3-14.printfメソッド

printfメソッド

はじめに

これまでは `System.out.println()` や `System.out.print()` を使用していました。
これらは単純に値を出力するためのメソッドでしたね。
しかし、 `System.out.printf()` メソッドを使用すれば出力する文字などに **様々な装飾** を施すことが可能です。

※`System.out.print()`と同じように、`System.out.printf()`メソッドも改行はされずに出力される点は注意しておいてください。

Step1: System.out.printf()と書式指定子

文字や数値へ装飾を施すためには **書式指定子** というものを記述する必要があります。
簡単にいうと「文字の好きな部分をちょっとだけ変える記号」です。

書式指定子は多くの種類が存在するため、順を追って少しずつ見ていきましょう。
（頻出するものを抜粋しています）

【型を指定する書式指定子】

指定子	指定子を付与することで得られる効果
%d	10進整数表記になる
%o	8進整数表記になる
%x	16進整数表記になる
%f	浮動小数点数表記になる

【その他の書式指定子】

指定子	指定子を付与することで得られる効果
-	左寄せで出力される（幅指定必須）
+	符号を出力する
0	前に0を詰める（幅指定必須）

なお、書式指定子のほかに、フォーマット指定子、出力変換指定子などの呼び方があります。
そのため、以降、本項では書式指定子を「指定子」と呼称します。

具体的な使用例を見てみましょう。

> **例)** int型整数「12345」を8進数表記に変更したい場合

まず、下記変数を準備します。

```
int math = 12345;
```

このint型変数mathの中身「12345」を8進数に変換してみましょう。

```
System.out.printf("%o", math);
```

【出力結果】

```
30071
```

整数「12345」が8進数表記に変換され、「30071」と出力されました。

続けてもうひとつ例を見ていきましょう。

> **例）int型整数「1」の前に0を付けたい場合（0詰めして表示したい場合）**

先ほどの例と同じように、変数oneを準備し、「1」を「01」に変換してみましょう。

```
int one = 1;
System.out.printf("%02d", one);
```

%dは10進数表記の指定子です。データ型で言えばint型の値に対応します。
この指定子の間の数値「02」は出力される数値の桁（けた）とゼロ詰めであることの2つを表しています。

【出力結果】

```
01
```

今回は数値の「1」を「0詰め」かつ「2桁で表示」させるので、結果は「01」となります。
ただし、数値が「10」の場合は、ゼロ詰めが適用されず、そのまま「10」が表示されますので注意してください。

この記述を使用すれば、

- 1月～12月を表示(ただし1～9は0詰めして表示させる)

といった要件にも簡単に対応することが可能です。

【サンプルコード】

```
public class monthTest {
    public static void main(String[] args) {
        for(int i = 0; i <= 12; i++) {
            if(i == 0) {
                continue;
            }
            System.out.printf("%02d\n", i);
        }
    }
}
```

【出力結果1】

```
01
02
03
04
05
06
07
08
09
10
11
12
```

"%02" と指定したので1～9は0詰めで表示されましたが、10～12は0詰めされずにそのまま表示されました。

もし"%03"と記述すれば、出力結果は以下のようになります。

【出力結果2（"%03"と記述した場合）】

```
001
002
003
004
005
006
007
008
009
010
011
012
```

"%03"は第二引数を3文字で出力するため、3文字に満たない場合は不足分を0詰めされて上記のような結果となります。
この性質を覚えておくと、書式指定子への理解がグッと深まるでしょう。

Step2: System.out.printf()による日時の出力

printf()の指定子を用いることで、日時データのフォーマットを変換して出力することが可能です。
下記の表は、よく日時データの変換に用いられる指定子を抜粋したものです。

指定子	指定子を付与することで得られる効果
tH, TH	時間　(00 - 23)　0 が付加される
tI, TI (大文字のアイ)	時間　(01 - 12)　0 が付加される
tk, Tk	時間　(0 - 23)　0 は付加されない
tl, Tl (小文字のエル)	時間　(1 - 12)　0 は付加されない
tM, TM	分　(00 - 59)　0 が付加される
tS, TS	秒　(00 - 60)　0 が付加される
tp, Tp	午前・午後の表記
tB, TB	月の表示
tb, Tb	月表示の省略形
tA, TA	曜日の表示
ta, Ta	曜日の省略形
tY, TY	年の4桁　0 が付加される
ty, Ty	年の下2桁　(00 - 99)　0 が付加される
tm, Tm	月　(01 - 13)　0 が付加される（13は太陰暦を使用する場合に使う）
td, Td	日　(01 - 31)　0 が付加される
te, Te	日　(1 - 31)　0 は付加されない

これらの指定子を用いれば、取得した現在時刻を私たちが普段目にするフォーマットへ変換するといったことも
簡単に行えます。

まず、現在時刻を取得します。

```
Date date = new Date();
```

Date型変数「Date」の出力結果
※2019/03/06日時点の出力内容になります。

```
Wed Mar 06 16:39:32 JST 2019
```

では、取得した現在時刻の形式を「 **年月日(曜日)午前or午後〇〇時〇〇分** 」に変換してみましょう。

```
System.out.printf("%tY年%<tB%<te日(%<ta)%<tp%<tI時%<tM分", date);
```

このprintf()内では、下記の指定子が記述されているので要チェックです。

指定子	指定子を付与することで得られる効果
tY, TY	年の4桁　0 が付加される
tB, TB	月の表示
te, Te	日　(1 - 31)　0 は付加されない
ta, Ta	曜日の省略形
tp, Tp	午前・午後の表記
tI, TI (大文字のアイ)	時間　(01 - 12)　0 が付加される
tM, TM	分　(00 - 59)　0 が付加される

【出力結果（`printf()`で「date」のフォーマットを変換した場合）】

```
2018年12月29日(土)午後04時56分
```

%の後ろの<は何？

これは、インデックスの相対指定というやり方になります。
https://docs.oracle.com/javase/jp/8/docs/api/java/util/Formatter.html
(「引数のインデックス」の部分)

https://teratail.com/questions/86385

printfで複数の指定子を書くと、その後ろにある引数を順番に読み取ってくれます

```
int a1 = 1, a2 = 2, a3 = 3;
System.out.printf("%d、%d、%d", a1, a2, a3);
//→1、2、3
```

しかし今回のコードだと、指定子がいくつもあるのに対し、後ろの引数が`date`1個しかありません。

```
System.out.printf("%tY年%<tB%<te日(%<ta)%<tp%<tI時%<tM分", date);
```

`%>`と書くことで、直前の参照場所と同じところを参照できるようになります。

最初の`%tY年`だけは、`>`が入ってませんね。
これは、1つ目の指定子なので、自動的に1つ目の`date`を参照しているからです。

あとのものは、同じ`date`を参照したいから相対指定、ということですね。
また、何番目の引数を参照する、というのを`1$`、`2$`などで明示的に指定することもできます(絶対指定)。
なので、以下のコードでも同じ出力になります。

```
System.out.printf("%1$tY年%1$tB%1$te日(%1$ta)%1$tp%1$tI時%1$tM分", date);
```

まとめ

現場での業務において、値の形式を変更したいケースに遭遇することは非常に多いです。

日時の形式は現場によって千差万別なので、前回学んだ`SimpleDateFormat`と併せて覚えておくと、様々な要件へ柔軟に対応できるようになります。
今のうちにしっかりと学習しておくと良いでしょう。

課題

提出課題はありませんので、一通り学習が終わったら次の章に進んで下さい。

最終更新日時: 2022年 08月 21日(日曜日) 17:04