



Recipe Collection Organizer

Dzhanteliev Adilkhan

EEAIR24

Date: 17th april 12:10-12:40

My project organizes recipes, allowing users to add, edit, and delete recipes along with managing categories and ingredients.

- Описание проекта: Данный проект реализует мини-приложение для управления и работой над рецептами. Пользователь может добавлять, редактировать, просматривать и удалять рецепты. Также реализованы расширенные функции, такие как поиск, фильтрация, избранное, экспорт/импорт JSON, графический интерфейс (Swing) и подключение к базе данных (SQLite).

Цель проекта:

- Реализовать приложение на Java с возможностью управления рецептами.
- Освоить работу с файлами, базой данных, графическим интерфейсом и валидацией.
- Применить принципы ООП и модульности.
- Поддержка импорта/экспорта рецептов и расширенного поиска.

Вспомогательные инструменты, которые я использовал:

- **Java (Swing)** – GUI
- **SQLite + JDBC** – база данных
- **JSON.simple 1.1.1** – для импорта/экспорта
- **Javadoc** – документация

10 методов и возможностей:

- Добавление, редактирование и удаление рецептов
- Просмотр всех рецептов
- Поиск по ингредиенту
- Фильтрация по категории
- Сортировка по времени
- Избранные рецепты
- Запланированные рецепты
- Расширенный поиск
- Импорт и экспорт JSON
- Отображение даты создания рецепта
- Работа с базой данных SQLite

Структура проекта:

- `Recipe.java` – модель рецепта и конструктор
- `RecipeManager.java` – основная логика (работа с БД и рецептами)
- `Main.java` – графический интерфейс (через Swing)
- `FileHandler.java` – валидация данных
- `DatabaseHandler.java` – работа с SQL

© DatabaseHandler

© FileHandler

© Main

© Recipe

© RecipeManager

) .gitignore

] Recipe Collection Organizer.iml

] recipes.db

- recipes.json

Recipe.java

Здесь хранится один рецепт

- Хранит все данные одного рецепта
- Используется во всех остальных классах как объект

```
import java.util.*;
// Файл для использования хранения информации о рецепте и взаимодействия с пользователем.
public class Recipe { 29 usages  👤 AdilkhanEEAIR *
    private int id; 3 usages
    private String name; 3 usages
    private String description; 3 usages
    private List<String> ingredients; 3 usages
    private List<String> steps; 3 usages
    private String category; 3 usages
    private String cookingTime; 3 usages
    private String servingSize; 3 usages
    private boolean isFavorite; 3 usages
    private String createdAt; 3 usages
    private boolean inPlan; 3 usages

    public Recipe(int id, String name, String description, List<String> ingredients, List<String> steps, String category, String
        this.id = id;
        this.name = name;
        this.description = description;
        this.ingredients = ingredients;
        this.steps = steps;
        this.category = category;
        this.cookingTime = cookingTime;
        this.servingSize = servingSize;
        this.isFavorite = isFavorite;
        this.createdAt = createdAt;
        this.inPlan = inPlan;
    }
}
```

RecipeManager.java - ЛОГИКА

- `recipes`: список всех рецептов, которые загружаются из базы данных.
- `nextId`: переменная для генерации уникальных ID.

```
public class RecipeManager { 2 usages  👤 AdilKhanEEAIR *  
    private List<Recipe> recipes; 21 usages  
    private int nextId = 1; 5 usages
```

```
public RecipeManager() { 1 usage  👤 AdilKhanEEAIR *  
    recipes = new ArrayList<Recipe>();  
    loadFromDatabase();  
}
```

Добавление и удаление рецепта:

AddRecipe - Создаёт новый рецепт, добавляет в список и сохраняет в базу.

