Modul Pendamping Perkuliahan Pemrograman Berbasis Mobile

Pertemuan ke : 3
Topik : OOP pada Dart

A. Implementasi Konsep-Konsep Dasar OOP

1. Buatlah sebuah file bernama **bangun-datar.dart** dan tuliskan kode program berikut ke dalamnya:

```dart
abstract class BangunDatar {
  double panjang;
  double lebar;

  // Konstruktor
  BangunDatar(double p, double l) {
    panjang = p;
    lebar = l;
  }

  // Abstract method
  double luas();
  double keliling();
}
```

2. Buat kembali file baru bernama **persegi-panjang.dart** yang berada pada direktori yang sama dengan **bangun-datar.dart** dan tuliskan kode program berikut ke dalamnya:

```dart
import 'bangun-datar.dart';

class PersegiPanjang extends BangunDatar {
  PersegiPanjang(double p, double l) : super(p, l);

  PersegiPanjang.samaSisi(double s) : super(s, s);

  double luas() {
    return panjang * lebar;
  }

  double keliling() {
    return 2 * (panjang + lebar);
  }
}
```

3. Untuk menguji kedua class tersebut, buat file baru bernama **tes-persegi-panjang.dart** dan tuliskan main function berikut ke dalamnya:

```
import './persegi-panjang.dart';
void main() {
  var p1 = PersegiPanjang(10, 5);
  var p2 = PersegiPanjang.samaSisi(6);

  print('''
    Persegi Panjang
    ---------------
    - Luas = ${p1.luas()}
    - Keliling = ${p1.keliling()}

    Persegi
    -------
    - Luas = ${p2.luas()}
    - Keliling = ${p2.keliling()}
  ''');
}
```

4. Jalankan kode program tersebut dengan perintah **dart tes-persegi-panjang.dart** pada Terminal. Anda akan mendapatkan hasil semacam ini:

```
PS D:\Lecturing\Matakuliah\Pemrograman Berbasis Mobile\Codes> dart tes-persegi-panjang.dart
    Persegi Panjang
    ---------------
    - Luas = 50.0
    - Keliling = 30.0

    Persegi
    -------
    - Luas = 36.0
    - Keliling = 24.0
```

B. Implementasi Setter & Getter

1. Buat sebuah file bernama **lingkaran.dart** dan tuliskan kode program berikut di dalamnya:

```
import 'dart:math';

class Lingkaran {
  // Property
  double _phi, _radius;
  int _x, _y;

  // Constructor
  Lingkaran({int x, int y, double radius}) {
    _phi = 22 / 7;
    _x = x ?? 0;
    _y = y ?? 0;
    _radius = radius ?? 5;
```

```
  }

  // Setter
  set x(int x) => _x = x;
  set y(int y) => _y = y;
  set radius(double r) => _radius = r;

  // Getter
  int get x => _x;
  int get y => _y;
  double get radius => _radius;

  // Methods
  double _luas() {
    return _phi * pow(_radius, 2);
  }

  double _keliling() {
    return 2 * _phi * _radius;
  }

  double _jarakPusat(Lingkaran lingkaran) {
    return sqrt(pow(_x - lingkaran.x, 2) + pow(_y - lingkaran.y, 2));
  }

  double _jarakTepi(Lingkaran lingkaran) {
    return _jarakPusat(lingkaran) - (_radius + lingkaran.radius);
  }

  String luas() {
    return _luas().toStringAsFixed(2);
  }

  String keliling() {
    return _keliling().toStringAsFixed(2);
  }

  String jarak(Lingkaran lingkaran) {
    if (_jarakTepi(lingkaran) == 0)
      return "tidak berjarak karena tepi kedua lingkaran tepat bersinggungan";
    else if (_jarakTepi(lingkaran) < 0)
      return "tidak berjarak karena kedua lingkaran saling beririsan";
    else
      return _jarakTepi(lingkaran).toStringAsFixed(2);
  }
}
```

2. Deklarasikan main function pada file berbeda dengan nama **tes-lingkaran.dart**:

```dart
import './lingkaran.dart';

void main() {
  Lingkaran o1 = Lingkaran();
  Lingkaran o2 = Lingkaran(radius: 2);
  Lingkaran o3 = Lingkaran(x: 10, y: 4);
  Lingkaran o4 = Lingkaran(x: 20, y: 20, radius: 3);

  // Mengubah properti y dengan memanfaatkan setter-nya
  o3.y = 0;

  // Akses properti x, y, dan radius pada string di bawah ini merupakan
  // contoh pemanfaatan getter untuk mendapatkan nilai masing-masing
  // properti
  print('''
    Pusat lingkaran pertama = (${o1.x}, ${o1.y}) & radius = ${o1.radius}
    Pusat lingkaran kedua = (${o2.x}, ${o2.y}) & radius = ${o2.radius}
    Pusat lingkaran ketiga = (${o3.x}, ${o3.y}) & radius = ${o3.radius}
    Pusat lingkaran keempat = (${o4.x}, ${o4.y}) & radius = ${o4.radius}

    Luas lingkaran pertama = ${o1.luas()} & keliling = ${o1.keliling()}
    Luas lingkaran kedua = ${o2.luas()} & keliling = ${o2.keliling()}
    Luas lingkaran ketiga = ${o3.luas()} & keliling = ${o3.keliling()}
    Luas lingkaran keempat = ${o4.luas()} & keliling = ${o4.keliling()}

    Jarak lingkaran pertama & kedua = ${o1.jarak(o2)}
    Jarak lingkaran pertama & ketiga = ${o1.jarak(o3)}
    Jarak lingkaran kedua & ketiga = ${o2.jarak(o3)}
    Jarak lingkaran ketiga & keempat = ${o3.jarak(o4)}
  ''');
}
```

