ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		С.Ю. Гуков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ

ВАРИАНТ 5

по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4128		В.А. Воробьев
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

1 Цель работы

Ознакомиться с основными паттернами проектирования и понять их назначение для решения общих задач проектирования в конкретном контексте, научиться применять паттерны на практике при проектировании и разработке программного обеспечения.

Задание:

Необходимо разработать программу по варианту задания, используя предложенный паттерн проектирования. Также требуется нарисовать в UML диаграмму классов реализуемой программы. Проект может быть выполнен либо в качестве консольного приложения (тогда обязателен командно-текстовый интерфейс), либо иметь графический пользовательский интерфейс (User Interface, UI), а также может быть написан на любом языке программирования.

Вариант задания:

Разработать программу для заведения общественного питания, которая упростит приготовление блюд. Использовать паттерн «Шаблонный метод (Template)».

Пример для шавермы:

Каждая шаверма заворачивается в лаваш, поджаривается на гриле, содержит в себе начинку и поливается сверху небольшим количеством соуса. Для нашего заведения у шаверм будет отличаться только начинка, значит метод «ДобавитьНачинку» должен быть реализован для каждого типа шавермы по-своему.

Пример логики:

• Пользователь выбирает шаверму

• Пользователю выводятся шаги приготовления шавермы

2 Выполнение задания

Для выполнения задания был выбран фреймворк Flutter. Исходный код доступен на GitHub (URL: https://github.com/vladcto/SUAI_homework/tree/65a14fa07cd15d4be9649c126375a2616c7d0576/4_semester/PT/shaverma_book) и в приложении А. В приложение не был включен листинг виджетов не необходимых для понимания паттерна.

Описание разработки и технологии:

Шаблонный метод — это поведенческий паттерн проектирования, который определяет скелет алгоритма, перекладывая ответственность за некоторые его шаги на подклассы. Паттерн позволяет подклассам переопределять шаги алгоритма, не меняя его общей структуры.

Реализация паттерна:

В ходе выполнения лабораторной работы был создан абстрактный класс *Dish*. Он определяет 3 абстрактных защищенных свойства *prepareRecipe*, *addFillingRecipe*, *fryRecipe* и 1 защищённое свойство с реализацией по умолчанию *addSauceRecipe*. Также он определяет публичный метод *createRecipe()*, который вызывает последовательно 4 вышеуказанных свойства, что является реализацией паттерна “Шаблонный метод”. Классы *Shaverma* и *Taco* наследуется от Dish, реализовывая вышеуказанные свойства. Эти два класса не знают о шаблонном методе и не заботятся о последовательности шагов. Реализация паттерна “Шаблонный метод” позволяет однозначно определить структуру алгоритма, которой будут следовать все подклассы, избежать дублирование кода, а также облегчить изменяемость кода подклассами.

