Unicodeを扱う際の小細工色々覚書

2023-09-02 IDSの正規化とか方法が思いつかなかったのでパス
2023-08-28 最後といったな？あれは嘘だ
2023-08-26 一応これが最終形になります
2023-08-25 もう少し整理しようやと手直し
2023-08-24 結合文字まで入れたので公開

前提

Java11で実行。

仕事で色々やりそうな感じがあったので自分のリファレンス用で品質関係は二の次。
OSS等で公開されたライブラリは使わない（多分使わせてくれないだろうからなー）

用意したクラス

- UTF-16な文字列を1文字単位に分解する。サロゲートペア表現になる文字も対象。異字体セレクタを伴う文字も対象。
- 結合文字も扱うけど、平仮名・片仮名用に限定。全部なんてやってらんない□u+200D？知らない子ですね…
- 分解結果は文字で扱わずUnicodeのコードポイントで取得する。

```
List<Integer> codepoints = new ArrayList<Integer>();
Integer codepoint;
// String targetStr;
for(Integer idx = 0; idx < targetStr.length(); idx++) {
    codepoint = targetStr.codePointAt(idx);
    codepoints.add(codepoint);
    if (codepoint > 65535) idx++; // サロゲートペア上位コード位置なので下位コード位置に補正しておく
}
```

戻す場合はコードポイントから文字に戻す。

```
StringBuilder sb = new StringBuilder();
String targetStr;
// List<Integer> codepoints;
for(Integer idx = 0; idx < codepoints.size(); idx++) {
    sb.appendCodePoint(codepoints.get(idx)); // コードポイントから文字に戻す
}
targetStr = sb.toString();
```

例えば文字列 " 叱　　野家葛　 を与えたら、以下の形に分解する。

List要素	1文字のList要素	格納するコードポイント	説明
0	0	u+53f1	叱

List要素	1文字のList要素	格納するコードポイント	説明
1	0	u+20bb7	
2	0	u+91ce	野
3	0	u+5bb6	家
4	0	u+845b	葛
	1	u+e0100	

JSON風イメージだとこうかな。なおJSONだと16進数は使えないしコメントも書けない。

```
  "codepoints": [
      [ 0x53f1 ],         // 叱
      [ 0x20bb7 ],        //
      [ 0x91ce ],         // 野
      [ 0x5bb6 ],         // 家
      [ 0x845b, 0xe0100 ] // 葛
  ]
```

あと、結合文字が理解できないソフトウエアは結構危険で、Macで作ったファイルをWindowsに持ってきた時にこの問題にぶつかる事が多い。片仮名・平仮名の処理はやっておいた方が無難。

java.text.Normalizerつかえや？

片仮名・平仮名の正規化ならjava.text.Normalize使えるやろ？と言われるけど、それ以外がねー。

なんか弄ってほしくないものまで弄られちゃうんで、ちょっと使うのが難しいかな…

NFC

```
----------------------------------
Str        : □□¼□2□□□□□
StrLen     : 10
convedStr  : □□¼□2□□□□□
convedLen  : 10
Normalize  : NFC
Pos 0      : □ u+3232
Pos 1      : □ u+3231
Pos 2      : ¼ u+bc
Pos 3      : □ u+2460
Pos 4      : 2□ u+32, u+20e3
Pos 5      : □ u+24f7
Pos 6      : □ u+2779
Pos 7      : □ u+3224
Pos 8      : □ u+3285
----------------------------------
Str        : ﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ
StrLen     : 10
convedStr  : ﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ
convedLen  : 10
Normalize  : NFC
Pos 0      : ﾊ u+ff8a
Pos 1      : □ u+ff9f
Pos 2      : ﾋ u+ff8b
Pos 3      : □ u+ff9e
Pos 4      : ﾌ u+ff8c
Pos 5      : □ u+ff9f
Pos 6      : ﾍ u+ff8d
Pos 7      : □ u+ff9e
Pos 8      : ﾎ u+ff8e
Pos 9      : □ u+ff9f
----------------------------------
Str        : ぱ　び　ぷ　べ　ぽ
StrLen     : 10
convedStr  : ぱびぷべぽ
convedLen  : 5
Normalize  : NFC
Pos 0      : ぱ u+3071
Pos 1      : び u+3073
Pos 2      : ぷ u+3077
Pos 3      : べ u+3079
Pos 4      : ぽ u+307d
```

NFKC

```
----------------------------------
Str        : □□¼□2□□□□□
StrLen     : 10
convedStr  : (有)(株)1／412　③❹(五)六
convedLen  : 18
Normalize  : NFKC
Pos 0      : ( u+28
Pos 1      : 有 u+6709
Pos 2      : ) u+29
Pos 3      : ( u+28
Pos 4      : 株 u+682a
Pos 5      : ) u+29
Pos 6      : 1 u+31
Pos 7      : □ u+2044
Pos 8      : 4 u+34
Pos 9      : 1 u+31
Pos 10     : 2□ u+32, u+20e3
Pos 11     : □ u+24f7
Pos 12     : □ u+2779
Pos 13     : ( u+28
Pos 14     : 五 u+4e94
Pos 15     : ) u+29
Pos 16     : 六 u+516d
----------------------------------
Str        : ﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ
StrLen     : 10
convedStr  : パピプペポ
convedLen  : 5
Normalize  : NFKC
Pos 0      : パ u+30d1
Pos 1      : ピ u+30d3
Pos 2      : プ u+30d7
Pos 3      : ペ u+30d9
Pos 4      : ポ u+30dd
----------------------------------
Str        : ぱ　び　ぷ　べ　ぽ
StrLen     : 10
convedStr  : ぱびぷべぽ
convedLen  : 5
Normalize  : NFKC
Pos 0      : ぱ u+3071
Pos 1      : び u+3073
Pos 2      : ぷ u+3077
Pos 3      : べ u+3079
Pos 4      : ぽ u+307d
```

Normalizer.Form.NFC, Normalizer.Form.NFD, Normalizer.Form.NFKC, Normalizer.Form.NFKD とあるので、それぞれ自分で試してみるといいと思うよ！

```
package myKanjiDemos;

import java.text.Normalizer;
import java.util.EnumSet;
import java.util.List;
```

```
import myKanjiDemos.MyStringToChars.CharsNormalize;

public class MyKanjiDemo01 {

    public static void main(String[] args) {

        String Str0 = "□□¼□2□□□□□";
        String Str1 = "ﾊﾟﾋﾞﾌﾟﾍﾞﾎﾟ"          ;
        String Str2 = "ぱ　び　ぷ　べ　ぽ "; //濁音・半濁音記号の結合文字入り

        CharDisp(Str0, Normalizer.Form.NFC);
        CharDisp(Str1, Normalizer.Form.NFC);
        CharDisp(Str2, Normalizer.Form.NFC);

        CharDisp(Str0, Normalizer.Form.NFKC);
        CharDisp(Str1, Normalizer.Form.NFKC);
        CharDisp(Str2, Normalizer.Form.NFKC);
    }

    public static void CharDisp(String str, Normalizer.Form normalize)
    {
        // java.text.Normalizer でノーマライズして
        String convedStr = Normalizer.normalize(str, normalize);

        //1文字毎に分解
        List<List<Integer>> chars =
MyStringToChars.parseToCodepoint(convedStr,
EnumSet.of(CharsNormalize.NONE));

        //分解結果の表示
        System.out.println("----------------------------------");
        System.out.println("Str       : " + str);
        System.out.println("StrLen    : " + str.length());
        System.out.println("convedStr : " + convedStr);
        System.out.println("convedLen : " + convedStr.length());
        System.out.println("Normalize : " + normalize);

        StringBuilder sbChars = new StringBuilder();
        StringBuilder sbCodeStr = new StringBuilder();

        for(Integer pos = 0; pos<chars.size(); pos++)
        {
            List<Integer> cs = chars.get(pos);
            sbChars.setLength(0);
            sbCodeStr.setLength(0);

            for(Integer idx = 0; idx<cs.size(); idx++)
            {
                sbChars.appendCodePoint(cs.get(idx));
                if (idx != 0) sbCodeStr.append( ", " );
```

```
                sbCodeStr.append( "u+" + Integer.toHexString(cs.get(idx)) );
            }

            System.out.println("Pos " + pos + "     : " + sbChars.toString()
+ " " + sbCodeStr.toString());
        }
    }
}
```

デモコード

コード中のdemo_Normalizeにオプションを渡して変化を確認します。以下に示しているものはオプション指定なし（正規化指定なし）の状態。

MyKanjiDemo02

サロゲートペア、異字体セレクタ、結合文字を混ぜ込んだ文字列のデモ

```
package myKanjiDemos;

import java.util.EnumSet;

import myKanjiDemos.MyStringToChars.CharsNormalize;

public class MyKanjiDemo02 {

    public static void main(String[] args) {

        String Str0 = "伊藤";          //通常
        String Str1 = "叱    家 ;  "   //サロゲートペア混在
        String Str2 = "葛葛    葛 ;  "     //通常＋異字体セレクタ
        String Str3 = "          ;       "//サロゲートペア＋異字体セレクタ

        EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.NONE);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.HALF2FULL_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA,
CharsNormalize.HALF2FULL_KANA);

        MyDemoMethod.charDisp(Str0, demo_Normalize);
        MyDemoMethod.charDisp(Str1, demo_Normalize);
        MyDemoMethod.charDisp(Str2, demo_Normalize);
        MyDemoMethod.charDisp(Str3, demo_Normalize);

    }
```

```
}
```

MyKanjiDemo03

カナと濁音・半濁音記号混在の文字列のデモ

```
package myKanjiDemos;

import java.util.EnumSet;

import myKanjiDemos.MyStringToChars.CharsNormalize;

public class MyKanjiDemo03 {

    public static void main(String[] args) {

        String Str0 = "ﾊﾋﾌﾍﾎﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ"             ;                          //半角カタカナ
        String Str1 = "ﾊﾞ ﾋﾟ ﾌﾞ ﾍﾟ ﾎﾞ "    ;                                 //半角カタカナに結合文字の濁音・半濁音記号を付与
        String Str2 = "ﾊﾟ ﾋﾞ ﾌﾟ ﾍﾞ ﾎﾟ ﾊﾟﾋﾞﾌﾟﾍﾞﾎﾟ"    ;    //片仮名に濁音・半濁音記号付与
        String Str3 = "は゜ び゛ ふ゜ へ゛ ほ゜ は゜び゛ふ゜へ゛ほ゜"    ;    //平仮名に濁音・半濁音記号付与
        String Str4 = "ぱびぷべぽバビブベボ";                    //カナに濁音・半濁音記号付き
        String Str5 = "ぱ び ぷ べ ぽ バ ピ ブ ペ ボ ";
//カナに結合文字の濁音・半濁音記号付与

        EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.NONE);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.HALF2FULL_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA,
CharsNormalize.HALF2FULL_KANA);

        MyDemoMethod.charDisp(Str0, demo_Normalize);
        MyDemoMethod.charDisp(Str1, demo_Normalize);
        MyDemoMethod.charDisp(Str2, demo_Normalize);
        MyDemoMethod.charDisp(Str3, demo_Normalize);
        MyDemoMethod.charDisp(Str4, demo_Normalize);
        MyDemoMethod.charDisp(Str5, demo_Normalize);

    }
}
```

