

Rumah Coding Course



# Flutter

Studi Kasus Aplikasi "SekolahKu"

Day 2



# Conditional



# Conditional

- Hanya nilai Boolean yang diterima dalam dart
- Menggunakan Pernyataan If untuk satu kondisi
- Menggunakan pernyataan If..else untuk dua kondisi
- Menggunakan if ... else if ... else untuk lebih dari dua kondisi
- For if... kondisi lain memiliki tulisan cepat. Ini disebut operator ternary, beranotasi dengan ... ? ... : ....



# Conditional

```

● ● ●

// if statement
if (condition) {
    // condition true here
}

// if..else if..else
if (condition) {
    // condition true here
} else if(secondCondition) {
    // second condition here
} else {
    // otherwise here
}

```



# Conditional



```
// if..else
if (condition) {
    // condition true here
} else {
    // otherwise here
}

// ternary operator, if..else shorthand
condition ? doSomething : doSomethingElse
```



# Logical operator

Operator	Description
&&	AND
	OR
!	NOT



# Logical operator



```
var sholat = true;
var bersuci = true;
var message = '';
if (bersuci && sholat) {
    message = 'Sha dan Insya Allah diterima';
} else if (!bersuci && sholat) {
    message = 'Tidak sah';
} else {
    message = 'Muslim?';
}
```



# Comparison operator

Operator	Description
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to





# Comparison operator



```
var age = 17;  
var message = '';  
if (age < 18) {  
    message = 'For 18+ only';  
} else {  
    message = 'go ahead';  
}  
  
// use ternary operator expression  
message = age < 18 ? 'For 18+ only' : 'go ahead';
```



# Challenge


Create Conditional statement with other operator



# Setter Getter

- Properti pengakses
- Dapat digunakan sebagai pembungkus atas nilai properti "nyata" untuk mendapatkan kontrol lebih besar atas operasi dengannya.
- Izinkan untuk mengambil kendali atas properti data "biasa"
- Dapat digunakan sebagai properti hanya baca





```
class Person {  
    final String name;  
    final DateTime birthday;  
  
    const Person(this.name, this.birthday);  
  
    int get age {  
        var todayYear = new DateTime.now().year;  
  
        return todayYear - this.birthday.year;  
    }  
}  
  
// instance person  
var john = new Person('John', DateTime.parse('1969-07-20'));  
print(john);  
print(john.name);  
print(john.age);
```



# Setter Getter

- Try to set the age to 12 or whatever integer



```
john.age = 12;  
print(john.age);
```



# Setter Getter

- Kesalahan akan meningkat, sesuatu seperti ini
- Kesalahan ditemukan dalam waktu kompilasi
- Umur adalah properti pengakses hanya baca

```
main.dart:57:8:  
Error: The setter 'age' isn't defined for the class 'Person'.  
- 'Person' is from 'main.dart'.  
  john.age = 12;  
    ^^^  
Error: Compilation failed.
```



```
class Person {  
    final String name;  
    final DateTime birthday;  
    String _lastName;  
  
    Person(this.name, this.birthday);  
  
    int get age {  
        var todayYear = new DateTime.now().year;  
  
        return todayYear - this.birthday.year;  
    }  
  
    // inline style with arrow fat function  
    String get lastName => this._lastName; // getter  
    set lastName(String value) => this._lastName = value; // setter  
}  
  
// instance person  
var john = new Person('John', DateTime.parse('1969-07-20'));  
john.lastName = 'Lenon'; // setter  
print(john);  
print(john.name);  
print(john.lastName); // getter  
print(john.age); // getter (readonly)
```



# .map()

- Fungsi list.map
- Mengembalikan lazy Iterable baru dengan elemen yang dibuat dengan memanggil fungsi pada setiap elemen Iterable ini dalam urutan iterasi.
- Setiap widget seperti Collections. Katakanlah anak-anak, anak-anak adalah jenis daftar





# .map()



```
var list = [1, 4, 5, 6];
```

```
var newList = list.map((x) => x * 2).toList();
```

```
print(list); // [1, 4, 5, 6]
```

```
print(newList); // expected output [2, 8, 10, 12]
```



# .map()



```
var genders = ['pria', 'wanita'];  
  
Row(  
    children: genders  
        .map((String val) => Text(val))  
        .toList(),  
);
```



# Back to our Project



# Our Project Structure

We need to re-organize our file and directory per responsibility.

