今週の一コマ画像

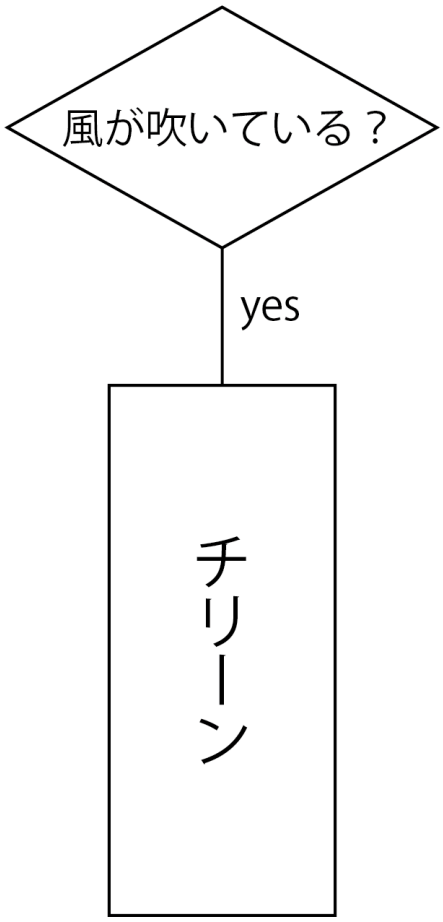

図0…ふふっとなったフローチャート “風鈴”

APJ04

学籍番号: 744366

氏名: 岸 典樹

2020年10月20日

第1章 フィールド群とメソッド群がクラスに内包されてる理由

本章ではフィールド群とメソッド群がクラスに内包されている理由について解題する。

1.1 クラス内に、あるとき〜

フィールド群やクラス群クラスに包含されているときは、それがどのクラスのメソッドで、どのクラスのフィールドなのか分かることができる。そして、同一クラスに包含されているそれ同士は、Javaにおいてthis句で表現している。

1.2 ないとき〜

フィールドやメソッドがクラス内にない時、それはグローバル変数や関数(ルーチン)として扱われるものだと考える。これが、C言語の構造体の中で定義された変数となれば話は変わってくるが、最上位階層に変数やルーチンを宣言してしまうと同一クラス(同一ファイル)の変数か、別クラスの変数か、直接参照か、間接参照か、値をどの程度記憶するかなどの意味合いが大きく変わってくるため、全く別のものになると考える。

1.3 では、なぜ内包するのか

オブジェクト指向はデータやアルゴリズムに構造を持ち込むことでカプセル化したオブジェクトを登場させ、そのオブジェクト同士でメッセージングを行う手法であり、その要素としてグローバル変数やメソッドが存在する。グローバル変数はJavaではフィールドと呼ばれており、オブジェクト指向を提唱しているJavaは概念上これを取り入れる必要がある。これらの概念が崩れてしまうと、複数のオブジェクトを呼び出す際に、同一名のフィールドやメソッドが存在する場合、強豪が起こる。これを防ぐために、カプセル化したものをインスタンス化し、メッセージングで操作することが求められているのだと考える。

第2章 配列リストと連結リスト

本章ではプリミティブ型の基本データとラッパークラスのオブジェクトについて、配列リストと連結リストの場合でどのようなメモリ構造をしているのかまとめる。

2.1 プリミティブ型 × 配列リスト

図2.1.1のプログラムに基づくメモリ構造を図2.1.2に記す。

```
int [] arrA = new int[3];
arrA[0] = 1000;
arrA[1] = 2000;
arrA[2] = 3000;
```