MyKanjiDemo04

半角・全角カナを混在した文字列のデモ

```
package myKanjiDemos;

import java.util.EnumSet;

import myKanjiDemos.MyStringToChars.CharsNormalize;

public class MyKanjiDemo04 {
    public static void main(String[] args) {

        String Str0 = "ﾟ ﾞ ﾐﾟ ﾐﾞ ﾐﾟ ﾐﾞ 伊 藤 ";                    //濁音・半濁音記号の組み合わせが無いカナに結合文字で濁音・半濁音記号を付与
        String Str1 = "ﾞ ﾟ ﾐﾞ ﾐﾟ ﾐﾞ ﾐﾟ 伊ﾟ 藤ﾞ ";   //濁音・半濁音記号の組み合わせが無いカナに濁音・半濁音記号を付与
        String Str2 = "ﾞﾟ ﾐﾞﾐﾟﾐﾞﾐﾟ 伊ﾟ藤ﾞ";            //濁音・半濁音記号の組み合わせが無いカナに半角濁音・半濁音記号を付与

        EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.NONE);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.HALF2FULL_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA,
CharsNormalize.HALF2FULL_KANA);

        MyDemoMethod.charDisp(Str0, demo_Normalize);
        MyDemoMethod.charDisp(Str1, demo_Normalize);
        MyDemoMethod.charDisp(Str2, demo_Normalize);

    }

}
```

MyKanjiDemo05

濁音・半濁音記号で構成した文字列のデモ

```
package myKanjiDemos;

import java.util.EnumSet;

import myKanjiDemos.MyStringToChars.CharsNormalize;

public class MyKanjiDemo05 {
```

```
    public static void main(String[] args) {
        StringBuilder sb2 = new StringBuilder();

        sb2.appendCodePoint(0x3099);
        sb2.appendCodePoint(0x309A);

        String Str0 = "□□";                    //濁音・半濁音記号を連続
        String Str1 = "□□";                      //濁音・半濁音記号を連続
        String Str2 = sb2.toString();            //濁音記号の結合文字を連続
        String Str3 = "は゛゜ ひ゛゜ ふ゛゜ へ゛゜"; //濁音・半濁音記号を連続
        String Str4 = "ば゜ び゜ ぶ゜ べ゛゜"; //濁音・半濁音記号を連続
        String Str5 = "ﾊﾞﾋﾞﾌﾟﾍﾟ"       ;      //濁音・半濁音記号を連続
        String Str6 = "ﾝ゛ん"  ;                  //濁音・半濁音記号を連続
        String Str7 = "ﾝ゛ん";                  //濁音・半濁音記号を連続

        EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.NONE);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.HALF2FULL_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA,
CharsNormalize.HALF2FULL_KANA);

        MyDemoMethod.charDisp(Str0, demo_Normalize);
        MyDemoMethod.charDisp(Str1, demo_Normalize);
        MyDemoMethod.charDisp(Str2, demo_Normalize);
        MyDemoMethod.charDisp(Str3, demo_Normalize);
        MyDemoMethod.charDisp(Str4, demo_Normalize);
        MyDemoMethod.charDisp(Str5, demo_Normalize);
        MyDemoMethod.charDisp(Str6, demo_Normalize);
        MyDemoMethod.charDisp(Str7, demo_Normalize);

    }

}
```

MyKanjiDemo06

記号・絵文字を含む文字列のデモ

```
package myKanjiDemos;

import java.util.EnumSet;

import myKanjiDemos.MyStringToChars.CharsNormalize;

public class MyKanjiDemo06 {
    public static void main(String[] args) {
```

```
        String Str0 = "□□□□□□";                    //半角記号
        String Str1 = "㈱¼ﾍﾟｾﾝﾄ①2  ③❹㈤㈥ⓓ㎜";   //記号
        String Str2 = "🙇";                          //絵文字

        EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.NONE);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.HALF2FULL_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA);
        // EnumSet<CharsNormalize> demo_Normalize =
EnumSet.of(CharsNormalize.COMBININGVOICEMARK_KANA,
CharsNormalize.HALF2FULL_KANA);

        MyDemoMethod.charDisp(Str0, demo_Normalize);
        MyDemoMethod.charDisp(Str1, demo_Normalize);
        MyDemoMethod.charDisp(Str2, demo_Normalize);

    }
}
```

実行結果

MyKanjiDemo02の実行結果

呼び出し結果。**サロゲートペアに異字体セレクタがついてもとりあえず分解できた。**

実行結果。

NONE	HALF2FULL_KANA	COMBININGVOICEMARK_KANA	COMBININGVOICEMARK_KANA + HALF2FULL_KANA
----------	----------	----------	----------
Str : 伊藤	Str : 伊藤	Str : 伊藤	Str : 伊藤
StrLen : 2	StrLen : 2	StrLen : 2	StrLen : 2
RebuildStr : 伊藤	RebuildStr : 伊藤	RebuildStr : 伊藤	RebuildStr : 伊藤
RebuildLen : 2	RebuildLen : 2	RebuildLen : 2	RebuildLen : 2
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	parseOption : [COMBININGVOICEMARK_KANA]	parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos 0 : 伊 u+4f0a	Pos 0 : 伊 u+4f0a	Pos 0 : 伊 u+4f0a	Pos 0 : 伊 u+4f0a
Pos 1 : 藤 u+85e4	Pos 1 : 藤 u+85e4	Pos 1 : 藤 u+85e4	Pos 1 : 藤 u+85e4
----------	----------	----------	----------
Str : 叱　家	Str : 叱　家	Str : 叱　家	Str : 叱　家
StrLen : 6	StrLen : 6	StrLen : 6	StrLen : 6
RebuildStr : 叱　家	RebuildStr : 叱　家	RebuildStr : 叱　家	RebuildStr : 叱　家
RebuildLen : 6	RebuildLen : 6	RebuildLen : 6	RebuildLen : 6
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	parseOption : [COMBININGVOICEMARK_KANA]	parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos 0 : 叱 u+53f1	Pos 0 : 叱 u+53f1	Pos 0 : 叱 u+53f1	Pos 0 : 叱 u+53f1
Pos 1 : 　 u+20bb7	Pos 1 : 　 u+20bb7	Pos 1 : 　 u+20bb7	Pos 1 : 　 u+20bb7
Pos 2 : 家 u+5bb6	Pos 2 : 家 u+5bb6	Pos 2 : 家 u+5bb6	Pos 2 : 家 u+5bb6
Pos 3 : 　 u+20b9f	Pos 3 : 　 u+20b9f	Pos 3 : 　 u+20b9f	Pos 3 : 　 u+20b9f
----------	----------	----------	----------
Str : 葛葛　葛	Str : 葛葛　葛	Str : 葛葛　葛	Str : 葛葛　葛
StrLen : 7	StrLen : 7	StrLen : 7	StrLen : 7
RebuildStr : 葛葛　葛	RebuildStr : 葛葛　葛	RebuildStr : 葛葛　葛	RebuildStr : 葛葛　葛
RebuildLen : 7	RebuildLen : 7	RebuildLen : 7	RebuildLen : 7
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	parseOption : [COMBININGVOICEMARK_KANA]	parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos 0 : 葛 u+845b	Pos 0 : 葛 u+845b	Pos 0 : 葛 u+845b	Pos 0 : 葛 u+845b
Pos 1 : 葛 u+845b, u+e0109	Pos 1 : 葛 u+845b, u+e0109	Pos 1 : 葛 u+845b, u+e0109	Pos 1 : 葛 u+845b, u+e0109
Pos 2 : 葛 u+845b, u+e0100	Pos 2 : 葛 u+845b, u+e0100	Pos 2 : 葛 u+845b, u+e0100	Pos 2 : 葛 u+845b, u+e0100
----------	----------	----------	----------
Str :	Str :	Str :	Str :
StrLen : 10	StrLen : 10	StrLen : 10	StrLen : 10
RebuildStr :	RebuildStr :	RebuildStr :	RebuildStr :
RebuildLen : 10	RebuildLen : 10	RebuildLen : 10	RebuildLen : 10
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	parseOption : [COMBININGVOICEMARK_KANA]	parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos 0 : 　 u+26e40	Pos 0 : 　 u+26e40	Pos 0 : 　 u+26e40	Pos 0 : 　 u+26e40
Pos 1 : 　 u+26e40, u+e0103	Pos 1 : 　 u+26e40, u+e0103	Pos 1 : 　 u+26e40, u+e0103	Pos 1 : 　 u+26e40, u+e0103
Pos 2 : 　 u+26e40, u+e0102	Pos 2 : 　 u+26e40, u+e0102	Pos 2 : 　 u+26e40, u+e0102	Pos 2 : 　 u+26e40, u+e0102

