岐 阜 工 業 高 等 専 門 学 校

# 情 報 工 学 実 験 報 告 書

実 験 番 号　　2

| | |
|---|---|
| 実 験 題 目 | キューとスタック |

| | |
|---|---|
| 報 告 書 提 出 者 | 班<br>学年　4　No.　38　　長尾颯真 |
| 共 測 者 氏 名 | *No.*<br>*No.*<br>*No.*<br>*No.*<br>*No.*<br>*No.* |

実 験 実 施 日　　2020 年　10 月　23 日

年　月　日

年　月　日

報 告 書 提 出 日　　2020 年　11 月　1 日

採点

| | |
|---|---|
| 体裁 | |
| 原理 | |
| 内容 | |
| 考察 | |
| 合計点 | |

教員のコメントページ（このページは空白にしてください）

# 1 実験目的・課題

次のデータを計算機上で表現し、

| Number | Name | Score0 | Score1 | Score2 | Score3 |
|---|---|---|---|---|---|
| 1 | Michael | 75 | 86 | 89 | 31 |
| 2 | Emma | 74 | 63 | 70 | 48 |
| 3 | Hannah | 73 | 45 | 82 | 31 |
| 4 | Emily | 40 | 30 | 49 | 48 |
| 5 | Daniel | 46 | 59 | 47 | 70 |

- ArrayList を使って名列番号順 (Number) に表示
- Stack を使って名列番号の逆順に表示
- HashMap を使って名前 (Name) の昇順に表示
- 名前と平均点を表示
- 平均点で昇順にソートして表示
- 名前が E で始まる学生だけを表示

を行う。

# 2 実装方法

## 2.1 方針

個人の Number,Name,Score0...3 をメンバにもつクラスを作成し、それを ArrayList でまとめる。それを引数として与え、各課題の処理を行う関数を作成する。

## 2.2 成績クラス

メンバ変数として Number, Name, Scores を持つクラスを作成する。それぞれの型は int, String, ArrayList<Integer> である。Scores を ArrayList にするのは今後成績が 4 つだけでなくそれより多く与えられる可能性も考えられるので可変長配列で受け取るのが今後そういった変更に対して対応しやすいためである。

この 3 つの変数の初期値はコンストラクタで与え、今後変更されることはないとする。

メンバ関数として、それぞれの変数の値を返す、Scores の平均値を計算する、自身のデータを出力する、以上の 3 種類のものを用意する。

## 2.3 solve 関数

### ArrayList を使って名列番号順 (Number) に表示

ArrayList< 成績クラス > を引数として受け取り、拡張 for 文で ArrayList の要素それぞれに対して表

示を行う。

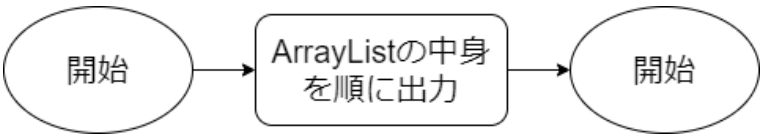


図 1　ArrayList を使って名列番号順に表示

### Stack を使って名列番号の逆順に表示

Stack< 成績クラス > を宣言してそれにデータを前から順にプッシュする。それを順に取り出すと Stack の構造上、プッシュしたときの逆順にデータを取り出すことが出来るので、その順に出力操作を行う。

