

ТЕСТОВОЕ ЗАДАНИЕ СТАЖИРОВКИ JAVA-РАЗРАБОТЧИКОВ

Библиотека autocomplete вводимого текста

Постановка задачи

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

Данные для программы берутся из файла [airports.dat](#). В нем находится таблица аэропортов со свойствами в формате CSV. За что отвечает каждая колонка – не важно.

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, указывая параметром номер колонки (нумерация начинается с 1, а не с 0) по которой требуется поиск:

```
java -jar airports-search-*.jar 2 // запуск приложения в режиме поиска по второй колонке
```

После запуска программа выводит в консоль предложение ввести текст. Например, пользователь вводит «Bo» и нажимает «Enter». Программа выводит список всех строк из файла [airports.dat](#), вторая колонка которых начинается на «Bo», отсортированных по этой колонке в формате «<Найденное значение нужной колонки>[<Полностью строка>]».

Для строковых колонок, значения которых заключены в кавычки, должен быть лексикографический порядок, а для числовых – числовой. Не буквенные и не цифровые символы также участвуют в поиске. Регистр букв не имеет значения.

Пример

Введите строку:

Bo

```
"Bo Airport"[566, "Bo", "Sierra Leone", "KBS", "6F80", "7.944399833679199, -11.76899967956543, 328, 0, "N", "Africa/Freetown", "airport", "OurAirports"]
"Bob Baker Memorial Airport"[7179, "Bob Baker Memorial Airport", "Kiana", "United States", "IAN", "PAIK", "66.9759979248, -168.43699646, 166, -9, "A", "America/Anchorage", "airport", "OurAirports"]
"Bob Hope Airport"[8477, "Bob Hope Airport", "Burbank", "United States", "BUR", "KBUR", "34.20069885253906, -118.35980115966797, 778, -8, "A", "America/Los_Angeles", "airport", "OurAirports"]
"Bob Quinn Lake Airport"[8477, "Bob Quinn Lake Airport", "Canada", "YBO", "CBW4", "56.9667915976, -139.25, 2800, -8, "U", "America/Vancouver", "airport", "OurAirports"]
"Bob Sikes Airport"[10555, "Bob Sikes Airport", "Crestview", "United States", "CEW", "KCEW", "30.778799857, -86.522102356, 213, -6, "A", "America/Chicago", "airport", "OurAirports"]
"Bobo Dioulasso Airport"[247, "Bobo Dioulasso Airport", "Bobo-dioulasso", "Burkina Faso", "BOY", "DF00", "11.160899983215332, -4.33896981848584, 1511, 0, "N", "Africa/Ouagadougou", "airport", "OurAirports"]
"Boca Raton Airport"[13831, "Boca Raton Airport", "Boca Raton", "United States", "BCT", "KBCT", "26.36861111111111, -80.1876965332, 13, -5, "A", "America/New_York", "airport", "OurAirports"]
```

```
> java -jar airports-search-*.jar 2
```

```
"Bowman Field"[3600, "Bowman Field", "Louisville", "United States", "LOU", "KLOU", "38.2280006409, -85.6636962891, 546, -5, "A", "America/New_York", "airport", "OurAirports"]
Количество найденных строк: 68 Затраченное на поиск: 25 мс
```

Количество найденных строк: 68
Затраченное на поиск: 25 мс

- > После вывода всех строк программа должна вывести число найденных строк и время в миллисекундах, затраченное на поиск.
- > Затем предложить снова ввести текст для поиска.
- > Для того, чтобы завершить программу, нужно ввести текст «quit».

ТЕСТОВОЕ ЗАДАНИЕ СТАЖИРОВКИ JAVA-РАЗРАБОТЧИКОВ

Библиотека `autocomplete` вводимого текста

Нефункциональные требования

- > Перечитывать все строки файла при каждом поиске нельзя (в том числе читать только определенную колонку у каждой строки).
- > Создавать новые файлы или редактировать текущий нельзя (в том числе использовать СУБД).
- > 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).
- > Для корректной работы программе требуется **не более 7 МБ** памяти (все запуски `java -jar` должны выполняться с `jvm` флагом `-Xmx7m`).
- > 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**).
- > Сложность поиска меньше чем $O(n)$, где n — число строк файла.
- > Должны соблюдаться принципы **ООП** и **SOLID**.
- > Ошибочные и краевые ситуации должны быть корректно обработаны.
- > Использовать готовые библиотеки для парсинга CSV формата нельзя.
- > Решенное тестовое задание — код в публичном репозитории на GitHub.



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.