変数と演算子２

前回は標準出力を行う方法として println、print、printf について学習した。その際、変数についても少し触れたが、今回はより詳しく解説していく。また、演算子や標準入力、乱数の生成についても解説する。

変数

プログラムでデータを取り扱う場合には対象となるデータを保存する必要がでてくる。このデータを保存する場所のことを「変数」と呼び、変数を説明する上ではよく「データを入れておく箱」として例えられ、必要に応じて中身（データ）は後から自由に変更することが可能である。

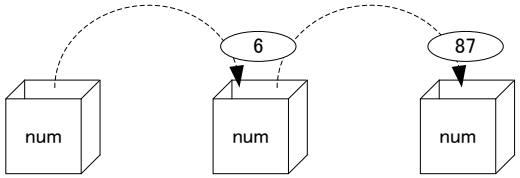

図１　変数

変数は以下のように宣言を行うことで利用する。

書式
データ型　変数名;

Java に用意されているデータ型には大きく分けて int 型や double 型といった「基本型」とクラス型や配列型などのオブジェクトへのアドレスを指し示す「参照型」の２つがある。以下に Java で使われる型を示す。

型				値
基本型	算術型	整数型	byte	-128～127
			short	-32,768～32,767
			int	-2,147,483,648～2,147,483,647
			long	-9,223,372,036,854,775,808～ 9,223,372,036,854,775,807
			char	0～65,535
		実数型 （浮動小数点型）	float	±1.40239846E-45～±3.40282347E+38
			double	±4.94065645841246544E-324～ ±1.79769313486231507E+378
	論理型	boolean		true　false
参照型	クラス型			オブジェクトの参照
	インターフェイス型			
	配列型			

以下のプログラムは算術型変数で使える数値の範囲を出力するプログラム例である。

プログラム例１

```
public class Prog03_01
{
        public static void main(String[ ] args)
        {
                System.out.println("byte    : " + Byte.MIN_VALUE + "～" + Byte.MAX_VALUE);
                System.out.println("short   : " + Short.MIN_VALUE + "～" + Short.MAX_VALUE);
                System.out.println("int     : " + Integer.MIN_VALUE + "～" + Integer.MAX_VALUE);
                System.out.println("long    : " + Long.MIN_VALUE + "～" + Long.MAX_VALUE);
```

```
                System.out.println("char    : " + (int)Character.MIN_VALUE + "～"
                                                + (int)Character.MAX_VALUE);
                System.out.println("float   : " + Float.MIN_VALUE + "～" + Float.MAX_VALUE);
                System.out.println("double  : " + Double.MIN_VALUE + "～" + Double.MAX_VALUE);
        }
}
```

出力例

```
byte    : -128～127
short   : -32768～32767
int     : -2147483648～2147483647
long    : -9223372036854775808～9223372036854775807
char    : 0～65535
float   : 1.4E-45～3.4028235E38
double  : 4.9E-324～1.7976931348623157E308
```

変数名は識別子のところで示したルールに従い、各自で自由に設定できる。ただし、変数名については全て小文字を用いる習慣がある。また、変数は以下のようにして、データを保存することで利用する。この変数にデータ（値）を保存する処理のことを「変数に値を代入する」という。

書式

```
変数名 = 値;
```

たとえば、num という変数名の int 型の変数に「100」を代入するには、以下の使用例のように行う。

使用例

```
int num;
num = 100;
```

また、変数の宣言と値の代入は以下のように同時に行うことができる。

使用例

```
int num = 100;
```

「変数の宣言と同時に値の代入を行う」場合を特に「初期化」と言い、その時の値を「初期値」という。ただし、より一般的には初めて変数に値を代入する場合も含めて「初期化」と言っているので注意が必要である。

演算子

前回では数値や変数を用いた四則演算も紹介した。これは以下のような形で計算し、変数に値を代入することが出来る。

```
a = b + 10;
```

ここで「a」、「b」、「10」のように演算の対象となるものを「オペランド」という。また、「=」や「+」は「演算子」と呼ばれ、「=」は特に「代入演算子」、「+」は「算術演算子」というものに分類される。以下に算術演算子と代入演算子を示す。ただし、代入演算子はここに示したもの以外にもある。

算術演算子	
+	加算する
−	減算する
*	乗算する
/	除算する

%	剰余を求める

代入演算子	
=	代入
+=	加算したものを代入する
-=	減算したものを代入する
*=	乗算したものを代入する
/=	除算したものを代入する
%=	剰余を求め、代入する