図2.1.1 プリミティブ型の配列リストのプログラム

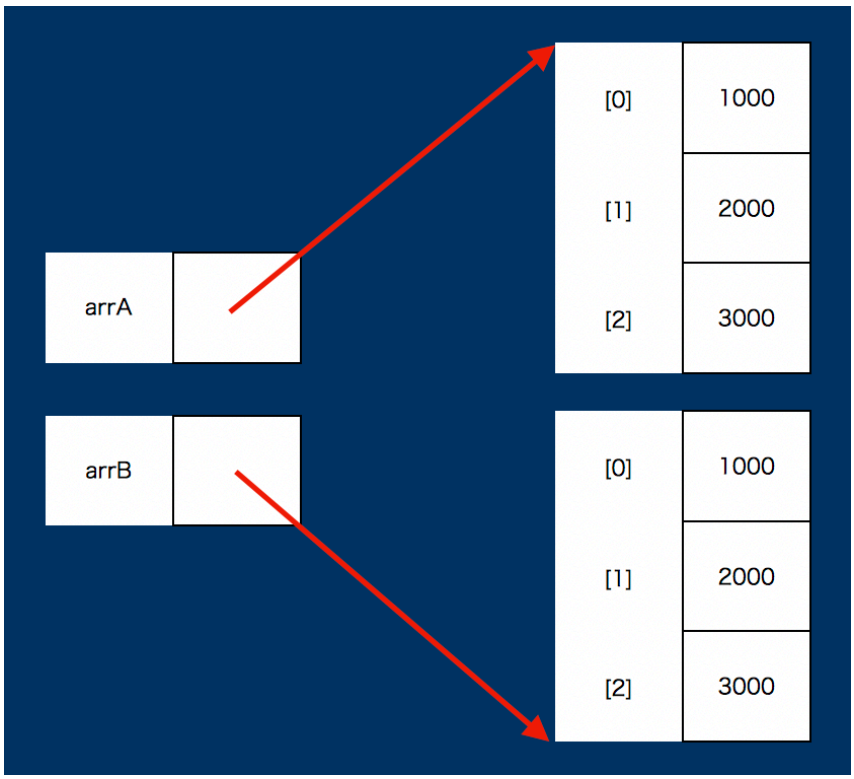

図2.1.2 プリミティブ型の配列リストのメモリ構造

2.2 プリミティブ型 × 連結リスト

図2.1.1のプログラムに基づくメモリ構造を図2.1.2に記す。

```
// 前提プログラム
public class LinkedObject {
      int data;
      LinkedObject next;
}

public class LinkedTerminal {
      LinkedObject first;
      LinkedObject last;
}

// 呼び出し
public void add(LinkTerminal linkTerminal, int item) {
      for(i = 1000; i < 4000; i += 1000) {
            LinkObject addLink = new LinkObject();
            addLink.data = item;
            if( linkedTerminal.first == null) {
                  linkedTerminal.first = addLink;
                  linkedTerminal.last = addLink;
            } else {
                  linkedTerminal.last.next = addLink;
                  linkedTerminal.last = addLink;
            }
      }
}
```