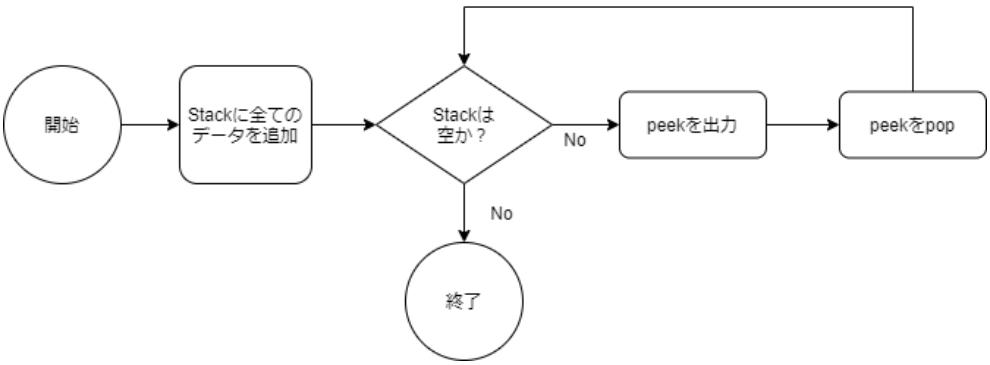


図 2　Stack を使って名列番号の逆順に表示

### HashMap を使って名前 (Name) の昇順に表示

String を key、成績クラスを value としてもつ HashMap を宣言し、データを拡張 for 文で順に HashMap に追加していく。このときに成績クラスのメンバ変数の値を取得する get_Name() を使用する。同様に Name を取得して ArrayList<String> に名前を追加する。この名前だけからなる ArrayList を昇順にソートし、それに対して前から名前を取得していき、この名前で HashMap にアクセスすると、その人の成績クラスが帰ってくるので、それぞれでデータの出力を行う。

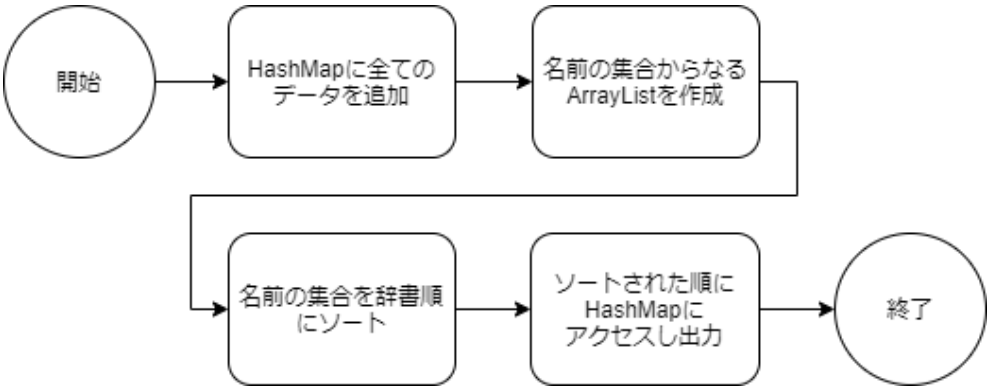


図 3　HashMap を使って名前 (Name) の昇順に表示

**名前と平均点を表示**

拡張 for 文でデータを舐め、それぞれで平均点を計算して、出力を行う。

平均点は成績のデータ数を $N$ として　$\frac{1}{N}\sum_{i=0}^{N-1} Scores_i$ で計算できる。

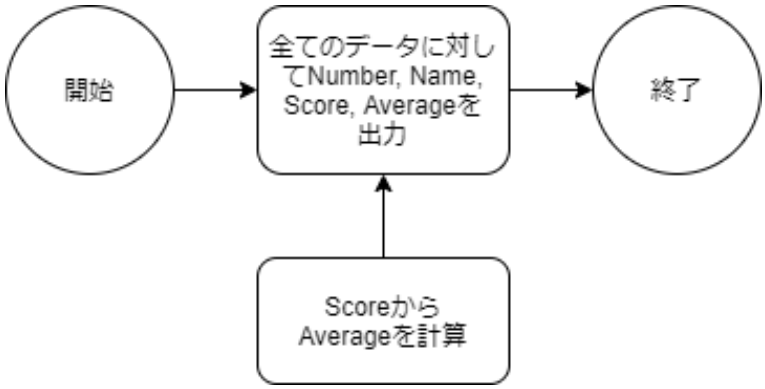


図 4　名前と平均点を表示

**平均点で昇順にソートして表示**

java.util.Collections を import することで Collections.sort() が使えるようになる。この関数は、データと比較クラスを引数として与えることで比較クラスに基づいてデータをソートをすることが出来る。今回は平均点の昇順にソートしたいので、成績クラス $S_1, S_2$ を引数にとり、それぞれで get_average() によって平均値を計算し、その値同士で比較をする comparator クラスを作成した。