Path	Responsibility	Kind
lib/	Our internal code here.	directory
lib/domain/	For domain file	directory
lib/repository/	For repository	directory
lib/service/	For service file	director
lib/screens/	For our Screen component file	directory
lib/widgets	For our UI component file	directory
lib/model/	For our model database file	directory
assets/	For our static asset (image, icon) file	directory
lib/util/	For utility function file	directory
lib/seed/	For dummy data	directory



```
> assets
> build
> ios
▼ lib
  > domain
  > model
  > repository
  > screens
  > seed
  > service
  > util
  > widgets
  main.dart
```

# App Project Structure



# SOLID Principle

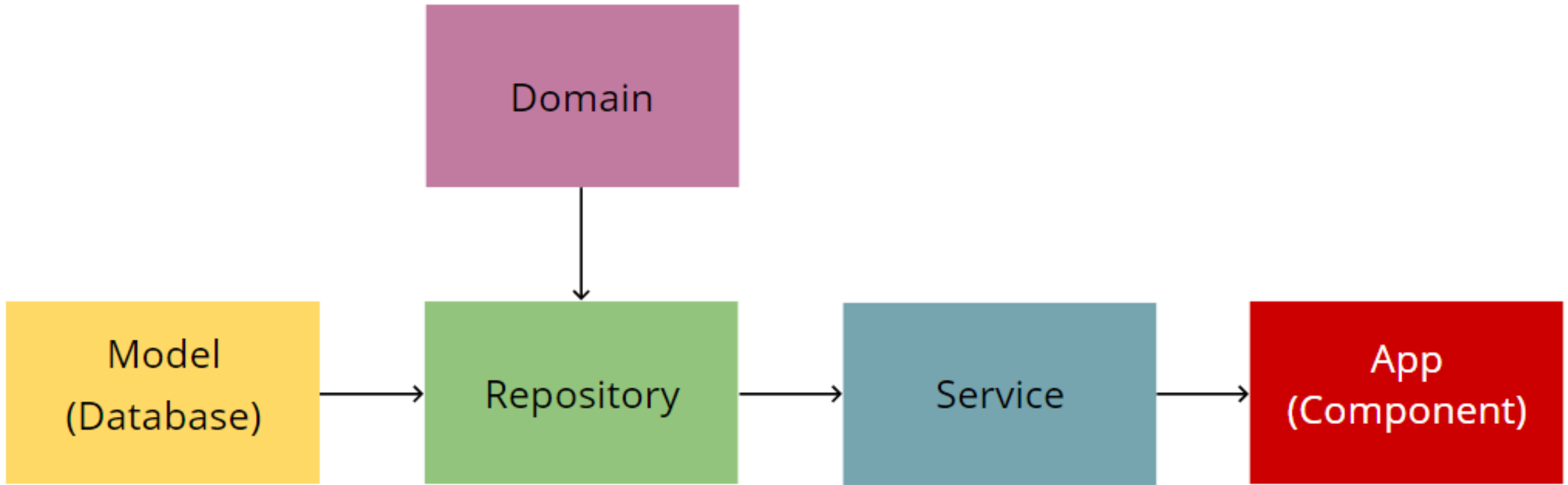
- **S**ingle Responsibility Principle(Prinsip Tanggung Jawab Tunggal)
- **O**pen/Closed Principle (Prinsip Terbuka / Tertutup)
- **L**iskov Subtitution Principle (Prinsip Pergantian Liskov)
- **I**nterface Segregation Principle(Prinsip Pemisahan Antarmuka)
- Prinsip Pembalikan Ketergantungan



# Dengan mengikuti SOLID:

- Bantu kami untuk mencegah kode kami menjadi kode spaghetti. Berarti, itu akan sulit untuk dipertahankan, sulit untuk berubah dan sulit untuk diuji.
- 1 kelas / modul untuk satu tanggung jawab.
- Ketahui alasan untuk berubah
- Memisahkan logika / kode Anda
- Jangan pedulikan ketergantungan Anda yang dapat berubah secara internal







