```java
package jp.project1.testsumlist;

/**********************************************************************************************
 *   試験結果集計プログラム：メインクラス
 *
 *   プログラム名： TestSumList
 *   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
 *   作成日付：     2020/09/25
 *   版数：         1.0版
 *   作成者(班:PL)：杉谷一祝(1:京岡大純)
 *   修正履歴：     なし
 *   備考：         なし
 *   課題No：       3
 **********************************************************************************************/

import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;
import java.util.regex.Pattern;
import java.util.stream.Collectors;

/**
 * TestSumListクラス
 */
public class TestSumList {
    /** 異常終了コード */
    public static final int     ABNORMAL        = -1;
    /** データにある科目数 */
    public static final int     SCORES_QUANTITY = 3;
    /** データの氏名に該当するインデックス */
    public static final int     NAME_INDEX      = 0;
    /** データの氏名の次に該当するインデックス */
    public static final int     NAME_INDEX_NEXT = NAME_INDEX + 1;
    /** ランクを求める際のひとつ前のインデックス算出用 */
    public static final int     PREV            = 1;
    /** ランク算出用 */
    public static final int     ONE             = 1;
    /** prev_rankの初期値 */
    public static final int     PREV_RANK_INIT  = 0;
    /** for文カウンター変数の初期値 */
    public static final int     ZERO            = 0;
    /** データファイルのパス */
    public static final String  FILE_PATH       = "C:/Users/5191007/Desktop/wsjava/TestSumList/bin/testsum.txt";
    /** データファイルの文字コード */
    public static final String  CHARSET         = "MS932";
    /** 入出力エラーメッセージ */
    public static final String  E001            = "I/O エラーが発生しました。";
    /** データ区切り文字 */
    public static final String  SPLITER         = ",";
    /** 正常データの正規表現 */
    public static final String  REGEX           = "¥¥A(?:[^¥¥w,]+)(?:" + SPLITER + "(?:-1|[0-9]?[0-9]|100)){" + SCORES_QUANTITY + "}¥¥Z";
    /** 出力フォーマット用 */
    public static final String  FORMAT_1        = "%";
    /** 出力フォーマット用 */
    public static final String  FORMAT_2        = "d%s%-";
    /** 出力フォーマット用 */
    public static final String  FORMAT_3        = "s";
    /** 出力フォーマット用 */
    public static final String  FORMAT_4        = "  %s%";
    /** 出力フォーマット用 */
    public static final String  FORMAT_5        = "d";
    /** 試験成績順位の見出し */
    public static final String  TITLE_1         = "【試験成績順位】";
    /** 再試験者の見出し */
    public static final String  TITLE_2         = "【再試験者】";
    /** 再試験該当者なしの場合のメッセージ */
    public static final String  TITLE_3         = "該当者なし";
    /** 得点の最大値に付けるマーク */
    public static final String  MAX_MARK        = "*";
    /** 得点の最大値以外に付けるマーク */
    public static final String  NON_MAX_MARK    = " ";

    /** 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 */
    public static int           counter;
    /** ランクを求める際に使用するひとつ前のランク格納用 */
    public static int           prev_rank       = PREV_RANK_INIT;


/**********************************************************************************************
 *   メインメソッド：main(String[] args)
 *
 *   メソッド名：   main
 *   概要：         プログラムエントリ
 *   引数：         String[] args ： コマンドライン引数
 *   返却値:        なし
 *   備考:          なし
 **********************************************************************************************/
    /**
     * メインメソッド
     * @param args
     */
    public static void main(String[] args) {
        /* データ格納用変数の初期化 */
        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(Student::getIsRetester).map(Student::getName).collect(Collectors.toList());
114
115         /* 試験成績順位を出力 */
116         if (students.stream().anyMatch(s -> !s.getIsRetester())) {
117             /* 見出し */
118             System.out.println(TITLE_1);
119
120             /* 合計得点の降順、氏名の昇順で並び替え */
121             students = students.stream().sorted(Comparator.comparing(Student::getSum, Comparator.reverseOrder()).thenComparing(Student::getName)).collect(Collectors.toList());
122
123             /* 順位を算出 */
124             for (int i = ZERO; i < students.size(); i++) {
125                 int rank = prev_rank == PREV_RANK_INIT || students.get(i).getSum() != students.get(i - PREV).getSum() ? i + ONE : prev_rank;
126                 students.get(i).rank = rank;
127                 prev_rank = rank;
128             }
129
130             /* 最高点を算出 */
131             int sum_max = students.stream().max(Comparator.comparingInt(Student::getSum)).get().getSum();
132             int [] scores_max = new int[SCORES_QUANTITY];
133             for (counter = ZERO; counter < SCORES_QUANTITY; counter++) {
134                 scores_max[counter] = students.stream().map(Student::getScores).mapToInt(s -> s[counter]).max().getAsInt();
135             }
136
137             /* インデント数の設定 */
138             int rank_len = String.valueOf(students.stream().max(Comparator.comparingInt(s -> s.rank)).get().rank).length();
139             int [] scores_len = Arrays.stream(scores_max).map(s -> String.valueOf(s).length()).toArray();
140
141             /* 氏名の最大長 */
142             int name_max_len = students.stream().max((s1, s2) -> s1.getName().length() - s2.getName().length()).get().getName().length();
143
144             /* 一覧 */
145             students.forEach(s -> {
146                 /* 合計得点の最大値にマークを付ける */
147                 String sum_max_mark = s.getSum() == sum_max ? MAX_MARK : NON_MAX_MARK;
148
149                 /* 氏名のインデント数の設定 */
150                 int name_len = name_max_len + name_max_len - s.getName().length();
151
152                 /* フォーマットして出力 */
153                 String result = String.format(FORMAT_1 + rank_len + FORMAT_2 + name_len + FORMAT_3, s.rank, sum_max_mark, s.getName());
154                 int [] scores = s.getScores();
155                 for (int j = ZERO; j < SCORES_QUANTITY; j++) {
156                     /* 各科目の最高得点にマークを付ける */
157                     String score_max_mark = scores[j] == scores_max[j] ? MAX_MARK : NON_MAX_MARK;
158
159                     result += String.format(FORMAT_4 + scores_len[j] + FORMAT_5, score_max_mark, scores[j]);
160                 }
161                 System.out.println(result);
162             });
163         }
164
165         /* 再試験者を出力 */
166         System.out.println(TITLE_2);
167         if (!retesters.isEmpty()) {
168             /* 該当者ありの場合 */
169             retesters.forEach(System.out::println);
170         } else {
171             /* 該当者なしの場合 */
172             System.out.println(TITLE_3);
173         }
174     }
175 }
```