Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |
| --- |
| Институт информационных технологий и анализа данных |

наименование института

|  |
| --- |
| Допускаю к защите  Руководитель:  М.Д. Каташевцев |
| И.О. Фамилия |

Обработка иерархии классов с использованием шаблонов

архитектурного проектирования

наименование темы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

|  |
| --- |
| Технологии программирования |

|  |
| --- |
| 1.001.00.00 ПЗ |
| обозначение документа |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | ИС\*\*\* 18-1 |  |  |  | Иванов И.И. |
|  | шифр группы |  | подпись |  | Фамилия И.О |
| Нормоконтроль |  |  |  |  | Каташевцев М.Д. |
|  |  |  | подпись |  | Фамилия И.О |

|  |  |
| --- | --- |
| Курсовая работа защищена с оценкой |  |

Иркутск 2020 г.

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| По курсу | Технологии программирования | |
| Студенту | Иванову Ивану Ивановичу | |
|  | (фамилия, инициалы) | |
| Тема работы: | Обработка иерархии классов с использованием шаблонов | | |
| архитектурного проектирования | | | |
| Исходные данные: | | Вариант 1 | |
| Рекомендуемая литература: | | | |
| 1. Лаврищева Е. М. Программная инженерия. Парадигмы, технологии и CASE-средства 2-е изд. Учебник для вузов. 2018 – 280с. Серия: Университеты России ISBN: 978-5-534-01056-5  https://biblio-online.ru/viewer/programmnaya-inzheneriya-i-tehnologii-programmirovaniyaslozhnyh-sistem-436514#page/1 | | | |
|  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Графическая часть на | - | | листах. | |
| Дата выдачи задания |  | 23 / 11 / 2020 г. | | |
| Задание получил студент | | | |  | |  | Иванов И.И. |
|  | | | | подпись | |  | Фамилия И.О. |

|  |  |
| --- | --- |
| Дата представления работы руководителю | 21 / 12 / 2020 г. |
| Руководитель курсового проектирования |  |  | Каташевцев М.Д. |
|  | подпись | Фамилия И.О. |

Содержание

[Введение 4](#_Toc27587670)

[1. Индивидуальный вариант 5](#_Toc27587671)

[2. Диаграмма приложения и классов предметной области 6](#_Toc27587672)

[3. Внешний вид окон 7](#_Toc27587673)

[4. Код основных событий 8](#_Toc27587674)

[4.1 Описание модели 8](#_Toc27587675)

[4.1.1 Сохранение в файл 8](#_Toc27587676)

[4.1.2 Загрузка из файла 8](#_Toc27587677)

[4.1.3 Добавление экземпляра еды 8](#_Toc27587678)

[4.1.4 Редактирование экземпляра еды 9](#_Toc27587679)

[4.1.5 Удаление экземпляра еды 9](#_Toc27587680)

[4.1.6 Оповещение об изменениях 9](#_Toc27587681)

[4.2. Описание контроллера 10](#_Toc27587682)

[4.2.1 Инициализация контроллера 10](#_Toc27587683)

[5. Таблица тестов 12](#_Toc27587684)

[Заключение 17](#_Toc27587685)

[Список использованной литературы 18](#_Toc27587686)

[Приложение 1. Файл данных data.json 19](#_Toc27587687)

# Введение

Необходимо разработать иерархию классов программу с использованием шаблона архитектурного проектирования MVP для выполнения следующих действий:

• Добавление экземпляров всех классов в коллекцию List

• Удаление экземпляра класса из коллекции

• Замена экземпляра класса в коллекции

• Вывод всех экземпляров всех классов

• Выборочный вывод экземпляров только одного класса

• Десериализация объектов из текстового файла в формате JSON

Язык программирования Java.

Java –язык программирования, во многом схожий с языком C#. Является строго типизированным и объектно-ориентированным. Благодаря большому количеству встроенных библиотек Java обеспечивает возможность быстрой разработки приложений. Будучи изначально разрабатываемый как мультиплатформенный язык, позволяет запускать однажды скомпилированные исходники на разных операционных системах без необходимости перекомпиляции.

Необходимо на основании индивидуального задания разработать приложение на языке Java, таблицу тестов и оттестировать программу для создания файла, добавления записи, удаления записи, изменения записи, просмотра файла и выполнения двух запросов. Удаление и изменение записи происходит по ее номеру. Способы выполнения действий – согласно индивидуальному заданию. При ошибочных ситуациях (отсутствие записи, отсутствие файла и т.д.) необходимо выдавать сообщения.

