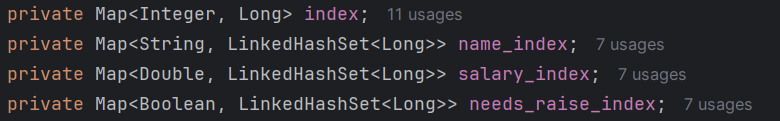
**Языкова Мария. Лабораторная работа №2.**  
  
**Предметная область:** справочник, хранящий и управляющий иформацией о сотрудниках. Позволяет эффективно отслеживать данные о каждом работнике, включая идентификатор, имя, зарплату и информацию о необходимости повышения.



* ID – ключевое поле (тип int)
* Name – имя сотрудника (тип String)
* Salary – зарплата сотрудника (тип double)
* Needs Raise – желание получить повышение (тип boolean)

**Анализ временной сложности алгоритмов:**

****

Для оптимизации обращения к записи были созданы хеш-таблицы для каждого из полей со следующей структурой: ключ – название поля, значения – позиции в файле формата .csv (от начала файла).

**Добавление записи O(1)**

****

Проверка наличия ключа в хеш-таблице – O(1)

Открытие файла для записи – O(1)

Нахождение текущей длины файла – O(1)

Формирование строки – O(m), где m – кол-во полей, так как m фиксировано и не зависит от конкретных данных, будет считать, что сложность – **константная.**

Запись строки в файл – O(n), где n – длина записи, будем считать, что сложность **– константная.**

Вставка значений в хеш-таблицы – O(1)

**Удаление записи по ключевому полю O(1)**

****

Получение значения по ключу из хеш-таблицы – O(1)

Проверка наличия значения в хеш-таблице – O(1)

Открытие файла для чтения-записи – O(1)

Перемещение к позиции – O(1)

Чтение строки – O(n), где n – длина записи, будем считать, что сложность **– константная.**

Разделение строки по запятым - O(m), где m – кол-во полей, так как m фиксировано и не зависит от конкретных данных, будет считать, что сложность – **константная.**

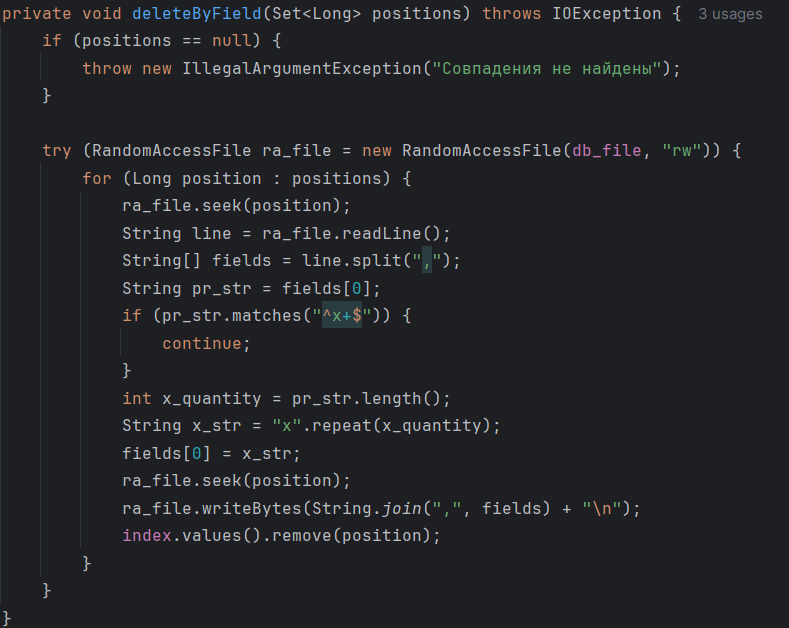
Замена первого элемента на маркер – O(1), принебрегая длиной первого элемента (id).

Перемещение к позиции – O(1)

Запись строки в файл - O(n), где n – длина записи, будем считать, что сложность **– константная.**

