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

|  |
| --- |
| Вычислительная техника |

наименование кафедры

|  |
| --- |
| 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 |

|  |
| --- |
| 1.017.00.00 ПЗ |
| обозначение документа |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | ИСТб 17-1 |  |  |  | Мускатин И.А. |
|  | шифр группы |  | подпись |  | Фамилия И.О |
| Нормоконтроль |  |  |  |  | Каташевцев М.Д. |
|  |  |  | подпись |  | Фамилия И.О |

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

Иркутск 2019 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 | Мускатину Игорю Алексеевичу | |
|  | (фамилия, инициалы) | |
| Тема работы: | Обработка иерархии классов с использованием шаблонов | | |
| архитектурного проектирования | | | |
| Исходные данные: | | Вариант 7-3-2-2 | |
| 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 / 2019 г. | | |
| Задание получил студент | | | |  | |  | Мускатин И.А. |
|  | | | | подпись | |  | Фамилия И.О. |

|  |  |
| --- | --- |
| Дата представления работы руководителю | 20 / 12 / 2019 г. |
| Руководитель курсового проектирования |  |  | Каташевцев М.Д. |
|  | подпись | Фамилия И.О. |

Содержание

[Введение 4](#_Toc27737153)

[1. Индивидуальный вариант 5](#_Toc27737154)

[2. Диаграмма приложения и классов предметной области 6](#_Toc27737155)

[3. Внешний вид окон 7](#_Toc27737156)

[4. Код основных событий 8](#_Toc27737157)

[4.1 Описание модели 8](#_Toc27737158)

[4.2. Описание контроллера 11](#_Toc27737159)

[4.3. Описание контроллера транспорта 14](#_Toc27737160)

[5. Таблица тестов 16](#_Toc27737161)

[Заключение 21](#_Toc27737162)

[Список использованной литературы 22](#_Toc27737163)

[Приложение 1. Файл данных data.json 23](#_Toc27737164)

[Приложение 2. Файл данных data.json 24](#_Toc27737165)

# 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

• Выполнение двух запросов.

• Десериализация объектов из текстового файла в формате JSON

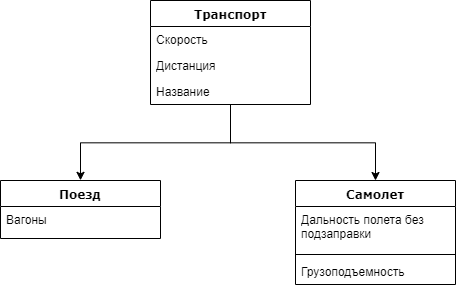
Язык программирования C# или Java.

C# – это язык программирования, предназначенный для разработки самых разнообразных приложений. Язык C# прост, строго типизирован и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую языкам C.

Java –язык программирования, во многом схожий с языком C#. Так же 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 или С#, 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