MyKanjiDemo03の実行結果

かなと濁音・半濁音記号**の組み合わせ**。

実行結果。

NONE	HALF2FULL_KANA	COMBININGVOICEMARK_KANA	COMBININGVOICEMARK_KANA + HALF2FULL_KANA

NONE

```
----------------------------------
Str          : ﾊﾋﾌﾍﾎﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ
StrLen       : 15
RebuildStr   : ﾊﾋﾌﾍﾎﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ
RebuildLen   : 15
parseOption  : [NONE]
Pos       0 : ﾊ u+ff8a
Pos       1 : ﾋ u+ff8b
Pos       2 : ﾌ u+ff8c
Pos       3 : ﾍ u+ff8d
Pos       4 : ﾎ u+ff8e
Pos       5 : ﾊﾟ  u+ff8a, u+ff9f
Pos       6 : ﾋﾟ  u+ff8b, u+ff9e
Pos       7 : ﾌﾟ  u+ff8c, u+ff9f
Pos       8 : ﾍﾟ  u+ff8d, u+ff9e
Pos       9 : ﾎﾟ  u+ff8e, u+ff9f
----------------------------------
Str          : バ ピ ブ ペ ポ
StrLen       : 10
RebuildStr   : バ ピ ブ ペ ポ
RebuildLen   : 10
parseOption  : [NONE]
Pos       0 : バ  u+ff8a, u+3099
Pos       1 : ピ  u+ff8b, u+309a
Pos       2 : ブ  u+ff8c, u+3099
Pos       3 : ペ  u+ff8d, u+309a
Pos       4 : ポ  u+ff8e, u+3099
----------------------------------
Str          : パ ピ プ ペ ポ パ ピ プ ペ ポ
StrLen       : 20
RebuildStr   : パ ピ プ ペ ポ パ ピ プ ペ ポ
RebuildLen   : 20
parseOption  : [NONE]
Pos       0 : パ  u+30cf, u+309c
Pos       1 : ピ  u+30d2, u+309b
Pos       2 : プ  u+30d5, u+309c
Pos       3 : ペ  u+30d8, u+309b
Pos       4 : ポ  u+30db, u+309c
Pos       5 : パ  u+30cf, u+ff9f
Pos       6 : ピ  u+30d2, u+ff9e
Pos       7 : プ  u+30d5, u+ff9f
Pos       8 : ペ  u+30d8, u+ff9e
Pos       9 : ポ  u+30db, u+ff9f
----------------------------------
Str          : ぱ ぴ ぷ ぺ ぽ ぱ ぴ ぷ ぺ ぽ
StrLen       : 20
RebuildStr   : ぱ ぴ ぷ ぺ ぽ ぱ ぴ ぷ ぺ ぽ
RebuildLen   : 20
parseOption  : [NONE]
Pos       0 : ぱ  u+306f, u+309c
Pos       1 : ぴ  u+3072, u+309b
Pos       2 : ぷ  u+3075, u+309c
Pos       3 : ぺ  u+3078, u+309b
Pos       4 : ぽ  u+307b, u+309c
Pos       5 : ぱ  u+306f, u+ff9f
Pos       6 : ぴ  u+3072, u+ff9e
Pos       7 : ぷ  u+3075, u+ff9f
Pos       8 : ぺ  u+3078, u+ff9e
Pos       9 : ぽ  u+307b, u+ff9f
----------------------------------
Str          : ぱぴぷぺぽバビブベボ
StrLen       : 10
RebuildStr   : ぱぴぷぺぽバビブベボ
RebuildLen   : 10
parseOption  : [NONE]
Pos       0 : ぱ u+3071
Pos       1 : ぴ u+3073
Pos       2 : ぷ u+3077
Pos       3 : ぺ u+3079
Pos       4 : ぽ u+307d
Pos       5 : バ u+30d0
Pos       6 : ビ u+30d3
Pos       7 : ブ u+30d6
Pos       8 : ベ u+30d9
Pos       9 : ボ u+30dc
----------------------------------
Str          : ぱ ぴ ぷ ぺ ぽ バ ピ ブ ペ ボ
StrLen       : 20
RebuildStr   : ぱ ぴ ぷ ぺ ぽ バ ピ ブ ペ ボ
RebuildLen   : 20
parseOption  : [NONE]
Pos       0 : ぱ  u+306f, u+309a
Pos       1 : ぴ  u+3072, u+3099
Pos       2 : ぷ  u+3075, u+309a
Pos       3 : ぺ  u+3078, u+3099
Pos       4 : ぽ  u+307b, u+309a
Pos       5 : バ  u+30cf, u+3099
Pos       6 : ピ  u+30d2, u+309a
Pos       7 : ブ  u+30d5, u+3099
Pos       8 : ペ  u+30d8, u+309a
Pos       9 : ボ  u+30db, u+3099
```

HALF2FULL_KANA

```
----------------------------------
Str          : ﾊﾋﾌﾍﾎﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ
StrLen       : 15
RebuildStr   : ハヒフヘホハ゜ヒ゜フ゜ヘ゜ホ゜
RebuildLen   : 15
parseOption  : [HALF2FULL_KANA]
Pos       0 : ハ u+30cf
Pos       1 : ヒ u+30d2
Pos       2 : フ u+30d5
Pos       3 : ヘ u+30d8
Pos       4 : ホ u+30db
Pos       5 : パ  u+30cf, u+309c
Pos       6 : ピ  u+30d2, u+309b
Pos       7 : プ  u+30d5, u+309c
Pos       8 : ペ  u+30d8, u+309b
Pos       9 : ポ  u+30db, u+309c
----------------------------------
Str          : バ ピ ブ ペ ボ
StrLen       : 10
RebuildStr   : バ ピ ブ ペ ボ
RebuildLen   : 10
parseOption  : [HALF2FULL_KANA]
Pos       0 : バ  u+30cf, u+3099
Pos       1 : ピ  u+30d2, u+309a
Pos       2 : ブ  u+30d5, u+3099
Pos       3 : ペ  u+30d8, u+309a
Pos       4 : ボ  u+30db, u+3099
----------------------------------
Str          : パ ピ プ ペ ポ パ ピ プ ペ ポ
StrLen       : 20
RebuildStr   : パ ピ プ ペ ポ パ ピ プ ペ ポ
RebuildLen   : 20
parseOption  : [HALF2FULL_KANA]
Pos       0 : パ  u+30cf, u+309c
Pos       1 : ピ  u+30d2, u+309b
Pos       2 : プ  u+30d5, u+309c
Pos       3 : ペ  u+30d8, u+309b
Pos       4 : ポ  u+30db, u+309c
Pos       5 : パ  u+30cf, u+309c
Pos       6 : ピ  u+30d2, u+309b
Pos       7 : プ  u+30d5, u+309c
Pos       8 : ペ  u+30d8, u+309b
Pos       9 : ポ  u+30db, u+309c
----------------------------------
Str          : ぱ ぴ ぷ ぺ ぽ ぱ ぴ ぷ ぺ ぽ
StrLen       : 20
RebuildStr   : ぱ ぴ ぷ ぺ ぽ ぱ ぴ ぷ ぺ ぽ
RebuildLen   : 20
parseOption  : [HALF2FULL_KANA]
Pos       0 : ぱ  u+306f, u+309c
Pos       1 : ぴ  u+3072, u+309b
Pos       2 : ぷ  u+3075, u+309c
Pos       3 : ぺ  u+3078, u+309b
Pos       4 : ぽ  u+307b, u+309c
Pos       5 : ぱ  u+306f, u+309c
Pos       6 : ぴ  u+3072, u+309b
Pos       7 : ぷ  u+3075, u+309c
Pos       8 : ぺ  u+3078, u+309b
Pos       9 : ぽ  u+307b, u+309c
----------------------------------
Str          : ぱぴぷぺぽバビブベボ
StrLen       : 10
RebuildStr   : ぱぴぷぺぽバビブベボ
RebuildLen   : 10
parseOption  : [HALF2FULL_KANA]
Pos       0 : ぱ u+3071
Pos       1 : ぴ u+3073
Pos       2 : ぷ u+3077
Pos       3 : ぺ u+3079
Pos       4 : ぽ u+307d
Pos       5 : バ u+30d0
Pos       6 : ビ u+30d3
Pos       7 : ブ u+30d6
Pos       8 : ベ u+30d9
Pos       9 : ボ u+30dc
----------------------------------
Str          : ぱ ぴ ぷ ぺ ぽ バ ヒ ブ ペ ボ
StrLen       : 20
RebuildStr   : ぱ ぴ ぷ ぺ ぽ バ ヒ ブ ペ ボ
RebuildLen   : 20
parseOption  : [HALF2FULL_KANA]
Pos       0 : ぱ  u+306f, u+309a
Pos       1 : ぴ  u+3072, u+3099
Pos       2 : ぷ  u+3075, u+309a
Pos       3 : ぺ  u+3078, u+3099
Pos       4 : ぽ  u+307b, u+309a
Pos       5 : バ  u+30cf, u+3099
Pos       6 : ピ  u+30d2, u+309a
Pos       7 : ブ  u+30d5, u+3099
Pos       8 : ペ  u+30d8, u+309a
Pos       9 : ボ  u+30db, u+3099
```

COMBININGVOICEMARK_KANA

```
----------------------------------
Str          : ﾊﾋﾌﾍﾎﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ
StrLen       : 15
RebuildStr   : ﾊﾋﾌﾍﾎﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ
RebuildLen   : 15
parseOption  : [COMBININGVOICEMARK_KANA]
Pos       0 : ﾊ u+ff8a
Pos       1 : ﾋ u+ff8b
Pos       2 : ﾌ u+ff8c
Pos       3 : ﾍ u+ff8d
Pos       4 : ﾎ u+ff8e
Pos       5 : ﾊﾟ  u+ff8a, u+ff9f
Pos       6 : ﾋﾟ  u+ff8b, u+ff9e
Pos       7 : ﾌﾟ  u+ff8c, u+ff9f
Pos       8 : ﾍﾟ  u+ff8d, u+ff9e
Pos       9 : ﾎﾟ  u+ff8e, u+ff9f
----------------------------------
Str          : バ ピ ブ ペ ポ
StrLen       : 10
RebuildStr   : ﾊﾞﾋﾞﾌﾞﾍﾞﾎﾞ
RebuildLen   : 10
parseOption  : [COMBININGVOICEMARK_KANA]
Pos       0 : ﾊﾞ  u+ff8a, u+309b
Pos       1 : ﾋﾟ  u+ff8b, u+309c
Pos       2 : ﾌﾞ  u+ff8c, u+309b
Pos       3 : ﾍﾟ  u+ff8d, u+309c
Pos       4 : ﾎﾞ  u+ff8e, u+309b
----------------------------------
Str          : パ ピ プ ペ ポ パ ピ プ ペ ポ
StrLen       : 20
RebuildStr   : パピプペポパﾋﾟプペポ
RebuildLen   : 15
parseOption  : [COMBININGVOICEMARK_KANA]
Pos       0 : パ u+30d1
Pos       1 : ピ u+30d3
Pos       2 : プ u+30d7
Pos       3 : ペ u+30d9
Pos       4 : ポ u+30dd
Pos       5 : パ  u+30cf, u+ff9f
Pos       6 : ヒ゛  u+30d2, u+ff9e
Pos       7 : フ゜  u+30d5, u+ff9f
Pos       8 : ヘ゜  u+30d8, u+ff9e
Pos       9 : ホ゜  u+30db, u+ff9f
----------------------------------
Str          : ぱ ぴ ぷ ぺ ぽ ぱぴぷぺぽ
StrLen       : 20
RebuildStr   : ぱぴぷぺぽぱ゜ぴ゜ぷ゜ぺ゜ぽ゜
RebuildLen   : 15
parseOption  : [COMBININGVOICEMARK_KANA]
Pos       0 : ぱ u+3071
Pos       1 : ぴ u+3073
Pos       2 : ぷ u+3077
Pos       3 : ぺ u+3079
Pos       4 : ぽ u+307d
Pos       5 : ば  u+306f, u+ff9f
Pos       6 : び  u+3072, u+ff9e
Pos       7 : ふ゜  u+3075, u+ff9f
Pos       8 : へ゜  u+3078, u+ff9e
Pos       9 : ほ゜  u+307b, u+ff9f
----------------------------------
Str          : ぱぴぷぺぽバビブベボ
StrLen       : 10
RebuildStr   : ぱぴぷぺぽバビブベボ
RebuildLen   : 10
parseOption  : [COMBININGVOICEMARK_KANA]
Pos       0 : ぱ u+3071
Pos       1 : ぴ u+3073
Pos       2 : ぷ u+3077
Pos       3 : ぺ u+3079
Pos       4 : ぽ u+307d
Pos       5 : バ u+30d0
Pos       6 : ビ u+30d3
Pos       7 : ブ u+30d6
Pos       8 : ベ u+30d9
Pos       9 : ボ u+30dc
----------------------------------
Str          : ぱ ぴ ぷ ぺ ぽ バ ヒ ブ ペ ボ
StrLen       : 20
RebuildStr   : ぱぴぷぺぽバビブベボ
RebuildLen   : 10
parseOption  : [COMBININGVOICEMARK_KANA]
Pos       0 : ぱ u+3071
Pos       1 : ぴ u+3073
Pos       2 : ぷ u+3077
Pos       3 : ぺ u+3079
Pos       4 : ぽ u+307d
Pos       5 : バ u+30d0
Pos       6 : ピ゛ u+30d2, u+309a
Pos       7 : ブ u+30d6
Pos       8 : ペ u+30da
Pos       9 : ボ u+30dc
```

