

プログラミング演習2 課題 補足資料

第11週：整列（ソート）（2）

課題11-1～11-3の注意事項

- 「準備」に
「第10週に引き続き, String クラスの配列を対象として…」
とあるように、課題11-1～11-3では、
第10週で作成したStringSortクラスにメソッドを追加していく
 - いずれもstaticなメソッドである(第10週同様)

練習課題11-1

クイックソートの準備段階: partitionメソッド

- 第9週講義資料(p.15)の疑似コード(=教科書 p.317 List14.1)の「適当な要素a[v]を枢軸として, ... a[r]に集める(List14.1の(2))」を行うメソッドである
 - pp.19-20に処理の流れが、実装の例は p.21 に示されている
 - p.21の例は、intの配列が整列対象
課題で整列したいのは、**Stringクラスの配列**
 - 大小関係の比較で**compareToメソッド**を適切に利用すること
- インクリメント／デクリメント演算子の前置型(++j)と後置型(j++)の違いについては、次ページも参照のこと

復習： インクリメント／デクリメント演算子

- プロ言教科書 4-7-2
- 前置型と後置型の違いをよく理解すること

- `i=0;`
- `System.out.println(i++);`
- `System.out.println(i);`



出力結果

0
1

- `i=0;`
- `System.out.println(++i);`
- `System.out.println(i);`



1
1

- その文を実行してから、値を増加(減少)するか、
値を増加(減少)してから、その文を実行するか が異なる

練習課題11-1

動作確認プログラム

- partitionメソッドはprivateなので、partitionメソッド単独での動作確認するには、TestEx1103.javaを参考に、StringSortクラス内のmainメソッド作成するのが簡単
- TestEx1103クラスのような別クラスからはアクセスできない

```
public static void main(String[] arg){
    String[] array = new String[10];

    array[0] = "Orange" ;
    : 途中省略
    array[9] = "Grapefruit" ;

    int idx = partition(array, 引数1, 引数2);

    System.out.println("分割後枢軸のidx:"+idx);
    for (int i = 0 ; i < array.length ; i++) {
        System.out.println "[" + i + "]: " + array[i] ;
    }
}
```

出力結果

分割後枢軸のidx:4

[0]:Cherry
[1]:Banana
[2]:Apple
[3]:Grape
[4]:Grapefruit
[5]:Peach
[6]:Strawberry
[7]:Orange
[8]:Mango
[9]:Pineapple

p.21のプログラム例では、partition前の右端(array[9])を枢軸としている

枢軸を基準に分割なので、Array[0]-[3]にはGrapefruitより**小さい値**がarray[5]-[9]にはGrapefruitより**大きい値**が格納されていればよい

練習課題11-2

クイックソートの準備段階: recQuickSortメソッド

- 第9週講義資料(p.15)の疑似コード(=教科書 p.317 List14.1)の「左の部分を整列」、「右の部分を整列」を行うメソッドである
 - 実装の例は p.22 のquickSortメソッドとして示されている
 - メソッド名が違っていることに注意
 - p.22の例では、int型の配列を対象としていることや、変数名等が異なっているため、適切に修正すること

必須課題11-3

クイックソート: quickSortメソッド

- 第9週講義資料(p.15)の疑似コード(=教科書 p.317 List14.1)のメソッド全体
 - 実装の例は p.22 の`sort`メソッドとして示されている
 - メソッド名が違っていることに注意
 - p.22の例では、`int`型の配列を対象としていることや、変数名等が異なっているため、適切に修正すること
- 動作検証には、`TestEx1103.java`を利用すること
 - 練習課題11-2の時点では、動作検証しにくいため、必須課題11-3まで含めた形で動作検証する

練習課題11-4

コマンドライン引数とテキストファイルの読み込み

- コマンドライン引数については、
プロ演1 第14週課題やプロ言教科書 10-1-1 (p.198)などを参考
 - 今回の課題では、課題に書かれている通り
% java Ex1104 testfile.txt
と実行すると、testfile.txtという文字列が
args[0]に代入されることを理解していればよい
- テキストファイルの読み込みについては、
プロ言教科書「10-2 ファイルからのスキャン」(pp.201-202)を参考に
 - FileReaderクラスとScannerクラスを利用する方法が紹介されている
 - Scannerクラスの代わりに、
BufferedReaderクラス (p.203)を利用する方法もある
- 動作検証用には、rits.txtを使うと良い(動作を把握しやすい)

練習課題11-5

文字列を単語に分割 : StringTokenizerクラス

- 練習課題11-4で、テキストファイルを1行ごとに読み込み、表示ができているはず
- 1行ごとに読み込まれた文字列を空白で分割して表示するように、StringTokenizerクラスのプログラム例を参考にしながら、練習課題11-4のプログラムを変更する
- 例えば、右図のように単語ごとに出力できていればOK

The Kinugasa Campus is located in
northwestern Kyoto , a 1200 year old
word : The
word : Kinugasa
word : Campus
word : is
word : located
word : in
word : northwestern
word : Kyoto

必須課題11-6

ファイル中の単語をクイックソート

- 練習課題11-5では、テキストファイル中の単語を分割して表示できている
- 処理の流れは、
 - まず、ファイル中の単語全てを配列に格納
 - その後、必須課題11-3で作成したquickSortメソッドを利用し、単語を格納した配列に対し整列を行う
- 処理時間の計測にはSystemクラスのcurrentTimeMillisメソッドを利用する
 - 1970年1月1日0時を起点とした経過時刻(ミリ秒)が返される
 - quickSortメソッドの前後の時刻を取得し、差分を計算すれば、整列に要する時間を計測できる