

3回目 変数

■ 今日の講義で学ぶ内容 ■

- 変数とは
- 変数の使い方
- キーボード入力の仕方

変数

変数

一時的に値を記憶させておく機能です
変数は、型（データ型ともいいます）と識別子をもちます



型

変数に記憶できる値の種類です
型は、値の種類に応じて次の 8 種類があり、これを基本型といいます

基本型	値の種類	値の範囲または例
boolean	真偽値	true または false
char	16ビット文字(16ビットUnicode)	'a'、'b'、...
byte	8ビット符号付き整数	- 128 ~ 127
short	16ビット符号付き整数	- 32768 ~ 32767
int	32ビット符号付き整数	- 2^{31} ~ $2^{31} - 1$
long	64ビット符号付き整数	- 2^{63} ~ $2^{63} - 1$
float	32ビット単精度浮動小数点数	約 $\pm 3.4 \times 10^{38}$ ~ 1.4×10^{-45}
double	64ビット倍精度浮動小数点数	約 $\pm 1.8 \times 10^{308}$ ~ 4.9×10^{-324}

C言語では型ごとに符号なしや符号ありの指定ができます
たとえば、unsigned int a; や signed long b; です
Javaでは基本型は符号ありのみ（booleanとcharを除く）です

C言語では各型が扱う値の範囲はプログラム環境ごとに様々です
Javaでは各型が扱う値の範囲は一定です

'a'や'b'の文字リテラルはJava内部で16ビットUnicodeで表現されています
char型は16ビット符号なし整数を用いて16ビットUnicodeを扱います
char型は16ビット符号なし整数（0~65535）を扱うこともできます

識別子

変数につける名前です

識別子は、変数を一意に識別します

識別子には、規則があります

- 使える記号は、`a~z`、`A~Z`、`0~9`、`_`(アンダースコア)、`$`(ドル記号)です
- 最初の記号は数字以外である必要があります
- 名前の長さは無制限です
- 大文字と小文字は異なるものとして区別されます
- 途中に空白(スペース)を含めることはできません
- 次の Java のキーワード (すべて小文字) は使用できません

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



識別子として次のものは良いです

- `a`、○ `num`、○ `pos_mouse`、○ `Return`

しかし、次のものはエラーです

- × `12years` ← 数字から始まることはできません
- × `return` ← Java のキーワードは使用できません
- × `num-apples` ← ハイフンは使用できません
- × `have cats` ← 空白は使用できません

変数の宣言

変数の宣言

変数を使用できるようにするための準備です


変数の型と識別子を指定して次のように記述します

```
型 識別子;
```

コード例 | `int num;`

変数の初期化

変数を宣言した際に適切な値を代入しておくことで宣言された変数には予期しない値が入っていることがあります

 初期化していない変数を利用しようとすると
「変数ooは初期化されていない可能性があります」
というコンパイルエラーがでます

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記述します

```
識別子 = 値;
```

コード例 | `num = 0;`

 プログラミング言語では代入と等しいを明確に区別します

- 代入は、= (シングルイコール) で表現します
 - 等しいは、== (ダブルイコール) で表現します
- (※) == (ダブルイコール) は後の回で紹介します
(※) 数学では代入と等しいを同じ記号 = (イコール) で表記しますので注意しましょう

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
        // ,(カンマ)で変数を区切る
        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;
    }
}
```

変数は宣言された直後から利用することができます

 宣言されていない（宣言する前に）変数を利用しようとすると「シンボルを見つけられません」というコンパイルエラーがでます

変数の値の変更

変数をもつ値を変更します
変数への値の代入や変数の値の上書き・変更が行えます

 変数の初期化が終わればそれ以降その変数は初期化で代入した値を保持しています

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記述します

```
識別子 = 値;
```

コード例 | `num = 2;`

または

```
識別子 1 = 識別子 2;
```

コード例 | `num = a;`

 上の例で、変数 a の値を変数 num へ代入した後も、変数 a の値はそのままです

変数の値の出力

変数をもつ値を画面に表示します

変数の識別子を指定して次のように記述します