COMBININGVOICEMARK_KANA + HALF2FULL_KANA

```
----------------------------------
Str          : ﾊﾋﾌﾍﾎﾊﾟﾋﾟﾌﾟﾍﾟﾎﾟ
StrLen       : 15
RebuildStr   : ハヒフヘホパピプペポ
RebuildLen   : 10
parseOption  : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos       0 : ハ u+30cf
Pos       1 : ヒ u+30d2
Pos       2 : フ u+30d5
Pos       3 : ヘ u+30d8
Pos       4 : ホ u+30db
Pos       5 : パ u+30d1
Pos       6 : ピ u+30d3
Pos       7 : プ u+30d7
Pos       8 : ペ u+30d9
Pos       9 : ポ u+30dd
----------------------------------
Str          : バ ピ ブ ペ ボ
StrLen       : 10
RebuildStr   : バピブペボ
RebuildLen   : 5
parseOption  : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos       0 : バ u+30d0
Pos       1 : ピ u+30d4
Pos       2 : ブ u+30d6
Pos       3 : ペ u+30da
Pos       4 : ボ u+30dc
----------------------------------
Str          : パ ピ プ ペ ポ パ ピ プ ペ ポ
StrLen       : 20
RebuildStr   : パピプペポパピプペポ
RebuildLen   : 10
parseOption  : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos       0 : パ u+30d1
Pos       1 : ピ u+30d3
Pos       2 : プ u+30d7
Pos       3 : ペ u+30d9
Pos       4 : ポ u+30dd
Pos       5 : パ u+30d1
Pos       6 : ピ u+30d3
Pos       7 : プ u+30d7
Pos       8 : ペ u+30d9
Pos       9 : ポ u+30dd
----------------------------------
Str          : ぱ ぴ ぷ ぺ ぽ ぱぴぷぺぽ
StrLen       : 20
RebuildStr   : ぱぴぷぺぽぱぴぷぺぽ
RebuildLen   : 10
parseOption  : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos       0 : ぱ u+3071
Pos       1 : ぴ u+3073
Pos       2 : ぷ u+3077
Pos       3 : ぺ u+3079
Pos       4 : ぽ u+307d
Pos       5 : ぱ u+3071
Pos       6 : ぴ u+3073
Pos       7 : ぷ u+3077
Pos       8 : ぺ u+3079
Pos       9 : ぽ u+307d
----------------------------------
Str          : ぱぴぷぺぽバビブベボ
StrLen       : 10
RebuildStr   : ぱぴぷぺぽバビブベボ
RebuildLen   : 10
parseOption  : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos       0 : ぱ u+3071
Pos       1 : ぴ u+3073
Pos       2 : ぷ u+3077
Pos       3 : ぺ u+3079
Pos       4 : ぽ u+307d
Pos       5 : バ u+30d0
Pos       6 : ビ u+30d3
Pos       7 : ブ u+30d6
Pos       8 : ベ u+30d9
Pos       9 : ボ u+30dc
----------------------------------
Str          : ぱ ぴ ぷ ぺ ぽ バ ヒ ブ ペ ボ
StrLen       : 20
RebuildStr   : ぱぴぷぺぽバビブベボ
RebuildLen   : 10
parseOption  : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos       0 : ぱ u+3071
Pos       1 : ぴ u+3073
Pos       2 : ぷ u+3077
Pos       3 : ぺ u+3079
Pos       4 : ぽ u+307d
Pos       5 : バ u+30d0
Pos       6 : ピ゛ u+30d2, u+309a
Pos       7 : ブ u+30d6
Pos       8 : ペ u+30da
Pos       9 : ボ u+30dc
```

MyKanjiDemo04の実行結果

めったにない組み合わせの確認。大丈夫**そう**。

実行結果。

NONE	HALF2FULL_KANA	COMBININGVOICEMARK_KANA	COMBININGVOICEMARK_KANA + HALF2FULL_KANA
----------	----------	----------	----------
Str : ぺ゙ ゔ゙ ペ゙ ベ゙ ペ゚ ベ゚	Str : ぺ゙ ゔ゙ ペ゙ ベ゙ ペ゚ ベ゚	Str : ぺ゙ ゔ゙ ペ゙ ベ゙ ペ゚ ベ゚	Str : ぺ゙ ゔ゙ ペ゙ ベ゙ ペ゚ ベ゚
伊 藤	伊 藤	伊 藤	伊 藤
StrLen : 16	StrLen : 16	StrLen : 16	StrLen : 16
RebuildStr : ぺ゙ ゔ゙ ペ゙ ベ゙ ペ゚ ベ゚	RebuildStr : ぺ゙ ゔ゙ ペ゙ ベ゙ ペ゚ ベ゚	RebuildStr : ぺ゙ ゔ゙ ペ゙ ベ゙ ペ゚ ベ゚	RebuildStr : ぺ゙ ゔ゙ ペ゙ ベ゙ ペ゚ ベ゚
伊 藤	伊 藤	伊゙ 藤゚	伊゙ 藤゚
RebuildLen : 16	RebuildLen : 16	RebuildLen : 16	RebuildLen : 16
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	parseOption : [COMBININGVOICEMARK_KANA]	parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos 0 : ぺ゙ u+307f, u+309a	Pos 0 : ぺ゙ u+307f, u+309a	Pos 0 : ぺ゚ u+307f, u+309c	Pos 0 : ぺ゚ u+307f, u+309c
Pos 1 : ゔ゙ u+307f, u+3099	Pos 1 : ゔ゙ u+307f, u+3099	Pos 1 : ゙ u+307f, u+309b	Pos 1 : ゙ u+307f, u+309b
Pos 2 : ミ゙ u+30df, u+309a	Pos 2 : ミ゙ u+30df, u+309a	Pos 2 : ミ゚ u+30df, u+309c	Pos 2 : ミ゚ u+30df, u+309c
Pos 3 : ミ゙ u+30df, u+3099	Pos 3 : ミ゙ u+30df, u+3099	Pos 3 : ミ゙ u+30df, u+309b	Pos 3 : ミ゙ u+30df, u+309b
Pos 4 : ミ゙ u+ff90, u+309a	Pos 4 : ミ゙ u+30df, u+309a	Pos 4 : ミ゚ u+ff90, u+309c	Pos 4 : ミ゚ u+30df, u+309c
Pos 5 : ミ゙ u+ff90, u+3099	Pos 5 : ミ゙ u+30df, u+3099	Pos 5 : ミ゙ u+ff90, u+309b	Pos 5 : ミ゙ u+30df, u+309b
Pos 6 : 伊 u+4f0a, u+3099	Pos 6 : 伊 u+4f0a, u+3099	Pos 6 : 伊゙ u+4f0a, u+309b	Pos 6 : 伊゙ u+4f0a, u+309b
Pos 7 : 籐 u+85e4, u+309a	Pos 7 : 籐 u+85e4, u+309a	Pos 7 : 藤゚ u+85e4, u+309c	Pos 7 : 藤゚ u+85e4, u+309c
----------	----------	----------	----------
Str : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚	Str : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚	Str : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚	Str : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚
伊゚ 藤゙	伊゚ 藤゙	伊゚ 藤゙	伊゚ 藤゙
StrLen : 16	StrLen : 16	StrLen : 16	StrLen : 16
RebuildStr : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚	RebuildStr : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚	RebuildStr : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚	RebuildStr : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚
伊゚ 藤゙	伊゚ 藤゙	伊゚ 藤゙	伊゚ 藤゙
RebuildLen : 16	RebuildLen : 16	RebuildLen : 16	RebuildLen : 16
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	parseOption : [COMBININGVOICEMARK_KANA]	parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos 0 : み゙ u+307f, u+309b	Pos 0 : み゙ u+307f, u+309b	Pos 0 : み゙ u+307f, u+309b	Pos 0 : み゙ u+307f, u+309b
Pos 1 : み゚ u+307f, u+309c	Pos 1 : み゚ u+307f, u+309c	Pos 1 : み゚ u+307f, u+309c	Pos 1 : み゚ u+307f, u+309c
Pos 2 : ミ゙ u+30df, u+309b	Pos 2 : ミ゙ u+30df, u+309b	Pos 2 : ミ゙ u+30df, u+309b	Pos 2 : ミ゙ u+30df, u+309b
Pos 3 : ミ゚ u+30df, u+309c	Pos 3 : ミ゚ u+30df, u+309c	Pos 3 : ミ゚ u+30df, u+309c	Pos 3 : ミ゚ u+30df, u+309c
Pos 4 : ミ゙ u+ff90, u+309b	Pos 4 : ミ゙ u+30df, u+309b	Pos 4 : ミ゙ u+ff90, u+309b	Pos 4 : ミ゙ u+30df, u+309b
Pos 5 : ミ゚ u+ff90, u+309c	Pos 5 : ミ゚ u+30df, u+309c	Pos 5 : ミ゚ u+ff90, u+309c	Pos 5 : ミ゚ u+30df, u+309c
Pos 6 : 伊゚ u+4f0a, u+309c	Pos 6 : 伊゚ u+4f0a, u+309c	Pos 6 : 伊゚ u+4f0a, u+309c	Pos 6 : 伊゚ u+4f0a, u+309c
Pos 7 : 藤゙ u+85e4, u+309b	Pos 7 : 藤゙ u+85e4, u+309b	Pos 7 : 藤゙ u+85e4, u+309b	Pos 7 : 藤゙ u+85e4, u+309b
----------	----------	----------	----------
Str : み゙み゚ ミ゙ミ゚ミ゙ミ゚	Str : み゙み゚ ミ゙ミ゚ミ゙ミ゚	Str : み゙み゚ ミ゙ミ゚ミ゙ミ゚	Str : み゙み゚ ミ゙ミ゚ミ゙ミ゚
伊゚藤゙	伊゚藤゙	伊゚藤゙	伊゚藤゙
StrLen : 16	StrLen : 16	StrLen : 16	StrLen : 16
RebuildStr : み゙み゚ ミ゙ミ゚ミ゙ミ゚	RebuildStr : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚	RebuildStr : み゙み゚ ミ゙ミ゚ミ゙ミ゚	RebuildStr : み゙ み゚ ミ゙ ミ゚ ミ゙ ミ゚
伊゚藤゙	伊゚ 藤゙	伊゚藤゙	伊゚ 藤゙
RebuildLen : 16	RebuildLen : 16	RebuildLen : 16	RebuildLen : 16
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	parseOption : [COMBININGVOICEMARK_KANA]	parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA]
Pos 0 : み゙ u+307f, u+ff9e	Pos 0 : み゙ u+307f, u+309b	Pos 0 : み゙ u+307f, u+ff9e	Pos 0 : み゙ u+307f, u+309b
Pos 1 : み゚ u+307f, u+ff9f	Pos 1 : み゚ u+307f, u+309c	Pos 1 : み゚ u+307f, u+ff9f	Pos 1 : み゚ u+307f, u+309c
Pos 2 : ミ゙ u+30df, u+ff9e	Pos 2 : ミ゙ u+30df, u+309b	Pos 2 : ミ゙ u+30df, u+ff9e	Pos 2 : ミ゙ u+30df, u+309b
Pos 3 : ミ゚ u+30df, u+ff9f	Pos 3 : ミ゚ u+30df, u+309c	Pos 3 : ミ゚ u+30df, u+ff9f	Pos 3 : ミ゚ u+30df, u+309c
Pos 4 : ミ゙ u+ff90, u+ff9e	Pos 4 : ミ゙ u+30df, u+309b	Pos 4 : ミ゙ u+ff90, u+ff9e	Pos 4 : ミ゙ u+30df, u+309b
Pos 5 : ミ゚ u+ff90, u+ff9f	Pos 5 : ミ゚ u+30df, u+309c	Pos 5 : ミ゚ u+ff90, u+ff9f	Pos 5 : ミ゚ u+30df, u+309c
Pos 6 : 伊゚ u+4f0a, u+ff9f	Pos 6 : 伊゚ u+4f0a, u+309c	Pos 6 : 伊゚ u+4f0a, u+ff9f	Pos 6 : 伊゚ u+4f0a, u+309c
Pos 7 : 藤゙ u+85e4, u+ff9e	Pos 7 : 藤゙ u+85e4, u+309b	Pos 7 : 藤゙ u+85e4, u+ff9e	Pos 7 : 藤゙ u+85e4, u+309b

