Java ハンドブック

オブジェクト倶楽部 天野勝

m-amano@esm.co.jp

初版 2002 年 11 月 1 日
第 2 版 2002 年 11 月 21 日
第 3 版 2004 年 11 月 19 日

オリジナル：http://www.ObjectClub.jp/technicaldoc/java/javahandbook.doc

1　方針

この Java ハンドブックは、Java2 にもとづく Java プログラムを使ったセミナーなどにおいて、必要でなるであろう Java の言語概要、およびイディオムをまとめたものである。Java 言語を経験したことはあるが、ちょっとしたキーワードを忘れてしまった場合や、一度作ったプログラムだけどクラス名が思い出せないといった時に、本書をぱらぱらめくって忘れていた記憶が蘇れば幸いである。

本書は、大きく「言語要素」、「クラスのコンパイル&実行」、「イディオム集」の 3 つで構成されている。

このハンドブックで、Java 言語をマスターできるものではないので、Java 入門といった趣旨のセミナーのテキストとしては不向きであることをご了承願いたい。あくまでも、副読本として使っていただきたい。

2　謝辞

この Java ハンドブック作成にあたり、渡辺 義則さんにご意見を頂戴いたしました。ありがとうございます。

3　言語要素

(1)　キーワード

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

※"goto"はキーワードとして予約されているが、Java の言語要素としては存在しない。

(2)　プリミティブ型

boolean	リテラル:true,false
byte	-128～127
short	-32768～32767
int	-2147483648～2147483647 リテラル: 0、0372、0xDadaCafe、0x00FF00FF
long	-9223372036854775808～9223372036854775807 リテラル: 0l、0777L、0x100000000L、0xC0B0L
char	'¥u0000' to '¥uffff'、0～65536 リテラル: 'a'、'%'、'¥t'、'¥¥'、'¥"'、'¥u03a9'、'¥uFFFF'、'¥177'
float	仮数部 2^23(有効桁数 6～7)、指数部 2^8 リテラル:1e1f、2.f、.3f、0f、3.14f、6.022137e+23f
double	仮数部 2^52(有効桁数 15～16)、指数部 2^11 リテラル:1e1、2.、.3、0.0、3.14、1e-9d、1e137
null	リテラル:null

(3) エスケープシーケンス

¥b	¥u0008: backspace BS
¥t	¥u0009: horizontal tab HT
¥n	¥u000a: linefeed LF
¥f	¥u000c: form feed FF
¥r	¥u000d: carriage return CR
¥"	¥u0022: double quote "
¥'	¥u0027: single quote
¥¥	¥u005c: backslash ¥

(4) 演算子

算術演算子	+	加算	a=b+c;
	-	減算	a=b-c;
	*	乗算	a=b*c;
	/	除算	a=b/c;
	%	剰余	a=b%c;
論理演算子	&	AND	a=b&c;
	|	OR	a=b|c;
	^	XOR	a=b^c;
	!	否定	a=!c;
ビット操作演算子	<<	左シフト	a=b<<c;
	>>	右シフト	a=b>>c;
	>>>	右シフト 0 埋め	a=b>>>c;
	~	反転	a=~c;
関係演算子	<	左項<右項のとき true	(a<b)
	<=	左項≦右項のとき true	(a<=b)
	>	左項>右項のとき true	(a>b)
	>=	左項≧右項のとき true	(a>=b)
	==	左項と右項が等しいとき true	(a==b)
	!=	左項と右項が異なるとき true	(a!=b)
ショートサーキット演算子	&&	AND。左項が false ならば、右項の評価をしない	((a==b)&&(c==d))
	||	OR。左項が true ならば、右項の評価をしない	((a==b)||(c==d))

(5) 可視性

public	どのクラスからでもアクセス可能
protected	そのクラスを継承したクラス、および同一パッケージからのみアクセス可能
private	自分のクラスだけがアクセス可能
指定なし (package private)	同一パッケージ内のクラスからのみアクセス可能

(6) 配列

配列を宣言するには型名に「[]」をつける。宣言と同時に配列を初期化する場合は「int[][] a = new int[][]{{1, 2, 3}, {4, 5, 6, 7}};」と書きます。配列の添え字は 0 から始まる。配列の長さは、「length」で調べることができます。

```
class ArrayTest{
	public static void main(String[] args){
		int[][] a;
		a = new int[][]{{1, 2, 3}, {4, 5, 6, 7}};
		for(int i = 0; i < a.length; i++){
			for(int j = 0; j < a[i].length; j++){
				System.out.println(a[i][j]);
			}
		}
	}
}
```

(7) if

```
if(...){
}else if(...){
}else{
}
```

(8) switch

式およびラベルに指定できるのは、byte、char、short、int のみ。break を省略すると、次のラベルに処理が移ります。どのラベルにも一致しない場合に、default ラベルがあると、デフォルト処理がされ、default ラベルがない場合はそのまま switch ブロックを抜けます。

```
switch(式){
	case ラベル 1:
		処理 1
		// break 省略
	case ラベル 2:
		処理 2
		break;
	default:
		デフォルト処理
		break;
}
```

