

# Библиотека autocomplete вводимого текста

## Постановка задачи

Требуется написать консольное Java-приложение (JDK 11), позволяющее быстро искать данные аэропортов по вводимому пользователем названию аэропорта и фильтрам.

Данные для программы берутся из файла [airports.csv](#). В нем находится таблица аэропортов со свойствами в формате CSV. Название аэропорта — 2 колонка. За что отвечают другие колонки — не важно, на них навешиваются фильтры.

Фильтры могут быть — отношения равенства: равно (=), не равно (<>), больше (>), меньше (<). Фильтр передается в формате:

```
column[1]>10 & column[5]='GKA' || column[номер колонки с 1]<операция сравнения>...
```

Фильтры могут соединяться отношением И (&) и ИЛИ (||). Также могут участвовать скобки для обозначения приоритета и группировки. Отношение И имеет более высокий приоритет нежели ИЛИ. Фильтр может быть не указан.

Сборка проекта осуществляется с помощью Maven. После сборки исходного кода командой `mvn clean package`, получаем `airports-search-*.jar` в качестве артефакта для запуска.

- > Пользователь запускает приложение:

```
java -jar airports-search-*.jar // запуск приложения
```

- > После запуска программа выводит в консоль предложение ввести фильтр.

Пользователь вводит «`column[1]>10&column[5]='GKA'`», нажимает «`Enter`».

- > Программа просит ввести начало имени аэропорта.

Например, пользователь вводит «`Bo`» и нажимает «`Enter`». Программа выводит список всех строк из файла [airports.csv](#), вторая колонка которых начинается на «`Bo`», с учетом фильтров, при этом строки отсортированные по колонке имени аэропорта в формате «`<Найденное значение нужной колонки>[<Полностью строка>]`».

Не буквенные и не цифровые символы также участвуют в поиске. Регистр букв не имеет значения.

- > После вывода всех строк программа должна вывести число найденных строк и время в миллисекундах, затраченное на поиск.

```
"Bowman Field"[3600, "Bowman Field", "Louisville", "United States", "LOU", "KLOU", 38.22800064]
Количество найденных строк: 68 Время, затраченное на поиск: 25 мс
```

- > Затем предложить снова ввести текст фильтра для поиска.

- > Для того, чтобы завершить программу, нужно ввести текст «`!quit`».

# Библиотека autocomplete вводимого текста

## Нефункциональные требования

1. Перечитывать все строки файла при каждом поиске нельзя.  
В том числе читать только определенную колонку у каждой строки.
  2. Создавать новые файлы или редактировать текущий нельзя.  
В том числе использовать СУБД.
  3. Хранить весь файл в памяти нельзя.  
Не только в качестве массива байт, но и в структуре, которая так или иначе содержит все данные из файла.
  4. Для корректной работы программе требуется не более 7 МБ памяти.  
Все запуски `java -jar` должны выполняться с `jvm` флагом `-Xmx7m`.
  5. Скорость поиска должна быть максимально высокой с учетом требований выше.  
В качестве ориентира можно взять число из скриншота выше: на поиск по «Во», который выдает 68 строк, требуется 25 мс, поиск по «Bower», который выдает 1 строку без фильтров — 5 мс.
  6. Сложность поиска меньше чем  $O(n)$ , где  $n$  — число строк файла.
  7. Должны соблюдаться принципы ООП и SOLID.
  8. Ошибочные и краевые ситуации должны быть корректно обработаны.
  9. Использовать готовые библиотеки для парсинга CSV формата нельзя.
  10. Решенное тестовое задание — код в публичном репозитории на GitHub. По готовности ссылку на репозиторий отправить в чат в Telegram контакту, от которого было получено задание.
- i** В случае, если возникает вопрос, который не покрывает данная постановка задачи, кандидат должен сам выбрать любое его решение, не противоречащее постановке. В readme должно быть отражен вопрос и принятое решение.