

3 回 目 変 数

● 変 数
変 数

一時的に値を記憶させておく機能
型（データ型）と識別子をもつ



型（データ型）

変数の種類
型に応じて記憶できる値の種類や範囲が決まる

型	値の種類	値の範囲
boolean	真偽値	true / false
char	2 バイト文字(16 ビット Unicode)	0x0000 ~ 0xffff
byte	1 バイト整数	$-2^7 \sim 2^7 - 1$
short	2 バイト整数	$-2^{15} \sim 2^{15} - 1$
int	4 バイト整数	$-2^{31} \sim 2^{31} - 1$
long	8 バイト整数	$-2^{63} \sim 2^{63} - 1$
float	4 バイト単精度浮動小数点数	
double	8 バイト倍精度浮動小数点数	

識別子

変数の名前
(規則)

- ・ a～z、A～Z(英字)、0～9(数字)、_(アンダーライン)、\$(ドル記号)を用いる
- ・ 数字ではじめることはできない
- ・ 長さには制限はない
- ・ 大文字と小文字は異なるものとして区別される
- ・ 空白(スペース)を含めることはできない
- ・ 次の Java のキーワード（すべて小文字）は名前にできない

abstract	const	final	int	public	throw
assert	continue	finally	interface	return	throws
boolean	default	float	long	short	transient
break	do	for	native	static	true
byte	double	goto	new	strictfp	try
case	else	if	null	super	void
catch	enum	implements	package	switch	volatile
char	extends	import	private	synchronized	while
class	false	instanceof	protected	this	

例えば、

○ a, abc, ab_c, F1

× 12a, return, is-a

● 変数の宣言

変数の宣言

変数を使用できるようにする準備

型と識別子を指定して次のように行う。

```
型 識別子;
```

変数の初期化

変数を宣言した際に適当な値を代入しておくこと
宣言されたままの変数にはゴミ（予期しない値）が入っている
※Java では、初期化していない変数を利用しようとすると
「変数〇〇は初期化されていない可能性があります」
というメッセージで警告がある

右辺を左辺に代入する演算子 =（イコール）を用いて次のように行う。

```
識別子 = 値;
```

ソースコード例

ソースファイル名 : Sample3_1.java

```
// 変数の宣言と初期化
class Sample3_1
{
    public static void main(String[] args)
    {
        int num1; // 変数の宣言
        num1 = 0; // 変数の初期化

        // 変数の宣言と初期化を同時に行う
        int num2 = 0;

        // 同一の型の変数を複数同時に宣言する
        int num3, num4;

        // 同一の型の変数を複数同時に宣言・初期化する
        int num5 = 0, num6 = 0;

        // 同一の型の変数を複数同時に宣言、一部初期化する
        int num7 = 0, num8, num9 = 0;
    }
}
```

● 変数の利用

演算子 `=` を用いて変数への値の代入や値の変更、他の変数からの値の代入ができる。変数の値の出力は次のように行う。

```
System.out.println( 識別子 );
```

ソースコード例

ソースファイル名 : Sample3_2.java

```
// 変数の利用
class Sample3_2
{
    public static void main(String[] args)
    {
        // 変数の宣言と初期化
        int num1 = 0;
        int num2 = 0;

        // 変数の値の出力
        System.out.println("変数 num1 の値は"+ num1 + "です。");
        System.out.println("変数 num2 の値は"+ num2 + "です。");

        // 変数の値を変更
        num1 = 5;
        System.out.println("変数 num1 の値を変更しました。");

        System.out.println("変数 num1 の値は" + num1 + "です。");
        System.out.println("変数 num2 の値は" + num2 + "です。");

        // ほかの変数の値を代入
        num2 = num1;
        System.out.println("変数 num1 の値を変数 num2 に代入しました。");

        System.out.println("変数 num1 の値は" + num1 + "です。");
        System.out.println("変数 num2 の値は" + num2 + "です。");
    }
}
```

ここで、演算子 `+` は文字列を連結する機能をもつ (参照) 第2回目プリント, p.7

実行画面

```
>java Sample3_2
変数 num1 の値は 0 です。
変数 num2 の値は 0 です。
変数 num1 の値を変更しました。
変数 num1 の値は 5 です。
変数 num2 の値は 0 です。
変数 num1 の値を変数 num2 に代入しました。
変数 num1 の値は 5 です。
変数 num2 の値は 5 です。
-- Press any key to exit (Input "c" to continue) --
```

● キーボード入力の基本形（文字列を入力する場合）

ソースファイル名 : クラス名.java

```
import java.io.*;
class クラス名
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br;
        br = new BufferedReader(new InputStreamReader(System.in));

        String str;
        str = br.readLine();

        ...

    }
}
```

