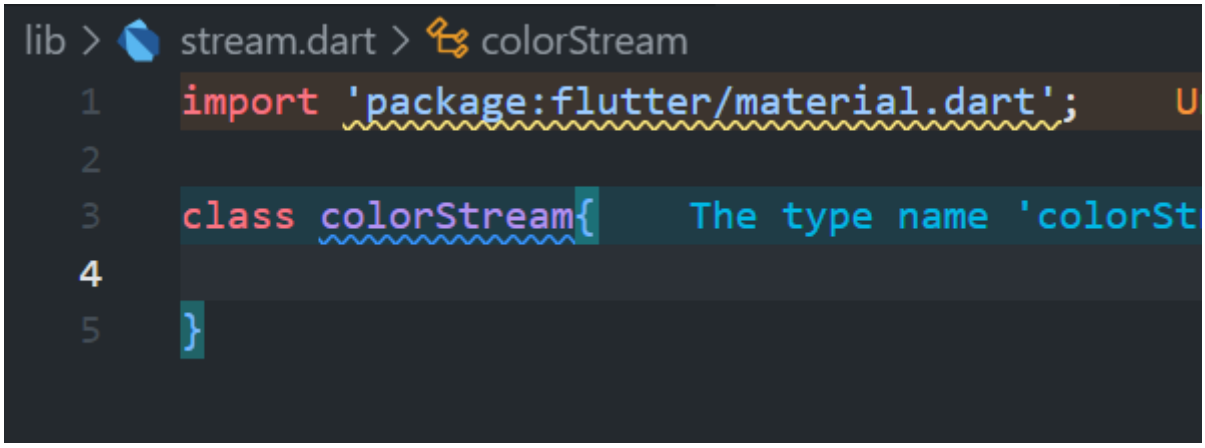
Pertemuan 13

Praktikum 1

1. Buka file main.dart

```dart
        // tested with just a hot reload.
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ), // ThemeData
      home: const StreamHomePage(),
    ); // MaterialApp
  }
}


class StreamHomePage extends StatefulWidget{
  const StreamHomePage({super.key});

  @override
  State<StreamHomePage> createState() => _StreamHomePageState();
}

class _StreamHomePageState extends State<StreamHomePage> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

2. Buat file baru stream.dart



3. Buat variabel colors

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

class colorStream {    The type name 'colorS
  final List<Color> colors = [
    Colors.blueGrey,
    Colors.amber,
    Colors.deepPurple,
    Colors.lightBlue,
    Colors.teal,

  ];
}
```

4. Tambahkan method getColors()

```dart
  Stream<Color> getColor() async*{

  }
```

5. Tambah perintah yield*

```dart
Stream<Color> getColor() async* {
  yield* Stream.periodic(const Duration(seconds: 1), (int t) {
    int index = t % colors.length;
    return colors[index];
  }); // Stream.periodic
}
```

6. Buka main.dart

```dart
import 'stream.dart';
```

7. Tambah variabel

```dart
Color bgColor = Colors.blueGrey;
late ColorStream colorStream;
```

8. tambahkan method changeColor

```
53
54    void changeColor() async{
55      await for (var eventColor in colorStream.getColor()){
56        setState(() {
57          bgColor = eventColor;
58        });
59      }
60    }
```

9. Lakukan override

```
  @override
  void initState() {
    colorStream = ColorStream();
    changeColor();
    super.initState();
  }
```

10. Ubah isi scaffold()

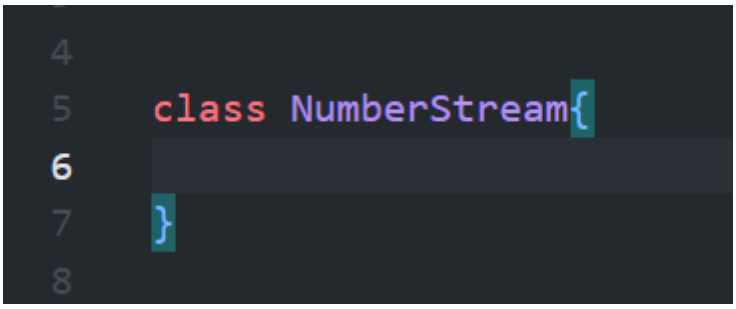
```
65
66    @override
67    Widget build(BuildContext context) {
68      return Scaffold(
69          appBar: AppBar(
70            title: const Text("Stream"),
71          ), // AppBar
72          body: Container(
73            decoration: BoxDecoration(color: bgColor),
74          )); // Container // Scaffold
75    }
76  }
77
```

Praktikum 2

1. Buka file Stream.dart

```
1  import 'package:flutter/material.dart';
2  import 'dart:async';
```

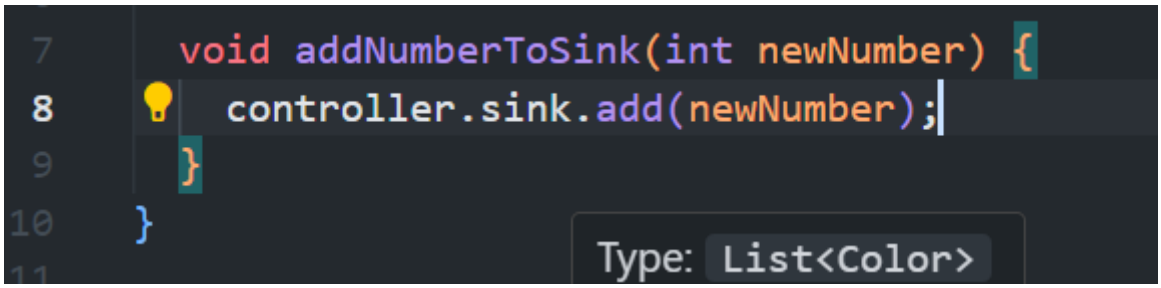
2. Tambah class NumberStream



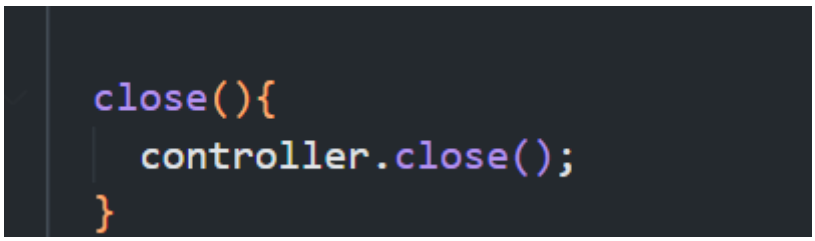
3. Tambah StreamController

```
class NumberStream {
  final StreamController<int> controller = StreamController<int>();
```

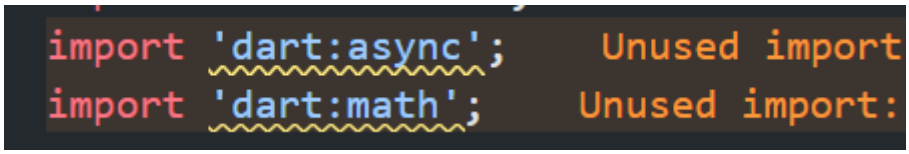
4. Tambah method addNumberToSink



5. Tambah method Close()



6. buka main.dart



7. Tambah variabel

```
class _StreamHomePageState extends State<StreamHomePage>
  int lastNumber =0;
  late StreamController numberStreamController;
  late NumberStream numberStream;
```

8. Edit initState()

```
  // changeColor();
  stream.listen((event){
    setState(() {
      lastNumber = event;
    });
  });
```

9. Edit dispose()

```dart
@override
void dispose() {
  numberStreamController.close();
  super.dispose();
}
```

10. Tambah method addRandomNumber()

```dart
void addRandomNumber(){
  Random random = Random();
  int myNum = random.nextInt(10);
  numberStream.addNumberToSink(myNum);
}
```

11. Edit method build

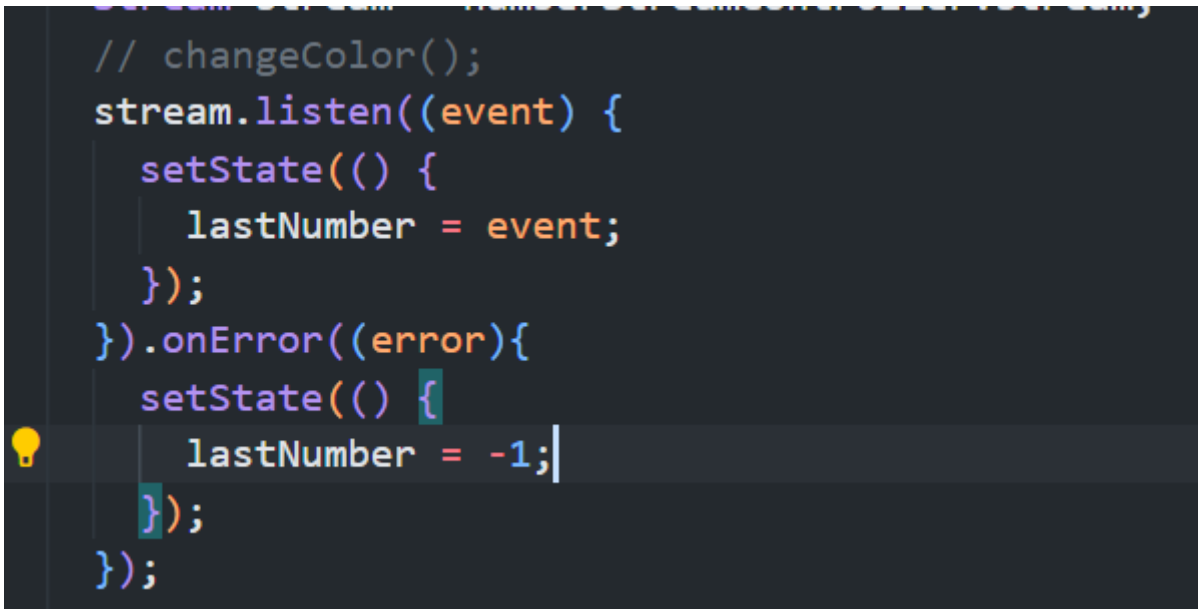
```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(
        title: const Text("Stream"),
      ), // AppBar
      body: SizedBox(
        width: double.infinity,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Text(lastNumber.toString()),
            ElevatedButton(
                onPressed: () => addRandomNumber(),
                child: Text('New Random Number')) // ElevatedButton
          ],
        ), // Column
      )); // SizedBox // Scaffold
}
}
```

