プログラミング応用第３回

スタックとキュー

今日の課題：　スタック、キューのデータ構造の演習を行います.

１．スタック（Stack）

スタックとは、トランプゲームで用いる用語で，カードを積み重ねたものをいいます。スタックの特徴は以下の通りです.

- 新しいカードは、スタックの一番上(トップ)に置く。
- カードを取るときには、スタックのトップのカードを取る.
- カードを取ると、次の二番目のカードが現れる。

この仕組みをコンピュータで実現したものを「スタック」といいます.

- 新しいカードを、スタックのトップに記録する操作を　**push**　と呼ぶ.
- トップのデータを取る操作を **pop** という．pop を行うと，得られたデータはスタックから削除されます.

図１にその仕組みを示します．スタックは、最後に入ってきたものが最初に出て行くので、Last-In, First-Out とも言います。スタックは、計算機では，大変よく用いられるデータ構造です.

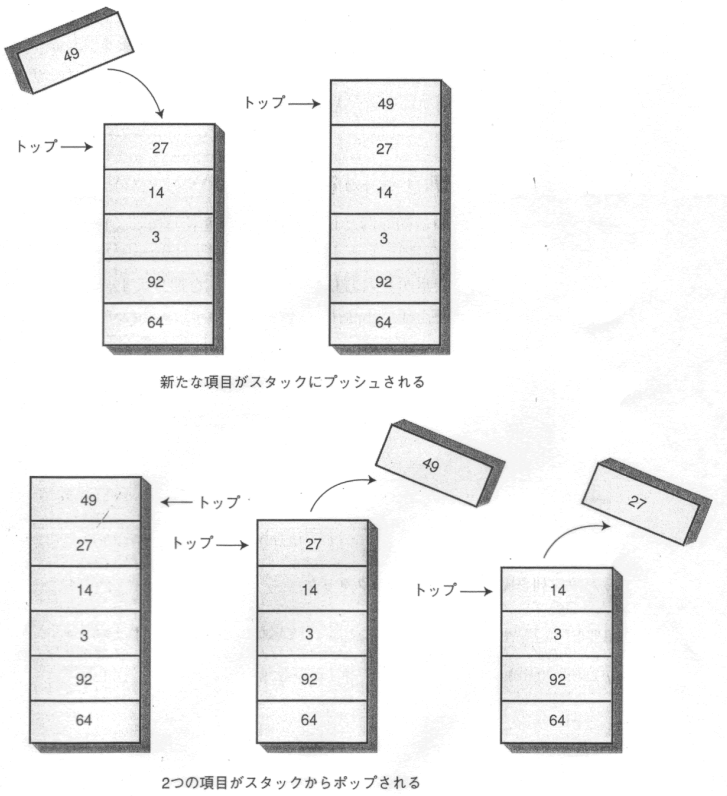


図１．スタック

２．キュー（Queue）

キューとは，待ち行列のことです．以下の性質があります．

- 新しく来た人は最後尾(Rear)に並びます．
- 最初に並んだ人からサービスを受けます。
- サービスを受けたら，キューからいなくなります．

データ構造では，最初に入ってきたものが最初に出て行く仕組み（First-In, First-Out）をキュー（Queue）といいます。図２にそのイメージを示します。

- 新しく追加されたデータは、キューの最後尾に置かれます。この操作を **insert** と呼びます。
- 先頭(Front)から取り出します。取るとなくなるので，この操作を **remove** と呼びます．

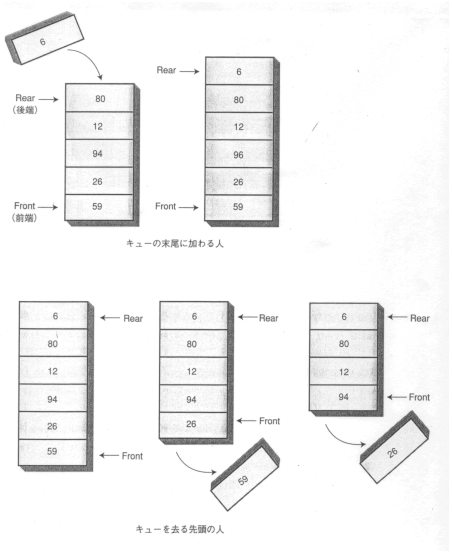


図２．キュー

課題：問題１と問題２のプログラムを作成しなさい．

ITC-LMS（https://itc-lms.ecc.u-tokyo.ac.jp/）にて提出してください．

- **提出先: ITC-LMS の「プログラミング応用 C」のページ**
- **課題 →第 3 回「スタックとキュー」に作成したプログラムを アップロードする**
- **締切: 6/28（水） 14:55 まで**

