ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА
федеральное государственное бюджетное образовательное
учреждение высшего образования
«ОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПУТЕЙ СООБЩЕНИЯ»
(ОмГУПС (ОмИИТ))

Кафедра «Автоматика и системы управления»

Программа для обработки данных

Пояснительная записка к курсовой работе
по дисциплине «Объектно-ориентированное программирование»

ИНМВ.400000.000 ПЗ

Студент гр. _____
_____ Хусаинов К.А.
«__»_____2023 г.

Руководитель –
доцент кафедры АиСУ
_____ Альтман Е.А.
«__»_____2023 г.

Омск 2023

Реферат

УДК 004.42

Пояснительная записка к курсовой работе содержит 19 страниц, 10 рисунков, 5 использованных источника, 1 приложение.

Объектом курсовой работы является программа для вычисления рейтинга.

Цель курсовой работы – освоение и применение базы данных MongoDB в разработке программного обеспечения, а также использование ключевых принципов объектно-ориентированного программирования. Мы стремились получить практический опыт в работе с базами данных и создании программ, которые обеспечивают удобное управление и анализ данных.

Результатом курсовой работы является программа, которая подчитывает рейтинг студентов и формирует HTML страницу с результатами. Программа написана на языке программирования Kotlin в программе IntelliJ IDEA.

Пояснительная записка выполнена в текстовом редакторе Microsoft Word 2021.

Содержание

Введение

Объектно-ориентированное программирование представляет собой подход к разработке программного кода, где основное внимание уделяется организации данных и объектов, а не функций и логическим структурам. В ООП объекты являются основными строительными блоками и содержат набор характеристик и функций. Эти объекты могут представлять сущности как из реального мира, например, человека с различными данными, так и абстрактные утилиты с минимальными характеристиками, но большим набором функциональности. Взаимодействие объектов происходит как между собой, так и с пользователем и другими компонентами программы.

MongoDB – это система управления базами данных, основанная на документной модели и не требующая предварительного описания схемы таблиц. Она является примером NoSQL-системы и использует документы в формате JSON и схему базы данных. Написана на языке C++ и широко применяется в веб-разработке, особенно в рамках JavaScript-ориентированного стека MEAN.

MongoDB подходит для различных сценариев использования, включая регистрацию и хранение информации о событиях, управление документами и контентом, электронную коммерцию, игры, мониторинг данных от датчиков, мобильные приложения и хранение операционных данных веб-страниц, таких как комментарии, рейтинги, профили пользователей и сеансы.

Основные преимущества MongoDB заключаются в отсутствии строгой схемы, понятной структуре объектов, отсутствии сложных операций объединения данных (Join), расширенных возможностях запросов, гибкой настройке и простоте масштабирования. Кроме того, нет необходимости преобразовывать объекты приложения для сохранения в базе данных, а сама MongoDB использует внутреннюю память для хранения рабочего набора данных, обеспечивая быстрый доступ к информации.

1 Задание

Программа для вычисления рейтинга. Данные для расчета рейтингов хранятся в отдельной коллекции. Программа должна иметь возможность просмотреть формулу для расчета рейтинга, изменить формулу, рассчитать рейтинг на основании оценок из коллекции заданий, сформировать HTML страницу с результатами рейтинга.

Данные о студентах, их заданиях и весах заданий в программе загружаются из текстового формата JSON. А после, данные из файла JSON загружаются в базу данных MongoDB для дальнейшего расчета рейтинга и формирования HTML страницы.

Для создания HTML-разметки с использованием языка программирования Kotlin была использована библиотека kotlinx-html. Эта библиотека предоставляет удобные инструменты для генерации HTML-кода внутри кода Kotlin, что упрощает процесс создания и форматирования веб-страниц.

2 Реализация программы

Для реализации программы использовались следующие библиотеки:

`kotlin-stdlib` – для ввода и вывода данных на экран;

`kotlinx-serialization-json` – для сериализации данных в JSON формат;

`kmongo-serialization` – для работы с базой данных MongoDB и документами;

`slf4j-simple` – для вывода информации о системе и о подключении к MongoDB;

`kotlinx-html-jvm` – для создания кода разметки HTML на языке программирования Kotlin.

В программе используется 3 класса.