12. run

13. buka stream.dart

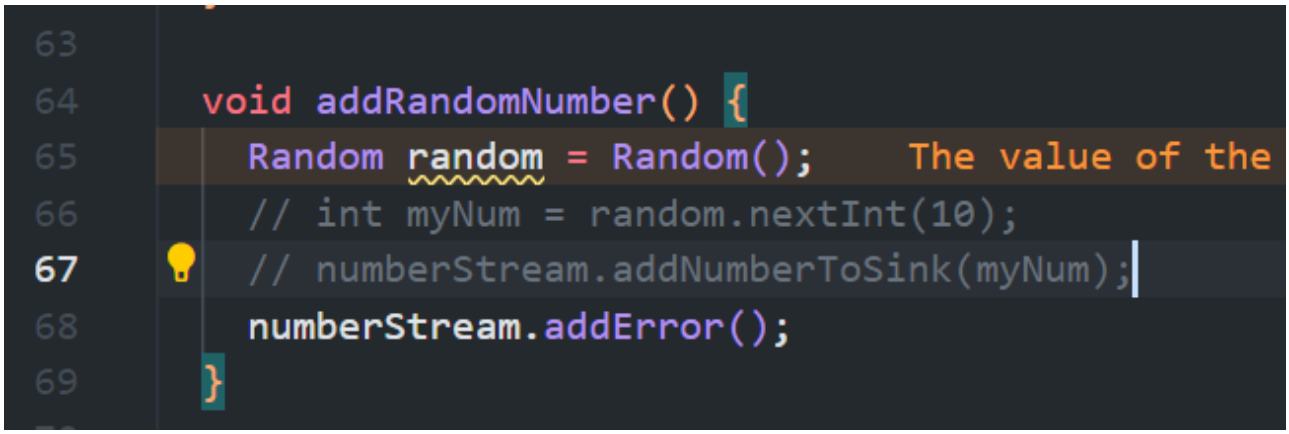
```
addError(){
  controller.sink.addError('error');
}
```

14. Buka main.dart



```
// changeColor();
stream.listen((event) {
  setState(() {
    lastNumber = event;
  });
}).onError((error){
  setState(() {
    lastNumber = -1;
  });
});
```

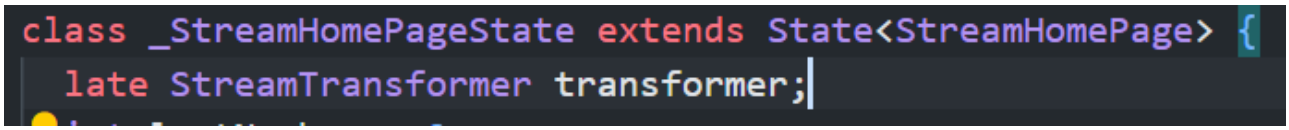
15. Edit method addRandomNumber()



```
void addRandomNumber() {
  Random random = Random();    The value of the
  // int myNum = random.nextInt(10);
  // numberStream.addNumberToSink(myNum);
  numberStream.addError();
}
```

Praktikum 3

1. buka main.dart



```
class _StreamHomePageState extends State<StreamHomePage> {
  late StreamTransformer transformer;
```

2. Tambahkan kode ini di initstate

```
    transformer = StreamTransformer<int, int>.fromHandlers(
        handleData: (value, sink) {
          sink.add(value * 10);
        },
        handleError: (error, trace, sink) {
          sink.add(-1);
        },
        handleDone: (sink) => sink.close());
    super.initState();
  }
```

3. Tetap di initState

```
    // changeColor();
    stream.transform(transformer).listen((event) {
      setState(() {
        lastNumber = event;
      });
    }).onError((error) {
      print(error);    Don't invoke 'print' in production code.⏎Try u
      setState(() {
        lastNumber = -1;
      });
    });
```

4. Run

Praktikum 4

1. Tambah variabel

```
  late StreamSubscription subscription;
```

2. Edit initState()

```
    subscription = stream.listen((event){
      setState(() {
        lastNumber = event;
      });
    });
```

3. tetap initState()

```
subscription.onError((error){
  setState(() {
    lastNumber = -1;
  });
});
```

4. tambah properti onDone()

```
subscription.onDone((){
  print('OnDone was called');
});
```

5. Tambah method baru

```
void stopStream() {
  numberStreamController.close();
}
```

6. Pindah ke method dispose

```
@override
void dispose() {
  numberStreamController.close();
  subscription.cancel();
  super.dispose();
}
```

7. Pindah ke method build

```
ElevatedButton(
  onPressed: () => stopStream(),
  child: const Text('Stop Subscription')) //
```

8. Edit method addRandomNumber

```dart
void addRandomNumber() {
  Random random = Random();
  int myNum = random.nextInt(10);
  if (!numberStreamController.isClosed) {
    numberStream.addNumberToSink(myNum);
  } else {
    setState(() {
      lastNumber = -1;
    });
  }

  // numberStream.addError();
}
```

9. run

```
D/EGL_emulation( 3225): app_time_stats:
I/flutter ( 3225): OnDone was called
D/EGL_emulation( 3225): app_time_stats:
```

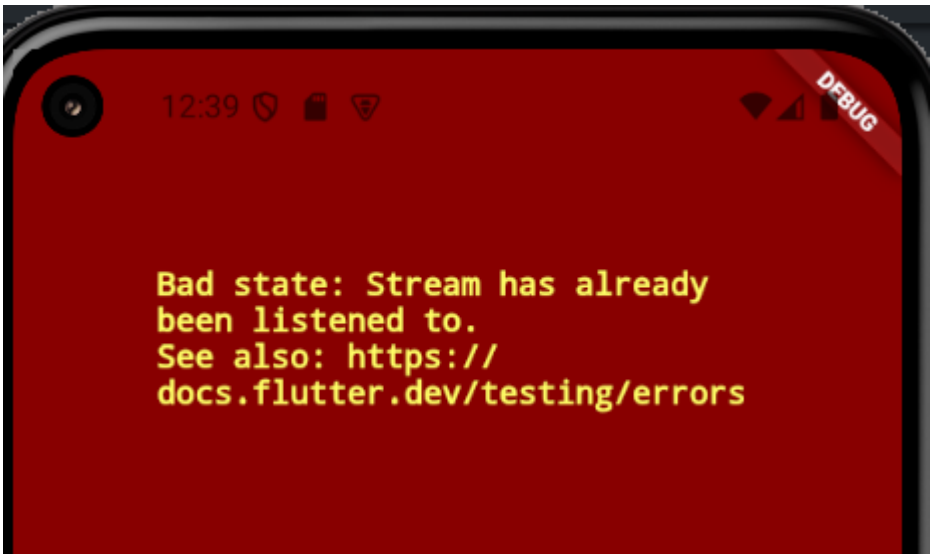
Praktikum 5

1. buka file main.dart

```dart
late StreamSubscription subscription2;
String values = '';
```

2. Edit initState()

```dart
subscription = stream.listen((event) {
  setState(() {
    values += '$event - ';
  });
});

subscription2 = stream.listen((event){
  values += '$event - ';
});
```