※ 例えば、「a += b;」は「a = a + b;」と同じ処理となる。

演算子には演算の優先順位がある。括弧を含め、ここに示した演算子の優先順位は以下である。

1	( )
2	*, /, %
3	+, -
4	=, +=, -=, *=, /=. %=

また、ここで示した演算子の他にも「関係演算子」や「ビット演算子」など演算子の種類は多数ある。

標準入力

Java の標準入力はキーボードに割り当てられている。ここではキーボードから値を入力する方法を示す。現状は使用方法を把握しておくだけで良い。処理の詳細は今後の授業で学ぶ。

使用例

```
import java.util.Scanner;

public class A
{
        public static void main(String[ ] args)
        {
                Scanner stdIn = new Scanner(System.in);
                int x = stdIn.nextInt();
        }
}
```

①プログラムの先頭(クラス宣言の前)に書く

②mainメソッドの先頭(読み込み処理の③より前)に置く

③キーボードから整数値を読み込む

次にキーボードから整数値を２つ読み込み、それらに対して算術演算をおこなったプログラム例を示す。

プログラム例２

```
import java.util.Scanner;

public class Prog03_02
{
        public static void main(String[ ] args)
        {
                Scanner stdIn = new Scanner(System.in);

                System.out.println("x と y で算術演算を行います");

                System.out.println("x を入力してください");
                int x = stdIn.nextInt();

                System.out.println("y を入力してください");
                int y = stdIn.nextInt();
```

```
                System.out.println("x + y = " + ( x + y ));
                System.out.println("x - y = " + ( x - y ));
                System.out.println("x * y = " + ( x * y ));
                System.out.println("x / y = " + ( x / y ));
                System.out.println("x % y = " + ( x % y ));
        }
}
```

出力例(斜体はキーボードから入力した値)

```
x と y で算術演算を行います
x の値：9
y の値：4
x + y = 13
x - y = 5
x * y = 36
x / y = 2
x % y = 1
```

注目したいのは「x / y = 2」と「x % y = 1」の結果である。「x / y = 2」では、入力が 9 と 4 にしてあるので、割り算の結果は本来 2.25 であるが、整数同士の演算であるので、小数点以下が切り捨てられて 2 が表示されている。また、「x % y = 1」では剰余演算子「%」を用いている。これは 9 を 4 で割った際の余りが得られる演算子であるので、1 が表示されている。

整数型の入力には「stdIn.nextInt();」を用いたが。「nextInt()」の部分についてはデータの型に応じていくつか種類がある。

nextByte()	byte 型整数を読み込む
nextShort()	short 型整数を読み込む
nextInt()	int 型整数を読み込む
nextLong()	long 型整数を読み込む
nextFloat()	float 型実数を読み込む
nextDouble()	double 型実数を読み込む
nextBoolean()	boolean 型を読み込む
next()	文字列を読み込む（スペースやタブで区切られる）
nextLine()	文字列を 1 行読み込む

以下はプログラム例２を double 型での読み込みに変えた場合の例である。変更部分には下線を付けた。

プログラム例３

```
import java.util.Scanner;

public class Prog03_03
{
        public static void main(String[ ] args)
        {
                Scanner stdIn = new Scanner(System.in);

                System.out.println("x と y で算術演算を行います");

                System.out.print("x の値:");
                double x = stdIn.nextDouble();

                System.out.print("y の値:");
```

```
                double y = stdIn.nextDouble();

                System.out.println("x + y = " + ( x + y ));
                System.out.println("x - y = " + ( x - y ));
                System.out.println("x * y = " + ( x * y ));
                System.out.println("x / y = " + ( x / y ));
                System.out.println("x % y = " + ( x % y ));
        }
}
```

出力例(斜体はキーボードから入力した値)

```
xとyで算術演算を行います
xの値：9
yの値：4
x + y = 13.0
x - y = 5.0
x * y = 36.0
x / y = 2.25
x % y = 1.0
```

次にnext()とnextLine()を用いて文字列を入力した例を示す。

プログラム例４

```
import java.util.Scanner;

public class Prog03_04
{
        public static void main(String[ ] args)
        {
                Scanner stdIn = new Scanner(System.in);

                System.out.print("Please enter your name:");
                String s = stdIn.next();

                System.out.println("Hello, " + s + "!");
        }
}
```

出力例(斜体はキーボードから入力した値)

```
Please enter your name:Daiki Takeshita
Hello, Daiki!
```

プログラム例５

```
import java.util.Scanner;

public class Prog03_05
{
        public static void main(String[ ] args)
        {
                Scanner stdIn = new Scanner(System.in);

                System.out.print("Please enter your name:");
                String s = stdIn.nextLine();
```

```
                System.out.println("Hello, " + s + "!");
        }
}
```

出力例(斜体はキーボードから入力した値)

```
Please enter your name:Daiki Takeshita
Hello, Daiki Takeshita!
```

next()を用いたプログラム例４の出力例では文字列にスペースが入っている部分で入力が区切られていることが確認できる。空白やタブを含めた文字列を入力する場合にはプログラム例５のようにnextLine()を用いなくてはならない。

