

## 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：       1.0版
10 *   作成者(班:PL)：杉谷一祝(1:京岡大純)
11 *   修正履歴：   なし
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 = "D:\\Users\\¥¥5191007¥¥Documents¥¥ws¥¥TestSumList¥¥testdata.txt";
48     /** データファイルの文字コード */
49     public static final String CHARSET = "UTF-8";
50     /** 入出力エラーメッセージ */
51     public static final String E001 = "I/O エラーが発生しました。";
52     /** データ区切り文字 */
53     public static final String SPLITER = ",";
54     /** 正常データの正規表現 */
55     public static final String REGEX = "^(?:¥¥W+)(?:" + SPLITER + "(?:-1|[0-9]?[0-9]|100)){" + SCORES_QUANTITY + "} $" ;
56     /** 出力フォーマット用 */
57     public static final String FORMAT_1 = "%";
58     /** 出力フォーマット用 */
59     public static final String FORMAT_2 = "%d%s%-";
60     /** 出力フォーマット用 */
61     public static final String FORMAT_3 = "%s";
62     /** 出力フォーマット用 */
63     public static final String FORMAT_4 = " %s%";
64     /** 出力フォーマット用 */
65     public static final String FORMAT_5 = "%d";
66     /** 試験成績順位の見出し */
67     public static final String TITLE_1 = "【試験成績順位】";
68     /** 再試験者の見出し */
69     public static final String TITLE_2 = "【再試験者】";
70     /** 再試験該当者なしの場合のメッセージ */
71     public static final String TITLE_3 = "該当者なし";
72     /** 得点の最大値に付けるマーク */
73     public static final String MAX_MARK = "*";
74     /** 得点の最大値以外に付けるマーク */
75     public static final String NON_MAX_MARK = " ";
76
77     /** 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 */
78     public static int counter;
79     /** ランクを求める際に使用するひとつ前のランク格納用 */
80     public static int prev_rank = PREV_RANK_INIT;
81
82
83 /*****
84  *   メインメソッド：main(String[] args)
85  *
86  *   メソッド名：   main
87  *   概要：       プログラムエントリ
88  *   引数：       String[] args : コマンドライン引数
89  *   返却値：     なし
90  *   備考：       なし
91 *****/
92 /**
93  * メインメソッド
94  * @param args
95  */
96 public static void main(String[] args) {
97     /** データ格納用変数の初期化 */
98     List<Student> students = List.of();

```

```

99
100  /* ファイルから正常データを抽出 */
101  try {
102      students = Files.lines(Paths.get(FILE_PATH), Charset.forName(CHARSET))
103          .filter(s -> Pattern.matches(REGEX, s))
104          .map(s -> s.split(SPLITER))
105          .map(s -> new Student(s[NAME_INDEX], Arrays.stream(s).skip(NAME_INDEX_NEXT).mapToInt(Integer::parseInt).toArray()))
106          .collect(Collectors.toList());
107  } catch (IOException e) {
108      System.out.println(E001);
109      System.exit(ABNORMAL);
110  }
111
112  /* 再試験者の氏名を抽出 */
113  List<String> retesters = students.stream().filter(s -> s.isRetester).map(s -> s.name).collect(Collectors.toList());
114
115  /* 合計得点の降順、氏名の昇順で並び替え */
116  students = students.stream().sorted(Comparator.comparing(Student::getSum, Comparator.reverseOrder()).thenComparing(s -> s.name)).collect(Collectors.toList());
117
118  /* 最高点を算出 */
119  int sum_max = students.stream().max(Comparator.comparingInt(Student::getSum)).get().getSum();
120  int [] scores_max = new int[SCORES_QUANTITY];
121  for (counter = ZERO; counter < SCORES_QUANTITY; counter++) {
122      scores_max[counter] = students.stream().mapToInt(s -> s.scores[counter]).max().getAsInt();
123  }
124
125  /* 試験成績順位を出力 */
126  if (students.stream().anyMatch(s -> !s.isRetester)) {
127      /* 見出し */
128      System.out.println(TITLE_1);
129
130      /* インデント数の設定 */
131      int rank_len = String.valueOf(students.size()).length();
132      int [] scores_len = Arrays.stream(scores_max).map(s -> String.valueOf(s).length()).toArray();
133
134      /* 氏名の最大長 */
135      int name_max_len = students.stream().max((s1, s2) -> s1.name.length() - s2.name.length()).get().name.length();
136
137      /* 一覧 */
138      for (int i = ZERO; i < students.size(); i++) {
139          /* 合計得点の最大値にマークを付ける */
140          String sum_max_mark = students.get(i).getSum() == sum_max ? MAX_MARK : NON_MAX_MARK;
141
142          /* ランクを求める */
143          int rank = prev_rank == PREV_RANK_INIT || students.get(i).getSum() != students.get(i - PREV).getSum() ? i + ONE : prev_rank;
144          prev_rank = rank;
145
146          /* 氏名のインデント数の設定 */
147          int name_len = name_max_len + name_max_len - students.get(i).name.length();
148
149          /* フォーマットして出力 */
150          String result = String.format(FORMAT_1 + rank_len + FORMAT_2 + name_len + FORMAT_3, rank, sum_max_mark, students.get(i).name);
151          for (int j = ZERO; j < SCORES_QUANTITY; j++) {
152              /* 各科目の最高得点にマークを付ける */
153              String score_max_mark = students.get(i).scores[j] == scores_max[j] ? MAX_MARK : NON_MAX_MARK;
154
155              result += String.format(FORMAT_4 + scores_len[j] + FORMAT_5, score_max_mark, students.get(i).scores[j]);
156          }
157          System.out.println(result);
158      }
159  }
160
161  /* 再試験者を出力 */
162  System.out.println(TITLE_2);
163  if (!retesters.isEmpty()) {
164      /* 該当者ありの場合 */
165      retesters.forEach(System.out::println);
166  } else {
167      /* 該当者なしの場合 */
168      System.out.println(TITLE_3);
169  }
170  }
171  }

```