図2.2.1 プリミティブ型の連結リストのプログラム

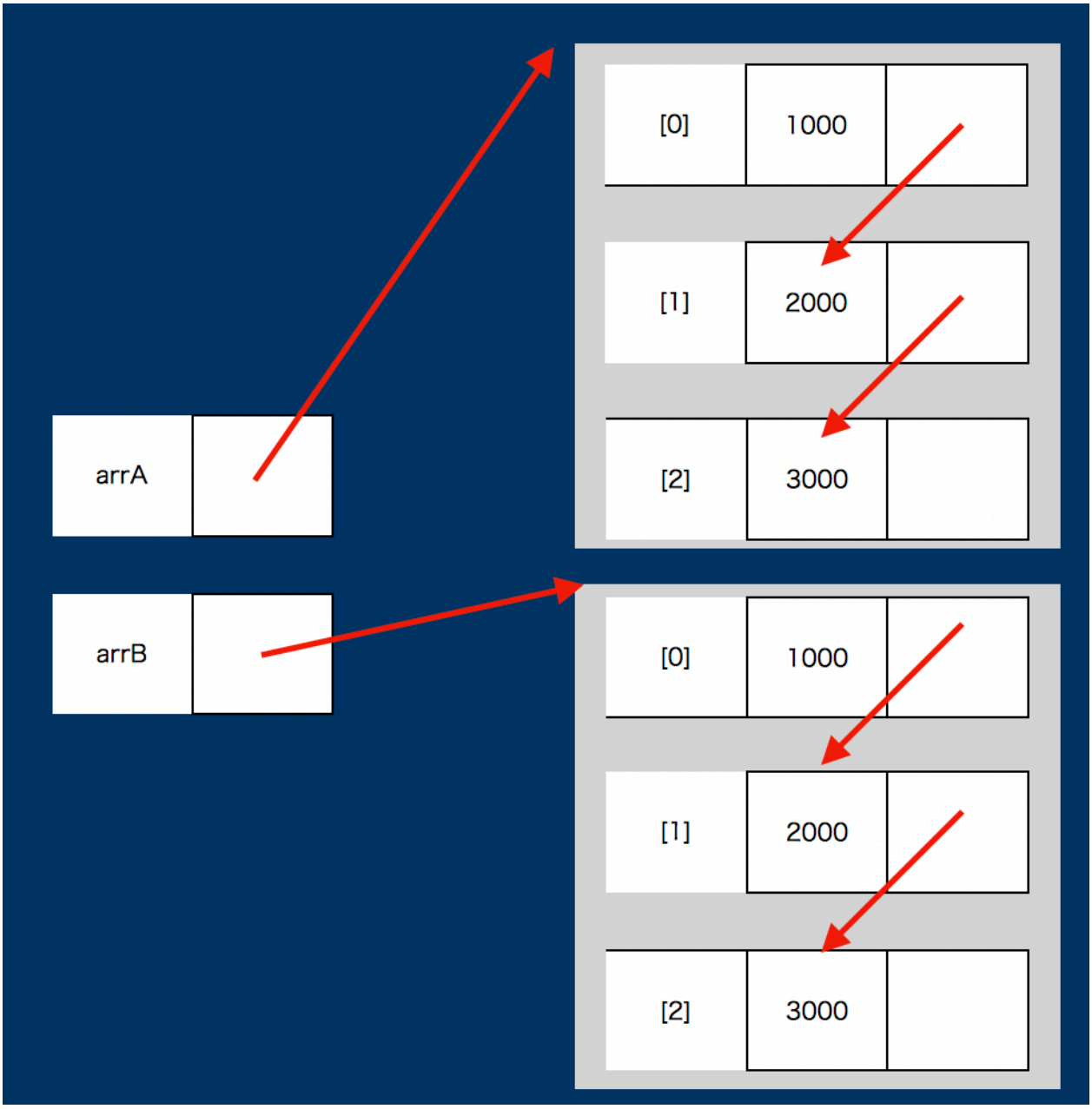

図2.2.2 プリミティブ型の連結リストのメモリ構造

2.3 ラッパークラス × 配列リスト

図2.1.1のプログラムに基づくメモリ構造を図2.1.2に記す。

```
Integer [] arrA = new Integer[3];
arrA[0] = 1000;
arrA[1] = 2000;
arrA[2] = 3000;
```