また、プログラム中の変数宣言に用いた「String 型」は参照型変数であり、文字列を取り扱うことができる。基本的な宣言方法と初期化、値の代入を確認するため、以下にプログラムを示す。

プログラム例６

```
public class Prog03_06
{
        public static void main(String[ ] args)
        {
                String s1 = "ABC";
                String s2 = "DEF";

                s2 = "GHI";
                System.out.println("文字列は" + s1 + s2 + "です");
        }
}
```

出力例

```
文字列は ABCGHI です
```

参照型変数のイメージを以下に示す。参照型変数そのものはどの領域を指し示しているかの情報を入れておく箱であり、実際のデータは別の領域にある。よって代入の場合は文字列が代入されるのではなく、指し示しているアドレスが変更される。

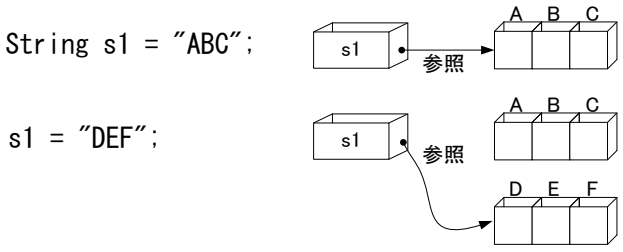

図２　参照型変数のイメージ

final 変数

プログラム上では一度値を決定したら、その後変更することがなく、定数のように扱いたい場合もある。その場合、まちがって変更してしまわないようにするため、final 変数を利用する。final 変数は以下のように宣言する。

書式

```
final データ型　変数名;
```

具体的には以下のようにして用いる。このプログラムは円の半径をキーボードから入力し、円周と面積を求めて表示するプログラムである。

プログラム例７

```
import java.util.Scanner;

public class Prog03_07
{
        public static void main(String[ ] args)
        {
                final double pai = 3.14159;
                Scanner stdIn = new Scanner(System.in);

                System.out.print("円の半径を入力してください:");
                double r = stdIn.nextDouble();

                System.out.println("円周は" + (2 * pai * r) + "です");
                System.out.println("面積は" + (pai * r * r) + "です");
        }
}
```

出力例(斜体はキーボードから入力した値)

円の半径を入力してください:*5*
円周は 31.4159 です
面積は 78.53975 です

final 変数では最初に値を代入する場合以外で代入を行うとコンパイルエラーとなる。

コンパイルエラーとなる例

```
final int x = 1;

x = 2;      // エラー
```

```
final int x;

x = 1;      // OK
x = 2;      // エラー
```

乱数の生成

キーボードから値を入力するのではなく、コンピュータに値を生成してもらうこともできる。以下のプログラムは0～5のランダムな値を生成し、1をプラスすることでサイコロの出目を生成する。

プログラム例８

```
import java.util.Random;

public class Prog03_08
{
        public static void main(String[ ] args)
        {
                Random rand = new Random();

                int x = rand.nextInt(6) + 1;

                System.out.println("サイコロの目は" + x + "です");
        }
}
```

出力例（実行するたびに数値の部分は変わる）

サイコロの目は3です

乱数の生成に関する部分は、現在は決まり文句として使用すればよいが、肝心の処理部分はrand.nextInt(n)である。この処理はnに値を設定することで、0以上n未満のランダムな整数値を生成する。このプログラム例ではrand.nextInt(6)としているので、0～5のいずれかの値を生成する。また、実数値を生成したい場合にはrand.nextDouble()を用いる。rand.nextDouble()は0.0以上1.0未満の範囲でdouble型の乱数を生成する。

演習

・テキストで示したプログラム例１～８を作成すること。

課題１

Kadai1_1

三つの整数型変数に値を代入し、合計と平均を求めるプログラムを作成しなさい。平均は実数で表示するものとし、小数点以下が切り捨てられていないことを確認すること。

Kadai1_2

三角形の底辺と高さを実数値としてキーボードから読み込んで、その面積を表示するプログラムを作成しなさい。

Kadai1_3

キーボードから読み込んだ整数値プラスマイナス5の範囲の整数値をランダムに生成して表示するプログラムを作成しなさい。（例：100が入力されたならば95～105の範囲で出力される）

Kadai1_4

-1.0以上1.0未満の範囲で実数値をランダムに生成して表示するプログラムを作成しなさい。

レポート提出要領

期限	次回授業日の8：50まで
用紙	A4
提出場所	竹下研究室の入口のポスト
表紙	表紙を１枚つけて、以下の項目を記述すること。 授業科目名 課題番号 クラス クラス番号 氏名 提出日
内容	ソースプログラム（プリントアウトのみ、手書きは不可）