Выборочная печать: по типу транспорта

Запрос 1: Подсчет времени транспорта в пути

Запрос 2: Подсчет общего пути транспорта по выбранному типу

# 2. Диаграмма приложения и классов предметной области

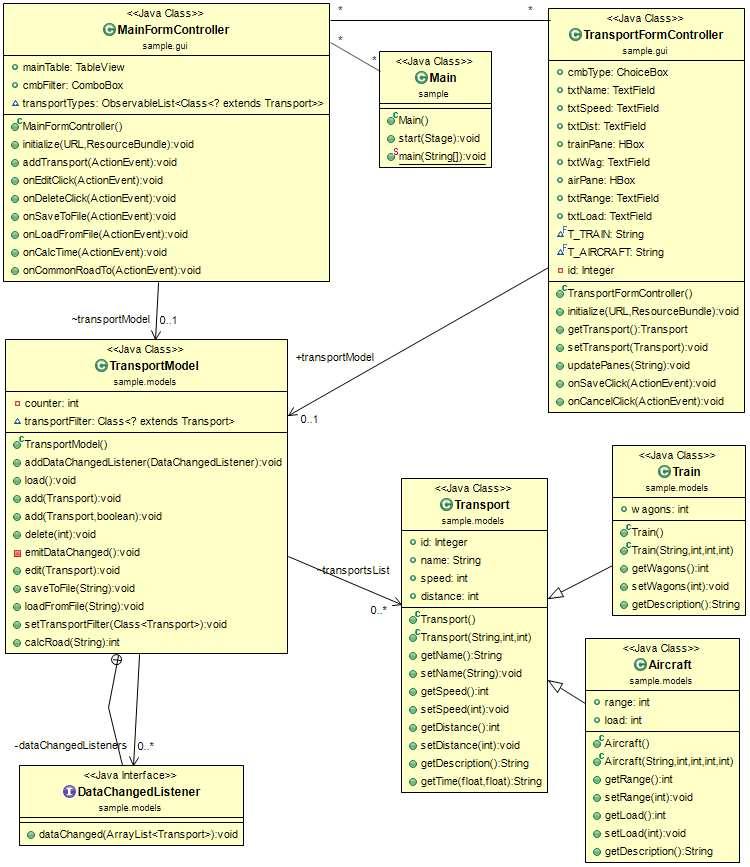


Схема 2.1 – Иерархия классов

# 3. Внешний вид окон

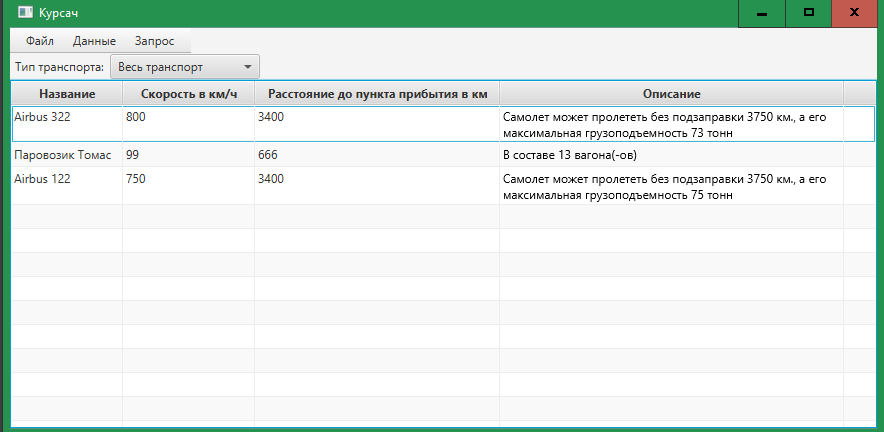


Рисунок 3.1 – Главная форма

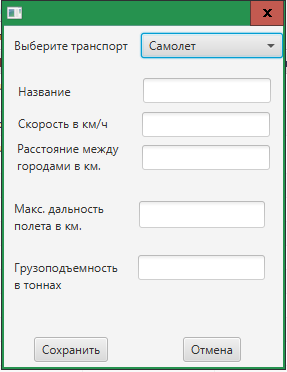
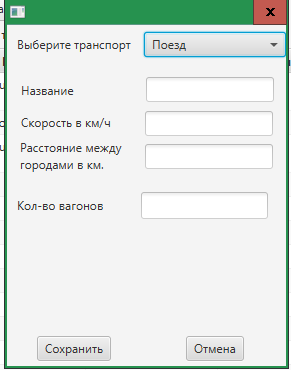


Рисунок 3.2 – Форма добавления/редактирования объекта

# 4. Код основных событий

## 4.1 Описание модели

public class TransportModel {

ArrayList<Transport> transportsList = new ArrayList<>();//список с данными

private int counter = 1;//счетчик

Class<? extends Transport> transportFilter = Transport.class;

public interface DataChangedListener {

void dataChanged(ArrayList<Transport> transportsList);

}

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)) {

// создаем сериализатор

ObjectMapper mapper = new ObjectMapper();

// записываем данные списка list в файл

mapper.writerFor(new TypeReference<ArrayList<Transport>>() {

}) // указали какой тип

.withDefaultPrettyPrinter()

.writeValue(writer, transportsList); // тут запись

} catch (IOException e) {

e.printStackTrace();