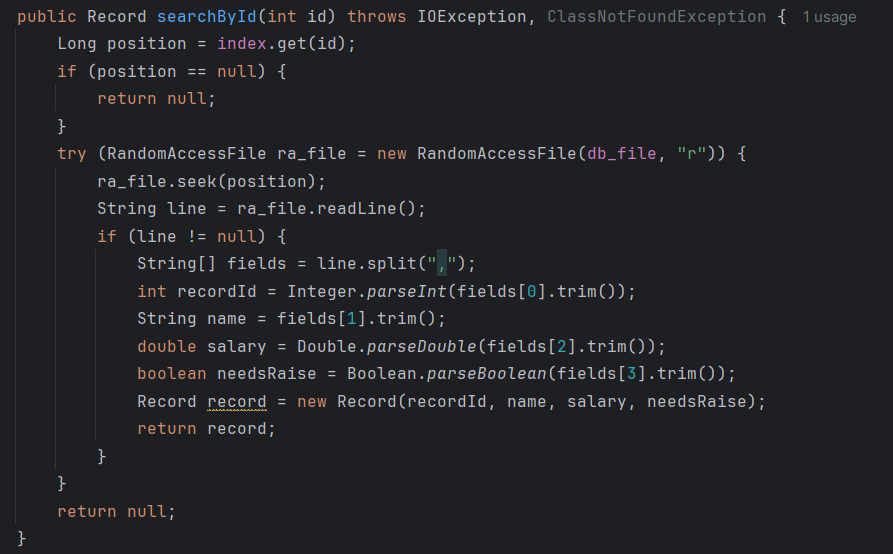
Удаление элемента из зеш-таблицы по ключу – O(1)

**Удаление записи по не ключевому полю O(k), где k – кол-во совпадений по ключу**

****

Удаление происходит аналогично удалению по ключевому полю, только теперь передается не одна позиция, а список позиций. Обработка происходит в цикле, который проходится по переданным позициям – отсюда сложность O(k), где k – кол-во переданных позиций.

**Поиск записи по ключевому полю O(1)**



Получение значение по ключу – O(1)

Проверка на наличие значения в хеш-таблице – O(1)

Открытие файла для чтения – O(1)

Перемещение курсора – O(1)

Чтение строки - O(n), где n – длина записи, будем считать, что сложность **– константная.**

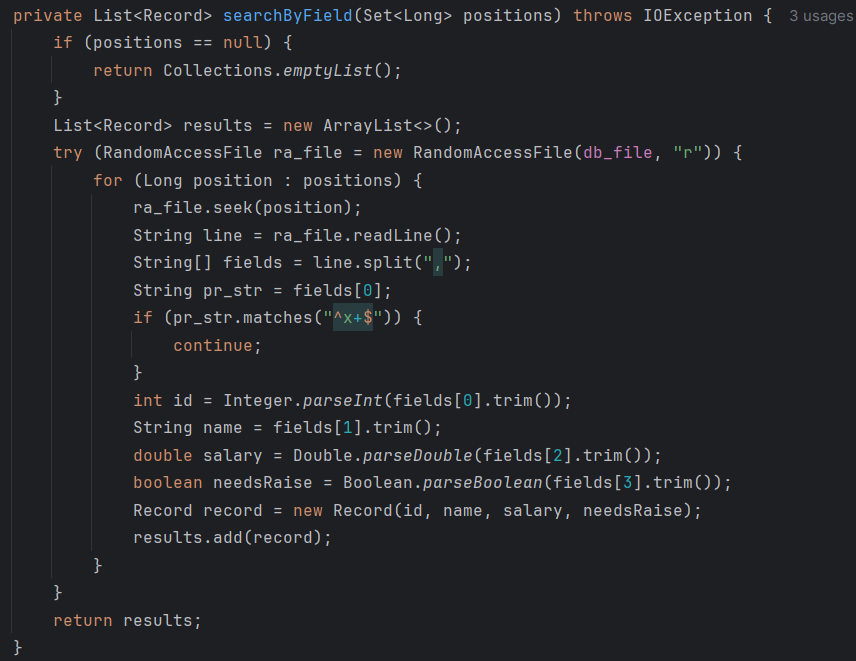
Проверка на null – O(1)

Разделение строки по запятым - O(m), где m – кол-во полей, так как m фиксировано и не зависит от конкретных данных, будет считать, что сложность – **константная.**

Преобразование строк в элементы – O(1)

Создание объекта Record – O(1)

**Поиск записи по не ключевому полю O(k)**



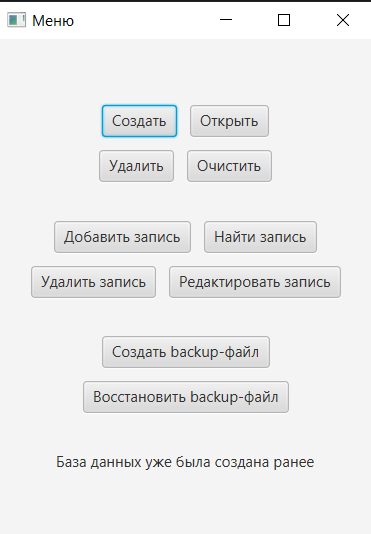
Поиск происходит аналогично поиску по ключевому полю, только теперь передается не одна позиция, а список позиций. Обработка происходит в цикле, который проходится по переданным позициям – отсюда сложность O(k), где k – кол-во переданных позиций.

