SJIS長換算の話

2024-11-29
入力した文字列をShift-JIS換算のバイト長にする際の話。

概要

ホストコンピュータで処理するために、テキストファイルをShift_JISに変換する作業がしばしばあります。
1行が固定長のファイルに格納してホストへ送るのだけれど、変換したテキストがレコード長を越えていたら切り詰める等の処理が必要になるので、テキストのShift_JIS換算時バイト長を得られる事が必須になります。
※文字数じゃないんです、バイト数なんですよ。それにホスト（メインフレーム）ではSJIS利用している所が多いです

環境

Windows11上でJava11利用

確認用コード

SJISLength.java

```java
package sjisLength;

import java.io.UnsupportedEncodingException;

public class SJISLength {

    public int getStrLen1(String str)  {
        try {
            return str.getBytes("MS932").length;
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return -1;
    }

    public int getStrLen2(String str) {
        int len = 0;
        int cp;

        for(int idx=0; idx <str.length(); idx++) {
            cp = str.codePointAt(idx);
```

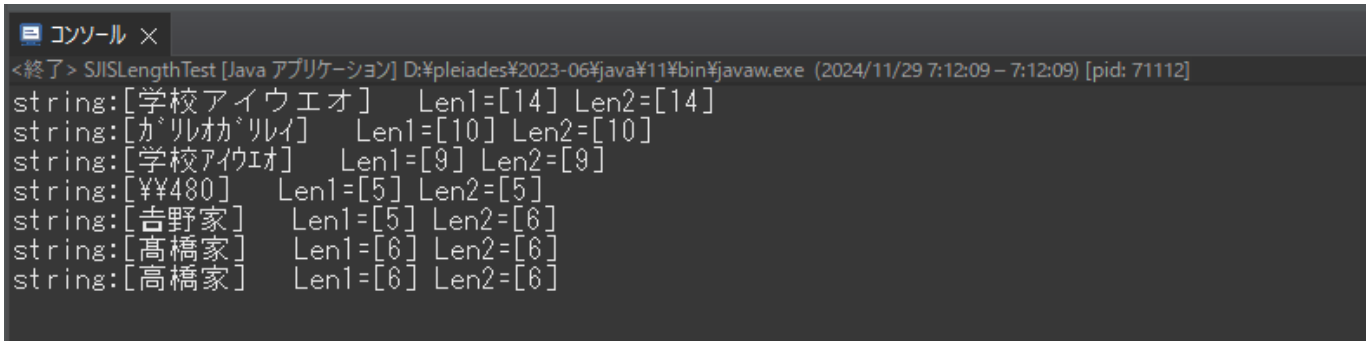
```
            len += ((0x20 <= cp)&&(cp <= 0x7E)) ? 1 : ((0xFF60 <=
cp)&&(cp <= 0xFFFE)) ? 1 : 2;
            if (cp > 65535) idx++;
        }

        return len;
    }
}
```

テストのコードはこれ。

SJISLengthTest.java

```
package sjisLength;

public class SJISLengthTest {

    public static void main(String[] args) {
        SJISLength st = new SJISLength();

        String str1 = "学校アイウエオ";
        String str2 = "ｶﾞﾘﾚｵｶﾞﾘﾚｲ"          ;
        String str3 = "学校ｱｲｳｴｵ"     ;
        String str4 = "\\\\480";
        String str5 = "    野家"
        String str6 = "髙橋家";
        String str7 = "髙橋家";

        System.out.println(String.format("string:[%s]   Len1=[%d]
Len2=[%d]", str1, st.getStrLen1(str1), st.getStrLen2(str1)));
        System.out.println(String.format("string:[%s]   Len1=[%d]
Len2=[%d]", str2, st.getStrLen1(str2), st.getStrLen2(str2)));
        System.out.println(String.format("string:[%s]   Len1=[%d]
Len2=[%d]", str3, st.getStrLen1(str3), st.getStrLen2(str3)));
        System.out.println(String.format("string:[%s]   Len1=[%d]
Len2=[%d]", str4, st.getStrLen1(str4), st.getStrLen2(str4)));
        System.out.println(String.format("string:[%s]   Len1=[%d]
Len2=[%d]", str5, st.getStrLen1(str5), st.getStrLen2(str5)));
        System.out.println(String.format("string:[%s]   Len1=[%d]
Len2=[%d]", str6, st.getStrLen1(str6), st.getStrLen2(str6)));
        System.out.println(String.format("string:[%s]   Len1=[%d]
Len2=[%d]", str7, st.getStrLen1(str7), st.getStrLen2(str7)));
    }

}
```

