

Nama: Innocentia Handani

NIM: 0806022410010

```
class CounterModel {  
  int value;  
  String label;  
  Color color;  
  
  CounterModel({  
    required this.value,  
    required this.label,  
    required this.color,  
  });  
}
```

Ini merupakan *class* yang mendefinisikan apa saja yang dimiliki oleh *counter*, yaitu *value*, *label* (nama counter), dan *color* (warna card). *Required* memastikan bahwa semua parameter tersebut diisi.

```
class GlobalState with ChangeNotifier {  
  final List<CounterModel> _counters = [];  
  
  List<CounterModel> get counters => _counters;  
}
```

ChangeNotifier merupakan salah satu *behavioral design pattern* yang dapat digunakan sebagai *pattern observer* ketika user mengubah data untuk memberi tahu

objek yang bekerja dengan data tersebut. Dan supaya *multiple counter* dapat bekerja, maka *_counters* menggunakan *encapsulation* dan menggunakan getter agar aksesnya hanya *read only*.

Project ini menggunakan pendekatan state management *Provider Pattern* yang sederhana dan sudah built-in dalam ekosistem Flutter. Dengan *Provider*, dapat menggunakan *ChangeNotifier* untuk membantu mengelola perubahan UI ketika terdapat perubahan data, *notifyListeners* akan dipanggil tiap kali state berubah (jadi selalu ada di method-method yang akan merubah data). Widget akan *rebuild* secara otomatis.

```
void decrementCounter(int index) {  
  if (_counters[index].value > 0) {  
    _counters[index].value--;  
    notifyListeners();  
  }  
}  
  
void reorderCounters(int oldIndex, int newIndex) {  
  if (oldIndex < newIndex) {  
    newIndex -= 1;  
  }  
  final CounterModel item = _counters.removeAt(oldIndex);  
  _counters.insert(newIndex, item);  
  notifyListeners();  
}  
  
void updateCounterLabel(int index, String newLabel) {  
  if (index >= 0 && index < _counters.length) {  
    _counters[index].label = newLabel.isEmpty ? 'Counter ${index + 1}' : newLabel;  
    notifyListeners();  
  }  
}
```

```
void addCounter() {  
  _counters.add(CounterModel(  
    value: 0,  
    label: 'Counter ${_counters.length + 1}',  
    color: Colors.primaryes[Random().nextInt(Colors.primaryes.length)],  
  )); // CounterModel  
  notifyListeners();  
}  
  
void changeCounterColor (int index) {  
  if (index >= 0 && index < _counters.length) {  
    final newColor = Colors.primaryes[Random().nextInt(Colors.primaryes.length)];  
    _counters[index].color = newColor;  
    notifyListeners();  
  }  
}  
  
void removeCounter(int index) {  
  _counters.removeAt(index);  
  notifyListeners();  
}  
  
void incrementCounter(int index) {  
  _counters[index].value++;  
  notifyListeners();  
}
```

Untuk method-method yang digunakan ada *addCounter* untuk menambahkan counter baru dengan *value* 0 dan *label* otomatis berdasarkan urutan dan warnanya di berikan acak dari color palette *Primaryes* Flutter.

Nama: Innocentia Handani

NIM: 0806022410010

changeCounterColor menggunakan validasi index untuk mencegah terjadi error dan disini yang menentukan warna random yang diberikan pada inisiasi awal. removeCounter akan menghapus counter berdasarkan indexnya. incrementCounter dan decrementCounter digunakan untuk menambah dan mengurangi value dan pada method decrement diberikan validasi bahwa value tidak bisa dibawah 0. reorderCounters menggunakan implementasi advanced yaitu drag and drop rendering sehingga newIndex akan menyesuaikan ketika terjadi drag and drop.

updateCounterLabel digunakan jika user melakukan perubahan nama label, dan jika user menyimpan perubahan dengan label kosong, maka secara default akan menggunakan counter + index.

Project ini telah mengikuti best practice Flutter dalam hal:

1. Encapsulation melalui private `_counters` list dengan public getter
2. Immutable Constructor dengan menggunakan required parameter
3. Consistent naming dengan CamelCase dan descriptive name (bukan asal nama variable)
4. Single Responsibility yang telah dijelaskan diatas bahwa tiap method memiliki 1 fungsi.
5. Observer Pattern dengan memanfaatkan ChangeNotifier
6. ReactiveUpdate dengan menggunakan notifyListeners

Advanced feature yang digunakan yaitu:

1. Reorder Functionality, menggunakan drag and drop untuk reordering list dan index adjustment
2. Dynamic Color System dengan menggunakan material design colors dan assign warna otomatis dengan random selection
3. Flexible Labeling yaitu auto numbering, custom labels dan fallback system jika input kosong.

Credit:

1. Aristo Iskandar, kami bertukar pandangan terhadap bagaimana konsep untuk menciptakan global state management.
2. Derick Norlan dan Michael C S, untuk memberikan insight tentang Flutter.