図2.3.1 ラッパークラスの配列リストのプログラム

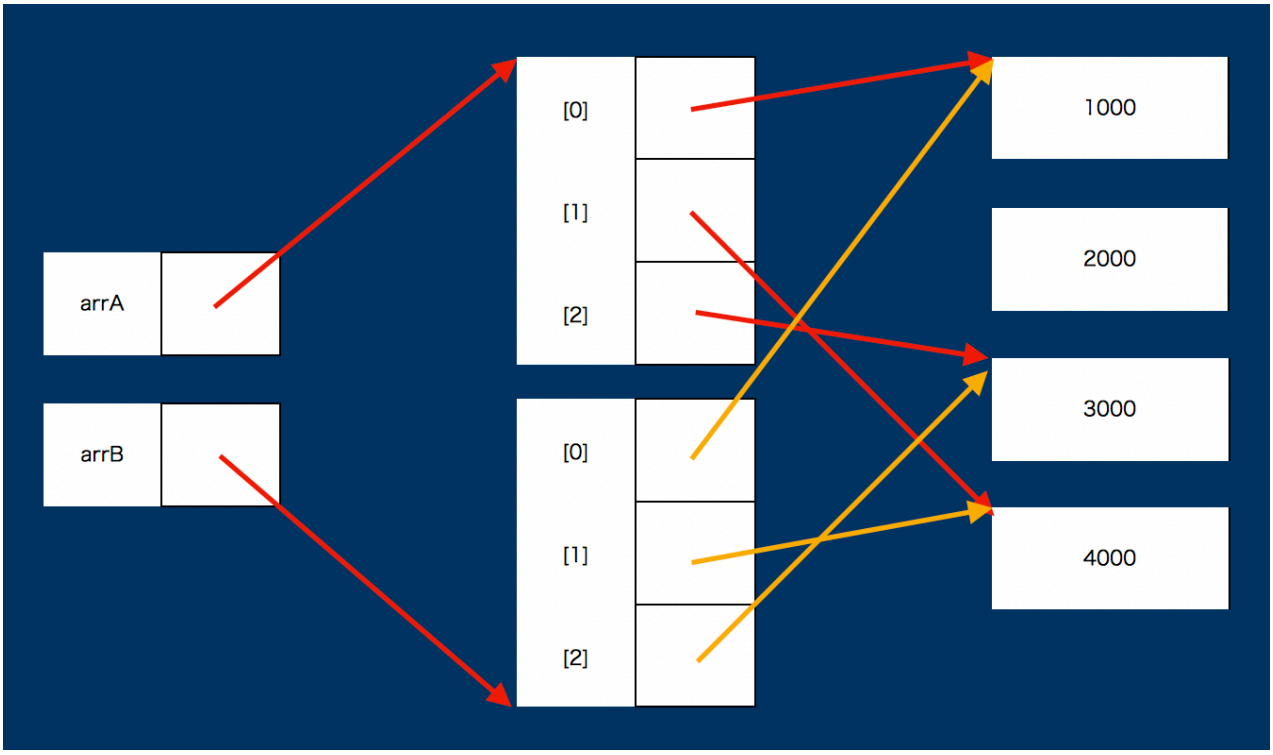

図2.3.2 ラッパークラスの配列リストのメモリ構造

2.4 ラッパークラス × 連結リスト

図2.1.1のプログラムに基づくメモリ構造を図2.1.2に記す。

```
// 前提プログラム
public class LinkedObject {
      Integer data;
      LinkedObject next;
}

public class LinkedTerminal {
      LinkedObject first;
      LinkedObject last;
}

// 呼び出し
public void add(LinkTerminal linkTerminal, Integer item) {
      for(i = 1000; i < 4000; i += 1000) {
            LinkObject addLink = new LinkObject();
            addLink.data = item;
            if( linkedTerminal.first == null) {
                  linkedTerminal.first = addLink;
                  linkedTerminal.last = addLink;
            } else {
                  linkedTerminal.last.next = addLink;
                  linkedTerminal.last = addLink;
            }
      }
}
```