# Domain

- Pada dasarnya, ini adalah "model" objek yang diperlukan untuk tujuan bisnis Anda.
- Jelaskan entitas objek (properti)
- Jelaskan perilaku objek dan logika (metode)
- Sebagai skema untuk basis data
- Di JAVA, itu disebut kelas POJO.
- Seperti pada contoh prev, kami menggambarkan kelas Object Person / Animal



# Repository

- Lapisan Persistensi - berarti interaksi dengan basis data
- Jembatan antara logika bisnis dan ketekunan(persistence) logika .
- Pada dasarnya, kelas mendefinisikan metode CRUD



# Service

- Logika bisnis Anda - Suka otorisasi
- Lapisan jaringan (seperti dari REST API)
- Jembatan antara logika bisnis dan aplikasi Anda (client/ UI)



# Model

Dalam kasus penggunaan kami:

- Konfigurasi basis data
- Kelola koneksi basis data
- Tentukan Skema Basis Data, Migrasi, dll



# Let's start

Back to our project



# User Stories

Klien kami membutuhkan aplikasi untuk menyediakan data siswa, mereka memerlukan beberapa informasi untuk setiap siswa. Yang mereka butuhkan sangat sederhana, yaitu nama lengkap, nomor ponsel, jenis kelamin, kelas, alamat, dan hobi.



# What to do

Pertama kami mengerjakan tugas sederhana:

- Buat domain untuk siswa
- Buat repositori untuk siswa
- Buat layanan untuk siswa dengan nama metode yang jelas (spesifik)
- Tampilkan data ke UI kami
- Buat fungsi utilitas untuk memanfaatkan huruf besar



# capitalize.dart



```
// lib/util/capitalize.dart
String capitalize(String str) {
  if (str == null) return '';

  return '${str[0].toUpperCase()} + str.substring(1)';
}
```





# Student Domain

[bit.ly/StudentDart1](https://bit.ly/StudentDart1)



# Student Repository

[bit.ly/StudentRepositoryDart1](https://bit.ly/StudentRepositoryDart1)



# Student Service

[bit.ly/StudentServiceDart1](https://bit.ly/StudentServiceDart1)



# Create a seed

dummy data



# Create a seed

- Instal faker sebagai dependensi dev
- Buat fungsi utilitas ke nomor acak, fungsi ini digunakan untuk mengakses array indeks secara acak
- Buat file seed



# Add Faker

```
dev_dependencies:  
  flutter_test:  
    sdk: flutter  
  faker: ^1.1.1
```



# Create random function

- lib/util/random.dart



```
// lib/util/random.dart
import 'dart:math';

int getRandom(int num) {
  return Random().nextInt(num);
}
```



# Create seed file

- [bit.ly/SeedStudentDart](https://bit.ly/SeedStudentDart)





# App Service

- Untuk layanan root
- dikonsumsi oleh aplikasi secara langsung





```
// lib/service/app_service.dart
import 'package:sekolahku/repository/student_repository.dart';
import 'package:sekolahku/service/student_service.dart';

StudentRepository _studentRepository = StudentRepository();
StudentService _studentService = StudentService(_studentRepository);

class AppService {
  // static getter
  static StudentService get studentService => _studentService;
}
```



# Static Class property

- Bukan properti instan
- Tapi itu properti kelas
- Mengakses seperti objek biasa (e.g. `AppService.staticPropname`)
- Dapat menggabungkan dengan pengambil(getter).