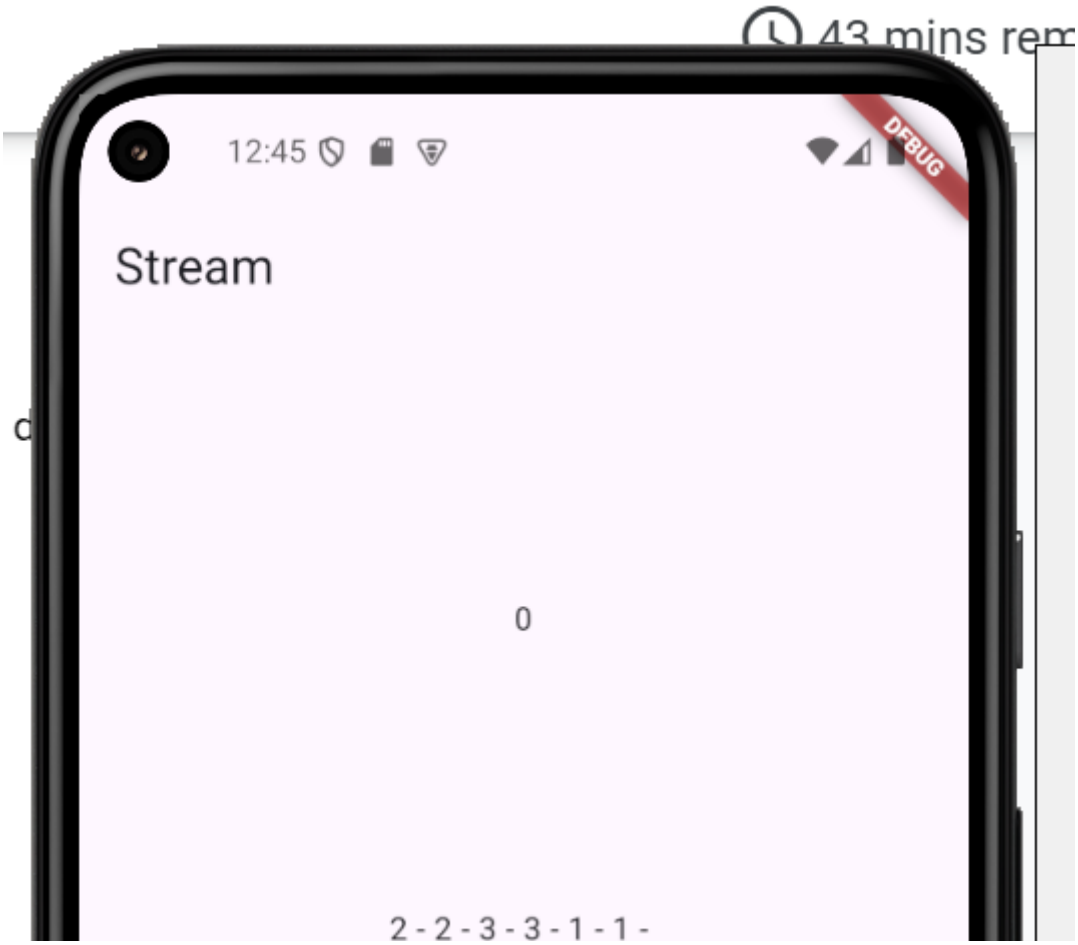
3. run 

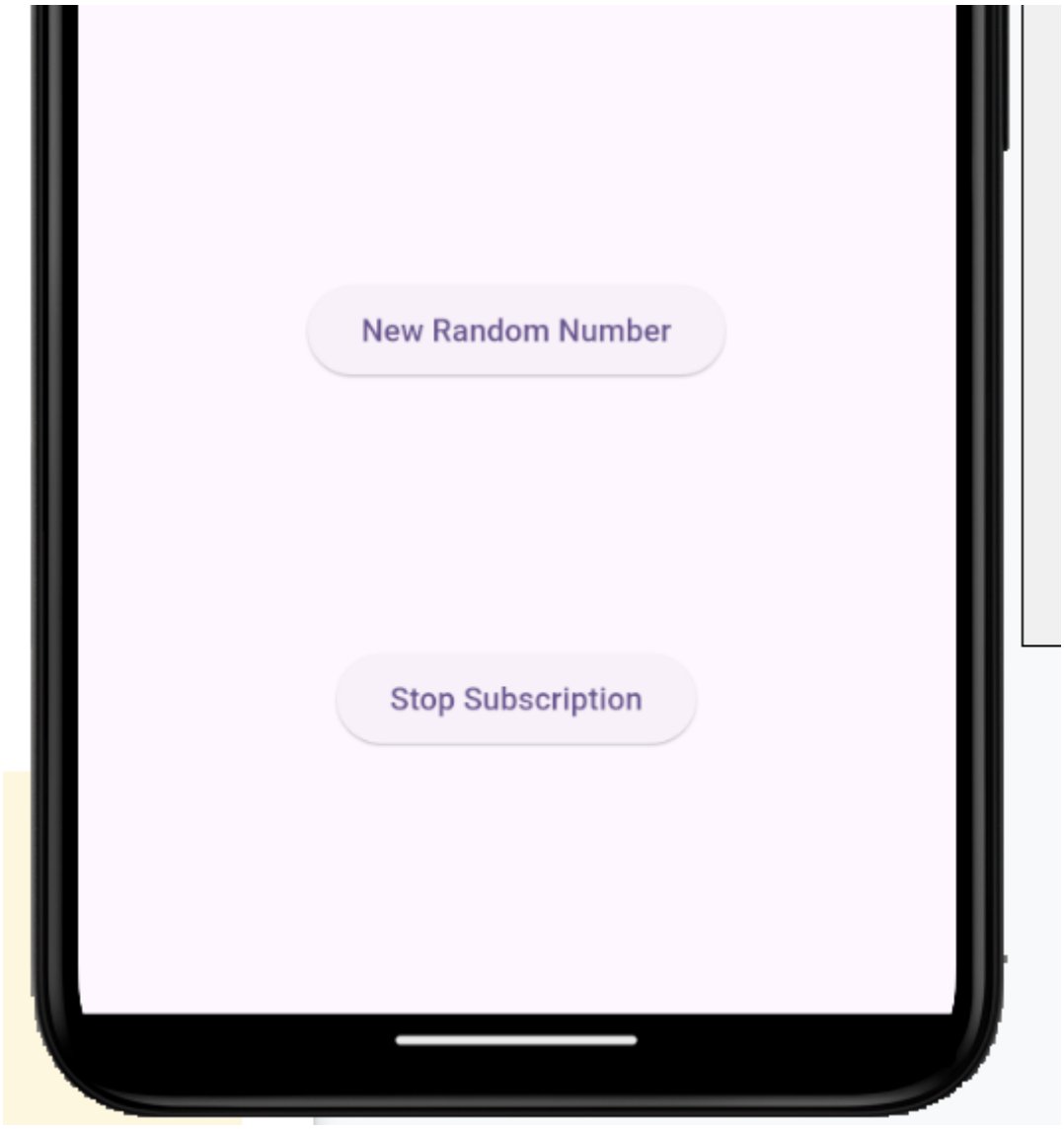
4. Set broadcast stream

```
Stream stream = numberStreamController.stream.asBroadcastStream();
// changeColor();
```

5. Edit method build()

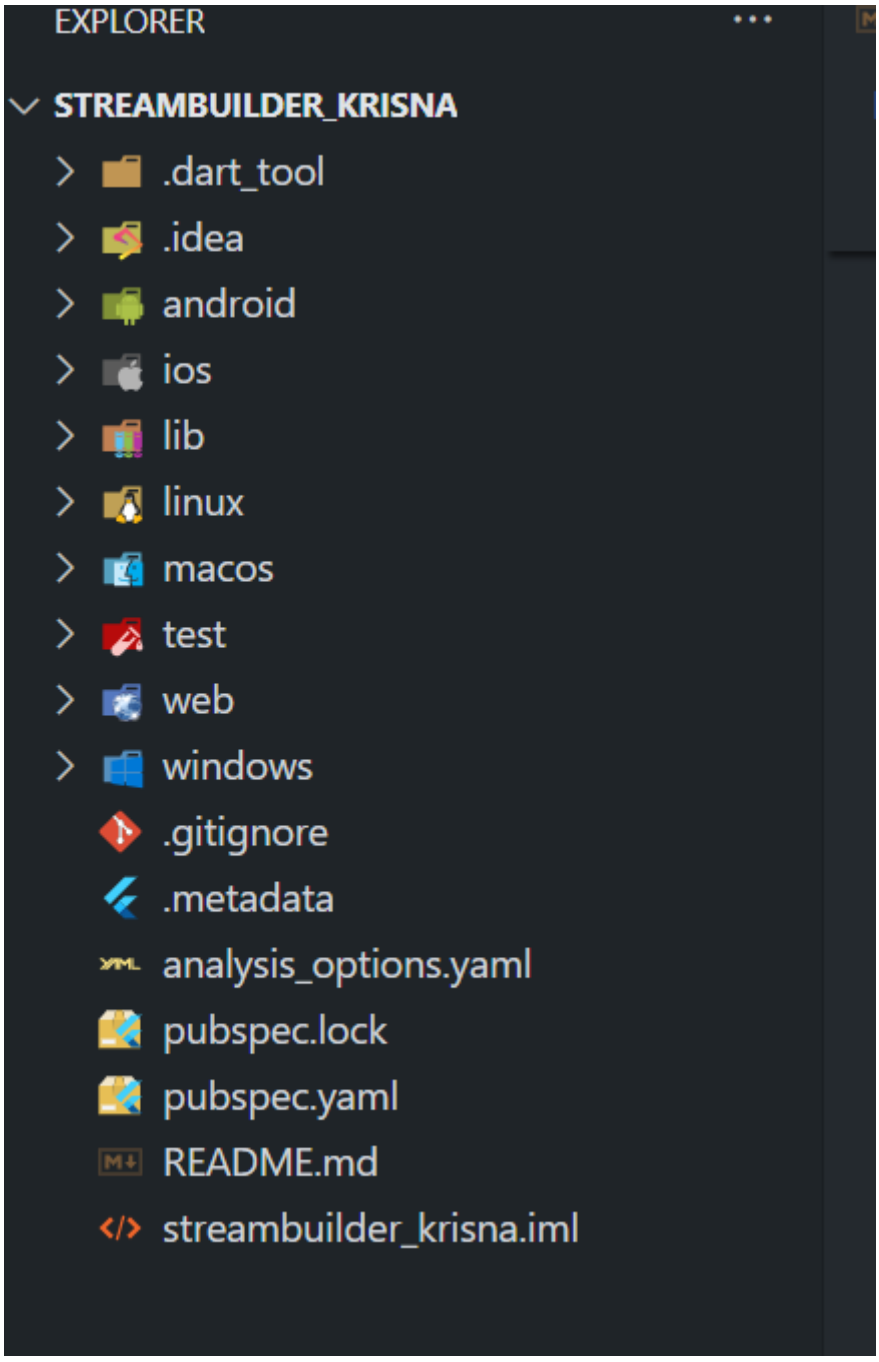
```
body: Column(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [Text(values)],
)); // Column // Scaffold
```



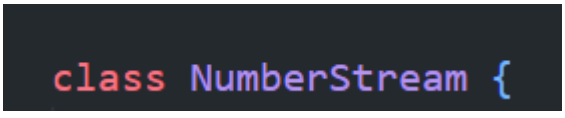
6. Run 

Praktikum 6

1. Buat project baru



2. Buat file baru stream.dart



3. Tetap di file stream.dart

```dart

class NumberStream {
  Stream<int> getNumbers() async* {
    yield* Stream.periodic(const Duration(seconds: 1), (int t) {
      Random random = Random();
      int myNum = random.nextInt(10);
      return myNum;
    }); // Stream.periodic
  }
}

```

4. Edit main.dart

```dart
class StreamHomePage extends StatefulWidget {
  const StreamHomePage({super.key});

  @override
  State<StreamHomePage> createState() => _StreamHomePageState();
}

class _StreamHomePageState extends State<StreamHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Stream'),
      ), // AppBar
      body: Container(),
    ); // Scaffold
  }
}
```

5. Tambah variabel

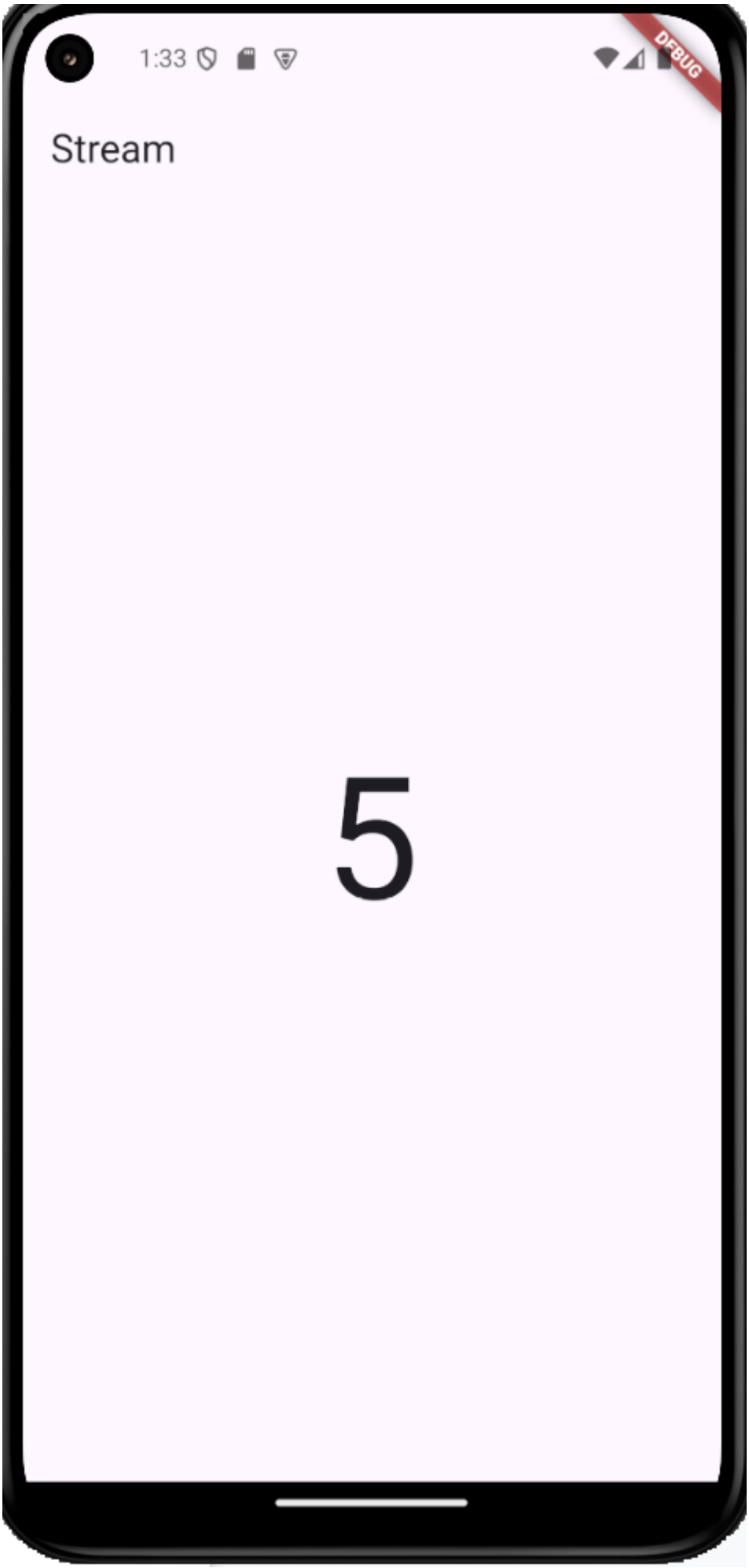
```dart
late Stream<int> numberStream;
```

6. Edit initState()

```dart
@override
void initState() {
  numberStream = NumberStream().getNumbers();
  super.initState();
}
```

7. Edit method build()

```
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text(data: 'Stream'),
        ), // AppBar
        body: StreamBuilder<int>(
            stream: numberStream,
            initialData: 0,
            builder: (BuildContext context, AsyncSnapshot<int> snapshot)
              if (snapshot.hasError) {
                print(object: 'Error');    Don't invoke 'print' in produ
              }

              if (snapshot.hasData) {
                return Center(
                  child: Text(
                    data: snapshot.data.toString(),
                    style: const TextStyle(fontSize: 96),
                  ), // Text
                ); // Center
              } else {
                return const SizedBox.shrink();
              }
            })); // StreamBuilder // Scaffold
  }
}
```