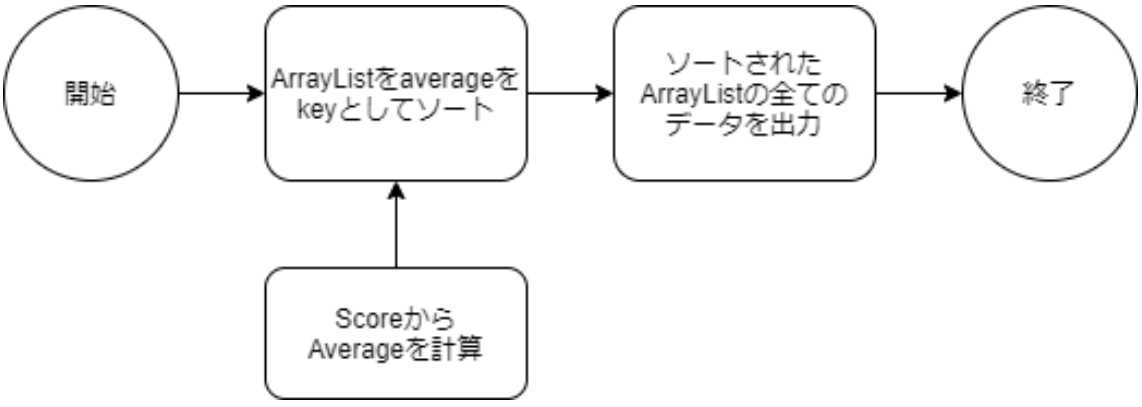


図 5　平均点で昇順にソートして表示

**名前が E で始まる学生だけを表示**

拡張 for 文でデータを舐め、それぞれで get_Name().charAt(0) で名前の初めの文字を取得し、それが'E' と等しければデータを出力する。

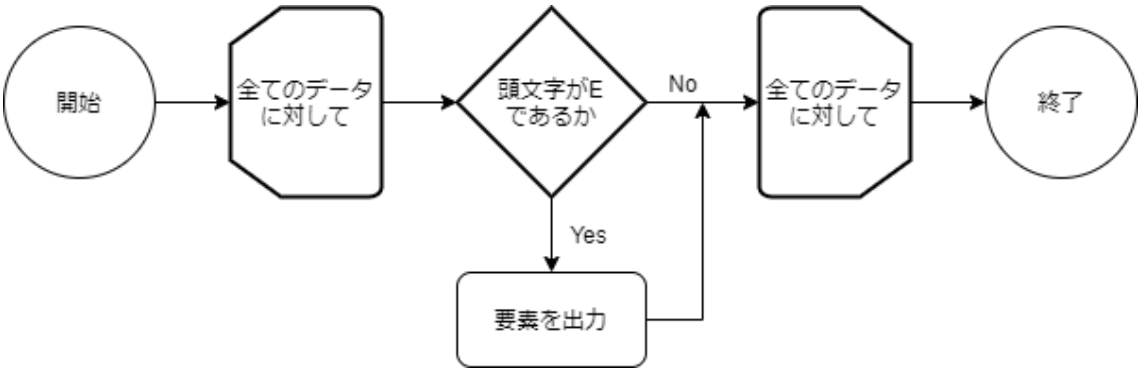


図 6　名前が E で始まる学生だけを表示

# 3　結果と考察

## 3.1　ArrayList を使って名列番号順 (Number) に表示

結果を以下に示す。

```
1 Michael 75,86,89,31,
2 Emma 74,63,70,48,
3 Hannah 73,45,82,31,
4 Emily 40,30,49,48,
5 Daniel 46,59,47,70,
```