実行結果



“　　野家の結果だけ異なっている。

説明

ざっくりいうと、漢字は全角文字で2バイト、半角文字は1バイト、としてカウントしたい、という事。

getStrLen1()

メソッド getStrLen1() はエンコーディングに“ MS932”を指定して(現行のWindowsで言うところのCP932,Windows-31J)バイト列に換算し、その要素数を返します※“Shift_JIS”とは異なる箇所があります
Stringクラスの getBytes()メソッドを利用しているのでお手軽です。

ただし“MS932”で変換できない文字は問題を起こします。この例だと“　　野家の“　　が該当する箇所です。“　　野家と“ 髙橋家 ”のバイト換算結果を見てみます。

```
package sjisLength;

import java.io.UnsupportedEncodingException;

public class Hexdump {

    public static void main(String[] args) {
        String str1 = "    野家"
        String str2 = "髙橋家";

        HexDump(str1, getByte(str1));
        HexDump(str2, getByte(str2));
    }

    public static byte[] getByte(String str)  {
        try {
            return str.getBytes("MS932");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return new byte[0];
```

```
    }

    public static void HexDump(String cap, byte[] buff) {
        System.out.println(String.format("=== [%s] ===", cap));

        for(int idx = 0; idx < buff.length; idx++) {
            System.out.println(String.format("buff[%02d] : %02x", idx,
buff[idx]));
        }
    }
}
```

結果はこちら。

```
コンソール ×
<終了> Hexdump [Java アプリケーション] D:¥pleiades¥2023-06¥java¥11¥bin¥javaw.exe (2024/11/29 7:58:17 – 7:58:17) [pid: 74816]
=== [吉野家] ===
buff[00] : 3f
buff[01] : 96
buff[02] : ec
buff[03] : 89
buff[04] : c6
=== [高橋家] ===
buff[00] : 8d
buff[01] : 82
buff[02] : 8b
buff[03] : b4
buff[04] : 89
buff[05] : c6
```

“　　野家の変換結果の1バイト目ですが0x3f は疑問符 “？” の文字コードです。つまりString.getBytes(“MS932”)は “　　の変換に失敗してこの文字に置換えたという事ですね。
0x96ec, 0x89c6 はそれぞれ “野”, “家” の文字コードになります。つまり、漢字3文字中1文字だけ半角文字になってしまったという事。

書き込む領域のサイズが決まっていて、テキストがその領域サイズを越えたか否かの判定だけするのであればこれでも問題無いです。長くなってしまう訳では無いので。

getStrLen2()

メソッド getStrLen2() は1文字ずつコードポイントを見て、半角とされる文字だったら長さ1、そうでなければ長さ2、としてカウントアップしています。
コードポイントが U+0020 ～ U+007E の範囲と、U+FF60 ～ U+FFFE の範囲を、半角文字としています。
65535を超えたコードポイントの時は、その文字がサロゲートペアで表現されている事になるので、下位サロゲートを読み込まないようにしています※Javaは内部表現にUTF-16を利用しているので

半角文字の範囲を外れるものはとりあえず2バイトとしています。なので “　　であっても問題ありません。

メソッド getStrLen1() と比べ、正確（？）にサイズを導き出せているのですが、特定の文字が入ってくると問題を起こすかもしれません。※入力仕様と決め方の問題なのですが…

以下のような例があります。

U+00A5	¥	YEN SIGN 普通は U+005C が来るんじゃないかな
U+00A3	£	POUND SIGN CP1013だと 0x23□CP932の“＃“とコードが重なる
U+20A9	₩	WON SIGN CP949だと 0x5C □CP932の”＼”とコードが重なる

技術資料, SJIS, CP932, CP949, CP1013, Shift JIS, Windows-31J, java