

課題3_1班_杉谷一祝

```

1 package jp.project1.testsumlist;
2
3 /*****
4  *   試験結果集計プログラム：メインクラス
5  *
6  *   プログラム名： TestSumList
7  *   概要：       テキストファイル内の試験結果データを集計し表示
8  *   作成日付：   2020/09/25
9  *   版数：       1.1版
10  *   作成者(班:PL)：杉谷一祝(1:京岡大純)
11  *   修正履歴：   1.1版   順位の算出及びインデント整列を修正
12  *   備考：       なし
13  *   課題No：     3
14  *****/
15
16 import java.io.IOException;
17 import java.nio.charset.Charset;
18 import java.nio.file.Files;
19 import java.nio.file.Paths;
20 import java.util.Arrays;
21 import java.util.Comparator;
22 import java.util.List;
23 import java.util.regex.Pattern;
24 import java.util.stream.Collectors;
25
26 /**
27  * TestSumListクラス
28  */
29 public class TestSumList {
30     /** 異常終了コード */
31     public static final int ABNORMAL = -1;
32     /** データにある科目数 */
33     public static final int SCORES_QUANTITY = 3;
34     /** データの氏名に該当するインデックス */
35     public static final int NAME_INDEX = 0;
36     /** データの氏名の次に該当するインデックス */
37     public static final int NAME_INDEX_NEXT = NAME_INDEX + 1;
38     /** ランクを求める際のひとつ前のインデックス算出用 */
39     public static final int PREV = 1;
40     /** ランク算出用 */
41     public static final int ONE = 1;
42     /** prev_rankの初期値 */
43     public static final int PREV_RANK_INIT = 0;
44     /** for文カウンタ変数の初期値 */
45     public static final int ZERO = 0;
46     /** データファイルのパス */
47     public static final String FILE_PATH = "C:/Users/5191007/Desktop/wsjava/TestSumList/bin/testsum.txt";
48     /** データファイルの文字コード */
49     public static final String CHARSET = "MS932";
50     /** 入出力エラー時メッセージ */
51     public static final String E001 = "入出力エラーが発生しました。";
52     /** 異常終了時メッセージ */
53     public static final String I001 = "強制終了します。";
54     /** データ区切り文字 */
55     public static final String SPLITER = ",";
56     /** 正常データの正規表現 */
57     public static final String REGEX = "^[YyAa](?:[YyWw,])+(?:-|[0-9]?[0-9]|100)){" + SCORES_QUANTITY + "}[YyZz]";
58     /** 出力フォーマット用 */
59     public static final String FORMAT_1 = "%";
60     /** 出力フォーマット用 */
61     public static final String FORMAT_2 = "d%s%-";
62     /** 出力フォーマット用 */
63     public static final String FORMAT_3 = "s";
64     /** 出力フォーマット用 */
65     public static final String FORMAT_4 = " %s%";
66     /** 出力フォーマット用 */
67     public static final String FORMAT_5 = "d";
68     /** 試験成績順位の見出し */
69     public static final String RANKING_TITLE = "【試験成績順位】";
70     /** 再試験者の見出し */
71     public static final String RETESTERS_TITLE = "【再試験者】";
72     /** 再試験該当者なしの場合のメッセージ */
73     public static final String NON_RETESTER_MS = "該当者なし";
74     /** 得点の最大値に付けるマーク */
75     public static final String MAX_MARK = "*";
76     /** 得点の最大値以外に付けるマーク */
77     public static final String NON_MAX_MARK = " ";
78
79     /** 各科目の最高得点を算出する際Studentのscoresのインデックスに使用 */
80     public static int counter;
81     /** ランクを求める際に使用するひとつ前のランク格納用 */
82     public static int prev_rank = PREV_RANK_INIT;
83
84
85 /*****
86  *   メインメソッド：main(String[] args)
87  *
88  *   メソッド名：   main
89  *   概要：       プログラムエントリ
90  *   引数：       String[] args： コマンドライン引数
91  *   返却値：     なし
92  *   備考：       なし
93  *****/
94     /**
95     * メインメソッド
96     * @param args
97     */
98     public static void main(String[] args) {