# 1. Индивидуальный вариант

**Вариант – 0**

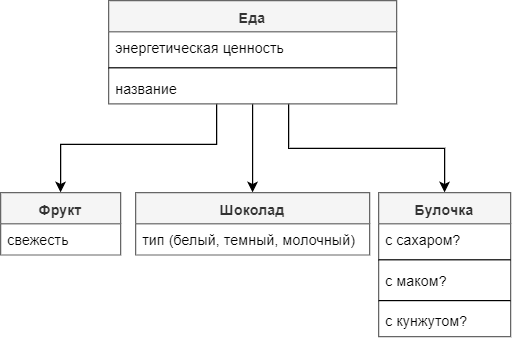


Схема 1.1 – Иерархия классов

Выборочная печать: по типу еды

# 2. Диаграмма приложения и классов предметной области

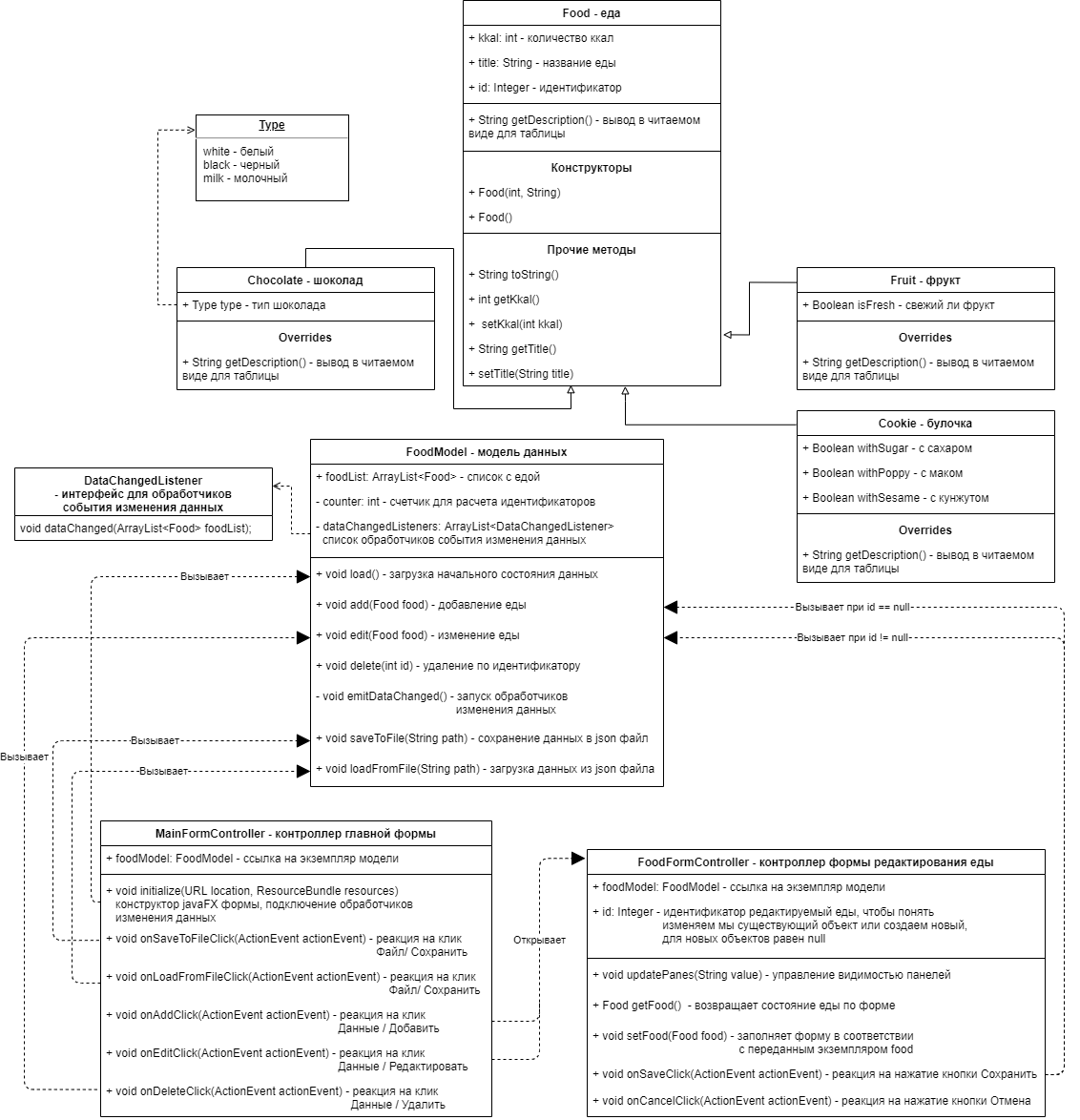


Схема 2.1 – Иерархия классов

# 3. Внешний вид окон

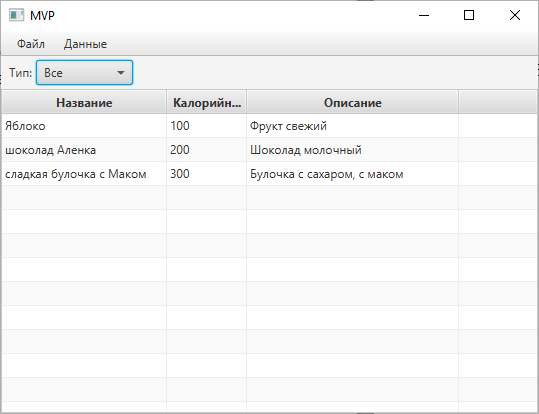


Рисунок 4.1 – Главная форма

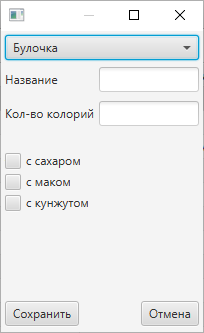
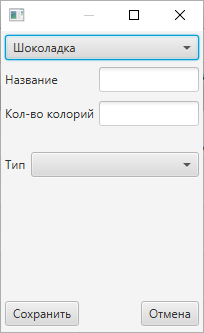
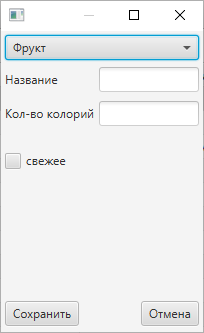


Рисунок 4.2 – Форма добавления/редактирования объекта

# 4. Код основных событий

## 4.1 Описание модели

**public class** FoodModel {

// список с данными  
 ArrayList<Food> **foodList** = **new** ArrayList<>();

// счетчик  
 **private int counter** = 1;

// фильтр по типу  
 Class<? **extends** Food> **foodFilter** = Food.**class**;  
  
 **public interface** DataChangedListener {  
 **void** dataChanged(ArrayList<Food> foods);  
 }  
**private** ArrayList<DataChangedListener> **dataChangedListeners** = **new** ArrayList<>();  
**public void** addDataChangedListener(DataChangedListener listener) {  
 **this**.**dataChangedListeners**.add(listener);  
 }

### 4.1.1 Сохранение в файл

**public void** saveToFile(String path) {**try** (Writer writer = **new** FileWriter(path)) {  
 ObjectMapper mapper = **new** ObjectMapper();  
 mapper.writerFor(**new** TypeReference<ArrayList<Food>>() {  
 })  
 .withDefaultPrettyPrinter()  