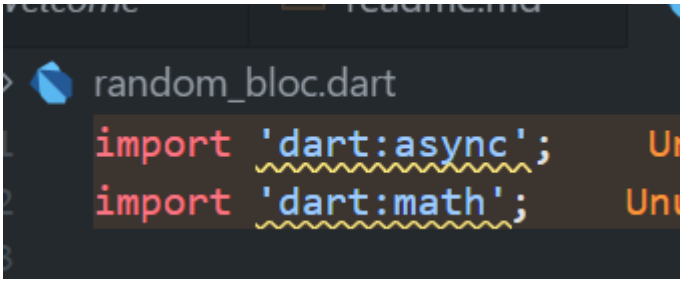
8. Run

Praktikum 7

1. Buat project baru

```
PS D:\Kuliahe wong jenius\Semester 5\Mobile\Pertemuan 13> flutter create bloc_random_krisna
Creating project bloc_random_krisna...
Resolving dependencies in `bloc_random_krisna`...
Downloading packages...
Got dependencies in `bloc_random_krisna`.
Wrote 129 files.
```

2. isi kode random_bloc.dart



3. Buat class RandomNumberBloc()

```
class RandomBloc {
```

4. Buat variabelStreamController

```
class RandomBloc {
  final _generateRandomController = StreamController<void>();
  final _randomNumber
  Sink<void> get gene

  Stream<int> get ran
```

```
(new) Random Random([int? seed])
dart:math
Creates a random number generator.
```

5. Buat constructor

```
RandomBloc(){
  _generateRandomController.stream.listen((_){
    final random = Random().nextInt(10);
    _randomNumberController.sink.add(random);
  });
}
```

6. buat method dispose()

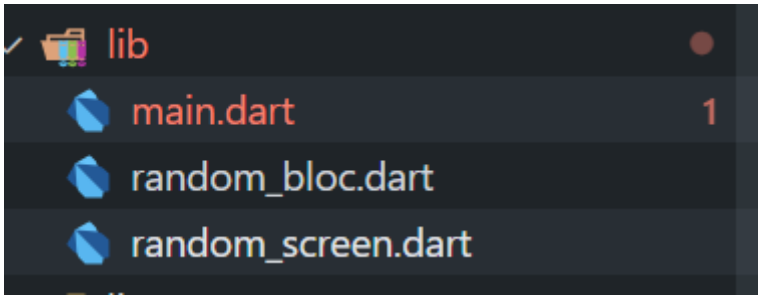
```
void dispose(){
  _generateRandomController.close();
  _randomNumberController.close();
}
```

7. Edit main.dart

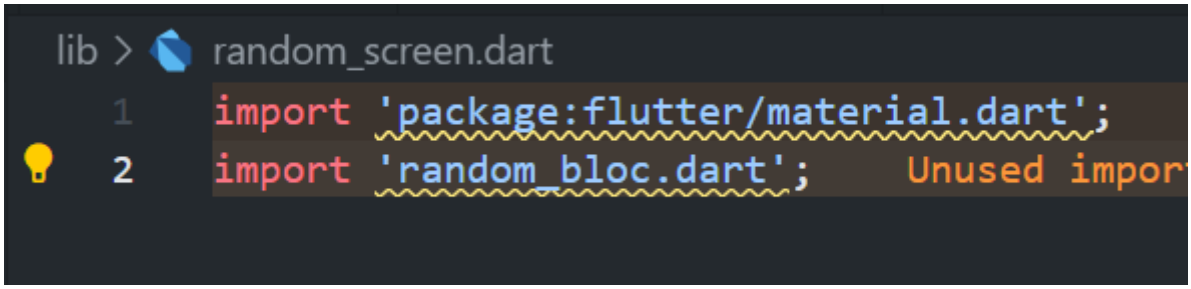
```dart
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // TRY THIS: Try running your application with "flutter r
        // the application has a purple toolbar. Then, without qu
        // try changing the seedColor in the colorScheme below to
        // and then invoke "hot reload" (save your changes or pre
        // reload" button in a Flutter-supported IDE, or press "r
        // the command line to start the app).
        //
        // Notice that the counter didn't reset back to zero; the
        // state is not lost during the reload. To reset the stat
        // restart instead.
        //
        // This works for code too, not just values: Most code ch
        // tested with just a hot reload.
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.dee
        useMaterial3: true,
      ), // ThemeData
      home: const RandomScreen(),    The name 'RandomScreen' isn'
    ); // MaterialApp
  }
}
```

```
context BuildContext
AboutDialog
AboutListTile
AbsorbPointer
Accumulator
Action
ActionChip
```

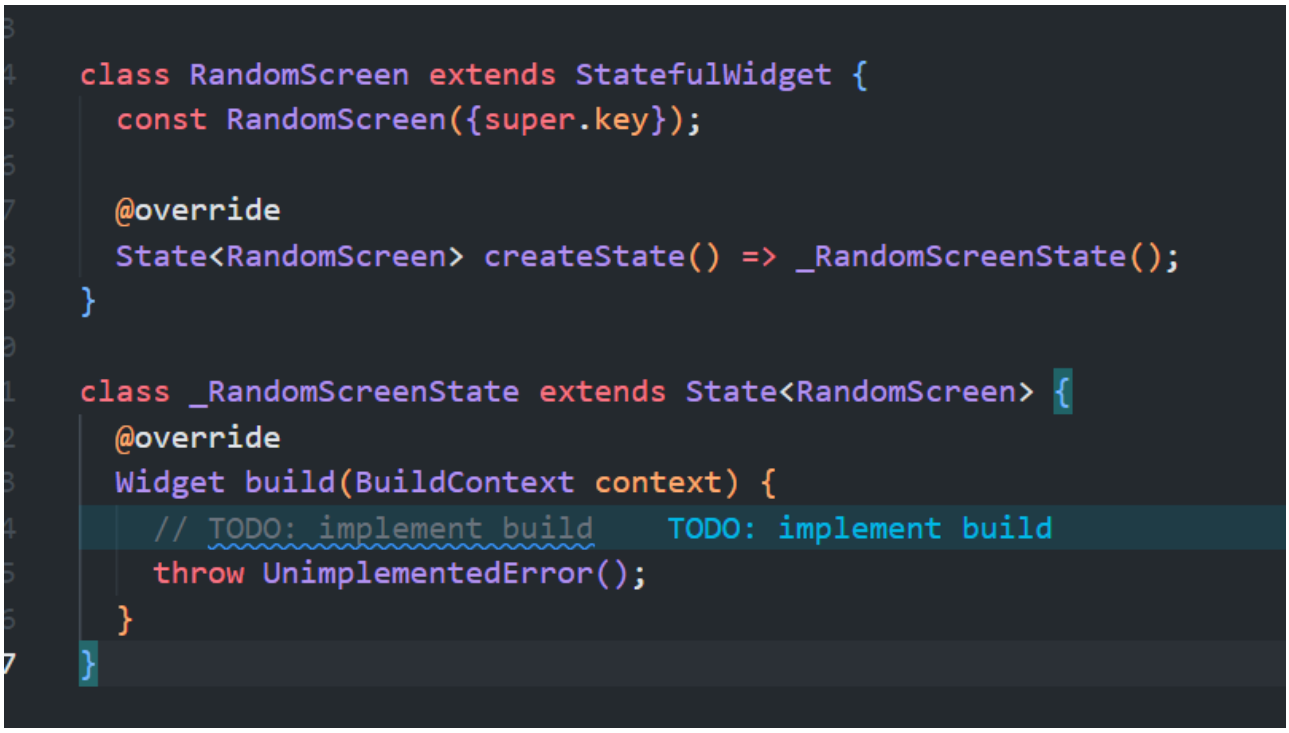
8. Buat file baru random_screen



9. Lakukan import material dan random_bloc.dart

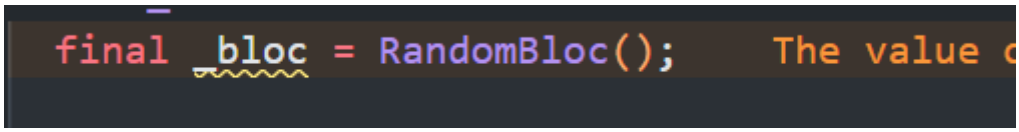


10. Buat statefull widget randomscreen



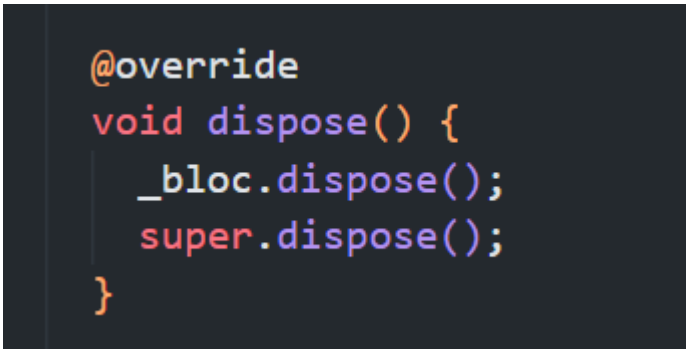
```dart
class RandomScreen extends StatefulWidget {
  const RandomScreen({super.key});

  @override
  State<RandomScreen> createState() => _RandomScreenState();
}

class _RandomScreenState extends State<RandomScreen> {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build    TODO: implement build
    throw UnimplementedError();
  }
}
```

11. Buat variabel



```dart
final _bloc = RandomBloc();    The value
```

12. Buat method dispose()



```dart
@override
void dispose() {
  _bloc.dispose();
  super.dispose();
}
```

13. Edit method build

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Random Number'),
    ), // AppBar
    body: Center(
      child: StreamBuilder<int>(
          stream: _bloc.randomNumber,
          builder: (context, snapshot) {
            return Text(
              'Random numberv : ${snapshot.data}',
              style: const TextStyle(fontSize: 24),
            ); // Text
          }), // StreamBuilder
    ), // Center
    floatingActionButton: FloatingActionButton(
      onPressed: () => _bloc.generateRandom.add(null),
      child: const Icon(Icons.refresh),
    ), // FloatingActionButton
  ); // Scaffold
}
}
```

Soal 1

- Tambahkan nama panggilan Anda pada title app sebagai identitas hasil pekerjaan Anda.