(9) while

式に指定できるのは、boolean のみ。式が true の間は、ブロック内の処理が繰り返し実行される。

```
while(式){
	処理
}
```

(10) do~while

式に指定できるのは、boolean のみ。式が評価される前に処理が必ず 1 度実行されます。式が true の間は、ブロック内の処理が繰り返し実行されます。

```
do{
	処理
}while(式)
```

(11) for

式に指定できるのは、boolean のみ。式が評価される前に処理が 1 度実行される。式が true の間は、ブロック内の処理が繰り返し実行されます。

```
for(初期設定; 条件式; 変化式){
        処理
}

例 1
for(int i = 0; i < 10; i++){
        sum += i;
}

例 2
for(int i = 0, j = 0; i < 10; i++, j+=2){
        sum += (i * j);
}
```

(12) try~catch~finally

```
try{
        例外が発生する処理
}catch(Exception e){
        例外発生時の処理
}

try{
        例外が発生する処理
}catch(Exception e){
        例外発生時の処理
}finally{
        try ブロックを抜けるときに必ず実行される処理
}
```

例外がスローされるメソッドを使う場合は、そこを try～catch ブロックで囲む。catch ではキャッチする例外クラス名を指定する。try～catch ブロックで囲まない場合は、呼び出し側のメソッドに throws 追加し、より上流側で処理させるようにする必要があります。

4　クラス、コンパイル&実行

(1).　Hello World

```
class HelloWorld{
	public static void main(String[] args){
		System.out.println("Hello World");
	}
}
```

実行開始ポイントは「publid static void main(String[] args)」。どのクラスにも main メソッドを書くことができます。クラス名と大文字小文字も同じファイル名にコードを書かなくてはなりません。クラス名が「HelloWorld」ならば、ソースファイル名は「HelloWorld.java」。

「c:¥tutrial」ディレクトリに「HelloWorld.java」ファイルを作成し、そこに上記のコードを記述したときは、DOS プロンプト上でコンパイル&実行するには以下のようにします。

```
C:¥tutorial>javac HelloWorld.java

C:¥tutorial>java HelloWorld
Hello World
```

(2).　パッケージ

```
package sample;

class HelloWorld{
	public static void main(String[] args){
		System.out.println("Hello World at sample");
	}
}
```

「c:¥tutrial」ディレクトリの下にパッケージ名と同じ「sample」ディレクトリを作成し、そこに「HelloWorld.java」ファイルを作成し、そこに上記のコードを記述したとします。DOS プロンプト上でコンパイルおよび実行するには以下のようにタイプします

```
C:¥tutorial>javac sample¥HelloWorld.java

C:¥tutorial>java sample.HelloWorld
Hello World at sample
```

(3). 継承

「class サブクラス名 extends スーパクラス名」で継承することができます。継承できるクラスは 1 つまで。多重継承はできません。

```
class Super{
	public void method1(){
		System.out.println("Super");
	}
}
```

```
class Sub extends Super{
	public void method1(){
		System.out.println("Sub");
	}
}
```

```
class ExtendsTest{
	public static void main(String[] args){
		Super s = new Super();
		s.method1();
		s = new Sub();
		s.method1();
	}
}
```

```
C:¥tutorial>javac Super.java Sub.java ExtendsTest.java

C:¥tutorial>java ExtendsTest
Super
Sub
```

(4). インターフェースの実装

インターフェースは、メソッドの本体を持たない。ファイル名はインターフェース名となる。例「ITest1.java」。

```
interface ITest1{
	public void method1();
}
```

「class クラス名 implements インターフェース名, ...」または「class クラス名 extends スーパクラス名 implements インターフェース名, ...」でクラスにインターフェースを実装できる。クラスではインターフェースにあるメソッドを必ず実装する必要があります。

```
class InterfaceTest implements ITest1 {
	public void method1(){
		System.out.println("method1");
	}

	public static void main(String[] args){
		ITest1 it1 = new InterfaceTest();
		it1.method1();
		System.out.print("it1 instanceof ITest1=");
		System.out.println(it1 instanceof ITest1);
		System.out.print("it1 instanceof InterfaceTest=");
		System.out.println(it1 instanceof InterfaceTest);

		InterfaceTest it = new InterfaceTest();
		it.method1();
	}
}
```

```
C:¥tutorial>javac ITest1.java ITest2.java InterfaceTest.java

C:¥tutorial>java InterfaceTest
method1
it1 instanceof ITest1=true
it1 instanceof InterfaceTest=true
method1
```

