プログラミング演習2
課題　補足資料

第４週：待ち行列（キュー）

練習課題4-1
DoubleQueueクラス

- 大枠としては、DoubleStackクラスと似たような形となるが、配列をリングバッファとして取り扱う点に注意
（第３週講義資料pp.4-17）

- フィールド変数として
  - データを格納する配列変数 dataArray に加えて
  - 先頭と末尾の場所を指す変数 **front, rear** が必要（p.5)
  - キューの大きさを表す変数 maxSize も用意した方がよい
（dataArray.length=キューの大きさではないため）

- コンストラクタでは
  - データ格納領域のサイズは、**maxSize+1** 分確保する（p.17）
  - 変数 front, rear の初期化：空の時はどうすべきだったか？（p.15）
  - データの初期化はDoubleStackと同様に、例えば0.0を代入
（用途によって初期化の値は異なる）

練習課題4-2
isFull, isEmptyメソッド

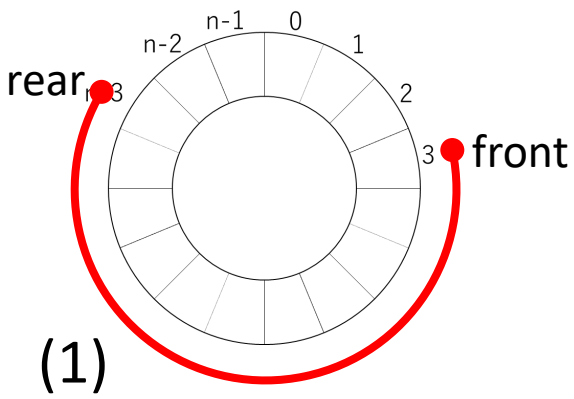
- 第3週講義資料 pp. 18, 20を参考に考える
- isFullメソッド:
  - p.18より、リアの一つ次にフロントがあればいっぱい
  - ただしラップアラウンドの処理が必要（＝剰余(%)を使う）
    - 例えばmaxSize = 15（dataArray.length = 16）の場合
      rear = 15, front = 0の時はいっぱいだが、条件式としてどう書けばよいか？
- isEmptyメソッド:
  - p.20より、フロントとリアが同じ場所にあれば空
  - こちらは、そのまま書けばよい

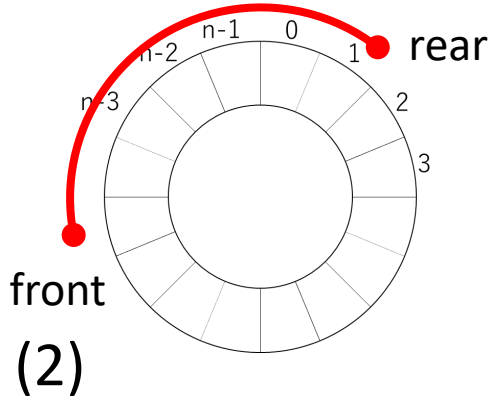
練習課題4-3
size, showメソッド

- sizeメソッド:
  - 例えば、rear=2, front=14の時、格納データ数は4
    rearとfrontから計算するにはどうしたらよいか？
  - 格納データ数を保存する変数を別途用意してもよい（より簡単）
- showメソッド:

  (1) rearとfrontの間に n-1, 0 を含まない場合は簡単

  (2) rearとfrontの間に n-1, 0 を含む場合には、どうしたらよいか？





必須課題4-4
enqueueメソッド

- 第3週講義資料 p. 19、教科書4.4.4を参考に考える
- 手順は、
  - 待ち行列がいっぱいでないかを確認
    → isFullメソッドを使えばよい
  - いっぱいでなければ、リアにデータを追加
    → リアが配列の終端の場合、データ追加後に逆端に回り込ませる
    （ラップアラウンドの処理）
  - スタックがいっぱいであれば、エラーメッセージを出力
    → 標準エラー出力：System.err.printlnを使う

