プログラミング演習2
課題　補足資料

第３週：スタック

練習課題3-1
DoubleStackクラス

- データ構造とアルゴリズムの第2週講義資料 p. 17に、大枠は書かれている
- コンストラクタ内に、フィールド変数の初期化を追加すればよい
  - dataArrayの初期化では、例えば0.0を入力しておく（用途によって初期化の値は異なることに注意）
  - spの初期化では、どうすべきか？

練習課題3-2,3-3
isFull, isEmpty, size, showメソッド

- 第2週講義資料 pp. 19-20を参考に考える
- 配列 double[] dataArrayの要素数は、dataArray.length で取得できる

必須課題3-4
pushメソッド

- 第2週講義資料 p. 21を参考に考える
- 手順は、
  - スタック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、末尾にデータを追加
    → 教科書4.4.2節を参考に考える
  - 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3-4
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

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

// スタックにデータを5個追加
for(int i = 0; i < 5; i++){
  testStack.push(i * 0.5 + 1);
}

// 現在のスタックを表示
testStack.show();
System.out.println("格納データ数 : " + testStack.size());

// さらにスタックに追加すると？
testStack.push(5);
System.out.println("格納データ数 : " + testStack.size());
```

以下のような
出力が得られればOK

```
[ 0 ] : 1.0
[ 1 ] : 1.5
[ 2 ] : 2.0
[ 3 ] : 2.5
[ 4 ] : 3.0
格納データ数 : 5
スタックは一杯です
格納データ数 : 5
```

必須課題3-5
popメソッド

- 第2週講義資料 pp. 21-22を参考に考える
- 手順は、
  - スタック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、末尾からデータを取り出し
    → 教科書4.4.2節を参考に考える
  - 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3-5
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

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

// スタックにデータを5個追加
for(int i = 0; i < 5; i++){
  testStack.push(i * 0.5 + 1);
}
// 空になるまでポップする
int count = 0 ;
while (!testStack.isEmpty()) {
  System.out.println(count + " : " + testStack.pop() );
  count++ ;
}
// スタックが空でDouble.NaNが返れば正解
System.out.println("さらにpop : " + testStack.pop() );
```

以下のような
出力が得られればOK

```
0 : 3.0
1 : 2.5
2 : 2.0
3 : 1.5
4 : 1.0
スタックは空です
さらにpop : NaN
```

練習課題3-6
clearメソッド

- スタックの中身をすべて消去するとは？
  - 単にスタックの中身を初期化すれば良いわけではない
- 中身を初期化した上で、
  sp: スタックポインタ = 格納されているデータの個数
  をどうすべきか？

練習課題3-6
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

```
DoubleStack testStack = new DoubleStack(5);

// 適当にスタックにデータを5個追加
for(int i = 0; i < 5; i++){
   testStack.push(i * 0.5 + 1);
}

// 現在のスタックを表示
testStack.show();
System.out.println("格納データ数 : " + testStack.size());

// データクリア
testStack.clear();
System.out.println(“クリア後データ数 : " + testStack.size());
```

以下のような
出力が得られればOK

```
[ 0 ] : 1.0
[ 1 ] : 1.5
[ 2 ] : 2.0
[ 3 ] : 2.5
[ 4 ] : 3.0
格納データ数 : 5
クリア後データ数 : 0
```