出力の左端の名列番号をみると、1,2,3,4,5 の順になっていることからこの出力は正しいと考えらえる。

## 3.2　Stack を使って名列番号の逆順に表示

結果を以下に示す。

```
5 Daniel 46,59,47,70,
4 Emily 40,30,49,48,
3 Hannah 73,45,82,31,
2 Emma 74,63,70,48,
1 Michael 75,86,89,31,
```

出力の左端の名列番号をみると、5,4,3,2,1 の順、つまり先の逆順になっていることからこの出力は正しいと考えらえる。

## 3.3　HashMap を使って名前 (Name) の昇順に表示

結果を以下に示す。

5 Daniel 46,59,47,70,
4 Emily 40,30,49,48,
2 Emma 74,63,70,48,
3 Hannah 73,45,82,31,
1 Michael 75,86,89,31,

学生を辞書順で並べると Daniel < Emily < Emma < Hannah < Michael であり、出力と比べると一致しているため、この出力は正しいと言える。

### 3.4 名前と平均点を表示

結果を以下に示す。

1 Michael 75,86,89,31, 70
2 Emma 74,63,70,48, 63
3 Hannah 73,45,82,31, 57
4 Emily 40,30,49,48, 41
5 Daniel 46,59,47,70, 55

実際に Michael の平均点を計算すると、$\frac{75+86+89+31}{4} = \frac{281}{4} = 70.25$ と結果と一番右端の数字が一致しているため、この出力は正しいと考えられる。しかし、Michael の平均点は 70.25 だが、表示されている点数は 70 点になっている。これは、平均点を int で計算しているため、割り算をするときに小数点以下が切り捨てされるためである。

今回の実装では点数の和を先に計算して、最後に N で割っているため小数点以下の切り捨てが一度しか起こらないため誤差は 1 未満になる。もし、$\sum_{i=0}^{N-1} \frac{Scores_i}{N}$ のように計算すると、誤差の量は $\sum_{i=0}^{N-1} \frac{Scores_i \% N}{N}$ のように計算出来て、最悪 N-1 点の誤差が生まれることになる。

### 3.5 平均点で昇順にソートして表示

結果を以下に示す。

4 Emily 40,30,49,48, 41
5 Daniel 46,59,47,70, 55
3 Hannah 73,45,82,31, 57
2 Emma 74,63,70,48, 63
1 Michael 75,86,89,31, 70

平均点を上から見ていくと小さい順に並んでいることがわかる。つまりこの出力は正しいと考えられる。

この平均点のソートには学生の数を N、スコアの数を M として、O(MNlogN) かかっている。ソートは O(NlogN) だが今回は比較のたびに O(M) かけて点数の平均値を計算しているためである。例えばコンストラクタでスコアを受け取った時に平均値を計算しておくなどすれば平均値の取得が O(1) でできるようにな

り、O(NlogN) で処理を完了できる。ここからさらにオーダーを改善することもでき、テストの平均値としてありうる値の範囲を [0,A] の整数値とするとバケットソートで O(N+A) で計算できる。

### 3.6 名前が E で始まる学生だけを表示

結果を以下に示す。

```
4 Emily 40,30,49,48,
2 Emma 74,63,70,48,
```

Daniel,Hannah,Michael は名前が E で始まる学生ではないので表示されないのが正しい。つまりこの出力も正しいと考えられる。

ここで、学籍番号が 4,2 の順番で表示されている。最初に宣言したときは 1,2,3,4,5 の順番であり、main 関数内ではデータを変更していないのでこうなるのは不自然である。これは、平均値でソートしたときにデータが変更されたのが原因であると考えられる。それは、java には値渡ししか存在しないがプリミティブ型以外の型ではほかの言語でいうところの参照渡しと似た挙動をするためである。

## 参考文献


[1] ArrayList 要素のソートと Comparator
https://java.keicode.com/lib/collections-sort.php

[2] もう参照渡しとは言わせない
https://qiita.com/mdstoy/items/2ef4ada6f88341466783