```dart
@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Stream krisna',
    theme: ThemeData(
```

Soal 2

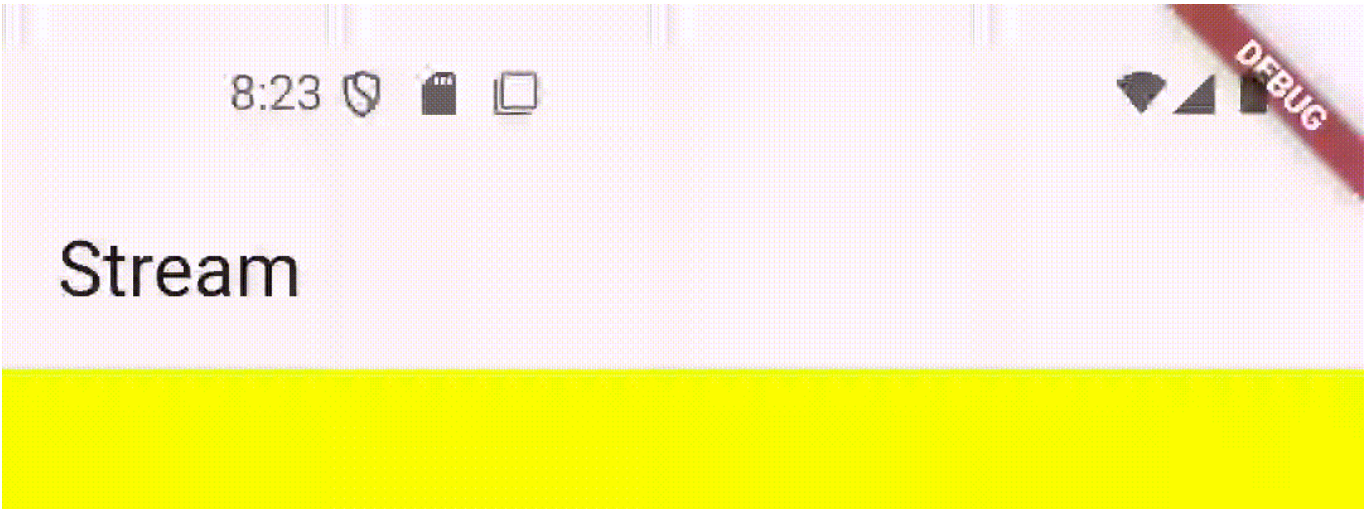
- Tambahkan 5 warna lainnya sesuai keinginan Anda pada variabel colors tersebut.

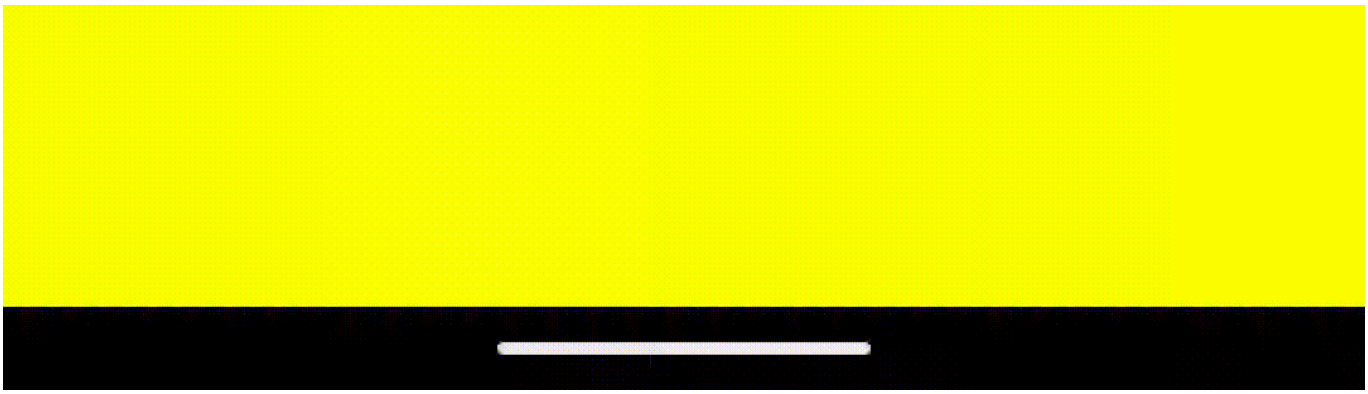
```dart
import 'package:flutter/material.dart';

