Agility for Business
Technologic Arts

できるプログラマーを本気で育てる Java超入門 Webプログラマーへの第一歩

テクノロジックアート
長瀬　嘉秀

内容

Agility for Business
Technologic Arts

- 変数、数値型、文字列
- 演算子
- 配列、列挙型
- 制御文
- 繰り返し
- 簡単なサンプル開発

勉強会の参考書

Agility for Business
Technologic Arts

Java
(アジャイルソフトウェア開発技術シリーズ・基礎編)
【発売日】2012年5月10日
【著作】株式会社テクノロジックアート
【監修】長瀬 嘉秀
【編者】浜川 剛、山下 智也
【出版】東京電機大学出版局
【ISBN】978-4501550400

変数

プログラミングにおける変数は、そのプログラムで扱うデータを記憶しておく「入れ物」のことです。データの取り扱いが容易になるように、個別に「名札」を付けておく、と考えるとよいでしょう。

リテラル

Agility for Business
Technologic Arts

- 変数を定義するときに、型を決めておきます。

表 2.1　Java で使用できるリテラル

種類	説明	例	参照先
Integer Literals 整数リテラル	整数	1 123 -1 0xd（16 進数） 015（8 進数）	第 3 章
Floating-Point Literals 浮動小数点リテラル	小数点数	3.4 3.4f 3.4F 3.4d 3.4D 1E3（=1000.0）	第 3 章
Boolean Literals ブール型リテラル	真理値	true false	第 2 章
Charactor Literals 文字リテラル	「'」（シングルクォート）で囲んだ範囲	'a'	第 4 章
String Literals 文字列リテラル	「"」（ダブルクォート）で囲んだ範囲	"Java" "Ruby" "123"	第 4 章
Escape Sequences for Character and String Literals エスケープ文字リテラル	「¥」の次の「'」（シングルクォート）や「"」（ダブルクォート）を通常の文字として扱う	¥' ¥"	第 4 章
The Null Literal Null リテラル	null 参照を表わす	null	第 2 章

プリミティブ型

Agility for Business
Technologic Arts

表 3.1　プリミティブ型

型	記憶バイト数	範囲
byte	1	−128 ～ 127
short	2	−32768 ～ 32767
int	4	−2147483648 ～ 2147483647
long	8	−9223372036854775808 ～ 9223372036854775807
float	4	−3.40282347E38 ～ 1.40239846E-45, 1.40239846E-45 ～ 3.40282347E38
double	8	−1.79769313486232E308 ～ −94065645841247E-324, 4.94065645841247E-324 ～ 1.79769313486232E308

変数（コード１）

Agility for Business
Technologic Arts

```
public class Study1 {
public static void main(String[] args){

int currentLoginAttempt = 0;
double π = 3.14159;

int 現在のログイン回数 = 0;

double x=0.0,y=1.0,z=2.0;

System.out.println(currentLoginAttempt);
System.out.println(π);
System.out.println(現在のログイン回数);
System.out.println(x+y+z);

}
}
```

変数（コード２）

Agility for Business
Technologic Arts

```
int currentLoginAttempt = 1;


System.out.println("現在のログイン回数は
    "+currentLoginAttempt+"回です");
```

変数のスコープ

Agility for Business
Technologic Arts

表 2.2　その変数が定義されているクラス内以外からのアクセス可否

public	制限なし
修飾子を省略	同一パッケージのクラスからアクセスできる
protected	同一パッケージのクラスおよびサブクラスからアクセスできる
private	その変数が定義されているクラス内からしかアクセスできない
static	オブジェクトを生成しなくてもアクセスできる。ただし、private の場合はアクセスできない

予約語

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

数値型(Int, Float, Double)

Agility for Business
Technologic Arts

- 整数型
- 浮動小数点型

数値型

浮動小数点型(Float, Double)

単精度浮動小数点型 : Float

倍精度浮動小数点型 : Double

数値の計算

Agility for Business
Technologic Arts

表 3.3　算術演算子

演算子	意味
+	加算。左項と右項を足す。
-	減算。左項から右項を引く。
*	乗算。左項と右項を掛ける。
/	除算。左項を右項で割る。
%	剰余。左項を右項で割った余り。

```
int a = 100000; // 10万
int b = 10000; // 1 万
int result = a*b;

System.out.println(result);
```

数値の計算２

Agility for Business
Technologic Arts

```
int i=10;
int j=5;
int k=3;

int result = 0;
result=i+j;
System.out.println(result);
result=i-j;
System.out.println(result);
result=i*j;
System.out.println(result);
result=i/j;
System.out.println(result);
result=i%k;
System.out.println(result);
```

数値の比較

Agility for Business
Technologic Arts

表 3.4　比較演算子

演算子	説明
==	左項と右項が等しい場合は true を返す。それ以外の場合は false を返す。
!=	左項と右項が等しくない場合は true を返す。それ以外の場合は false を返す。
<	左項が右項よりも小さい場合は true を返す。それ以外の場合は false を返す。
<=	左項が右項以下の場合は true を返す。それ以外の場合は false を返す。
>	左項が右項よりも大きい場合は true を返す。それ以外の場合は false を返す。
>=	左項が右項以上の場合は true を返す。それ以外の場合は false を返す。