MyKanjiDemo05の実行結果

濁音・半濁音記号単体**の時の**処理。

実行結果。

NONE	HALF2FULL_KANA	COMBININGVOICEMARK_KANA	COMBININGVOICEMARK_KANA + HALF2FULL_KANA
		----------	----------
----------	----------	Str : □□	Str : □□
Str : □□	Str : □□	StrLen : 2	StrLen : 2
StrLen : 2	StrLen : 2	RebuildStr : □□	RebuildStr : □□
RebuildStr : □□	RebuildStr : □□	RebuildLen : 2	RebuildLen : 2
RebuildLen : 2	RebuildLen : 2	parseOption :	parseOption : [HALF2FULL_KANA,
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	[COMBININGVOICEMARK_KANA]	COMBININGVOICEMARK_KANA]
Pos 0 : □ u+309b	Pos 0 : □ u+309b	Pos 0 : □ u+309b	Pos 0 : □ u+309b
Pos 1 : □ u+309c	Pos 1 : □ u+309c	Pos 1 : □ u+309c	Pos 1 : □ u+309c
----------	----------	----------	----------
Str : □□	Str : □□	Str : □□	Str : □□
StrLen : 2	StrLen : 2	StrLen : 2	StrLen : 2
RebuildStr : □□	RebuildStr : □□	RebuildStr : □□	RebuildStr : □□
RebuildLen : 2	RebuildLen : 2	RebuildLen : 2	RebuildLen : 2
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	parseOption :	parseOption : [HALF2FULL_KANA,
Pos 0 : □ u+ff9e	Pos 0 : □ u+309b	[COMBININGVOICEMARK_KANA]	COMBININGVOICEMARK_KANA]
Pos 1 : □ u+ff9f	Pos 1 : □ u+309c	Pos 0 : □ u+ff9e	Pos 0 : □ u+309b
----------	----------	Pos 1 : □ u+ff9f	Pos 1 : □ u+309c
Str : □□	Str : □□	----------	----------
StrLen : 2	StrLen : 2	Str : □□	Str : □□
RebuildStr : □□	RebuildStr : □□	StrLen : 2	StrLen : 2
RebuildLen : 2	RebuildLen : 2	RebuildStr : □□	RebuildStr : □□
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	RebuildLen : 2	RebuildLen : 2
Pos 0 : □ u+309b	Pos 0 : □ u+309b	parseOption :	parseOption : [HALF2FULL_KANA,
Pos 1 : □ u+309c	Pos 1 : □ u+309c	[COMBININGVOICEMARK_KANA]	COMBININGVOICEMARK_KANA]
----------	----------	Pos 0 : □ u+309b	Pos 0 : □ u+309b
Str : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ	Str : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ	Pos 1 : □ u+309c	Pos 1 : □ u+309c
StrLen : 12	StrLen : 12	----------	----------
RebuildStr : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ	RebuildStr : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ	Str : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ	Str : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ
RebuildLen : 12	RebuildLen : 12	StrLen : 12	StrLen : 12
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	RebuildStr : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ	RebuildStr : ﾊﾟ ﾋﾟ ﾌﾟ ﾍﾟ
Pos 0 : ﾊﾞ u+306f, u+ff9e	Pos 0 : ﾊﾞ u+306f, u+309b	RebuildLen : 10	RebuildLen : 8
Pos 1 : □ u+ff9f	Pos 1 : □ u+309c	parseOption :	parseOption : [HALF2FULL_KANA,
Pos 2 : ﾋﾞ u+30d2, u+ff9e	Pos 2 : ﾋﾞ u+30d2, u+309b	[COMBININGVOICEMARK_KANA]	COMBININGVOICEMARK_KANA]
Pos 3 : □ u+ff9f	Pos 3 : □ u+309c	Pos 0 : ﾊﾞ u+306f, u+ff9e	Pos 0 : ﾊ u+3070
Pos 4 : ﾌﾞ u+3075, u+309b	Pos 4 : ﾌﾞ u+3075, u+309b	Pos 1 : □ u+ff9f	Pos 1 : □ u+309c
Pos 5 : □ u+309c	Pos 5 : □ u+309c	Pos 2 : ﾋﾞ u+30d2, u+ff9e	Pos 2 : ﾋﾞ u+30d3
Pos 6 : ﾍﾞ u+30d8, u+309b	Pos 6 : ﾍﾞ u+30d8, u+309b	Pos 3 : □ u+ff9f	Pos 3 : □ u+309c
Pos 7 : □ u+309c	Pos 7 : □ u+309c	Pos 4 : ﾌﾞ u+3076	Pos 4 : ﾌﾞ u+3076
----------	----------	Pos 5 : □ u+309c	Pos 5 : □ u+309c
Str : ﾊﾞﾋﾞﾌﾞﾍﾞ	Str : ﾊﾞﾋﾞﾌﾞﾍﾞ	Pos 6 : ﾍﾞ u+30d8, u+309b	Pos 6 : ﾍﾞ u+30d9
StrLen : 12	StrLen : 12	Pos 7 : □ u+309c	Pos 7 : □ u+309c
RebuildStr : ﾊﾞﾋﾞﾌﾞﾍﾞ	RebuildStr : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ	----------	----------
RebuildLen : 12	RebuildLen : 12	Str : ﾊﾞﾋﾞﾌﾞﾍﾞ	Str : ﾊﾞﾋﾞﾌﾞﾍﾞ
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	StrLen : 12	StrLen : 12
Pos 0 : ﾊ u+3070	Pos 0 : ﾊ u+3070	RebuildStr : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ	RebuildStr : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ
Pos 1 : □ u+ff9e	Pos 1 : □ u+309b	RebuildLen : 12	RebuildLen : 12
Pos 2 : □ u+ff9f	Pos 2 : □ u+309c	parseOption :	parseOption : [HALF2FULL_KANA,
Pos 3 : ビ u+30d3	Pos 3 : ビ u+30d3	[COMBININGVOICEMARK_KANA]	COMBININGVOICEMARK_KANA]
Pos 4 : □ u+ff9e	Pos 4 : □ u+309b	Pos 0 : ﾊ u+3070	Pos 0 : ﾊ u+3070
Pos 5 : □ u+ff9f	Pos 5 : □ u+309c	Pos 1 : □ u+ff9e	Pos 1 : □ u+309b
Pos 6 : ぶ u+3076	Pos 6 : ぶ u+3076	Pos 2 : □ u+ff9f	Pos 2 : □ u+309c
Pos 7 : □ u+309b	Pos 7 : □ u+309b	Pos 3 : ビ u+30d3	Pos 3 : ビ u+30d3
Pos 8 : □ u+309c	Pos 8 : □ u+309c	Pos 4 : □ u+ff9e	Pos 4 : □ u+309b
Pos 9 : ベ u+30d9	Pos 9 : ベ u+30d9	Pos 5 : □ u+ff9f	Pos 5 : □ u+309c
Pos 10 : □ u+309b	Pos 10 : □ u+309b	Pos 6 : ぶ u+3076	Pos 6 : ぶ u+3076
Pos 11 : □ u+309c	Pos 11 : □ u+309c	Pos 7 : □ u+309b	Pos 7 : □ u+309b
----------	----------	Pos 8 : □ u+309c	Pos 8 : □ u+309c
Str : ﾊﾞﾋﾞﾌﾞﾍﾞ	Str : ﾊﾞﾋﾞﾌﾞﾍﾞ	Pos 9 : ベ u+30d9	Pos 9 : ベ u+30d9
StrLen : 12	StrLen : 12	Pos 10 : □ u+309b	Pos 10 : □ u+309b
RebuildStr : ﾊﾞﾋﾞﾌﾞﾍﾞ	RebuildStr : ﾊﾞ ﾋﾞ ﾌﾞ ﾍﾞ	Pos 11 : □ u+309c	Pos 11 : □ u+309c
RebuildLen : 12	RebuildLen : 12	----------	----------
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	Str : ﾊﾞﾋﾞﾌﾞﾍﾞ	Str : ﾊﾞﾋﾞﾌﾞﾍﾞ
Pos 0 : ﾊﾞ u+ff8a, u+ff9e	Pos 0 : ﾊﾞ u+30cf, u+309b	StrLen : 12	StrLen : 12
Pos 1 : □ u+ff9f	Pos 1 : □ u+309c	RebuildStr : ﾊﾞﾋﾞﾌﾞﾍﾞ	RebuildStr : ﾊﾟ ﾋﾟ ﾌﾟ ﾍﾟ
Pos 2 : ﾋﾞ u+ff8b, u+ff9e	Pos 2 : ﾋﾞ u+30d2, u+309b	RebuildLen : 12	RebuildLen : 8
Pos 3 : □ u+ff9f	Pos 3 : □ u+309c	parseOption :	parseOption : [HALF2FULL_KANA,
Pos 4 : ﾌﾞ u+ff8c, u+309b	Pos 4 : ﾌﾞ u+30d5, u+309b	[COMBININGVOICEMARK_KANA]	COMBININGVOICEMARK_KANA]
Pos 5 : □ u+309c	Pos 5 : □ u+309c	Pos 0 : ﾊﾞ u+ff8a, u+ff9e	Pos 0 : ﾊ u+30d0
Pos 6 : ﾍﾞ u+ff8d, u+309b	Pos 6 : ﾍﾞ u+30d8, u+309b	Pos 1 : □ u+ff9f	Pos 1 : □ u+309c
Pos 7 : □ u+309c	Pos 7 : □ u+309c	Pos 2 : ﾋﾞ u+ff8b, u+ff9e	Pos 2 : ビ u+30d3
----------	----------	Pos 3 : □ u+ff9f	Pos 3 : □ u+309c
Str : □□ん	Str : □□ん	Pos 4 : ﾌﾞ u+ff8c, u+309b	Pos 4 : ブ u+30d6
StrLen : 3	StrLen : 3	Pos 5 : □ u+309c	Pos 5 : □ u+309c
RebuildStr : □□ん	RebuildStr : □□ん	Pos 6 : ﾍﾞ u+ff8d, u+309b	Pos 6 : ベ u+30d9
RebuildLen : 3	RebuildLen : 3	Pos 7 : □ u+309c	Pos 7 : □ u+309c
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	----------	----------
Pos 0 : □ u+ff9e	Pos 0 : □ u+309b	Str : □□ん	Str : □□ん
Pos 1 : □ u+ff9f	Pos 1 : □ u+309c	StrLen : 3	StrLen : 3
Pos 2 : ん u+3093	Pos 2 : ん u+3093	RebuildStr : □□ん	RebuildStr : □□ん
----------	----------	RebuildLen : 3	RebuildLen : 3
Str : □□ん	Str : □□ん	parseOption :	parseOption : [HALF2FULL_KANA,
StrLen : 3	StrLen : 3	[COMBININGVOICEMARK_KANA]	COMBININGVOICEMARK_KANA]
RebuildStr : □□ん	RebuildStr : □□ん	Pos 0 : □ u+ff9e	Pos 0 : □ u+309b
RebuildLen : 3	RebuildLen : 3	Pos 1 : □ u+ff9f	Pos 1 : □ u+309c
parseOption : [NONE]	parseOption : [HALF2FULL_KANA]	Pos 2 : ん u+3093	Pos 2 : ん u+3093
Pos 0 : □ u+309b	Pos 0 : □ u+309b	----------	----------
Pos 1 : □ u+309c	Pos 1 : □ u+309c	Str : □□ん	Str : □□ん
Pos 2 : ん u+3093	Pos 2 : ん u+3093	StrLen : 3	StrLen : 3
		RebuildStr : □□ん	RebuildStr : □□ん
		RebuildLen : 3	RebuildLen : 3
		parseOption :	parseOption : [HALF2FULL_KANA,
		[COMBININGVOICEMARK_KANA]	COMBININGVOICEMARK_KANA]
		Pos 0 : □ u+309b	Pos 0 : □ u+309b
		Pos 1 : □ u+309c	Pos 1 : □ u+309c
		Pos 2 : ん u+3093	Pos 2 : ん u+3093