}

}

4.1.2 Загрузка из файла

public void loadFromFile(String path) {

try (Reader reader = new FileReader(path)) {

ObjectMapper mapper = new ObjectMapper();

transportsList = mapper.readerFor(new TypeReference<ArrayList<Transport>>() {

}).readValue(reader);

// рассчитываем счетчик как максимальное значение id плюс 1

this.counter = transportsList.stream()

.map(transport -> transport.id)

.max(Integer::compareTo)

.orElse(0) + 1;

} catch (IOException e) {

e.printStackTrace();

}

this.emitDataChanged();

}

4.1.3 Добавление экземпляра еды

public void add(Transport transport, boolean emit) {

transport.id = counter; // присваиваем id, значение счетчика

counter++; // увеличиваем счетчик на единицу

this.transportsList.add(transport);

if (emit) {

this.emitDataChanged();

}

}  
 public void add(Transport transport) {

add(transport, true);

}

4.1.4 Редактирование экземпляра еды

public void edit(Transport transport) {

// ищем объект в списке

for (int i = 0; i < this.transportsList.size(); ++i) {

// чтобы id совпадал с id переданным форме

if (this.transportsList.get(i).id == transport.id) {

// если нашли, то подменяем объект

this.transportsList.set(i, transport);

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

public void delete(int id) {

for (int i = 0; i < this.transportsList.size(); ++i) {

// ищем в цикле еду с данным айдишником

if (this.transportsList.get(i).id == id) {

// если нашли то удаляем

this.transportsList.remove(i);

break;

}

}

this.emitDataChanged();

}

4.1.7 Фильтрация по типу еды

public void setTransportFilter(Class<? extends Transport> transportFilter) {

this.transportFilter = transportFilter;

this.emitDataChanged();

}

4.1.8 Подсчет общего расстояния для определенного типа транспорта

public int calcRoad(String type) {

Integer road = 0;

if (type == "all") {

for (int i = 0; i < transportsList.size(); i++) {

road += transportsList.get(i).getDistance();

}

} else if (type == "train") {

for (int i = 0; i < transportsList.size(); i++) {

if (transportsList.get(i) instanceof Train) {

road += transportsList.get(i).getDistance();

}

}

} else if (type == "aircraft") {

for (int i = 0; i < transportsList.size(); i++) {

if (transportsList.get(i) instanceof Aircraft) {

road += transportsList.get(i).getDistance();

}

}

}

return road;

}

## 4.2. Описание контроллера

public class MainFormController implements Initializable {

public TableView mainTable;

public ComboBox cmbFilter;

public Label z1;

TransportModel transportModel = new TransportModel();

ObservableList<Class<? extends Transport>> transportTypes = FXCollections.observableArrayList(

Transport.class,

Train.class,

Aircraft.class

);

4.2.1 Инициализация контроллера

@Override

public void initialize(URL location, ResourceBundle resources) {

transportModel.addDataChangedListener(transports -> {

mainTable.setItems(FXCollections.observableArrayList(transports));

});

cmbFilter.getSelectionModel().selectedItemProperty().addListener((observable, oldValue, newValue) -> {

this.transportModel.setTransportFilter((Class<? extends Transport>) newValue);

});

//запрос 1 подсчет времени в пути

mainTable.getSelectionModel().selectedItemProperty().addListener((observable, oldValue, newValue) -> {

Transport transport = (Transport) observable.getValue();

z1.setText(String.format("%s пробудет в пути %s часа", transport.getName(), transport.getTime(transport.distance, transport.speed)));

});

cmbFilter.setItems(transportTypes);

// выбрали первый элемент в списке

cmbFilter.getSelectionModel().select(0);

// переопределил метод преобразования имени класса в текст

cmbFilter.setConverter(new StringConverter<Class>() {

@Override

public String toString(Class object) {

// просто перебираем тут все возможные варианты

if (Transport.class.equals(object)) {

return "Весь транспорт";

} else if (Train.class.equals(object)) {

return "Поезда";

} else if (Aircraft.class.equals(object)) {

return "Самолеты";

}

return null;

}

@Override

public Class fromString(String string) {

return null;

}

});

TableColumn<Transport, String> nameColumn = new TableColumn<>("Название");

nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<Transport, String> sumColumn = new TableColumn<>("Скорость в км/ч");

sumColumn.setCellValueFactory(new PropertyValueFactory<>("speed"));