DeleteRecipeById - Удаляет рецепт из списка по ID и сохраняет обновлённый список в БД.

```
public void addRecipe(String name, String description, String ingredientsStr, String stepsStr, 1 usage  👤 AdilkhanEEAIR *  
                        String category, String cookingTime, String servingSize,  
                        boolean isFavorite, boolean inPlan) {  
    List<String> ingredients = Arrays.asList(ingredientsStr.split(regex: "\\s*,\\s*"));  
    List<String> steps = Arrays.asList(stepsStr.split(regex: "\\s*,\\s*"));  
    String createdAt = getCurrentDate();  
    int id = nextId++;  
    Recipe recipe = new Recipe(id, name, description, ingredients, steps, category,  
                                cookingTime, servingSize, isFavorite, createdAt, inPlan);  
    recipes.add(recipe);  
    saveToDatabase();  
}  
  
public void deleteRecipeById(int id) { 1 usage  👤 AdilkhanEEAIR *  
    for (int i = 0; i < recipes.size(); i++) {  
        if (recipes.get(i).getId() == id) {  
            recipes.remove(i);  
            break;  
        }  
    }  
    saveToDatabase();  
}
```


updateRecipe - обновляет рецепт в списке по ID и сохраняет в БД.

getRecipeById - Возвращает рецепт по заданному ID.

getAllRecipes - Формирует строку с описанием всех рецептов.

```
public void updateRecipe(Recipe updatedRecipe) { 1 usage  👤 AdilkhanEEAIR *
    for (int i = 0; i < recipes.size(); i++) {
        if (recipes.get(i).getId() == updatedRecipe.getId()) {
            recipes.set(i, updatedRecipe);
            break;
        }
    }
    saveToDatabase();
}

public Recipe getRecipeById(int id) { 1 usage  👤 AdilkhanEEAIR *
    for (Recipe r : recipes) {
        if (r.getId() == id) {
            return r;
        }
    }
    return null;
}

public String getAllRecipes() { 1 usage  new *
    StringBuilder sb = new StringBuilder();
    for (Recipe recipe : recipes) {
        sb.append(recipeToString(recipe)).append("\n\n");
    }
    return sb.toString();
}
```

Recipe recipe -
преобразует объект
Recipe в читаемую
строку.

searchByIngredient -
поиск по ингредиенту.
Возвращает рецепты,
содержащие указанный
ингредиент.

```
public String recipeToString(Recipe recipe) { 7 usages new *
    return "ID: " + recipe.getId() +
        "\nНазвание: " + recipe.getName() +
        "\nОписание: " + recipe.getDescription() +
        "\nИнгредиенты: " + String.join(delimiter: ", ", recipe.getIngredients()) +
        "\nШаги: " + String.join(delimiter: ", ", recipe.getSteps()) +
        "\nКатегория: " + recipe.getCategory() +
        "\nВремя приготовления: " + recipe.getCookingTime() +
        "\nПорции: " + recipe.getServingSize() +
        "\nИзбранное: " + (recipe.isFavorite() ? "Да" : "Нет") +
        "\nДата создания: " + recipe.getCreateDate() +
        "\nЗапланировано: " + (recipe.isInPlan() ? "Да" : "Нет");
}
```

```
public String searchByIngredient(String ingredient) { 1 usage AdilkhanEEAIR *
    StringBuilder sb = new StringBuilder();
    for (Recipe r : recipes) {
        if (r.getIngredients().contains(ingredient)) {
            sb.append(recipeToString(r)).append("\n\n");
        }
    }
    return sb.length() == 0 ? "Нет совпадений." : sb.toString();
}
```

filterByCategory - Возвращает
рецепты с заданной
категорией.

sortByCookingTime -
Сортирует рецепты по времени
готовки (в порядке
увеличения).

getFavorites - возвращает
только рецепты, помеченные
как избранные в GUI.

```
public String filterByCategory(String category) { 1 usage new *
    StringBuilder sb = new StringBuilder();
    for (Recipe r : recipes) {
        if (r.getCategory().equalsIgnoreCase(category)) {
            sb.append(recipeToString(r)).append("\n\n");
        }
    }
    return sb.length() == 0 ? "Нет совпадений." : sb.toString();
}

public String sortByCookingTime() { 1 usage AdilkhanEEAIR *
    List<Recipe> sorted = new ArrayList<>(recipes);
    sorted.sort((Recipe a, Recipe b) -> a.getCookingTime().compareTo(b.getCookingTime()));
    StringBuilder sb = new StringBuilder();
    for (Recipe r : sorted) {
        sb.append(recipeToString(r)).append("\n\n");
    }
    return sb.toString();
}

public String getFavorites() { 1 usage AdilkhanEEAIR *
    StringBuilder sb = new StringBuilder();
    for (Recipe r : recipes) {
        if (r.isFavorite()) {
            sb.append(recipeToString(r)).append("\n\n");
        }
    }
    return sb.length() == 0 ? "Нет избранных рецептов." : sb.toString();
}
```

getPlannedRecipes -
Возвращает рецепты,
отмеченные как
запланированные.