class colorStream {
  final List<Color> colors = [
    Colors.blueGrey,
    Colors.amber,
    Colors.deepPurple,
    Colors.lightBlue,
    Colors.teal,
    Colors.orange,
    Colors.cyan,
    Colors.teal.shade400,
    Colors.deepOrangeAccent,
    Colors.lime,
    Colors.yellowAccent
  ];
}
```

Soal 3

- Jelaskan fungsi keyword `yield*` pada kode tersebut! `yield*` digunakan untuk mengembalikan banyak data yang biasanya juga dari method stream
- Apa maksud isi perintah kode tersebut? Mengembalikan nilai dari sebuah fungsi stream periodic dengan periode waktu 1 second , dengan dan sebuah function yang mereturn list colors dengan index

soal 4





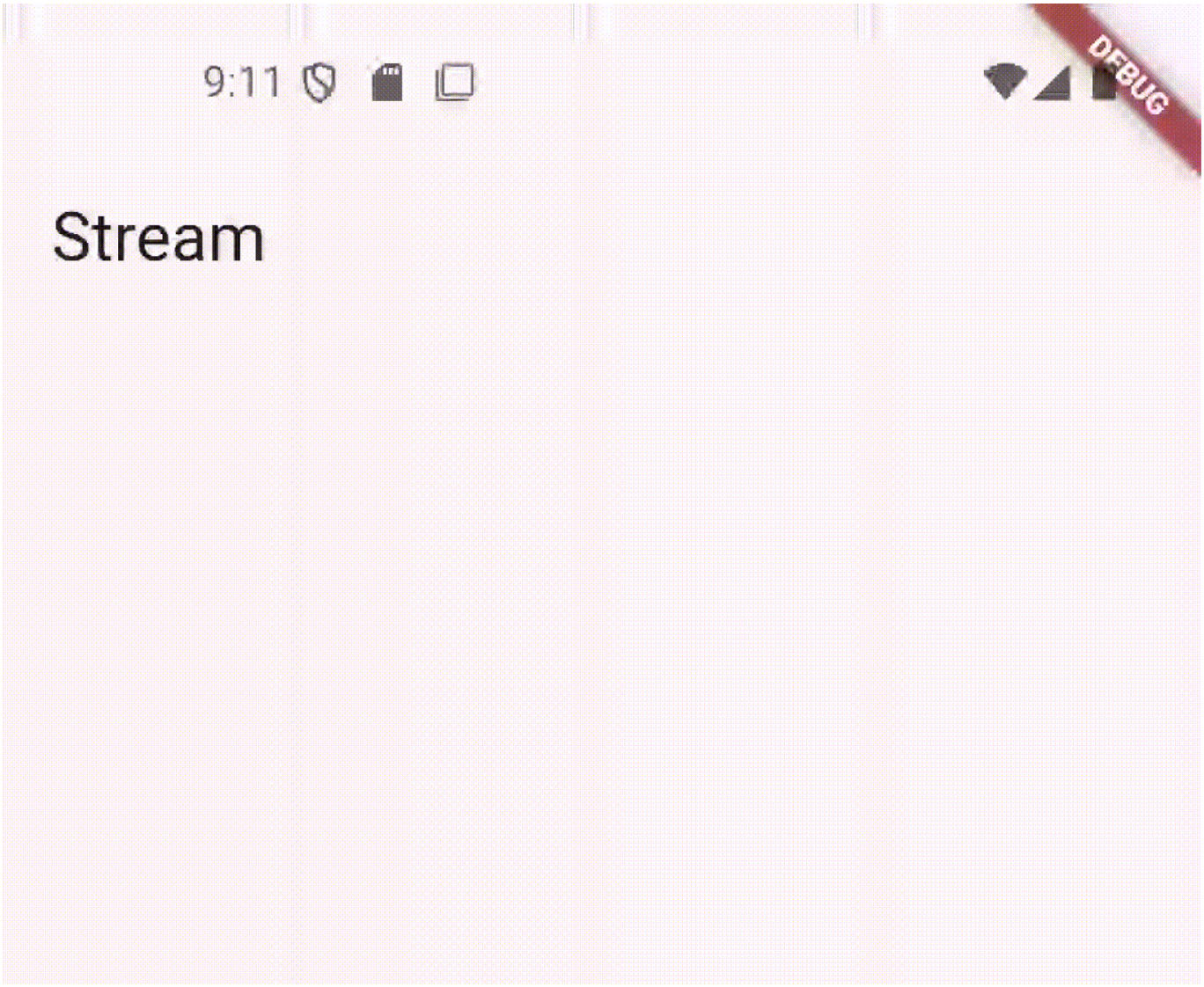
soal 5

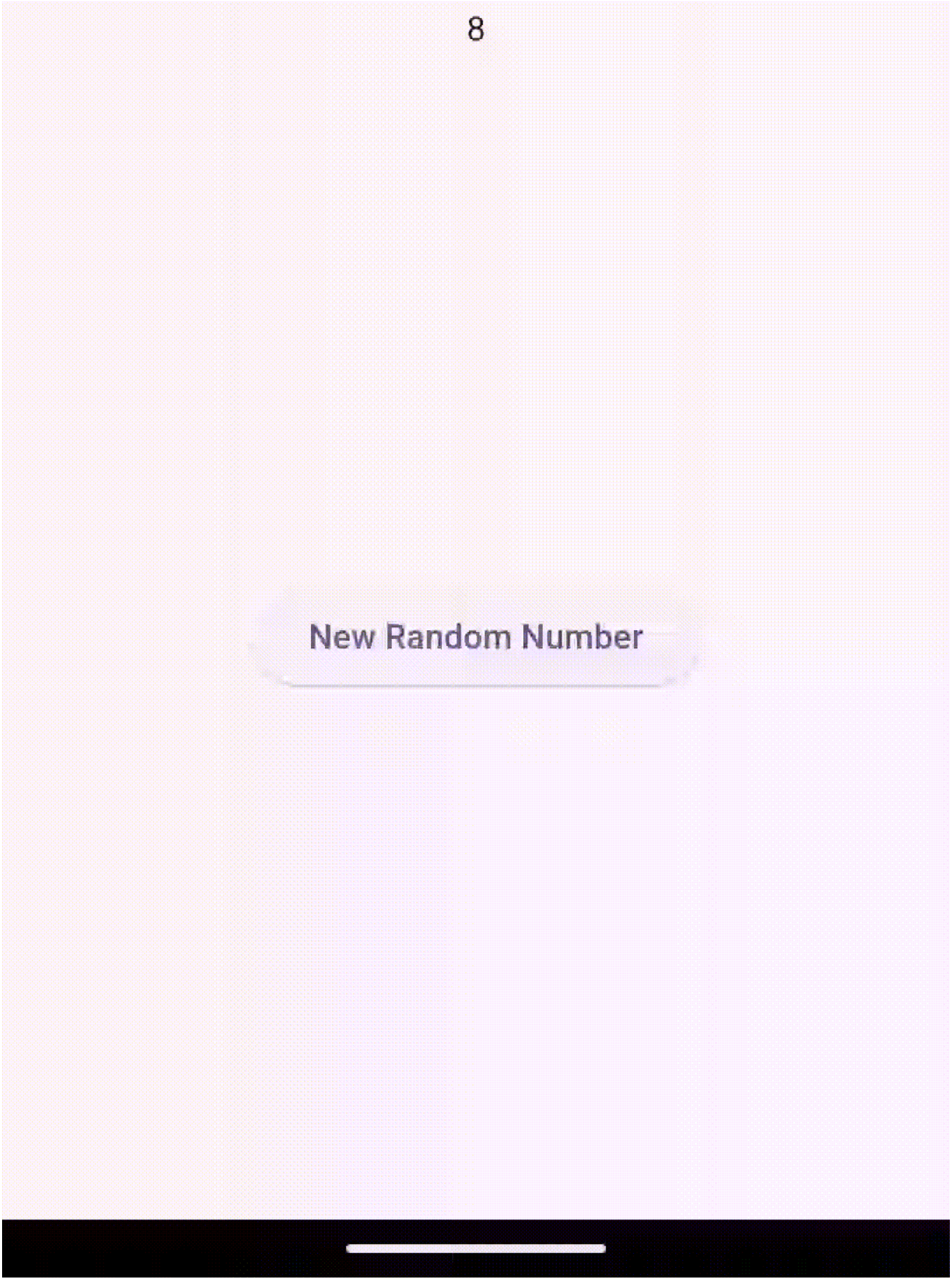
- Jelaskan perbedaan menggunakan listen dan await for (langkah 9) ! `listen` = digunakan untuk mendapatkan setiap data dari stream `await for` = digunakan untuk mengiterate untuk setiap objek dengan menunggu setiap proses

soal 6

- Jelaskan maksud kode langkah 8 dan 10 tersebut!

  > langkah 8 init state digunakan untuk pendeklarasian dari class stream dan controller langkah 10 Digunakan untuk menambahkan nomer secara dan menambahkan ke sink





