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

void main() {
  runApp(const MaterialApp(
    home: FirstScreen(),
  ));
}

// アプリ起動時に表示されるFirstScreenウィジェット
class FirstScreen extends StatefulWidget {
  const FirstScreen({super.key});

  @override
  State<StatefulWidget> createState() => _FirstScreenState();
}

class _FirstScreenState extends State<FirstScreen> {
  int _number = 0; ❶

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('FirstScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'number = $_number', ❷
            ),
            ElevatedButton(
              onPressed: () async {
                final newNumber = await Navigator.of(context).push<int>( ❸
                  MaterialPageRoute(
                    builder: (_) => SecondScreen(number: _number),
                  ),
                );
                setState(() {
                  if (newNumber != null) {
                    _number = newNumber; ❹
                  }
                });
              },
              child: const Text('Secondへ'),
            ),
```

```
        ],
      ),
    ),
  );
 }
}
```

_FirstScreenStateクラスは_numberというフィールドを持ち(❶)、Textウィジェットで表示します(❷)。ボタンをタップするとSecondScreen画面へ遷移します。このとき、_numberの値を引数として渡します(❸)。

続いてSecondScreen画面を見てみましょう。

```
class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key, required this.number});

  final int number; ——❺

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('IncrementScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('Increment'),
              onPressed: () {
                Navigator.of(context).pop(number + 1); ——❻
              },
            ),
            ElevatedButton(
              child: const Text('Decrement'),
              onPressed: () {
                Navigator.of(context).pop(number - 1); ——❼
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

SecondScreen画面は引数で渡された数値をフィールドで保持します(❺)。

ボタンをタップすると数値を増減させてFirstScreen画面へ戻ります(❻、❼)。NavigatorStateクラスのpopメソッドの引数が、遷移元の画面に渡ります。

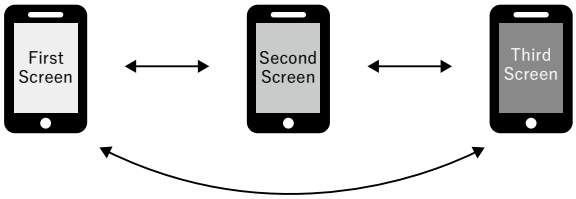
再びFirstScreen画面の❸に注目してください。pushメソッドへ型パラメータを指定しています。これは遷移先からの戻り値の型を指定しています。また、pushメソッドの戻り値はFutureクラスなので、awaitキーワードで戻り値を待つ実装となっています。戻ってきた数値で状態を更新して、このサンプルは完了です(❹)。

名前付きルートによる画面遷移 —— Navigator 1.0(非推奨)

名前付きルートは、遷移先の画面に任意の名前を付けて名前で画面遷移する方法です。画面遷移に関わるコードを一ヵ所に集約したり、ボイラープレートを削減することができます。ただし、名前付きルートは後述の制限事項により現在は推奨されていません。本書では次項のRouterウィジェットとの比較のため解説します。

サンプルコードを見てみましょう。First、Second、Thirdの3つの画面を行き来できるアプリです(**図5.7**)。

図5.7 **名前付きルートサンプルの画面遷移イメージ**



3つの画面はそれぞれ以下のように用意します。

```
// アプリ起動時に表示されるFirstScreenウィジェット
class FirstScreen extends StatelessWidget {
  const FirstScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('FirstScreen'),
      ),
      body: Center(
        child: Column(
```

```
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('FirstからSecondへ'),
              onPressed: () {
              },
            ),
            ElevatedButton(
              child: const Text('FirstからThirdへ'),
              onPressed: () {
              },
            ),
          ],
        ),
      ),
    );
  }
}

// 画面遷移先として用意したSecondScreenウィジェット
class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('SecondScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('SecondからThirdへ'),
              onPressed: () {
              },
            ),
            ElevatedButton(
              child: const Text('戻る'),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ),
          ],
        ),
      ),
```

```
    );
  }
}

// 画面遷移先として用意したThirdScreenウィジェット
class ThirdScreen extends StatelessWidget {
  const ThirdScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('ThirdScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('戻る'),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

画面の遷移先と名前のマッピングはMaterialAppウィジェットの引数で指定します。

```
void main() {
  runApp(
    MaterialApp(
      initialRoute: '/', ——❶
      routes: {
        '/': (context) => const FirstScreen(),
        '/second': (context) => const SecondScreen(),
        '/second/third': (context) => const ThirdScreen(),
      },
    ),
  );
}
```

❷

routesパラメータ(❷)はMap型の引数でキーにはStringを、バリューには

ウィジェットを戻り値とする関数型を渡します。このキーの文字列を画面遷移先として指定することができます。**initialRoute**パラメータ(❶)は最初に表示する画面を指定します。

続いて、名前付きルートで画面遷移するコードを追加しましょう。

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('FirstScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('FirstからSecondへ'),
              onPressed: () {
                Navigator.of(context).pushNamed('/second'); ❶
              },
            ),
            ElevatedButton(
              child: const Text('FirstからThirdへ'),
              onPressed: () {
                Navigator.of(context).pushNamed('/second/third'); ❷
              },
            ),
          ],
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('SecondScreen'),
      ),
      body: Center(
```

```
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('SecondからThirdへ'),
              onPressed: () {
                Navigator.of(context).pushNamed('/second/third'); ❸
              },
            ),
            ElevatedButton(
              child: const Text('戻る'),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ),
          ],
        ),
      ),
    );
  }
}

class ThirdScreen extends StatelessWidget {
  const ThirdScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('ThirdScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('戻る'),
              onPressed: () {
                Navigator.of(context).pop();
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

NavigatorStateクラスのpushNamedメソッドで画面遷移します。遷移先はMaterialAppウィジェットのroutesパラメータに渡したMapのキーで指定します(❶〜❸)。

MaterialAppウィジェットのinitialRouteパラメータは、デフォルト値が/となっています。initialRouteに渡された名前の画面が存在しない場合はデフォルト値が採用されます。

先ほどのサンプルを以下のように変更してみましょう。

```
void main() {
  runApp(
    MaterialApp(
      initialRoute: '/second/third', ——❶
      routes: {
        '/': (context) => const FirstScreen(),
        '/second': (context) => const SecondScreen(),
        '/second/third': (context) => const ThirdScreen(),
      },
    ),
  );
}
```

initialRouteパラメータに/second/thirdを指定します(❶)。アプリを起動するとThirdScreen画面が表示され、「戻る」ボタンを押すとSecondScreen画面が表示されます。さらに「戻る」ボタンを押すとFirstScreen画面が表示されます。このように、initialRouteに指定した画面の中間の画面も生成されます。

一方で、FirstScreen画面からThirdScreen画面へpushNamed関数で遷移した場合は中間のSecondScreen画面は生成されません。この違いに注意してください。

```
ElevatedButton(
  child: const Text('FirstからThirdへ'),
  onPressed: () {
    // 中間のSecondScreenは生成されない
    Navigator.of(context).pushNamed('/second/third');
  },
),
```

名前付きルートの制限事項

/から始まるルートはディープリンクとして扱われる特徴があります。ディープリンクとは、ブラウザなどからアプリ内の特定の画面に遷移させるしくみです。名前付きルートをディープリンクとして利用した場合、常に同じ動作となり、たとえば「ログイン状態によって遷移先を変える」といったカスタマイズ