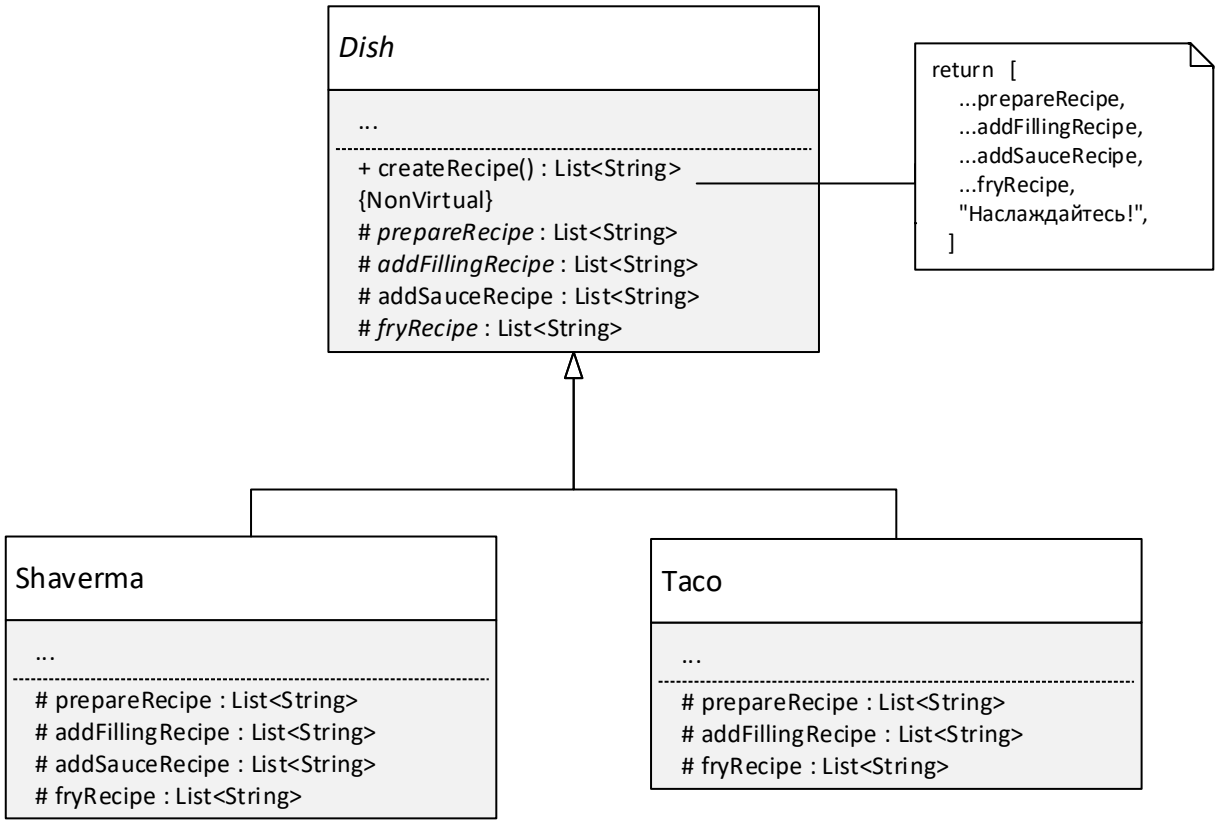

Рисунок 1 – UML-д muchachoиаграмма паттерна

Результат работы:

Для тестирования функционала программы, рассмотрим вывод для тако и для шавермы.