問題１．スタックとキュー

[1] 配列を用いてスタックを実現したプログラムを Stack.java というファイルに作成します．入力されるデータは文字列です。

(1) Stack.java に、スタックのクラス Stack を作ります。

```
class Stack {
  private int capacity;        // スタック用配列のサイズ
  private int size;            // 使用した配列の数
  private String [] dataSet;   // スタックのデータ項目
```

(2) Stack のコンストラクタおよびメソッドを作ります。

public Stack (), public Stack (int initSize)

スタック用の配列の初期サイズで作成する．dataSet の初期サイズは引数なしで 16，ありで initSize とする．(デフォルトでは initSize=4)

次に、以下の５つのメソッドを作成します。

- public boolean isEmpty() → スタックが空なら true を返す。

- public void push(String s) → 配列が満杯でなければスタックのトップに s を入れる。すでに配列が満杯なら配列を大きくしてから値を格納する.
- public String pop() → スタックが空でなければスタックのトップから項目を取る。空なら, “” を返す。
- public String peek() → スタックが空でなければスタックのトップの値を返す(取り除かない)。空なら “” を返す。
- private void doubleCapacity () → 配列サイズを 2 倍にする. 配列は以下のように管理します.
  - 初期配列の大きさは, コンストラクタの引数で決まります. たとえば, new Stack (16) とした場合, capacity = 16 となり, 16 個の配列が dataSet に確保されます.
  - 配列をすべて使い切ったときには,capacity の値を 2 倍にして, dataSet の配列サイズを 2 倍します. その際, 元の dataSet の内容を新しく確保した配列にコピーします.
  - ここでは, 配列サイズを大きくしたとき, 確認のために以下のようなメッセージをプリントするようにします.
    Capacity: 16 -> 32

(3) StackMain.java の main 関数が、次の入力に対して適切に処理し、push, pop, peek, quit が正しく動作するように StackMain.java および Stack.java を作成しなさい。ITC-LMS にプログラムの雛形を用意してあるので、ダウンロードして使ってください。

- **push string** [*Return*]
  string がスタックに格納されます。
- **pop** [*Return*]
  値を一つ pop してそれをプリントします。ただしスタックが空ならそのことを知らせる文をプリントします。
- **peek** [*Return*]
  直前に push された値をプリントします. 直前のデータが存在しなければ, そのことを知らせる文をプリントします.
- **quit** [*Return*]
  プログラムを終了します。

[2] 配列を用いてキューを実現したプログラムを Queue.java というファイルに作成します. 入力されるデータは文字列です。(注：insert は Stack の push と同じ機能. remove は pop と同じ機能. push/pop は Stack 特有の用語なので, ここでは言い方を変えている.)
クラスの構成は, 下記のようにします.

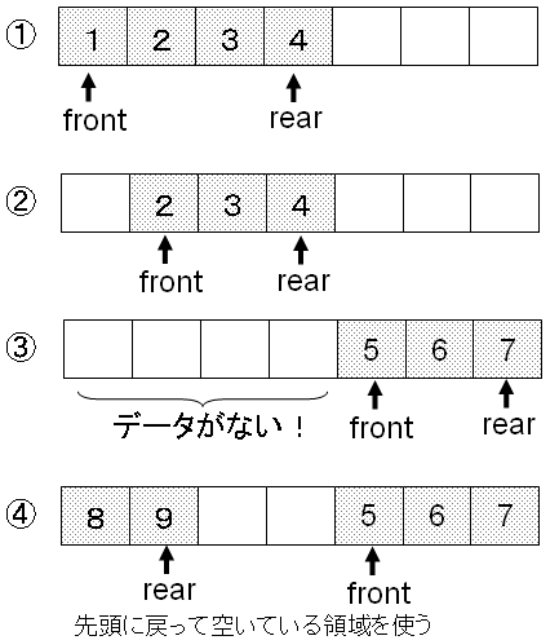
```
class Queue
{
    private int capacity;    // 使用可能な配列の大きさ
    private int size;        // 使用した配列の個数
    private int front;       // 先頭の位置 (0 ～ size-1)
    private int rear;        // 末尾の位置 (0 ～ size-1)
    private String [] dataSet;  // 配列
    ...
```