このように記述

このように記述

文字列を
扱う変数

このように記述
※main()メソッド
の最初に一度だけ
でよい

このように記述

ユーザからの入力待つ状態で止まる。文字列をキーボードから入力し Enter キーを押すとその文字列が変数 **str** に代入される。

ソースコード例

ソースファイル名 : Sample3_3.java

```
// キーボードから文字列を入力する
import java.io.*;

class Sample3_3
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br;
        br = new BufferedReader(new InputStreamReader(System.in));

        // キーボードから文字列を読み込む
        String str1, str2;

        // キーボードからの入力を促すメッセージと入力
        System.out.println(" 1 つ目の文字列を入力してください。");
        str1 = br.readLine();

        System.out.println(" 2 つ目の文字列を入力してください。");
        str2 = br.readLine();

        // 読み込まれた文字列を表示する
        System.out.println(str1 + "と" + str2 + "が入力されました。");
    }
}
```

実行画面

```
>java Sample3_3
1 つ目の文字列を入力してください。
楽しい
2 つ目の文字列を入力してください。
J a v a
楽しいと J a v a が入力されました。
-- Press any key to exit (Input "c" to continue) --
```

● キーボード入力の基本形（数値を入力する場合）

ソースファイル名：`クラス名.java`

```
import java.io.*;
class クラス名
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br;
        br = new BufferedReader(new InputStreamReader(System.in));

        String str;
        str = br.readLine();

        int num;
        num = Integer.parseInt(str);

        ...

    }
}
```

文字列を
扱う変数

このように記述

このように記述

このように記述
※main()メソッド
の最初に一度だけ
でよい

このように記述

ユーザからの入力を待つ状態で止まる。文
字列をキーボードから入力し **Enter** キーを
押すとその文字列が変数 **str** に代入される。

このように記述

入力された文字列が表す **int** 型の数値に変
換され、変数 **num** に代入される。※

※変換したい型に応じて次のような関数を用いる

boolean 型	<code>Boolean.parseBoolean();</code>
byte 型	<code>Byte.parseByte();</code>
short 型	<code>Short.parseShort();</code>
int 型	<code>Integer.parseInt();</code>
long 型	<code>Long.parseLong();</code>
float 型	<code>Float.parseFloat();</code>
double 型	<code>Double.parseDouble();</code>

ソースコード例

ソースファイル名 : Sample3_4.java

```
// キーボードから数値を入力する
import java.io.*;

class Sample3_4
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader br;
        br = new BufferedReader(new InputStreamReader(System.in));

        // キーボードからの入力を促すメッセージ
        System.out.println("整数を入力してください。");

        // キーボードから文字列を読み込む
        String str;
        str = br.readLine();

        // 文字列が表す int 型の数値に変換する
        int num;
        num = Integer.parseInt(str);

        // 読み込まれた数値を表示する
        System.out.println(num + “が入力されました。”);
    }
}
```

実行画面

```
>java Sample3_4
整数を入力してください。
123
123 が入力されました。
-- Press any key to exit (Input "c" to continue) --
```

参考：Sample3_4 実行時に数値を入力するのを間違えて文字を入力したら？

実行画面

```
>java Sample3_4
整数を入力してください。
a
Exception in thread "main" java.lang.NumberFormatException: For input string: "a"
    at java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)
    at java.lang.Integer.parseInt(Integer.java:447)
    at java.lang.Integer.parseInt(Integer.java:497)
    at Sample3_4.main(Sample3_4.java:20)
    -- Press any key to exit (Input "c" to continue) --
```

Java ではこのような実行時におけるエラーを処理する“例外処理”という枠組みが備えられている。ここでは詳細にはふれず、後期の Java プログラミング II で詳しく解説する。

参考：キーボード入力のその他の方法は？

ソースコード例

ソースファイル名：Ext3_1.java

```
// キーボードから数値を直接読み込む
import java.util.*;

class Ext3_1
{
    public static void main(String[] args) // この場合は、throws IOException 記述不要
    {
        Scanner s;
        s = new Scanner(System.in);

        // こちらの方法では以下の例のように直接 int 型整数、long 型整数、
        // float 型実数、double 型実数を変数に読み込むことができる。
        int i = s.nextInt();
        long l = s.nextLong();
        float f = s.nextFloat();
        double d = s.nextDouble();
        String str = s.next();

        // 変数の中身をみてみましょう。
        System.out.println("i="+i+", l="+l+", f="+f+", d="+d+", str="+str);
    }
}
```

このように記述

このように記述
※main()メソッドの最初に一度だけでよい

このように記述
それぞれユーザからの入力を待つ状態で止まる。文字列をキーボードから入力し Enter キーを押すとその文字列が指定の型に変換されて、変数に代入される。

実行画面

```
>java Ext3_1  
12  
2007  
12.4  
3.1415  
Hello  
i=12, l=2007, f=12.4, d=3.1415, str=Hello  
-- Press any key to exit (Input "c" to continue) --
```