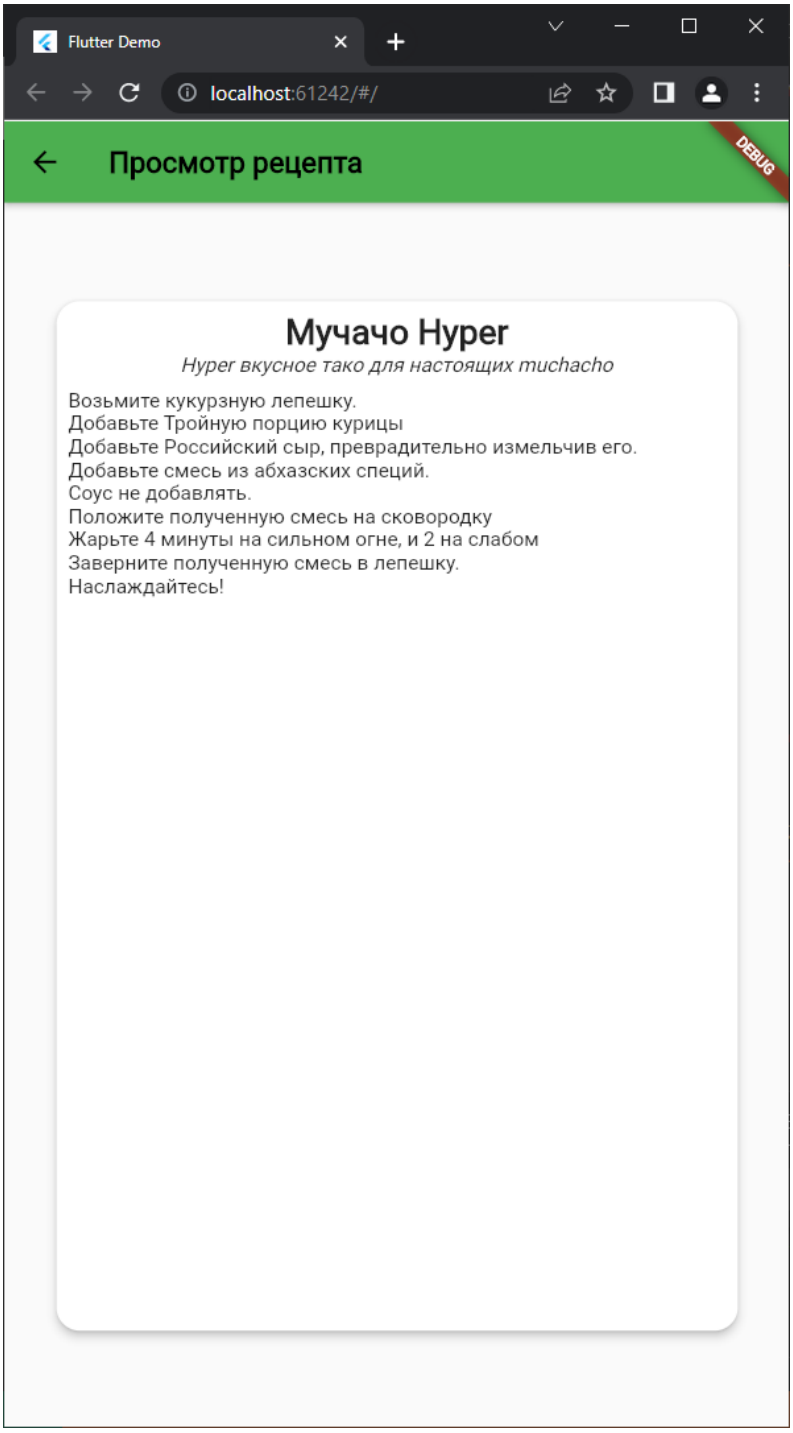

Рисунок 2 – Просмотр тако

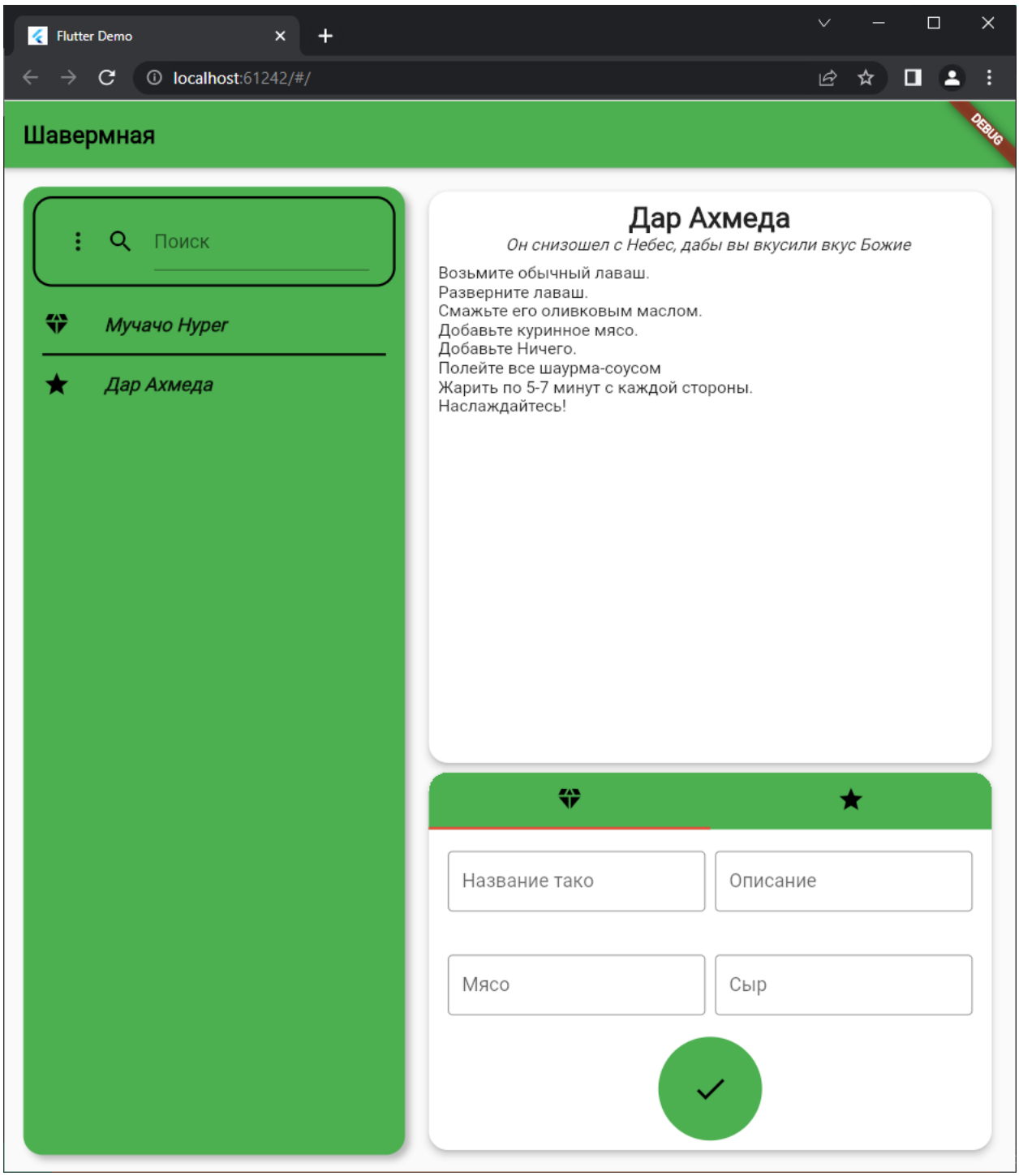

Рисунок 3 – Просмотр шавермы

3 Вывод

В ходе выполнения лабораторной работы на языке программирования Dart была написана программа, реализующая поставленную задачу с использованием объектно-ориентированного программирования и паттерна проектирования “Шаблонный метод”. Паттерны проектирования используют для решения типовых задач, а также поддержания стандартизированного подхода к решению рабочих задач. Поставленные задачи выполнены, исходный код доступен в Приложении А и на GitHub`e.

ПРИЛОЖЕНИЕ А
ЛИСТИНГИ ПРОГРАММЫ

Листинг Dish.dart:

```
import "package:meta/meta.dart";

abstract class Dish {
  final String name;
  final String description;

  Dish({required this.name, required this.description});

  @protected
  List<String> get prepareRecipe;
  @protected
  List<String> get addFillingRecipe;
  @protected
  List<String> get addSauceRecipe => ["Соус не добавлять."];
  @protected
  List<String> get fryRecipe;

  @nonVirtual
  List<String> createRecipe() {
    return [
      ...prepareRecipe,
      ...addFillingRecipe,
      ...addSauceRecipe,
      ...fryRecipe,
      "Наслаждайтесь!",
    ];
```

```
  }
}
```

Листинг Shaverma.dart:

```
import 'package:shaverma_book/model/dish.dart';

enum ShavermaLavash { ordinary, chesee, dense }

class Shaverma extends Dish {
  final List<String> topings;
  final ShavermaLavash lavash;

  Shaverma({
    required super.name,
    required super.description,
    required this.lavash,
    this.topings = const [],
  });