Для создания студента с оценками используется класс Student (листинг 2.1). В классе хранится информация о имени студента, возрасте, общих оценках, среднем балле, рейтинге и группе.

```
@Serializable
data class Student(
    val name: String,
    val age: Int,
    val marks: MutableMap<String, List<Int>>,
    val gpa: Double,
    val rating: Int,
    val group: String
)
```

Листинг 2.1 – Класс Student

Для создания задания по определенной дисциплине используется класс Task (листинг 2.2). В классе хранится информация о названии дисциплины, о задании, и о студентах, который получили оценки за данное задание.

```
@Serializable
data class Task(
    val subject: String,
    val task: String,
    val students: MutableMap<String, Int>
)
```

Листинг 2.2 – Класс Task

Для создания информации о весах заданий по определенной дисциплине используется класс Rating (листинг 2.3). В классе хранится информация о названии дисциплины и о заданиях и их весах.

```
@Serializable
data class Rating(
    val subject: String,
    var formula: MutableMap<String, Int>
)
```

Листинг 2.3 – Класс Rating

В программе используются девять функций:

Функция loadDataToDatabaseFromJson() предназначена для загрузки данных из JSON-файла в базу данных MongoDB. Данная функция имеет три параметра: path – путь к JSON-файлу, из которого нужно загрузить данные, collection – коллекция MongoDB, в которую нужно загрузить данные, serializer – сериализатор, который будет использоваться для преобразования JSON в объекты определенного класса. Первоначально функция считывает содержимое JSON-файла в виде строки, затем преобразует JSON-строку в список объектов, далее выполняет очистку коллекции базы данных MongoDB, и наконец она загружает все элементы списка в определенную коллекцию.

Функция loadDataToMongoDB() предназначена для удобной загрузки документов из JSON-файлов в базу данных MongoDB. Первоначально она предоставляет пользователю выбор, хочет ли он загрузить данные в базу данных. В случае положительного ответа, функция loadDataToMongoDB() вызывает функцию loadDataToDatabaseFromJson() и передает соответствующие аргументы. У данной функции нет параметров.

Функция printTasksAndWeights() предназначена для вывода информации о заданиях и их весах, хранящихся в коллекции Rating. Первоначально функция перебирает каждый документ коллекции Rating, а затем перебирает пару ключ-значение поля formula и выводит ключ и соответствующее значение. У данной функции нет параметров.

Функция setTaskWeight() предназначена для установки веса для конкретного задания по указанной дисциплине. Первоначально функция создает фильтр по названию дисциплины и проверяет в коллекции Rating наличие соответствующей дисциплины, если дисциплина не найдена выбрасывается исключение, иначе функция в поле formula пытается получить вес задания (значение) по ключу (задание), если значение не найдено (null) – выбрасывается исключение, иначе изменяется вес задания и происходит обновление данных в MongoDB. Данная функция имеет три параметра: subject – название дисциплины,

task – название задания, для которого нужно установить вес и weight – значение веса задания.

Функция printStudentRating() предназначена для вывода информации о рейтинге студентов, хранящихся в коллекции Student. Первоначально функция перебирает каждый документ коллекции Student и выводит имя студента, его группу и рейтинг. У данной функции нет параметров.

Функция calculateAndSetStudentRating() предназначена для расчета и установки рейтинга конкретного студента на основе оценок заданий и их весов. Первоначально функция создает фильтр по имени студента и проверяет его наличие в коллекции Student, если студент не найден выбрасывается исключение, иначе перебирается коллекция Task и получается оценка студента за определенное задание, далее получается вес этого определенного задания и считается рейтинг студента. После подсчета рейтинга, происходит обновление документа в коллекции Student. Данная функция принимает один параметр studentName – имя студента, для которого необходимо рассчитать рейтинг.

Функция createRatingHTMLPage() предназначена для создания HTML страницы с результатами рейтинга и открытия сформированной таблицы в браузере. Первоначально функция формирует таблицу, в которую записываются данные, далее создается HTML файл и предлагается открыть сформированный файл в браузере. У данной функции нет параметров.

Функция selectionMenu() предназначена для отображения меню выбора действий и выполнения соответствующих действий на основе выбора пользователя, если выбор некорректен выбрасывается исключение. Функция продолжается до тех пор, пока пользователь не выйдет из программы. У данной функции нет параметров.

Функция main() является точкой входа в программу и определяет ход ее выполнения. Первоначально функция вызывает функцию loadDataToMongoDB(), затем программа входит в цикл, внутри которого вызывается функция selectionMenu(), если во время выполнения программы выбрасывается исключение, то обработчик перехватывает его и выводит сообщение на экран. У данной функции нет параметров.