TableColumn<Transport, String> distColumn = new TableColumn<>("Расстояние до пункта прибытия в км");

distColumn.setCellValueFactory(new PropertyValueFactory<>("distance"));

TableColumn<Transport, String> descriptionColumn = new TableColumn<>("Описание");

descriptionColumn.setCellValueFactory(cellData -> {

return new SimpleStringProperty(cellData.getValue().getDescription());

});

descriptionColumn.setCellFactory(x -> {

TableCell<Transport, String> cell = new TableCell<>();

Text text = new Text();

cell.setGraphic(text);

cell.setPrefHeight(Control.USE\_COMPUTED\_SIZE);

text.wrappingWidthProperty().bind(descriptionColumn.widthProperty());

text.textProperty().bind(cell.itemProperty());

return cell;

});

// подцепляем столбцы к таблице

mainTable.getColumns().addAll(nameColumn, sumColumn, distColumn, descriptionColumn);

}

4.2.2 Реакция на нажатие кнопки добавить

public void addTransport(ActionEvent actionEvent) throws IOException {  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("TransportForm.fxml"));  
 Parent root = loader.load();  
  
 // ну а тут создаем новое окно  
 Stage stage = new Stage();  
 stage.setScene(new Scene(root));  
 stage.setResizable(false);  
 stage.initModality(Modality.WINDOW\_MODAL);  
 stage.initOwner(this.mainTable.getScene().getWindow());  
  
 TransportFormController controller = loader.getController();  
 // передаем модель  
 controller.transportModel = transportModel;  
 // открываем окно и ждем пока его закроют  
 stage.showAndWait();  
}

4.2.3 Реакция на нажатие кнопки изменить

public void onEditClick(ActionEvent actionEvent) throws IOException {

FXMLLoader loader = new FXMLLoader();

loader.setLocation(getClass().getResource("TransportForm.fxml"));

Parent root = loader.load();

Stage stage = new Stage();

stage.setScene(new Scene(root));

stage.initModality(Modality.WINDOW\_MODAL);

stage.setResizable(false);

stage.initOwner(this.mainTable.getScene().getWindow());

TransportFormController controller = loader.getController();

controller.setTransport((Transport) this.mainTable.getSelectionModel().getSelectedItem());

controller.transportModel = transportModel; // передаем модель в контроллер

stage.showAndWait();

}

4.2.4 Реакция на нажатие кнопки удалить

public void onDeleteClick(ActionEvent actionEvent) {

Transport transport = (Transport) this.mainTable.getSelectionModel().getSelectedItem();

Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

alert.setTitle("Подтверждение");

alert.setHeaderText(String.format("Точно удалить %s?", transport.getName()));

// если пользователь нажал OK

Optional<ButtonType> option = alert.showAndWait();

if (option.get() == ButtonType.OK) {

transportModel.delete(transport.id);

}

}

4.2.5 Реакция на нажатие кнопки сохранить

public void onSaveToFile(ActionEvent actionEvent) {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Сохранить данные");

fileChooser.setInitialDirectory(new File("."));

File file = fileChooser.showSaveDialog(mainTable.getScene().getWindow());

if (file != null) {

transportModel.saveToFile(file.getPath());

}

}

4.2.6 Реакция на нажатие кнопки загрузить

public void onLoadFromFile(ActionEvent actionEvent) {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Загрузить данные");

fileChooser.setInitialDirectory(new File("."));

File file = fileChooser.showOpenDialog(mainTable.getScene().getWindow());

if (file != null) {

transportModel.loadFromFile(file.getPath());

}

}

4.2.7 Реакция на нажатие кнопки запрос 1

public void onCalcTime(ActionEvent actionEvent) {

Transport transport = (Transport) this.mainTable.getSelectionModel().getSelectedItem();

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Время в пути");

alert.setHeaderText("Запрос 1: нахождение времени пути транспорта");

alert.setContentText(String.format("%s пробудет в пути %s", transport.getName(), transport.getTime(transport.distance, transport.speed)));

alert.showAndWait();