**Реализованный функционал**

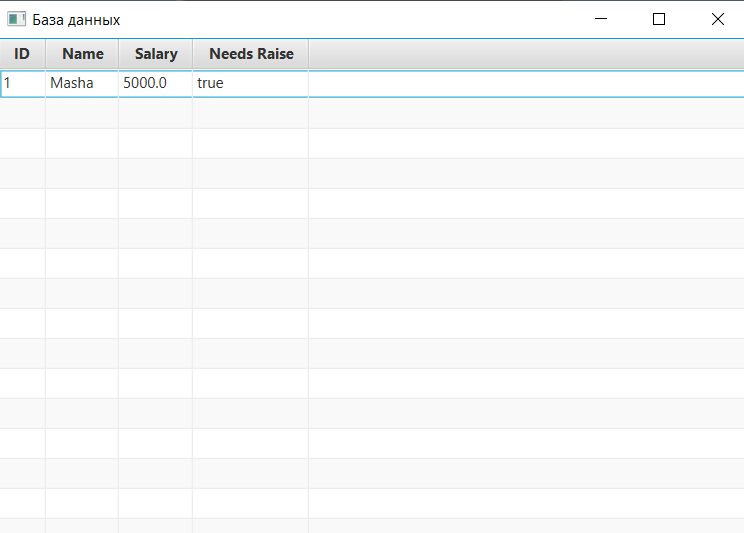
* создание, открытие, удаление, очистка, сохранение БД,
* добавление новой записи в БД (с проверкой уникальности по ключевым полям),
* удаление записи из БД по значению некоторого поля (ключевого и не ключевого (в последнем случае удаляются все записи, совпадающие по значению)),
* поиск по БД по значению некоторого поля (ключевого и не ключевого (в последнем случае найти нужно все записи, совпадающие по значению)) с выводом на экран результатов поиска,
* редактирование записи БД,
* создание backup-файла БД,
* восстановление БД из backup-файла

**Интерфейс**

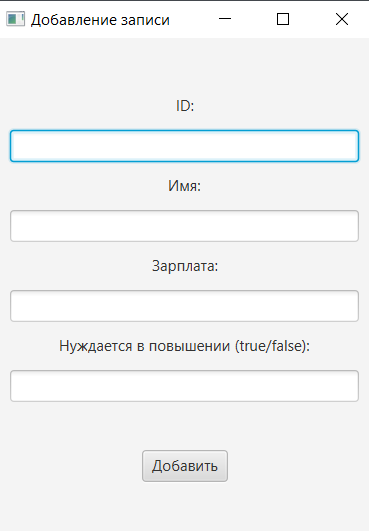
Главный экран



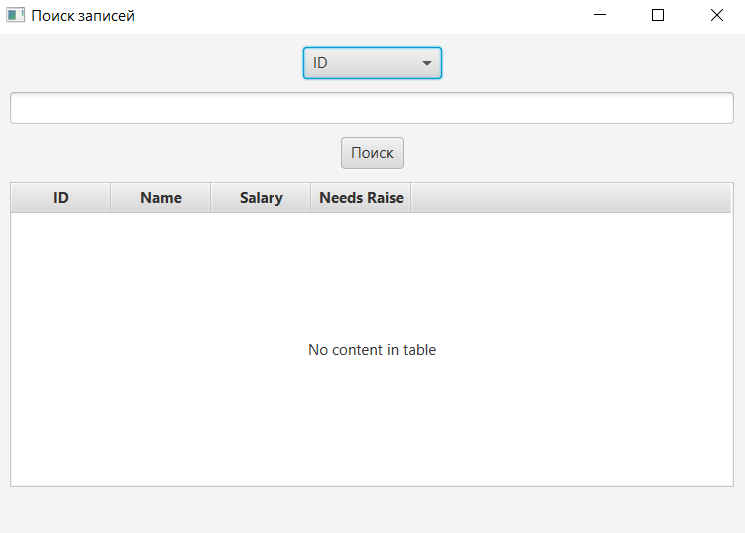
Открыть ->



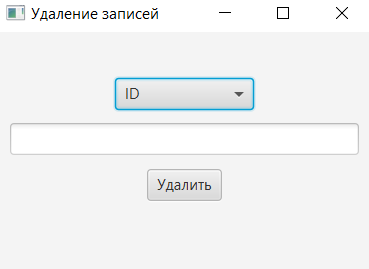
Добавить запись ->



Найти запись ->



Удалить запись ->



Редактировать запись ->