数値の比較（コード）

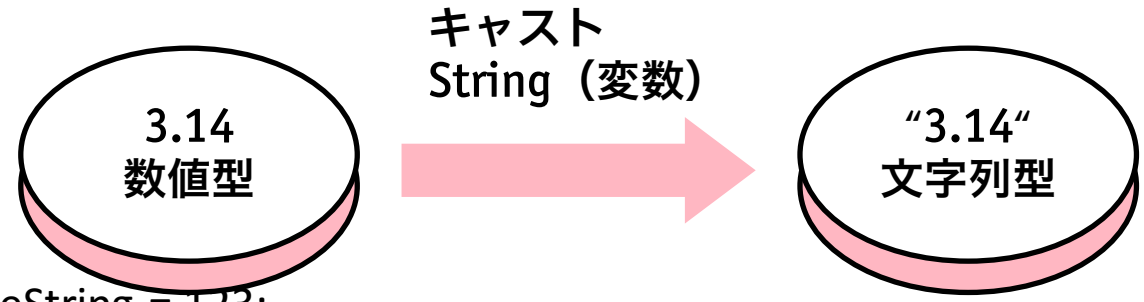
Agility for Business
Technologic Arts

```
int i=5;
int j=8;
int k=3;
int m=5;

System.out.println(i == j); // false
System.out.println(i != j); // true
System.out.println(i < k); // false
System.out.println(i <= m); // true
System.out.println(i > m); // false
System.out.println(i >= k); // true
```

数値の文字列変換、型変換

Agility for Business
Technologic Arts




```
int intBeforeString = 123;
String intToString = Integer.toString(intBeforeString);

System.out.println(intToString);

String intToString2 = String.valueOf(intBeforeString);

System.out.println(intToString2);

double doubleBeforeInt = 3.14;
int doubleToInt = (int)doubleBeforeInt;

System.out.println(doubleToInt);
```

文字列(String)

Agility for Business
Technologic Arts

```
String str = "ここまで";
char chr = 'こ';

System.out.println(str);
System.out.println(chr);
```

特殊文字

表 4.1　エスケープシーケンス一覧

エスケープシーケンス	意味
¥n	改行
¥t	タブ
¥¥	バックスラッシュ
¥'	シングルクォート
¥"	ダブルクォート
¥b	バックスペース
¥r	キャリッジリターン

空文字


```
String emptyString1 = new String();
String emptyString2 = "";

System.out.println(emptyString1);
System.out.println(emptyString2);
```

部分文字

Agility for Business
Technologic Arts

```
String sayJava = "hello";
System.out.println(sayJava.substring(0,1));

int countChar = sayJava.length();
System.out.println(countChar);

System.out.println(sayJava.substring(countCh
    ar-1,countChar));
```

文字の比較

Agility for Business
Technologic Arts

```
String hello = "hello";
String helloSame = "hello";
System.out.println( hello == helloSame );
```

文字列の連結

Agility for Business
Technologic Arts

```
String word = "";
String comma = ",";
String vegetable1 = "apple";
String vegetable2 = "berry";

word = vegetable1 + comma;
System.out.println(word.length());

word = word + vegetable2;
System.out.println(word.length());
```

文字列の分割

Agility for Business
Technologic Arts

```
String word = "orange,apple,grape";
String[] cells = word.split( ",");
System.out.println(cells[0]);
```

日付や時刻

Agility for Business
Technologic Arts

```
import java.util.Date;

public class Study13 {
public static void main(String[] args){

Date basetime = new Date();

System.out.println(basetime);

}
}
```

演算子

Agility for Business
Technologic Arts

表 2.3　演算子一覧

優先度	演算子	意味
高い	. [] ()	オブジェクトのメンバを呼び出す，配列要素へアクセスする，メソッドの引数を指定する
	++ -- + - ~ !	インクリメント，デクリメント，正符号，負符号，ビット反転，否定
	new (型)x	new 演算子，キャスト
	* / %	乗算，除算，剰余
	+ -	加算，減算
	<< >> >>>	ビットシフト
	< > <= >= instanceof	比較，instanceof 演算子
	== !=	同値，非同値
	&	ビット積
	^	排他的ビット和
	|	ビット和
	&&	論理積
	||	論理和
	?:	3 項演算子
低い	= *= /= %= += -= <<= >>= >>>= &= ^= |=	代入，複合代入演算子

演算子の優先度

Agility for Business
Technologic Arts

```
int result = 3 + 4 * 5;


System.out.println(result);
System.out.println(1>0&&1>0);
```

複合代入演算子


```
int point = 10;
point += 1;

System.out.println(point);
------------
int point = 10;
point=point+1;

System.out.println(point);
```

比較演算子

Agility for Business
Technologic Arts

```
String animal1 = "cat", animal2 = "cat", animal3 = "dog";

System.out.println(animal1 == animal2);
System.out.println(animal1 == animal3);
```

```
int num1=5,num2=20,num3=30;

System.out.println(num1 >= 5);
System.out.println(num2 > num3);
```