}

4.2.8 Реакция на нажатие кнопки запрос 2

public void onCommonRoadTo(ActionEvent actionEvent) {

Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

alert.setTitle("Общий путь для...");

alert.setHeaderText("Запрос 2: нахождение общего пути всего транспорта определенного типа.");

ButtonType all = new ButtonType("Всего транспорта");

ButtonType train = new ButtonType("Поездов");

ButtonType aircraft = new ButtonType("Самолетов");

alert.getButtonTypes().clear();

alert.getButtonTypes().addAll(all, train, aircraft);

Optional<ButtonType> option = alert.showAndWait();

if (option.get() == all) {

alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Общий путь для всего транспорта");

alert.setHeaderText("Запрос 2: нахождение общего пути всего транспорта определенного типа");

alert.setContentText(String.format("Общий путь для всего транспорта составляет %s", transportModel.calcRoad("all")));

alert.showAndWait();

} else if (option.get() == train) {

alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Общий путь для поездов");

alert.setHeaderText("Запрос 2: нахождение общего пути всего транспорта определенного типа");

alert.setContentText(String.format("Общий путь для поездов составляет %s", transportModel.calcRoad("train")));

alert.showAndWait();

} else if (option.get() == aircraft) {

alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Общий путь для самолетов");

alert.setHeaderText("Запрос 2: нахождение общего пути всего транспорта определенного типа");

alert.setContentText(String.format("Общий путь для самолетов составляет %s", transportModel.calcRoad("aircraft")));

alert.showAndWait();

}

}

}

## 4.3. Описание контроллера транспорта

public class TransportFormController implements Initializable {  
 public ChoiceBox cmbType;  
 public TextField txtName;  
 public TextField txtSpeed;  
 public TextField txtDist;  
 public HBox trainPane;  
 public TextField txtWag;  
 public HBox airPane;  
 public TextField txtRange;  
 public TextField txtLoad;  
 public TransportModel transportModel;//модель  
 final String T\_TRAIN = "Поезд";

final String T\_AIRCRAFT = "Самолет";  
 private Integer id = null;

4.3.1 Инициализация контроллера

@Override

public void initialize(URL location, ResourceBundle resources) {

cmbType.setItems(FXCollections.observableArrayList(

T\_TRAIN,

T\_AIRCRAFT

));

cmbType.getSelectionModel().selectedItemProperty().addListener((observable, oldValue, newValue) -> {

updatePanes((String) newValue);

});

updatePanes("");

}

4.3.2 Запоминаем новый экземпляр с заполненной формы

public Transport getTransport() {

Transport result = null;

int speed = Integer.parseInt(this.txtSpeed.getText());

int distance = Integer.parseInt(this.txtDist.getText());

String name = this.txtName.getText();

switch ((String)this.cmbType.getValue()) {

case T\_TRAIN:

result = new Train(name, speed,distance, Integer.parseInt(this.txtWag.getText()));

break;

case T\_AIRCRAFT:

result = new Aircraft(name, speed,distance,Integer.parseInt(this.txtRange.getText()),Integer.parseInt(this.txtLoad.getText()));

break;

}

return result;

}

4.3.2 Заполняем форму редактирования для выбранного объекта

public void setTransport(Transport transport) {

// делаем так что если объект редактируется, то нельзя переключать тип

this.id = transport != null ? transport.id : null;

this.cmbType.setDisable(transport != null);

if (transport != null) {

// ну а тут стандартное заполнение полей в соответствии с переданной едой

this.txtName.setText(String.valueOf(transport.getName()));

this.txtSpeed.setText(String.valueOf(transport.getSpeed()));

this.txtDist.setText(String.valueOf(transport.getDistance()));

if (transport instanceof Train) { // если фрукт

this.cmbType.setValue(T\_TRAIN);

this.txtWag.setText(String.valueOf(((Train) transport).getWagons()));

} else if (transport instanceof Aircraft) { // если булочка

this.cmbType.setValue(T\_AIRCRAFT);

this.txtRange.setText(String.valueOf(((Aircraft) transport).getRange()));

this.txtLoad.setText(String.valueOf(((Aircraft) transport).getLoad()));