(1) メソッドとして，insert()、remove()、peek()、isEmpty()、doubleCapacity()の５つを作成しなさい。ただし，以下に留意すること．

- Queue では，最初に入力されたデータが最初に出て行くので先頭位置が移動する．そこで，先頭と最後尾の位置を格納するために，front と rear という変数を用意する。
- 次ページの図では，最初，１～４までを insert している（①）．この状態で remove すると先頭の「１」が除去されて，front の位置がずれる（②）．
- 次に，５～７を insert し，２～4 を remove する（③）．既に配列の最後に来ているので，これ以上 rear を後ろにもっていくことはできないが，先頭にはデータの空きが発生している．
- そこで，配列の最後まで行った場合には，先頭の空き領域にデータを入れていく（④）．この場合，front ＞ rear となる．
- もし，すべての配列を使い切った場合には，配列の capacity を 2 倍にする．元のデータをコピーする際は，front 位置が配列の先頭に来るように並べなおす．

(2)スタック同様に QueueMain.java の main 関数が、次の入力に対して適切に処理し、insert, remove, peek, quit が正しく動作するように QueueMain.java および Queue.java を作成しなさい。

- **insert string** ［*Return*］
  string をキューに入力する．
- **remove** ［*Return*］
  先頭を取り除き，その文字列をプリントする．
- **peek** ［*Return*］
  先頭の文字列をプリントする．
- **quit** ［*Return*］
  終了。



問題２．括弧付きの式を計算する電卓プログラム

四則演算からなる算術式を文字列として読み込み、計算して結果を表示するクラス Calculator.java とその main 関数を実装した Calculator.Main.java を作成しなさい．ただし、入力する算術式は，逆ポーランド記法(後置記法)とする．

※ 通常の式（中置記法）はスタックを用いて逆ポーランド記法に変換することができますが，時間が足りませんので，それは発展的課題とします（興味のある人はやってみてください).

```
class Calculator {
    //  単語が演算子ならば true を返す
    private boolean isOperator (String token)

    // 単語が数値ならば true を返す
    private boolean isNumber (String token)

    // 逆ポーランド記法の文字列を計算して答えを返す
    private double  getAnswer (String equation)

    // 逆ポーランド記法の文字列を通常の式（中置記法）にする.
    // これはオプションとする（省いてもよい）
    private String  getEquation (String equation)
}
```

実行クラス CalculatorMain

(ア)プログラムは，”７ ２ １ + - “ が入力されたとき，以下が出力されるようにします.（このとき入力は必ず半角スペースで区切ること）

7-(2+1) = 4　　または　　(7-(2+1)) = 4

(イ)なお，余裕のない人は，以下の出力でも OK です.

Answer = 4

問題２について，Calculator.java，CalculatorMain.java を提出してください.

【逆ポーランド記法(後置記法)」とは？】

算術式の記述法の一つで、演算子を値の後におきます。

たとえば A + B という式は A B + になります.

(3+4)*9/(7-(2+1)) の場合には，３ ４ + ９ * ７ ２ １ + - / となります。

逆ポーランド記法で算術式を記述した場合、括弧がなくても元の算術式の意図が失われることはありませんので、算術式を解析するアルゴリズムがシンプルなものになるという利点があります.

ヒント：

- スタックを使ってください.その際,Calculator.java　は,Stack.java と同じディレクトリ（フォルダ）でコンパイルしてください．これは，クラス Calculator の中で，問題１で作成した Stack を呼ぶためです.
- スタックに格納されるのは文字列なので，計算にあたっては，文字列と数字の相互変換が必要です．以下を参考にしてください.

（ア） 数字から文字列への変換

```
double value = 10;
String string = String.valueOf (value);
```

（イ）文字列から数字への変換

```
String string ( “3.14” );
double value = Double.valueOf(string).doubleValue;
```

提出方法

提出するものは以下の 6 つです．情報基盤センターの端末で動作を確認し、

指定時間内に itc-lms に提出できた人は試問を受けてください。

問題 1：Stack.java，StackMain.java，Queue.java，QueueMain.java

問題 2：Calculator.java，CalculatorMain.java

※問題 2 は答えを算出する getAnswer のみの実装で可とします．

発展的課題として getEquation，中置記法を逆ポーランド記法に変換する 2 つがありますが，採点には含めません．しかし，勉強になるので実装してみることをお勧めします．

以上