必須課題4-4
動作確認プログラムの例

- 以下のようなプログラムで動作確認をすること

```
// DoubleQueueのインスタンスを生成(サイズは5で)
DoubleQueue testQueue = new DoubleQueue(5);

// 待ち行列にデータを5個追加
for(int i = 0; i < 5; i++){
  testQueue.enqueue(i * 0.5 + 1);
}

// 現在の待ち行列を表示
testQueue.show();
System.out.println("格納データ数 : " + testQueue.size());

// さらにデータを追加すると
testQueue.enqueue(5);
System.out.println("格納データ数 : " + testQueue.size());
```

以下のような
出力が得られればOK

```
1番目: Array[ 0 ] = 1.0
2番目: Array[ 1 ] = 1.5
3番目: Array[ 2 ] = 2.0
4番目: Array[ 3 ] = 2.5
5番目: Array[ 4 ] = 3.0
格納データ数 : 5
キューは一杯です
格納データ数 : 5
```

必須課題4-5
dequeueメソッド

- 第3週講義資料 p. 21、教科書4.4.4を参考に考える
- 手順は、
  - 待ち行列が空でないかを確認
    → isEmptyメソッドを使えばよい
  - 空でなければ、フロントからデータを取り出し
    → フロントが配列の終端の場合には、データの取得後に逆端に回り込ませる（ラップアラウンドの処理）
  - 空いた要素を初期化
  - 空であれば、エラーメッセージを出力
    → 標準エラー出力：System.err.printlnを使う

必須課題4-5
動作確認プログラムの例

- 以下のようなプログラムで動作確認をすること

```
DoubleQueue testQueue = new DoubleQueue(5);

for(int i = 0; i < 5; i++) {  // データを5個追加
  testQueue.enqueue(i * 0.5 + 1);
}
for(int i = 0; i < 2; i++) {  // データを2個取り出し
  testQueue.dequeue();
}
for(int i = 0; i < 2; i++) {  // データを2個追加
  testQueue.enqueue(i * 0.5 + 5);
}
testQueue.show();
for(int i = 0; i < 5; i++) {  // データを5個取り出し
  testQueue.dequeue();
}
// さらにデータを取り出すと？
testQueue.dequeue();
```

以下のような
出力が得られればOK

```
1番目: Array[ 2 ] = 2.0
2番目: Array[ 3 ] = 2.5
3番目: Array[ 4 ] = 3.0
4番目: Array[ 5 ] = 5.0
5番目: Array[ 0 ] = 5.5
格納データ数：5
キューは空です
```

size, showメソッドでの
ラップアラウンドの処理が
できているかどうかについても
きちんとチェックしておくこと

練習課題4-6
clearメソッド

- 待ち行列の中身をすべて消去するとは？
- 待ち行列の中身を初期化した上で、
  rear, frontの値をどうしたら良いか？
  （＝初期化時にどうしていたかを考える）

練習課題4-6
動作確認プログラムの例

- 以下のようなプログラムで動作確認をすること

```
// DoubleQueueのインスタンスを生成(サイズは5で)
DoubleQueue testQueue = new DoubleQueue(5);

// 待ち行列にデータを5個追加
for(int i = 0; i < 5; i++){
  testQueue.enqueue(i * 0.5 + 1);
}
// 現在の待ち行列を表示
testQueue.show();
System.out.println("格納データ数 : " + testQueue.size());

// データクリア
testQueue.clear();
System.out.println("格納データ数 : " + testQueue.size());
```

以下のような
出力が得られればOK

```
1番目: Array[ 0 ] = 1.0
2番目: Array[ 1 ] = 1.5
3番目: Array[ 2 ] = 2.0
4番目: Array[ 3 ] = 2.5
5番目: Array[ 4 ] = 3.0
格納データ数 : 5
格納データ数 : 0
```