```
System.out.println( 識別子 );
```

コード例 | `System.out.println(num);`

 `System.out.println();` の他に、

- `System.out.print();`
- `System.out.printf();`

を用いてもよいですが、行末に改行が自動的に入るかどうか気に付けましょう

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持ちます
(参照) 第2回目講義7°リト

ある変数の値を別の変数に代入することもできます

実行画面

```
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。
```

キーボード入力の基本形（文字列を入力する場合）

キーボードからの文字列入力を行うコードは以下のような形です

```
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
C言語の#includeに対応します
キーボード入力の機能が使用可能になります

このように記述
throws IOException

このように記述
キーボード入力を
する前に一度だけ記述
します

String str;
文字列を扱うString型の変数を宣言します
キーボードから入力される文字列を代入する
ために用います


br.readLine();
ユーザからの入力を待つ状態で止まります
文字列をキーボードから入力しEnterキーを押すとその
文字列が代入演算子=により変数strに代入されます

String型

文字列を代入できる型です

"Hello"や"こんにちは"など文字列リテラルを代入できます

 変数の基本型には、
boolean、char、byte、short、int、long、float、double
の8種類があります

 **String型**は基本型ではなく、**参照型**とよばれる型です
これらの違いは後の回で詳しく解説します
ここでは、基本型と同様に考えてください

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));

        // キーボードからの文字列を受け取る変数の宣言
        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

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。

キーボード入力の基本形（整数を入力する場合）

キーボードからの整数入力を行うコードは以下のような形です

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));
```

```
        :
```

```
        int num;
```

```
        num = Integer.parseInt(br.readLine( ));
```

```
        :
```

```
    }
```

```
}
```

このように記述

C 言語の #include に対応します

キーボード入力の機能が使用可能になります

このように記述

このように記述

キーボード入力をする前に一度だけ記述します

int num;

整数を扱う int 型の変数を宣言します

Integer.parseInt(br.readLine());

ユーザからの入力を待つ状態で止まります

文字列をキーボードから入力し Enter キーを押すと入力された文字列が int 型の数値に変換され、代入演算子=により int 型の変数 num に代入されます



この部分 — は
変換したい型に応じて次のように使い分けます
(入力したいデータ型) → (コード)

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

さらに、
入力された値を代入する変数の型も変更します

たとえば、
実数を入力したい場合は、
double d;
d = Double.parseDouble(br.readLine());
とすればよいです

ソースコード例

ソースファイル名：Sample3_4.java

```
// キーボードから整数を入力する
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。");

        // キーボードから整数を読み込む
        int num;
        num = Integer.parseInt(br.readLine());

        // 読み込まれた整数を表示する
        System.out.println(num + "が入力されました。");
    }
}
```

実行画面

整数を入力してください。

123 

123 が入力されました。



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

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)
```

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ます。ここでは詳細にはふれず、Java プログラミング II で詳しく解説します。



キーボード入力のその他の方法～Scanner クラス～

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.*;
```

このように記述

C 言語の#include に対応します
以下のキーボード入力の機能が使用可能になります

```
class Ext3_1
{
    public static void main(String[] args)
    {
        Scanner s;
        s = new Scanner(System.in);
    }
}
```

このように記述

キーボード入力をする前に一度だけ
記述します

```
// こちらの方法では以下の例のように 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ます
int i = s.nextInt();
long l = s.nextLong();
float f = s.nextFloat();
double d = s.nextDouble();
String str = s.next();
}
```

このように記述

ユーザからの入力を待つ状態で止まります
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、代入演
算子=により各変数に代入されます

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

実行画面

```
12
2007
12.4
3.1415
Hello
i=12, l=2007, f=12.4, d=3.1415, str=Hello
```

■ 今日の講義のまとめ ■

- 変数は値を一時的に保持しておく機能を持ち、型と識別子を指定して宣言します。
- 変数の型はその変数がどのような値を保持できるかを表わします。変数の型には `boolean` や `char`、`byte`、`short`、`int`、`long`、`float`、`double` があり、真偽値や文字、整数、実数を管理できます。これらは基本型と呼ばれます。
- 変数の識別子は変数の名前です。識別子には一定のルールがあります。例えば、識別子は数字で始まってはいけません。
- 変数に値を代入するには、右辺を左辺に代入する演算子 `=` を用います。
- キーボード入力により、文字列や数字を入力することができます。文字列は `String` 型の変数により保持できます。