3 Инструкция пользователя

При запуске программы нам предлагается загрузить JSON-файлы в базу данных MongoDB (рисунки 1-4).

```
Загрузить документы в базу данных "MongoDB"? (Да/Нет): Да
Данные успешно загружены в MongoDB.
```

Рисунок 1 – Загрузка документов в базу данных


```
[{
  "name": "Sophia Williams",
  "age": 18,
  "marks": {
    "OOP": [3, 4, 5, 4, 5],
    "TOAPS": [4, 4, 3, 5, 4],
    "Electronics": [2, 2, 3, 2, 5],
    "Mathematics": [4, 5, 4, 5, 3],
    "Economics": [3, 3, 2, 3, 2],
    "Physics": [2, 3, 3, 2, 5],
    "Metrology": [4, 2, 2, 5, 4]
  },
  "gpa": 3.4,
  "rating": 0,
  "group" : "21z"
},
```


Рисунок 2 – Пример содержимого JSON-файла студентов


```
[
  {
    "subject": "Economics",
    "task": "Что такое инфляция?",
    "students": {
      "Sophia Williams": 3,
      "Ethan Kim": 4,
      "Avery Thomas": 2,
      "Sophia Taylor": 5,
      "Mason Anderson": 3,
      "Isabella Mitchell": 2,
      "Aria Smith": 5,
      "Lucas Rodriguez": 4,
      "Liam Johnson": 2,
      "Olivia Smith": 3,
      "William Garcia": 4,
      "Emma Robinson": 2,
      "Noah Rodriguez": 5,
      "Isabella Hernandez": 3,
      "James Nguyen": 4,
      "Sophie Lee": 2
    }
```


Рисунок 3 – Пример содержимого JSON-файла заданий и оценок

```
[{
    "subject": "Electronics",
    "formula": {
        "Какой элемент электронной цепи выглядит как буква T?": 5,
        "Что такое диод?": 7,
        "Что такое транзистор?": 4,
        "Что такое операционный усилитель?": 10
    }
}, {
```


Рисунок 4 – Пример содержимого JSON-файла заданий и их весов

Установим веса 10 и 7 соответственно к заданиям по дисциплинам «OOP» и «TOAPS» (рисунки 5-6).

```
Введите действие, которое необходимо выполнить:
1 - Просмотреть все задачи и их веса.
2 - Установить вес для определенной задачи.
3 - Просмотреть студентов и их рейтинг.
4 - Рассчитать рейтинг для определенного студента (для всех).
5 - Сформировать html страницу с результатами рейтинга.
0 - Выход из программы.
Ввод: 2
Введите название дисциплины: OOP
Введите название задания: Каким образом можно реализовать наследование в Java?
Введите вес задания: 10
Заданию "Каким образом можно реализовать наследование в Java?", по дисциплине "OOP", задан вес - 10.
```

Рисунок 5 – Установление веса к заданию по OOP

```
Введите действие, которое необходимо выполнить:
1 - Просмотреть все задачи и их веса.
2 - Установить вес для определенной задачи.
3 - Просмотреть студентов и их рейтинг.
4 - Рассчитать рейтинг для определенного студента (для всех).
5 - Сформировать html страницу с результатами рейтинга.
0 - Выход из программы.
Ввод: 2
Введите название дисциплины: TOAPS
Введите название задания: Что такое фильтр нижних частот?
Введите вес задания: 7
Заданию "Что такое фильтр нижних частот?", по дисциплине "TOAPS", задан вес - 7.
```

Рисунок 6 – Установление веса к заданию по TOAPS

Посчитаем рейтинг для всех студентов (рисунки 7-8).

```
Введите действие, которое необходимо выполнить:
1 - Просмотреть все задачи и их веса.
2 - Установить вес для определенной задачи.
3 - Просмотреть студентов и их рейтинг.
4 - Рассчитать рейтинг для определенного студента (для всех).
5 - Сформировать html страницу с результатами рейтинга.
0 - Выход из программы.
Ввод: 4

Введите имя студента (введите "Все", чтобы посчитать рейтинг для всех студентов): Все
```