MyKanjiDemo06の実行結果

記号・絵文字の時の処理。

実行結果。

NONE	HALF2FULL_KANA	COMBININGVOICEMARK_KANA	COMBININGVOICEMARK_KANA + HALF2FULL_KANA
----------	----------	----------	----------
Str : □□□□□□ StrLen : 6 RebuildStr : □□□□□□ RebuildLen : 6 parseOption : [NONE] Pos 0 : □ u+ffe9 Pos 1 : □ u+ffeb Pos 2 : □ u+ffea Pos 3 : □ u+ffec Pos 4 : □ u+ffed Pos 5 : □ u+ffee	Str : □□□□□□ StrLen : 6 RebuildStr : ←→↑↓■○ RebuildLen : 6 parseOption : [HALF2FULL_KANA] Pos 0 : ← u+2190 Pos 1 : → u+2192 Pos 2 : ↑ u+2191 Pos 3 : ↓ u+2193 Pos 4 : ■ u+25a0 Pos 5 : ○ u+25cb	Str : □□□□□□ StrLen : 6 RebuildStr : □□□□□□ RebuildLen : 6 parseOption : [COMBININGVOICEMARK_KANA] Pos 0 : □ u+ffe9 Pos 1 : □ u+ffeb Pos 2 : □ u+ffea Pos 3 : □ u+ffec Pos 4 : □ u+ffed Pos 5 : □ u+ffee	Str : □□□□□□ StrLen : 6 RebuildStr : ←→↑↓■○ RebuildLen : 6 parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA] Pos 0 : ← u+2190 Pos 1 : → u+2192 Pos 2 : ↑ u+2191 Pos 3 : ↓ u+2193 Pos 4 : ■ u+25a0 Pos 5 : ○ u+25cb
----------	----------	----------	----------
Str : □㌔①2 ③❹㈤㈥ⓓ㎜ StrLen : 12 RebuildStr : □㌔①2 ③❹㈤㈥ⓓ㎜ RebuildLen : 12 parseOption : [NONE] Pos 0 : □ u+3231 Pos 1 : ⅘ u+bc Pos 2 : ㌔ u+332b Pos 3 : □ u+2460 Pos 4 : 2□ u+32, u+20e3 Pos 5 : □ u+24f7 Pos 6 : □ u+2779 Pos 7 : □ u+3224 Pos 8 : □ u+3285 Pos 9 : □ u+24d3 Pos 10 : □ u+24a8	Str : □㌔①2 ③❹㈤㈥ⓓ㎜ StrLen : 12 RebuildStr : □㌔①2 ③❹㈤㈥ⓓ㎜ RebuildLen : 12 parseOption : [HALF2FULL_KANA] Pos 0 : □ u+3231 Pos 1 : ⅘ u+bc Pos 2 : ㌔ u+332b Pos 3 : □ u+2460 Pos 4 : 2□ u+32, u+20e3 Pos 5 : □ u+24f7 Pos 6 : □ u+2779 Pos 7 : □ u+3224 Pos 8 : □ u+3285 Pos 9 : □ u+24d3 Pos 10 : □ u+24a8	Str : □㌔①2 ③❹㈤㈥ⓓ㎜ StrLen : 12 RebuildStr : □㌔①2 ③❹㈤㈥ⓓ㎜ RebuildLen : 12 parseOption : [COMBININGVOICEMARK_KANA] Pos 0 : □ u+3231 Pos 1 : ⅘ u+bc Pos 2 : ㌔ u+332b Pos 3 : □ u+2460 Pos 4 : 2□ u+32, u+20e3 Pos 5 : □ u+24f7 Pos 6 : □ u+2779 Pos 7 : □ u+3224 Pos 8 : □ u+3285 Pos 9 : □ u+24d3 Pos 10 : □ u+24a8	Str : □㌔①2 ③❹㈤㈥ⓓ㎜ StrLen : 12 RebuildStr : □㌔①2 ③❹㈤㈥ⓓ㎜ RebuildLen : 12 parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA] Pos 0 : □ u+3231 Pos 1 : ⅘ u+bc Pos 2 : ㌔ u+332b Pos 3 : □ u+2460 Pos 4 : 2□ u+32, u+20e3 Pos 5 : □ u+24f7 Pos 6 : □ u+2779 Pos 7 : □ u+3224 Pos 8 : □ u+3285 Pos 9 : □ u+24d3 Pos 10 : □ u+24a8
----------	----------	----------	----------
Str : □◌゙ StrLen : 3 RebuildStr : □◌゙ RebuildLen : 3 parseOption : [NONE] Pos 0 : □ u+1f40e Pos 1 : ◌゙ u+270c	Str : □◌゙ StrLen : 3 RebuildStr : □◌゙ RebuildLen : 3 parseOption : [HALF2FULL_KANA] Pos 0 : □ u+1f40e Pos 1 : ◌゙ u+270c	Str : □◌゙ StrLen : 3 RebuildStr : □◌゙ RebuildLen : 3 parseOption : [COMBININGVOICEMARK_KANA] Pos 0 : □ u+1f40e Pos 1 : ◌゙ u+270c	Str : □◌゙ StrLen : 3 RebuildStr : □◌゙ RebuildLen : 3 parseOption : [HALF2FULL_KANA, COMBININGVOICEMARK_KANA] Pos 0 : □ u+1f40e Pos 1 : ◌゙ u+270c

作成した文字列分解クラス

1文字毎に切り出してコードポイントのリストを返すクラスをでっちあげる。異字体セレクタがあればこれもまとめる。
分解処理を提供するメソッド parseToCodepoints() を用意。

半角(HALFWIDE)□全角(FULLWIDE)に関わらず、濁音と半濁音記号は直前の文字に紐づくものとして扱う。ただし、直前の文字が濁音と半濁音記号だった場合は別の文字として扱う。※MyKanjiDemo05のあたり。