 .writeValue(writer, **foodList**);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }

### 4.1.2 Загрузка из файла

**public void** loadFromFile(String path) {  
 **try** (Reader reader = **new** FileReader(path)) {  
 *// создаем сериализатор* ObjectMapper mapper = **new** ObjectMapper();  
 **foodList** = mapper.readerFor(**new** TypeReference<ArrayList<Food>>() {  
 })  
 .readValue(reader);  
 **this**.**counter** = **foodList**.stream()  
 .map(food -> food.**id**)  
 .max(Integer::compareTo)  
 .orElse(0) + 1;  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 **this**.emitDataChanged();  
 }

### 4.1.3 Добавление экземпляра еды

**public void** add(Food food, **boolean** emit) {  
 food.**id** = **counter**;  
 **counter** += 1;  
  
 **this**.**foodList**.add(food);  
  
 **if** (emit) {  
 **this**.emitDataChanged();  
 }  
 }  
  
 **public void** add(Food food) {  
 add(food, **true**);  
 }

### 4.1.4 Редактирование экземпляра еды

**public void** edit(Food food) {  
 **for** (**int** i = 0; i < **this**.**foodList**.size(); ++i) {  
 **if** (**this**.**foodList**.get(i).**id** == food.**id**) {  
 **this**.**foodList**.set(i, food);  
 **break**;  
 }  
 }  
  
 **this**.emitDataChanged();  
 }

### 4.1.5 Удаление экземпляра еды

**public void** delete(**int** id) {  
 **for** (**int** i = 0; i < **this**.**foodList**.size(); ++i) {  
 **if** (**this**.**foodList**.get(i).**id** == id) {  
 **this**.**foodList**.remove(i);  
 **break**;  
 }  
 }  
  
