Ex_07_02.java 解説

このプログラムは、4桁以下の正の整数に対して各桁の数を並べ替えてできる「最大の整数値」と「最小の整数値」の差を計算し、それを繰り返して収束するまで出力します。

```
import java.util.Arrays;
import java.util.Scanner;

public class Ex_07_02 {

    // 数値を受け取り、その数値に対する次の数値を計算するメソッド
    public static int nextValue(int n) {
        // 4桁の文字列に変換する
        char[] digits = String.format("%04d", n).toCharArra
y();

        // 最大値を計算
        Arrays.sort(digits);
        int minValue = Integer.parseInt(new String(digit
s));

        // 最小値を計算
        reverseArray(digits);
        int maxValue = Integer.parseInt(new String(digit
s));

        // 差を計算して次の値を返す
        return maxValue - minValue;
    }

    // 配列を反転するメソッド
    public static void reverseArray(char[] array) {
        int left = 0;
        int right = array.length - 1;
        while (left < right) {
            char temp = array[left];
```

```
            array[left] = array[right];
            array[right] = temp;
            left++;
            right--;
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // コマンドライン引数として4桁以下の正の整数を受け取る
        int n = Integer.parseInt(args[0]);

        // 計算結果を出力する
        while (true) {
            // 次の値を計算
            int next = nextValue(n);

            // 結果を表示
            System.out.println(next);

            // 同じ値が再度出現したら終了
            if (next == n) {
                break;
            }

            // 次のループのために現在の値を更新
            n = next;
        }

        scanner.close();
    }
}
```

プログラムの詳細説明

1. **プログラムの概要**:

- このプログラムは、4桁以下の整数に対して「最大の整数値」と「最小の整数値」の差を計算し、その結果を繰り返し出力します。最終的に収束する値（同じ値が2回連続して現れる）に達するまでこのプロセスを続けます。

2. `nextValue` **メソッド**:
   - このメソッドは、与えられた整数 `n` に対する「次の整数値」を計算します。
   - `n` を4桁の文字列に変換し、各桁を含む配列 `digits` に格納します。この際、4桁未満の数値は先頭に0を追加して処理します。
   - `Arrays.sort` を使用して `digits` 配列を昇順にソートし、それを整数として解釈して `minValue` とします。
   - 反転された `digits` 配列（降順）を使って `maxValue` を計算します。
   - 最後に、`maxValue` から `minValue` を引いた値を返します。

`sort` メソッドの詳細説明

`sort` メソッドは、Java標準ライブラリの `Arrays` クラスに定義されているメソッドです。このメソッドは配列を昇順にソートするために使用されます。

使用箇所

```
Arrays.sort(digits);
```

- このコードでは、文字配列 `digits` を昇順にソートしています。
- `Arrays.sort(digits)` は、`digits` 配列の各要素を小さい順に並べ替えます。例えば、`digits` に `['5', '1', '3', '8']` という値が格納されていた場合、これを `Arrays.sort(digits)` でソートすると、`['1', '3', '5', '8']` のように並べ替えられます。

動作の流れ

- `sort` メソッドは、`O(n log n)` の時間複雑度で動作するソートアルゴリズムを内部的に使用しています。通常は、`Timsort` というアルゴリズムが使われます。
- このソートにより、配列内の各要素が比較されて並べ替えられます。結果として、配列は昇順（小さい値から大きい値へ）に整列されます。

`reverseArray` メソッドについて

1. `reverseArray` **メソッド**:
   - このメソッドは、`digits` 配列を反転させるために使用されます。
   - 反転させることで、降順に並べ替えた状態にします。
2. `main` **メソッド**:
   - コマンドラインから4桁以下の整数を受け取り、これを元に計算を開始します。
   - `nextValue` メソッドを使用して、次の値を計算し、それを標準出力に表示します。
   - もし、計算結果が前回と同じであれば、計算は終了し、プログラムを終了します。

`reverseArray` メソッドの詳細説明

`reverseArray` メソッドは、配列内の要素を逆順に並べ替えるためのカスタムメソッドです。このメソッドを使用することで、降順に並べ替える操作を実現しています。

メソッドの定義

```
public static void reverseArray(char[] array) {
    int left = 0;
    int right = array.length - 1;
    while (left < right) {
        char temp = array[left];
        array[left] = array[right];
        array[right] = temp;
        left++;
        right--;
    }
}
```

動作の流れ

- `reverseArray` メソッドは、与えられた配列 `array` の要素を逆順にします。たとえば、`['1', '3', '5', '8']` という配列を `reverseArray` に渡すと、`['8', '5', '3',`

`'1']` という逆順の配列が得られます。

メソッドの詳細

1. **初期設定**:
   - `int left = 0;` と `int right = array.length - 1;` で、配列の先頭 (`left`) と末尾 (`right`) のインデックスを設定します。
2. **要素の入れ替え**:
   - `while (left < right)` ループを使用して、`left` と `right` の位置にある要素を入れ替えます。具体的には、次のような操作を行います：
     - `char temp = array[left];` で、`left` 位置の要素を一時的に `temp` に保存します。
     - `array[left] = array[right];` で、`right` 位置の要素を `left` 位置に移動します。
     - `array[right] = temp;` で、`temp` に保存しておいた元々の `left` 位置の要素を `right` 位置に移動します。
3. **インデックスの更新**:
   - `left++` と `right--` で、それぞれ `left` を右に、`right` を左に1つずつ移動させます。
4. **ループ終了**:
   - `left` が `right` に達するかそれを超えるまでこの処理を繰り返します。結果として、配列の要素が完全に逆順になります。

実行例

- **コマンドライン入力 `5138` の場合**:
  - プログラムは次のように出力します：

```
7173
6354
3087
8352
6174
6174
```

- **コマンドライン入力 `21` の場合**:
  - 出力結果は次のようになります：

```
2088
8532
6174
6174
```

- **コマンドライン入力 `5555` の場合**:
  - 出力結果は次のようになります：

```
0
0
```