  @override
  List<String> get addFillingRecipe => [
        "Добавьте куринное мясо.",
        ...topings.map((e) => "Добавьте $e."),
      ];

  @override
  List<String> get addSauceRecipe => ["Полейте все шаурма-соусом"];

  @override
  List<String> get fryRecipe => ["Жарить по 5-7 минут с каждой стороны."];
```

```
  @override
  List<String> get prepareRecipe {
    String lavashName;
    if (lavash == ShavermaLavash.chesee) {
      lavashName = "сырный лаваш";
    } else if (lavash == ShavermaLavash.dense) {
      lavashName = "плотный лаваш";
    } else {
      lavashName = "обычный лаваш";
    }
    return [
      "Возьмите $lavashName.",
      "Разверните лаваш.",
      "Смажьте его оливковым маслом.",
    ];
  }
}
```

Листинг Taco.dart:

```
import 'package:shaverma_book/model/dish.dart';

class Taco extends Dish {
  String meat;
  String cheese;
  bool crispy;

  Taco({
    required super.name,
    required super.description,
```

```
    required this.meat,
    required this.cheese,
    this.crispy = false,
  });

  @override
  List<String> get addFillingRecipe => [
        "Добавьте $meat",
        "Добавьте $cheese, преврадительно измельчив его.",
        "Добавьте смесь из абхазских специй."
      ];

  @override
  List<String> get fryRecipe => [
        "Положите полученную смесь на сковородку",
        crispy
            ? "Жарьте 8 минут на сковородке"
            : "Жарьте 4 минуты на сильном огне, и 2 на слабом",
        "Заверните полученную смесь в лепешку."
      ];

  @override
  List<String> get prepareRecipe => ["Возьмите кукурзную лепешку."];
}
```

