |
| --- |

Jika kita tidak menentukan tipe data pada *collection* tersebut, maka itu bisa dimasukkan elemen dengan tipe data apa pun ke dalamnya karena itu tidak akan sesuai dengan tipe data generik yang ditentukan.

## Nullability di dalam Generics

Seperti yang telah kita pelajari di bagian *Null safety* pada *pertemuan Pengantar Bahasa Pemrograman Dart*, jika sebuah variabel dapat menerima nilai null, maka variabel itu harus dideklarasikan pada tipe data variabelnya. Ini juga berlaku pada tipe data *generics* jika tipe data *collection* dapat menerima nilai null.

Misalnya, pada variabel map landmarks, beberapa tempat boleh tidak memiliki nilai (null). Kita perlu mendeklarasikan ini sehingga ketika kita mengakses variabel map, kita akan tahu bahwa null adalah nilai yang mungkin ada. Mari kita perbarui dari contoh sebelumnya untuk melihat apa yang mungkin bisa diubah seperti berikut:

| main() {  var landmarks = <String, String?>{  "Middlesbrough": "Transporter bridge",  "New York": "Statue of Liberty",  "Barnmouth": null,  };  } |
| --- |

Kita telah menentukan bahwa nilai variabel map adalah String?, artinya itu dapat menerima nilai baik itu berupa String atau bernilai null. Kemudian kita tambahkan data baru ke variabel map yang berisi nilai null.

# Praktikum 1: Menerapkan enum

Duration: 30:00

Selesaikan langkah-langkah praktikum berikut ini menggunakan DartPad di browser Anda.

## Langkah 1:

Ketik atau salin kode program berikut ke DartPad.

| enum SkemaTSA { IoT, Mobile, Web } |
| --- |

## Langkah 2:

Buatlah sebuah class seperti kode berikut.

| class TSA {  SkemaTSA? mySkema;  } |
| --- |

## Langkah 3:

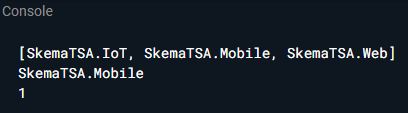
Ketik atau salin kode program berikut ke void main()

| print(SkemaTSA.values);  TSA myTSA = TSA();  myTSA.type = SkemaTSA.Mobile;  print(myTSA.type);  print(myTSA.type.index);  print(describeEnum(myTSA.Mobile)); |
| --- |

## Langkah 4:

Klik **Run** untuk mulai mengeksekusi kode. Analisis dan pahami dari kode program tersebut. Apa yang terjadi? Mengapa demikian? Jelaskan dalam laporan praktikum Anda!

Pastikan keluaran pada *console* seperti berikut.



# Praktikum 2: Menerapkan generic pada collection

Duration: 30:00

Selesaikan langkah-langkah praktikum berikut ini menggunakan DartPad di browser Anda.

## Langkah 1:

Ketik atau salin kode program berikut ke dalam void main().

| List namaKota = ["Kota Malang", "Kota Surabaya"];  namaKota.add(7);  print("Nama Kota di Jawa Timur: $NamaKota"); |
| --- |

## Langkah 2:

Klik **Run** untuk mulai mengeksekusi kode tersebut. Apa yang terjadi? Jika terjadi *error*, silakan diperbaiki. Mengapa ada angka 7 (tujuh) diantara nama kota tersebut? Jelaskan!

## Langkah 3:

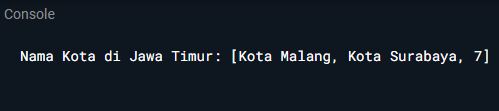
Ubah deklarasi List pada variabel namaKota seperti berikut.

| List<String> namaKota = ["Kota Malang", "Kota Surabaya"]; |
| --- |

## Langkah 4:

Lalu klik **Run** kembali. Analisis dan pahami dari kode program tersebut. Apa yang terjadi? Mengapa demikian? Jelaskan dalam laporan praktikum Anda!

Pastikan keluaran pada console seperti berikut.



# Praktikum 3: Menerapkan generic pada class

Duration: 30:00

Selesaikan langkah-langkah praktikum berikut ini menggunakan DartPad di browser Anda.

## Langkah 1:

Buatlah sebuah class generic di DartPad seperti berikut.

| class MyData<T> {    } |
| --- |

## Langkah 2:

Buat variabel generic di dalam class MyData dengan kode berikut.

| T data; |
| --- |

## Langkah 3:

Lalu buat konstruktornya.

| MyData(this.data); |
| --- |

## Langkah 4:

Buat method generic untuk mengembalikan nilai seperti kode berikut.

| T getData(){  return data;  } |
| --- |

## Langkah 5:

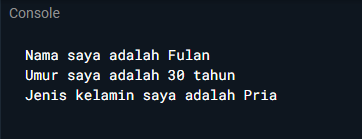
Salin atau ketik kode berikut ke void main()

| var namaSaya = MyData<String>("Fulan");  var umurSaya = MyData(30);  var genderSaya = MyData(true); |
| --- |

## Langkah 6:

Klik **Run** untuk mulai mengeksekusi kode. Analisis dan pahami dari kode program tersebut. Apa yang terjadi? Mengapa demikian? Jelaskan dalam laporan praktikum Anda!

Lengkapi kode pada Langkah 5, pastikan keluaran pada *console* seperti berikut.



# Tugas Praktikum

Duration: 90:00

1. Silakan selesaikan Praktikum 1 sampai 3, lalu dokumentasikan berupa screenshot hasil pekerjaan beserta penjelasannya!
2. Buatlah sebuah program yang terdiri dari class generic, enum, dan collection generic yang didalamnya terdapat indentitas nama lengkap dan alamat tinggal Anda!
3. Kumpulkan dalam format PDF ke LMS pada tautan yang telah disediakan!

# Selamat!

Duration: 0:00

Selamat Anda telah menyelesaikan Codelab ini sebagai langkah awal untuk memahami kelas dan konstruktor pada bahasa pemrograman Dart yang digunakan dalam framework Flutter.

Pada codelab berikutnya, Anda akan mempelajari tentang pemrograman *asynchronous* melalui Dart Futures dan Dart Isolates.

## Apa selanjutnya?

Silakan cek beberapa sumber belajar lainnya…

* [FlutLab is a modern Flutter online IDE](https://flutlab.io/)
* [Flutter System Architecture](https://docs.google.com/presentation/d/1cw7A4HbvM_Abv320rVgPVGiUP2msVs7tfGbkgdrTy0I/edit#slide=id.p)
* [The Dart type system](https://dart.dev/guides/language/type-system)
* [DartPad in tutorials: Best practices](https://dart.dev/resources/dartpad-best-practices)
* [Pengantar Dart untuk Developer Java (Codelab)](https://developers.google.com/codelabs/from-java-to-dart)
* <https://dart.dev/codelabs>
* <https://dart.dev/resources/videos>

## Referensi

* [Flutter for Beginners Second Edition](https://drive.google.com/file/d/1hJbctx-_O3B6kUeG70bJD5xfbnsWwBXY/view?usp=sharing)
* <https://www.youtube.com/flutterdev>
* <https://dart.dev/guides/language/language-tour>
* <https://github.com/PacktPublishing/Flutter-for-Beginners-Second-Edition>
* Buku Ajar Pemrograman Mobile Dengan Flutter, Jurusan Teknologi Informasi, Politeknik Negeri Malang. 2021.