3. Jalankan kode program tersebut dengan perintah **dart tes-lingkaran.dart** pada Terminal. Anda akan mendapatkan tampilan hasil semacam ini:

```
PS D:\Lecturing\Matakuliah\Pemrograman Berbasis Mobile\Codes> dart tes-lingkaran.dart
    Pusat lingkaran pertama = (0, 0) & radius = 5.0
    Pusat lingkaran kedua = (0, 0) & radius = 2.0
    Pusat lingkaran ketiga = (10, 0) & radius = 5.0
    Pusat lingkaran keempat = (20, 20) & radius = 3.0

    Luas lingkaran pertama = 78.57 & keliling = 31.43
    Luas lingkaran kedua = 12.57 & keliling = 12.57
    Luas lingkaran ketiga = 78.57 & keliling = 31.43
    Luas lingkaran keempat = 28.29 & keliling = 18.86

    Jarak lingkaran pertama & kedua = tidak berjarak karena kedua lingkaran saling beririsan
    Jarak lingkaran pertama & ketiga = tidak berjarak karena tepi kedua lingkaran tepat bersinggungan
    Jarak lingkaran kedua & ketiga = 3.00
    Jarak lingkaran ketiga & keempat = 14.36
```

C. Implementasi Interface

1. Buat sebuah file bernama **person.dart** lalu tuliskan kode program berikut:

```
class Person {
  final _name;

  Person(this._name);

  String greet(String who) => 'Hello, $who. I am $_name.';
}
```

2. Buat file bernama **impostor.dart** lalu tuliskan kode program berikut:

```
import './person.dart';

class Impostor implements Person {
  get _name => '';

  String greet(String who) => 'Hi $who. Do you know who I am?';
}
```

3. Buat file bernama **tes-person.dart** lalu tuliskan kode program berikut:

```
import './person.dart';
import './impostor.dart';

String greetBob(Person person) => person.greet('Bob');

void main() {
  print(greetBob(Person('Kathy')));
  print(greetBob(Impostor()));
}
```

4. Jalankan kode program tersebut dengan perintah **dart tes-person.dart** pada Terminal. Anda akan mendapatkan hasil sebagai berikut:

```
PS D:\Lecturing\Matakuliah\Pemrograman Berbasis Mobile\Codes> dart tes-person.dart
Hello, Bob. I am Kathy.
Hi Bob. Do you know who I am?
```

5. Terlihat bahwa sekalipun fungsi **greetBob** menerima sebuah parameter dengan tipe **Person**, fungsi tersebut juga tetap dapat menerima objek dari class Impostor karena class Impostor sendiri mengimplementasikan class Person sebagai interface-nya.

D. Implementasi Mixin

1. Tuliskan kode program berikut ke dalam sebuah file bernama **tes-mixin.dart**:

```
mixin Musical {
  bool canPlayPiano = false;
  bool canCompose = false;
```

```
  bool canConduct = false;

  void entertainMe() {
    if (canPlayPiano) {
      print('Playing piano');
    } else if (canConduct) {
      print('Waving hands');
    } else {
      print('Humming to self');
    }
  }
}

class Person {
  String name;

  Person(this.name);

  void shoutOut() {
    print('My name is $name');
  }
}

class Maestro extends Person with Musical {
  Maestro(name) : super(name) {
    canConduct = true;
  }
}

void main() {
  Maestro fulan = new Maestro('Fulan');

  fulan.shoutOut();
  fulan.entertainMe();
}
```

2. Jalankan kode program tersebut dengan perintah **dart tes-mixin.dart** pada Terminal. Anda akan mendapatkan hasil sebagai berikut:

```
PS D:\Lecturing\Matakuliah\Pemrograman Berbasis Mobile\Codes> dart tes-mixin.dart
My name is Fulan
Waving hands
```

3. Perhatikan bahwa properti **canConduct** tetap dapat diinisiasi dari dalam konstruktor class Maestro meski class Maestro tidak mendeklarasikan properti tersebut.
4. Objek **fulan** yang merupakan instansiasi dari class Maestro pun dapat mengakses method **entertainMe()** yang berasal dari mixin Musical.