Рисунок 7 – Вызов функции расчета рейтинга

```
Студенту "Sophia Williams", группа "21z" выставлен рейтинг 73.
Студенту "Liam Johnson", группа "21z" выставлен рейтинг 72.
Студенту "Olivia Smith", группа "21i" выставлен рейтинг 62.
Студенту "William Garcia", группа "21z" выставлен рейтинг 66.
Студенту "Emma Robinson", группа "21m" выставлен рейтинг 67.
Студенту "Noah Rodriguez", группа "21z" выставлен рейтинг 69.
Студенту "Isabella Hernandez", группа "21m" выставлен рейтинг 75.
Студенту "James Nguyen", группа "21i" выставлен рейтинг 60.
Студенту "Sophie Lee", группа "21i" выставлен рейтинг 75.
Студенту "Ethan Kim", группа "21i" выставлен рейтинг 71.
Студенту "Avery Thomas", группа "21z" выставлен рейтинг 72.
Студенту "Sophia Taylor", группа "21m" выставлен рейтинг 67.
Студенту "Mason Anderson", группа "21i" выставлен рейтинг 61.
Студенту "Isabella Mitchell", группа "21z" выставлен рейтинг 72.
Студенту "Aria Smith", группа "21m" выставлен рейтинг 75.
Студенту "Lucas Rodriguez", группа "21z" выставлен рейтинг 69.
Всем студентам выставлен рейтинг.
```

Рисунок 8 – Итог

Сформируем HTML страницу с результатами и откроем ее (рисунки 9-10).

```
Введите действие, которое необходимо выполнить:
1 - Просмотреть все задачи и их веса.
2 - Установить вес для определенной задачи.
3 - Просмотреть студентов и их рейтинг.
4 - Рассчитать рейтинг для определенного студента (для всех).
5 - Сформировать html страницу с результатами рейтинга.
0 - Выход из программы.
Ввод: 5

HTML страница с результатами рейтингов сформирована по пути: "C:/Users/KhusainovKA/IdeaProjects/Coursework_OOP/src/main/kotlin/Rating.html".
Открыть HTML страницу? (Да/Нет): Да
Открытие...
```

Рисунок 9 – Формирование HTML страницы и открытие

Таблица рейтингов студентов

Имя студента	Группа	Рейтинг
Sophia Williams	21z	74
Liam Johnson	21z	72
Olivia Smith	21i	62
William Garcia	21z	66
Emma Robinson	21m	67
Noah Rodriguez	21z	69
Isabella Hernandez	21m	57
James Nguyen	21i	61
Sophie Lee	21i	76
Ethan Kim	21i	72
Avery Thomas	21z	72
Sophia Taylor	21m	67
Mason Anderson	21i	57
Isabella Mitchell	21z	72
Aria Smith	21z	96
Lucas Rodriguez	21z	66

Рисунок 10 – Итог

Заключение

За время выполнения курсовой работы мы создали программу для вычисления рейтинга студентов, в которой были применены различные концепции программирования и инструменты, мы также освоили работу с различными библиотеками такими как: kotlin-stdlib, kotlinx-serialization-json, kmongo-serialization, slf4j-simple и kotlinx-html-jvm.

Результатом выполнения курсовой работы является программа для вычисления рейтинга. В программе мы использовали базу данных MongoDB для хранения информации о студентах, их заданиях и весах. Также были реализованы функции для просмотра задач, установки веса для определенной задачи, просмотра рейтинга студентов и расчета рейтинга как для отдельного студента, так и для всех студентов сразу. Для вывода результатов рейтинга была создана HTML страница, в которой ячейки подсвечиваются в зависимости от значения рейтинга студента.

Библиографический список

1 Википедия. Свободная энциклопедия [Электронный ресурс] / Режим доступа: https://ru.wikipedia.org/wiki/Kotlin

2 Kotlin. Документация [Электронный ресурс] / Режим доступа: https://kotlinlang.org/docs/home.html

3 Википедия. Свободная энциклопедия [Электронный ресурс] / Режим доступа: https://ru.wikipedia.org/wiki/MongoDB

4 GitHub. Репозиторий [Электронный ресурс] / Режим доступа: https://github.com/kotlin/kotlinx.html/wiki

5 MongoDB. Документация [Электронный ресурс] / Режим доступа: https://www.mongodb.com/docs/

Приложение А

Код программы «Connect.kt»

```
import com.mongodb.client.MongoDatabase
import org.litote.kmongo.KMongo

val client = KMongo.createClient("mongodb://localhost:27017")

val mongoDatabase: MongoDatabase =
client.getDatabase("Coursework")
```

