

2回目 “ようこそJ a v aへ”

■ 今日の講義で学ぶ内容 ■

- 画面へのメッセージの表示
- 文字や文字列、数値を表現するリテラル
- 制御コードを表すエスケープシーケンス

画面出力の基本形

ソースファイル名：クラス名.java

```
class クラス名
{
    public static void main(String[] args)
    {
        System.out.println("ここに出力したい文字列 1 行目");
        System.out.println("ここに出力したい文字列 2 行目");
        :
    }
}
```

ソースコード例

ソースファイル名：Sample2_1.java

```
// 画面に文字列を出力するコード
class Sample2_1
{
    public static void main(String[] args)
    {
        System.out.println("ようこそJ a v aへ！");
        System.out.println("J a v aをはじめましょう！");
    }
}
```

実行画面

```
>java Sample2_1
ようこそJ a v aへ！
J a v aをはじめましょう！
```

いろいろな出力方法

1. `System.out.println("ここに出力したい文字列");`

ここに出力したい文字列 が画面に表示された後、行末に**改行が挿入されます**

2. `System.out.print("ここに出力したい文字列");`

ここに出力したい文字列 が画面に表示された後、行末に**改行が挿入されません**

3. `System.out.printf("ここに出力したい文字列");`

ここに出力したい文字列 が画面に表示された後、行末に**改行が挿入されません**
C言語 `printf()` 関数と類似しており、**%n** により**改行が可能です**

4. `System.out.format("ここに出力したい文字列");`

`System.out.printf()` とまったく同じふるまいをします

ソースコード例

ソースファイル名 : `Sample2_2.java`

```
class Sample2_2
{
    public static void main(String[] args)
    {
        System.out.println("1. println()による出力（行末に改行あり）");
        System.out.print("2. print()による出力（行末に改行なし）");
        System.out.printf("3. printf()による出力%n");
        System.out.format("4. format()による出力%n");
    }
}
```

実行画面

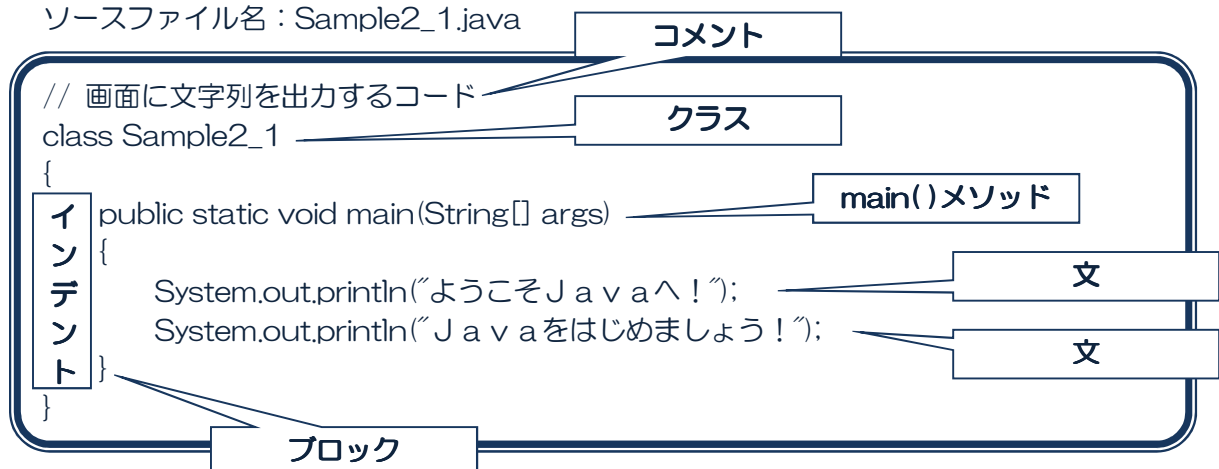
```
>java Sample2_2
1. println()による出力（行末に改行あり）
2. print()による出力（行末に改行なし）3. printf()による出力
4. format()による出力
```