advancedSearch - ищет по
нескольким вещам:
ингредиенты, категория,
время и порции.

```
public String getPlannedRecipes() { 1 usage  👤 AdilkhanEEAIR *
    StringBuilder sb = new StringBuilder();
    for (Recipe r : recipes) {
        if (r.isInPlan()) {
            sb.append(recipeToString(r)).append("\n\n");
        }
    }
    return sb.length() == 0 ? "Нет запланированных рецептов." : sb.toString();
}
```

```
public String advancedSearch(String ingredientsStr, String category, String time, String servings) { 1 usage  👤 Adilkha
    List<String> ingredients = Arrays.asList(ingredientsStr.split(regex: "\\s*,\\s*"));
    StringBuilder sb = new StringBuilder();
    for (Recipe r : recipes) {
        if (r.getCategory().equalsIgnoreCase(category) &&
            r.getCookingTime().equalsIgnoreCase(time) &&
            r.getServingSize().equalsIgnoreCase(servings) &&
            r.getIngredients().containsAll(ingredients)) {
            sb.append(recipeToString(r)).append("\n\n");
        }
    }
    return sb.length() == 0 ? "Нет совпадений." : sb.toString();
}
```

Import JSON

- Читает JSON файл и добавляет рецепты в список.
- Увеличивает nextId, если нужно
- Затем сохраняет в базу данных.

```
public void importFromJson(String path) { 1 usage  👤 AdilkhanEEAIR *
    try {
        JSONParser parser = new JSONParser();
        JSONArray array = (JSONArray) parser.parse(new FileReader(path));
        for (Object obj : array) {
            JSONObject json = (JSONObject) obj;
            int id = Integer.parseInt(json.get("id").toString());
            String name = (String) json.get("name");
            String description = (String) json.get("description");
            List<String> ingredients = parseList((JSONArray) json.get("ingredients"));
            List<String> steps = parseList((JSONArray) json.get("steps"));
            String category = (String) json.get("category");
            String cookingTime = (String) json.get("cookingTime");
            String servingSize = (String) json.get("servingSize");
            boolean isFavorite = Boolean.parseBoolean(json.get("isFavorite").toString());
            String createdAt = (String) json.get("createdAt");
            boolean inPlan = Boolean.parseBoolean(json.get("inPlan").toString());

            Recipe r = new Recipe(id, name, description, ingredients, steps, category,
                                   cookingTime, servingSize, isFavorite, createdAt, inPlan);
            recipes.add(r);
            if (id >= nextId) nextId = id + 1;
        }
        saveToDatabase();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Export JSON

Создаёт JSON-массив рецептов и сохраняет в файл.

```
public void exportToJson(String path) { 1 usage new *
    JSONArray array = new JSONArray();
    for (Recipe r : recipes) {
        JSONObject json = new JSONObject();
        json.put("id", r.getId());
        json.put("name", r.getName());
        json.put("description", r.getDescription());
        json.put("ingredients", r.getIngredients());
        json.put("steps", r.getSteps());
        json.put("category", r.getCategory());
        json.put("cookingTime", r.getCookingTime());
        json.put("servingSize", r.getServingSize());
        json.put("isFavorite", r.isFavorite());
        json.put("createdDate", r.getCreatedDate());
        json.put("inPlan", r.isInPlan());
        array.add(json);
    }
    try (FileWriter writer = new FileWriter(path)) {
        writer.write(array.toJSONString());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Дополнительные требования: графический интерфейс Swing и SQL database.