Листинг А.1, лист 1

Код программы «Main.kt»

```
import com.mongodb.client.MongoCollection
import org.litote.kmongo.getCollection

val mStudent: MongoCollection<Student> =
mongoDatabase.getCollection<Student>()
val mTask: MongoCollection<Task> =
mongoDatabase.getCollection<Task>()
val mRating: MongoCollection<Rating> =
mongoDatabase.getCollection<Rating>()

const val pathToJsonStudents: String =
"C:/Users/KhusainovKA/IdeaProjects/Coursework_OOP/" +
        "src/main/kotlin/Database/Students.json"
const val pathToJsonTasks: String =
"C:/Users/KhusainovKA/IdeaProjects/Coursework_OOP/" +
        "src/main/kotlin/Database/Tasks.json"
const val pathToJsonRatings: String =
"C:/Users/KhusainovKA/IdeaProjects/Coursework_OOP" +
        "/src/main/kotlin/Database/Ratings.json"
const val pathToSaveHTMLPage: String =
"C:/Users/KhusainovKA/IdeaProjects/Coursework_OOP" +
        "/src/main/kotlin/Rating.html"

fun main() {
    loadDataToMongoDB()
    do {
        try {
            selectionMenu()
        }
        catch (error: ProgramExceptions) {
            println(error.message)
        }
    } while (!exit)
    client.close()
}
```

Листинг А.2, лист 1

Код программы «Exceptions.kt»

```
open class ProgramExceptions(message: String): Exception(message)

object WrongCommandException: ProgramExceptions("Ошибка:
некорректный выбор действия.")

class SubjectException(string: String): ProgramExceptions(string)

class TaskException(string: String): ProgramExceptions(string)

class StudentException(string: String): ProgramExceptions(string)
```

Листинг А.3, лист 1

Код программы «Student.kt»

```
import kotlinx.serialization.Serializable

@Serializable
data class Student
(
    val name: String,
    val age: Int,
    val marks: MutableMap<String, List<Int>>,
    val gpa: Double,
    val rating: Int,
    val group: String
)
```

Листинг А.4, лист 1

Код программы «Task.kt»

```
import kotlinx.serialization.Serializable

@Serializable
data class Task
(
    val subject: String,
    val task: String,
    val students: MutableMap<String, Int>
)
```

Листинг А.5, лист 1

Код программы «Rating.kt»

```
import kotlinx.serialization.Serializable

@Serializable
data class Rating
(
    val subject: String,
    var formula: MutableMap<String, Int>
)
```

Листинг А.6, лист 1

Код программы «Selection.kt»

```
var exit = false
fun selectionMenu() {
    while (!exit) {
        println("\nВведите действие, которое необходимо
выполнить:")
        println("1 - Просмотреть все задачи и их веса.")
        println("2 - Установить вес для определенной задачи.")
        println("3 - Просмотреть студентов и их рейтинг.")
        println("4 - Рассчитать рейтинг для определенного студента
(для всех).")
        println("5 - Сформировать html страницу с результатами
рейтинга.")
        println("0 - Выход из программы.")
        print("Ввод: ")
        when (readln()) {
            "0" -> {
                println("Выход из программы...")
                exit = true
            }
            "1" -> printTasksAndWeights()
            "2" -> {
                print("Введите название дисциплины: ")
                val subject = readln()
                print("Введите название задания: ")
                val task = readln()
                print("Введите вес задания: ")
                val weight = readln().toIntOrNull() ?: 0
                setTaskWeight(subject, task, weight)
            }
            "3" -> printStudentRating()
            "4" -> {
                print("\nВведите имя студента (введите \"Все\",
чтобы посчитать рейтинг для всех студентов): ")
                val studentName = readln()
                when (studentName.lowercase()) {
                    "все" -> {
                        mStudent.find().forEach {
                            calculateAndSetStudentRating(it.name)
                        }
                        println("Всем студентам выставлен
рейтинг.")
                    }
                    else -> calculateAndSetStudentRating(studentName)
                }
            }
            "5" -> createRatingHTMLPage()

            else -> throw WrongCommandException
        } } }
```