前々回より、不明瞭だった仕様が明確になりました。

MyStringToChars.java

```
package myKanjiDemos;

import java.util.ArrayList;
import java.util.EnumSet;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MyStringToChars {
```

```
    //文字列分解時のオプション
    public enum CharsNormalize
    {
        //ダミー
        NONE,
        //半角カナを全角カナへ変換する
        HALF2FULL_KANA,
        //文字と濁音・半濁音記号を1文字に統合する
        COMBININGVOICEMARK_KANA
    };

    //1文字リストの内容から文字列を生成する
    public static String makeStringFromList(List<List<Integer>>
parsedList)
    {
        StringBuilder sbRebuild = new StringBuilder();

        sbRebuild.setLength(0);
        for(int pos=0; pos < parsedList.size(); pos++) {
            for(int idx = 0; idx < parsedList.get(pos).size(); idx++) {
sbRebuild.appendCodePoint(parsedList.get(pos).get(idx));
            }
        }

        return sbRebuild.toString();
    }

    //文字列を1文字ずつ分解してリストにする
    public static List<List<Integer>> parseToCodepoints(String str,
EnumSet<CharsNormalize> normalize)
    {
        List<List<Integer>> ans = new ArrayList<List<Integer>>();

        Integer currentCodepoint;
        Integer codepoint;

        for(int idx = 0; idx<str.length(); idx++) {
            currentCodepoint = codepoint = str.codePointAt(idx);

            if(isDiacriticalMarksSymbols(codepoint)) { //異字体セレクタに対
応
                appendCharCodepoint(ans, codepoint);
            }
            else if ( isVariationSelectors(codepoint) ) { //記号用ダイアク
リティカルマーク処理
                appendCharCodepoint(ans, codepoint);
            }
            else { //通常

                if(normalize.contains(CharsNormalize.HALF2FULL_KANA)) {
```

```
//カナのFULLWIDE化
                    codepoint = normalizationHalf2FullKana(codepoint);
                }

if(normalize.contains(CharsNormalize.COMBININGVOICEMARK_KANA)) { //濁音・半濁音記号正規化処理
                    if (isSoundmarkKana(codepoint)) { //濁音・半濁音記号処理
                        concatCharAndSoundmarkKana(ans, codepoint);
                    }
                    else {
                        appendChar(ans, codepoint);
                    }
                }
                else { //通常!!!!!
                    if (isSoundmarkKana(codepoint)) {
                        mergeCharAndSoundmarkKana(ans, codepoint);
                    }
                    else {
                        appendChar(ans, codepoint);
                    }

                }
            }

             //入力文字列ではサロゲートペアの上位コードを指しているので
            if(currentCodepoint > 65535) idx++;

        }

        return ans;
    }

    //記号用ダイアクリティカルマーク判定
    private static boolean isDiacriticalMarksSymbols(int codepoint)
    {
        boolean ans = false;

        if( (codepoint >= 0x20D0)&& (codepoint <= 0x20FF) ) ans = true;

        return ans;
    }

    //異字体セレクタ判定
    private static boolean isVariationSelectors(int codepoint)
    {
        boolean ans = false;

        if ( ((codepoint>=0xe0100)&&(codepoint<=0xe01ef)) ||
             ((codepoint>=0xef00 )&&(codepoint<=0xef0f ))    ) {
            ans = true;
```

```
        }

        return ans;
    }

    //濁音半濁音記号判定
    private static boolean isSoundmarkKana(int soundmarkCodepoint)
    {
        boolean ans = false;

        if(mapVoicemarkKana.containsKey(soundmarkCodepoint)) {
            ans = true;
        }

        return ans;
    }

    //かな濁音半濁音記号付き判定
    private static boolean isKanaWithSoundmark(int kanaCodepoint)
    {
        boolean ans = false;

        if(mapKanaWithJudgeVoicemark.containsKey(kanaCodepoint)) {
            ans = !mapKanaWithJudgeVoicemark.get(kanaCodepoint);
        }

        return ans;
    }


    //半角カナを全角カナへ正規化する
    private static Integer normalizationHalf2FullKana(int
kanaCodepoint)
    {
        Integer ans = kanaCodepoint;

        if(mapHalf2FullKana.containsKey(kanaCodepoint)) {
            ans = mapHalf2FullKana.get(kanaCodepoint);
        }

        return ans;
    }

    //結合文字の濁音・半濁音記号を正規化する
    private static Integer normalizationSoundmarkKana(int
soundmarkCodepoint)
    {
        Integer ans = soundmarkCodepoint;

        if(mapVoicemarkKana.containsKey(soundmarkCodepoint)) {
            ans = mapVoicemarkKana.get(soundmarkCodepoint);
```

```
        }

        return ans;
    }


    //文字の構成コードポイントを追加する
    private static void appendCharCodepoint(List<List<Integer>>
charlist, int codepoint)
    {
        if(charlist != null) {

            if (charlist.size()==0) {
                charlist.add(new ArrayList<Integer>());
            }

            charlist.get(charlist.size()-1).add(codepoint);
        }
    }

    //文字を追加する
    private static void appendChar(List<List<Integer>> charlist, int
codepoint)
    {
        if(charlist != null) {
            charlist.add(new ArrayList<Integer>());
            charlist.get(charlist.size()-1).add(codepoint);
        }
    }


    //濁音・半濁音記号を直前の文字の構成コードポイントとして追加する
    private static void mergeCharAndSoundmarkKana(List<List<Integer>>
charlist, int soundmarkCodepoint)
    {
        List<Integer> targetChar;
        int kanaCodepoint;


        if(charlist.size()==0) { //統合先の文字がないので単体のまま
            charlist.add( new ArrayList<Integer>() );
charlist.get(0).add(normalizationSoundmarkKana(soundmarkCodepoint));
            return;
        }

        targetChar = charlist.get(charlist.size()-1);
        kanaCodepoint = targetChar.get(targetChar.size()-1);

        if (isSoundmarkKana(kanaCodepoint)) { // 統合先の文字が濁音・半濁音記
号なら単体文字追加にする
            charlist.add( new ArrayList<Integer>() );
```

```
charlist.get(charlist.size()-1).add(normalizationSoundmarkKana(soundmar
kCodepoint));
            return;
        }

        if (isKanaWithSoundmark(kanaCodepoint)) { //統合先の文字が濁音・半濁
音記号付きなら単体文字追加にする
            charlist.add( new ArrayList<Integer>() );
charlist.get(charlist.size()-1).add(normalizationSoundmarkKana(soundmar
kCodepoint));
            return;
        }

        //統合先の文字の構成要素に構成コードポイントを追加
        targetChar.add(normalizationSoundmarkKana(soundmarkCodepoint));
    }

    //濁音・半濁音記号付きの文字に正規化する
    private static void concatCharAndSoundmarkKana(List<List<Integer>>
charlist, int soundmarkCodepoint)
    {
        List<Integer> targetChar;
        int kanaCodepoint;
        int newsoundmark;

        if(charlist.size()==0) { //統合先の文字がないので単体のまま
            charlist.add( new ArrayList<Integer>() );
charlist.get(0).add(normalizationSoundmarkKana(soundmarkCodepoint));
            return;
        }

        targetChar = charlist.get(charlist.size()-1);
        kanaCodepoint = targetChar.get(targetChar.size()-1);

        if (isSoundmarkKana(kanaCodepoint)) { // 統合先の文字が濁音・半濁音記
号なら単体文字追加にする
            charlist.add( new ArrayList<Integer>() );
charlist.get(charlist.size()-1).add(normalizationSoundmarkKana(soundmar
kCodepoint));
            return;
        }

        if (isKanaWithSoundmark(kanaCodepoint)) { //統合先の文字が濁音・半濁
音記号付きなら単体文字追加にする
            charlist.add( new ArrayList<Integer>() );
charlist.get(charlist.size()-1).add(normalizationSoundmarkKana(soundmar
kCodepoint));
            return;
        }

        //統合先の文字のが濁音・半濁音記号を付与可能な場合は文字の差し替え
```

```
        newsoundmark = mapVoicemarkKana.get(soundmarkCodepoint);

        switch(newsoundmark)
        {
            case 0x309B:
                if (map309B.containsKey(kanaCodepoint)) {
                    targetChar.remove(targetChar.size()-1);
                    targetChar.add(map309B.get(kanaCodepoint));
                }
                else {
                    targetChar.add(newsoundmark);
                }
                break;

            case 0x309C:
                if (map309C.containsKey(kanaCodepoint)) {
                    targetChar.remove(targetChar.size()-1);
                    targetChar.add(map309C.get(kanaCodepoint));
                }
                else {
                    targetChar.add(newsoundmark);
                }
                break;

            default:
                targetChar.add(soundmarkCodepoint);
                break;
        }
    }

    private static Map<Integer, Integer> mapVoicemarkKana = new
HashMap<Integer, Integer>()
    {{
        put( 0x3099 , 0x309B ); //結合文字濁音
        put( 0x309A , 0x309C ); //結合文字半濁音
        put( 0x309B , 0x309B ); //濁音
        put( 0x309C , 0x309C ); //濁音
        put( 0xFF9E , 0x309B ); //ハーフワイド濁音
        put( 0xFF9F , 0x309C ); //ハーフワイド半濁音
    }};

    private static Map<Integer, Boolean> mapKanaWithJudgeVoicemark =
new HashMap<Integer, Boolean>()
    {{
        put( 0x3094  , false);  //ゔ
        put( 0x304C  , false);  //が
        put( 0x304E  , false);  //ぎ
        put( 0x3050  , false);  //ぐ
        put( 0x3052  , false);  //げ
        put( 0x3054  , false);  //ご
        put( 0x3056  , false);  //ざ
```

```
        put( 0x3058  , false);  //じ
        put( 0x305A  , false);  //ず
        put( 0x305C  , false);  //ぜ
        put( 0x305E  , false);  //ぞ
        put( 0x3060  , false);  //だ
        put( 0x3062  , false);  //ぢ
        put( 0x3065  , false);  //づ
        put( 0x3067  , false);  //で
        put( 0x3069  , false);  //ど
        put( 0x3070  , false);  //ば
        put( 0x3071  , false);  //ぱ
        put( 0x3073  , false);  //び
        put( 0x3074  , false);  //ぴ
        put( 0x3076  , false);  //ぶ
        put( 0x3077  , false);  //ぷ
        put( 0x3079  , false);  //べ
        put( 0x307A  , false);  //ぺ
        put( 0x307C  , false);  //ぼ
        put( 0x307D  , false);  //ぽ
        put( 0x30F4  , false);  //ヴ
        put( 0x30AC  , false);  //ガ
        put( 0x30AE  , false);  //ギ
        put( 0x30B0  , false);  //グ
        put( 0x30B2  , false);  //ゲ
        put( 0x30B4  , false);  //ゴ
        put( 0x30B6  , false);  //ザ
        put( 0x30B8  , false);  //ジ
        put( 0x30BA  , false);  //ズ
        put( 0x30BC  , false);  //ゼ
        put( 0x30BE  , false);  //ゾ
        put( 0x30C0  , false);  //ダ
        put( 0x30C2  , false);  //ヂ
        put( 0x30C5  , false);  //ヅ
        put( 0x30C7  , false);  //デ
        put( 0x30C9  , false);  //ド
        put( 0x30D0  , false);  //バ
        put( 0x30D1  , false);  //パ
        put( 0x30D3  , false);  //ビ
        put( 0x30D4  , false);  //ピ
        put( 0x30D6  , false);  //ブ
        put( 0x30D7  , false);  //プ
        put( 0x30D9  , false);  //ベ
        put( 0x30DA  , false);  //ペ
        put( 0x30DC  , false);  //ボ
        put( 0x30DD  , false);  //ポ
        put( 0x30F7  , false);  //ヷ
        put( 0x30F8  , false);  //ヸ
        put( 0x30F9  , false);  //ヹ
        put( 0x30FA  , false);  //ヺ
        put( 0x3046  , true );  //う
        put( 0x304B  , true );  //か
```

```
        put( 0x304D  , true );  //き
        put( 0x304F  , true );  //く
        put( 0x3051  , true );  //け
        put( 0x3053  , true );  //こ
        put( 0x3055  , true );  //さ
        put( 0x3057  , true );  //し
        put( 0x3059  , true );  //す
        put( 0x305B  , true );  //せ
        put( 0x305D  , true );  //そ
        put( 0x305F  , true );  //た
        put( 0x3061  , true );  //ち
        put( 0x3064  , true );  //つ
        put( 0x3066  , true );  //て
        put( 0x3068  , true );  //と
        put( 0x306F  , true );  //は
        put( 0x3072  , true );  //ひ
        put( 0x3075  , true );  //ふ
        put( 0x3078  , true );  //へ
        put( 0x307B  , true );  //ほ
        put( 0x30A6  , true );  //ウ
        put( 0x30AB  , true );  //カ
        put( 0x30AD  , true );  //キ
        put( 0x30AF  , true );  //ク
        put( 0x30B1  , true );  //ケ
        put( 0x30B3  , true );  //コ
        put( 0x30B5  , true );  //サ
        put( 0x30B7  , true );  //シ
        put( 0x30B9  , true );  //ス
        put( 0x30BB  , true );  //セ
        put( 0x30BD  , true );  //ソ
        put( 0x30BF  , true );  //タ
        put( 0x30C1  , true );  //チ
        put( 0x30C4  , true );  //ツ
        put( 0x30C6  , true );  //テ
        put( 0x30C8  , true );  //ト
        put( 0x30CF  , true );  //ハ
        put( 0x30D2  , true );  //ヒ
        put( 0x30D5  , true );  //フ
        put( 0x30D8  , true );  //ヘ
        put( 0x30DB  , true );  //ホ
        put( 0x30EF  , true );  //ワ
        put( 0x30F0  , true );  //ヰ
        put( 0x30F1  , true );  //ヱ
        put( 0x30F2  , true );  //ヲ
    }};

    private static Map<Integer, Integer> mapHalf2FullKana = new
HashMap<Integer, Integer>()
    {{
        put( 0xFF9E , 0x309B ); // Voicemark
        put( 0xFF9F , 0x309C ); // Voicemark
```

```
        put( 0xFFE9 , 0x2190 ); // ←
        put( 0xFFEA , 0x2191 ); // ↑
        put( 0xFFEB , 0x2192 ); // →
        put( 0xFFEC , 0x2193 ); // ↓
        put( 0xFFE8 , 0x2502 ); // │
        put( 0xFFED , 0x25A0 ); // ■
        put( 0xFFEE , 0x25CB ); // ○
        put( 0x0020 , 0x3000 ); // 　
        put( 0xFF64 , 0x3001 ); // 、
        put( 0xFF61 , 0x3002 ); // 。
        put( 0xFF62 , 0x300C ); // 「
        put( 0xFF63 , 0x300D ); // 」
        put( 0x003D , 0x30A0 ); // ゠
        put( 0xFF67 , 0x30A1 ); //ァ
        put( 0xFF71 , 0x30A2 ); //ア
        put( 0xFF68 , 0x30A3 ); //ィ
        put( 0xFF72 , 0x30A4 ); //イ
        put( 0xFF69 , 0x30A5 ); //ゥ
        put( 0xFF73 , 0x30A6 ); //ウ
        put( 0xFF6A , 0x30A7 ); //ェ
        put( 0xFF74 , 0x30A8 ); //エ
        put( 0xFF6B , 0x30A9 ); //ォ
        put( 0xFF75 , 0x30AA ); //オ
        put( 0xFF76 , 0x30AB ); //カ
        put( 0xFF77 , 0x30AD ); //キ
        put( 0xFF78 , 0x30AF ); //ク
        put( 0xFF79 , 0x30B1 ); //ケ
        put( 0xFF7A , 0x30B3 ); //コ
        put( 0xFF7B , 0x30B5 ); //サ
        put( 0xFF7C , 0x30B7 ); //シ
        put( 0xFF7D , 0x30B9 ); //ス
        put( 0xFF7E , 0x30BB ); //セ
        put( 0xFF7F , 0x30BD ); //ソ
        put( 0xFF80 , 0x30BF ); //タ
        put( 0xFF81 , 0x30C1 ); //チ
        put( 0xFF6F , 0x30C3 ); //ッ
        put( 0xFF82 , 0x30C4 ); //ツ
        put( 0xFF83 , 0x30C6 ); //テ
        put( 0xFF84 , 0x30C8 ); //ト
        put( 0xFF85 , 0x30CA ); //ナ
        put( 0xFF86 , 0x30CB ); //ニ
        put( 0xFF87 , 0x30CC ); //ヌ
        put( 0xFF88 , 0x30CD ); //ネ
        put( 0xFF89 , 0x30CE ); //ノ
        put( 0xFF8A , 0x30CF ); //ハ
        put( 0xFF8B , 0x30D2 ); //ヒ
        put( 0xFF8C , 0x30D5 ); //フ
        put( 0xFF8D , 0x30D8 ); //ヘ
        put( 0xFF8E , 0x30DB ); //ホ
        put( 0xFF8F , 0x30DE ); //マ
        put( 0xFF90 , 0x30DF ); //ミ
```

```
        put( 0xFF91 , 0x30E0 ); //ム
        put( 0xFF92 , 0x30E1 ); //メ
        put( 0xFF93 , 0x30E2 ); //モ
        put( 0xFF6C , 0x30E3 ); //ャ
        put( 0xFF94 , 0x30E4 ); //ヤ
        put( 0xFF6D , 0x30E5 ); //ュ
        put( 0xFF95 , 0x30E6 ); //ユ
        put( 0xFF6E , 0x30E7 ); //ョ
        put( 0xFF96 , 0x30E8 ); //ヨ
        put( 0xFF97 , 0x30E9 ); //ラ
        put( 0xFF98 , 0x30EA ); //リ
        put( 0xFF99 , 0x30EB ); //ル
        put( 0xFF9A , 0x30EC ); //レ
        put( 0xFF9B , 0x30ED ); //ロ
        put( 0xFF9C , 0x30EE ); //ヮ
        put( 0xFF9C , 0x30EF ); //ワ
        put( 0xFF66 , 0x30F2 ); //ヲ
        put( 0xFF9D , 0x30F3 ); //ン
        put( 0xFF65 , 0x30FB ); // □
        put( 0xFF70 , 0x30FC ); // □
    }};


    private static Map<Integer, Integer> map309C = new HashMap<Integer,
Integer>()
    {{
        put( 0x306F , 0x3071 ); //ぱ
        put( 0x3072 , 0x3074 ); //ぴ
        put( 0x3075 , 0x3077 ); //ぷ
        put( 0x3078 , 0x307A ); //ぺ
        put( 0x307B , 0x307D ); //ぽ
        put( 0x30CF , 0x30D1 ); //パ
        put( 0x30D2 , 0x30D4 ); //ピ
        put( 0x30D5 , 0x30D7 ); //プ
        put( 0x30D8 , 0x30DA ); //ペ
        put( 0x30DB , 0x30DD ); //ポ
    }};


    private static Map<Integer, Integer> map309B = new HashMap<Integer,
Integer>()
    {{
        put( 0x309D , 0x309E ); //ゝ
        put( 0x30FD , 0x30FE ); //ヽ
        put( 0x3046 , 0x3094 ); //ゔ
        put( 0x304B , 0x304C ); //が
        put( 0x304D , 0x304E ); //ぎ
        put( 0x304F , 0x3050 ); //ぐ
        put( 0x3051 , 0x3052 ); //げ
        put( 0x3053 , 0x3054 ); //ご
        put( 0x3055 , 0x3056 ); //ざ
```

```
        put( 0x3057 , 0x3058 ); //じ
        put( 0x3059 , 0x305A ); //ず
        put( 0x305B , 0x305C ); //ぜ
        put( 0x305D , 0x305E ); //ぞ
        put( 0x305F , 0x3060 ); //だ
        put( 0x3061 , 0x3062 ); //ぢ
        put( 0x3064 , 0x3065 ); //づ
        put( 0x3066 , 0x3067 ); //で
        put( 0x3068 , 0x3069 ); //ど
        put( 0x306F , 0x3070 ); //ば
        put( 0x3072 , 0x3073 ); //び
        put( 0x3075 , 0x3076 ); //ぶ
        put( 0x3078 , 0x3079 ); //べ
        put( 0x307B , 0x307C ); //ぼ
        put( 0x30A6 , 0x30F4 ); //ヴ
        put( 0x30AB , 0x30AC ); //ガ
        put( 0x30AD , 0x30AE ); //ギ
        put( 0x30AF , 0x30B0 ); //グ
        put( 0x30B1 , 0x30B2 ); //ゲ
        put( 0x30B3 , 0x30B4 ); //ゴ
        put( 0x30B5 , 0x30B6 ); //ザ
        put( 0x30B7 , 0x30B8 ); //ジ
        put( 0x30B9 , 0x30BA ); //ズ
        put( 0x30BB , 0x30BC ); //ゼ
        put( 0x30BD , 0x30BE ); //ゾ
        put( 0x30BF , 0x30C0 ); //ダ
        put( 0x30C1 , 0x30C2 ); //ヂ
        put( 0x30C4 , 0x30C5 ); //ヅ
        put( 0x30C6 , 0x30C7 ); //デ
        put( 0x30C8 , 0x30C9 ); //ド
        put( 0x30CF , 0x30D0 ); //バ
        put( 0x30D2 , 0x30D3 ); //ビ
        put( 0x30D5 , 0x30D6 ); //ブ
        put( 0x30D8 , 0x30D9 ); //ベ
        put( 0x30DB , 0x30DC ); //ボ
        put( 0x30EF , 0x30F7 ); //ヷ
        put( 0x30F0 , 0x30F8 ); //ヸ
        put( 0x30F1 , 0x30F9 ); //ヹ
        put( 0x30F2 , 0x30FA ); //ヺ
    }};
}
```

