SJC-P 書式

[Java][SJC-P]
http://docs.oracle.com/javase/jp/6/api/java/util/Formatter.html

・SJC-P 日付と数値

構文 (一般)

%[argument_index$][flags][width][.precision]conversion

項目	必須	内容
argument_index	no	引数の位置。最初の引数は「1$」。
flags	no	出力書式を変更する文字のセット。有効なフラグのセットは、変換によって異なる。
width	no	出力に書き込む最小文字数。
precision	no	文字数を制限するために通常使用される。動作は、変換によって異なる。
conversion	yes	引数を書式設定する方法。有効な変換は、データ型によって異なる。

構文 (日付および時刻表現)

%[argument_index$][flags][width]conversion
conversion は、2 つの文字シーケンスで、最初の文字は「t」または「T」。2 番目の文字は使用する書式を示す。

変換のカテゴリ

1. 一般 - 任意
2. 文字 - Unicode 文字を表す基本型 char、Character、byte、Byte、short、Short、Character.isValidCodePoint(int) が true を返す int および Integer。
3. 数値
    1. 整数 - byte、Byte、short、Short、int、Integer、long、Long、および BigInteger などの Java 整数型に適用される
    2. 浮動小数点 - float、Float、double、Double、および BigDecimal などの Java 浮動小数点型に適用される
4. 日付 / 時刻 - long、Long、Calendar、および Date など、日付または時刻のエンコーディングが可能な Java 型に適用される
5. パーセント - リテラル「%」(\u0025) を生成する
6. 行区切り文字 - プラットフォーム固有の行区切り文字を生成する

変換	引数のカテゴリ	説明

b、 B	一般	null → false。boolean or Boolean → String.valueOf() により返される文字列。そうでない場合 → true
h、H	一般	null → null。そうでない場合 → Integer.toHexString(arg.hashCode()) の呼び出
s、 S	一般	null → null。Formattable を実装する場合 → arg.formatTo が呼び出される。そうでない場合、結果は arg.toString() の呼び出しで取得。
c、C	文字	Unicode 文字。
'd'	整数	10 進整数。
'o'	整数	8 進整数。
'x'、'X'	整数	16 進整数。
'e'、'E'	浮動小数点	浮動小数点表示形式の 10 進数。
'f'	浮動小数点	10 進数。
g、 G	浮動小数点	四捨五入処理後の精度および値に応じて浮動小数点表示形式または 10 進数書式。
a、A	浮動小数点	有効数字および指数を持つ浮動小数点数。
t、T	日付 / 時刻	日付および時刻変換文字用の接頭辞。。
'%'	パーセント	リテラル「%」(\u0025)。
'n'	行区切り文字	プラットフォーム固有の行区切り文字。

時刻の書式設定

変換	説明
'H'	24 時間制の時 (00 - 23)
'I'	12 時間制の時 (00 - 12)
'k'	24 時間制の時 (0 - 23)
'l'	12 時間制の時 (1 - 12)
'M'	分 (00 - 59)
'S'	秒 (00 - 60)「60」はうるう年での秒のサポートに必要な特殊な値。

'L'	ミリ秒 (000 - 999)
'N'	ナノ秒 (000000000 - 999999999)。
'p'	午前または午後を表すロケール固有の小文字のマーカ (例、「am」や「pm」)。
'z'	RFC 822 に準拠した、GMT からの数値タイムゾーンオフセット (例、-0800)
'Z'	タイムゾーンの省略形を表す文字列。Formatter のロケールは、引数のロケール (存在する場合) よりも優先。
's'	1970 年 1 月 1 日 00:00:00 UTC のエポック開始からの秒 (Long.MIN_VALUE/1000 から Long.MAX_VALUE/1000 まで)
'Q'	1970 年 1 月 1 日 00:00:00 UTC のエポック開始からのミリ秒 (Long.MIN_VALUE から Long.MAX_VALUE まで)

日付の書式設定

変換	説明
'B'	ロケール固有の月の完全名 (例、「January」、「February」)
'b'	ロケール固有の月の省略名 (例、「Jan」、「Feb」)
'h'	'b' と同じ
'A'	ロケール固有の曜日の完全名 (例、「Sunday」、「Monday」)
'a'	ロケール固有の曜日の短縮名 (例、「Sun」、「Mon」)
'C'	4 桁の年を 100 で割った値 (00 - 99)。
'Y'	年
'y'	年の下 2 桁 (00 - 99)。
'j'	年の何日目かを表す日、グレゴリオ歴の場合、001 - 366。
'm'	月 (01 - 13)。
'd'	月の何日目かを表す日 (01 -31)。
'e'	月の何日目かを表す日 (1 - 31)。