論理演算子

Agility for Business
Technologic Arts

```
int set_id = 1234;
String set_password  = "hirakegoma";

int id = 1234;
String password = "hirake";

if (id == set_id && password == set_password){
   System.out.println("ログインに成功しました。" );
}else{
   System.out.println("IDまたはパスワードが異なります。");
}
```

配列(Array)

Agility for Business
Technologic Arts


```
int score[] = {80, 70, 100};
int[] score2 = {10, 50, 30};

System.out.println(score[0]+" "+score[1]+"
    "+score[2]+" ");
```

列挙型(enum)

Agility for Business
Technologic Arts

```
public class Study22 {

enum Score {
    Low,
    Middle,
    Hight
}

public static void main(String[] args){

Score suzuki = Score.Hight; // 鈴木さんの数学のテスト結果

switch (suzuki) {
    case Low:
        System.out.println("normal");
        break;
    case Middle:
        System.out.println("good");
        break;
    case Hight:
        System.out.println("excellent");
        break;
    default:
        System.out.println("Go for it!");
        break;
}

}
}
```

制御文／条件分岐 (if, switch)

Agility for Business
Technologic Arts

if文

If (条件式){

条件式が真の場合に行う処理

}

```
int temperatureInRoom = 32;
if (temperatureInRoom >= 30) {
    System.out.println("とても暑いです、
                脱水症状には気を付けましょう。");
}
```

条件分岐（if else）

Agility for Business
Technologic Arts

if (条件式1){ 条件式が真の場合に行う処理

}elseif(条件式2){

条件式 1 が偽、かつ条件式 2 が真の場合に
　行う処理

}else{

条件式 1、2 が偽の場合に行う処理

}

条件分岐（if else　コード）

Agility for Business
Technologic Arts

```
int temperatureInRoom = 25;


if (temperatureInRoom >= 30) {
    System.out.println ( " とても暑いです、脱水症状には気を付けましょう。" );
} else if (temperatureInRoom <= 10) {
    System.out.println ( " 寒いです。風邪に気を付けましょう。" );
}else{
    System.out.println("過ごしやすい室温です。");
}
```

switch文

Agility for Business
Technologic Arts

リスト8-6

```
switch 考察する値 {
    case 値1 :
        値1の場合に行う処理
    case 値2,値3:
        値2,値3の場合に行う処理
    case 値4 where 条件:
        値4の場合で条件に当てはまるときに行う処理
    default:
        上記以外の場合に行う処理
}
```

switch文（コード）

Agility for Business
Technologic Arts

```
char JapaneseCharacter = 'さ';

switch (JapaneseCharacter) {
   case 'あ':
   case 'い':
   case 'う':
   case 'え':
   case 'お':
      System.out.println(JapaneseCharacter
                  +"はあ行です。");
      break;
   case 'か':
   case 'き':
   case 'く':
   case 'け':
   case 'こ':
      System.out.println(JapaneseCharacter+"はか行です。");
      break;
   case 'わ':
   case 'を':
      System.out.println(JapaneseCharacter+"はわ行です。");
      break;
   default:
      System.out.println(JapaneseCharacter
            +"はあ行でもか行でもわ行でも ありません。");
      break;
}
```

繰り返し(for, while)　for 文

Agility for Business
Technologic Arts

```
1  for 変数 : オブジェクト {
2
3      繰り返し実行する処理
4
5  }
```

```
int index = 1;

for (; index < 6;){
System.out.println(index+"回目です。");
  index++;
}
```

while文

Agility for Business
Technologic Arts

```
1  while 条件式 {
2
3      繰り返し実行する処理
4
5  }
```

```
int i=0;

while ( i < 5 ) {
    i+=1;
    System.out.println(i);
}
```

do-while文

Agility for Business
Technologic Arts

```
1  do      {
2
3     繰り返し実行する処理
4
5  } while 条件式
```

```
int i = 5;

do {
   i+=1;
   System.out.println("do "+i);
} while ( i < 5 );

i = 5;
while ( i < 5 ) {
   i+=1;
   System.out.println("while "+i);
}
```

簡単な サンプル開発

Agility for Business
Technologic Arts

偶数/奇数の 判定アプリケーション

```
import java.io.*;

public class Study29{
public static void main(String[] args){


int inputNumber;
System.out.print("数字を入力してください:");
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
try{
	String buf = br.readLine();
	inputNumber = Integer.parseInt(buf);
}catch(Exception e){
	inputNumber = 0;
}
```

奇数偶数の判定処理（続き　コード）

Agility for Business
Technologic Arts

```
if (inputNumber >0) {
      if (inputNumber % 2 == 0)
      {
        System.out.println(" 偶数 ");
      }
      else if (inputNumber % 2 == 1)
      {
        System.out.println(" 奇数 ");
      }
}
else {
      System.out.println(" エラー ");
}

}
}
```

演習問題

１０００までの間の素数を求めて、表示してください。

Agility for Business
Technologic Arts

できるプログラマーを本気で育てる Java超入門 Webプログラマーへの第一歩

テクノロジックアート
長瀬　嘉秀