デモで使う表示メソッドを分離。

MyDemoMethod.java

```
package myKanjiDemos;

import java.util.EnumSet;
import java.util.List;
```

```
import myKanjiDemos.MyStringToChars.CharsNormalize;

public class MyDemoMethod {

    public static void charDisp(String str, EnumSet<CharsNormalize>
normalize)
    {
        List<List<Integer>> chars =
MyStringToChars.parseToCodepoints(str, normalize);
        String rebuildStr = MyStringToChars.makeStringFromList(chars);

        System.out.println("---------------------------------");
        System.out.println("Str         : " + str);
        System.out.println("StrLen      : " + str.length());
        System.out.println("RebuildStr  : " + rebuildStr);
        System.out.println("RebuildLen  : " + rebuildStr.length());
        System.out.println("parseOption : " + normalize);

        StringBuilder sbChars = new StringBuilder();
        StringBuilder sbCodeStr = new StringBuilder();

        for(int pos=0; pos<chars.size(); pos++)
        {
            sbChars.setLength(0);
            sbCodeStr.setLength(0);

            for(int idx=0; idx<chars.get(pos).size(); idx++)
            {
                sbChars.appendCodePoint(chars.get(pos).get(idx));
                if (idx != 0) sbCodeStr.append( ", ");
                sbCodeStr.append( "u+" +
Integer.toHexString(chars.get(pos).get(idx)) );
            }

            System.out.printf("Pos %7d : %s %s\n", pos,
sbChars.toString(), sbCodeStr.toString());
        }
    }
}
```