 **this**.emitDataChanged();  
 }

### 4.1.6 Оповещение об изменениях

**private void** emitDataChanged() {  
 **for** (DataChangedListener listener : **dataChangedListeners**) {  
 ArrayList<Food> filteredList = **new** ArrayList<>(  
 **foodList**.stream()  
 .filter(food -> **foodFilter**.isInstance(food))  
 .collect(Collectors.*toList*())  
 );  
 listener.dataChanged(filteredList);  
 }  
 }

4.1.7 Фильтрация по типу еды

**public void** setFoodFilter(Class<? **extends** Food> foodFilter) {  
 **this**.**foodFilter** = foodFilter;  
  
 **this**.emitDataChanged();  
 }  
}

## 4.2. Описание контроллера

**public class** MainFormController **implements** Initializable {  
 **public** TableView **mainTable**;  
 **public** ComboBox **cmbFoodType**;  
  
 ObservableList<Food> **foodList** = FXCollections.*observableArrayList*();  
 ObservableList<Class<? **extends** Food>> **foodTypes** = FXCollections.*observableArrayList*(  
 Food.**class**,  
 Fruit.**class**,  
 Chocolate.**class**,  
 Cookie.**class** );  
  
 FoodModel **foodModel** = **new** FoodModel();

### 4.2.1 Инициализация контроллера

@Override  
 **public void** initialize(URL location, ResourceBundle resources) {  
 **foodModel**.addDataChangedListener(foods -> {  
 **mainTable**.setItems(FXCollections.*observableArrayList*(foods));  
 });  
  
  
 *// создаем столбец, указываем что столбец преобразует Food в String,  
 // указываем заголовок колонки как "Название"* TableColumn<Food, String> titleColumn = **new** TableColumn<>(**"Название"**);  
 *// указываем какое поле брать у модели Food  
 // в данном случае title, кстати именно в этих целях необходимо было создать getter и setter для поля title* titleColumn.setCellValueFactory(**new** PropertyValueFactory<>(**"title"**));  
  
 *// тоже самое для калорийности* TableColumn<Food, String> kkalColumn = **new** TableColumn<>(**"Калорийность"**);  
 kkalColumn.setCellValueFactory(**new** PropertyValueFactory<>(**"kkal"**));  
  
 *// добавляем столбец с описанием* TableColumn<Food, String> descriptionColumn = **new** TableColumn<>(**"Описание"**);  
 descriptionColumn.setCellValueFactory(cellData -> {  
 **return new** SimpleStringProperty(cellData.getValue().getDescription());  
 });  
  
 *// подцепляем столбцы к таблице* **mainTable**.getColumns().addAll(titleColumn, kkalColumn, descriptionColumn);  
  
 **cmbFoodType**.setItems(**foodTypes**);  
 **cmbFoodType**.getSelectionModel().select(0);  
 **cmbFoodType**.setConverter(**new** StringConverter<Class>() {  
 @Override  
 **public** String toString(Class object) {  
 **if** (Food.**class**.equals(object)) {  
 **return "Все"**;  
 } **else if** (Fruit.**class**.equals(object)) {  
 **return "Фрукты"**;  
 } **else if** (Chocolate.**class**.equals(object)) {  
 **return "Шоколад"**;  
 } **else if** (Cookie.**class**.equals(object)) {  
 **return "Булочка"**;  
 }  
 **return null**;  
 }  
  
 @Override  
 **public** Class fromString(String string) {  
 **return null**;  
 }  
 });  
 **cmbFoodType**.getSelectionModel().selectedItemProperty().addListener((observable, oldValue, newValue) -> {  
 **this**.**foodModel**.setFoodFilter((Class<? **extends** Food>) newValue);  
 });  
 }

и т. д.

# 5. Таблица тестов

Таблица 5.1 – Тесты для проверки работоспособности программы

| **№** | **Функция** | **Назначение** | **Входные данные** | **Выходные данные** |
| --- | --- | --- | --- | --- |
| 1 | Меню Файл/ Загрузить… | Загрузка данных из json файла | Файл data.json[[1]](#footnote-1) |  |
| 2 | Меню Файл/ Сохранить… | Сохранение данных в json файла | - | Файл data.json[[2]](#footnote-2) |
| 3 | Меню Данные/Добавить | Открытие формы на добавление файла | - |  |
| 4 | Добавление фрукта | Добавление записи с объектом фрукта |  |  |
| 5 | Добавление шоколадки | Добавление записи с объектом шоколадки |  |  |
| 6 | Добавление булочки | Добавление записи с объектом булочка |  |  |
| 7 | Фильтр по типу фрукты | Вывод только фруктов | Тип: фрукты |  |
| 8 | Фильтр по типу шоколад | Вывод только шоколадок | Тип: Шоколад |  |
| 9 | Фильтр по типу булочка | Вывод только булочек | Тип: булочка |  |