```

```

99      /* データ格納用変数の初期化 */
100     List<Student> students = List.of();
101
102     /* ファイルから正常データを抽出 */
103     try {
104         students = Files.lines(Paths.get(FILE_PATH), Charset.forName(CHARSET))
105             .filter(s -> Pattern.matches(REGEX, s))
106             .map(s -> s.split(SPLITER))
107             .map(s -> new Student(s[NAME_INDEX], Arrays.stream(s).skip(NAME_INDEX_NEXT).mapToInt(Integer::parseInt).toArray()))
108             .collect(Collectors.toList());
109     } catch (IOException e) {
110         System.out.println(E001 + I001);
111         System.exit(ABNORMAL);
112     }
113
114     /* 再試験者の氏名を抽出 */
115     List<String> retesters = students.stream().filter(Student::getIsRetester).map(Student::getName).collect(Collectors.toList());
116
117     /* 試験成績順位を出力 */
118     if (students.stream().anyMatch(s -> !s.getIsRetester())) {
119         /* 見出し */
120         System.out.println(RANKING_TITLE);
121
122         /* 合計得点の降順、氏名の昇順で並び替え */
123         students = students.stream().sorted(Comparator.comparing(Student::getSum, Comparator.reverseOrder()).thenComparing(Student::getName)).collect(Collectors.toList());
124
125         /* 順位を算出 1.1版 */
126         for (int i = ZERO; i < students.size(); i++) {
127             int rank = prev_rank == PREV_RANK_INIT || students.get(i).getSum() != students.get(i - PREV).getSum() ? i + ONE : prev_rank;
128             students.get(i).rank = rank;
129             prev_rank = rank;
130         }
131
132         /* 最高点を算出 */
133         int sum_max = students.stream().max(Comparator.comparingInt(Student::getSum)).get().getSum();
134         int [] scores_max = new int[SCORES_QUANTITY];
135         for (counter = ZERO; counter < SCORES_QUANTITY; counter++) {
136             scores_max[counter] = students.stream().map(Student::getScores).mapToInt(s -> s[counter]).max().getAsInt();
137         }
138
139         /* インデント数の設定 */
140         int rank_len = String.valueOf(students.stream().max(Comparator.comparingInt(s -> s.rank)).get().rank).length(); // 1.1版
141         int [] scores_len = Arrays.stream(scores_max).map(s -> String.valueOf(s).length()).toArray();
142
143         /* 氏名の最大長 */
144         int name_max_len = students.stream().max((s1, s2) -> s1.getName().length() - s2.getName().length()).get().getName().length();
145
146         /* 一覧 */
147         students.forEach(s -> {
148             /* 合計得点の最大値にマークを付ける */
149             String sum_max_mark = s.getSum() == sum_max ? MAX_MARK : NON_MAX_MARK;
150
151             /* 氏名のインデント数の設定 */
152             int name_len = name_max_len + name_max_len - s.getName().length();
153
154             /* フォーマットして出力 */
155             String result = String.format(FORMAT_1 + rank_len + FORMAT_2 + name_len + FORMAT_3, s.rank, sum_max_mark, s.getName());
156             int [] scores = s.getScores();
157             for (int j = ZERO; j < SCORES_QUANTITY; j++) {
158                 /* 各科目の最高得点にマークを付ける */
159                 String score_max_mark = scores[j] == scores_max[j] ? MAX_MARK : NON_MAX_MARK;
160
161                 result += String.format(FORMAT_4 + scores_len[j] + FORMAT_5, score_max_mark, scores[j]);
162             }
163             System.out.println(result);
164         });
165     }
166
167     /* 再試験者を出力 */
168     System.out.println(RETESTERS_TITLE);
169     if (!retesters.isEmpty()) {
170         /* 該当者ありの場合 */
171         retesters.forEach(System.out::println);
172     } else {
173         /* 該当者なしの場合 */
174         System.out.println(NON_RETESTER_MS);
175     }
176 }
177 }

```