Листинг остальных виджетов:

```
import 'package:shaverma_book/model/shaverma.dart';
import 'package:shaverma_book/model/taco.dart';
import 'package:shaverma_book/view/home/abstract_dish_page.dart';
```

```
import '../model/dish.dart';

enum DishSortType { none, shaverma, taco }

class DishPresenter {
  final List<Dish> _dishes = [
    Shaverma(
      name: "тест",
      description: "Описание тест",
      lavash: ShavermaLavash.ordinary,
    ),
    Shaverma(
      name: "тест2",
      description: "Описание текста",
      lavash: ShavermaLavash.chesee,
    ),
    Taco(
      name: "тест3",
      description: "Описание текст",
      meat: "Курица",
      cheese: "Сыр",
    ),
  ];

  DishSortType _sortType = DishSortType.none;
  String _findName = "";
  List<Dish> get sortedDishes => _dishes
      .where((e) => equalType(e, _sortType))
      .where((e) => e.name.startsWith(_findName))
      .toList();
```

```
late AbstractDishPage page;

int get dishCount => sortedDishes.length;

String getNameAt(int i) => sortedDishes[i].name;

void onDishTap(int iDish) {
  page.showDish(
    sortedDishes[iDish].name,
    sortedDishes[iDish].description,
    sortedDishes[iDish].createRecipe(),
  );
}

void deleteAt(int i) {
  _dishes.removeAt(i);
  page.updateList();
}

void createTaco(String name, String description, String meat, String chesse) {
  _dishes.add(
    Taco(
      name: name,
      description: description,
      meat: meat,
      cheese: chesse,
    ),
  );
  page.updateList();
```

```
  }

  void createShaverma(String name, String description, int i, List<String> topings)
{
    _dishes.add(
      Shaverma(
        name: name,
        description: description,
        lavash: ShavermaLavash.ordinary,
        topings: topings,
      ),
    );
    page.updateList();
  }

  void changeFilter(String findName, DishSortType filterType) {
    _findName = findName;
    _sortType = filterType;
    page.updateList();
  }

  bool isShavermaAt(int i) => sortedDishes[i] is Shaverma;

  static bool equalType(Dish dish, DishSortType sortType) {
    if (sortType == DishSortType.shaverma) {
      return dish is Shaverma;
    } else if (sortType == DishSortType.taco) {
      return dish is Taco;
    } else {
      return true;
```

```
    }
  }
}

abstract class AbstractDishPage {
  void showDish(String name, String description, List<String> recipe);
  void updateList();
}

import 'package:flutter/material.dart';
import 'package:shaverma_book/presenter/dish_presenter.dart';
import 'package:shaverma_book/view/home/tall_dish_page.dart';
import 'package:shaverma_book/view/home/wide_dish_page.dart';

class HomeScreen extends StatelessWidget {
  final DishPresenter _dishPresenter = DishPresenter();
  HomeScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Шавермная',
          style: TextStyle(
            color: Colors.black,
            fontWeight: FontWeight.w900,
          ),
        ),
      ),
```

```
      body: LayoutBuilder(
        builder: (bcontext, constraints) {
          if (constraints.maxHeight / constraints.maxWidth > 1.2) {
            return TallDishPage(_dishPresenter);
          } else {
            return WideDishPage(_dishPresenter);
          }
        },
      ),
    );
  }
}

import 'package:flutter/material.dart';
import 'package:shaverma_book/presenter/dish_presenter.dart';
import 'package:shaverma_book/view/home/dish_listview.dart';
import 'package:shaverma_book/view/home/preview_dish_page.dart';

import '../../Globals.dart';
import 'abstract_dish_page.dart';
import 'create_dish_page.dart';
import 'filter_card.dart';

class TallDishPage extends StatefulWidget {
  final DishPresenter _dishPresenter;
  const TallDishPage(this._dishPresenter, {Key? key}) : super(key: key);

  @override
  State<TallDishPage> createState() => _TallDishPageState();
}
```

```
class _TallDishPageState extends State<TallDishPage> implements
AbstractDishPage {
  late Function(String name, String description, List<String> recipe) _showDish;

  @override
  void initState() {
    super.initState();
    widget._dishPresenter.page = this;
  }

  @override
  Widget build(BuildContext context) {
    _showDish = (name, description, recipe) {
      Navigator.of(context).push(
        MaterialPageRoute<PreviewDishPage>(
          builder: (_) => PreviewDishPage(
            name: name,
            description: description,
            steps: recipe,
          ),
        ),
