

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！
-- Press any key to exit (Input "c" to continue) --
```

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

ソースコード例

ソースファイル名: `Sample2_2.java`

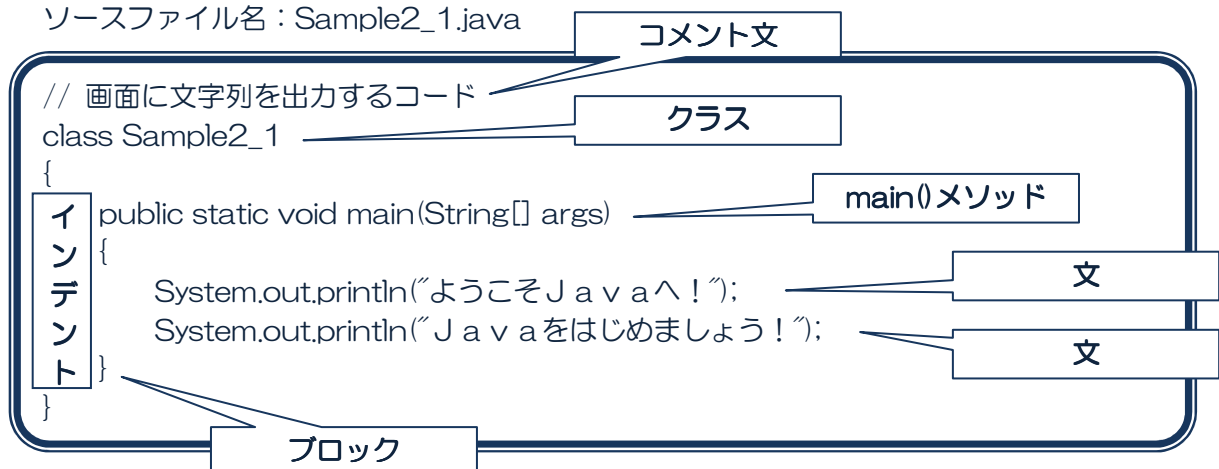
```
class Sample2_2
{
    public static void main(String[] args)
    {
        System.out.println("1. println()による出力（行末に改行あり）");
        System.out.print("2. print()による出力（行末に改行なし）");
        System.out.printf("3. printf()による出力%n");
    }
}
```

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
-- Press any key to exit (Input "c" to continue) --
```

コードの内容

ソースファイル名：Sample2_1.java



ブロック

{ } (括弧) で囲まれた処理の集まり部分

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

main()メソッド

「main」がついた**ブロック**

- ・ 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) がついた個々の処理

- ・ この他、**文** には **ブロック** を置くことができます
※ if **文** で詳しく説明します
- ・ さらに、**文** には if **文**、for **文**、while **文** などがあります

インデント

行頭での字下げ

- ・ 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

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)

- ・ 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

「class」がついた { } (括弧) 内

- ・ 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
※ **クラス** に関する説明は Java プログラミング II でします

文字や文字列、数値の表記

リテラル	文字や文字列、数値の表記をいう。 表記する対象に応じて、 文字リテラル、文字列リテラルなどと呼ばれる。
文字リテラル（一文字の表記）	``（シングルクォート）で文字を囲む 例えば、`A`、`b`、`c`
文字列リテラル（文字列の表記）	``（ダブルクォート）で文字列を囲む 例えば、`Hello`、`こんにちは`  `A` はOKですが、`Hello` はダメです
数値のリテラル（数値の表記）	`` や `` で囲まない 例えば、123、-23、0.24 数値のリテラルには、 整数リテラルや浮動小数点数リテラルなどがある。

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);
    }
}
```

実行画面

```
>java Sample2_3
A
Hello
123
0.24
-- Press any key to exit (Input "c" to continue) --
```



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つの文字の組合せにより表記される1文字

例えば、'¥n'、'¥t'

この他に

¥b	バックスペース	¥t	水平タブ
¥n	改行	¥f	改ページ
¥r	復帰	¥'	'
¥"	"	¥¥	¥



"¥n"や"Hello¥n"としてもよい

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
¥
-- Press any key to exit (Input "c" to continue) --
```

整数の進数表現

整数リテラルと進数表現

10 の 8 進数表現	<u>0</u> 12	0 で数値を始める → 8 進数表現とみなされます 例えば System.out.println(012);
10 の 16 進数表現	<u>0x</u> A	0x で数値を始める → 16 進数表現とみなされます 例えば、 System.out.println(0xA);
10 の 10 進数表現	10	上記以外 → 10 進数表現とみなされます 例えば、 System.out.println(10);



019 や 0x4g はエラーです
8 進数の各桁は 0～7 までです
16 進数の各桁は 0～9、A～F までです

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 進数の 10 は");
        System.out.print(10);
        System.out.println("です。");
        System.out.print("8 進数の 10 は");
        System.out.print(010);
        System.out.println("です。");
        System.out.print("16 進数の 10 は");
        System.out.print(0x10);
        System.out.println("です。");
    }
}
```

実行画面

```
>java Sample2_5
10 進数の 10 は 10 です。
8 進数の 10 は 8 です。
16 進数の 10 は 16 です。
-- Press any key to exit (Input "c" to continue) --
```



文字列リテラルは、+（プラス）により文字リテラルや数値のリテラルと接続できます

ソースファイル名：Ext2_1.java

```
class Ext2_1{
    public static void main(String[] args){
        // Java 言語風な記述
        System.out.println("10 進数の 10 は"+10+"です。");
        System.out.println("8 進数の 10 は"+010+"です。");

        // C 言語風な記述（出力形式を整える%d や%f、%c などが利用できます）
        System.out.printf("16 進数の 10 は%d です。¥n",0x10);
    }
}
```

実行画面

```
>java Ext2_1
10 進数の 10 は 10 です。
8 進数の 10 は 8 です。
16 進数の 10 は 16 です。
-- Press any key to exit (Input "c" to continue) --
```

ソースファイル名：Ext2_2.java

```
class Ext2_2{
    public static void main(String[] args){
        // 整数 10 と整数 1 の加算を行い、結果は整数 11
        System.out.println(10+1);
        // 'a'の文字コード 97 と整数 1 の加算を行い、結果は整数 98
        System.out.println('a'+1);
        // 文字列"Hello"と整数 1 を連結し、結果は文字列"Hello1"
        System.out.println("Hello"+1);
    }
}
```

実行画面

```
>java Ext2_2
11
98
Hello1
-- Press any key to exit (Input "c" to continue) --
```