}

}

}

# 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 data1.json[[2]](#footnote-2) |
| 3 | Меню Данные/Добавить | Открытие формы на добавление файла | - |  |
| 4 | Добавление поезда | Добавление записи с объектом поезд |  |  |
| 5 | Добавление самолета | Добавление записи с объектом самолет |  |  |
| 7 | Фильтр по типу транспорт | Вывод всего транспорта | Тип: транспорт |  |
| 8 | Фильтр по типу поезд | Вывод только поездов | Тип: поезд |  |
| 9 | Фильтр по типу самолет | Вывод только самолетов | Тип: самолет |  |
| 10 | Запрос 1 подсчет времени пути для выбранного транспорта | Вывод времени пути транспорта | - |  |
| 11 | Запрос 2 подсчет общего пути по типу транспорта | Вывод общего пути для типа транспорта | Самолеты |  |

# Заключение

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

* Создание БД
* Запись БД в файл в форматах .json
* Чтение БД из файла в форматах .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 MVP, а также опыт работы в среде разработки программного обеспечения JetBrains IntelliJ IDEA.

# Список использованной литературы

1. Сосинская С. С. Использование языка C# в различных информационных технологиях : учебник для студентов вузов, обучающихся по направлению "Информационные системы и технологии" / С. С. Сосинская, 2014. - 367 с.
2. Лаврищева Е. М. Программная инженерия и технологии программирования сложных систем [Электронный ресурс] : Учебник / Лаврищева Е.М., 2018. - 432 с
3. Фаулер Мартин Шаблоны корпоративных приложений / Мартин Фаулер, 2016. – 664с
4. Шилдт Герберт Java 8. Полное руководство / Герберт Шилдт. – Вильямс, 2017. – 1376с
5. Васильев, А.Н. Программирование на C# для начинающих. Основные сведения. / А.Н. Васильев. – М.: Эксмо, 2018. 592 с.
6. Гуннерсон Э. Введение в C#. СПб: Питер, 2001.
7. Вагнер, Билл С# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 c.
8. Ишкова, Э. А. Самоучитель С#. Начала программирования / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 c.
9. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 486 c.
10. Биллинг В.А. Основы программирования на C#. - Т.: Интернет-университет информационных технологий, Бином, 2012. - 488 с.
11. Евдокимов П.В. C# на примерах. - М.: Наука и техника,2016. - 304 с.
12. Нейгел К., Ивьен Б., Глинн Д., Уотсон К., Скиннер М. C# 4.0 и платформа .NET 4, 2011.

# Приложение 1. Файл данных data.json

[ {

"@class" : "sample.models.Aircraft",

"id" : 1,

"name" : "Airbus 322",

"speed" : 800,

"distance" : 3400,

"range" : 3750,

"load" : 73

}, {

"@class" : "sample.models.Train",

"id" : 2,

"name" : "Паровозик Томас",

"speed" : 99,

"distance" : 666,

"wagons" : 13

}, {

"@class" : "sample.models.Aircraft",

"id" : 3,

"name" : "Airbus 122",

"speed" : 750,

"distance" : 3400,

"range" : 3750,

"load" : 75

} ]

# Приложение 2. Файл данных data1.json

[ {

"@class" : "sample.models.Aircraft",

"id" : 1,

"name" : "Airbus 322",

"speed" : 800,

"distance" : 3400,

"range" : 3750,

"load" : 73

}, {

"@class" : "sample.models.Train",

"id" : 2,

"name" : "Паровозик Томас",

"speed" : 99,

"distance" : 666,

"wagons" : 13

}, {

"@class" : "sample.models.Aircraft",

"id" : 3,

"name" : "Airbus 122",

"speed" : 750,

"distance" : 3400,

"range" : 3750,

"load" : 75

}, {

"@class" : "sample.models.Train",

"id" : 4,

"name" : "Паровоз",

"speed" : 75,

"distance" : 750,

"wagons" : 11

}, {

"@class" : "sample.models.Aircraft",

"id" : 5,

"name" : "Кукурузник",

"speed" : 650,

"distance" : 1234,

"range" : 2000,

"load" : 13

} ]

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