5 イディオム集

(1). 比較

プリミティブ型の場合

```
if(a == b){
	//同じ
}
```

オブジェクト型の場合

```
if(a.equals(b)){
	//同じ
}
```

(2). 整数のリテラル

```
class IntLiteral{
	public static void main(String[] args){
		System.out.println(123);   //10進数
		System.out.println(0123);  //8進数
		System.out.println(0x123); //16進数
	}
}
```

```
C:¥tutorial>java IntLiteral
123
83
291
```

(3). ファイルから１行ずつ読み込む

自分自身のソースコードをファイルから読み込んで出力します。

```
import java.io.BufferedReader;
import java.io.FileReader;

class FileRead{
	public static void main(String[] args){
		try{
			BufferedReader reader = new BufferedReader(
					new FileReader("FileRead.java"));
			String line;
			while((line = reader.readLine()) != null){
				System.out.println(line);
			}
			reader.close();
		}catch(Exception e){
			e.printStackTrace();
		}
	}
}
```

(4). コンソールから１行ずつ読み込む

実行後、何かキーを入力して Enter を押すと、入力した内容を出力します。終了するときは、Ctrl+C または、Ctrl+Z 後 Enter。

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

class StdinRead{
	public static void main(String[] args){
		try{
			BufferedReader reader = new BufferedReader(
					new InputStreamReader(System.in));
			String line;
			while((line = reader.readLine()) != null){
				System.out.println(line);
			}
			reader.close();
		}catch(Exception e){
			e.printStackTrace();
		}
	}
}
```

```
C:¥tutorial>java StdinRead
adfaf
adfaf
あいうえお
あいうえお
^Z
```

(5). 文字列を指定した文字で分割する

```
// JDK1.4 前
import java.util.StringTokenizer;

class Split{
	public static void main(String[] args){
		StringTokenizer st =
			new StringTokenizer("abc,def,ghi jkl,mno,", ",");
		while(st.hasMoreTokens()){
			System.out.println(st.nextToken());
		}
	}
}
```

```
C:¥tutorial>java Split
4
abc
def
ghi jkl
mno
```

```
// JDK1.4 以降
class Split14{
	public static void main(String[] args){
		String[] s = "abc,def,ghi jkl,mno,".split(",", -1);
		for(int i = 0; i < s.length; i++){
			System.out.println(s[i]);
		}
	}
}
```

```
C:¥tutorial>java Split14
5
abc
def
ghi jkl
mno
```

(6). 文字列を整数に変換する

```
class ToInt{
	public static void main(String[] args){
		int num = Integer.parseInt("123456");
		System.out.println(num);
	}
}
```

```
C:¥tutorial>java ToInt
123456
```

(7). 数値を文字列に変換する

```
class ToString{
	public static void main(String[] args){
		String num = String.valueOf(1234);
	}
}
```

(8). 配列のコピー

配列間のコピーには System.arraycopy()メソッドを使います。

```
class ArrayCopy{
	public static void main(String[] args){
		String[] src = new String[]{"ABC", "DEF", "GHI"};
		String[] dest = new String[src.length + 1];
		System.arraycopy(src, 0, dest, 0, src.length);
		dest[dest.length - 1] = "JKL";
		for(int i = 0; i < dest.length; i++){
			System.out.println(dest[i]);
		}
	}
}
```

```
C:¥tutorial>java ArrayCopy
ABC
DEF
GHI
JKL
```

(9). 動的配列

ArrayList クラスはオブジェクト型のインスタンスを格納することができます。取り出すときには適切な型にキャストする必要があります。

```
import java.util.ArrayList;

class DynamicArray{
	public static void main(String[] args){
		ArrayList al = new ArrayList();
		al.add("1");
		al.add(new Integer(2));
		al.add("3");
		al.add(new Double(2.2));
		for(int i = 0; i < al.size(); i++){
			if(al.get(i) instanceof String){
				String s = (String)al.get(i);
				System.out.println("String:" + s);
			}else if(al.get(i) instanceof Integer){
				int integer =
					((Integer)al.get(i)).intValue();
				System.out.println("int:" + integer);
			}else{
				System.out.println(al.get(i));
			}
		}
	}
}
```

```
C:¥tutorial>java DynamicArray
String:1
int:2
String:3
2.2
```

(10). コマンドライン引数

```
class CommandLine{
	public static void main(String[] args){
		System.out.println("引数の数=" + args.length);
		for(int i = 0; i < args.length; i++){
			System.out.println("引数" + i + "=" + args[i]);
		}
	}
}
```

```
C:¥tutorial>java CommandLine
引数の数=0

C:¥tutorial>java CommandLine a b c d
引数の数=4
引数 0=a
引数 1=b
引数 2=c
引数 3=d
```

6　参考 URL

- 「The Java Language Specification Second Edition」
  http://java.sun.com/docs/books/jls/second_edition/html/j.title.doc.html
- 「Java™ 2 Platform, Standard Edition, 1.4.0 API 仕様」
  http://java.sun.com/j2se/1.4/ja/docs/ja/api/index.html