コードの内容



ソースファイル名：Sample2_1.java



コメント

// (ダブルスラッシュ) で始まる行、
または /* */ で囲まれた行 (複数行でも可) です

- ・コンパイル時には無視されるコードです
- ・処理の内容などメモを記入しておくのに大変に便利です

クラス

キーワード **class** がついた { } (括弧) 内をいいます

- ・Java のソースコードは少なくとも1つの**クラス**から成ります



一般に Java のソースコードは複数の**クラス**をもちますが、
ここでは1つの**クラス**をもつ場合を主に解説していきます

ブロック

{ } (括弧) で囲まれた処理の集まり部分です

- ・キーワード **class** がつく { } と区別します

main()メソッド

public static void main(String[] args) で始まる**ブロック**です

- ・Java ではここからプログラムの処理が始まります

文

最後に ; (セミコロン) がついた個々の単一の処理や命令です

- ・**ブロック**は複数の**文**をもつことができます
- ・**文**は上から下へ順番に実行されます



文には単一の処理や命令の他に if **文**や for **文**などがあります
また、**文**には**ブロック**を置くこともできます
これについては if **文**の回で詳しく解説します

インデント

行頭での字下げです

- ・ソースコードを読みやすくします
- ・**ブロック**毎に**インデント**を付けると見やすくなります

リテラル

コード内の**値の表現**です
値には、文字や文字列、数値などがあります

表現する対象に応じて、**○○リテラル**と呼ばれます
たとえば、文字リテラル、文字列リテラルなどです

• 文字リテラル

'' (シングルクォート) で文字を囲み、**一文字を表現**します

たとえば、'A'、'b'、'c' などです

• 文字列リテラル

"" (ダブルクォート) で文字列を囲み、**文字列を表現**します

たとえば、"Hello"、"こんにちは" などです



"A" は大丈夫ですが、'Hello' はエラーになります
一文字は文字列の特別な場合と考えることができますので、
文字列リテラルで表現することができます
しかし、文字リテラルは一文字である必要があります

• 整数リテラル

整数をそのまま記述し、整数を表現します

たとえば、123、-23 などです

• 浮動小数点数リテラル

実数値をそのまま記述し、実数値を表現します

たとえば、3.14、-1.2、0.24 などです

• 論理値リテラル

true 又は **false** を記述し、論理の**真と偽を表現**します



この他、空リテラルがありますが、Java プログラミングⅡで詳しく解説します

• 空リテラル

null を記述し、**空の参照を表現**します

ソースコード例

ソースファイル名 : Sample2_3.java

```
class Sample2_3
{
    public static void main(String[] args)
    {
        System.out.println('A');
        System.out.println("Hello");
        System.out.println(123);
        System.out.println(0.24);
        System.out.println(true);
    }
}
```


実行画面

```
>java Sample2_3
A
Hello
123
0.24
true
```

エスケープシーケンス

エスケープシーケンス **¥ (円マーク)** をつけた 2 つの文字により表現される**一文字**です

たとえば、'¥n'、'¥t' などです

 エスケープシーケンスは一文字の表現であるため、文字リテラルで表現することができます

エスケープシーケンスは、**改行やタブなどの機能**を表現します


たとえば、`System.out.print("¥n");` で改行が行われます

次のような種類があります

表記	表記の機能、または意味
¥b	バックスペース
¥t	水平タブ
¥n	改行
¥f	改ページ
¥r	復帰
¥"	" ←ダブルクォート自身の表現に用います
¥'	' ←シングルクォート自身の表現に用います
¥¥	¥ ←円マーク自身の表現に用います

エスケープシーケンスは、
A や B などの一文字と同様に、文字列を構成する文字として
用いることができ、文字列リテラルに含めることができます

たとえば、`"Hello¥n"` です

 一文字は文字列の特別な場合と考えることができますので、
1 つのエスケープシーケンスを`"¥n"`のように文字列リテラルで
表現しても問題はありません

ソースコード例

ソースファイル名：Sample2_4.java

```
class Sample2_4
{
    public static void main(String[] args)
    {
        System.out.println("バックスペースします¥b バックスペースしました");
        System.out.println("水平タブいれます¥t 水平タブいれました");
        System.out.println("改行します¥n 改行しました");
        System.out.println("復帰します¥r 復帰しました");
        System.out.println("ダブルクォートを表示します");
        System.out.println("¥");
        System.out.println("円マークを表示します");
        System.out.println("¥¥");
    }
}
```