soal 7

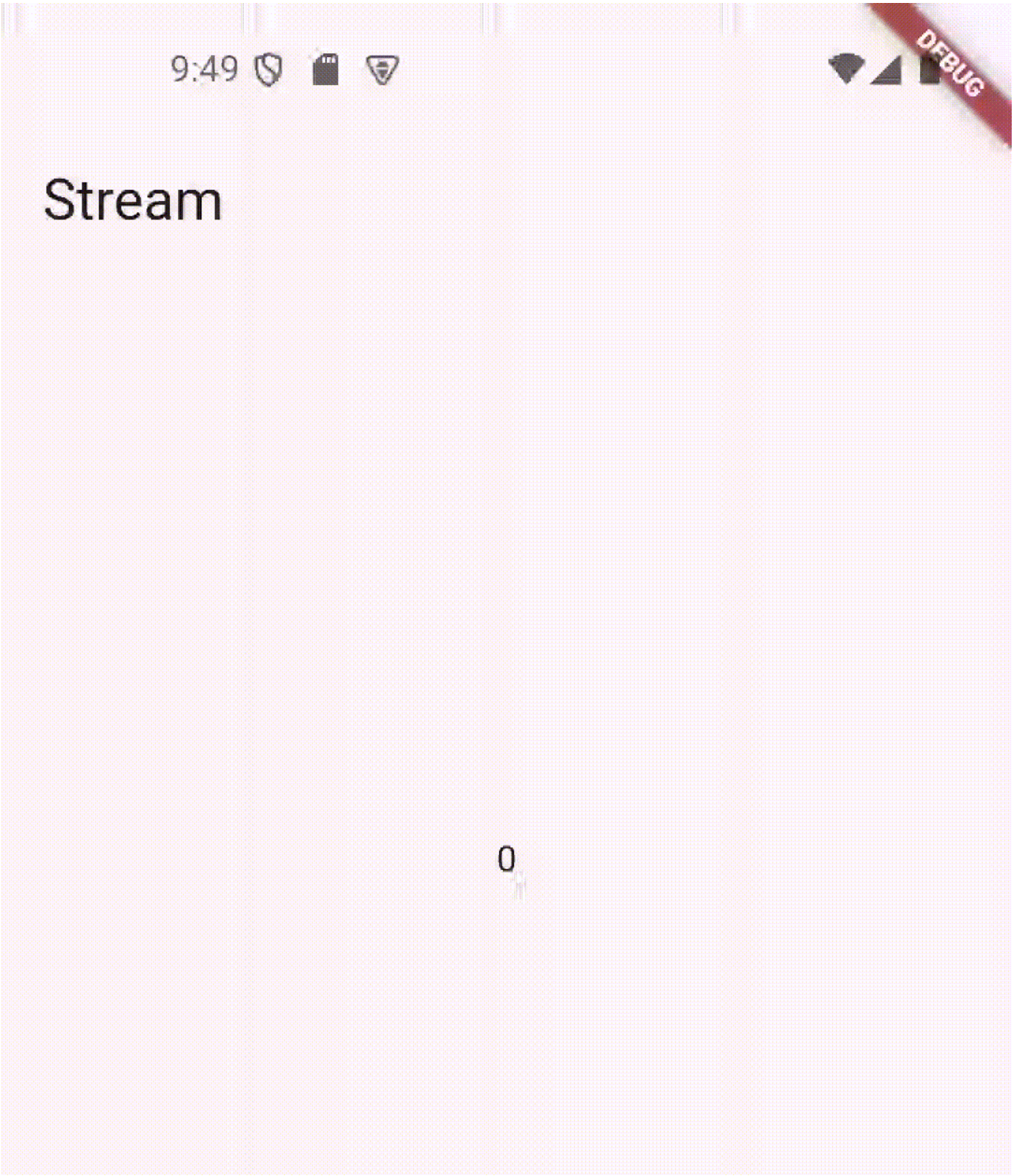
- jelaskan kode langkah 13 sampai 15 tersebut!

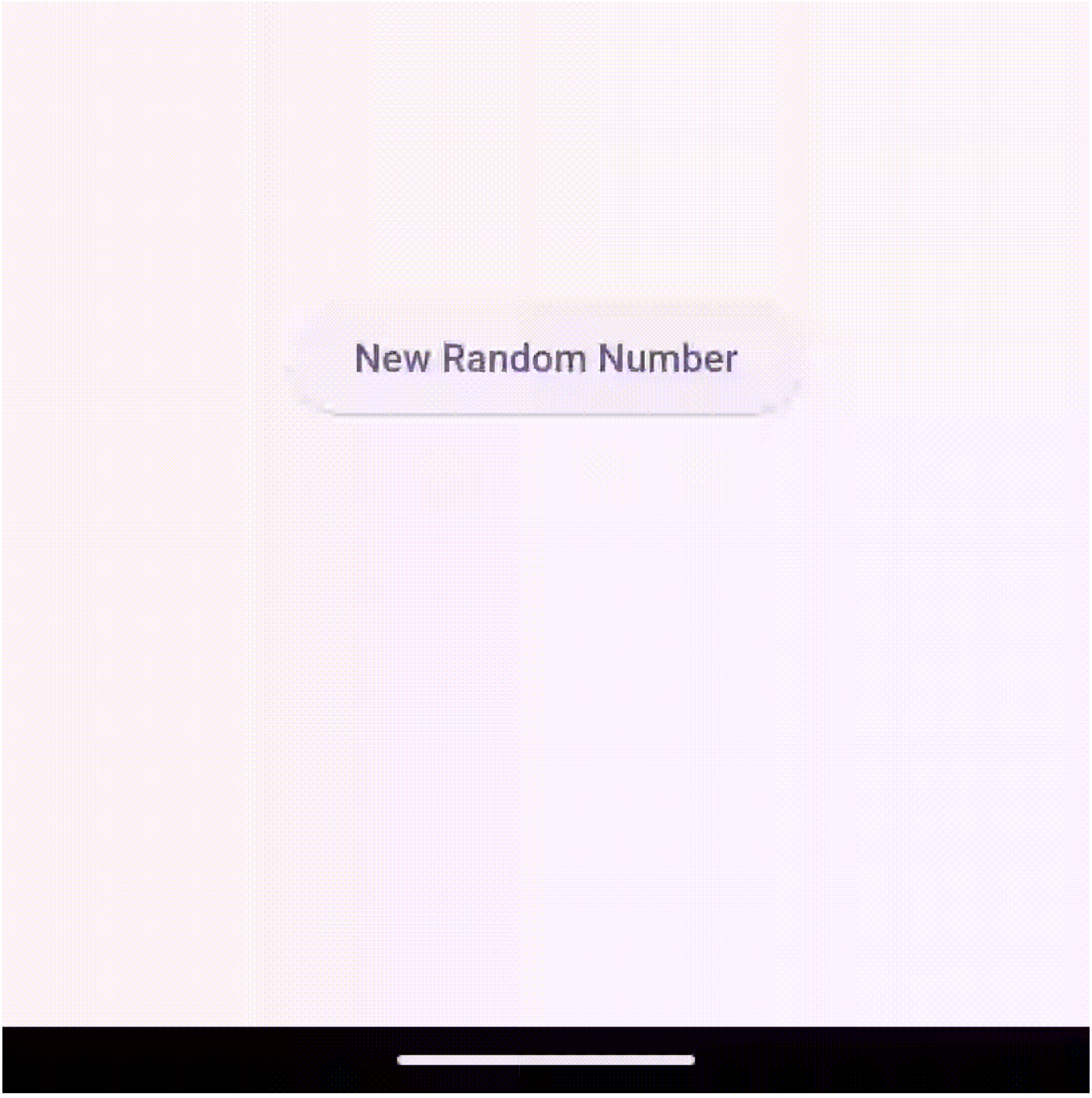
> pada langkah 13 menambahkan function untuk menghandel error pada langkah 14 menambahkan main menambahkan handling pada initstate pada langkah 15 menjadikan error karena tidak ada data yang diinputkan, hanya menambahkan onError

soal 8

- Jelaskan maksud kode langkah 1-3 tersebut!

> pada langkah 1 mendeklarasikan variabel StreamTransformer pada langkah 2 menambahkan handle data yaitu setiap data akan dikalikan 10 dari hasil random number pada langkah 3 mengubah variabel lastnumber dengan event dari StreamTransformer

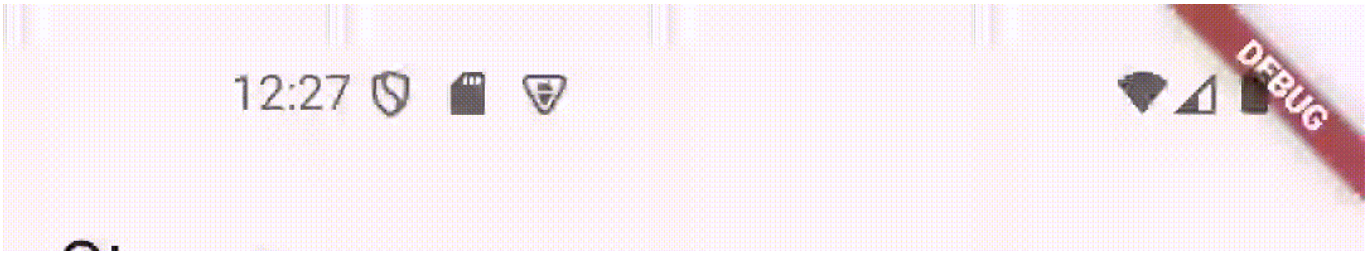


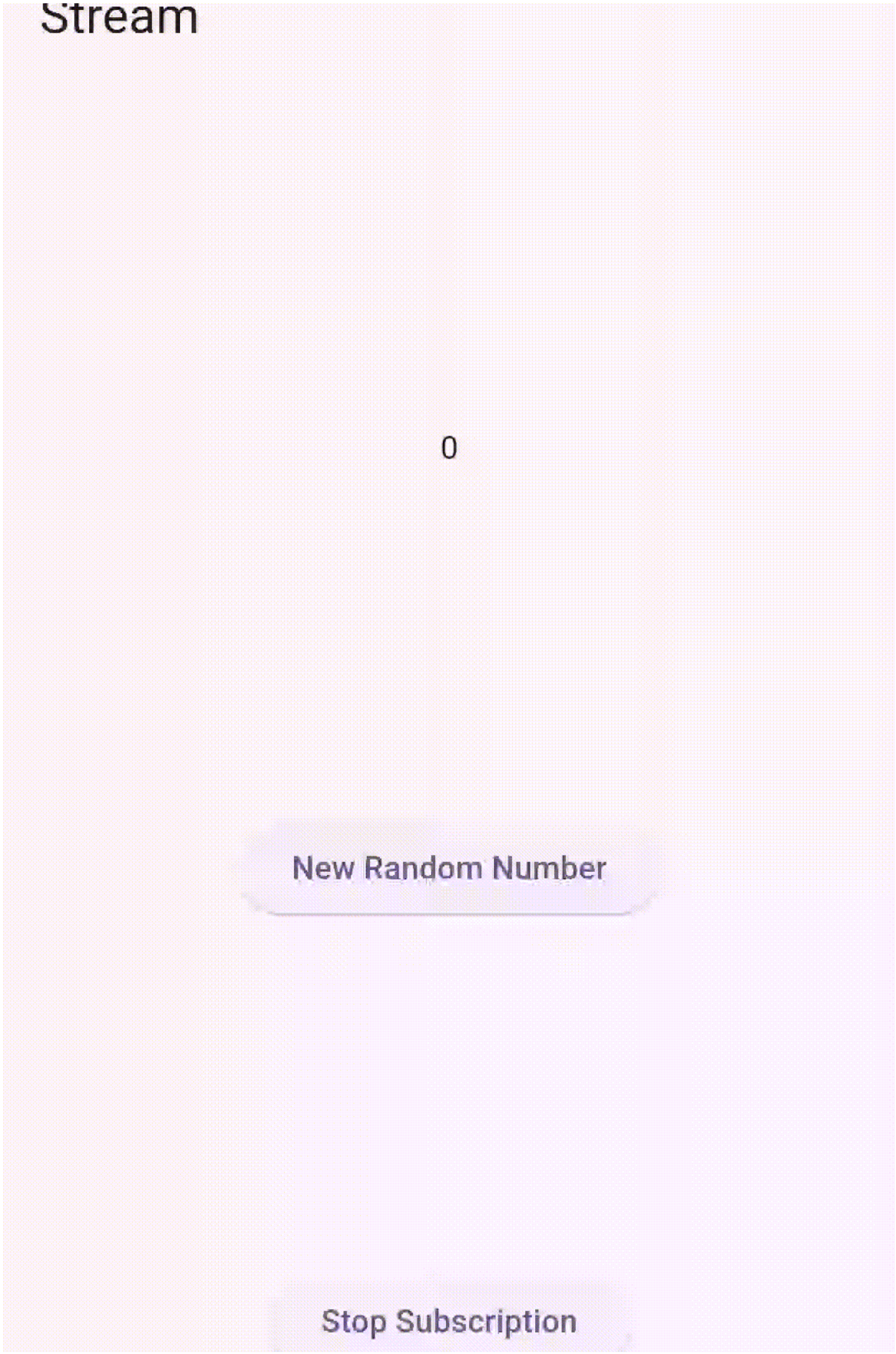


soal 9

- Jelaskan maksud kode langkah 2, 6 dan 8 tersebut!

> pada langkah 2 menambahkan subscription yang listen terhadap event dari stream lalu mengubah last number dengan event atau data dari stream pada langkah 6 dispose digunakan ketika stream sudah tidak terpakai, maka subscription akan di