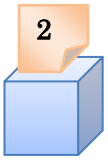
3回目　変　　数

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$\pm 3.4\times10^{38}$～1.4×10^{-45}
double	64ビット**倍精度浮動小数点数**	約$\pm 1.8\times10^{308}$～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、_(ｱﾝﾀﾞｰﾗｲﾝ)、$(ﾄﾞﾙ記号)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(ｽﾍﾟｰｽ)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**

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
などは良いですが、
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**適当な値を代入しておく**ことです
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**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
（※）＝＝（ﾀﾞﾌﾞﾙｲｺｰﾙ）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**がでます

変数の値の変更　　**変数**がもつ**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;

変数の値の出力　　**変数**がもつ**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**に気を付けましょう

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**する機能をもちます
（参照）第２回目講義ﾌﾟﾘﾝﾄ

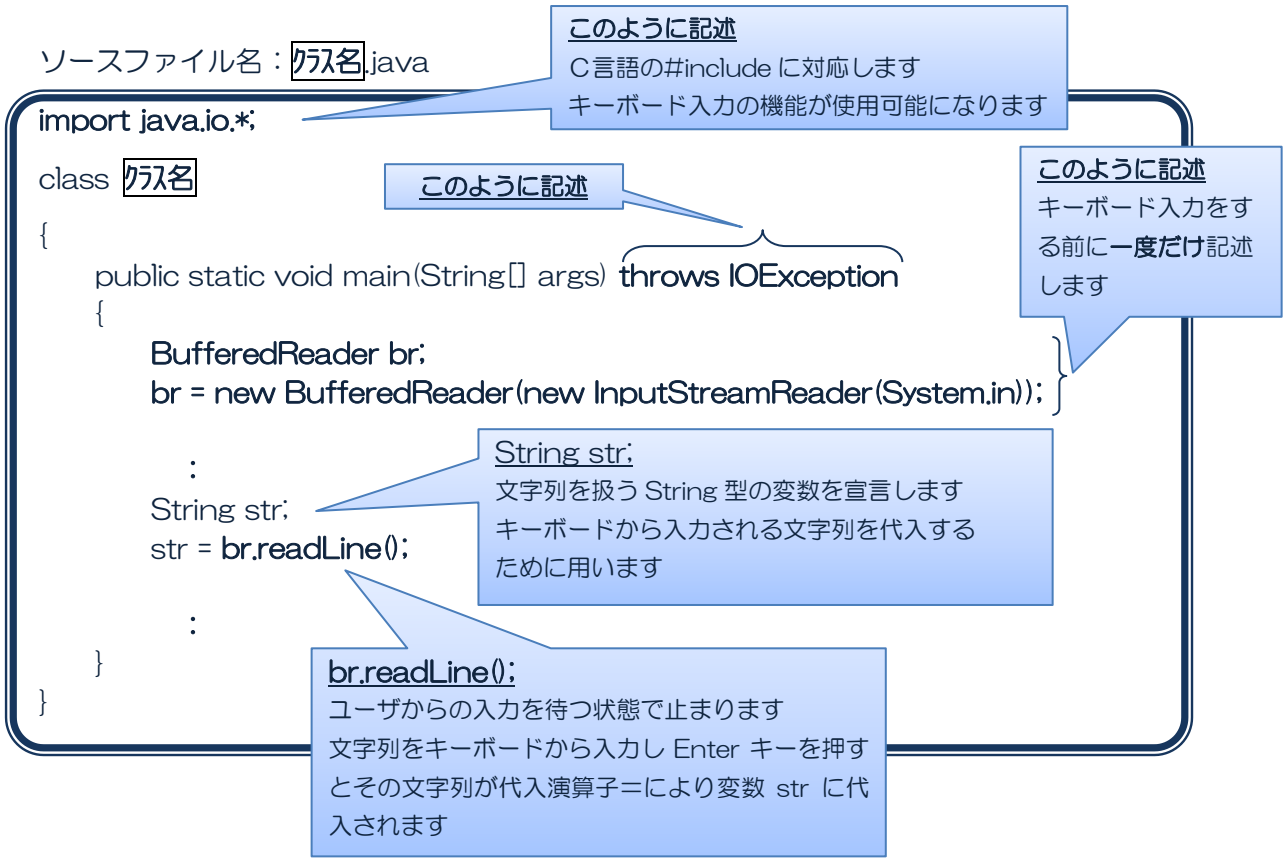
ある変数の値を別の変数に代入することもできます

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。
```

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



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することができます

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

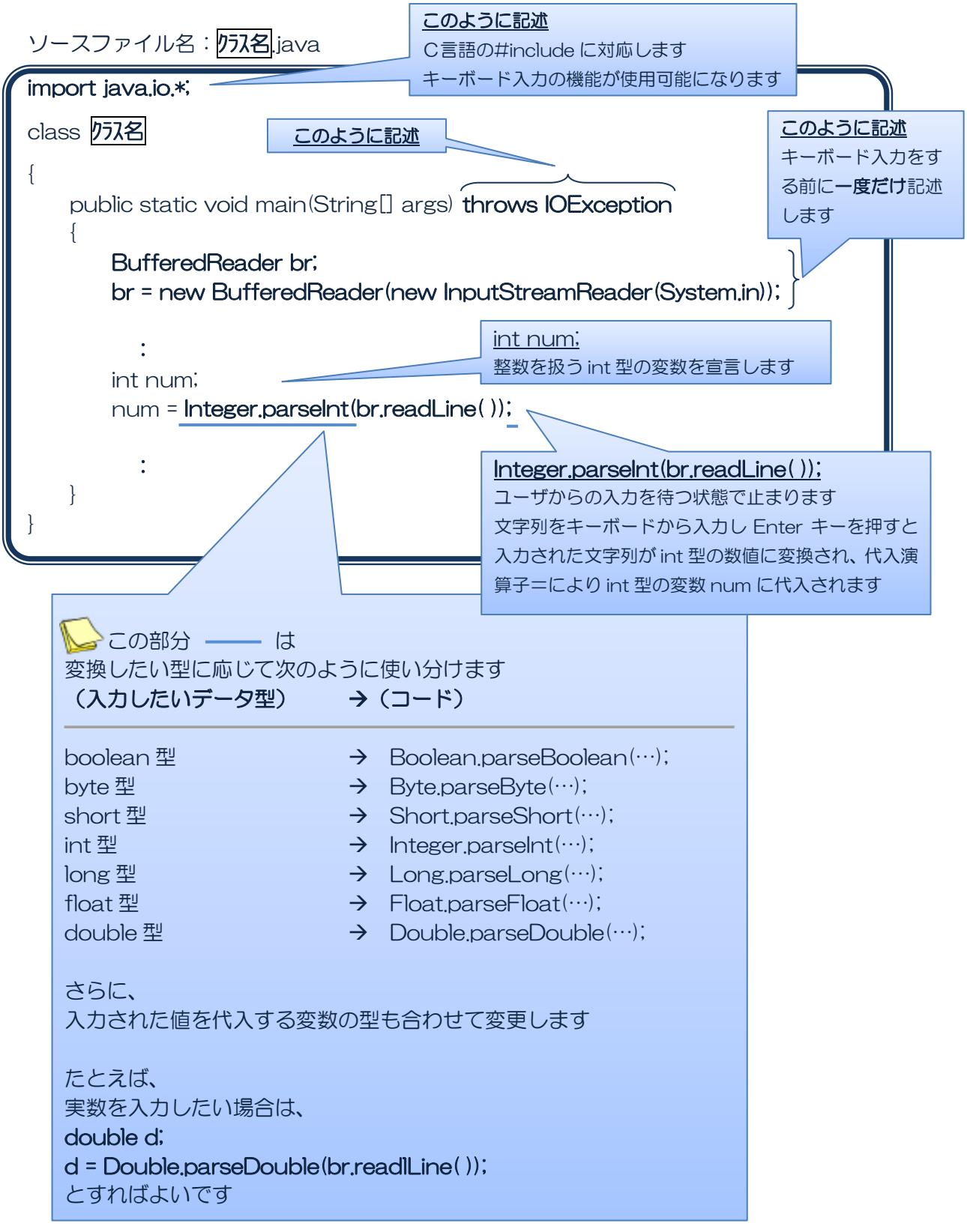
```
// キーボードから文字列を入力する
import java.io.*;

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。
```

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



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。
```

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)
```

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、代入演算子＝により各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
```