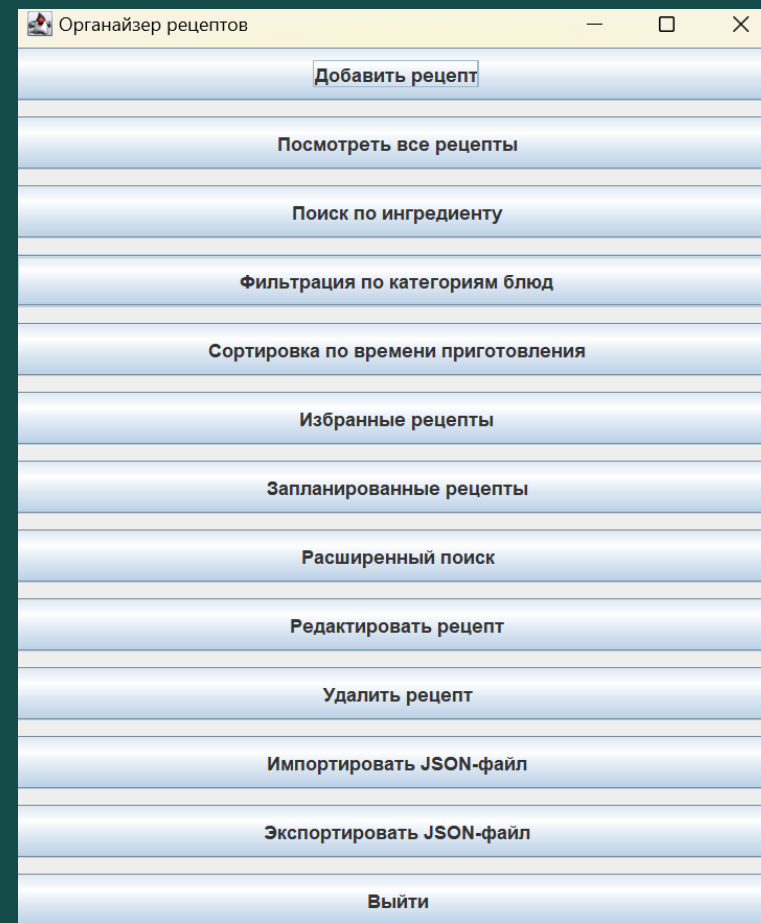
```
import java.sql.*;

public class DatabaseHandler { no usages new *
    private static final String DB_URL = "jdbc:sqlite:recipes.db"; 1 usage

    // Метод для установления соединения с базой данных
    public static Connection connect() throws SQLException { 1 usage new *
        return DriverManager.getConnection(DB_URL);
    }

    public static void initializeDatabase() { no usages new *
        String sql = "CREATE TABLE IF NOT EXISTS recipes (" +
            "id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "name TEXT NOT NULL," +
            "description TEXT," +
            "ingredients TEXT," +
            "steps TEXT," +
            "category TEXT," +
            "cookingTime TEXT," +
            "servingSize TEXT," +
            "isFavorite INTEGER," +
            "createdDate TEXT," +
            "inPlan INTEGER" +
            ");";


        // Используем try-with-resources для автоматического закрытия соединения
        try (Connection conn = connect(); Statement stmt = conn.createStatement()) {
            // Создание таблицы
            stmt.execute(sql);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```



Примеры выходных данных (остальные отправил в GitHub)

Сортировка по времени приготовления


Input

 Введите ингредиент:

OK Cancel

Редактировать рецепт

Message

 ID: 2
Название: Манты
Описание: Манты — это традиционное блюдо восточной кухни, предст
Ингредиенты: Тесто, мясо, лук, перец
Шаги: Замес теста, приготовление мясной начинки, формирование изд
Категория: Второе
Время приготовления: 4
Порции: 5
Избранное: Нет
Дата создания: 2025-04-13
Запланировано: Да

OK

Полный проект:

- <https://github.com/AdilkhaneAIR/Recipe-Collection-Organizer>

<https://github.com/AdilkhaneAIR/Recipe-Collection-Organizer>

<https://github.com/AdilkhaneAIR/Recipe-Collection-Organizer>