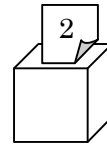


● 変 数

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	0x0000 ~ 0xffff
byte	1 バイト整数	- 2 ⁸ ~ 2 ⁸ - 1
short	2 バイト整数	- 2 ¹⁶ ~ 2 ¹⁶ - 1
int	4 バイト整数	- 2 ³² ~ 2 ³² - 1
long	8 バイト整数	- 2 ⁶⁴ ~ 2 ⁶⁴ - 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)

- ・英字、数字、アンダーライン、\$ などを用いる
- ・長さには制限はない
- ・Java のキーワードを名前にできない 例えば、if, while, for など
- ・数字ではじめることはできない
- ・大文字と小文字は異なるものとして区別される

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が入っているため)

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 num; // 変数の宣言
        num = 3; // 変数の初期化

        // 変数の宣言と初期化を同時に行うこともできる
        // int num = 3;
    }
}
```

● 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。");
    }
}
```

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）

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

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 **str** に読込まれる。

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）

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);
        ...
    }
}
```

このように記述

このように記述

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 **str** に読込まれる。

入力された文字列が表す **int** 型の数値に変換され、**num** に読込まれる。※

※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
    {
        // 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。");

        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 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”という仕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)
    {
        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);
    }
}
```


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) --
```