# Screen of list students

- Buat Home.dart di direktori lib
- Buat kelas Home untuk menampilkan daftar semua siswa
- Menggunakan ListView. Dipisahkan dari widget materi
- Jangan lupa untuk menggunakan Scaffold
- Untuk setiap siswa Item, gunakan widget ListTile
- Untuk garis bawah, gunakan widget Divider
- Tambahkan initial state, katakanlah `students` dan tetapkan dengan `AppService.studentService.findAllStudents()`;



# Stateful & Stateless Widget

```
lib / screens / ... / cobaz.dart / ... / stl  
1  stl
```

- ☐ Flutter **stateless** widget Insert a StatelessWidget... ⓘ
- ☐ Flutter **stateful** widget



```
import 'package:flutter/material.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  @override
  void initState() {
    super.initState();
  }


  @override
  void dispose() {
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

# Stateful Widget



# Stateless Widget



```
import 'package:flutter/material.dart';

class HomeLess extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```



# Perbedaan

- Stateful memiliki state
- State berarti data akan tersedia karena widget itu hidup dan dapat berubah jika diperlukan
- Stateful perlu Kelas state ,untuk menggunakan state
- Kelas state memiliki metode siklus hidup untuk berinteraksi dengan state
- setState () adalah metode untuk mengubah keadaan dan memaksa metode membangun untuk membangun kembali
- Singkatnya, widget stateless hanya memiliki metode build





# Screen of list students

- [bit.ly/HomeDart1](https://bit.ly/HomeDart1)





```
import 'package:flutter/material.dart';
import 'package:sekolahku/screens/home.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```







# Update lib/main.dart



# Challenge

- Create Detail Student
- Using `ListView` & `ListTile`
- Using `AppService.studentService.findStudentBy(index: 1);` to get one student in index 0 of List students
- Use icon as you like



Detail	
	Rossie Funk
	195-275-1977
	pria
	sma
	2451 Schultz Garden
	membaca, menulis, menggambar

# Challenge



# Next..

Add Image

# Add Image

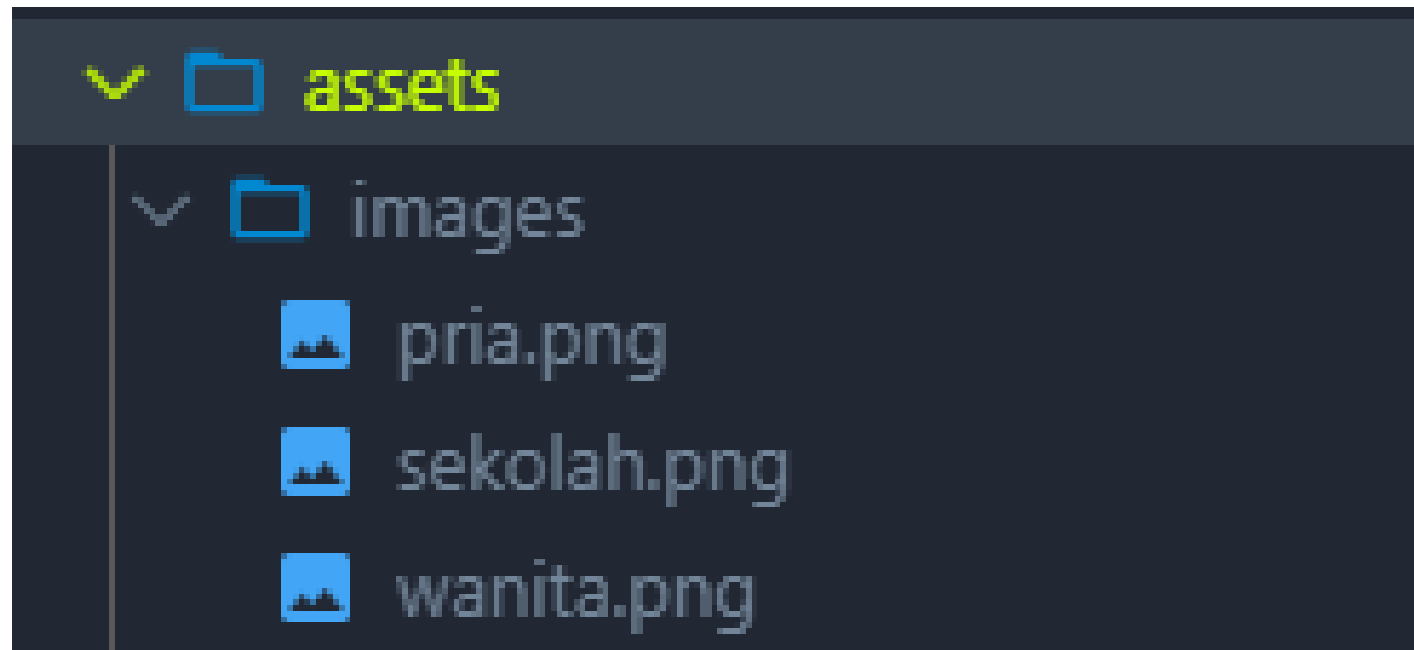


# Add Image



# Add Image

- Place image file Inside assets/images directory
- And include that in our [pubspec.yaml](#)








# Include that in pubspec