cancel pada langkah 8 terjadi pengecekan apabila controller masih dalam keadaan terbuka maka masih bisa menambahkan data ke stream, namun apabila sudah ditutup maka akan mengganti angka dengan -1



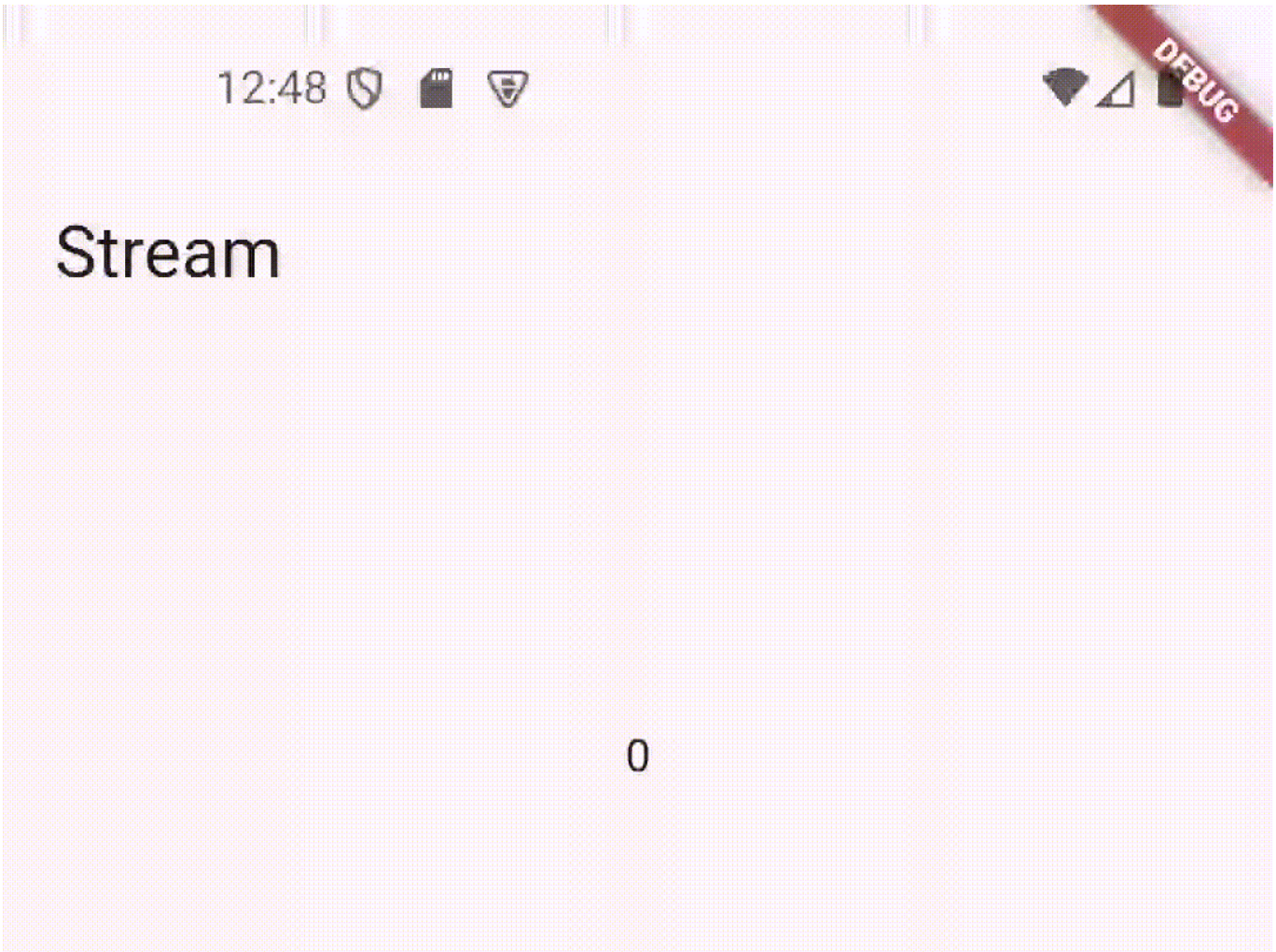

Stream
0
New Random Number
Stop Subscription

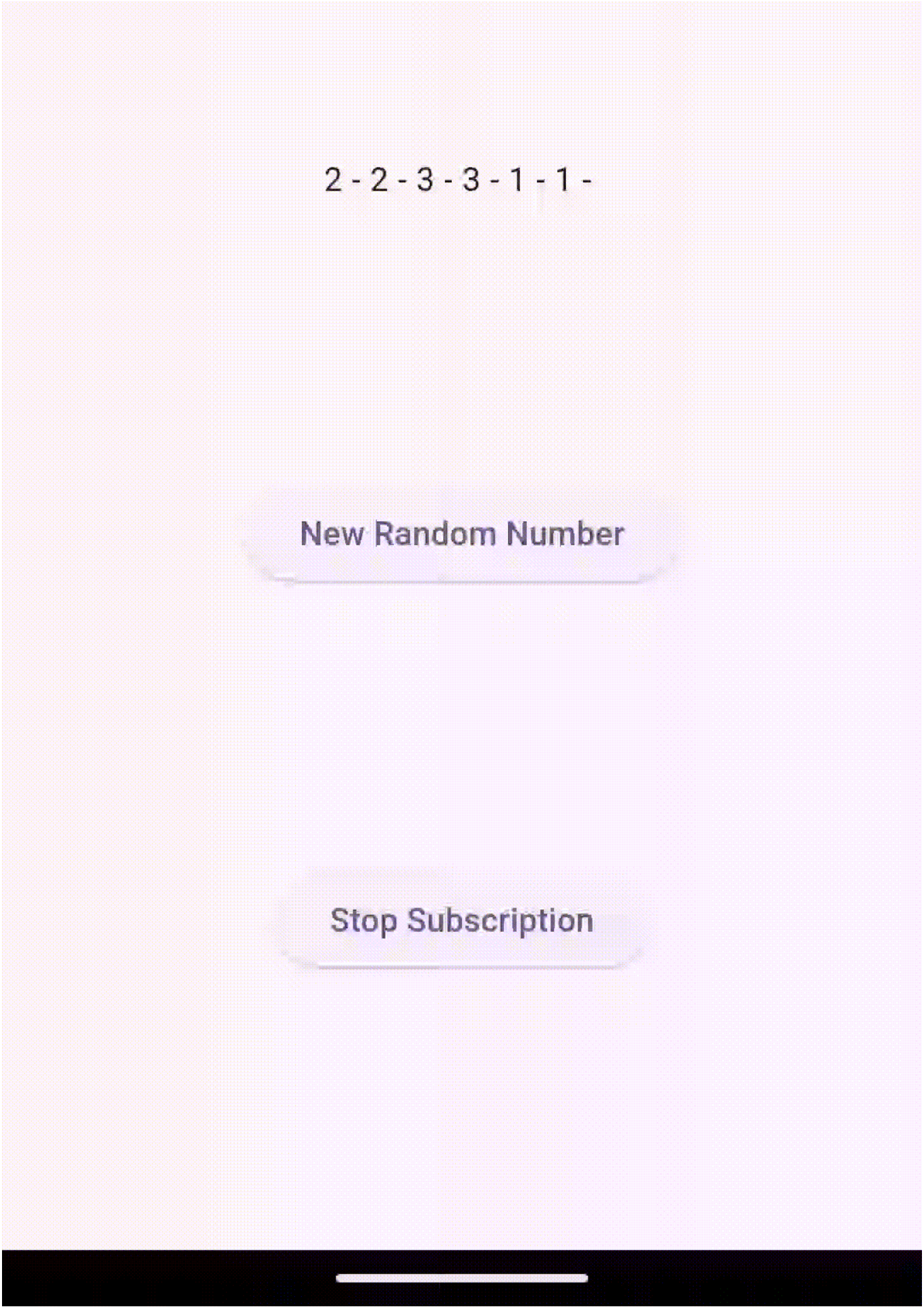
Soal 10

> Karena kedua subs mendengarkan pada streamController yang sama

Soal 11

> Karena terjadi broadcasting, sehinga subscription menjadi bekerja 2 kali untuk subs 1 dan subs 2 sehingga terjadi dobel angka

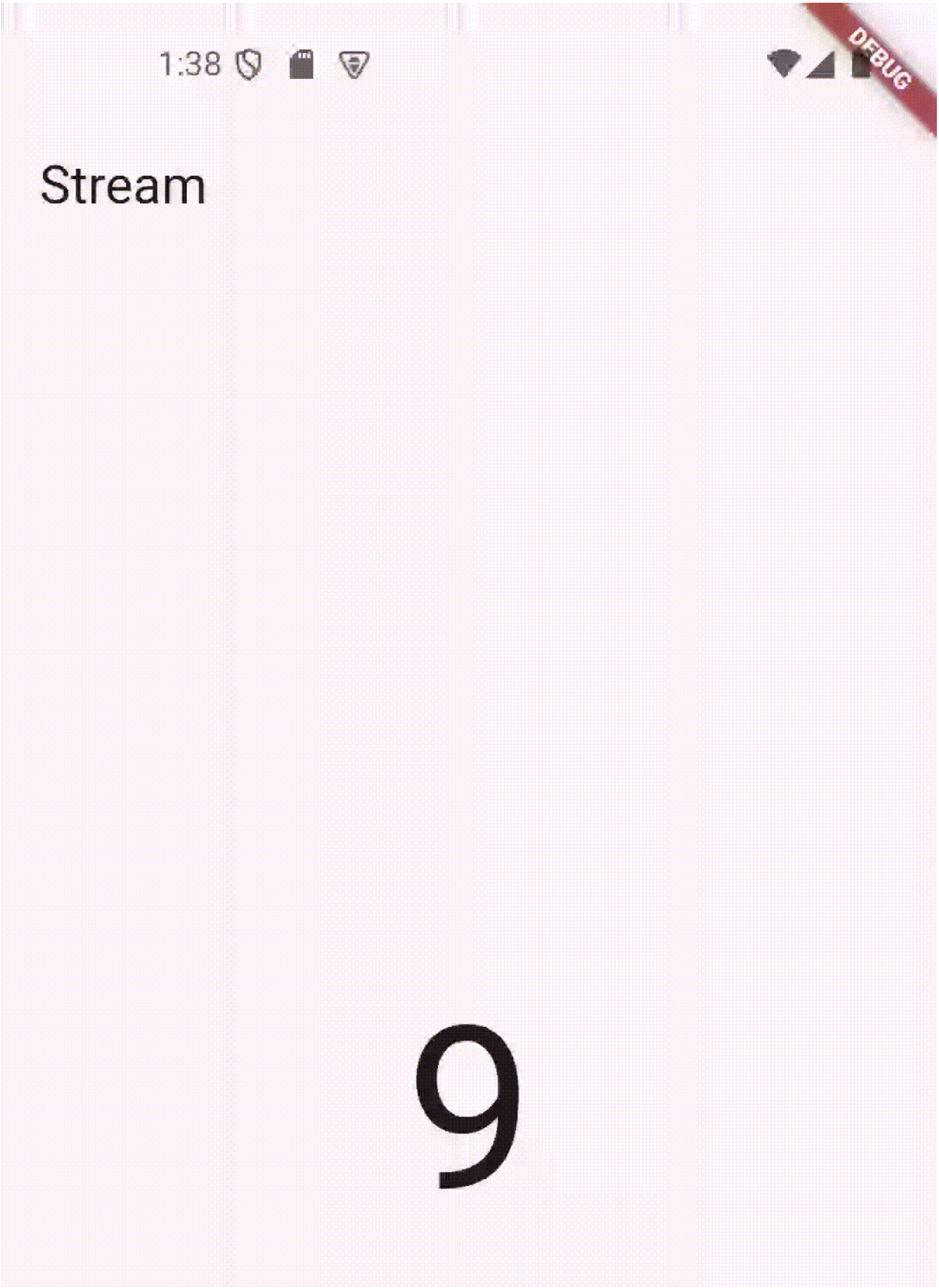


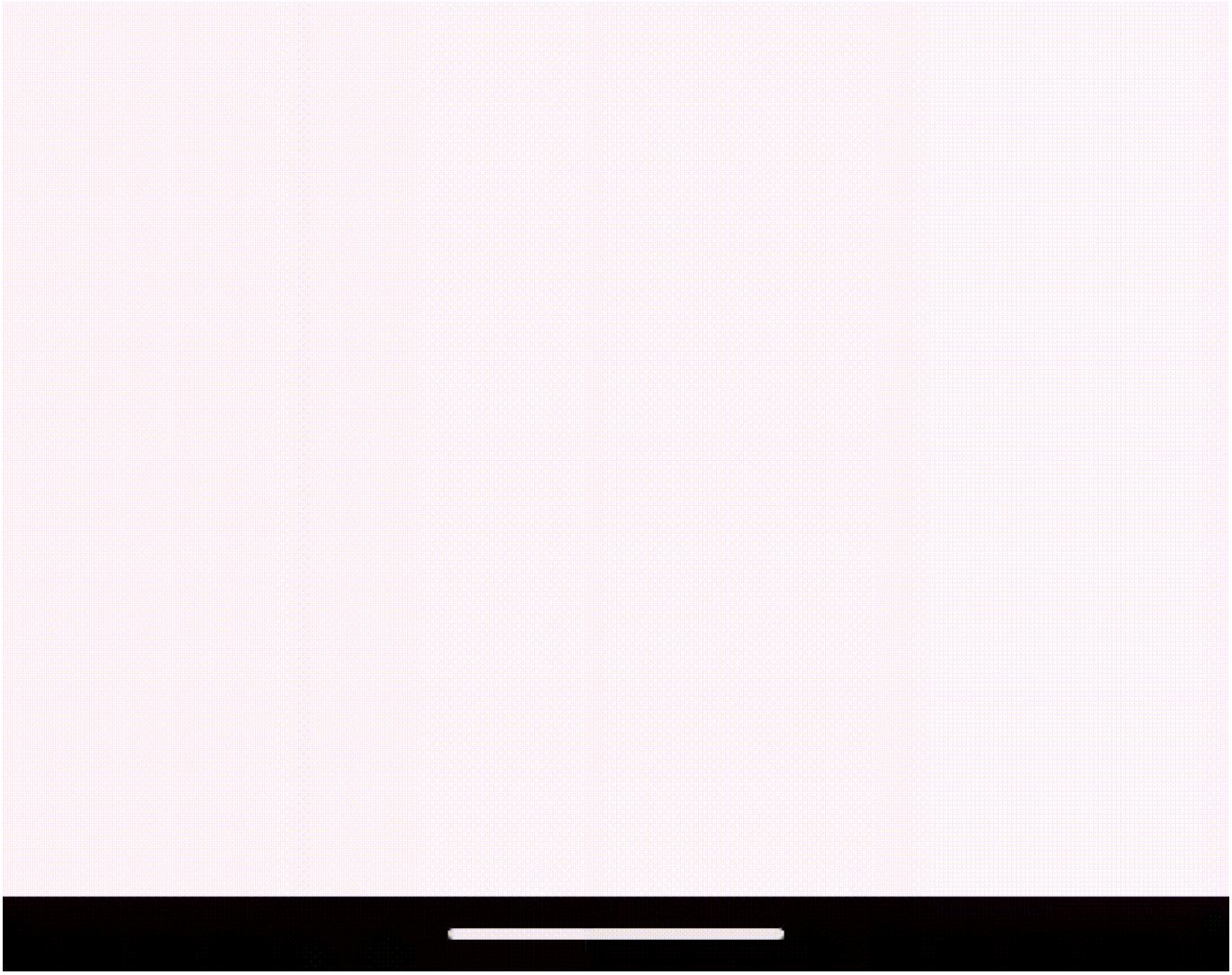


Soal 12

- Jelaskan maksud kode pada langkah 3 dan 7

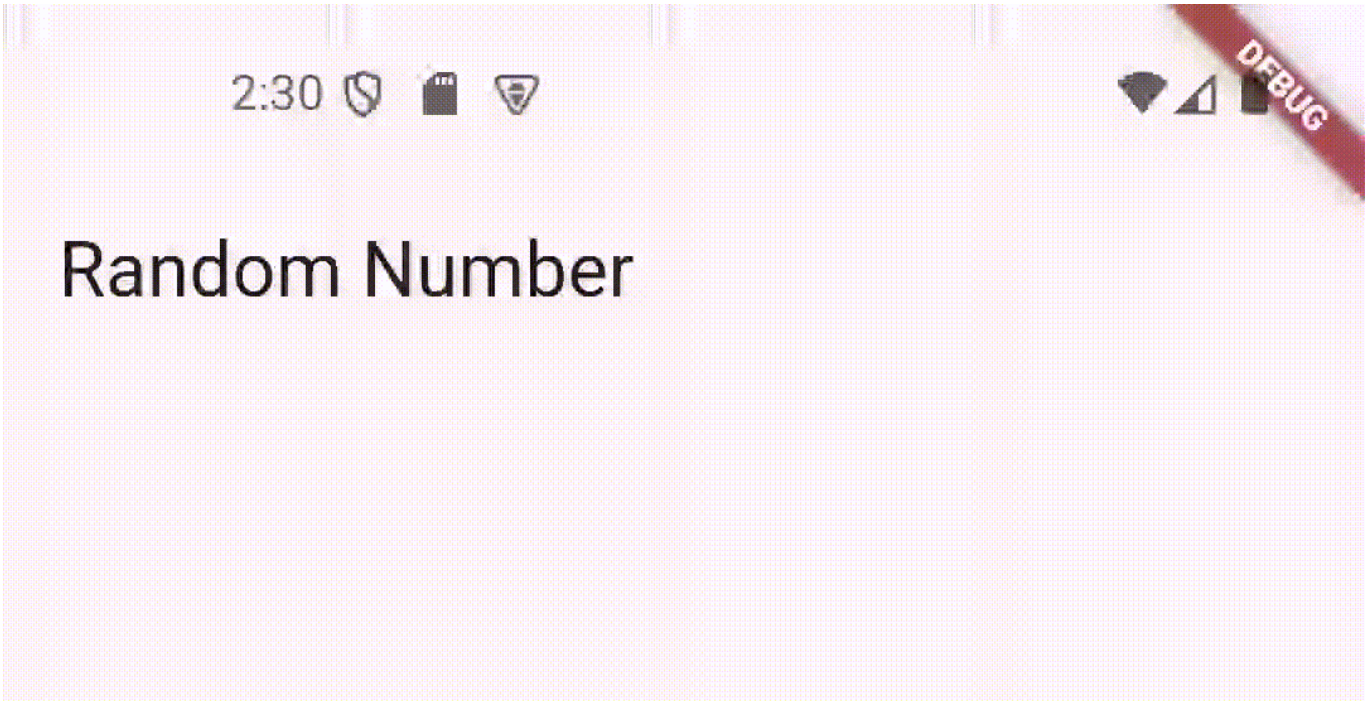
> pada langkah 3 membuat stream dengan periodic setiap 1 second menghasilkan angka random 1-10 pada langkah 7 membuat view menggunakan stream builder





soal 13

> pada praktikum ini terjadi pemisahan pada business logic pada random bloc, letak konsep BLoC terjadi di situ, dimana pemisahan setiap logic berada pada random_bloc dan pada random_screen hanya fokus pada tampilan




Random numberv : null