図2.4.1 ラッパークラスの連結リストのプログラム

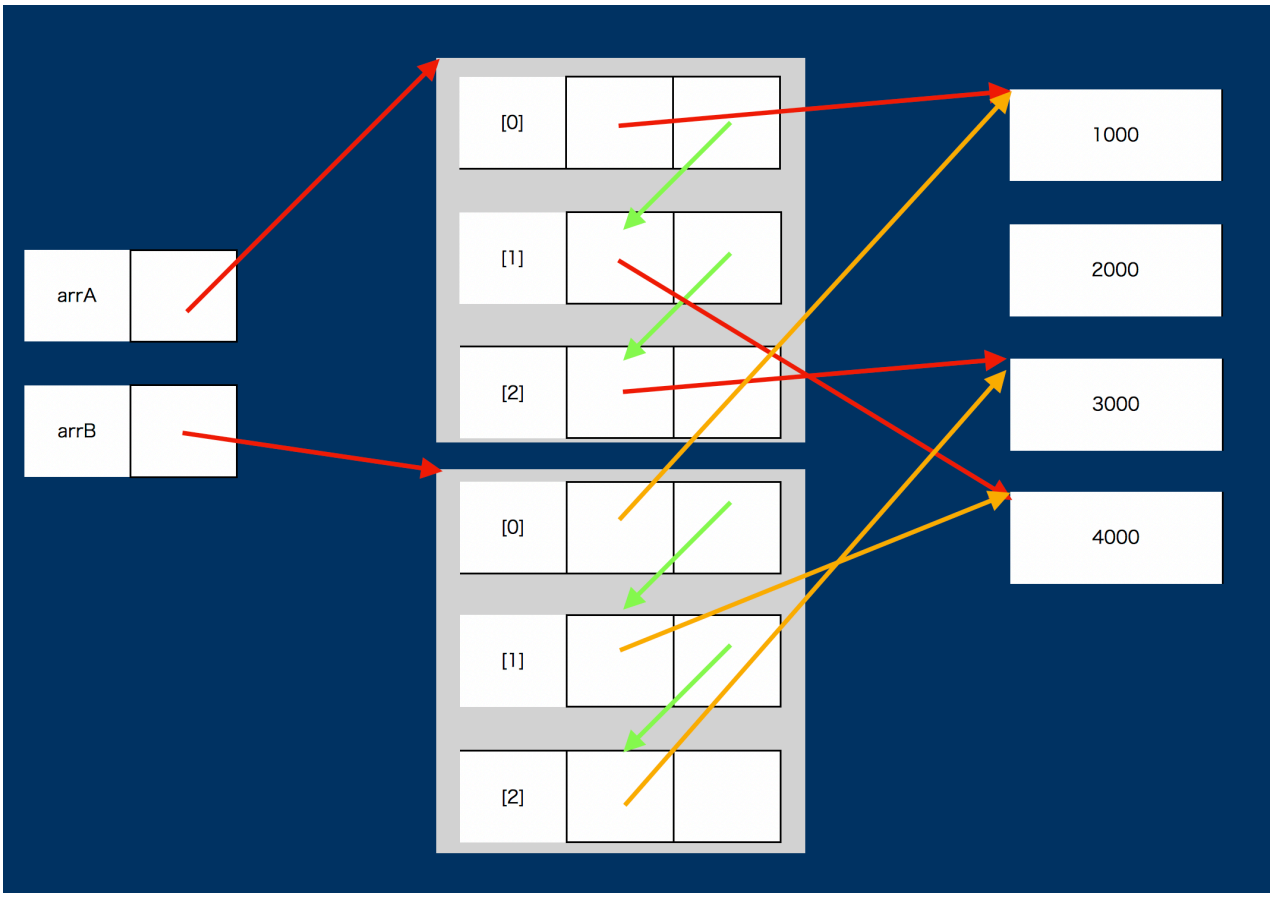

図2.4.2 ラッパークラスの連結リストのメモリ構造

第3章 自由記述

3.1 面接が終わりました。(The end)

先ほど、3次面接が終わりました。泣いても笑っても結果は結果です。とりあえず大きな失敗をしたので、お酒をたくさんのみながら溜まった課題に着手しております。今夜は夜泣きです。1週間、いくつかの授業をサボってまで面接準備をしていたので、溜まってる課題も、多いです。面接についての詳細は特別研究の進捗報告の際に、他の学生に伝えるような形でお話しできればと思います。本レポートは取り急ぎになってしまい申し訳ありませんが、これからはもう少しレポートに割く時間が戻ってきそうです。(10/20 20:59)

3.2 斜視を指摘されました。。

父に、急性斜視を指摘されました。毎日13〜15時間ほどパソコンに向き合って、凝視してるせいでしょうか。コロナ禍で外出せず、遠くを見ないことも大きな理由だと考えています。治したいですが、意識すると少し怖くなってしまいます。

3.3 ちょっとした質問

授業ないで内でキャメルケースの使用を促されてしましたが、これはJavaやJS等のキャメルケースが推奨される言語に限っての話でしょうか。PythonやPHPのような、標準関数がスネークケースで書かれている言語に関してはどのように書くべきか、特に、Pythonについては書き方がスネークケースとキャメルケースで分かれているようなので、どちらを信頼すれば良いか、揺れています。