      );
    };
    return Column(
      children: [
        Expanded(
          child: DishListView(widget._dishPresenter),
        ),
        SizedBox(
```

```
height: 156,
child: Padding(
  padding: const EdgeInsets.only(
    left: 32,
    right: 32,
    bottom: 28,
  ),
  child: Row(
    children: [
      Expanded(
        child: FilterCard(
          onChangedFilter: widget._dishPresenter.changeFilter,
        ),
      ),
      const SizedBox(width: 16),
      AspectRatio(
        aspectRatio: 1,
        child: Container(
          decoration: BoxDecoration(
              color: Globals.mainColor,
              borderRadius: BorderRadius.circular(100),
              border: Border.all(color: Colors.black, width: 2),
              boxShadow: [Globals.shadow]),
          child: GestureDetector(
            onTap: () => Navigator.of(context).push(
              MaterialPageRoute<CreateDishPage>(
                builder: (_) => CreateDishPage(
                  onCreateShaverma: widget._dishPresenter.createShaverma,
                  onCreateTaco: widget._dishPresenter.createTaco,
                ),
```

```
                ),
              ),
              child: const Icon(
                Icons.add,
                color: Colors.black,
                size: 64,
              ),
            ),
          ),
        ),
      ],
    ),
  ),
 )
],
 );
}

@override
void showDish(String name, String description, List<String> recipe) =>
    _showDish(name, description, recipe);

@override
void updateList() {
 setState(() {});
}
}

import 'package:flutter/material.dart';
import 'package:shaverma_book/presenter/dish_presenter.dart';
```

```
import 'package:shaverma_book/view/home/dish_creator.dart';
import 'package:shaverma_book/view/home/abstract_dish_page.dart';
import 'package:shaverma_book/view/home/filter_card.dart';
import 'package:shaverma_book/view/home/recipe_previewer.dart';

import '../../Globals.dart';
import 'dish_listview.dart';

class WideDishPage extends StatefulWidget {
  final DishPresenter _dishPresenter;
  const WideDishPage(this._dishPresenter, {Key? key}) : super(key: key);

  @override
  State<WideDishPage> createState() => _WideDishPageState();
}

class _WideDishPageState extends State<WideDishPage> implements
AbstractDishPage {
  String dishName = "Ничего не выбрано";
  String dishDescription = "Нажмите на элемент списка, чтобы посмотреть
рецепт";
  List<String> dishRecipe = [];

  @override
  void initState() {
    super.initState();
    widget._dishPresenter.page = this;
  }

  @override
```

```
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.all(16),
    child: Center(
      child: ConstrainedBox(
        constraints: const BoxConstraints(maxWidth: 1080),
        child: Row(
          children: [
            Flexible(
              flex: 2,
              child: Container(
                decoration: BoxDecoration(
                    color: Colors.green,
                    borderRadius: BorderRadius.circular(16),
                    boxShadow: [
                      Globals.shadow,
                    ]),
                clipBehavior: Clip.hardEdge,
                child: Column(
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8),
                      child: SizedBox(
                          height: 76,
                          child: FilterCard(
                              onChangedFilter:
                                  widget._dishPresenter.changeFilter)),
                    ),
                    Expanded(
                      child: DishListView(widget._dishPresenter),
```

```
        ),
      ],
    ),
  ),
),
const SizedBox(
  width: 16,
),
Flexible(
  flex: 3,
  child: Column(
    children: [
      Flexible(
        flex: 3,
        child: RecipePreviewer(
          name: dishName,
          description: dishDescription,
          recipe: dishRecipe,
        ),
      ),
      Flexible(
        flex: 2,
        child: DishCreator(
          onCreateTaco: widget._dishPresenter.createTaco,
          onCreateShaverma: widget._dishPresenter.createShaverma,
        ),
      ),
    ],
  ),
)
```

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

  @override
  void showDish(String name, String description, List<String> recipe) {
    setState(() {
      dishName = name;
      dishDescription = description;
      dishRecipe = recipe;
    });
  }

  @override
  void updateList() {
    setState(() {});
  }
}
```