実行画面

```
>java Sample2_4
バックスペースしまバックスペースしました
水平タブいれます      水平タブいれました
改行します
改行しました
復帰しました
ダブルクォートを表示します
"
円マークを表示します
¥
```

整数リテラルと進数表現

整数リテラルを用いて整数を表現するとき、12 や -4 のように通常は10進数を用います
このほかに、**8進数**や**16進数**での**整数の表現が可能**です

10の8進数表現 012 **0で数値を始める → 8進数表現**とみなされます

たとえば、System.out.println(012); は 10 と出力されます


10の16進数表現 0xA **0xで数値を始める → 16進数表現**とみなされます

たとえば、System.out.println(0xA); は 10 と出力されます

 記号 **x** は**大文字でも小文字でもよい**です
0xA と OXA は同じ整数 10 を表現します

10の10進数表現 10 **上記以外 → 10進数表現**とみなされます

たとえば、System.out.println(10); は 10 と出力されます

 019 や 0x4g はエラーとなります
これは **8進数の各桁を0~7**までを用いて表現するため、
また **16進数の各桁は0~15**であり、アルファベットで
0~9、a~f（または大文字**A~F**でもよい）を用いて
表現するためです

ソースコード例

ソースファイル名：Sample2_5.java

```
class Sample2_5
{
    public static void main(String[] args)
    {
        System.out.print("10 -> ");
        System.out.print(10);
        System.out.print(", 012 -> ");
        System.out.print(012);
        System.out.print(", 0xa -> ");
        System.out.print(0xa);
    }
}
```


実行画面

```
>java Sample2_5  
10 -> 10, 012 -> 10, 0xa -> 10
```



浮動小数点数リテラルと指数表現



浮動小数点数リテラルを用いて実数を表現するとき、通常 3.14 や 0.2 のように書きます
しかし、非常に**大きな実数**や**小さな実数**には**指数表現**を用いると**大変に便利**です

たとえば、

1.2e+2 は 1.2×10^2 を表現します

System.out.println(1.2e+2); とすると、120.0 と出力されます

1.0e-2 は 1.0×10^{-2} を表現します

System.out.println(1.0e-2); とすると、0.01 と出力されます



記号 **e** は**大文字でも小文字でもよい**です

1.2e+2 と 1.2E+2 は同じ実数 120.0 を表現します

ソースコード例

ソースファイル名 : Sample2_6.java

```
class Sample2_6  
{  
    public static void main(String[] args)  
    {  
        System.out.print("1.2e+2 -> ");  
        System.out.print(1.2e+2);  
        System.out.print(", 1.0e-2 -> ");  
        System.out.print(1.0e-2);  
    }  
}
```

実行画面

```
>java Sample2_6  
1.2e+2 -> 120.0, 1.0e-2 -> 0.01
```

文字列リテラルと他のリテラルとの連結

文字列リテラルは、整数リテラルや文字リテラルなど他のリテラルと+により連結できます
連結を上手に用いると、画面出力をするコードがすっきり見やすくなります

たとえば、“9 の次は”+10 とすると、“9 の次は 10”となり連結されます



連結の記号+は演算子といいます

演算子+の詳しい解説は「演算子」の回で行います

ここでは演算子+と連結の機能を押さえておきましょう

ソースファイル名：Sample2_7.java

```
class Sample2_7
{
    public static void main(String[] args)
    {
        // 文字列リテラル“9 の次は”と整数リテラル 10 を連結します
        System.out.println("9 の次は"+10);

        // 文字列リテラル“πは”と浮動小数点数リテラル 3.14 を連結します
        System.out.println("πは"+3.14);

        // 文字列リテラル“A の小文字は”と文字リテラル'a'を連結します
        System.out.println("A の小文字は'+a");

        // 文字列リテラル“true の反対は”と論理値リテラル false を連結します
        System.out.println("true の反対は"+false);
    }
}
```

実行画面

```
>java Sample2_7
9 の次は 10
πは 3.14
A の小文字は a
true の反対は false
```