Ｊａｖａプログラミング Ｉ

２回目　“ようこそＪａｖａへ”
２００６年４月２４日（月）

● 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  );
    }
}
```

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Ｊａｖａへ！”);
        System.out.println(“Ｊａｖａをはじめましょう！”);
    }
}
```

実行画面

```
>java Sample2_1
ようこそＪａｖａへ！
Ｊａｖａ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( );　　行末に改行が挿入される
２．System.out.print( );　　行末に改行が挿入されない
３．System.out.printf( );　　Ｃ言語 printf()関数と類似した出力関数

ソースコード例

ソースファイル名：Sample2_2.java

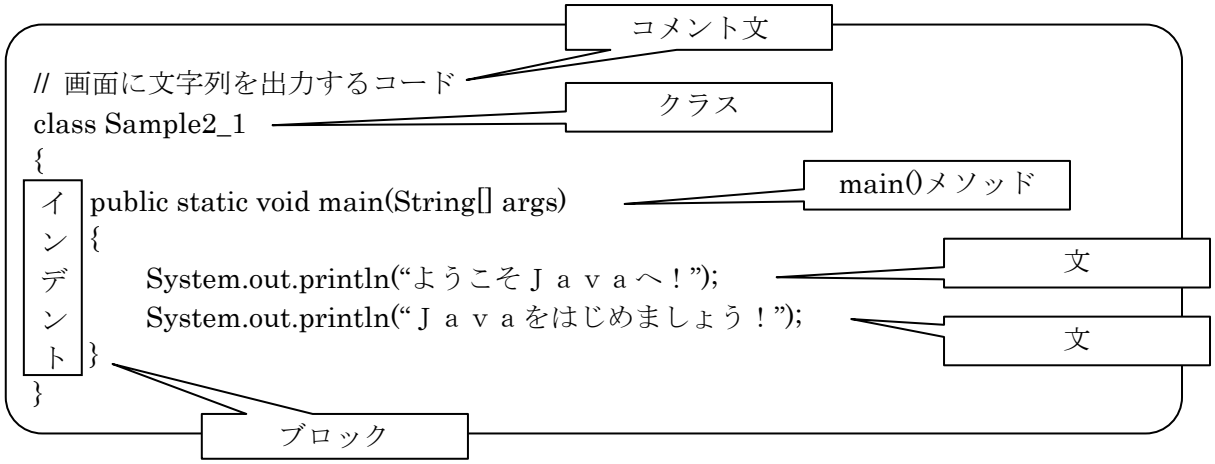
```
class Sample2_2
{
    public static void main(String[] args)
    {
        System.out.println(“1. println()による出力（行末に改行あり）”);
        System.out.print(“2. print()による出力（行末に改行なし）”);
        System.out.printf(“3. printf()による出力¥n”);
    }
}
```

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	{} で囲まれた部分
main()メソッド	「main」がついたブロック プログラムの処理が始まる部分
文	処理の単位で最後に ;(セミコロン)を付ける 上から順番に実行される
インデント	読み易くするための行頭での字下げ
コメント	// 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ありメモなどを記載する
クラス	「class」がついたブロック コードは最低１つのクラスから成り立つ

● 文字や文字列、数値の表記

一文字の表記（文字リテラル）	‘’で文字を囲む	例えば、‘A’、‘b’、‘c’
文字列の表記（文字列リテラル）	“”で文字列を囲む	例えば、“Hello”、“こんにちは”
数値の表記（数値のリテラル）	‘’や“”で囲まない	例えば、123、-23、0.24

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。
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　¥を最初につけた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’

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

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) --
```

● 数値の進数表現

10 の 10 進数表現　10　System.out.println(10);
10 の 8 進数表現　012　System.out.println(012);
10 の 16 進数表現　0xA　System.out.println(0xA);

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) --
```

参考：＋（プラス）を用いて文字列や数値をつなぐことができる
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
		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) --
```