# Заключение

В рамках курсовой работы было разработано приложение, позволяющее работать с базой данных в формате JSON. Организация кода в программе соответствует архитектуре MVC. В приложении реализованы следующие возможности:

* Создание БД
* Запись БД в файл в формате .json
* Чтение БД из файла в формате .json
* Добавление записей в БД
* Удаление записей из БД
* Изменение записей в БД
* Выполнение поисковых запросов в БД

Приложение было протестировано и отлажено.

Все поставленные в рамках курсовой работы задачи были выполнены. В рамках выполнения данной работы был получен обширный спектр сведений об объектно-ориентированном языке программирования Java, событийно-ориентированной парадигме программирования, платформе JavaFX, способах организации код в соответствии с архитектурой MVC, а также опыт работы в среде разработки программного обеспечения JetBrains IntelliJ IDEA.

# Список использованной литературы

1. Лаврищева Е. М. Программная инженерия и технологии программирования сложных систем [Электронный ресурс] : Учебник / Лаврищева Е.М., 2018. - 432 с
2. Фаулер Мартин Шаблоны корпоративных приложений / Мартин Фаулер, 2016. – 664с
3. Шилдт Герберт Java 8. Полное руководство / Герберт Шилдт. – Вильямс, 2017. – 1376с
4. MVP в JavaFX, часть 1. [Электронный ресурс] // velvet.tealeaf.su : чаинка, 2020. URL: http://velvet.tealeaf.su/mvp-javafx-part-1.html (дата обращения: 01.12.2020).

# Приложение 1. Файл данных data.json

[ {  
 **"@class"** : **"sample.models.Fruit"**,  
 **"kkal"** : 100,  
 **"title"** : **"Яблоко"**,  
 **"id"** : 1,  
 **"isFresh"** : **true**}, {  
 **"@class"** : **"sample.models.Chocolate"**,  
 **"kkal"** : 200,  
 **"title"** : **"шоколад Аленка"**,  
 **"id"** : 2,  
 **"type"** : **"milk"**}, {  
 **"@class"** : **"sample.models.Cookie"**,  
 **"kkal"** : 300,  
 **"title"** : **"сладкая булочка с Маком"**,  
 **"id"** : 3,  
 **"withSugar"** : **true**,  
 **"withPoppy"** : **true**,  
 **"withSesame"** : **false**}, {  
 **"@class"** : **"sample.models.Fruit"**,  
 **"kkal"** : 200,  
 **"title"** : **"Апельсин"**,  
 **"id"** : 4,  
 **"isFresh"** : **true**}, {  
 **"@class"** : **"sample.models.Fruit"**,  
 **"kkal"** : 250,  
 **"title"** : **"Груша"**,  
 **"id"** : 5,  
 **"isFresh"** : **false**}, {  
 **"@class"** : **"sample.models.Chocolate"**,  
 **"kkal"** : 532,  
 **"title"** : **"Alpen Gold Миндаль и кокос"**,  
 **"id"** : 6,  
 **"type"** : **"white"**}, {  
 **"@class"** : **"sample.models.Chocolate"**,  
 **"kkal"** : 540,  
 **"title"** : **"Шоколад Бабаевский Горький"**,  
 **"id"** : 7,  
 **"type"** : **"black"**}, {  
 **"@class"** : **"sample.models.Cookie"**,  
 **"kkal"** : 294,  
 **"title"** : **"Кунжутные холмы"**,  
 **"id"** : 8,  
 **"withSugar"** : **true**,  
 **"withPoppy"** : **false**,  
 **"withSesame"** : **true**}, {  
 **"@class"** : **"sample.models.Cookie"**,  
 **"kkal"** : 300,  
 **"title"** : **"Ватрушка"**,  
 **"id"** : 9,  
 **"withSugar"** : **true**,  
 **"withPoppy"** : **false**,  
 **"withSesame"** : **false**}, {  
 **"@class"** : **"sample.models.Chocolate"**,  
 **"kkal"** : 560,  
 **"title"** : **"Merci Конфеты ассорти"**,  
 **"id"** : 10,  
 **"type"** : **"milk"**} ]

1. см. приложение 1 [↑](#footnote-ref-1)
2. см. приложение 1 [↑](#footnote-ref-2)