



Dart Generic

Eko Kurniawan Khannedy

Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 10+ years experiences
- www.programmerzamannow.com
- youtube.com/c/ProgrammerZamanNow





Eko Kurniawan Khannedy

- Telegram : [@khannedy](https://t.me/khannedy)
- Facebook : fb.com/ProgrammerZamanNow
- Instagram : instagram.com/programmerzamannow
- Youtube : youtube.com/c/ProgrammerZamanNow
- Telegram Channel : t.me/ProgrammerZamanNow
- Email : echo.khannedy@gmail.com



Sebelum Belajar

- Dart Dasar
- Dart Object Oriented Programming



Agenda

- Pengenalan Generic
- Generic Class
- Generic Function
- Bounded Type Parameter
- Type Checking
- Dan lain-lain

Pengenalan Generic



Pengenalan Generic

- Generic adalah kemampuan menambahkan parameter type saat membuat class atau method
- Berbeda dengan tipe data yang biasa kita gunakan di class, di function, generic memungkinkan kita bisa mengubah-ubah bentuk tipe data sesuai dengan yang kita mau.



Manfaat Generic

- Pengecekan ketika proses kompilasi
- Tidak perlu manual menggunakan pengecekan tipe data dan konversi tipe data
- Memudahkan programmer membuat kode program yang generic sehingga bisa digunakan oleh berbagai tipe data

Kode : Bukan Generic

```
class Data {  
    dynamic data;  
}  
  
void main(List<String> arguments) {  
    var data = Data();  
    data.data = "Eko Kurniawan";  
    print(data.data);  
}
```



Kode : Generic

```
class Data<T> {  
    T? data;  
}  
  
void main(List<String> arguments) {  
    var data = Data<String>();  
    data.data = "Eko Kurniawan";  
    print(data.data);  
}
```

Membuat Project



Membuat Project

- `dart create belajar_dart_generic`

Generic Class



Generic Class

- Generic class adalah class yang memiliki parameter type
- Tidak ada ketentuan dalam pembuatan generic parameter type, namun biasanya kebanyakan orang menggunakan 1 karakter sebagai generic parameter type
- Nama generic parameter type yang biasa digunakan adalah :
 - E - Element (biasa digunakan di collection atau struktur data)
 - K - Key
 - N - Number
 - T - Type
 - V - Value
 - S,U,V etc. - 2nd, 3rd, 4th types



Kode : Generic Class

```
class MyData<T> {  
    T data;  
  
    MyData(this.data);  
}
```



Kode : Membuat Generic Object

```
import 'data/mydata.dart';

void main() {
  var dataString = MyData<String>("Eko");
  var dataNumber = MyData(100);
  var dataBool = MyData(true);

  print(dataString.data);
  print(dataNumber.data);
  print(dataBool.data);
}
```




Multiple Parameter Type

- Parameter type di Generic class boleh lebih dari satu
- Namun harus menggunakan nama type berbeda
- Ini sangat berguna ketika kita ingin membuat generic parameter type yang banyak



Kode : Multiple Parameter Type

```
class Pair<K, V> {  
    K first;  
    V second;  
  
    Pair(this.first, this.second);  
}
```



Kode : Multiple Parameter Type Object

```
import 'data/pair.dart';

void main(){
  var pair1 = Pair("Eko", 20);
  var pair2 = Pair<String, int>("Eko", 20);

  print(pair1.first);
  print(pair1.second);

  print(pair2.first);
  print(pair2.second);
}
```

Generic Function



Generic Function

- Generic parameter type tidak hanya bisa digunakan pada class
- Kita juga bisa menggunakan generic parameter type di function
- Generic parameter type yang kita deklarasikan di function, hanya bisa diakses di function tersebut, tidak bisa digunakan di luar function
- Ini cocok jika kita ingin membuat generic function, tanpa harus mengubah deklarasi class



Kode : Generic Function

```
class ArrayHelper {  
    static int count<T>(List<T> list) {  
        return list.length;  
    }  
}
```

Kode : Menggunakan Generic Function

```
import 'helper/array_helper.dart';

void main() {
  var numbers = [1, 2, 3, 4, 5, 6];
  var names = ["Eko", "Kurniawan", "Khannedy"];

  print(ArrayHelper.count(numbers));
  print(ArrayHelper.count(names));
}
```

Bounded Type Parameter



Bounded Type Parameter

- Kadang kita ingin membatasi data yang boleh digunakan di generic parameter type
- Kita bisa menambahkan constraint di generic parameter type dengan menyebutkan tipe yang diperbolehkan
- Secara otomatis, type data yang bisa digunakan adalah type yang sudah kita sebutkan, atau class-class turunannya
- Secara default, constraint type untuk generic parameter type adalah Object, sehingga semua tipe data bisa digunakan



Kode : Bounded Type Parameter

```
class NumberData<T extends num> {  
    T data;  
  
    NumberData(this.data);  
}
```



Kode : Menggunakan Bounded Type Parameter

```
import 'data/number_data.dart';

void main(){
  var dataString = NumberData("Eko"); // error
  var dataInt = NumberData(10);
}
```

—

Dynamic



Dynamic

- Kadang ada kasus kita tidak peduli dengan generic parameter type pada object
- Misal kita hanya ingin mem-print data T, tidak peduli tipe apapun
- Jika kita mengalami kasus seperti ini, kita bisa menggunakan dynamic
- Dynamic bisa dibuat dengan menghapus tipe data generic nya
- Semua tipe data generic otomatis menjadi tipe data dynamic ketika menggunakan fitur ini



Kode : Dynamic

```
import 'data/mydata.dart';

void printData(MyData data){
  print(data.data);
}

void main(){
  printData(MyData("Eko"));
  printData(MyData(100));
  printData(MyData(true));
}
```

Covariant



Covariant

- Covariant artinya kita bisa melakukan substitusi subtype (child) dengan supertype (parent)
- Namun hati-hati ketika melakukan covariant, karena jika sampai salah mengubah datanya, maka akan terjadi error pada saat runtime, tidak akan terdeteksi ketika proses compile

Kode : Covariant

```
import 'data/mydata.dart';

void main() {
  MyData<Object> data = MyData<String>("Eko");

  print(data.data);

  data.data = 100; // error ketika berjalan
}
```

Type Checking



Type Checking

- Generic di Dart mendukung Type Checking, berbeda dengan Java yang menggunakan fitur type erasure, yang artinya ketika di-compile, informasi generic nya dihilangkan. Pada Dart, semua informasi generic tetap ada
- Oleh karena itu kita bisa melakukan Type Checking di Dart walaupun sampai ke level parameterized type nya



Kode : Type Checking

```
void check(dynamic data) {  
    if (data is MyData<String>) {  
        print("String");  
    } else if (data is MyData<num>) {  
        print("num");  
    } else {  
        print("Object");  
    }  
}  
  
void main() {  
    check(MyData("Eko"));  
    check(MyData(100));  
    check(MyData(true));  
}
```

Materi Selanjutnya



Materi Selanjutnya

- Dart Packages
- Dart Unit Test
- Dart Standard Library
- Dart Async
- Dart Reflection