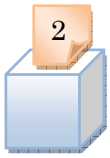
3回目　変　　数

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
変数は、型（データ型）と識別子をもちます



型（データ型）　　変数に記憶する値の種類
変数の型は、記憶できる値の種類と範囲を決定します

次の型が利用でき、これらの型は特に基本型とよばれます

基本型	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} ～ $2^{7}-1$
short	2バイト整数	-2^{15} ～ $2^{15}-1$
int	4バイト整数	-2^{31} ～ $2^{31}-1$
long	8バイト整数	-2^{63} ～ $2^{63}-1$
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　　変数につける名前
変数の識別子は、変数を一意に識別します

変数の識別子は、次の規則を守りましょう
・a～z、A～Z(英字)、0～9(数字)、_(ｱﾝﾀﾞｰﾗｲﾝ)、$(ﾄﾞﾙ記号)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(ｽﾍﾟｰｽ)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	


識別子として、
a, abc, ab_c, F1
は良いですが、
12a, return, is-a
はエラーです

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

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;
```

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ります

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;
```

⚠ 数学では、代入と等しいを同じ記号＝（ｲｺｰﾙ）で表記しますが
プログラミング言語では、これらを明確に区別します
・代入は、＝（ｼﾝｸﾞﾙｲｺｰﾙ）で表現します
・等しいは、＝＝（ﾀﾞﾌﾞﾙｲｺｰﾙ）で表現します ※これは後の回で紹介します

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;
    }
}
```

変数の利用

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出ます

右辺を左辺に代入する演算子 ＝（ｲｺｰﾙ）を用いて
・変数への値の代入
・変数の値の上書き・変更
ができます

変数の値の出力は次のように記述します

```
System.out.println(　識別子　);
```

⚠ System.out.printf( ); C言語の printf()関数のように%d, %c, %ld などを用いて出力の書式を指定することができます
int num=10;
System.out.printf(″変数の値は%d です¥n″,num);

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回目講義ﾌﾟﾘﾝﾄ

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

```
>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
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

このように記述
文字列を扱う String 型の変数を宣言します

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変数 str に代入されます

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。
Ｊａｖａ
楽しいとＪａｖａ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（整数を入力する場合）

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));

            :
        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

このように記述
整数を扱う int 型の変数を宣言します

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 キーを押すと入力された文字列が int 型の数値に変換され、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（真偽値）	→	Boolean.parseBoolean(…);
byte 型（整数）	→	Byte.parseByte(…);
short 型（整数）	→	Short.parseShort(…);
int 型（整数）	→	Integer.parseInt(…);
long 型（整数）	→	Long.parseLong(…);
float 型（実数）	→	Float.parseFloat(…);
double 型（実数）	→	Double.parseDouble(…);

さらに、
入力された値を代入する変数の型も合わせて変更します

たとえば、
実数を入力したい場合は、

```
double d;
d = Double.parseDouble(br.readlLine( ));
```

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込む
        int num;
        num = Integer.parseInt(br.readLine());

        // 読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

```
>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) --
```

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

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 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();

        // 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
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

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、変数に代入されます

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) --
```