一般の日付 / 時刻変換の書式設定

変換	説明
'R'	「%tH:%tM」24 時間制で書式設定された時刻
'T'	「%tH:%tM:%tS」24 時間制で書式設定された時刻
'r'	「%tI:%tM:%tS %Tp」12 時間制で書式設定された時刻
'D'	「%tm/%td/%ty」
'F'	「%tY-%tm-%td」ISO 8601 に準拠した日付
'c'	「%ta %tb %td %tT %tZ %tY」 (例、「Sun Jul 20 16:17:00 EDT 1969」)

フラグ

サポートされるフラグの概要を示します。y は、指定された引数型でフラグがサポートされることを意味します。

フラグ	一般	文字	整数	浮動小数点	日付 / 時刻	説明
'-'	y	y	y	y	y	左揃え
'#'	y ※ 1	-	y ※ 3	y	-	変換に依存する代替フォームを使用する必要がある。
'+'	-	-	y ※ 4	y	-	常に符号。
' '	-	-	y ※ 4	y	-	先頭に正の値を示す空白。
'0'	-	-	y	y	-	ゼロが追加。
','	-	-	y ※ 2	y ※ 5	-	ロケール固有のグループ化区切り文字。
'('	-	-	y ※ 4	y ※ 5	-	負の数値を括弧で囲む。

- ※ 1 Formattable の定義に依存する
- ※ 2 'd' 変換のみ
- ※ 3 'o'、'x'、および 'X' 変換のみ
- ※ 4 'd'、'o'、'x'、および 'X' 変換が BigInteger に適用されるか、'd' が byte、Byte、short、Short、int、Integer、long、および Long に適用される場合
- ※ 5 'e'、'E'、'f'、'g'、および 'G' 変換のみ

■例

```
import java.util.Date;
import java.util.Formattable;
```

```
import java.util.Formatter;

public class FormatTest1 {
    public static void main(String[] args) {
        FormatTest1 me = new FormatTest1();

        System.out.format("%b%n", true);           // true
        System.out.format("%B%n", true);           // TRUE
        System.out.format("%h%n", me.new Demo1()); // 999
        System.out.format("%s%n", me.new Demo2()); // ### sjcp.format.FormatTest1$Demo2 ###
        System.out.format("%c%n", 0x41);           // A
        System.out.format("%d%n", 0xff);           // 255
        System.out.format("%o%n", 0xff);           // 377
        System.out.format("%X%n", 0xff);           // FF
        System.out.format("%e%n", 0.0001d);        // 1.000000e-04
        System.out.format("%f%n", 0.0001d);        // 0.000100
        System.out.format("%#.2f%n",123.123d);     // 123.12
        System.out.format("%g%n", 0.0001d);        // 0.000100000
        System.out.format("%a%n", 0.0001d);        // 0x1.a36e2eb1c432dp-14

        Date dt = new Date();
        System.out.format("%tH 時%n", dt);         // 16 時
        System.out.format("%tI 時%n", dt);         // 04 時
        System.out.format("%tk 時%n", dt);         // 16 時
        System.out.format("%tl 時%n", dt);         // 4 時
        System.out.format("%tM 分%n", dt);         // 32 分
        System.out.format("%tS 秒%n", dt);         // 14 秒
        System.out.format("%tL ミリ秒%n", dt);     // 186 ミリ秒
        System.out.format("%tN ナノ秒%n", dt);     // 186000000 ナノ秒

        System.out.format("%tp%n", dt);            // 午後
        System.out.format("%tz%n", dt);            // +0900
        System.out.format("%tZ%n", dt);            // JST
        System.out.format("%ts%n", dt);            // 1175844734
        System.out.format("%tQ%n", dt);            // 1175844734186

        System.out.format("%tB%n", dt);            // 4 月
        System.out.format("%tb%n", dt);            // 4
        System.out.format("%tA%n", dt);            // 金曜日
        System.out.format("%ta%n", dt);            // 金
        System.out.format("%tC%n", dt);            // 20
        System.out.format("%tY%n", dt);            // 2007
        System.out.format("%tj%n", dt);            // 096
        System.out.format("%tm%n", dt);            // 04
        System.out.format("%td%n", dt);            // 06
        System.out.format("%te%n", dt);            // 6

        System.out.format("%tR%n", dt);            // 16:32
        System.out.format("%tT%n", dt);            // 16:32:14
        System.out.format("%tr%n", dt);            // 04:32:14 午後
        System.out.format("%tD%n", dt);            // 04/06/07
        System.out.format("%tF%n", dt);            // 2007-04-06
        System.out.format("%tc%n", dt);            // 金 4 06 16:32:14 JST 2007
    }
    private class Demo1 {
        public int hashCode() {
            return 0x999;
        }
    }
    private class Demo2 implements Formattable {
        public void formatTo(Formatter formatter, int flags, int width,
                int precision) {
            formatter.format("### %s ###", this.getClass().getName());
        }
    }
}
```

■日付の場合

・いつも忘れてしばらく悩むが、書式設定の数分、Date オブジェクトを引数として渡してあげる必要がある

```
Date now = new Date();
String filename =  String.format("/%tY%tm%td%tH%tM%tS.txt", now, now, now, now, now, now);
```