```
37 # The following section is specific to Flutter.
38 flutter:
39
40 # The following line ensures that the Material Icons font is
41 # included with your application, so that you can use the icons in
42 # the material Icons class.
43 uses-material-design: true
44
45 # To add assets to your application, add an assets section, like this:
46 # assets:
47 #   - images/a_dot_burr.jpeg
48 #   - images/a_dot_ham.jpeg
49 assets:
50   - assets/images/pria.png
51   - assets/images/sekolah.png
52   - assets/images/wanita.png
53
```





# Update DetailStudent


- [bit.ly/DetailStudentDart1](http://bit.ly/DetailStudentDart1)
- Note: to see result change [main.dart](#)


 Detail Siswa  




 John Wick


 666666666666

 Pria

 SMA

 Unknown



 **Tambah Siswa**

Nama Depan

Nama Belakang

No. Hp

Jenis Kelamin

☒ Pria ☐ Wanita

Pilih jenjang


Hobi

☐ Membaca

☐ Menulis

☐ Menggambar

Alamat



# Create form student

# Before we create a form

- Create a custom widget for radio button
- Create custom widget for checkbox



# CheckBox

- Use InkWell
- Use Padding
- Use Row
- Has label in the right
- [bit.ly/CheckBoxDart](https://bit.ly/CheckBoxDart)



# Challenge

- Create custom radio button
- Use Row
- Has label in the right



# Challenge

- [bit.ly/RadioButtonDart](https://bit.ly/RadioButtonDart)



# Challenge

- Create form student
- Use `Form` widget
- Use `TextFormField` instead of `TextField` widget
- Use `Space` widget to create a space
- Use `DropDownButton`
- Use `RadioButton` (our custom)
- Use `CheckBox` (our custom)
- Use `FormLabel` (our custom)





# Challenge

- [bit.ly/FormStudentDart1](https://bit.ly/FormStudentDart1)



# FormLabel

- Move our **FormLabel** to widgets directory



# Form

- Widget Formulir bertindak sebagai wadah untuk mengelompokkan dan memvalidasi beberapa bidang formulir.
- Saat membuat formulir, berikan GlobalKey. Ini secara unik mengidentifikasi Formulir, dan memungkinkan validasi formulir di langkah selanjutnya.



# @required

- Fungsi dekorator dari widget materi
- Untuk memberi tahu kami bahwa diperlukan argumen atau parameter
- Ini akan berteriak kepada Anda jika Anda tidak melewati parameter yang diperlukan

```
The parameter 'title' is required. dart(missing_required_param)
```

[Peek Problem](#) [Quick Fix...](#)



# @required

- So in `main.dart` we Pass `FormStudent()`; with title params

```
MaterialApp(  
  debugShowCheckedModeBanner: false,  
  title: 'Sekolahku',  
  theme: ThemeData(  
    primarySwatch: Colors.blue,  
  ),  
  home: FormStudent(title: 'Buat Siswa'),  
);
```



# Stateful & widget Object

- In `FormStudentState` class you can see `widget.title`, where is this come from



```
AppBar(  
  leading: IconButton(  
    icon: Icon(Icons.arrow_back),  
    onPressed: () {},  
  ),  
  title: Text(widget.title),  
),
```



# Stateful & widget Object

- Objek widget berasal dari kelas atas kami, kelas FormStudent. Jadi setiap properti akan tersedia di dalam objek widget dan kita dapat mengaksesnya



```
class FormStudent extends StatefulWidget {  
  final String title;  
  
  const FormStudent({Key key, @required this.title}) : super(key: key);  
  
  @override  
  _FormStudentState createState() => _FormStudentState();  
}
```