Листинг А.7, лист 1

Код программы «Functions.kt»

```
import com.mongodb.client.MongoCollection
import kotlinx.html.*
import kotlinx.html.stream.appendHTML
import kotlinx.serialization.KSerializer
import kotlinx.serialization.builtins.ListSerializer
import kotlinx.serialization.json.Json
import org.litote.kmongo.eq
import org.litote.kmongo.setValue
import java.awt.Desktop
import java.io.File
import kotlin.math.roundToInt

fun <T> loadDataToDatabaseFromJson(path: String, collection:
MongoCollection<T>, serializer: KSerializer<List<T>>) {
    val json = File(path).readText()
    val col = Json.decodeFromString(serializer, json)
    collection.drop()
    collection.insertMany(col)
}

fun loadDataToMongoDB() {
    print("Загрузить документы в базу данных \"MongoDB\"?
(Да/Нет): ")
    if (readln().lowercase() == "да") {
        loadDataToDatabaseFromJson(pathToJsonStudents, mStudent,
ListSerializer(Student.serializer()))
        loadDataToDatabaseFromJson(pathToJsonTasks, mTask,
ListSerializer(Task.serializer()))
        loadDataToDatabaseFromJson(pathToJsonRatings, mRating,
ListSerializer(Rating.serializer()))
        println("Данные успешно загружены в MongoDB.")
    }
}

fun printTasksAndWeights() {
    mRating.find().forEach { rating ->
        print("\n")
        println("Дисциплина - \"${rating.subject}\"")
        rating.formula.forEach {
            println("Задание - \"${it.key}\", вес задания -
${it.value}")
        }
    }
}

fun setTaskWeight(subject: String, task: String, weight: Int) {
    val filterSubject = Rating::subject eq subject
    val rating = mRating.find(filterSubject).firstOrNull()
        ?: throw SubjectException("Ошибка: дисциплина:
\"$subject\" не найдена.")
```

Листинг А.8, лист 1

```
    if (rating.formula[task] == null) {
        throw TaskException("Ошибка: задание: \"$task\" не
найдено.")
    }
    rating.formula[task] = weight
    mRating.updateOne(
        filterSubject,
        setValue(Rating::formula, rating.formula)
    )
    println("Заданию \"$task\", по дисциплине \"$subject\", задан
вес - $weight.")
}

fun printStudentRating() {
    print("\n")
    println("Рейтинг студентов")
    mStudent.find().forEach { student ->
        println("Студент - \"${student.name}\", группа -
\"${student.group}\", рейтинг - ${student.rating}")
    }
}

fun calculateAndSetStudentRating(studentName: String) {
    val filterName = Student::name eq studentName
    val student = mStudent.find(filterName).firstOrNull()
        ?: throw StudentException("Ошибка: студент
\"$studentName\" не найден.")
    var totalRating = 0
    var studentRating = 0
    mTask.find().forEach { task ->
        val grade = task.students[studentName]!!
        val weight = mRating.find().first { rating ->
rating.formula.containsKey(task.task) }.formula[task.task]!!
        studentRating += grade * weight // Рейтинг студента
        totalRating += 5 * weight // Максимальный рейтинг
    }
    val finalRating = ((studentRating.toDouble() /
totalRating.toDouble()) * 100).roundToInt()
    mStudent.updateOne(
        filterName, setValue(Student::rating, finalRating)
    )
    println("Студенту \"${student.name}\", группа
\"${student.group}\" выставлен рейтинг ${finalRating.")
}

fun createRatingHTMLPage() {
    val htmlString = buildString {
        appendHTML().table {
            style = "font-size: 25px; border: 1px solid black;
margin: auto"
```

Листинг А.8, лист 2

```
            caption { +"Таблица рейтингов студентов" }

            tr {
                style = "text-align: center; background-color:
black; color: white;"
                th { +"Имя студента" }
                th { +"Группа"}
                th { +"Рейтинг" }
            }
            mStudent.find().forEach { student ->
                tr {
                    style = when (student.rating) {
                        in 0..59 -> {
                            "text-align: center; background-color:
red;"
                        }

                        in 60..75 -> {
                            "text-align: center; background-color:
yellow;"
                        }

                        in 76..92 -> {
                            "text-align: center; background-color:
green;"
                        }

                        else -> {
                            "text-align: center; background-color:
lime;"
                        }
                    }
                    td { +student.name }
                    td { +student.group }
                    td { +student.rating.toString() }
                }
            }
        }
    }
    val htmlFile = File(pathToSaveHTMLPage).apply {
writeText(htmlString) }
    println("\nHTML страница с результатами рейтингов сформирована
по пути: \"$pathToSaveHTMLPage\".")
    print("Открыть HTML страницу? (Да/Нет): ")
    if (readln().lowercase() == "да") {
        println("Открытие...")
        Desktop.getDesktop().browse(htmlFile.toURI())
    }
}
```

Листинг А.8, лист 3