#777 - Backend sort method should have a option to ignore leading whitespace when sorting

primeira.solucao.odt
@jorge.leao
2022-03-25

1. Primeiro problema (mais simples):

Desenvolver uma forma de ordenar dados fornecidos como um **array** em uma *“string json”* em JAVA.

Embora a string json seja um array, cada elemento do array é um objeto que possui uma chave (aKey) comum a todos os elementos deste array, além de outros pares chave-valor.

Exemplo:

```
[
  {"aKey":" valor7 Abraháo uber2","zzz": "yyy"},
  {"aKey":" valor7 Abraháo über","zzz": "yyy"},
  {"zzz": "yyy", "aKey":" valor7 Abrahãm über"},
  {"aKey":"valor5","zzz": "yyy"},
  {"aKey":"  valor8","zzz": "yyy"},
  {"aKey":"valor6","zzz": "yyy"},
  {"zzz": "yyy","aKey":"  valor2"},
  {"zzz": "yyy","anotherKey":"  valor33"},
  {"aKey":" valor4","zzz": "yyy"},
  {"aKey":" valor-1","zzz": "yyy"},
  {"aKey":" valor+1","zzz": "yyy"},
  {"aKey":" valor*1","zzz": "yyy"},
  {"aKey":" valor1","zzz": "yyy"},
  {"aKey":" valor5","zzz": "yyy"},
  {"aKey":"   00valor5","zzz": "yyy"},
  {"aKey":" valor9","zzz": "yyy"}
]
```

Observe que a linha em amarelo não possui a chave “aKey”.

Supõe-se o caso geral. em que a posição do par “key:value” em cada objeto do array é arbitrária e irrelevante.

O processo de ordenamento deve ignorar os espaços em branco na frente dos valores, mas não alterá-los.

Se houver um elemento do array sem a chave “aKey”, o objeto json será excluido da saída ordenada.

A saída do ordenamento foi:


```
[
  {"aKey":"   00valor5","zzz":"yyy"},
  {"aKey":" valor*1","zzz":"yyy"},
  {"aKey":" valor+1","zzz":"yyy"},
  {"aKey":" valor-1","zzz":"yyy"},
  {"aKey":" valor1","zzz":"yyy"},
  {"aKey":"  valor2","zzz":"yyy"},
  {"aKey":" valor4","zzz":"yyy"},
  {"aKey":" valor5","zzz":"yyy"},
  {"aKey":"valor6","zzz":"yyy"},
  {"aKey":" valor7 Abrahãm über","zzz":"yyy"},
  {"aKey":" valor7 Abraháo über","zzz":"yyy"},
  {"aKey":" valor7 Abraháo uber2","zzz":"yyy"},
  {"aKey":"  valor8","zzz":"yyy"},
  {"aKey":" valor9","zzz":"yyy"},
  null
]
```


2. O código para fazer esta ordenação é:

```
    public static void main(String[] args) {
        Gson gson = new Gson();
        TreeMapStringObjects.initHashMap();
        TwoKeyJsonObject[] twoKeyJsonObjectsSaida = null;
        String saida = null;
        try (Reader reader = new FileReader("jsonArray.json")) {
            TwoKeyJsonObject[] twoKeyJsonObjects =
                    gson.fromJson(reader, TwoKeyJsonObject[].class);
            TreeMap<String, TwoKeyJsonObject> firstMap = new TreeMap<>();
            stream(twoKeyJsonObjects)
                    .map(x -> { String str =
RemoveDiacriticals.remove(x.getaKey().trim().toUpperCase(Locale.ROOT));
                                firstMap.put(str,x);
                                return null;})
                    .collect(Collectors.toList());
            twoKeyJsonObjectsSaida = new TwoKeyJsonObject[twoKeyJsonObjects.length];
            int i = 0;
            for (String s : firstMap.keySet()) {
                System.out.println(s);
                twoKeyJsonObjectsSaida[i] = firstMap.get(s);
                i++;
            }
            saida = gson.toJson(twoKeyJsonObjectsSaida);
        } catch (IOException e) {
            e.printStackTrace();
        }
        try (FileWriter writer = new FileWriter("jsonArraySaida.json")){
            System.out.println(saida);
            gson.toJson(twoKeyJsonObjectsSaida, writer);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Os valores da chave “aKey” nos objetos do array foram trimmed, toUpperCase e tiveram os diacríticos (acentos, cedilha, etc) removidos.

Uma listagem destas chaves modificadas, que foram usadas para o ordenamento, é:

```
00VALOR5
VALOR*1
VALOR+1
VALOR-1
VALOR1
VALOR2
VALOR4
VALOR5
VALOR6
VALOR7 ABRAHAM UBER
VALOR7 ABRAHAO UBER
VALOR7 ABRAHAO UBER2
VALOR8
VALOR9
```

3. Este código completo pode ser obtido no Github:

https://github.com/jorgeleao/Sorting.a.Json.Array.by.the.values.of.a.key

Fim do documento